# NebulaGraph Database Manual

v3.4.1

# Table of contents

1.	We	elcome to NebulaGraph 3.4.1 Documentation	7
	1.1	Getting started	7
	1.2	Other Sources	7
	1.3	Symbols used in this manual	7
	1.4	Modify errors	8
2.	Int	roduction	9
	2.1	An introduction to graphs	9
:	2.2	Market overview of graph databases	23
	2.3	Related technologies	37
	2.4	What is NebulaGraph	51
	2.5	Data modeling	55
:	2.6	Path types	57
	2.7	VID	59
	2.8	NebulaGraph architecture	61
3.	Qu	ick start	79
	3.1	Getting started with NebulaGraph	79
	3.2	Step 1: Install NebulaGraph	83
	3.3	Step 2: Manage NebulaGraph Service	87
	3.4	Step 3: Connect to NebulaGraph	91
	3.5	Register the Storage Service	93
	3.6	Step 4: Use nGQL (CRUD)	94
	3.7	nGQL cheatsheet	104
4.	nG	QL guide	126
,	4.1	nGQL overview	126
	4.2	Data types	142
	4.3	Variables and composite queries	161
	4.4	Operators	166
	4.5	Functions and expressions	180
	4.6	General queries statements	224
	4.7	Clauses and options	265
	4.8	Space statements	295
	4.9	Tag statements	305
	4.10	Edge type statements	314
	4.11	Vertex statements	320
	4.12	Edge statements	328

4.13 Native index statements	335
4.14 Full-text index statements	347
4.15 Subgraph and path	356
4.16 Query tuning and terminating statements	363
4.17 Job manager and the JOB statements	369
5. Deployment and installation	373
5.1 Prepare resources for compiling, installing, and running NebulaGraph	373
5.2 Compile and install Nebula Graph	379
5.3 Standalone NebulaGraph	401
5.4 Deploy a license for NebulaGraph Enterprise Edition	403
5.5 Manage NebulaGraph Service	406
5.6 Connect to NebulaGraph	410
5.7 Manage Storage hosts	412
5.8 Upgrade	413
5.9 Uninstall NebulaGraph	420
6. Configurations and logs	422
6.1 Configurations	422
6.2 Log management	451
7. Monitor and metrics	456
7.1 Query NebulaGraph metrics	456
7.2 RocksDB statistics	465
7.3 Black-box monitoring	467
8. Data security	473
8.1 Authentication and authorization	473
8.2 SSL encryption	485
9. Backup & Restore	487
9.1 NebulaGraph BR (Community Edition)	487
9.2 NebulaGraph BR (Enterprise Edition)	499
9.3 Backup and restore data with snapshots	511
10. Synchronization & Migration	514
10.1 BALANCE syntax	514
10.2 Synchronize between two clusters	515
11. Practices	527
11.1 Compaction	527
11.2 Storage load balance	529
11.3 Graph data modeling suggestions	532
11.4 System design suggestions	536
11.5 Execution plan	537

11.6	Processing super vertices	538
11.7	Enable AutoFDO for NebulaGraph	540
11.8	Best practices	546
12. Cl	ient	547
12.1	Clients overview	547
12.2	NebulaGraph Console	548
12.3	NebulaGraph CPP	552
12.4	NebulaGraph Java	554
12.5	NebulaGraph Python	556
12.6	NebulaGraph Go	558
13. Ne	ebulaGraph Studio	560
13.1	About NebulaGraph Studio	560
13.2	Deploy and connect	563
13.3	Quick start	576
13.4	Troubleshooting	605
14. Ne	ebulaGraph Dashboard Community Edition	609
14.1	What is NebulaGraph Dashboard Community Edition	609
14.2	Deploy Dashboard Community Edition	611
14.3	Connect Dashboard	614
14.4	Dashboard	615
14.5	Metrics	622
15. Ne	ebulaGraph Dashboard Enterprise Edition	631
15.1	What is NebulaGraph Dashboard Enterprise Edition	631
15.2	Deploy Dashboard Enterprise Edition	633
15.3	Connect Dashboard	639
15.4	NebulaGraph Dashboard Enterprise Edition license	640
15.5	Create and import clusters	642
15.6	Cluster management	652
15.7	Authority management	690
15.8	Task Center	693
15.9	System settings	694
15.10	) Metrics	700
15.11	I FAQ	709
16. Ne	ebulaGraph Explorer	711
16.1	What is NebulaGraph Explorer	711
16.2	Deploy and connect	713
16.3	Page overview	725
16.4	Database management	728

16.5	Graph explorer	742
16.6	Visual Query	755
16.7	Canvas	760
16.8	Workflow	769
16.9	Inline frame	798
16.10	Basic operations and shortcuts	800
16.11	FAQ	801
17. Ne	ebulaGraph Importer	803
17.1	NebulaGraph Importer	803
17.2	Configuration with Header	810
17.3	Configuration without Header	813
18. Ne	ebulaGraph Exchange	816
18.1	Introduction	816
18.2	Get Exchange	822
18.3	Exchange configurations	824
18.4	Use NebulaGraph Exchange	838
18.5	Exchange FAQ	924
19. Ne	ebulaGraph Operator	927
19.1	What is NebulaGraph Operator	927
19.2	Overview of using NebulaGraph Operator	930
19.3	Deploy NebulaGraph Operator	931
19.4	Deploy clusters	936
19.5	Connect to NebulaGraph databases with Nebular Operator	950
19.6	Configure clusters	955
19.7	Upgrade NebulaGraph clusters created with NebulaGraph Operator	966
19.8	NebulaGraph cluster rolling update strategy	969
19.9	Backup and restore data using NebulaGraph Operator	970
19.10	Self-healing	974
19.11	FAQ	975
20. Gr	aph computing	976
20.1	Algorithm overview	976
20.2	NebulaGraph Algorithm	992
20.3	NebulaGraph Analytics	998
20.4	NebulaGraph Analytics license	1005
20.5	NebulaGraph Explorer Workflow	1007
21. Ne	ebulaGraph Spark Connector	1008
21.1	Version compatibility	1008
21.2	Use cases	1009

21.3	Benefits	1009
21.4	Release note	1009
21.5	Get NebulaGraph Spark Connector	1009
21.6	How to use	1010
22. No	ebulaGraph Flink Connector	1015
22.1	Use cases	1015
22.2	Release note	1015
23. No	ebulaGraph Bench	1016
23.1	Scenario	1016
23.2	Release note	1016
23.3	Test process	1016
24. Ap	ppendix	1017
24.1	Release Note	1017
24.2	NebulaGraph learning path	1025
24.3	About NebulaGraph licenses	1031
24.4	FAQ	1033
24.5	Ecosystem tools overview	1046
24.6	Import tools	1051
24.7	How to Contribute	1052
24.8	History timeline for NebulaGraph	1056
24.9	Error code	1062

# 1. Welcome to NebulaGraph 3.4.1 Documentation



This manual is revised on 2024-2-19, with GitHub commit f3601a98.

# 1.1 Getting started

- Learning path & Get NebulaGraph Certifications
- What is Nebula Graph
- Quick start
- Preparations before deployment
- nGQL cheatsheet
- FAO
- Ecosystem Tools

# 1.2 Other Sources

- To cite NebulaGraph
- NebulaGraph Homepage
- Forum
- Blogs
- Videos
- Chinese Docs

# 1.3 Symbols used in this manual



Additional information or operation-related notes.



Cautions that need strict observation. If not, systematic breakdown, data loss, and security issues may happen.



Operations that may cause danger. If not observed, systematic breakdown, data loss, and security issues will happen.



Operations that merit attention as for performance enhancement.

- 7/1069 - 2023 Vesoft Inc.



Frequently asked questions.



 $The \ compatibility \ notes \ between \ nGQL \ and \ open Cypher, \ or \ between \ the \ current \ version \ of \ nGQL \ and \ its \ prior \ ones.$ 



Differences between the NebulaGraph Community and Enterprise editions.

# 1.4 Modify errors

This NebulaGraph manual is written in the Markdown language. Users can click the pencil sign on the upper right side of each document title and modify errors.

Last update: February 19, 2024

- 8/1069 - 2023 Vesoft Inc.

# 2. Introduction

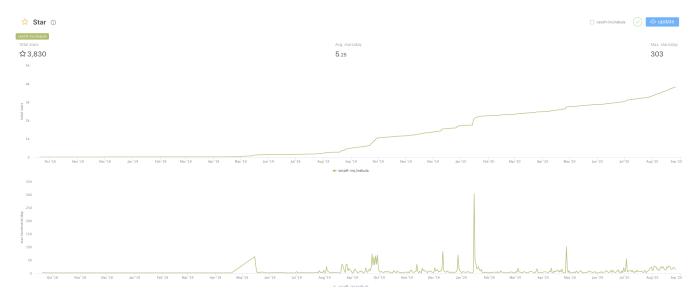
# 2.1 An introduction to graphs

People from tech giants (such as Amazon and Facebook) to small research teams are devoting significant resources to exploring the potential of graph databases to solve data relationships problems. What exactly is a graph database? What can it do? Where does it fit in the database landscape? To answer these questions, we first need to understand graphs.

Graphs are one of the main areas of research in computer science. Graphs can efficiently solve many of the problems that exist today. This topic will start with graphs and explain the advantages of graph databases and their great potential in modern application development, and then describe the differences between distributed graph databases and several other types of databases.

#### 2.1.1 What are graphs?

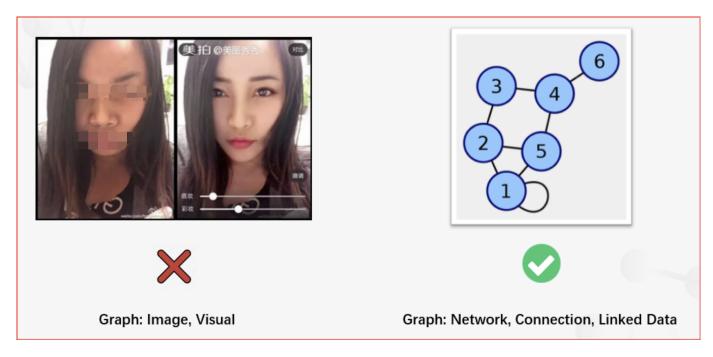
Graphs are everywhere. When hearing the word graph, many people think of bar charts or line charts, because sometimes we call them graphs, which show the connections between two or more data systems. The simplest example is the following picture, which shows the number of NebulaGraph GitHub repository stars over time.



This type of diagram is often called a line chart. As you can see, the number of starts rises over time. A line chart can show data changes over time (depending on the scale settings). Here we have given only examples of line charts. There are various graphs, such as pie charts, bar charts, etc.

Another kind of diagram is often used in daily conversation, such as image recognition, retouched photos. This type of diagram is called a picture/photo/image.

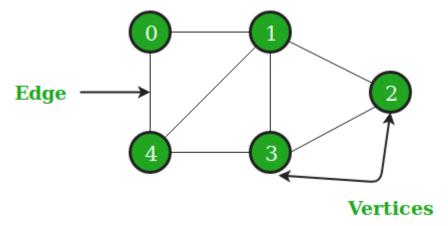
- 9/1069 - 2023 Vesoft Inc.



The diagram we discuss in this topic is a different concept, the graph in graph theory.

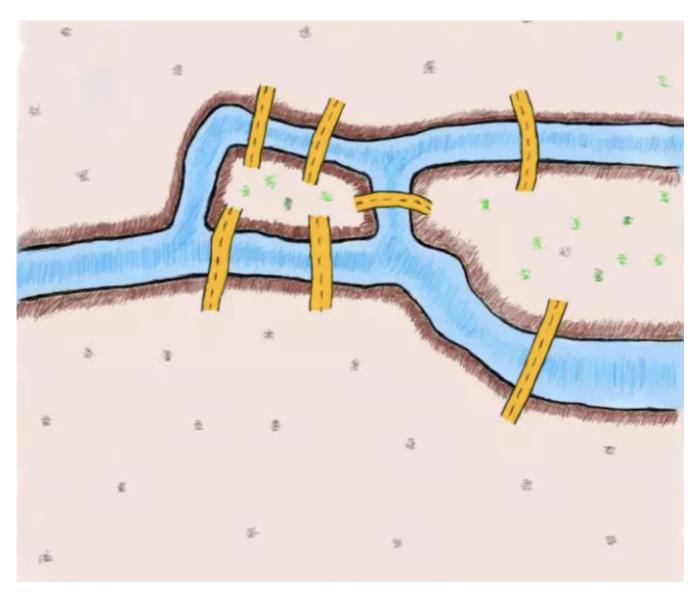
In graph theory, a branch of mathematics, graphs are used to represent the relationships between entities. A graph consists of several small dots (called vertices or nodes) and lines or curves (called edges) that connect these dots. The term graph was proposed by Sylvester in 1878.

The following picture is what this topic calls a graph.

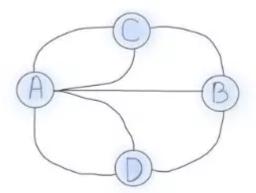


Simply put, graph theory is the study of graphs. Graph theory began in the early 18th century with the problem of the Seven Bridges of Königsberg. Königsberg was then a Prussian city (now part of Russia, renamed Kaliningrad). The river Preger crossed Königsberg and not only divided Königsberg into two parts, but also formed two small islands in the middle of the river. This divided the city into four areas, each connected by seven bridges. There was a game associated with Königsberg at the time, namely how to cross each bridge only once and navigate the entire four areas of the city. A simplified view of the seven bridges is shown below. Try to find the answer to this game if you are interested <sup>1</sup>.

- 10/1069 - 2023 Vesoft Inc.



To solve this problem, the great mathematician Euler proved that the problem was unsolvable by abstracting the four regions of the city into points and the seven bridges connecting the city into edges connecting the points. The simplified abstract diagram is as follows  $^{2}$ .



The four dots in the picture represent the four regions of Königsberg, and the lines between the dots represent the seven bridges connecting the four regions. It is easy to see that the area connected by the even-numbered bridges can be easily passed because different routes can be chosen to come and go. The areas connected by the odd-numbered bridges can only be used as starting or endings points because the same route can only be taken once. The number of edges associated with a node is called the node degree. Now it can be shown that the Königsberg problem can only be solved if two nodes have odd degrees and the other nodes

have even degrees, i.e., two regions must have an even number of bridges and the remaining regions have an odd number of bridges. However, as we know from the above picture, there is no even number of bridges in any region of Königsberg, so this puzzle is unsolvable.

# 2.1.2 Property graphs

From a mathematical point of view, graph theory studies the relationships between modeled objects. However, it is common to extend the underlying graph model. The extended graphs are called the **attribute graph model**. A property graph usually consists of the following components.

- Node, an object or entity. In this topic, nodes are called vertices.
- Relationship between nodes. In this topic, relationships are called edges. Usually, the edges can be directed or undirected to indicate a relationship between two entities.
- There can be properties on nodes and edges.

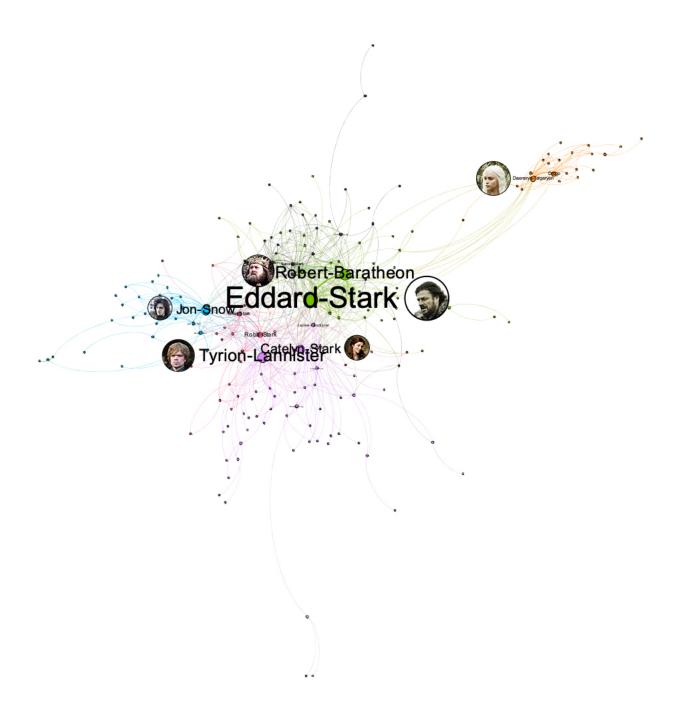
In real life, there are many examples of property graphs.

For example, Qichacha or BOSS Zhipin use graphs to model business equity relationships. A vertex usually represents a natural person or a business, and the edge represents the equity relationship between a person and a business. The properties on vertices can be the name, age, ID number, etc. of the natural person. The properties on edges can be the investment amount, investment time, position such as director and supervisor.

A vertex can be a listed company and an edge can be a correlation between listed companies. The vertex property can be a stock code, abbreviation, market capitalization, sector, etc. The edge property can be the time-series correlation coefficient of the stock price  $\frac{3}{2}$ .

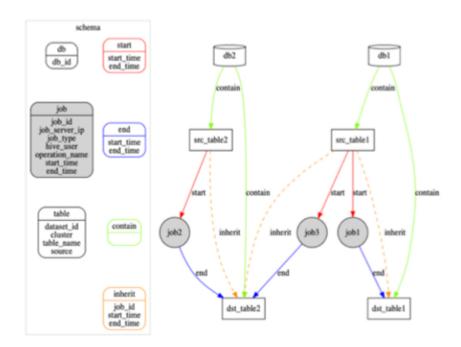
The graph relationship can also be similar to the character relationship in a TV series like Game of Thrones <sup>4</sup>. Vertices stand for the characters. Edges represent the interactions between the characters. Vertex properties are the character's names, ages, camps, etc., and edge properties are the number of interactions between two characters.

- 12/1069 - 2023 Vesoft Inc.



Graphs are also used for governance within IT systems. For example, a company like WeBank has a very large data warehouse and corresponding data warehouse management tools. These management tools record the ETL relationships between the Hive tables in the data warehouse through Job implementation <sup>5</sup>. Such ETL relationships can be very easily presented and managed in the form of graphs, and the root cause can be easily traced when problems arise.

- 13/1069 - 2023 Vesoft Inc.



Graphs can also be used to document the invocation relationships between the intricate microservices within a large IT system <sup>6</sup>, which is used by operations teams for service governance. Here each point represents a microservice and the edge represents the invocation relationship between two microservices; thus, Ops can easily find invocation links with availability below a threshold (99.99%) or discover microservice nodes that would be particularly affected by a failure.

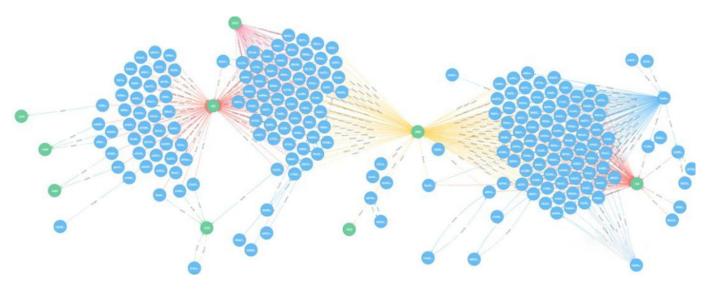
Graphs are also used to record the invocation relationships between the intricate microservices <sup>6</sup>. Each vertex represents a microservice and an edge represents the invocation relationship between two microservices. This allows Ops to easily find call links with availability below a threshold (99.99%), or to discover microservice nodes where a failure would have a particularly large impact.

Graphs can also be used to improve the efficiency of code development. Graphs store function call relationships between codes <sup>6</sup> to improve the efficiency of reviewing and testing the code. In such a graph, each vertex is a function or variable, each edge is a call relationship between functions or variables. When there is a new code commit, one can more easily see other interfaces that may be affected, which helps testers better assess potential go-live risks.

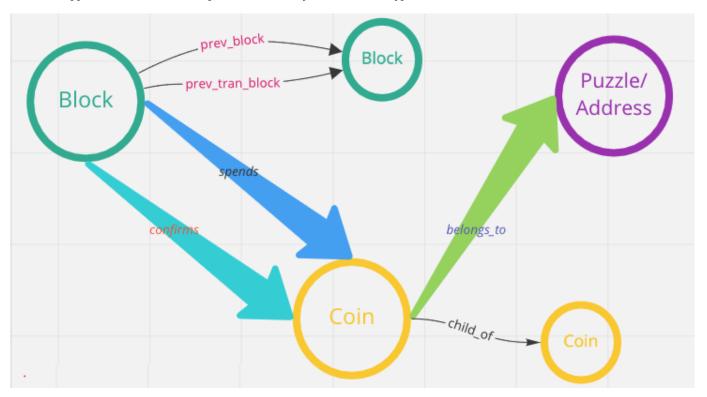
In addition, we can discover more scenarios by adding some temporal information as opposed to a static property graph that does not change.

For example, inside a network of interbank account fund flows <sup>7</sup>, a vertex is an account, an edge is the transfer record between accounts. Edge properties record the time, amount, etc. of the transfer. Companies can use graph technology to easily explore the graph to discover obvious misappropriation of funds, paying back a load to with the loan, loan gang scams, and other phenomena.

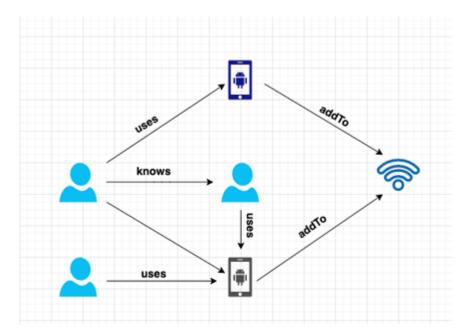
- 14/1069 - 2023 Vesoft Inc.



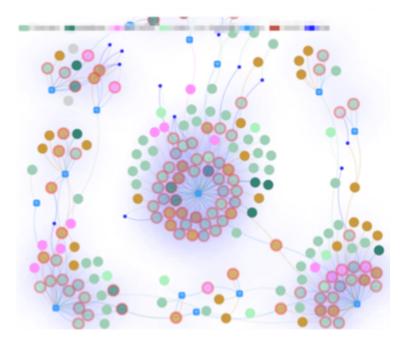
The same approach can be used to explore the discovery of the flow of cryptocurrencies.



In a network of accounts and devices <sup>8</sup>, vertices can be accounts, mobile devices, and WIFI networks, edges are the login relationships between these accounts and mobile devices, and the access relationships between mobile devices and WIFI networks.



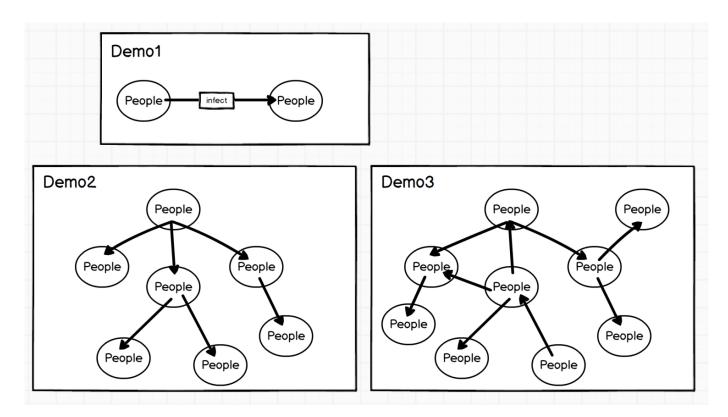
These graph data records the characteristic of the network black production operations. Some big companies such as 360 DigiTech<sup>8</sup>, Kuaishou<sup>9</sup>, WeChat<sup>10</sup>, Zhihu<sup>11</sup>, and Ctrip Finance all identified over a million crime groups through technology.



In addition to the dimension of time, you can find more scenarios for property graphs by adding some geographic location information.

For an example of tracing the source of the Coronavirus Disease (COVID-19) <sup>12</sup>, vertices are the person and edges are the contact between people. Vertex properties are the information of the person's ID card and onset time, and edge properties are the time and geographical location of the close contact between people, etc. It provides help for health prevention departments to quickly identify high-risk people and their behavioral trajectories.

- 16/1069 - 2023 Vesoft Inc.



The combination of geographic location and graph is also used in some O2O scenarios, such as real-time food recommendation based on POI (Point-of-Interest) <sup>13</sup>, which enables local life service platform companies like Meituan to recommend more suitable businesses in real-time when consumers open the APP.

A graph is also used for knowledge inference. Huawei, Vivo, OPPO, WeChat, Meituan, and other companies use graphs for the representation of the underlying knowledge relationships.

- 17/1069 - 2023 Vesoft Inc.

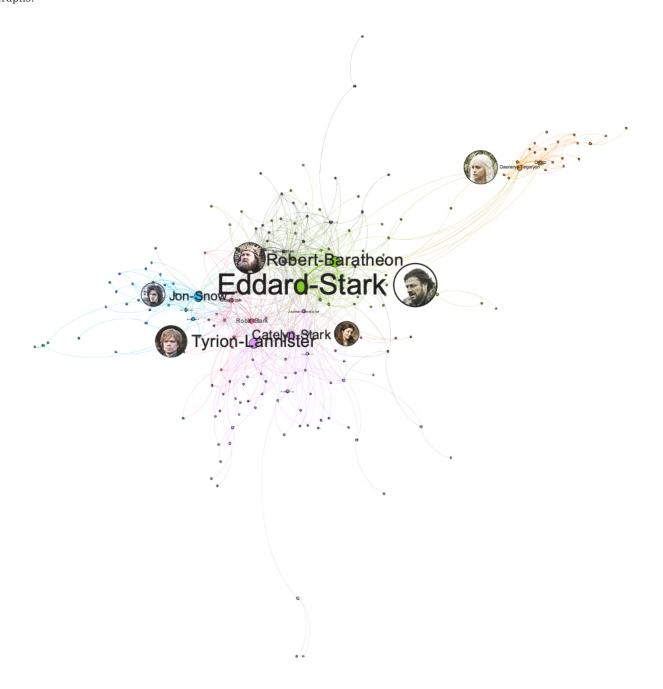
# 2.1.3 Why do we use graph databases?

Although relational databases and semi-structured databases such as XML/JSON can be used to describe a graph-structured data model, a graph (database) not only describes the graph structure and stores data itself but also focuses on handling the associative relationships between the data. Specifically, graph databases have several advantages:

• Graphs are a more visual and intuitive way of representing knowledge to human brains. This allows us to focus on the business problem itself rather than how to describe the problem as a particular structure of the database (e.g., a table structure).

- 18/1069 - 2023 Vesoft Inc.

• It is easier to show the characteristic of the data in graphs. Such as transfer paths and nearby communities. To analyze the relationships of characters and character importance in Game of Thrones, data displayed with tables is not as intuitive as with graphs.



Especially when some central vertices are deleted:

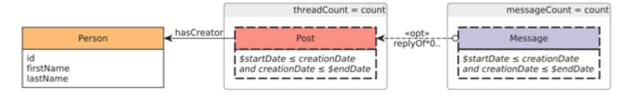


Adding an edge can completely change the entire topology.



We can intuitively sense the importance of minor changes in graphs rather than in tables.

• Graph query language is designed based on graph structures. The following is a query example in LDBC. Requirements: Query the posts posted by a person, and query the corresponding replies (the replies themselves will also be replied multiple times). Since the posting time and reply time both meet certain conditions, you can sort the results according to the number of replies.



Write querying statements using PostgreSQL:

```
-PostgreSQL
WITH RECURSIVE post_all(psa_threadid
                         , psa_thread_creatorid, psa_messageid
                          , psa_creationdate, psa_messagetype
                           AS (
    SELECT m_messageid AS psa_threadid
          , m_creatorid AS psa_thread_creatorid
           , m_messageid AS psa_messageid
           , m_creationdate, 'Post'
      FROM message
      WHERE 1=1 AND m c replyof IS NULL -- post, not comment
        AND m_creationdate BETWEEN :startDate AND :endDate
  UNION ALL
    SELECT psa.psa_threadid AS psa_threadid
          , psa.psa_thread_creatorid AS psa_thread_creatorid
            m messageid, m creationdate, 'Comment
       FROM message p, post_all psa
     WHERE 1=1 AND p.m_c_replyof = psa.psa_messageid
AND m_creationdate BETWEEN :startDate AND :endDate
SELECT p.p_personid AS "person.id"
, p.p_firstname AS "person.firstName"
      , p.p_lastname AS "person.lastNam
        count(DISTINCT psa.psa_threadid) AS threadCount
END) AS messageCount
        count(DISTINCT psa.psa_messageid) AS messageCount
  FROM person p left join post_all psa on
   1=1 AND p.p_personid = psa.psa_thread_creatorid
AND psa_creationdate BETWEEN :startDate AND :endDate
 GROUP BY p.p_personid, p.p_firstname, p.p_lastname
 ORDER BY messageCount DESC, p.p_personid
```

Write querying statements using Cypher designed especially for graphs:

```
--Cypher
MATCH (person:Person)<-[:HAS_CREATOR]-(post:Post)<-[:REPLY_OF*O..]-(reply:Message)
WHERE post.creationDate >= $startDate AND post.creationDate <= $endDate
AND reply.creationDate >= $startDate AND reply.creationDate <= $endDate
RETURN
person.id, person.firstName, person.lastName, count(DISTINCT post) AS threadCount,
count(DISTINCT reply) AS messageCount
ORDER BY
messageCount DESC, person.id ASC
LIMIT 100
```

- Graph traversal (corresponding to Join in SQL) is much more efficient because the storage and query engines are designed specifically for the structure of the graph.
- Graph databases have a wide range of application scenarios. Examples include data integration (knowledge graph), personalized recommendations, fraud, and threat detection, risk analysis, and compliance, identity (and control) verification, IT infrastructure management, supply chain, and logistics, social network research, etc.
- According to the literature <sup>14</sup>, the fields that use graph technology are (from the greatest to least): information technology (IT), research in academia, finance, laboratories in industry, government, healthcare, defense, pharmaceuticals, retail, and e-commerce, transportation, telecommunications, and insurance.
- In 2019, according to Gartner's questionnaire research, 27% of customers (500 groups) are using graph databases and 20% have plans to use them.

## 2.1.4 RDF

This topic does not discuss the RDF data model due to space limitations.

- 1. Souce of the picture: https://medium.freecodecamp.org/i-dont-understand-graph-theory-1c96572a1401. ←
- 2. Source of the picture: https://medium.freecodecamp.org/i-dont-understand-graph-theory-1c96572a1401  $\begin{tabular}{l} \leftarrow \end{tabular}$
- 3. https://nebula-graph.com.cn/posts/stock-interrelation-analysis-jgrapht-nebula-graph/ ←
- $4\cdot$  https://nebula-graph.com.cn/posts/game-of-thrones-relationship-networkx-gephi-nebula-graph/
- 5. https://nebula-graph.com.cn/posts/practicing-nebula-graph-webank/  $\leftarrow$
- $6.\ https://nebula-graph.com.cn/posts/meituan-graph-database-platform-practice/ \\ \longleftarrow \leftarrow$
- 7. https://zhuanlan.zhihu.com/p/90635957  $\boldsymbol{\leftarrow}$
- 8. https://nebula-graph.com.cn/posts/graph-database-data-connections-insight/  $\leftarrow \leftarrow$
- 9. https://nebula-graph.com.cn/posts/kuaishou-security-intelligence-platform-with-nebula-graph/  $\hookleftarrow$
- 10. https://nebula-graph.com.cn/posts/nebula-graph-for-social-networking/ ←
- 11. https://mp.weixin.qq.com/s/K2QinpR5Rplw1teHpHtf4w ←
- 12. https://nebula-graph.com.cn/posts/detect-corona-virus-spreading-with-graph-database/  $\boldsymbol{\leftarrow}$
- 13. https://nebula-graph.com.cn/posts/meituan-graph-database-platform-practice/  $\boldsymbol{\hookleftarrow}$
- 14. https://arxiv.org/abs/1709.03188 ←

Last update: February 19, 2024

- 22/1069 - 2023 Vesoft Inc.

# 2.2 Market overview of graph databases

Now that we have discussed what a graph is, let's move on to further understanding graph databases developed based on graph theory and the property graph model.

Different graph databases may differ slightly in terms of terminology, but in the end, they all talk about vertices, edges, and properties. As for more advanced features such as labels, indexes, constraints, TTL, long tasks, stored procedures, and UDFs, these advanced features will vary significantly from one graph database to another.

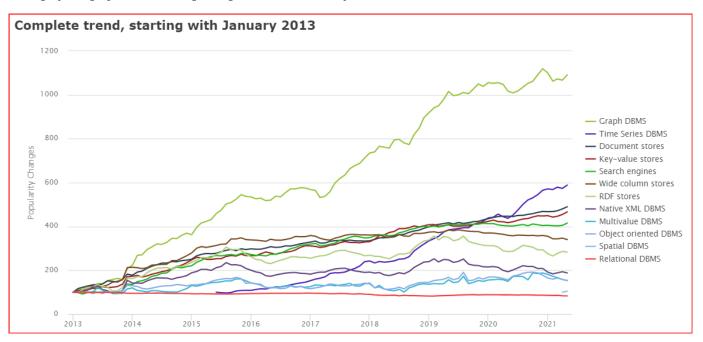
Graph databases use graphs to store data, and the graph structure is one of the structures that are closest to high flexibility and high performance. A graph database is a storage engine specifically designed to store and retrieve large information, which efficiently stores data as vertices and edges and allows high-performance retrieval and querying of these vertex-edge structures. We can also add properties to these vertices and edges.

# 2.2.1 Third-party services market predictions

#### **DB-Engines ranking**

According to DB-Engines.com, the world's leading database ranking site, graph databases have been the fastest growing database category since 2013 <sup>1</sup>.

The site counts trends in the popularity of each category based on several metrics, including records and trends based on search engines such as Google, technical topics discussed on major IT technology forums and social networking sites, job posting changes on job boards. 371 database products are included in the site and are divided into 12 categories. Of these 12 categories, a category like graph databases is growing much faster than any of the others.

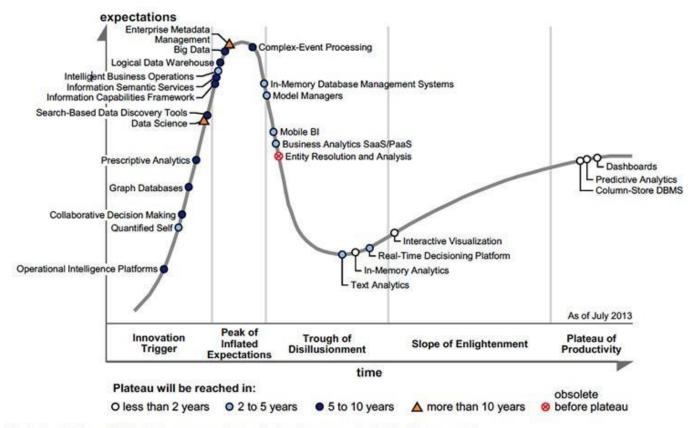


### **Gartner's predictions**

Gartner, one of the world's top think tanks, identified graph databases as a major business intelligence and analytics technology trend long before 2013 <sup>2</sup>. At that time, big data was hot as ever, and data scientists were in a hot position.

- 23/1069 - 2023 Vesoft Inc.

Figure 1. Hype Cycle for Business Intelligence and Analytics, 2013



BI = business intelligence; DBMS = database management system; SaaS = software as a service; PaaS = platform as a service

Until recently, graph databases and related graph technologies were ranked in the Top 10 Data and Analytics Trends for 2021 <sup>3</sup>.

- 24/1069 - 2023 Vesoft Inc.

# Gartner Top 10 Data and Analytics Trends, 2021



# Accelerating Change

- Smarter, Responsible, Scalable AI
- Composable Data and Analytics
- 3 Data Fabric Is the Foundation
- From Big to Small and Wide Data



# Operationalizing Business Value

- KOps
- 6 Engineering Decision Intelligence
- 7 D&A as a Core Business Function



# Distributed Everything

- Graph Relates Everything
- The Rise of the Augmented Consumer
- 10 D&A at the Edge

# gartner.com/SmarterWithGartner

Source: Gartner © 2021 Gartner, Inc. All rights reserved. CTMKT\_1164473 **Gartner** 

2023 Vesoft Inc.

# Trend 8: Graph Relates Everything

Graphs form the foundation of many modern data and analytics capabilities to find relationships between people, places, things, events, and locations across diverse data assets. D&A leaders rely on graphs to quickly answer complex business questions which require contextual awareness and an understanding of the nature of connections and strengths across multiple entities.

Gartner predicts that by 2025, graph technologies will be used in 80% of data and analytics innovations, up from 10% in 2021, facilitating rapid decision-making across the organization.

It can be noted that Gartner's predictions match the DB-Engines ranking well. There is usually a period of rapid bubble development, then a plateau period, followed by a new bubble period due to the emergence of new technologies, and then a plateau period again.

#### Market size of graph databases

According to statistics and forecasts from Verifiedmarketresearc<sup>4</sup>, fnfresearch<sup>5</sup>, MarketsandMarkets<sup>6</sup>, and Gartner<sup>7</sup>, the global graph database market size is about to grow from about USD 0.8 billion in 2019 to USD 3-4 billion by 2026, at a Compound Annual Growth Rate (CAGR) of about 25%, which corresponds to about 5%-10% market share of the global database market.



# 2.2.2 Market participants

## Neo4j, the pioneer of (first generation) graph databases

Although some graph-like data models and products, and the corresponding graph language G/G+ had been proposed in the 1970s (e.g. CODASYL <sup>8</sup>). But it is Neo4j, the main pioneer in this market, that has really made the concept of graph databases popular, and even the two main terms (labeled) property graphs and graph databases were first introduced and practiced by Neo4j.

!!! Info "This section on the history of Neo4j and the graph query language it created, Cypher, is largely excerpted from the ISO WG3 paper *An overview of the recent history of Graph Query Languages* <sup>10</sup> and <sup>9</sup>. To take into account the latest two years of development, the content mentioned in this topic has been abridged and updated by the authors of this book.

- 26/1069 - 2023 Vesoft Inc.

# About GQL (Graph Query Language) and the development of an International Standard

Readers familiar with databases are probably aware of the Structured Query Language SQL. by using SQL, people access databases in a way that is close to natural language. Before SQL was widely adopted and standardized, the market for relational databases was very fragmented. Each vendor's product had a completely different way of accessing. Developers of the database product itself, developers of the tools surrounding the database product, and end-users of the database, all had to learn each product. When the SQL-89 standard was developed in 1989, the entire relational database market quickly focus on SQL-89. This greatly reduced the learning costs for the people mentioned above.

GQL (Graph Query Language) assumes a role similar to SQL in the field of graph databases. Uses interacts with graphs with GQL. Unlike international standards such as SQL-89, there are no international standards for GQL. Two mainstream graph languages are Neo4j's Cypher and Apache TinkerPop's Gremlin. The former is often referred to as the DQL, Declarative Query Language. DQL tells the system "what to do", regardless of "how to do". The latter is referred to as the IQL, Imperative Query Language. IQL explicitly specifies the system's actions.

The GQL International Standard is in the process of being developed.

#### OVERVIEW OF THE RECENT HISTORY OF GRAPH DATABASES

- In 2000, the idea of modeling data as a network came to the founders of Neo4j.
- In 2001, Neo4j developed the earliest core part of the code.
- In 2007, Neo4j started operating as a company.
- In 2009, Neo4j borrowed XPath as a graph query language. Gremlin 11 is also similar to XPath.
- In 2010, Marko Rodriguez, a Neo4j employee, used the term Property Graph to describe the data model of Neo4j and TinkerPop (Gremlin).
- In 2011, the first public version Neo4j 1.4 was released, and the first version of Cypher was released.
- In 2012, Neo4j 1.8 enabled you to write a Cypher. Neo4j 2.0 added labels and indexes. Cypher became a declarative graph query language.
- In 2015, Cypher was opened up by Neo4j through the openCypher project.
- In 2017, the ISO WG3 organization discussed how to use SQL to query property graph data.
- In 2018, Starting from the Neo4j 3.5 GA, the core of Neo4j only for the Enterprise Edition will no longer be open source.
- In 2019, ISO officially established two projects ISO/IEC JTC 1 N 14279 and ISO/IEC JTC 1/SC 32 N 3228 to develop an international standard for graph database language.
- In 2021, the \$325 million Series F funding round for Neo4j marks the largest investment round in database history.

#### THE EARLY HISTORY OF NEO4J

The data model property graph was first conceived in 2000. The founders of Neo4j were developing a media management system, and the schema of the system was often changed. To adapt to such changes, Peter Neubauer, one of the founders, wanted to enable the system to be modeled to a conceptually interconnected network. A group of graduate students at the Indian Institute of Technology Bombay implemented the earliest prototypes. Emil Eifrém, the Neo4j co-founder, and these students spent a week extending Peter's idea into a more abstract model: vertices were connected by relationships, and key-values were used as properties of vertices and relationships. They developed a Java API to interact with this data model and implemented an abstraction layer on top of the relational database.

Although this network model greatly improved productivity, its performance has been poor. So Johan Svensson, Neo4j co-founder, put a lot of effort into implementing a native data management system, that is Neo4j. For the first few years, Neo4j was successful as an in-house product. In 2007, the intellectual property of Neo4j was transferred to an independent database company.

In the first public release of Neo4j (Neo4j 1.4, 2011), the data model was consisted of vertices and typed edges. Vertices and edges have properties. The early versions of Neo4j did not have indexes. Applications had to construct their search structure from the root vertex. Because this was very unwieldy for the applications, Neo4j 2.0 (2013.12) introduced a new concept label on vertices. Based on labels, Neo4j can index some predefined vertex properties.

- 27/1069 - 2023 Vesoft Inc.

"Vertex", "Relationship", "Property", "Relationships can only have one label.", "Vertices can have zero or multiple labels.". All these concepts form the data model definitions for Neo4j property graphs. With the later addition of indexing, Cypher became the main way of interacting with Neo4j. This is because the application developer only needs to focus on the data itself, not on the search structure that the developer built himself as mentioned above.

#### THE CREATION OF GREMLIN

Gremlin is a graph query language based on Apache TinkerPop, which is close in style to a sequence of function (procedure) calls. Initially, Neo4j was queried through the Java API. applications could embed the query engine as a library into the application and then use the API to query the graph.

The early Neo4j employees Tobias Lindaaker, Ivarsson, Peter Neubauer, and Marko Rodriguez used XPath as a graph query. Groovy provides loop structures, branching, and computation. This was the original prototype of Gremlin, the first version of which was released in November 2009.

Later, Marko found a lot of problems with using two different parsers (XPath and Groovy) at the same time and changed Gremlin to a Domain Specific Language (DSL) based on Groovy.

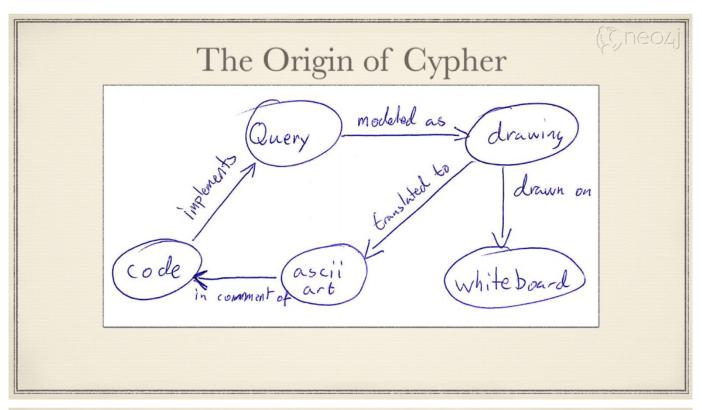
#### THE CREATION OF CYPHER

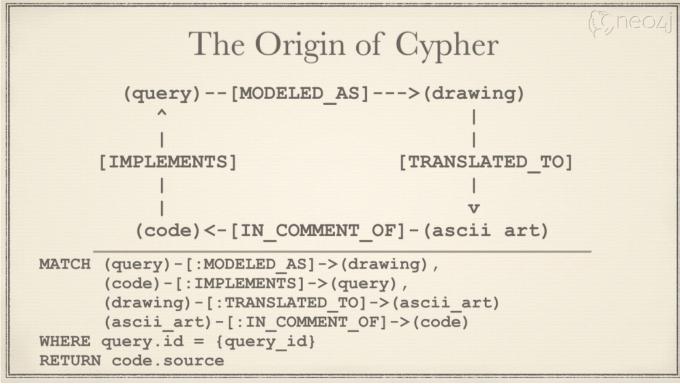
Gremlin, like Neo4j's Java API, was originally intended to be a procedural way of expressing how to query databases. It uses shorter syntaxes to query and remotely access databases through the network. The procedural nature of Gremlin requires users to know the best way to query results, which is still burdensome for application developers. Over the last 30 years, the declarative language SQL has been a great success. SQL can separate the declarative way to get data from how the engine gets data. So the Neo4j engineers wanted to develop a declarative graph query language.

In 2010, Andrés Taylor joined Neo4j as an engineer. Inspired by SQL, he started a project to develop graph query language, which was released as Neo4j 1.4 in 2011. The language is the ancestor of most graph query languages today - Cypher.

Cypher's syntax is based on the use of ASCII art to describe graph patterns. This approach originally came from the annotations on how to describe graph patterns in the source code. An example can be seen as follows.

- 28/1069 - 2023 Vesoft Inc.





Simply put, ASCII art uses printable text to describe vertices and edges. Cypher syntax uses () for vertices and -[]-> for edges. (query)-[modeled as]->(drawing) is used to represent a simple graph relationship (which can also be called graph schema): the starting vertex - query, the destination vertex - drawing, and the edge - modeled as.

The first version of Cypher implemented graph reading, but users should specify vertices from which to start querying. Only from these vertices could graph schema matching be supported.

In a later version, Neo4j 1.8, released in October 2012, Cypher added the ability to modify graphs. However, queries still need to specify which nodes to start from.

In December 2013, Neo4j 2.0 introduced the concept of a label, which is essentially an index. This allows the query engine to use the index to select the vertices matched by the schema, without requiring the user to specify the vertex to start the query.

With the popularity of Neo4j, Cypher has a wide community of developers and is widely used in a variety of industries. It is still the most popular graph query language.

In September 2015, Neo4j established the openCypher Implementors Group (oCIG) to open source Cypher to openCypher, to govern and advance the evolution of the language itself through open source.

#### SUBSEQUENT EVENTS

Cypher has inspired a series of graph query languages, including:

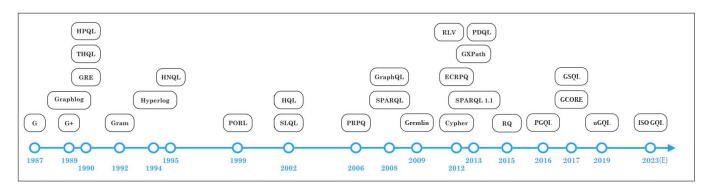
2015, Oracle released PGQL, a graph language used by the graph engine PGX.

2016, the Linked Data Benchmarking Council (short for LDBC) an industry-renowned benchmarking organization for graph performance, released G-CORE.

2018, RedisGraph, a Redis-based graph library, adopted Cypher as its graph language.

2019, the International Standards Organization ISO started two projects to initiate the process of developing an international standard for graph languages based on existing industry achievements such as openCypher, PGQL,  $GSQL^{12}$ , and G-CORE.

2019, NebulaGraph released NebulaGraph Query Language (nGQL) based on openCypher.



# Distributed graph databases

From 2005 to 2010, with the release of Google's cloud computing "Troika", various distributed architectures became increasingly popular, including Hadoop and Cassandra, which have been open-sourced. Several implications are as follows:

- 1. The technical and cost advantages of distributed systems over single machines (e.g. Neo4j) or small machines are more obvious due to the increasing volume of data and computation. Distributed systems allow applications to access these thousands of machines as if they were local systems, without the need for much modification at the code level.
- 2. The open-source approach allows more people to know emerging technologies and feedback to the community in a more cost-effective way, including code developers, data scientists, and product managers.

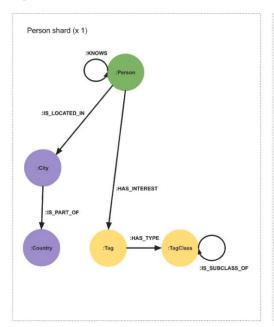
Strictly speaking, Neo4j also offers several distributed capabilities, which are quite different from the industry's sense of the distributed system.

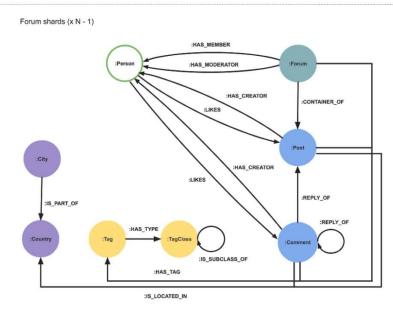
• Neo4j 3. x requires that the full amount of data must be stored on a single machine. Although it supports full replication and high availability between multiple machines, the data cannot be sliced into different subgraphs.

# Neo4j Causal Cluster LEADER READ REPLICA Transaction Replication READ READ REPLICA

Cluster architecture

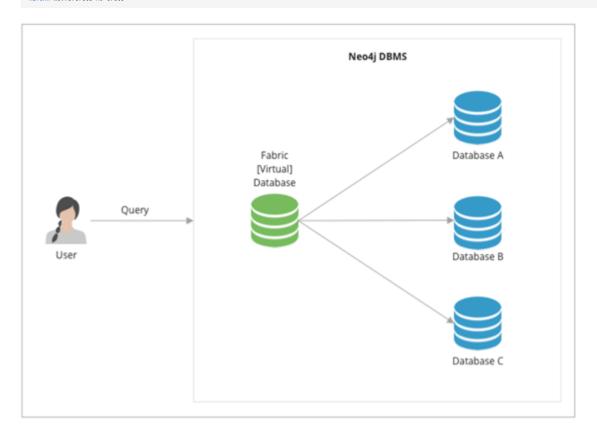
• Neo4j 4. x stores a part of data on different machines (subgraphs), and then the application layer assembles data in a certain way (called Fabric)<sup>13</sup> and distributes the reads and writes to each machine. This approach requires a log of involvement and work from the application layer code. For example, designing how to place different subgraphs on which machines they should be placed and how to assemble some of the results obtained from each machine into the final result.





# The style of its syntax is as follows:

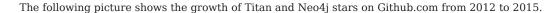
```
USE graphA
MATCH (movie:Movie)
Return movie.title AS title
UNTON
USE graphB
MATCH (move:Movie)
RETURN movie.title AS title
```

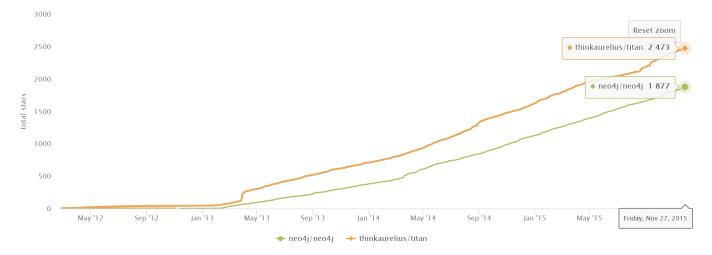


- 32/1069 - 2023 Vesoft Inc.

THE SECOND GENERATION (DISTRIBUTED) GRAPH DATABASE: TITAN AND ITS SUCCESSOR JANUSGRAPH

In 2011, Aurelius was founded to develop an open-source distributed graph database called Titan <sup>14</sup>. By the first official release of Titan in 2015, the backend of Titan can support many major distributed storage architectures (e.g. Cassandra, HBase, Elasticsearch, BerkeleyDB) and can reuse many of the conveniences of the Hadoop ecosystem, with Gremlin as a unified query language on the frontend. It is easy for programmers to use, develop and participate in the community. Large-scale graphs could be sharded and stored on HBase or Cassandra (which were relatively mature distributed storage solutions at the time), and the Gremlin language was relatively full-featured though slightly lengthy. The whole solution was competitive at that time (2011-2015).



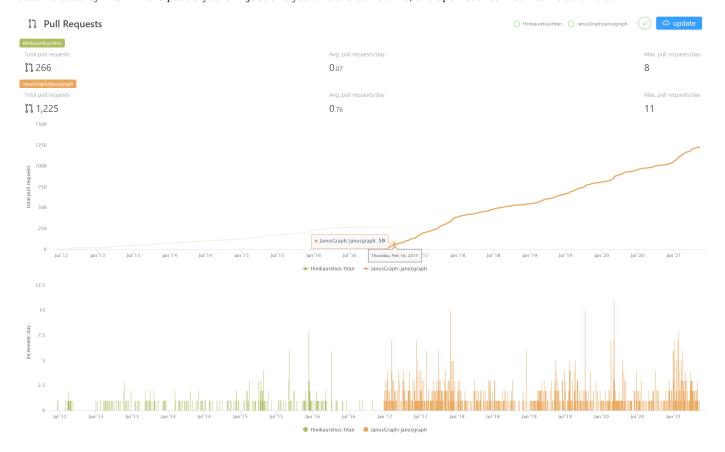


After Aurelius (Titan) was acquired by DataStax in 2015, Titan was gradually transformed into a closed-source commercial product(DataStax Enterprise Graph).

After the acquisition of Aurelius(Titan), there has been a strong demand for an open-source distributed graph database, and there were not many mature and active products in the market. In the era of big data, data is still being generated in a steady stream, far faster than Moore's Law. The Linux Foundation, along with some technology giants (Expero, Google, GRAKN.AI, Hortonworks, IBM, and Amazon) replicated and forked the original Titan project and started it as a new project JanusGraph <sup>15</sup>. Most of the community work including development, testing, release, and promotion, has been gradually shifted to the new JanusGraph.

The following graph shows the evolution of daily code commits (pull requests) for the two projects, and we can see:

- 1. Although Aurelius(Titan) still has some activity in its open-source code after its acquisition in 2015, the growth rate has slowed down significantly. This reflects the strength of the community.
- 2. After the new project was started in January 2017, its community became active quickly, surpassing the number of pull requests accumulated by Titan in the past 5 years in just one year. At the same time, the open-source Titan came to a halt.



FAMOUS PRODUCTS OF THE SAME PERIOD ORIENTDB, TIGERGRAPH, ARANGODB, AND DGRAPH

In addition to JanusGraph managed by the Linux Foundation, more vendors have been joined the overall market. Some distributed graph databases that were developed by commercial companies use different data models and access methods.

- 34/1069 - 2023 Vesoft Inc.

The following table only lists the main differences.

Vendors	Creation time	Core product	Open source protocol	Data model	Query language
OrientDB LTD (Acquired by SAP in 2017)	2011	OrientDB	Open source	Document + KV + Graph	OrientDB SQL (SQL- based extended graph abilities)
GraphSQL (was renamed TigerGraph)	2012	TigerGraph	Commercial version	Graph (Analysis)	GraphSQL (similar to SQL)
ArangoDB GmbH	2014	ArangoDB	Apache License 2.0	Document + KV + Graph	AQL (Simultaneous operation of documents, KVs and graphs)
DGraph Labs	2016	DGraph	Apache Public License 2.0 + Dgraph Community License	Originally RDF, later changed to GraphQL	GraphQL+-

TRADITIONAL GIANTS MICROSOFT, AMAZON, AND ORACLE

In addition to vendors focused on graph products, traditional giants have also entered the graph database field.

Microsoft Azure Cosmos DB<sup>16</sup> is a multimodal database cloud service on the Microsoft cloud that provides SQL, document, graph, key-value, and other capabilities. Amazon AWS Neptune<sup>17</sup> is a graph database cloud service provided by AWS support property graphs and RDF two data models. Oracle Graph<sup>18</sup> is a product of the relational database giant Oracle in the direction of graph technology and graph databases.

NEBULAGRAPH, A NEW GENERATION OF OPEN-SOURCE DISTRIBUTED GRAPH DATABASES

In the following topics, we will formally introduce NebulaGraph, a new generation of open-source distributed graph databases.

- 1. https://db-engines.com/en/ranking categories ←
- $2. \ https://www.yellowfinbi.com/blog/2014/06/yfcommunitynews-big-data-analytics-the-need-for-pragmatism-tangible-benefits-and-real-world-case-165305 \\ \longleftarrow$
- 3. https://www.gartner.com/smarterwithgartner/gartner-top-10-data-and-analytics-trends-for-2021/  $\leftarrow$
- 4. https://www.verifiedmarketresearch.com/product/graph-database-market/ ←
- 5. https://www.globenewswire.com/news-release/2021/01/28/2165742/0/en/Global-Graph-Database-Market-Size-Share-to-Exceed-USD-4-500-Million-By-2026-Facts-Factors.html  $\hookleftarrow$
- 6. https://www.marketsandmarkets.com/Market-Reports/graph-database-market-126230231.html ←
- 7. https://www.gartner.com/en/newsroom/press-releases/2019-07-01-gartner-says-the-future-of-the-database-market-is-the  $\hookleftarrow$
- 8. https://www.amazon.com/Designing-Data-Intensive-Applications-Reliable-Maintainable/dp/1449373321 🛩
- 9. I. F. Cruz, A. O. Mendelzon, and P. T. Wood. A Graphical Query Language Supporting Recursion. In Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data, pages 323–330. ACM Press, May 1987. ←
- 10. "An overview of the recent history of Graph Query Languages". Authors: Tobias Lindaaker, U.S. National Expert.Date: 2018-05-14 ←
- 11. Gremlin is a graph language developed based on Apache TinkerPop.  $\leftarrow$
- 12. https://docs.tigergraph.com/dev/gsql-ref ←
- 13. https://neo4j.com/fosdem20/  $\boldsymbol{\leftarrow}$
- 14. https://github.com/thinkaurelius/titan ←
- 15. https://github.com/JanusGraph/janusgraph  $\boldsymbol{\leftarrow}$
- 16. https://azure.microsoft.com/en-us/free/cosmos-db/ ←
- 17. https://aws.amazon.com/cn/neptune/ ←
- 18. https://www.oracle.com/database/graph/ ←

- 35/1069 - 2023 Vesoft Inc.

Last update: February 19, 2024

### 2.3 Related technologies

This topic introduces databases and graph-related technologies that are closely related to distributed graph databases.

#### 2.3.1 Databases

#### Relational databases

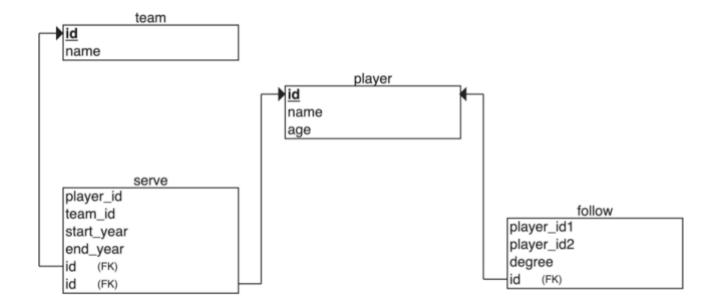
A relational database is a database that uses a relational model to organize data. The relational model is a two-dimensional table model, and a relational database consists of two-dimensional tables and the relationships between them. When it comes to relational databases, most people think of MySQL, one of the most popular database management systems that support database operations using the most common structured query language (SQL) and stores data in the form of tables, rows, and columns. This approach to storing data is derived from the relational data model proposed by Edgar Frank Codd in 1970.

In a relational database, a table can be created for each type of data to be stored. For example, the player table is used to store all player information, the team table is used to store team information. Each row of data in a SQL table must contain a primary key. The primary key is a unique identifier for the row of data. Generally, the primary key is self-incrementing with the number of rows as the field ID. Relational databases have served the computer industry very well since their inception and will continue to do so for a long time to come.

If you have used Excel, WPS, or other similar applications, you have a rough idea of how relational databases work. First, you set up the columns, then you add rows of data under the corresponding columns. You can average or otherwise aggregate the data in a column, similar to averaging in a relational database MySQL. Pivot tables in Excel are the equivalent of querying data in a relational database MySQL using aggregation functions and CASE statements. An Excel file can have multiple tables, and a single table is equivalent to a single table in MySQL. An Excel file is similar to a MySQL database.

#### RELATIONSHIPS IN RELATIONAL DATABASES

Unlike graph databases, edges in relational databases (or SQL-type databases) are also stored as entities in specialized edge tables. Two tables are created, player and team, and then player\_team is created as an edge table. Edge tables are usually formed by joining related tables. For example, here the edge table player\_team is made by joining the player table and the team table.



The way of storing edges is not a big problem when associating small data sets, but problems arise when there are too many relationships in a relational database. Specifically, when you want to query just one player's teammates, you have to join all the data in the table and then filter out all the data you don't need, which puts a huge strain on the relational database when your

dataset reaches a certain size. If you want to associate multiple different tables, the system may not be able to respond before the join bombs.

#### ORIGINS OF RELATIONAL DATABASES

As mentioned above, the relational data model was first proposed by Edgar Frank Codd, an IBM engineer, in 1970. Codd wrote several papers on database management systems that addressed the potential of the relational data model. The relational data model does not rely on linked lists of data (mesh or hierarchical data), but more on data sets. Using the mathematical method of tuple calculus, he argued that these datasets can perform the same tasks as a navigational database. The only requirement was that the relational data model needed a suitable query language to guarantee the consistency requirements of the database. This became the inspiration for declarative query languages such as Structured Query Language (SQL). IBM's System R was one of the first implementations of such a system. But Software Development Laboratories, a small company founded by ex-IBM people and one illustrious Mr.Larry Ellison, beat IBM to the market with the product that would become known as Oracle.

Since the relational database was a trendy term at the time, many database vendors preferred to use it in their product names, even though their products were not actually relational. To prevent this and reduce the misuse of the relational data model, Codd introduced the famous Codd's 12 Rules. All relational data systems must follow Codd's 12 Rules.

#### NoSQL databases

Graph databases are not the only alternative that can overcome the shortcomings of relational databases. There are many non-relational database products on the market that can be called NoSQL. The term NoSQL was first introduced in the late 1990s and can be interpreted as "not SQL" or "not only SQL". For the sake of understanding, NoSQL can be interpreted as a "non-relational database" here. Unlike relational databases, the data storage and retrieval mechanisms provided by NoSQL databases are not modeled based on table relationships. NoSQL databases can be divided into four categories.

- Key-value Data Store
- Columnar Store
- · Document Store
- Graph Store

The following describes the four types of NoSQL databases.

#### KEY-VALUE DATA STORE

Key-value databases store data in unique key-value pairs. Unlike relational databases, key-value stores do not have tables and columns. A key-value database itself is like a large table with many columns (i.e., keys). In a key-value store database, data are stored and queried by means of keys, usually implemented as hash lists. This is much simpler than traditional SQL databases, and for some web applications, it is sufficient.

The advantage of the key-value model for IT systems is that it is simple and easy to deploy. In most cases, this type of storage works well for unrelated data. If you are just storing data without querying it, there is no problem using this storage method. However, if the DBA only queries or updates some of the values, the key-value model becomes inefficient. Common key-value storage databases include Redis, Voldemort, and Oracle BDB.

### COLUMNAR STORE

A NoSQL database's columnar store has many similarities to a NoSQL database's key-value store because the columnar store is still using keys for storage and retrieval. The difference is that in a columnar store database, the column is the smallest storage unit, and each column consists of a key, a value, and a timestamp for version control and conflict resolution. This is particularly useful when scaling in a distributed manner, as timestamps can be used to locate expired data when the database is updated. Because of the good scalability of columnar storage, the columnar store is suitable for very large data sets. Common columnar storage databases include HBase, Cassandra, HadoopDB, etc.

#### DOCUMENT STORE

A NoSQL database document store is a key-value-based database, but with enhanced functionality. Data is still stored as keys, but the values in a document store are structured documents, not just a string or a single value. That is, because of the increased information structure, document stores are able to perform more optimized queries and make data retrieval easier. Therefore,

- 38/1069 - 2023 Vesoft Inc.

**Graph**Aware

document stores are particularly well suited for storing, indexing, and managing document-oriented data or similar semistructured data.

Technically speaking, as a semi-structured unit of information, a document in a document store can be any form of document available, including XML, JSON, YAML, etc., depending on the design of the database vendor. For example, JSON is a common choice. While JSON is not the best choice for structured data, JSON-type data can be used in both front-end and back-end applications. Common document storage databases include MongoDB, CouchDB, Terrastore, etc.

#### **GRAPH STORE**

The last class of NoSQL databases is graph databases. NebulaGraph, is also a graph database. Although graph databases are also NoSQL databases, graph databases are fundamentally different from the above-mentioned NoSQL databases. Graph databases store data in the form of vertices, edges, and properties. Its advantages include high flexibility, support for complex graph algorithms, and can be used to build complex relational graphs. We will discuss graph databases in detail in the subsequent topics. But in this topic, you just need to know that a graph database is a NoSQL type of database. Common graph databases include NebulaGraph, Neo4j, OrientDB, etc.

### 2.3.2 Graph-related technologies

Take a look at a panoramic view of graph technology in 2020 <sup>1</sup>.

## 7VORTEX Vlumify 🟡 🕟 📦 BeGraph 🖦 tin 🕽 ATORIAL\_BLAS DeepGraphLibrary Galons GQL A GraphQL SPARQL MissionGraph™ NEORIS turi 🦟 SemSpect O Unigraph:io W WordLift 🦂 osigmajs Springy.js neo4j graphgrid AYASDI Datarama ELLIPTIC W KBBS W Graph Story structr GraphConnect 🔁 🛦 🛦 Ancudii 鴡 BLAIR 🙈 🐭 🕏 **FOSDEM** graph 🤋 talend 🛡 =confluent 🔅 🚾 🌬 Paxata day structr 🕏 StyleTe ets THALES

#### GRAPH TECHNOLOGY LANDSCAPE 2020

There are many technologies that are related to graphs, which can be broadly classified into these categories:

- Infrastructure: Graph databases, graph computing (processing) engines, graph deep learning, cloud services, etc.
- · Applications: Visualization, knowledge graph, anti-fraud, cyber security, social network, etc.
- Development tools: Graph query languages, modeling tools, development frameworks, and libraries.
- E-books <sup>2</sup> and conferences, etc.

### **Graph language**

In the previous topic, we introduced the history of graph languages. In this section, we make a classification of the functions of graph languages.

- Nearest neighbor query (NNS): Query the neighboring edges, neighbors, or K-hops neighbors.
- Find one/all subgraphs that satisfy a given graph pattern. This problem is very close to Subgraph Isomorphism two seemingly different graphs that are actually identical <sup>3</sup> as shown below.

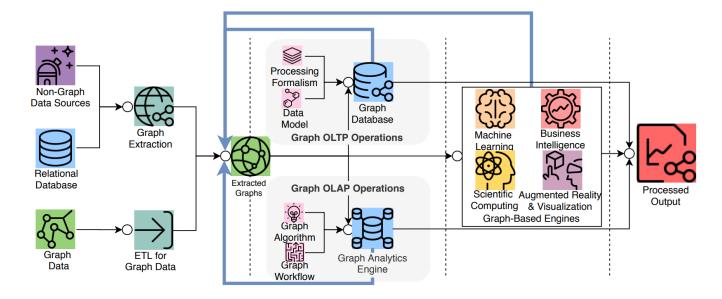
Graph G	Graph H	An isomorphism between G and H
		f(a) = 1
a g	2	f(b) = 6
	5 6	f(c) = 8
b		f(a) = 3
C	8 7	f(g) = 5
		f(h) = 2
d j	4	f(i) = 4
		f(j) = 7

- Reachability (connectivity) problems: The most common reachability problem is the shortest path problem. Such problems are usually described in terms of Regular Path Query a series of connected groups of vertices forming a path that needs to satisfy some regular expression.
- Analytic problems: It is related to some convergent operators, such as Average, Count, Max, Vertex Degree. Measures the distance between all two vertices, the degree of interaction between a vertex and other vertices.

### Graph database and graph processing systems

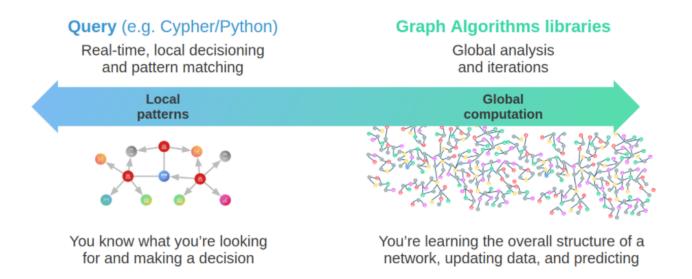
A graph system usually includes a complex data pipeline <sup>4</sup>. From the data source (the left side of the picture below) to the processing output (the right side), multiple data processing steps and systems are used, such as the ETL module, Graph OLTP module, OLAP module, BI, and knowledge graph.

- 40/1069 - 2023 Vesoft Inc.



Graph databases and graph processing systems have different origins and specialties (and weaknesses).

- (Online) The graph database is designed for persistent storage management of graphs and efficient subgraph operations. Hard disks and network are the target operating devices, physical/logical data mapping, data integrity, and (fault) consistency are the main goals. Each request typically involves only a small part of the full graph and can usually be done on a single server. Request latency is usually in milliseconds or seconds, and request concurrency is typically in the thousands or hundreds of thousands. The early Neo4j was one of the origins of the graph database space.
- (Offline) The graph processing system is for high-volume, concurrency, iteration, processing, and analysis of the full graph. Memory and network are the target operating devices. Each request involves all graph vertices and requires all servers to be involved in its completion. The latency of a single request is in the range of minutes to hours (days). The request concurrency is in single digits. Google's Pregel <sup>5</sup> represents the typical origin of graph processing systems. Its point-centric programming abstraction and BSP's operational model constitute a programming paradigm that is a more graph-friendly API abstraction than the previous Hadoop Map-Reduce.



- 41/1069 - 2023 Vesoft Inc.

#### Graph sharding methods

For large-scale graph data, it is difficult to store it in the memory of a single server, and even just storing the graph structure itself is not enough. By increasing the capacity of a single server, its cost price usually rises exponentially.

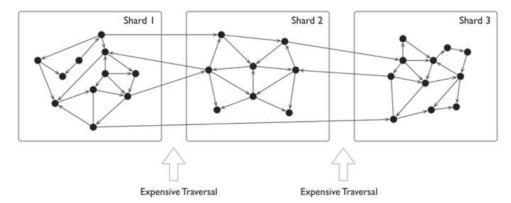
As the volume of data increases, for example, 100 billion data already exceeds the capacity of all commercially available servers on the market.

Another option is to shard data and place each shard on a different server to increase reliability and performance. For NoSQL systems, such as key-value or document systems, the sharding method is intuitive and natural. Each record and data unit can usually be placed on a different server based on the key or docID.

However, the sharding of data structures like graphs is usually less intuitive, because usually, graphs are "fully connected" and each vertex can be connected to any other vertex in usually 6 hops.

And it has been theoretically proven that the graph sharding problem is NP.

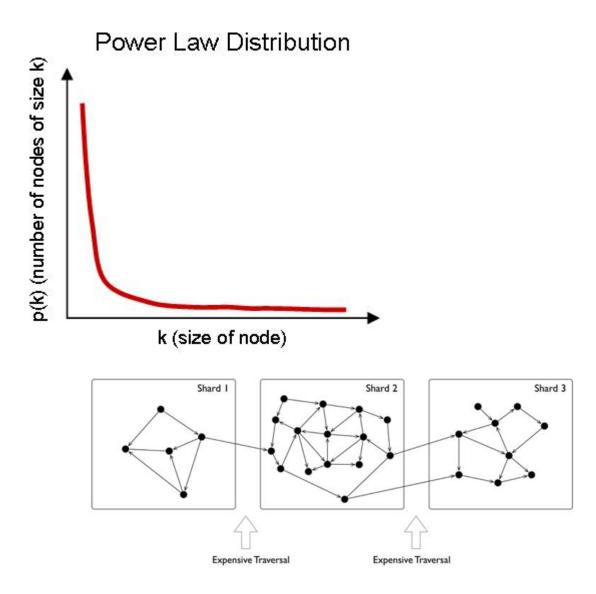
When distributing the entire graph data across multiple servers, the cross-server network access latency is 10 times higher than the hardware (memory) access time inside the same server. Therefore, for some depth-first traversal scenarios, a large number of cross-network accesses occur, resulting in extremely high overall latency.



7

Usually, graphs have a clear power-law distribution. A small number of vertices have much denser neighboring edges than the average vertices. Though processing these vertices can usually be within the same server which reduces cross-network access, load will be far more heavier than the average.

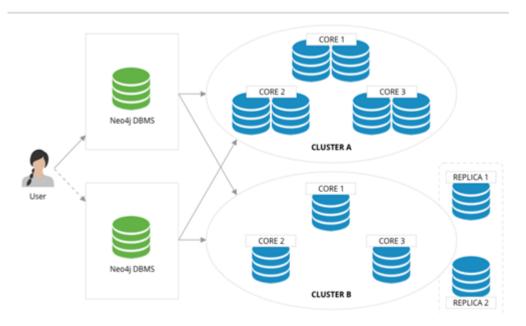
- 42/1069 - 2023 Vesoft Inc.



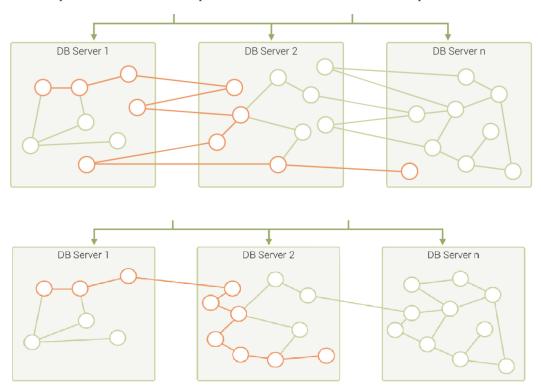
The common graph sharding methods are as follows:

• Application-level sharding: The application layer senses and controls which shard each vertex and edge should locate on based on the type of vertices and edges. A set of vertices of the same type is placed on one sharding and another set of vertices of the same type is placed on another sharding. Of course, for high reliability, the sharding itself can also be made multiple replicas. When used by the application, the desired vertices and edges are fetched from each shard, and then on the off-application side (or some proxy server-side), the fetched data is assembled into the final result. This is typically represented by the Neo4j 4. x Fabric.

- 43/1069 - 2023 Vesoft Inc.



- Using a distributed cache layer: Add a memory cache layer on the top of the hard disk and cache important portions of the sharding and data and preheat that cache.
- Adding read-only replicas or views: Add read-only replicas or create a view for some of the graph sharding, and pass the heavier load of read requests through these sharding servers.
- Performing fine-grained graph sharding: Form multiple small partitions of vertices and edges instead of one large sharding, and then place the more correlated partitions on the same server as much as possible. 8.



A mixture of these approaches is also used in specific engineering practices. Usually, offline graph processing systems perform some degree of graph preprocessing to improve the locality through an ETL process, while online graph database systems usually choose a periodic data rebalancing process to improve data locality.

#### Technical challenges

In the literature <sup>9</sup>, a thorough investigation of graphs and challenges is done, and the following lists the top ten graph technology challenges.

- Scalability: Loading and upgrading big graphs, performing graph computation and graph traversal, use of triggers and supernodes
- · Visualization: Customizable layouts, rendering and display big images, and display dynamic and updated display
- Query language and programming API: Language expressiveness, standards compatibility, compatibility with existing systems, design of subqueries, and associative queries across multiple graphs
- · Faster graph algorithms
- Easy to use (configuration and usage)
- · Performance metrics and testing
- General graph technology software (e.g., to handle offline, online, streaming computations.)
- ETL
- · Debug and test

#### Open-source graph tools on single machines

There is a common misconception about graph databases that any data access involving graph structure needs to be stored in a graph database.

When the amount of data is not large, single machine memory is enough to store the data. You can use some single-machine open-source tools to store tens of millions of vertices and edges.

- JGraphT<sup>10</sup>: A well-known open-source Java graph theory library, which implements a considerable number of efficient graph algorithms.
- igraph 11: A lightweight and powerful library, supporting R, Python, and C++.
- NetworkX<sup>12</sup>: The first choice for data scientists doing graph theory analysis.
- Cytoscape 13: A powerful visual open-source graph analysis tool.
- Gephi<sup>14</sup>: A powerful visual open-source graph analysis tool.
- arrows.app<sup>15</sup>: A simple brain mapping tool for visually generating Cypher statements.

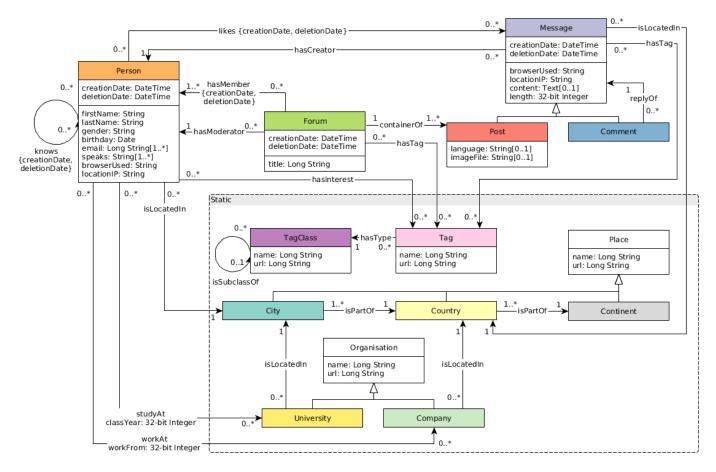
#### Industry databases and benchmarks

LDBC

LDBC<sup>16</sup> (Linked Data Benchmark Council) is a non-profit organization composed of hardware and software giants such as Oracle, Intel and mainstream graph database vendors such as Neo4j and TigerGraph, which is the benchmark guide developer and test result publisher for graphs and has a high influence in the industry.

SNB (Social Network Benchmark) is one of the benchmarks developed by the Linked Data Benchmark Committee (LDBC) for graph databases and is divided into two scenarios: interactive query (Interactive) and business intelligence (BI). Its role is similar to that of TPC-C, TPC-H, and other tests in SQL-type databases, which can help users compare the functions, performance, and capacity of various graph database products.

An SNB dataset simulates the relationship between people and posts of a social network, taking into account the distribution properties of the social network, the activity of people, and other social information.

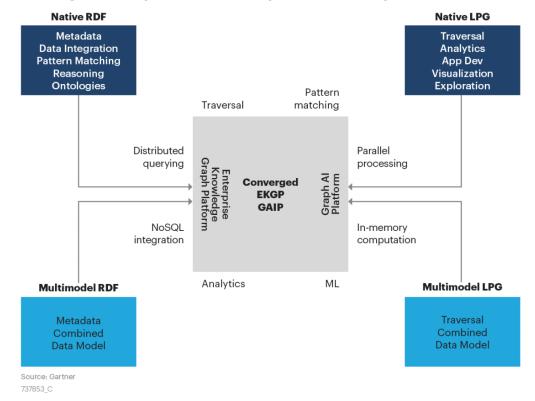


The standard data size ranges from 0.1 GB (scale factor 0.1) to 1000 GB (sf 1000). Larger data sets of 10 TB and 100 TB can also be generated. The number of vertices and edges is as shown below.

Scale Factor	0.1	0.3	1	3	10	30	100	300	1000
# of Persons	1.5K	3.5K	11K	27K	73K	182K	499K	1.25M	3.6M
# of nodes	327.6K	908K	3.2M	9.3M	30M	88.8M	282.6M	817.3M	2.7B
# of edges	1.5M	4.6M	17.3M	52.7M	176.6M	540.9M	1.8B	5.3B	17B

Graph technologies of different origins and goals are learning from and integrating with each other

## Convergence of Capabilities in the Graph DBMS Landscape



**Gartner** 

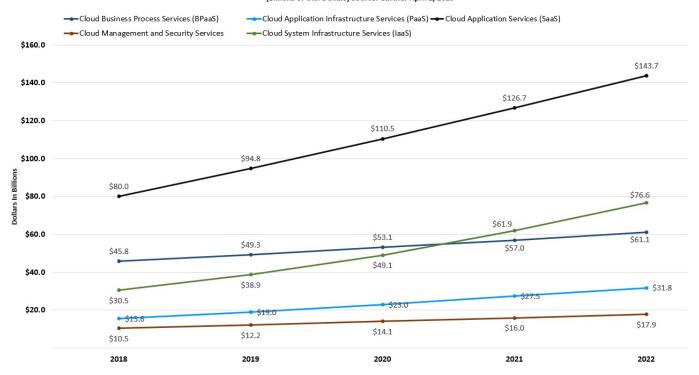
#### The trends in cloud computing place higher demands on scalability.

According to Gartner's projections, cloud services have been growing at a rapid rate and penetration <sup>17</sup>. A large number of commercial software is gradually moving from a completely local and private model 10 years ago to a cloud services-based business model. One of the major advantages of cloud services is that they offer near-infinite scalability. It requires that various cloud infrastructure-based software must have a better ability to scale quickly and elastically.

- 47/1069 - 2023 Vesoft Inc.

#### Worldwide Public Cloud Service Revenue Forecast, 2018 - 2022

(Billions of U.S. Dollars) Source: Gartner April 2, 2019

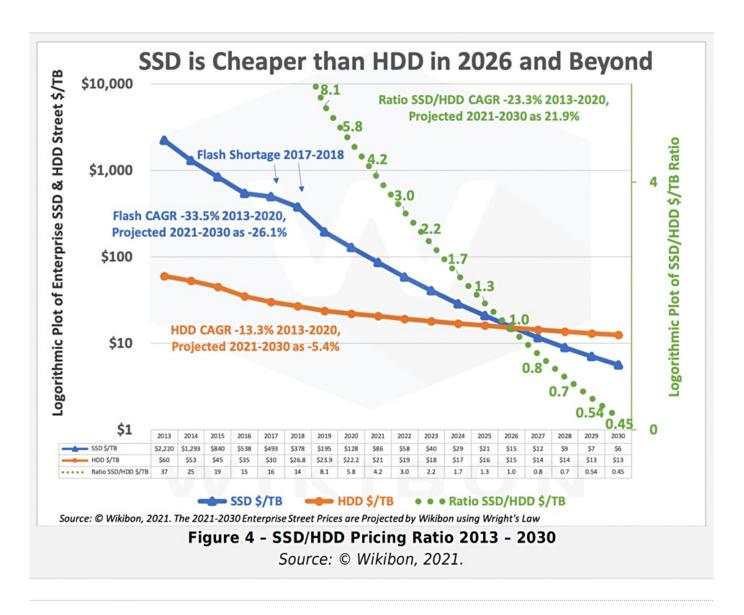


#### Trends in hardware that SSD will be the mainstream persistent device

Hardware determines software architecture. From the 1950s, when Moore's Law was discovered, to the 00s, when multi-core was introduced, hardware trends and speeds have profoundly determined software architecture. Database systems are mostly designed around "hard disk + memory", high-performance computing systems are mostly designed around "memory + CPU", and distributed systems are designed completely differently for 1 gigabit, 10 gigabits, and RDMA.

Graph traversals are featured as random access. Early graph database systems adopted the large memory + HDD architecture. By designing some data structure in memory, random access can be achieved in memory (B+ trees, Hash tables) for the purpose of optimizing graph topology traversal. And then the random access was converted into sequential reads and writes suitable for HDDs. The entire software architecture (including the storage and compute layers) must be based on and built around such IO processes. With the decline in SSD prices <sup>18</sup>, SSDs are replacing HDDs as the dominant device. Friendly random access, deep IO queue, fast access are the features of SSD that differ from HDD's highly repetitive sequence, random latency, and easily damaged disk. The redesign for all software architectures becomes a heavy historical technical burden.

- 48/1069 - 2023 Vesoft Inc.



- 1. https://graphaware.com/graphaware/2020/02/17/graph-technology-landscape-2020.html ←
- 2. Electronic copies are available for learning purposes by contacting Author. ←
- 3. https://en.wikipedia.org/wiki/Graph\_isomorphism ←
- 4. The Future is Big Graphs! A Community View on Graph Processing Systems. https://arxiv.org/abs/2012.06171 ←
- 5. G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In Proceedings of the International Conference on Management of data (SIGMOD), pages 135–146, New York, NY, USA, 2010. ACM ←
- 6. https://neo4j.com/graphacademy/training-iga-40/02-iga-40-overview-of-graph-algorithms/  $\leftarrow$
- 7. https://livebook.manning.com/book/graph-powered-machine-learning/welcome/v-8/ ←
- 8. https://www.arangodb.com/learn/graphs/using-smartgraphs-arangodb/ ←
- 9. https://arxiv.org/abs/1709.03188 ←
- 10. https://jgrapht.org/ ←
- 11. https://igraph.org/  $\leftarrow$
- 12. https://networkx.org/ ←
- 13. https://cytoscape.org/ ←
- 14. https://gephi.org/ ←
- 15. https://arrows.app/ ←
- 16. https://github.com/ldbc/ldbc\_snb\_docs ←
- $17.\ https://cloudcomputing-news.net/news/2019/apr/15/public-cloud-soaring-to-331b-by-2022-according-to-gartner/$
- 18. https://blocksandfiles.com/2021/01/25/wikibon-ssds-vs-hard-drives-wrights-law/ ←

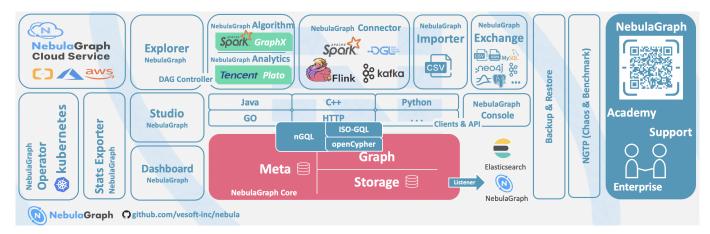
- 49/1069 - 2023 Vesoft Inc.

Last update: February 19, 2024

- 50/1069 - 2023 Vesoft Inc.

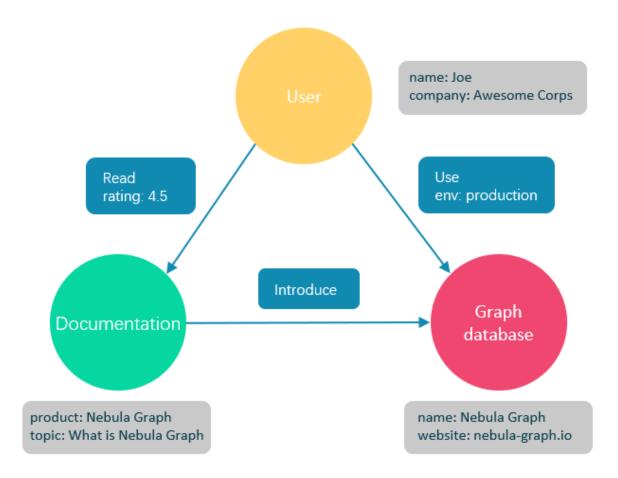
## 2.4 What is NebulaGraph

NebulaGraph is an open-source, distributed, easily scalable, and native graph database. It is capable of hosting graphs with hundreds of billions of vertices and trillions of edges, and serving queries with millisecond-latency.



### 2.4.1 What is a graph database

A graph database, such as NebulaGraph, is a database that specializes in storing vast graph networks and retrieving information from them. It efficiently stores data as vertices (nodes) and edges (relationships) in labeled property graphs. Properties can be attached to both vertices and edges. Each vertex can have one or multiple tags (labels).



Graph databases are well suited for storing most kinds of data models abstracted from reality. Things are connected in almost all fields in the world. Modeling systems like relational databases extract the relationships between entities and squeeze them into table columns alone, with their types and properties stored in other columns or even other tables. This makes data management time-consuming and cost-ineffective.

NebulaGraph, as a typical native graph database, allows you to store the rich relationships as edges with edge types and properties directly attached to them.

## 2.4.2 Advantages of NebulaGraph

#### Open source

NebulaGraph is open under the Apache 2.0 License. More and more people such as database developers, data scientists, security experts, and algorithm engineers are participating in the designing and development of NebulaGraph. To join the opening of source code and ideas, surf the NebulaGraph GitHub page.

### **Outstanding performance**

Written in C++ and born for graphs, NebulaGraph handles graph queries in milliseconds. Among most databases, NebulaGraph shows superior performance in providing graph data services. The larger the data size, the greater the superiority of NebulaGraph.For more information, see NebulaGraph benchmarking.

- 52/1069 - 2023 Vesoft Inc.

#### **High scalability**

NebulaGraph is designed in a shared-nothing architecture and supports scaling in and out without interrupting the database service.

#### **Developer friendly**

NebulaGraph supports clients in popular programming languages like Java, Python, C++, and Go, and more are under development. For more information, see NebulaGraph clients.

#### Reliable access control

NebulaGraph supports strict role-based access control and external authentication servers such as LDAP (Lightweight Directory Access Protocol) servers to enhance data security. For more information, see <u>Authentication and authorization</u>.

#### Diversified ecosystem

More and more native tools of NebulaGraph have been released, such as NebulaGraph Studio, NebulaGraph Console, and NebulaGraph Exchange. For more ecosystem tools, see Ecosystem tools overview.

Besides, NebulaGraph has the ability to be integrated with many cutting-edge technologies, such as Spark, Flink, and HBase, for the purpose of mutual strengthening in a world of increasing challenges and chances.

#### OpenCypher-compatible query language

The native NebulaGraph Query Language, also known as nGQL, is a declarative, openCypher-compatible textual query language. It is easy to understand and easy to use. For more information, see nGQL guide.

### Future-oriented hardware with balanced reading and writing

Solid-state drives have extremely high performance and they are getting cheaper. NebulaGraph is a product based on SSD. Compared with products based on HDD and large memory, it is more suitable for future hardware trends and easier to achieve balanced reading and writing.

### Easy data modeling and high flexibility

You can easily model the connected data into NebulaGraph for your business without forcing them into a structure such as a relational table, and properties can be added, updated, and deleted freely. For more information, see <u>Data modeling</u>.

#### High popularity

NebulaGraph is being used by tech leaders such as Tencent, Vivo, Meituan, and JD Digits. For more information, visit the NebulaGraph official website.

### 2.4.3 Use cases

NebulaGraph can be used to support various graph-based scenarios. To spare the time spent on pushing the kinds of data mentioned in this section into relational databases and on bothering with join queries, use NebulaGraph.

#### Fraud detection

Financial institutions have to traverse countless transactions to piece together potential crimes and understand how combinations of transactions and devices might be related to a single fraud scheme. This kind of scenario can be modeled in graphs, and with the help of NebulaGraph, fraud rings and other sophisticated scams can be easily detected.

- 53/1069 - 2023 Vesoft Inc.

#### Real-time recommendation

NebulaGraph offers the ability to instantly process the real-time information produced by a visitor and make accurate recommendations on articles, videos, products, and services.

### Intelligent question-answer system

Natural languages can be transformed into knowledge graphs and stored in NebulaGraph. A question organized in a natural language can be resolved by a semantic parser in an intelligent question-answer system and re-organized. Then, possible answers to the question can be retrieved from the knowledge graph and provided to the one who asked the question.

### Social networking

Information on people and their relationships is typical graph data. NebulaGraph can easily handle the social networking information of billions of people and trillions of relationships, and provide lightning-fast queries for friend recommendations and job promotions in the case of massive concurrency.

### 2.4.4 Related links

- Official website
- Docs
- Blogs
- Forum
- GitHub

Last update: February 19, 2024

- 54/1069 - 2023 Vesoft Inc.

## 2.5 Data modeling

A data model is a model that organizes data and specifies how they are related to one another. This topic describes the Nebula Graph data model and provides suggestions for data modeling with NebulaGraph.

#### 2.5.1 Data structures

NebulaGraph data model uses six data structures to store data. They are graph spaces, vertices, edges, tags, edge types and properties.

- **Graph spaces**: Graph spaces are used to isolate data from different teams or programs. Data stored in different graph spaces are securely isolated. Storage replications, privileges, and partitions can be assigned.
- Vertices: Vertices are used to store entities.
- In NebulaGraph, vertices are identified with vertex identifiers (i.e. VID). The VID must be unique in the same graph space. VID should be int64, or fixed string(N).
- A vertex has zero to multiple tags.



In NebulaGraph 2.x a vertex must have at least one tag. And in NebulaGraph 3.4.1, a tag is not required for a vertex.

- Edges: Edges are used to connect vertices. An edge is a connection or behavior between two vertices.
- There can be multiple edges between two vertices.
- Edges are directed. -> identifies the directions of edges. Edges can be traversed in either direction.
- An edge is identified uniquely with <a source vertex, an edge type, a rank value, and a destination vertex>. Edges have no EID.
- An edge must have one and only one edge type.
- The rank value is an immutable user-assigned 64-bit signed integer. It identifies the edges with the same edge type between two vertices. Edges are sorted by their rank values. The edge with the greatest rank value is listed first. The default rank value is zero.
- $\bullet \ Tags: {\tt Tags} \ are \ used \ to \ categorize \ vertices. \ Vertices \ that \ have \ the \ same \ tag \ share \ the \ same \ definition \ of \ properties.$
- **Edge types**: Edge types are used to categorize edges. Edges that have the same edge type share the same definition of properties.
- Properties: Properties are key-value pairs. Both vertices and edges are containers for properties.



Tags and Edge types are similar to "vertex tables" and "edge tables" in the relational databases.

### 2.5.2 Directed property graph

NebulaGraph stores data in directed property graphs. A directed property graph has a set of vertices connected by directed edges. Both vertices and edges can have properties. A directed property graph is represented as:

- 55/1069 - 2023 Vesoft Inc.

# $G = \langle V, E, P_{V'}, P_{E} \rangle$

- ullet V is a set of vertices.
- E is a set of directed edges.
- $\mathbf{P}_{\mathbf{V}}$  is the property of vertices.
- $\bullet$   $\boldsymbol{P}_{\boldsymbol{E}}$  is the property of edges.

The following table is an example of the structure of the basketball player dataset. We have two types of vertices, that is **player** and **team**, and two types of edges, that is **serve** and **follow**.

Element	Name	Property name (Data type)	Description
Tag	player	name (string) age (int)	Represents players in the team.
Tag	team	name (string)	Represents the teams.
Edge type	serve	start_year (int) end_year (int)	Represents actions taken by players in the team.  An action links a player with a team, and the direction is from a player to a team.
Edge type	follow	degree (int)	Represents actions taken by players in the team.  An action links a player with another player, and the direction is from one player to the other player.



NebulaGraph supports only directed edges.



NebulaGraph 3.4.1 allows dangling edges. Therefore, when adding or deleting, you need to ensure the corresponding source vertex and destination vertex of an edge exist. For details, see INSERT VERTEX, DELETE VERTEX, INSERT EDGE, and DELETE EDGE.

The MERGE statement in openCypher is not supported.

Last update: February 19, 2024

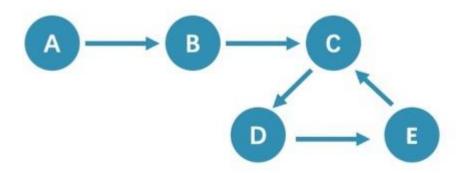
- 56/1069 - 2023 Vesoft Inc.

## 2.6 Path types

In graph theory, a path in a graph is a finite or infinite sequence of edges which joins a sequence of vertices. Paths are fundamental concepts of graph theory.

Paths can be categorized into 3 types: walk, trail, and path. For more information, see Wikipedia.

The following figure is an example for a brief introduction.



#### 2.6.1 Walk

A walk is a finite or infinite sequence of edges. Both vertices and edges can be repeatedly visited in graph traversal.

In the above figure C, D, and E form a cycle. So, this figure contains infinite paths, such as A->B->C->D->E, A->B->C->D->E->C, and A->B->C->D->E->C->D->C->D->E->C->D->E->C->D->E->C->D->E->C->D->E->C->D->E->C->D->E->C->D->



GO statements use walk.

### 2.6.2 Trail

A trail is a finite sequence of edges. Only vertices can be repeatedly visited in graph traversal. The Seven Bridges of Königsberg is a typical trail.

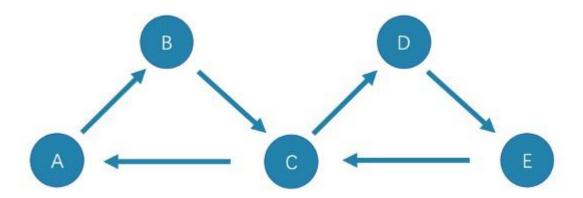
In the above figure, edges cannot be repeatedly visited. So, this figure contains finite paths. The longest path in this figure consists of 5 edges: A->B->C->D->E->C.



MATCH, FIND PATH, and GET SUBGRAPH statements use trail.

There are two special cases of trail, cycle and circuit. The following figure is an example for a brief introduction.

- 57/1069 - 2023 Vesoft Inc.



### • cycle

A cycle refers to a closed trail. Only the terminal vertices can be repeatedly visited. The longest path in this figure consists of 3 edges: A->B->C->A or C->D->E->C.

### • circuit

A circuit refers to a closed trail. Edges cannot be repeatedly visited in graph traversal. Apart from the terminal vertices, other vertices can also be repeatedly visited. The longest path in this figure: A->B->C->D->E->C->A.

### 2.6.3 Path

A path is a finite sequence of edges. Neither vertices nor edges can be repeatedly visited in graph traversal.

So, the above figure contains finite paths. The longest path in this figure consists of 4 edges: A->B->C->D->E.

Last update: February 19, 2024

- 58/1069 - 2023 Vesoft Inc.

### 2.7 VID

In a graph space, a vertex is uniquely identified by its ID, which is called a VID or a Vertex ID.

#### 2.7.1 Features

- The data types of VIDs are restricted to FIXED\_STRING(<N>) or INT64. One graph space can only select one VID type.
- A VID in a graph space is unique. It functions just as a primary key in a relational database. VIDs in different graph spaces are independent.
- The VID generation method must be set by users, because NebulaGraph does not provide auto increasing ID, or UUID.
- Vertices with the same VID will be identified as the same one. For example:
- A VID is the unique identifier of an entity, like a person's ID card number. A tag means the type of an entity, such as driver, and boss. Different tags define two groups of different properties, such as driving license number, driving age, order amount, order taking alt, and job number, payroll, debt ceiling, business phone number.
- When two INSERT statements (neither uses a parameter of IF NOT EXISTS) with the same VID and tag are operated at the same time, the latter INSERT will overwrite the former.
- When two INSERT statements with the same VID but different tags, like TAG A and TAG B, are operated at the same time, the operation of Tag A will not affect Tag B.
- VIDs will usually be indexed and stored into memory (in the way of LSM-tree). Thus, direct access to VIDs enjoys peak
  performance.

### 2.7.2 VID Operation

- NebulaGraph 1.x only supports INT64 while NebulaGraph 2.x supports INT64 and FIXED\_STRING(<N>). In CREATE SPACE, VID types can be set via vid\_type.
- id() function can be used to specify or locate a VID.
- ullet LOOKUP or MATCH statements can be used to find a VID via property index.
- Direct access to vertices statements via VIDs enjoys peak performance, such as DELETE xxx WHERE id(xxx) = "player100" or GO FROM "player100". Finding VIDs via properties and then operating the graph will cause poor performance, such as LOOKUP | GO FROM \$-.ids, which will run both LOOKUP and | one more time.

### 2.7.3 VID Generation

VIDs can be generated via applications. Here are some tips:

- (Optimal) Directly take a unique primary key or property as a VID. Property access depends on the VID.
- $\bullet \ \ \text{Generate a VID via a unique combination of properties. Property access depends on property index.}$
- Generate a VID via algorithms like snowflake. Property access depends on property index.
- If short primary keys greatly outnumber long primary keys, do not enlarge the N of FIXED\_STRING(<N>) too much. Otherwise, it will occupy a lot of memory and hard disks, and slow down performance. Generate VIDs via BASE64, MD5, hash by encoding and splicing.
- If you generate int64 VID via hash, the probability of collision is about 1/10 when there are 1 billion vertices. The number of edges has no concern with the probability of collision.

- 59/1069 - 2023 Vesoft Inc.

### 2.7.4 Define and modify a VID and its data type

The data type of a VID must be defined when you create the graph space. Once defined, it cannot be modified.

A VID is set when you insert a vertex and cannot be modified.

### 2.7.5 Query start vid and global scan

In most cases, the execution plan of query statements in NebulaGraph (MATCH, GO, and LOOKUP) must query the start vid in a certain way.

There are only two ways to locate start vid:

- 1. For example, 60 FROM "player100" OVER explicitly indicates in the statement that start vid is "player100".
- 2. For example, LOOKUP ON player WHERE player.name == "Tony Parker" or MATCH (v:player {name:"Tony Parker"}) locates start vid by the index of the property player.name.



For example, match (n) return n; returns an error: Scan vertices or edges need to specify a limit number, or limit number can not push down., because it is a global scan, you must use the LIMIT clause to limit the number of returns.

Last update: February 19, 2024

- 60/1069 - 2023 Vesoft Inc.

# 2.8 NebulaGraph architecture

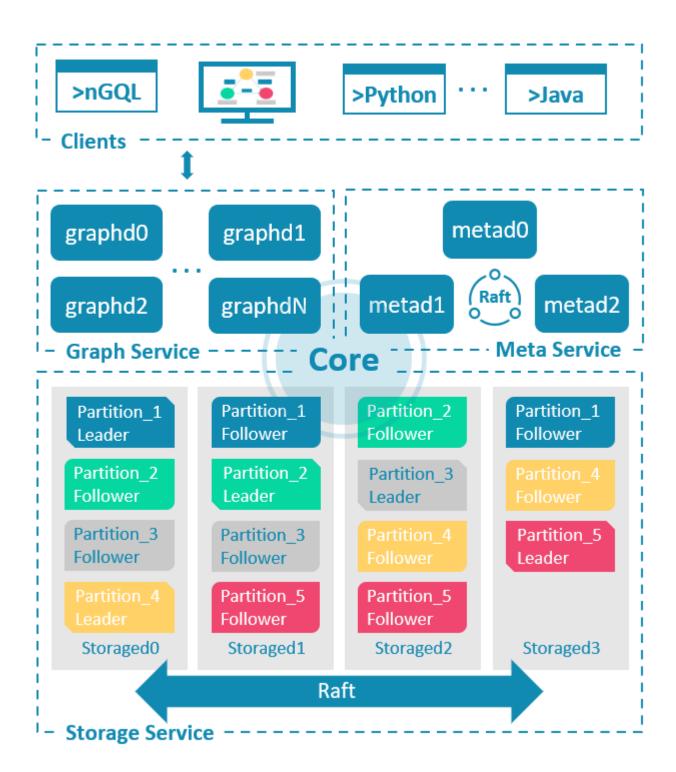
### 2.8.1 Architecture overview

NebulaGraph consists of three services: the Graph Service, the Storage Service, and the Meta Service. It applies the separation of storage and computing architecture.

Each service has its executable binaries and processes launched from the binaries. Users can deploy a NebulaGraph cluster on a single machine or multiple machines using these binaries.

The following figure shows the architecture of a typical Nebula Graph cluster.

- 61/1069 - 2023 Vesoft Inc.



### The Meta Service

The Meta Service in the NebulaGraph architecture is run by the nebula-metad processes. It is responsible for metadata management, such as schema operations, cluster administration, and user privilege management.

For details on the Meta Service, see Meta Service.

- 62/1069 - 2023 Vesoft Inc.

### The Graph Service and the Storage Service

NebulaGraph applies the separation of storage and computing architecture. The Graph Service is responsible for querying. The Storage Service is responsible for storage. They are run by different processes, i.e., nebula-graphd and nebula-storaged. The benefits of the separation of storage and computing architecture are as follows:

### · Great scalability

The separated structure makes both the Graph Service and the Storage Service flexible and easy to scale in or out.

#### · High availability

If part of the Graph Service fails, the data stored by the Storage Service suffers no loss. And if the rest part of the Graph Service is still able to serve the clients, service recovery can be performed quickly, even unfelt by the users.

### • Cost-effective

The separation of storage and computing architecture provides a higher resource utilization rate, and it enables clients to manage the cost flexibly according to business demands.

### • Open to more possibilities

With the ability to run separately, the Graph Service may work with multiple types of storage engines, and the Storage Service may also serve more types of computing engines.

For details on the Graph Service and the Storage Service, see Graph Service and Storage Service.

Last update: February 19, 2024

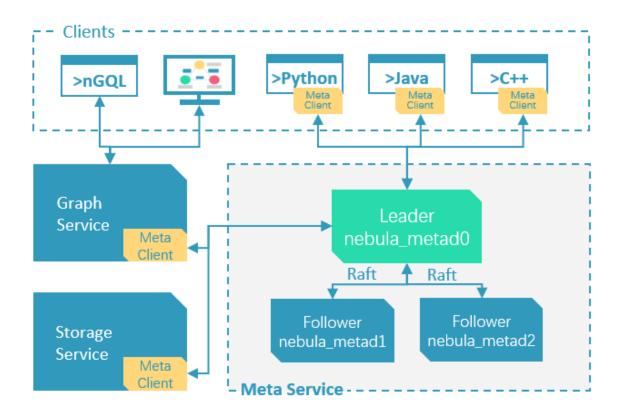
- 63/1069 - 2023 Vesoft Inc.

### 2.8.2 Meta Service

This topic introduces the architecture and functions of the Meta Service.

#### The architecture of the Meta Service

The architecture of the Meta Service is as follows:



The Meta Service is run by nebula-metad processes. Users can deploy nebula-metad processes according to the scenario:

- In a test environment, users can deploy one or three nebula-metad processes on different machines or a single machine.
- In a production environment, we recommend that users deploy three nebula-metad processes on different machines for high availability.

All the nebula-metad processes form a Raft-based cluster, with one process as the leader and the others as the followers.

The leader is elected by the majorities and only the leader can provide service to the clients or other components of NebulaGraph. The followers will be run in a standby way and each has a data replication of the leader. Once the leader fails, one of the followers will be elected as the new leader.



The data of the leader and the followers will keep consistent through Raft. Thus the breakdown and election of the leader will not cause data inconsistency. For more information on Raft, see Storage service architecture.

- 64/1069 - 2023 Vesoft Inc.

#### **Functions of the Meta Service**

#### MANAGES USER ACCOUNTS

The Meta Service stores the information of user accounts and the privileges granted to the accounts. When the clients send queries to the Meta Service through an account, the Meta Service checks the account information and whether the account has the right privileges to execute the queries or not.

For more information on NebulaGraph access control, see Authentication.

#### MANAGES PARTITIONS

The Meta Service stores and manages the locations of the storage partitions and helps balance the partitions.

#### MANAGES GRAPH SPACES

NebulaGraph supports multiple graph spaces. Data stored in different graph spaces are securely isolated. The Meta Service stores the metadata of all graph spaces and tracks the changes of them, such as adding or dropping a graph space.

#### MANAGES SCHEMA INFORMATION

NebulaGraph is a strong-typed graph database. Its schema contains tags (i.e., the vertex types), edge types, tag properties, and edge type properties.

The Meta Service stores the schema information. Besides, it performs the addition, modification, and deletion of the schema, and logs the versions of them.

For more information on NebulaGraph schema, see Data model.

#### MANAGES TTL INFORMATION

The Meta Service stores the definition of TTL (Time to Live) options which are used to control data expiration. The Storage Service takes care of the expiring and evicting processes. For more information, see TTL.

#### MANAGES JOBS

The Job Management module in the Meta Service is responsible for the creation, queuing, querying, and deletion of jobs.

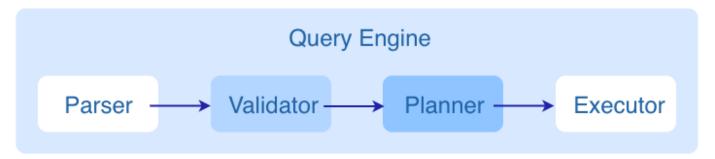
Last update: February 19, 2024

- 65/1069 - 2023 Vesoft Inc.

### 2.8.3 Graph Service

The Graph Service is used to process the query. It has four submodules: Parser, Validator, Planner, and Executor. This topic will describe the Graph Service accordingly.

#### The architecture of the Graph Service



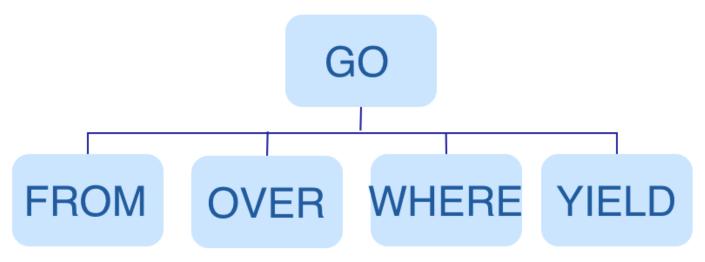
After a query is sent to the Graph Service, it will be processed by the following four submodules:

- 1. Parser: Performs lexical analysis and syntax analysis.
- 2. Validator: Validates the statements.
- 3. **Planner**: Generates and optimizes the execution plans.
- 4. Executor: Executes the plans with operators.

#### Parser

After receiving a request, the statements will be parsed by Parser composed of Flex (lexical analysis tool) and Bison (syntax analysis tool), and its corresponding AST will be generated. Statements will be directly intercepted in this stage because of their invalid syntax.

For example, the structure of the AST of GO FROM "Tim" OVER like WHERE properties(edge).likeness > 8.0 YIELD dst(edge) is shown in the following figure.



- 66/1069 - 2023 Vesoft Inc.

#### Validator

Validator performs a series of validations on the AST. It mainly works on these tasks:

#### · Validating metadata

Validator will validate whether the metadata is correct or not.

When parsing the OVER, WHERE, and YIELD clauses, Validator looks up the Schema and verifies whether the edge type and tag data exist or not. For an INSERT statement, Validator verifies whether the types of the inserted data are the same as the ones defined in the Schema.

#### · Validating contextual reference

Validator will verify whether the cited variable exists or not, or whether the cited property is variable or not.

For composite statements, like \$var = GO FROM "Tim" OVER like YIELD dst(edge) AS ID; GO FROM \$var.ID OVER serve YIELD dst(edge), Validator verifies first to see if var is defined, and then to check if the ID property is attached to the var variable.

#### • Validating type inference

Validator infers what type the result of an expression is and verifies the type against the specified clause.

For example, the WHERE clause requires the result to be a bool value, a NULL value, or empty.

### • Validating the information of \*

Validator needs to verify all the Schema that involves \* when verifying the clause if there is a \* in the statement.

Take a statement like GO FROM "Tim" OVER \* YIELD dst(edge), properties(edge).likeness, dst(edge) as an example. When verifying the OVER clause, Validator needs to verify all the edge types. If the edge type includes like and serve, the statement would be GO FROM "Tim" OVER like, serve YIELD dst(edge), properties(edge).likeness, dst(edge).

### · Validating input and output

Validator will check the consistency of the clauses before and after the ||.

In the statement GO FROM "Tim" OVER like YIELD dst(edge) AS ID | GO FROM \$-.ID OVER serve YIELD dst(edge), Validator will verify whether \$-.ID is defined in the clause before the | .

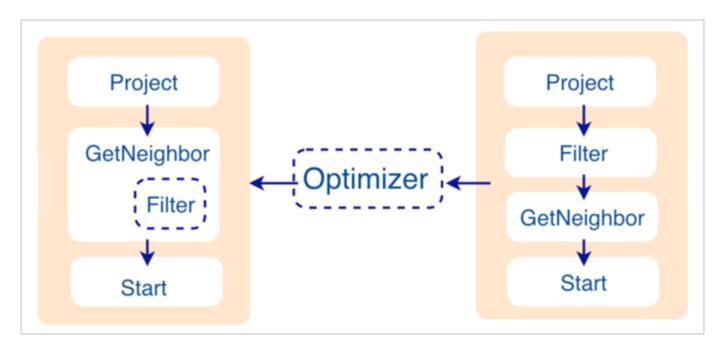
When the validation succeeds, an execution plan will be generated. Its data structure will be stored in the src/planner directory.

#### Planner

In the nebula-graphd.conf file, when enable\_optimizer is set to be false, Planner will not optimize the execution plans generated by Validator. It will be executed by Executor directly.

In the nebula-graphd.conf file, when enable\_optimizer is set to be true, Planner will optimize the execution plans generated by Validator. The structure is as follows.

- 67/1069 - 2023 Vesoft Inc.



#### · Before optimization

In the execution plan on the right side of the preceding figure, each node directly depends on other nodes. For example, the root node Project depends on the Filter node, the Filter node depends on the GetNeighbor node, and so on, up to the leaf node Start. Then the execution plan is (not truly) executed.

During this stage, every node has its input and output variables, which are stored in a hash table. The execution plan is not truly executed, so the value of each key in the associated hash table is empty (except for the Start node, where the input variables hold the starting data), and the hash table is defined in src/context/ExecutionContext.cpp under the nebula-graph repository.

For example, if the hash table is named as ResultMap when creating the Filter node, users can determine that the node takes data from ResultMap["GN1"], then puts the result into ResultMap["Filter2"], and so on. All these work as the input and output of each node.

#### • Process of optimization

The optimization rules that Planner has implemented so far are considered RBO (Rule-Based Optimization), namely the predefined optimization rules. The CBO (Cost-Based Optimization) feature is under development. The optimized code is in the src/optimizer/ directory under the nebula-graph repository.

RBO is a "bottom-up" exploration process. For each rule, the root node of the execution plan (in this case, the Project node) is the entry point, and step by step along with the node dependencies, it reaches the node at the bottom to see if it matches the rule.

As shown in the preceding figure, when the Filter node is explored, it is found that its children node is GetNeighbors, which matches successfully with the pre-defined rules, so a transformation is initiated to integrate the Filter node into the GetNeighbors node, the Filter node is removed, and then the process continues to the next rule. Therefore, when the GetNeighbor operator calls interfaces of the Storage layer to get the neighboring edges of a vertex during the execution stage, the Storage layer will directly filter out the unqualified edges internally. Such optimization greatly reduces the amount of data transfer, which is commonly known as filter pushdown.

#### Executor

The Executor module consists of Scheduler and Executor. The Scheduler generates the corresponding execution operators against the execution plan, starting from the leaf nodes and ending at the root node. The structure is as follows.

- 68/1069 - 2023 Vesoft Inc.



Each node of the execution plan has one execution operator node, whose input and output have been determined in the execution plan. Each operator only needs to get the values for the input variables, compute them, and finally put the results into the corresponding output variables. Therefore, it is only necessary to execute step by step from <code>Start</code>, and the result of the last operator is returned to the user as the final result.

### Source code hierarchy

The source code hierarchy under the nebula-graph repository is as follows.

```
|--src
       --context
                     //contexts for validation and execution
       --executor
                     //execution operators
        --gc
       --optimizer
--planner
                     //optimization rules
//structure of the execution plans
        --scheduler
                     //scheduler
                     //external service management
        --service
        -session
                     //session management
       --stats
                     //monitoring metrics
       --util
                     //basic components
                     //validation of the statements
       --visitor
                     //visitor expression
```

Last update: February 19, 2024

- 69/1069 - 2023 Vesoft Inc.

### 2.8.4 Storage Service

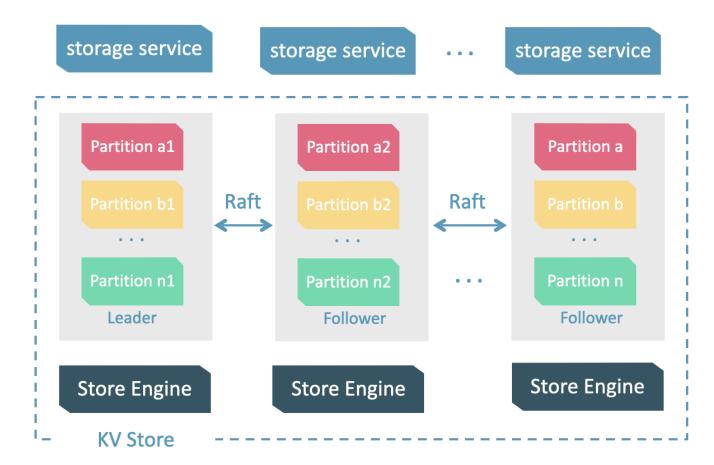
The persistent data of NebulaGraph have two parts. One is the Meta Service that stores the meta-related data.

The other is the Storage Service that stores the data, which is run by the nebula-storaged process. This topic will describe the architecture of the Storage Service.

#### Advantages

- High performance (Customized built-in KVStore)
- Great scalability (Shared-nothing architecture, not rely on NAS/SAN-like devices)
- Strong consistency (Raft)
- High availability (Raft)
- Supports synchronizing with the third party systems, such as Elasticsearch.

### The architecture of the Storage Service



The Storage Service is run by the nebula-storaged process. Users can deploy nebula-storaged processes on different occasions. For example, users can deploy 1 nebula-storaged process in a test environment and deploy 3 nebula-storaged processes in a production environment.

- 70/1069 - 2023 Vesoft Inc.

All the nebula-storaged processes consist of a Raft-based cluster. There are three layers in the Storage Service:

#### · Storage interface

The top layer is the storage interface. It defines a set of APIs that are related to the graph concepts. These API requests will be translated into a set of KV operations targeting the corresponding Partition. For example:

- getNeighbors : queries the in-edge or out-edge of a set of vertices, returns the edges and the corresponding properties, and supports conditional filtering.
- insert vertex/edge: inserts a vertex or edge and its properties.
- getProps : gets the properties of a vertex or an edge.

It is this layer that makes the Storage Service a real graph storage. Otherwise, it is just a KV storage.

### • Consensus

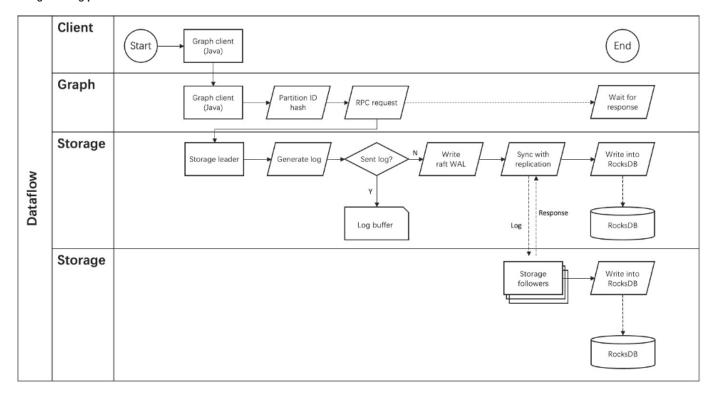
Below the storage interface is the consensus layer that implements Multi Group Raft, which ensures the strong consistency and high availability of the Storage Service.

#### · Store engine

The bottom layer is the local storage engine library, providing operations like get, put, and scan on local disks. The related interfaces are stored in KVStore.h and KVEngine.h files. You can develop your own local store plugins based on your needs.

The following will describe some features of the Storage Service based on the above architecture.

### Storage writing process



- 71/1069 - 2023 Vesoft Inc.

#### **KVStore**

NebulaGraph develops and customizes its built-in KVStore for the following reasons.

- It is a high-performance KVStore.
- It is provided as a (kv) library and can be easily developed for the filter pushdown purpose. As a strong-typed database, how to provide Schema during pushdown is the key to efficiency for NebulaGraph.
- It has strong data consistency.

Therefore, NebulaGraph develops its own KVStore with RocksDB as the local storage engine. The advantages are as follows.

- For multiple local hard disks, NebulaGraph can make full use of its concurrent capacities through deploying multiple data directories.
- The Meta Service manages all the Storage servers. All the partition distribution data and current machine status can be found in the meta service. Accordingly, users can execute a manual load balancing plan in meta service.



NebulaGraph does not support auto load balancing because auto data transfer will affect online business.

- NebulaGraph provides its own WAL mode so one can customize the WAL. Each partition owns its WAL.
- One NebulaGraph KVStore cluster supports multiple graph spaces, and each graph space has its own partition number and replica copies. Different graph spaces are isolated physically from each other in the same cluster.

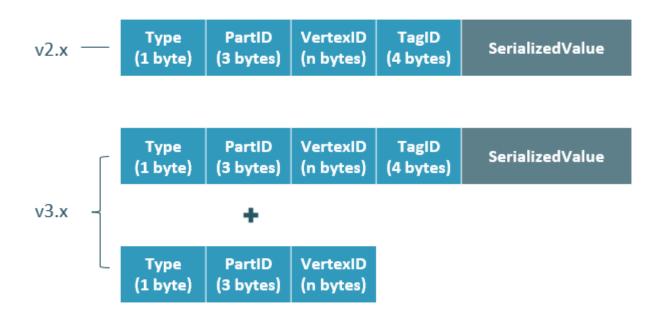
- 72/1069 - 2023 Vesoft Inc.

#### Data storage structure

Graphs consist of vertices and edges. NebulaGraph uses key-value pairs to store vertices, edges, and their properties. Vertices and edges are stored in keys and their properties are stored in values. Such structure enables efficient property filtering.

#### • The storage structure of vertices

Different from NebulaGraph version 2.x, version 3.x added a new key for each vertex. Compared to the old key that still exists, the new key has no TagID field and no value. Vertices in NebulaGraph can now live without tags owing to the new key.



Field	Description
Туре	One byte, used to indicate the key type.
PartID	Three bytes, used to indicate the sharding partition and to scan the partition data based on the prefix when re-balancing the partition.
VertexID	The vertex ID. For an integer VertexID, it occupies eight bytes. However, for a string VertexID, it is changed to <code>fixed_string</code> of a fixed length which needs to be specified by users when they create the space.
TagID	Four bytes, used to indicate the tags that vertex relate with.
SerializedValue	The serialized value of the key. It stores the property information of the vertex.

- 73/1069 - 2023 Vesoft Inc.

## $\bullet$ The storage structure of edges

(1	Type byte)	PartID (3 bytes)	VertexID (n bytes)	EdgeType (4 bytes)	Rank (8 bytes)	VertexID (n bytes)	PlaceHolder (1 byte)	SerializedValue
----	---------------	---------------------	-----------------------	-----------------------	-------------------	-----------------------	-------------------------	-----------------

Field	Description
Туре	One byte, used to indicate the key type.
PartID	Three bytes, used to indicate the partition ID. This field can be used to scan the partition data based on the prefix when re-balancing the partition.
VertexID	Used to indicate vertex ID. The former VID refers to the source VID in the outgoing edge and the dest VID in the incoming edge, while the latter VID refers to the dest VID in the outgoing edge and the source VID in the incoming edge.
Edge Type	Four bytes, used to indicate the edge type. Greater than zero indicates out-edge, less than zero means in-edge.
Rank	Eight bytes, used to indicate multiple edges in one edge type. Users can set the field based on needs and store weight, such as transaction time and transaction number.
PlaceHolder	One byte. Reserved.
SerializedValue	The serialized value of the key. It stores the property information of the edge.

#### PROPERTY DESCRIPTIONS

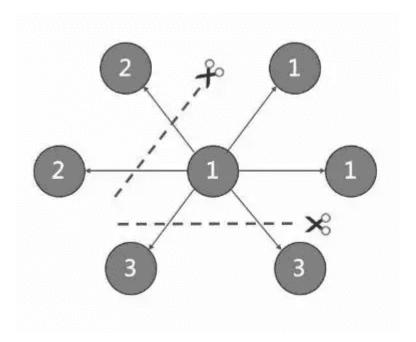
NebulaGraph uses strong-typed Schema.

NebulaGraph will store the properties of vertex and edges in order after encoding them. Since the length of properties is fixed, queries can be made in no time according to offset. Before decoding, NebulaGraph needs to get (and cache) the schema information in the Meta Service. In addition, when encoding properties, NebulaGraph will add the corresponding schema version to support online schema change.

#### Data partitioning

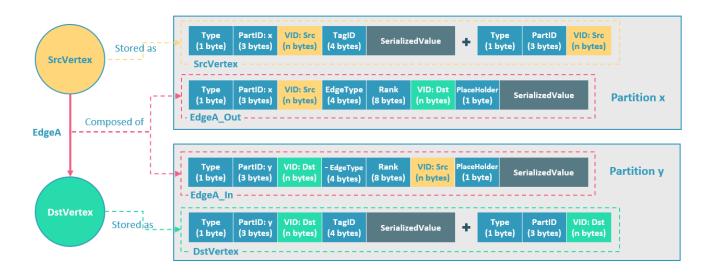
Since in an ultra-large-scale relational network, vertices can be as many as tens to hundreds of billions, and edges are even more than trillions. Even if only vertices and edges are stored, the storage capacity of both exceeds that of ordinary servers. Therefore, NebulaGraph uses hash to shard the graph elements and store them in different partitions.

- 74/1069 - 2023 Vesoft Inc.



## EDGE PARTITIONING AND STORAGE AMPLIFICATION

In NebulaGraph, an edge corresponds to two key-value pairs on the hard disk. When there are lots of edges and each has many properties, storage amplification will be obvious. The storage format of edges is shown in the figure below.



In this example, ScrVertex connects DstVertex via EdgeA, forming the path of (SrcVertex)-[EdgeA]->(DstVertex). ScrVertex, DstVertex, and EdgeA will all be stored in Partition x and Partition y as four key-value pairs in the storage layer. Details are as follows:

- The key value of SrcVertex is stored in Partition x. Key fields include Type, PartID(x), VID(Src), and TagID. SerializedValue, namely Value, refers to serialized vertex properties.
- The first key value of EdgeA, namely EdgeA\_Out, is stored in the same partition as the ScrVertex. Key fields include Type, PartID(x), VID(Src), EdgeType(+ means out-edge), Rank(0), VID(Dst), and PlaceHolder. SerializedValue, namely Value, refers to serialized edge properties.
- The key value of DstVertex is stored in Partition y. Key fields include Type, PartID(y), VID(Dst), and TagID. SerializedValue, namely Value, refers to serialized vertex properties.
- The second key value of EdgeA, namely EdgeA\_In, is stored in the same partition as the DstVertex. Key fields include Type, PartID(y), VID(Dst), EdgeType(- means in-edge), Rank(0), VID(Src), and PlaceHolder. SerializedValue, namely Value, refers to serialized edge properties, which is exactly the same as that in EdgeA Out.

EdgeA\_Out and EdgeA\_In are stored in storage layer with opposite directions, constituting EdgeA logically. EdgeA\_Out is used for traversal requests starting from SrcVertex, such as (a)-[]->(); EdgeA\_In is used for traversal requests starting from DstVertex, such as ()-[]->(a).

Like EdgeA\_Out and EdgeA\_In, NebulaGraph redundantly stores the information of each edge, which doubles the actual capacities needed for edge storage. The key corresponding to the edge occupies a small hard disk space, but the space occupied by Value is proportional to the length and amount of the property value. Therefore, it will occupy a relatively large hard disk space if the property value of the edge is large or there are many edge property values.

#### PARTITION ALGORITHM

NebulaGraph uses a **static Hash** strategy to shard data through a modulo operation on vertex ID. All the out-keys, in-keys, and tag data will be placed in the same partition. In this way, query efficiency is increased dramatically.



The number of partitions needs to be determined when users are creating a graph space since it cannot be changed afterward. Users are supposed to take into consideration the demands of future business when setting it.

When inserting into NebulaGraph, vertices and edges are distributed across different partitions. And the partitions are located on different machines. The number of partitions is set in the CREATE SPACE statement and cannot be changed afterward.

If certain vertices need to be placed on the same partition (i.e., on the same machine), see Formula/code.

The following code will briefly describe the relationship between VID and partition.

```
// If VertexID occupies 8 bytes, it will be stored in int64 to be compatible with the version 1.0.
uint64_t vid = 0;
if (id.size() = 8) {
    memcpy(static_cast<void*>(&vid), id.data(), 8);
} else {
    MurmurHash2 hash;
    vid = hash(id.data());
}
PartitionID pId = vid % numParts + 1;
```

Roughly speaking, after hashing a fixed string to int64, (the hashing of int64 is the number itself), do modulo, and then plus one, namely:

```
pId = vid % numParts + 1;
```

- 76/1069 - 2023 Vesoft Inc.

Parameters and descriptions of the preceding formula are as follows:

Parameter	Description
%	The modulo operation.
numParts	The number of partitions for the graph space where the VID is located, namely the value of partition_num in the CREATE SPACE statement.
pId	The ID for the partition where the VID is located.

Suppose there are 100 partitions, the vertices with VID 1, 101, and 1001 will be stored on the same partition. But, the mapping between the partition ID and the machine address is random. Therefore, we cannot assume that any two partitions are located on the same machine.

#### Raft

#### RAFT IMPLEMENTATION

In a distributed system, one data usually has multiple replicas so that the system can still run normally even if a few copies fail. It requires certain technical means to ensure consistency between replicas.

Basic principle: Raft is designed to ensure consistency between replicas. Raft uses election between replicas, and the (candidate) replica that wins more than half of the votes will become the Leader, providing external services on behalf of all replicas. The rest Followers will play backups. When the Leader fails (due to communication failure, operation and maintenance commands, etc.), the rest Followers will conduct a new round of elections and vote for a new Leader. The Leader and Followers will detect each other's survival through heartbeats and write them to the hard disk in Raft-wal mode. Replicas that do not respond to more than multiple heartbeats will be considered faulty.



Raft-wal needs to be written into the hard disk periodically. If hard disk bottlenecks to write, Raft will fail to send a heartbeat and conduct a new round of elections. If the hard disk IO is severely blocked, there will be no Leader for a long time.

Read and write: For every writing request of the clients, the Leader will initiate a Raft-wal and synchronize it with the Followers. Only after over half replicas have received the Raft-wal will it return to the clients successfully. For every reading request of the clients, it will get to the Leader directly, while Followers will not be involved.

Failure: Scenario 1: Take a (space) cluster of a single replica as an example. If the system has only one replica, the Leader will be itself. If failure happens, the system will be completely unavailable. Scenario 2: Take a (space) cluster of three replicas as an example. If the system has three replicas, one of them will be the Leader and the rest will be the Followers. If the Leader fails, the rest two can still vote for a new Leader (and a Follower), and the system is still available. But if any of the two Followers fails again, the system will be completely unavailable due to inadequate voters.



Raft and HDFS have different modes of duplication. Raft is based on a quorum vote, so the number of replicas cannot be even.

#### MULTI GROUP RAFT

The Storage Service supports a distributed cluster architecture, so NebulaGraph implements Multi Group Raft according to Raft protocol. Each Raft group stores all the replicas of each partition. One replica is the leader, while others are followers. In this way, NebulaGraph achieves strong consistency and high availability. The functions of Raft are as follows.

NebulaGraph uses Multi Group Raft to improve performance when there are many partitions because Raft-wal cannot be NULL. When there are too many partitions, costs will increase, such as storing information in Raft group, WAL files, or batch operation in low load.

- 77/1069 - 2023 Vesoft Inc.

There are two key points to implement the Multi Raft Group:

• To share transport layer

Each Raft Group sends messages to its corresponding peers. So if the transport layer cannot be shared, the connection costs will be very high.

• To share thread pool

Raft Groups share the same thread pool to prevent starting too many threads and a high context switch cost.

#### ВАТСН

For each partition, it is necessary to do a batch to improve throughput when writing the WAL serially. As NebulaGraph uses WAL to implement some special functions, batches need to be grouped, which is a feature of NebulaGraph.

For example, lock-free CAS operations will execute after all the previous WALs are committed. So for a batch, if there are several WALs in CAS type, we need to divide this batch into several smaller groups and make sure they are committed serially.

#### TRANSFER LEADERSHIP

Transfer leadership is extremely important for balance. When moving a partition from one machine to another, NebulaGraph first checks if the source is a leader. If so, it should be moved to another peer. After data migration is completed, it is important to balance leader distribution again.

When a transfer leadership command is committed, the leader will abandon its leadership and the followers will start a leader election.

#### PEER CHANGES

To avoid split-brain, when members in a Raft Group change, an intermediate state is required. In such a state, the quorum of the old group and new group always have an overlap. Thus it prevents the old or new group from making decisions unilaterally. To make it even simpler, in his doctoral thesis Diego Ongaro suggests adding or removing a peer once to ensure the overlap between the quorum of the new group and the old group. NebulaGraph also uses this approach, except that the way to add or remove a member is different. For details, please refer to addPeer/removePeer in the Raft Part class.

#### Differences with HDFS

The Storage Service is a Raft-based distributed architecture, which has certain differences with that of HDFS. For example:

- The Storage Service ensures consistency through Raft. Usually, the number of its replicas is odd to elect a leader. However, DataNode used by HDFS ensures consistency through NameNode, which has no limit on the number of replicas.
- In the Storage Service, only the replicas of the leader can read and write, while in HDFS all the replicas can do so.
- In the Storage Service, the number of replicas needs to be determined when creating a space, since it cannot be changed afterward. But in HDFS, the number of replicas can be changed freely.
- The Storage Service can access the file system directly. While the applications of HDFS (such as HBase) have to access HDFS before the file system, which requires more RPC times.

In a word, the Storage Service is more lightweight with some functions simplified and its architecture is simpler than HDFS, which can effectively improve the read and write performance of a smaller block of data.

Last update: February 19, 2024

- 78/1069 - 2023 Vesoft Inc.

# 3. Quick start

## 3.1 Getting started with NebulaGraph

This topic describes how to use NebulaGraph with Docker Desktop and on-premises deployment workflow to quickly get started with NebulaGraph.

- 79/1069 - 2023 Vesoft Inc.

## 3.1.1 Using NebulaGraph with Docker Desktop

NebulaGraph is available as a Docker Extension that you can easily install and run on your Docker Desktop. You can quickly deploy NebulaGraph using Docker Desktop with just one click.

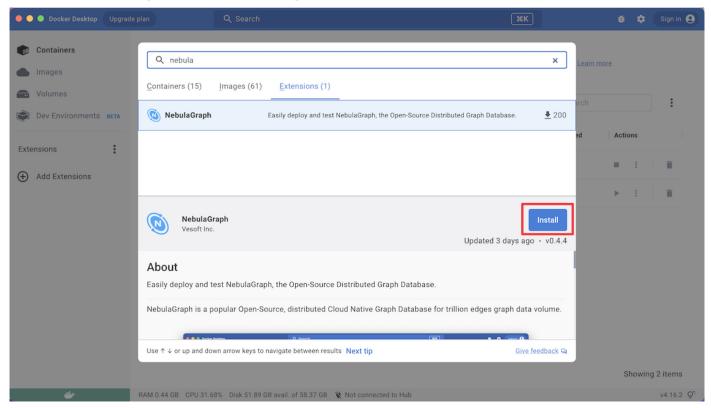
- 80/1069 - 2023 Vesoft Inc.

- 1. Install Docker Desktop
- Install Docker Desktop on Mac
- Install Docker Desktop on Windows

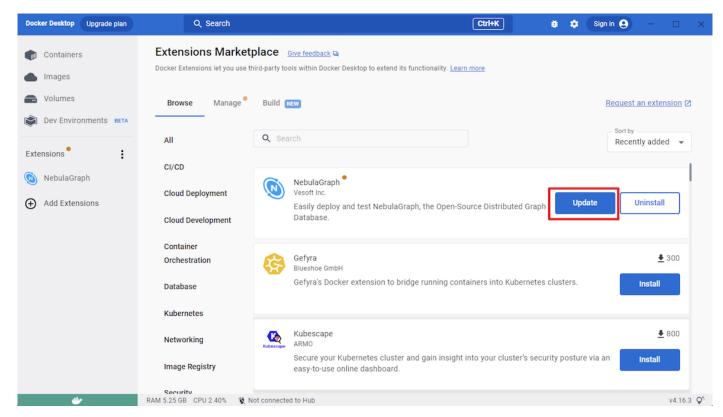


To install Docker Desktop, you need to install WSL 2 first.

- 2. In the left sidebar of Docker Desktop, click  ${\bf Extensions}$  or  ${\bf Add}$   ${\bf Extensions}$ .
- 3. On the Extensions Marketplace, search for NebulaGraph and click Install.



Click **Update** to update NebulaGraph to the latest version when a new version is available.



- 4. Click **Open** to navigate to the NebulaGraph extension page.
- 5. At the top of the page, click **Studio in Browser** to use NebulaGraph.

For more information about how to use NebulaGraph with Docker Desktop, see the following video:

#### 3.1.2 Deploying NebulaGraph on-premises workflow

The following workflow describes how to use NebulaGraph on-premises, including deploying NebulaGraph, connecting to NebulaGraph, and running basic CRUD.

## 1. Deploy NebulaGraph

Users can use the RPM or DEB file to quickly deploy NebulaGraph. For other deployment methods and the corresponding preparations, see the **Deployment and installation** chapter.

## 2. Start NebulaGraph

Users need to start NebulaGraph after deployment.

#### 3. Connect to NebulaGraph

Then users can use clients to connect to NebulaGraph. NebulaGraph supports a variety of clients. This topic will describe how to use NebulaGraph Console to connect to NebulaGraph.

#### 4. Register the Storage Service

When connecting to NebulaGraph for the first time, users must register the Storage Service before querying data.

#### 5. CRUD in NebulaGraph

Users can use nGQL (NebulaGraph Query Language) to run CRUD after connecting to NebulaGraph.

Last update: February 19, 2024

## 3.2 Step 1: Install NebulaGraph

RPM and DEB are common package formats on Linux systems. This topic shows how to quickly install NebulaGraph with the RPM or DEB package.



 $The \ console \ is \ not \ complied \ or \ packaged \ with \ Nebula Graph \ server \ binaries. \ You \ can \ install \ \underline{nebula-console} \ by \ yourself.$ 



For the Enterprise Edition, please contact us.

## 3.2.1 Prerequisites

Wget installed.

- 83/1069 - 2023 Vesoft Inc.

## 3.2.2 Download the package from cloud service



NebulaGraph is currently only supported for installation on Linux systems, and only CentOS 7.x, CentOS 8.x, Ubuntu 16.04, Ubuntu 18.04, and Ubuntu 20.04 operating systems are supported.

• Download the released version.

#### URL:

```
//Centos 6
https://oss-cdn.nebula-graph.io/package/<release_version>/nebula-graph-<release_version>.el6.x86_64.rpm

//Centos 7
https://oss-cdn.nebula-graph.io/package/<release_version>/nebula-graph-<release_version>.el7.x86_64.rpm

//Centos 8
https://oss-cdn.nebula-graph.io/package/<release_version>/nebula-graph-<release_version>.el8.x86_64.rpm

//Ubuntu 1604
https://oss-cdn.nebula-graph.io/package/<release_version>/nebula-graph-<release_version>.ubuntu1604.amd64.deb

//Ubuntu 1804
https://oss-cdn.nebula-graph.io/package/<release_version>/nebula-graph-<release_version>.ubuntu1804.amd64.deb

//Ubuntu 2004
https://oss-cdn.nebula-graph.io/package/<release_version>/nebula-graph-<release_version>.ubuntu2004.amd64.deb
```

### For example, download the release package 3.4.1 for Centos 7.5:

```
wget https://oss-cdn.nebula-graph.io/package/3.4.1/nebula-graph-3.4.1.el7.x86_64.rpm
wget https://oss-cdn.nebula-graph.io/package/3.4.1/nebula-graph-3.4.1.el7.x86_64.rpm.sha256sum.txt
```

#### Download the release package 3.4.1 for Ubuntu 1804:

```
wget https://oss-cdn.nebula-graph.io/package/3.4.1/nebula-graph-3.4.1.ubuntu1804.amd64.deb
wget https://oss-cdn.nebula-graph.io/package/3.4.1/nebula-graph-3.4.1.ubuntu1804.amd64.deb.sha256sum.txt
```

- 84/1069 - 2023 Vesoft Inc.

. Download the nightly version.



- Nightly versions are usually used to test new features. Do not use it in a production environment.
- Nightly versions may not be built successfully every night. And the names may change from day to day.

## URL:

```
//Centos 6
https://oss-cdn.nebula-graph.io/package/nightly/<yyyy.mm.dd>/nebula-graph-<yyyy.mm.dd>-nightly.el6.x86_64.rpm

//Centos 7
https://oss-cdn.nebula-graph.io/package/nightly/<yyyy.mm.dd>/nebula-graph-<yyyy.mm.dd>-nightly.el7.x86_64.rpm

//Centos 8
https://oss-cdn.nebula-graph.io/package/nightly/<yyyy.mm.dd>/nebula-graph-<yyyy.mm.dd>-nightly.el8.x86_64.rpm

//Ubuntu 1604
https://oss-cdn.nebula-graph.io/package/nightly/<yyyy.mm.dd>/nebula-graph-<yyyy.mm.dd>-nightly.ubuntu1604.amd64.deb

//Ubuntu 1804
https://oss-cdn.nebula-graph.io/package/nightly/<yyyy.mm.dd>/nebula-graph-<yyyy.mm.dd>-nightly.ubuntu1804.amd64.deb

//Ubuntu 2004
https://oss-cdn.nebula-graph.io/package/nightly/<yyyy.mm.dd>/nebula-graph-<yyyy.mm.dd>-nightly.ubuntu2004.amd64.deb
```

For example, download the Centos 7.5 package developed and built in 2021.11.28:

```
wget https://oss-cdn.nebula-graph.io/package/nightly/2021.11.28/nebula-graph-2021.11.28-nightly.el7.x86_64.rpm
wget https://oss-cdn.nebula-graph.io/package/nightly/2021.11.28/nebula-graph-2021.11.28-nightly.el7.x86_64.rpm.sha256sum.txt
```

For example, download the Ubuntu 1804 package developed and built in 2021.11.28:

```
wget https://oss-cdn.nebula-graph.io/package/nightly/2021.11.28/nebula-graph-2021.11.28-nightly.ubuntu1804.amd64.deb
wget https://oss-cdn.nebula-graph.io/package/nightly/2021.11.28/nebula-graph-2021.11.28-nightly.ubuntu1804.amd64.deb.sha256sum.txt
```

## 3.2.3 Install NebulaGraph

• Use the following syntax to install with an RPM package.

```
$ sudo rpm -ivh --prefix=<installation_path> <package_name>
```

The option  $\operatorname{\mathsf{--prefix}}$  indicates the installation path. The default path is  $\operatorname{\mathsf{/usr/local/nebula/}}$ .

For example, to install an RPM package in the default path for the 3.4.1 version, run the following command.

- 85/1069 - 2023 Vesoft Inc.

sudo rpm -ivh nebula-graph-3.4.1.el7.x86\_64.rpm

• Use the following syntax to install with a DEB package.

\$ sudo dpkg -i <package\_name>



Customizing the installation path is not supported when installing NebulaGraph with a DEB package. The default installation path is  $\frac{\text{Jusr}}{\text{local/nebula}}$ .

For example, to install a DEB package for the 3.4.1 version, run the following command.

sudo dpkg -i nebula-graph-3.4.1.ubuntu1804.amd64.deb



The default installation path is  $\mbox{\tt /usr/local/nebula/}$  .

#### 3.2.4 Next to do

- (Enterprise Edition)Deploy license
- Start NebulaGraph
- Connect to NebulaGraph

Last update: February 19, 2024

- 86/1069 - 2023 Vesoft Inc.

## 3.3 Step 2: Manage NebulaGraph Service

NebulaGraph supports managing services with scripts.



You can also manage NebulaGraph with systemd in the NebulaGraph Enterprise Edition.



The two methods are incompatible. It is recommended to use only one method in a cluster.

#### 3.3.1 Manage services with script

You can use the nebula.service script to start, stop, restart, terminate, and check the NebulaGraph services.



nebula.service is stored in the /usr/local/nebula/scripts directory by default. If you have customized the path, use the actual path in your environment.

#### **Syntax**

\$ sudo /usr/local/nebula/scripts/nebula.service
[-v] [-c <config\_file\_path>]
<start | stop | restart | kill | status>
<metad | graphd | storaged | all>

Parameter	Description
-V	Display detailed debugging information.
-c	Specify the configuration file path. The default path is $\mbox{\tt /usr/local/nebula/etc/}$ .
start	Start the target services.
stop	Stop the target services.
restart	Restart the target services.
kill	Terminate the target services.
status	Check the status of the target services.
metad	Set the Meta Service as the target service.
graphd	Set the Graph Service as the target service.
storaged	Set the Storage Service as the target service.
all	Set all the NebulaGraph services as the target services.

## 3.3.2 Manage services with systemd

For easy maintenance, NebulaGraph Enterprise Edition supports managing services with systemd. You can start, stop, restart, and check services with systemctl commands.

- 87/1069 - 2023 Vesoft Inc.



- After installing NebulaGraph Enterprise Edition, the .service files required by systemd are located in the etc/unit path in the installation directory. NebulaGraph installed with the RPM/DEB package automatically places the .service files into the path /usr/lib/system/system and the parameter ExecStart is generated based on the specified NebulaGraph installation path, so you can use systemctl commands directly.
- The systematic commands cannot be used to manage the Enterprise Edition cluster that is created with Dashboard of the Enterprise Edition.
- Otherwise, users need to move the .service files manually into the directory /usr/lib/systemd/system, and modify the file path of the parameter ExecStart in the .service files.

### Syntax

\$ systemctl <start | stop | restart | status > <nebula | nebula-metad | nebula-graphd | nebula-storaged>

Parameter	Description
start	Start the target services.
stop	Stop the target services.
restart	Restart the target services.
status	Check the status of the target services.
nebula	Set all the NebulaGraph services as the target services.
nebula-metad	Set the Meta Service as the target service.
nebula-graphd	Set the Graph Service as the target service.
nebula-storaged	Set the Storage Service as the target service.

## 3.3.3 Start NebulaGraph

Run the following command to start Nebula Graph.

```
$ sudo /usr/local/nebula/scripts/nebula.service start all
[INFO] Starting nebula-metad...
[INFO] Done
[INFO] Starting nebula-graphd...
[INFO] Done
[INFO] Starting nebula-storaged...
[INFO] Done
```

Users can also run the following command:

```
$ systemctl start nebula
```

If users want to automatically start NebulaGraph when the machine starts, run the following command:

```
$ systemctl enable nebula
```

## 3.3.4 Stop NebulaGraph



Do not run kill -9 to forcibly terminate the processes. Otherwise, there is a low probability of data loss.

- 88/1069 - 2023 Vesoft Inc.

Run the following command to stop NebulaGraph.

```
$ sudo /usr/local/nebula/scripts/nebula.service stop all
[INFO] Stopping nebula-metad...
[INFO] bone
[INFO] Stopping nebula-graphd...
[INFO] bone
[INFO] Stopping nebula-storaged...
[INFO] Done
```

Users can also run the following command:

```
$ systemctl stop nebula
```

#### 3.3.5 Check the service status

Run the following command to check the service status of NebulaGraph.

```
$ sudo /usr/local/nebula/scripts/nebula.service status all
```

• NebulaGraph is running normally if the following information is returned.

```
INFO] nebula-metad(33fd35e): Running as 29020, Listening on 9559
[INFO] nebula-graphd(33fd35e): Running as 29095, Listening on 9669
[WARN] nebula-storaged after v3.0.0 will not start service until it is added to cluster.
[WARN] See Manage Storage hosts:ADD HOSTS in https://docs.nebula-graph.io/
[INFO] nebula-storaged(33fd35e): Running as 29147, Listening on 9779
```



After starting NebulaGraph, the port of the nebula-storaged process is shown in red. Because the nebula-storaged process waits for the nebula-metad to add the current Storage service during the startup process. The Storage works after it receives the ready signal. Starting from NebulaGraph 3.0.0, the Meta service cannot directly read or write data in the Storage service that you add in the configuration file. The configuration file only registers the Storage service to the Meta service. You must run the ADD HOSTS command to enable the Meta to read and write data in the Storage service. For more information, see Manage Storage hosts.

• If the returned result is similar to the following one, there is a problem. You may also go to the NebulaGraph community for help.

```
[INFO] nebula-metad: Running as 25600, Listening on 9559
[INFO] nebula-graphd: Exited
[INFO] nebula-storaged: Running as 25646, Listening on 9779
```

Users can also run the following command:

```
$ systemctl status nebula

■ nebula.service
Loaded: loaded (/usr/lib/system/nebula.service; disabled; vendor preset: disabled)
Active: active (exited) since — 2022-03-28 04:13:24 UTC; 1h 47min ago
Process: 21772 ExecStart=/usr/local/ent-nightly/scripts/nebula.service start all (code=exited, status=0/SUCCESS)
Main PID: 21772 (code=exited, status=0/SUCCESS)
Tasks: 325
Memory: 424.5M
CGroup: /system.slice/nebula.service
—21789 /usr/local/ent-nightly/bin/nebula-metad --flagfile /usr/local/ent-nightly/etc/nebula-graphd.conf
—21827 /usr/local/ent-nightly/bin/nebula-graphd --flagfile /usr/local/ent-nightly/etc/nebula-storaged.conf
3月 28 04:13:24 xxxxxxx systemd[1]: Started nebula.service.
...
```

The NebulaGraph services consist of the Meta Service, Graph Service, and Storage Service. The configuration files for all three services are stored in the <code>/usr/local/nebula/etc/</code> directory by default. You can check the configuration files according to the returned result to troubleshoot problems.

#### 3.3.6 Next to do

Connect to NebulaGraph

Last update: February 19, 2024

- 90/1069 - 2023 Vesoft Inc.

## 3.4 Step 3: Connect to NebulaGraph

This topic provides basic instruction on how to use the native CLI client NebulaGraph Console to connect to NebulaGraph.



When connecting to NebulaGraph for the first time, you must register the Storage Service before querying data.

NebulaGraph supports multiple types of clients, including a CLI client, a GUI client, and clients developed in popular programming languages. For more information, see the client list.

#### 3.4.1 Prerequisites

- You have started NebulaGraph services.
- The machine on which you plan to run NebulaGraph Console has network access to the Graph Service of NebulaGraph.
- The NebulaGraph Console version is compatible with the NebulaGraph version.



NebulaGraph Console and NebulaGraph of the same version number are the most compatible. There may be compatibility issues when connecting to NebulaGraph with a different version of NebulaGraph Console. The error message incompatible version between client and server is displayed when there is such an issue.

#### Steps

1. On the NebulaGraph Console releases page, select a NebulaGraph Console version and click Assets.



It is recommended to select the **latest** version.

- 2. In the **Assets** area, find the correct binary file for the machine where you want to run NebulaGraph Console and download the file to the machine.
- 3. (Optional) Rename the binary file to  $\ensuremath{\,^{\text{nebula-console}}}$  for convenience.



For Windows, rename the file to  ${\tt nebula-console.exe}$  .

4. On the machine to run NebulaGraph Console, grant the execute permission of the nebula-console binary file to the user.



For Windows, skip this step.

\$ chmod 111 nebula-console

5. In the command line interface, change the working directory to the one where the nebula-console binary file is stored.

- 91/1069 - 2023 Vesoft Inc.

- $_{\rm 6.}$  Run the following command to connect to NebulaGraph.
- For Linux or macOS:

```
$ ./nebula-console -addr <ip> -port <port> -u <username> -p <password>
[-t 120] [-e "nGQL_statement" | -f filename.nGQL]
```

• For Windows:

```
> nebula-console.exe -addr <ip> -port <port> -u <username> -p <password>
[-t 120] [-e "nGQL_statement" | -f filename.nGQL]
```

Parameter descriptions are as follows:

Parameter	Description
-h/-help	Shows the help menu.
-addr/-address	Sets the IP address of the Graph service. The default address is 127.0.0.1.
-P/-port	Sets the port number of the graphd service. The default port number is 9669.
-u/-user	Sets the username of your NebulaGraph account. Before enabling authentication, you can use any existing username. The default username is $\ {\tt root}$ .
-p/-password	Sets the password of your NebulaGraph account. Before enabling authentication, you can use any characters as the password.
-t/-timeout	Sets an integer-type timeout threshold of the connection. The unit is millisecond. The default value is 120.
-e/-eval	Sets a string-type nGQL statement. The nGQL statement is executed once the connection succeeds. The connection stops after the result is returned.
-f/-file	Sets the path of an nGQL file. The nGQL statements in the file are executed once the connection succeeds. The result will be returned and the connection stops then.
-enable_ssl	Enables SSL encryption when connecting to NebulaGraph.
-ssl_root_ca_path	Sets the storage path of the certification authority file.
-ssl_cert_path	Sets the storage path of the certificate file.
ssl_private_key_path	Sets the storage path of the private key file.

For information on more parameters, see the  $\operatorname{project}$  repository.

Last update: February 19, 2024

- 92/1069 - 2023 Vesoft Inc.

## 3.5 Register the Storage Service

When connecting to NebulaGraph for the first time, you have to add the Storage hosts, and confirm that all the hosts are online.

# **P**mpatibility

- Starting from NebulaGraph 3.0.0, you have to run ADD HOSTS before reading or writing data into the Storage Service.
- For NebulaGraph of versions earlier than 3.0.0 and NebulaGraph Cloud clusters, ADD HOSTS is not needed.

### 3.5.1 Prerequisites

You have connected to NebulaGraph.

#### 3.5.2 Steps

1. Add the Storage hosts.

Run the following command to add hosts:

```
ADD HOSTS <ip>:<port> [,<ip>:<port> ...];
```

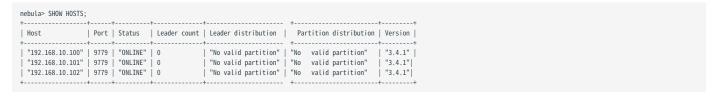
#### Example:

nebula> ADD HOSTS 192.168.10.100:9779, 192.168.10.101:9779, 192.168.10.102:9779;



Make sure that the IP you added is the same as the IP configured for <code>local\_ip</code> in the <code>nebula-storaged.conf</code> file. Otherwise, the Storage service will fail to start. For information about configurations, see Configurations.

2. Check the status of the hosts to make sure that they are all online.



The Status column of the result above shows that all Storage hosts are online.

Last update: February 19, 2024

- 93/1069 - 2023 Vesoft Inc.

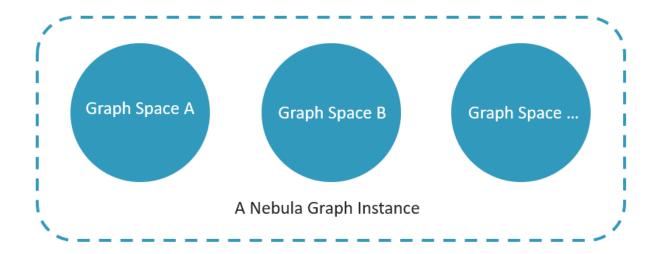
## 3.6 Step 4: Use nGQL (CRUD)

This topic will describe the basic CRUD operations in NebulaGraph.

For more information, see nGQL guide.

## 3.6.1 Graph space and NebulaGraph schema

A NebulaGraph instance consists of one or more graph spaces. Graph spaces are physically isolated from each other. You can use different graph spaces in the same instance to store different datasets.



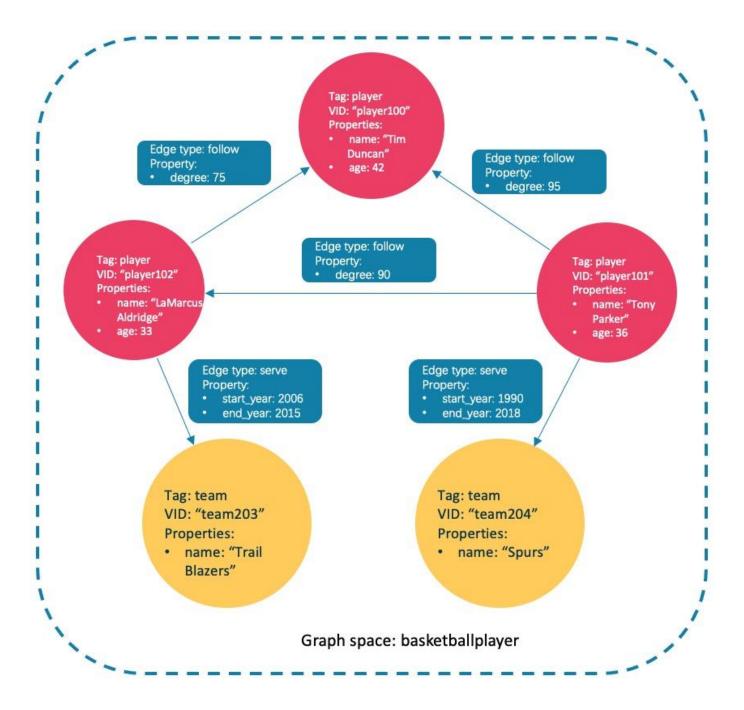
To insert data into a graph space, define a schema for the graph database. NebulaGraph schema is based on the following components.

Schema component	Description
Vertex	Represents an entity in the real world. A vertex can have zero to multiple tags.
Tag	The type of the same group of vertices. It defines a set of properties that describes the types of vertices.
Edge	Represents a <b>directed</b> relationship between two vertices.
Edge type	The type of an edge. It defines a group of properties that describes the types of edges.

For more information, see Data modeling.

In this topic, we will use the following dataset to demonstrate basic CRUD operations.

- 94/1069 - 2023 Vesoft Inc.



#### Async implementation of CREATE and ALTER



In NebulaGraph, the following CREATE or ALTER commands are implemented in an async way and take effect in the **next** heartbeat cycle. Otherwise, an error will be returned. To make sure the follow-up operations work as expected, Wait for two heartbeat cycles, i.e., 20 seconds.

- CREATE SPACE
- CREATE TAG
- CREATE EDGE
- ALTER TAG
- ALTER EDGE
- CREATE TAG INDEX
- CREATE EDGE INDEX



The default heartbeat interval is 10 seconds. To change the heartbeat interval, modify the heartbeat\_interval\_secs parameter in the configuration files for all services.

## 3.6.2 Create and use a graph space

#### nGQL syntax

• Create a graph space:

```
CREATE SPACE [IF NOT EXISTS] <graph_space_name> (
   [partition_num = <partition_number>,]
   [replica_factor = <replica_number>,]
   vid_type = {FIXED_STRING(<N>) | INT64}
   )
   [COMMENT = '<comment>'];
```

For more information on parameters, see CREATE SPACE.

• List graph spaces and check if the creation is successful:

```
nebula> SHOW SPACES;
```

• Use a graph space:

```
USE <graph_space_name>;
```

#### **Examples**

1. Use the following statement to create a graph space named basketballplayer.

```
nebula> CREATE SPACE basketballplayer(partition_num=15, replica_factor=1, vid_type=fixed_string(30));
```



If the system returns the error [ERROR (-1005)]: Host not enough!, check whether registered the Storage Service.

2. Check the partition distribution with SHOW HOSTS to make sure that the partitions are distributed in a balanced way.

nebula> SHOW H		_	_		+	
Host	Port	Status	Leader count	Leader distribution	Partition distribution	Version
"storaged0"   "storaged1"   "storaged2"	9779   9779   9779	"ONLINE"   "ONLINE"   "ONLINE"	5   5   5	"basketballplayer:5"   "basketballplayer:5"   "basketballplayer:5"	"basketballplayer:5"	"3.4.1"    "3.4.1"    "3.4.1"

If the Leader distribution is uneven, use BALANCE LEADER to redistribute the partitions. For more information, see BALANCE.

3. Use the basketballplayer graph space.

```
nebula[(none)]> USE basketballplayer;
```

You can use SHOW SPACES to check the graph space you created.

## 3.6.3 Create tags and edge types

#### nGQL syntax

For more information on parameters, see CREATE TAG and CREATE EDGE.

### **Examples**

Create tags player and team, and edge types follow and serve. Descriptions are as follows.

Component name	Туре	Property
player	Tag	name (string), age (int)
team	Tag	name (string)
follow	Edge type	degree (int)
serve	Edge type	start_year (int), end_year (int)

```
nebula> CREATE TAG player(name string, age int);

nebula> CREATE TAG team(name string);

nebula> CREATE EDGE follow(degree int);

nebula> CREATE EDGE serve(start_year int, end_year int);
```

## 3.6.4 Insert vertices and edges

You can use the INSERT statement to insert vertices or edges based on existing tags or edge types.

#### nGQL syntax

• Insert vertices:

```
INSERT VERTEX [IF NOT EXISTS] [tag_props, [tag_props] ...]
VALUES <vid>: ([prop_value_list])

tag_props:
    tag_name ([prop_name_list])

prop_name_list:
    [prop_name [, prop_name] ...]

prop_value_list:
    [prop_value [, prop_value] ...]
```

vid is short for Vertex ID. A vid must be a unique string value in a graph space. For details, see INSERT VERTEX.

· Insert edges:

```
INSERT EDGE [IF NOT EXISTS] <=edge_type> ( <prop_name_list> ) VALUES
<src_vid> -> <dst_vid>[e<rank>] : ( <prop_value_list> )
[, <src_vid> -> <dst_vid>[e<rank>] : ( <prop_value_list> ), ...];
<prop_name_list> ::=
[ <prop_name> [, <prop_name> ] ...]
<prop_value_list> ::=
[ <prop_value> [, <prop_value> ] ...]
```

For more information on parameters, see INSERT EDGE.

#### **Examples**

• Insert vertices representing basketball players and teams:

```
nebula> INSERT VERTEX player(name, age) VALUES "player100":("Tim Duncan", 42);

nebula> INSERT VERTEX player(name, age) VALUES "player101":("Tony Parker", 36);

nebula> INSERT VERTEX player(name, age) VALUES "player102":("LaMarcus Aldridge", 33);

nebula> INSERT VERTEX team(name) VALUES "team203":("Trail Blazers"), "team204":("Spurs");
```

• Insert edges representing the relations between basketball players and teams:

```
nebula> INSERT EDGE follow(degree) VALUES "player101" -> "player100":(95);

nebula> INSERT EDGE follow(degree) VALUES "player101" -> "player102":(90);

nebula> INSERT EDGE follow(degree) VALUES "player102" -> "player100":(75);

nebula> INSERT EDGE serve(start_year, end_year) VALUES "player101" -> "team204":(1999, 2018),"player102" -> "team203":(2006, 2015);
```

## 3.6.5 Read data

- The GO statement can traverse the database based on specific conditions. A GO traversal starts from one or more vertices, along one or more edges, and returns information in a form specified in the YIELD clause.
- The FETCH statement is used to get properties from vertices or edges.
- The LOOKUP statement is based on indexes. It is used together with the WHERE clause to search for the data that meet the specific conditions.
- The MATCH statement is the most commonly used statement for graph data querying. It can describe all kinds of graph patterns, but it relies on indexes to match data patterns in NebulaGraph. Therefore, its performance still needs optimization.

#### nGQL syntax

• G0

```
GO [[</a> TO] <a href="Told Steel">SETEP | STEEP | STE
```

```
[| ORDER BY <expression> [{ASC | DESC}]]
[| LIMIT [<offset>,] <number_rows>];
```

- FETCH
- Fetch properties on tags:

```
FETCH PROP ON {<tag_name>[, tag_name ...] | *} <vid>[, vid ...]
YIELD <return_list> [AS <alias>];
```

• Fetch properties on edges:

```
FETCH PROP ON <edge_type> <src_vid> -> <dst_vid>[@<rank>] [, <src_vid> -> <dst_vid> ...]
YIELD <output>;
```

LOOKUP

MATCH

```
MATCH <pattern> [<clause_1>] RETURN <output> [<clause_2>];
```

## Examples of 60 statement

 $\bullet$  Search for the players that the player with VID  $\,$  player101 follows.

- 99/1069 - 2023 Vesoft Inc.

```
| "player125" |
+-----+
```

• Filter the players that the player with VID player101 follows whose age is equal to or greater than 35. Rename the corresponding columns in the results with Teammate and Age.

- Search for the players that the player with VID player101 follows. Then retrieve the teams of the players that the player with VID player100 follows. To combine the two queries, use a pipe or a temporary variable.
- With a pipe:

Clause/Sign	Description
\$^	Represents the source vertex of the edge.
Ψ.	A pipe symbol can combine multiple queries.
\$-	Represents the outputs of the query before the pipe symbol.

• With a temporary variable:



Once a composite statement is submitted to the server as a whole, the life cycle of the temporary variables in the statement ends.

#### Example of FETCH statement

Use FETCH: Fetch the properties of the player with VID player100.



The examples of LOOKUP and MATCH statements are in indexes.

## 3.6.6 Update vertices and edges

Users can use the UPDATE or the UPSERT statements to update existing data.

UPSERT is the combination of UPDATE and INSERT. If you update a vertex or an edge with UPSERT, the database will insert a new vertex or edge if it does not exist.



UPSERT operates serially in a partition-based order. Therefore, it is slower than INSERT OR UPDATE. And UPSERT has concurrency only between multiple partitions.

## nGQL syntax

• UPDATE vertices:

```
UPDATE VERTEX <vid> SET <properties to be updated>
[WHEN <condition>] [YIELD <columns>];
```

• UPDATE edges:

```
UPDATE EDGE ON <edge_type> <source vid> -> <destination vid> [@rank]
SET Froperties to be updated> [WHEN <condition>] [YIELD <columns to be output>];
```

• UPSERT vertices or edges:

```
UPSERT {VERTEX <vid> | EDGE <edge_type>} SET <update_columns>
[WHEN <condition>] [YIELD <columns>];
```

#### **Examples**

 $\bullet \ \ \text{UPDATE} \ \ \text{the name} \ \ \text{property of the vertex with VID} \ \ \text{player100} \ \ \text{and check the result with the} \ \ \text{FETCH} \ \ \text{statement.}$ 

```
nebula> UPDATE VERTEX "player100" SET player.name = "Tim";

nebula> FETCH PROP ON player "player100" YIELD properties(vertex);

+-----+
| properties(VERTEX) |
+-----+
```

- 101/1069 - 2023 Vesoft Inc.

```
| {age: 42, name: "Tim"} |
+-----+
```

• UPDATE the degree property of an edge and check the result with the FETCH statement.

```
nebula> UPDATE EDGE ON follow "player101" -> "player100" SET degree = 96;

nebula> FETCH PROP ON follow "player101" -> "player100" YIELD properties(edge);

+-------+
| properties(EDGE) |
+------+
| {degree: 96} |
+-------+
```

• Insert a vertex with VID player111 and UPSERT it.

## 3.6.7 Delete vertices and edges

#### nGQL syntax

• Delete vertices:

```
DELETE VERTEX <vid1>[, <vid2>...]
```

• Delete edges:

```
DELETE EDGE <edge_type> <src_vid> -> <dst_vid>[@<rank>]
[, <src_vid> -> <dst_vid>...]
```

#### **Examples**

• Delete vertices:

```
nebula> DELETE VERTEX "player111", "team203";
```

• Delete edges:

```
nebula> DELETE EDGE follow "player101" -> "team204";
```

#### 3.6.8 About indexes

Users can add indexes to tags and edge types with the CREATE INDEX statement.

## Must-read for using indexes

Both MATCH and LOOKUP statements depend on the indexes. But indexes can dramatically reduce the write performance. **DO NOT** use indexes in production environments unless you are fully aware of their influences on your service.

Users **MUST** rebuild indexes for pre-existing data. Otherwise, the pre-existing data cannot be indexed and therefore cannot be returned in MATCH or LOOKUP statements. For more information, see REBUILD INDEX.

#### nGQL syntax

· Create an index:

```
CREATE {TAG | EDGE} INDEX [IF NOT EXISTS] <index_name>
ON {<tag_name> | <edge_name> | ([<prop_name_List>]) [COMMENT = '<comment>'];
```

· Rebuild an index:

```
REBUILD {TAG | EDGE} INDEX <index_name>;
```



Define the index length when creating an index for a variable-length property. In UTF-8 encoding, a non-ascii character occupies 3 bytes. You should set an appropriate index length according to the variable-length property. For example, the index should be 30 bytes for 10 non-ascii characters. For more information, see CREATE INDEX

#### Examples of LOOKUP and MATCH (index-based)

Make sure there is an index for LOOKUP or MATCH to use. If there is not, create an index first.

Find the information of the vertex with the tag player and its value of the name property is Tony Parker.

This example creates the index player\_index\_1 on the name property.

```
nebula> CREATE TAG INDEX IF NOT EXISTS player_index_1 ON player(name(20));
```

This example rebuilds the index to make sure it takes effect on pre-existing data.

This example uses the LOOKUP statement to retrieve the vertex property.

This example uses the MATCH statement to retrieve the vertex property.

Last update: February 19, 2024

## 3.7 nGQL cheatsheet

## 3.7.1 Functions

• Math functions

Function	Description
double abs(double x)	Returns the absolute value of the argument.
double floor(double x)	Returns the largest integer value smaller than or equal to the argument. (Rounds down)
double ceil(double x)	Returns the smallest integer greater than or equal to the argument. (Rounds up)
double round(double x)	Returns the integer value nearest to the argument. Returns a number farther away from $\boldsymbol{0}$ if the argument is in the middle.
double sqrt(double x)	Returns the square root of the argument.
double cbrt(double x)	Returns the cubic root of the argument.
<pre>double hypot(double x, double y)</pre>	Returns the hypotenuse of a right-angled triangle.
double pow(double x, double y)	Returns the result of $x^y$ .
double exp(double x)	Returns the result of $e^{X}$ .
double exp2(double x)	Returns the result of $2^{X}$ .
double log(double x)	Returns the base-e logarithm of the argument.
double log2(double x)	Returns the base-2 logarithm of the argument.
double log10(double x)	Returns the base-10 logarithm of the argument.
double sin(double x)	Returns the sine of the argument.
double asin(double x)	Returns the inverse sine of the argument.
double cos(double x)	Returns the cosine of the argument.
double acos(double x)	Returns the inverse cosine of the argument.
double tan(double x)	Returns the tangent of the argument.
double atan(double x)	Returns the inverse tangent of the argument.
double rand()	Returns a random floating point number in the range from 0 (inclusive) to 1 (exclusive); i.e. $[0,1)$ .
int rand32(int min, int max)	Returns a random 32-bit integer in [min, max).  If you set only one argument, it is parsed as max and min is 0 by default.  If you set no argument, the system returns a random signed 32-bit integer.
int rand64(int min, int max)	Returns a random 64-bit integer in [min, max).  If you set only one argument, it is parsed as max and min is 0 by default.  If you set no argument, the system returns a random signed 64-bit integer.
bit_and()	Bitwise AND.
bit_or()	Bitwise OR.
bit_xor()	Bitwise XOR.
int size()	Returns the number of elements in a list or a map or the length of a string.
int range(int start, int end, int step)	Returns a list of integers from [start,end] in the specified steps. step is 1 by default.
int sign(double x)	Returns the signum of the given number.  If the number is 0, the system returns 0.  If the number is negative, the system returns -1.  If the number is positive, the system returns 1.

- 106/1069 - 2023 Vesoft Inc.

Function	Description
double e()	Returns the base of the natural logarithm, e (2.718281828459045).
double pi()	Returns the mathematical constant pi (3.141592653589793).
double radians()	Converts degrees to radians. radians(180) returns 3.141592653589793.

## • Aggregating functions

Function	Description
avg()	Returns the average value of the argument.
count()	Syntax: count({expr   *}) .  count() returns the number of rows (including NULL).  count(expr) returns the number of non-NULL values that meet the expression.  count() and size() are different.
max()	Returns the maximum value.
min()	Returns the minimum value.
collect()	The collect() function returns a list containing the values returned by an expression. Using this function aggregates data by merging multiple records or values into a single list.
std()	Returns the population standard deviation.
sum()	Returns the sum value.

- 107/1069 - 2023 Vesoft Inc.

## • String functions

Function	Description
int strcasecmp(string a, string b)	Compares string $a$ and $b$ without case sensitivity. When $a=b$ , the return
string lower(string a)	Returns the argument in lowercase.
string toLower(string a)	The same as lower().
string upper(string a)	Returns the argument in uppercase.
string toUpper(string a)	The same as upper().
int length(a)	Returns the length of the given string in bytes or the length of a path in hops.
string trim(string a)	Removes leading and trailing spaces.
string ltrim(string a)	Removes leading spaces.
string rtrim(string a)	Removes trailing spaces.
string left(string a, int count)	Returns a substring consisting of count characters from the left side of
string right(string a, int count)	Returns a substring consisting of count characters from the right side of
string lpad(string a, int size, string letters)	Left-pads string a with string letters and returns a
string rpad(string a, int size, string letters)	Right-pads string a with string letters and returns a
string substr(string a, int pos, int count)	Returns a substring extracting count characters starting from
string substring(string a, int pos, int count)	The same as substr().
string reverse(string)	Returns a string in reverse order.
string replace(string a, string b, string c)	Replaces string b in string a with string c.
list split(string a, string b)	Splits string a at string b and returns a list of strings.
concat()	The concat() function requires at least two or more strings. All the parameters are concatenated into one string.  Syntax: concat(string1,string2,)
concat_ws()	The $concat\_ws()$ function connects two or more strings with a predefined separator.
extract()	extract() uses regular expression matching to retrieve a single substring or all substrings from a string.
json_extract()	The ${\sf json\_extract}()$ function converts the specified JSON string to map.

## • Data and time functions

Function	Description
int now()	Returns the current timestamp of the system.
timestamp timestamp()	Returns the current timestamp of the system.
date date()	Returns the current UTC date based on the current system.
time time()	Returns the current UTC time based on the current system.
datetime datetime()	Returns the current UTC date and time based on the current system.

- 108/1069 - 2023 Vesoft Inc.

## • Schema-related functions

## • For nGQL statements

Function	Description
id(vertex)	Returns the ID of a vertex. The data type of the result is the same as the vertex ID.
map properties(vertex)	Returns the properties of a vertex.
map properties(edge)	Returns the properties of an edge.
string type(edge)	Returns the edge type of an edge.
src(edge)	Returns the source vertex ID of an edge. The data type of the result is the same as the vertex ID.
dst(edge)	Returns the destination vertex ID of an edge. The data type of the result is the same as the vertex ID.
int rank(edge)	Returns the rank value of an edge.
vertex	Returns the information of vertices, including VIDs, tags, properties, and values.
edge	Returns the information of edges, including edge types, source vertices, destination vertices, ranks, properties, and values.
vertices	Returns the information of vertices in a subgraph. For more information, see GET SUBGRAPH.
edges	Returns the information of edges in a subgraph. For more information, see GET SUBGRAPH.
path	Returns the information of a path. For more information, see FIND PATH.

#### $\bullet$ For statements compatible with open Cypher

Function	Description
id( <vertex>)</vertex>	Returns the ID of a vertex. The data type of the result is the same as the vertex ID.
list tags( <vertex>)</vertex>	Returns the Tag of a vertex, which serves the same purpose as labels().
list labels( <vertex>)</vertex>	Returns the Tag of a vertex, which serves the same purpose as tags(). This function is used for compatibility with openCypher syntax.
map properties( <vertex_or_edge>)</vertex_or_edge>	Returns the properties of a vertex or an edge.
string type( <edge>)</edge>	Returns the edge type of an edge.
src( <edge>)</edge>	Returns the source vertex ID of an edge. The data type of the result is the same as the vertex ID.
dst( <edge>)</edge>	Returns the destination vertex ID of an edge. The data type of the result is the same as the vertex ID.
vertex startNode( <path>)</path>	Visits an edge or a path and returns its source vertex ID.
string endNode( <path>)</path>	Visits an edge or a path and returns its destination vertex ID.
int rank( <edge>)</edge>	Returns the rank value of an edge.

- 109/1069 - 2023 Vesoft Inc.

## • List functions

Function	Description
keys(expr)	Returns a list containing the string representations for all the property names of vertices, edges, or maps.
labels(vertex)	Returns the list containing all the tags of a vertex.
nodes(path)	Returns the list containing all the vertices in a path.
<pre>range(start, end [, step])</pre>	Returns the list containing all the fixed-length steps in $\[$ start,end $\]$ . step is 1 by default.
relationships(path)	Returns the list containing all the relationships in a path.
reverse(list)	Returns the list reversing the order of all elements in the original list.
tail(list)	Returns all the elements of the original list, excluding the first one.
head(list)	Returns the first element of a list.
last(list)	Returns the last element of a list.
reduce()	The <code>reduce()</code> function applies an expression to each element in a list one by one, chains the result to the next iteration by taking it as the initial value, and returns the final result.

## • Type conversion functions

Function	Description
bool toBoolean()	Converts a string value to a boolean value.
float toFloat()	Converts an integer or string value to a floating point number.
string toString()	Converts non-compound types of data, such as numbers, booleans, and so on, to strings.
int toInteger()	Converts a floating point or string value to an integer value.
set toSet()	Converts a list or set value to a set value.
int hash()	The hash() function returns the hash value of the argument. The argument can be a number, a string, a list, a boolean, null, or an expression that evaluates to a value of the preceding data types.

## • Predicate functions

 $Predicate \ functions \ return \ \ true \ \ or \ \ false \ . \ They \ are \ most \ commonly \ used \ in \ \ \ \ \ \ \ \ \ \ clauses.$ 

Function	Description
exists()	Returns $\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
any()	Returns $\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
all()	Returns $\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
none()	Returns $\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
single()	Returns $\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$

- 110/1069 - 2023 Vesoft Inc.

## • Conditional expressions functions

Function	Description
CASE	The CASE expression uses conditions to filter the result of an nGQL query statement. It is usually used in the YIELD and RETURN clauses. The CASE expression will traverse all the conditions. When the first condition is met, the CASE expression stops reading the conditions and returns the result. If no conditions are met, it returns the result in the ELSE clause. If there is no ELSE clause and no conditions are met, it returns NULL.
coalesce()	Returns the first not null value in all expressions.

# 3.7.2 General queries statements

• MATCH

MATCH <pattern> [<clause\_1>] RETURN <output> [<clause\_2>];

Pattern	Example	Description
Match vertices	(v)	You can use a user-defined variable in a pair of parentheses to represent a vertex in a pattern. For example: $(v)$ .
Match tags	MATCH (v:player) RETURN v	You can specify a tag with $:$ after the vertex in a pattern.
Match multiple tags	MATCH (v:player:team) RETURN v LIMIT 10	To match vertices with multiple tags, use colons (:).
Match vertex properties	MATCH (v:player{name:"Tim Duncan"})  RETURN v  MATCH (v) WITH v, properties(v) as props, keys(properties(v)) as kk LIMIT	You can specify a vertex property with { <pre>sprop_name&gt;:</pre> <pre><pre>sprop_value&gt;} after the tag in a pattern; or use a vertex property value to get vertices directly.</pre></pre>
	10000 WHERE [i in kk where props[i] == "Tim Duncan"] RETURN v	
Match a VID.	MATCH (v) WHERE $id(v) == 'player101'$ RETURN v	You can use the VID to match a vertex. The <code>id()</code> function can retrieve the VID of a vertex.
Match multiple VIDs.	MATCH (v:player { name: 'Tim  Duncan' })(v2) WHERE id(v2) IN  ["player101", "player102"] RETURN v2	To match multiple VIDs, use $\mbox{\sc WHERE}$ $\mbox{\sc id}(\mbox{\sc v})$ IN $[\mbox{\sc vid\_List}]$ .
Match connected vertices	MATCH (v:player{name:"Tim Duncan"}) (v2) RETURN v2.player.name AS Name	You can use the symbol to represent edges of both directions and match vertices connected by these edges.  You can add a > or < to the symbol to specify the direction of an edge.
Match paths	MATCH p=(v:player{name:"Tim Duncan"})>(v2) RETURN p	Connected vertices and edges form a path. You can use a user-defined variable to name a path as follows.
Match edges	MATCH (v:player{name:"Tim Duncan"})-[e]- (v2) RETURN e MATCH ()<-[e]-() RETURN e LIMIT 3	Besides using,>, or < to indicate a nameless edge, you can use a user-defined variable in a pair of square brackets to represent a named edge. For example: -[e]
Match an edge type	MATCH ()-[e:follow]-() RETURN e LIMIT 5	Just like vertices, you can specify an edge type with : <edge_type> in a pattern. For example: -[e:follow]</edge_type>
Match edge type properties	MATCH (v:player{name:"Tim Duncan"})- [e:follow{degree:95}]->(v2) RETURN e  MATCH ()-[e]->() WITH e, properties(e) as props, keys(properties(e)) as kk  LIMIT 10000 WHERE [i in kk where props[i] == 90] RETURN e	You can specify edge type properties with { <pre>sprop_value&gt;} in a pattern. For example: [e:follow{likeness:95}]; or use an edge type property value to get edges directly.</pre>
Match multiple edge types	MATCH (v:player{name:"Tim Duncan"})- [e:follow   :serve]->(v2) RETURN e	The   symbol can help matching multiple edge types. For example: [e:follow :serve]. The English colon (:) before the first edge type cannot be omitted, but the English colon before the subsequent edge type can be omitted, such as [e:follow serve].
Match multiple edges	MATCH (v:player{name:"Tim Duncan"})-[]->(v2)<-[e:serve]-(v3) RETURN v2, v3	You can extend a pattern to match multiple edges in a path.
Match fixed- length paths	MATCH p=(v:player{name:"Tim Duncan"})- [e:follow*2]->(v2) RETURN DISTINCT v2 AS Friends	You can use the : <edge_type>*<hop> pattern to match a fixed-length path. hop must be a non-negative integer. The data type of e is the list.</hop></edge_type>
Match variable- length paths	MATCH p=(v:player{name:"Tim Duncan"})- [e:follow*13]->(v2) RETURN v2 AS Friends	minHop: Optional. It represents the minimum length of the path. minHop: must be a non-negative integer. The default value is 1.

Pattern	Example	Description  minHop and maxHop are optional and the default value is 1 and infinity respectively. The data type of e is the list.
Match variable- length paths with multiple edge types	MATCH p=(v:player{name:"Tim Duncan"})- [e:follow   serve*2]->(v2) RETURN DISTINCT v2	You can specify multiple edge types in a fixed-length or variable-length pattern. In this case, hop, minHop, and maxHop take effect on all edge types. The data type of $\mathrm{e}$ is the list.
Retrieve vertex or edge information	MATCH (v:player{name:"Tim Duncan"})  RETURN v  MATCH (v:player{name:"Tim Duncan"})-[e]- >(v2) RETURN e	Use RETURN {-vertex_name> $ $ -edge_name>} to retrieve all the information of a vertex or an edge.
Retrieve VIDs	MATCH (v:player{name:"Tim Duncan"}) RETURN id(v)	Use the id() function to retrieve VIDs.
Retrieve tags	MATCH (v:player{name:"Tim Duncan"}) RETURN labels(v)	Use the labels() function to retrieve the list of tags on a vertex. To retrieve the nth element in the labels(v) list, use labels(v) $[n-1]$ .
Retrieve a single property on a vertex or an edge	MATCH (v:player{name:"Tim Duncan"}) RETURN v.player.age	Use RETURN { <vertex_name>   <edge_name>}.<property> to retrieve a single property.  Use AS to specify an alias for a property.</property></edge_name></vertex_name>
Retrieve all properties on a vertex or an edge	<pre>MATCH p=(v:player{name:"Tim Duncan"})- []-&gt;(v2) RETURN properties(v2)</pre>	Use the properties() function to retrieve all properties on a vertex or an edge.
Retrieve edge types	<pre>MATCH p=(v:player{name:"Tim Duncan"})- [e]-&gt;() RETURN DISTINCT type(e)</pre>	Use the type() function to retrieve the matched edge types.
Retrieve paths	MATCH p=(v:player{name:"Tim Duncan"})- [*3]->() RETURN p	Use ${\tt RETURN \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
Retrieve vertices in a path	MATCH p=(v:player{name:"Tim Duncan"})- []->(v2) RETURN nodes(p)	Use the <code>nodes()</code> function to retrieve all vertices in a path.
Retrieve edges in a path	<pre>MATCH p=(v:player{name:"Tim Duncan"})- []-&gt;(v2) RETURN relationships(p)</pre>	Use the relationships() function to retrieve all edges in a path.
Retrieve path length	MATCH p=(v:player{name:"Tim Duncan"})- [*2]->(v2) RETURN p AS Paths, length(p) AS Length	Use the <code>length()</code> function to retrieve the length of a path.

## • OPTIONAL MATCH

Pattern	Example	Description
Matches patterns against your graph database, just like MATCH does.	MATCH (m)-[]->(n) WHERE $id(m)==$ "player100"  OPTIONAL MATCH (n)-[]->( $L$ ) RETURN $id(m)$ , $id(n)$ , $id(L)$	If no matches are found, OPTIONAL MATCH will use a null for missing parts of the pattern.

- 114/1069 - 2023 Vesoft Inc.

## • LOOKUP

```
LOOKUP ON {<vertex_tag> | <edge_type>}
[WHERE <expression> [AND <expression> ...]]
YIELD <return_List> [AS <alias>]
```

Pattern	Example	Description
Retrieve vertices	LOOKUP ON player WHERE player.name == "Tony Parker" YIELD player.name AS name, player.age AS age	The following example returns vertices whose $$\operatorname{\textsc{name}}$$ is Tony Parker and the tag is player .
Retrieve edges	LOOKUP ON follow WHERE follow.degree == 90 YIELD follow.degree	Returns edges whose $\ensuremath{degree}$ is 90 and the edge type is $\ensuremath{follow}$ .
List vertices with a tag	LOOKUP ON player YIELD properties(vertex),id(vertex)	Shows how to retrieve the VID of all vertices tagged with $\ensuremath{player}$ .
List edges with an edge types	LOOKUP ON like YIELD edge AS e	Shows how to retrieve the source Vertex IDs, destination vertex IDs, and ranks of all edges of the like edge type.
Count the numbers of vertices or edges	LOOKUP ON player YIELD id(vertex)  YIELD COUNT(*) AS Player_Count	Shows how to count the number of vertices tagged with $\ensuremath{\text{player}}$ .
Count the numbers of edges	LOOKUP ON like YIELD id(vertex)  YIELD COUNT(*) AS Like_Count	Shows how to count the number of edges of the like edge type.

## • GO

```
GO [[<h>TO] <h> {STEP|STEPS} ] FROM <vertex_List>

OVER <edge_type_List> {{REVERSELY | BIDIRECT}}

[ WHERE <conditions> ]

YIELD [DISTINCT] <return_List>
[{SAMPLE <sample_List> | LIMIT <limit_List>}]

[| GROUP BY {col_name | expr | position} YIELD <col_name>]

[| ORDER BY <expression> [{ASC | DESC}]]

[| LIMIT [<offset_value>,] <number_rows>]
```

Example	Description
GO FROM "player102" OVER serve YIELD dst(edge)	Returns the teams that player 102 serves.
GO 2 STEPS FROM "player102" OVER follow YIELD dst(edge)	Returns the friends of player 102 with 2 hops.
GO FROM "player100", "player102" OVER serve WHERE properties(edge).start_year > 1995 YIELD DISTINCT properties(\$\$).name AS team_name, properties(edge).start_year AS start_year, properties(\$^).name AS player_name	Adds a filter for the traversal.
GO FROM "player100" OVER follow, serve YIELD properties(edge).degree, properties(edge).start_year	The following example traverses along with multiple edge types. If there is no value for a property, the output is <code>NULL</code> .
GO FROM "player100" OVER follow REVERSELY YIELD src(edge) AS destination	The following example returns the neighbor vertices in the incoming direction of player 100.
GO FROM "player100" OVER follow REVERSELY YIELD src(edge) AS id   GO FROM \$id OVER serve WHERE properties(\$^).age > 20 YIELD properties(\$^).name AS FriendOf, properties(\$ \$).name AS Team	The following example retrieves the friends of player 100 and the teams that they serve.
GO FROM "player102" OVER follow YIELD dst(edge) AS both	The following example returns all the neighbor vertices of player 102.
GO 2 STEPS FROM "player100" OVER follow YIELD src(edge) AS src, dst(edge) AS dst, properties(\$\$).age AS age   GROUP BY \$dst YIELD \$dst AS dst, collect_set(\$src) AS src, collect(\$age) AS age	The following example the outputs according to age.

## • FETCH

## • Fetch vertex properties

```
FETCH PROP ON {<tag_name>[, tag_name ...] | *} <vid>[, vid ...]
YIELD <return_List> [AS <alias>]
```

Example	Description
FETCH PROP ON player "player100" YIELD properties(vertex)	Specify a tag in the $\ensuremath{FETCH}$ statement to fetch the vertex properties by that tag.
FETCH PROP ON player "player100" YIELD player.name AS name	Use a YIELD clause to specify the properties to be returned.
FETCH PROP ON player "player101", "player102", "player103" YIELD properties(vertex)	Specify multiple VIDs (vertex IDs) to fetch properties of multiple vertices. Separate the VIDs with commas.
FETCH PROP ON player, t1 "player100", "player103" YIELD properties(vertex)	Specify multiple tags in the FETCH statement to fetch the vertex properties by the tags. Separate the tags with commas.
FETCH PROP ON * "player100", "player106", "team200" YIELD properties(vertex)	Set an asterisk symbol $^\star$ to fetch properties by all tags in the current graph space.

## • Fetch edge properties

FETCH PROP ON <code>sedge\_type></code> <code>src\_vid></code> -> <code>sdst\_vid>[@srank>]</code> [, <code>src\_vid></code> -> <code>sdst\_vid></code> ...] <code>YIELD soutput>;</code>

Example	Description
FETCH PROP ON serve "player100" -> "team204" YIELD properties(edge)	The following statement fetches all the properties of the serve edge that connects vertex "player100" and vertex "team204".
FETCH PROP ON serve "player100" -> "team204" YIELD serve.start_year	Use a YIELD clause to fetch specific properties of an edge.
FETCH PROP ON serve "player100" -> "team204", "player133" -> "team202" YIELD properties(edge)	Specify multiple edge patterns ( $<$ src_vid> -> $<$ dst_vid>[@ <rank>] ) to fetch properties of multiple edges. Separate the edge patterns with commas.</rank>
FETCH PROP ON serve "player100" -> "team204"@1 YIELD properties(edge)	To fetch on an edge whose rank is not 0, set its rank in the FETCH statement.
GO FROM "player101" OVER follow YIELD followsrc AS s, followdst AS d   FETCH PROP ON follow \$s -> \$d YIELD follow.degree	The following statement returns the degree values of the follow edges that start from vertex "player101" .
<pre>\$var = GO FROM "player101" OVER follow YIELD followsrc AS s, followdst AS d; FETCH PROP ON follow \$var.s -&gt; \$var.d YIELD follow.degree</pre>	You can use user-defined variables to construct similar queries.

## • SHOW

Statement	Syntax	Example	Description
SHOW CHARSET	SHOW CHARSET	SHOW CHARSET	Shows the available character sets.
SHOW COLLATION	SHOW COLLATION	SHOW COLLATION	Shows the collations supported by NebulaGraph.
SHOW CREATE SPACE	SHOW CREATE SPACE <space_name></space_name>	SHOW CREATE SPACE basketballplayer	Shows the creating statement of the specified graph space.
SHOW CREATE TAG/EDGE	SHOW CREATE {TAG <tag_name>   EDGE <edge_name>}</edge_name></tag_name>	SHOW CREATE TAG player	Shows the basic information of the specified tag.
SHOW HOSTS	SHOW HOSTS [GRAPH   STORAGE   META]	SHOW HOSTS SHOW HOSTS GRAPH	Shows the host and version information of Graph Service, Storage Service, and Meta Service.
SHOW INDEX STATUS	SHOW {TAG   EDGE} INDEX STATUS	SHOW TAG INDEX STATUS	Shows the status of jobs that rebuild native indexes, which helps check whether a native index is successfully rebuilt or not.
SHOW INDEXES	SHOW {TAG   EDGE} INDEXES	SHOW TAG INDEXES	Shows the names of existing native indexes.
SHOW PARTS	SHOW PARTS [ <part_id>]</part_id>	SHOW PARTS	Shows the information of a specified partition or all partitions in a graph space.
SHOW ROLES	SHOW ROLES IN <space_name></space_name>	SHOW ROLES in basketballplayer	Shows the roles that are assigned to a user account.
SHOW SNAPSHOTS	SHOW SNAPSHOTS	SHOW SNAPSHOTS	Shows the information of all the snapshots.
SHOW SPACES	SHOW SPACES	SHOW SPACES	Shows existing graph spaces in NebulaGraph.
SHOW STATS	SHOW STATS	SHOW STATS	Shows the statistics of the graph space collected by the latest STATS job.
SHOW TAGS/ EDGES	SHOW TAGS   EDGES	SHOW TAGS, SHOW EDGES	Shows all the tags in the current graph space.
SHOW USERS	SHOW USERS	SHOW USERS	Shows the user information.
SHOW SESSIONS	SHOW SESSIONS	SHOW SESSIONS	Shows the information of all the sessions.
SHOW SESSIONS	SHOW SESSION <session_id></session_id>	SHOW SESSION 1623304491050858	Shows a specified session with its ID.
SHOW QUERIES	SHOW [ALL] QUERIES	SHOW QUERIES	Shows the information of working queries in the current session.
	SHOW META LEADER	SHOW META LEADER	Shows the information of the leader in the

- 117/1069 - 2023 Vesoft Inc.

# 3.7.3 Clauses and options

Clause	Syntax	Example	Description
GROUP BY	GROUP BY <var> YIELD <var>, <aggregation_function(var)></aggregation_function(var)></var></var>	GO FROM "player100" OVER follow BIDIRECT YIELD \$\$.player.name as Name   GROUP BY \$Name YIELD \$Name as Player, count(*) AS Name_Count	Finds all the vertices connected directly to vertex "player100", groups the result set by player names, and counts how many times the name shows up in the result set.
LIMIT	YIELD <var> [  LIMIT [<offset_value>,] <number_rows>]</number_rows></offset_value></var>	GO FROM "player100" OVER follow REVERSELY YIELD \$\$.player.name AS Friend, \$\$.player.age AS Age   ORDER BY \$Age, \$Friend   LIMIT 1, 3	Returns the 3 rows of data starting from the second row of the sorted output.
SKIP	RETURN <var> [SKIP <offset>] [LIMIT <number_rows>]</number_rows></offset></var>	MATCH (v:player{name:"Tim Duncan"})> (v2) RETURN v2.player.name AS Name, v2.player.age AS Age ORDER BY Age DESC SKIP 1	SKIP can be used alone to set the offset and return the data after the specified position.
SAMPLE	<pre><go_statement> SAMPLE <sample_list>;</sample_list></go_statement></pre>	GO 3 STEPS FROM "player100" OVER * YIELD properties(\$\$).name AS NAME, properties(\$\$).age AS Age SAMPLE [1,2,3];	Takes samples evenly in the result set and returns the specified amount of data.
ORDER BY	<pre><yield clause=""> ORDER BY <expression> [ASC   DESC] [, <expression> [ASC   DESC]]</expression></expression></yield></pre>	FETCH PROP ON player "player100", "player101", "player102", "player103" YIELD player.age AS age, player.name AS name   ORDER BY \$age ASC, \$name DESC	The ORDER BY clause specifies the order of the rows in the output.
RETURN	RETURN { <vertex_name> <edge_name>  <vertex_name>.<property>  <edge_name>.<property> }</property></edge_name></property></vertex_name></edge_name></vertex_name>	MATCH (v:player) RETURN v.player.name, v.player.age LIMIT 3	Returns the first three rows with values of the vertex properties name and age.
TTL	<pre>CREATE TAG <tag_name>(<property_name_1> <property_value_1>, <property_name_2> <property_value_2>,) ttl_duration= <value_int>, ttl_col = <property_name></property_name></value_int></property_value_2></property_name_2></property_value_1></property_name_1></tag_name></pre>	<pre>CREATE TAG t2(a int, b int, c string) ttl_duration= 100, ttl_col = "a"</pre>	Create a tag and set the TTL options.
WHERE	<pre>WHERE {<vertex  edge_alias="">.<pre>roperty_name&gt; {&gt; == &lt; } <value>}</value></pre></vertex ></pre>	MATCH (v:player) WHERE v.player.name == "Tim Duncan" XOR (v.player.age < 30 AND v.player.name == "Yao Ming") OR NOT (v.player.name == "Yao Ming" OR v.player.name == "Tim Duncan") RETURN v.player.name, v.player.age	The WHERE clause filters the output by conditions. The WHERE clause usually works in Native nGQL 60 and LOOKUP statements, and OpenCypher MATCH and WITH statements.
YIELD	<pre>YIELD [DISTINCT] <col/> [AS <alias>] [,   <col/> [AS <alias>]] [WHERE   <conditions>];</conditions></alias></alias></pre>	GO FROM "player100" OVER follow YIELD dst(edge) AS ID   FETCH PROP ON player \$ID YIELD player.age AS Age   YIELD AVG(\$Age) as Avg_age, count(*)as Num_friends	Finds the players that "player100" follows and calculates their average age.
WITH	MATCH \$expressions WITH {nodes()  labels() }	MATCH p=(v:player{name:"Tim Duncan"}) () WITH nodes(p) AS n UNWIND n AS n1 RETURN DISTINCT n1	The WITH clause can retrieve the output from a query part, process it, and pass it to the next query part as the input.
UNWIND	UNWIND <list> AS <alias> <return clause=""></return></alias></list>	UNWIND [1,2,3] AS n RETURN n	Splits a list into rows.

# 3.7.4 Space statements

Statement	Syntax	Example	Description
CREATE SPACE	<pre>CREATE SPACE [IF NOT EXISTS]  <graph_space_name> ( [partition_num =</graph_space_name></pre>	<pre>CREATE SPACE my_space_1   (vid_type=FIXED_STRING(30))</pre>	Creates a graph space with
CREATE SPACE	<pre>CREATE SPACE <new_graph_space_name> AS <old_graph_space_name></old_graph_space_name></new_graph_space_name></pre>	CREATE SPACE my_space_4 as my_space_3	Clone a graph. space.
USE	USE <graph_space_name></graph_space_name>	USE space1	Specifies a graph space as the current working graph space for subsequent queries.
SHOW SPACES	SHOW SPACES	SHOW SPACES	Lists all the graph spaces in the NebulaGraph examples.
DESCRIBE SPACE	DESC[RIBE] SPACE <graph_space_name></graph_space_name>	DESCRIBE SPACE basketballplayer	Returns the information about the specified graph space.
CLEAR SPACE	CLEAR SPACE [IF EXISTS] <graph_space_name></graph_space_name>	Deletes the vertices and edges in a graph space, but does not delete the graph space itself and the schema information.	
DROP SPACE	DROP SPACE [IF EXISTS] <pre></pre>	DROP SPACE basketballplayer	Deletes everything in the specified graph

- 119/1069 - 2023 Vesoft Inc.

## 3.7.5 TAG statements

Statement	Syntax	Example	Description
CREATE TAG	CREATE TAG [IF NOT EXISTS] <tag_name> ( <prop_name> <data_type> [NULL   NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>'] [{, <prop_name> <data_type> [NULL   NOT NULL] [DEFAULT <default_value>] [COMMENT '<comment>']}] ) [TTL_DURATION = <ttl_duration>] [TTL_COL = <prop_name>] [COMMENT = '<comment>']</comment></prop_name></ttl_duration></comment></default_value></data_type></prop_name></comment></default_value></data_type></prop_name></tag_name>	CREATE TAG woman(name string, age int, married bool, salary double, create_time timestamp) TTL_DURATION = 100, TTL_COL = "create_time"	Creates a tag with the given name in a graph space.
DROP TAG	DROP TAG [IF EXISTS] <tag_name></tag_name>	DROP TAG test;	Drops a tag with the given name in the current working graph space.
ALTER TAG	ALTER TAG <tag_name> <alter_definition> [, alter_definition]] [ttl_definition [, ttl_definition]] [COMMENT = '<comment>']</comment></alter_definition></tag_name>	ALTER TAG t1 ADD (p3 int, p4 string)	Alters the structure of a tag with the given name in a graph space. You can add or drop properties, and change the data type of an existing property. You can also set a TTL (Time-To-Live) on a property, or change its TTL duration.
SHOW TAGS	SHOW TAGS	SHOW TAGS	Shows the name of all tags in the current graph space.
DESCRIBE TAG	DESC[RIBE] TAG <tag_name></tag_name>	DESCRIBE TAG player	Returns the information about a tag with the given name in a graph space, such as field names, data type, and so on.
DELETE TAG	DELETE TAG <tag_name_list> FROM <vid></vid></tag_name_list>	DELETE TAG test1 FROM "test"	Deletes a tag with the given name on a specified vertex.

- 120/1069 - 2023 Vesoft Inc.

# 3.7.6 Edge type statements

Statement	Syntax	Example	Description
CREATE EDGE	<pre>CREATE EDGE [IF NOT EXISTS] <edge_type_name>  ( <pre></pre></edge_type_name></pre>	CREATE EDGE e1(p1 string, p2 int, p3 timestamp) TTL_DURATION = 100, TTL_COL = "p2"	Creates an edge type with the given name in a graph space.
DROP EDGE	DROP EDGE [IF EXISTS] <edge_type_name></edge_type_name>	DROP EDGE e1	Drops an edge type with the given name in a graph space.
ALTER EDGE	ALTER EDGE <edge_type_name> <alter_definition> [, alter_definition]] [ttl_definition [, ttl_definition]] [COMMENT = '<comment>']</comment></alter_definition></edge_type_name>	ALTER EDGE e1 ADD (p3 int, p4 string)	Alters the structure of an edge type with the given name in a graph space.
SHOW EDGES	SHOW EDGES	SHOW EDGES	Shows all edge types in the current graph space.
DESCRIBE EDGE	<pre>DESC[RIBE] EDGE <edge_type_name></edge_type_name></pre>	DESCRIBE EDGE follow	Returns the information about an edge type with the given name in a graph space, such as field names, data type, and so on.

## 3.7.7 Vertex statements

Statement	Syntax	Example	Description
INSERT VERTEX	<pre>INSERT VERTEX [IF NOT EXISTS] [tag_props, [tag_props]] VALUES <vid>: ([prop_value_list])</vid></pre>	INSERT VERTEX t2 (name, age) VALUES "13":("n3", 12), "14":("n4", 8)	Inserts one or more vertices into a graph space in NebulaGraph.
DELETE VERTEX	DELETE VERTEX <vid> [, <vid>]</vid></vid>	DELETE VERTEX "team1"	Deletes vertices and the related incoming and outgoing edges of the vertices.
UPDATE VERTEX	UPDATE VERTEX ON <tag_name> <vid> SET <update_prop> [WHEN <condition>] [YIELD <output>]</output></condition></update_prop></vid></tag_name>	UPDATE VERTEX ON player "player101" SET age = age + 2	Updates properties on tags of a vertex.
UPSERT VERTEX	<pre>UPSERT VERTEX ON <tag> <vid> SET   <update_prop> [WHEN <condition>] [YIELD <output>]</output></condition></update_prop></vid></tag></pre>	UPSERT VERTEX ON player "player667" SET age = 31	The UPSERT statement is a combination of UPDATE and INSERT. You can use UPSERT VERTEX to update the properties of a vertex if it exists or insert a new vertex if it does not exist.

# 3.7.8 Edge statements

Statement	Syntax	Example	Description
INSERT EDGE	<pre>INSERT EDGE [IF NOT EXISTS] <edge_type>   ( <prop_name_list> ) VALUES <src_vid> -&gt;   <dst_vid>[@<rank>] : ( <prop_value_list> ) [,   <src_vid> -&gt; <dst_vid>[@<rank>] :   ( <prop_value_list> ),]</prop_value_list></rank></dst_vid></src_vid></prop_value_list></rank></dst_vid></src_vid></prop_name_list></edge_type></pre>	<pre>INSERT EDGE e2 (name, age) VALUES "11"- &gt;"13":("n1", 1)</pre>	Inserts an edge or multiple edges into a graph space from a source vertex (given by src_vid) to a destination vertex (given by dst_vid) with a specific rank in NebulaGraph.
DELETE EDGE	<pre>DELETE EDGE <edge_type> <src_vid> -&gt;   <dst_vid>[@<rank>] [, <src_vid> -&gt;   <dst_vid>[@<rank>]]</rank></dst_vid></src_vid></rank></dst_vid></src_vid></edge_type></pre>	DELETE EDGE serve "player100" -> "team204"@0	Deletes one edge or multiple edges at a time.
UPDATE EDGE	<pre>UPDATE EDGE ON <edge_type> <src_vid> -&gt; <dst_vid> [@<rank>] SET <update_prop> [WHEN <condition>] [YIELD <output>]</output></condition></update_prop></rank></dst_vid></src_vid></edge_type></pre>	<pre>UPDATE EDGE ON serve "player100" -&gt; "team204"@0 SET start_year = start_year + 1</pre>	Updates properties on an edge.
UPSERT EDGE	<pre>UPSERT EDGE ON <edge_type> <src_vid> -&gt; <dst_vid> [@rank] SET <update_prop> [WHEN <condition>] [YIELD <properties>]</properties></condition></update_prop></dst_vid></src_vid></edge_type></pre>	<pre>UPSERT EDGE on serve "player666" -&gt; "team200"@0 SET end_year = 2021</pre>	The UPSERT statement is a combination of UPDATE and INSERT. You can use UPSERT EDGE to update the properties of an edge if it exists or insert a new edge if it does not exist.

- 122/1069 - 2023 Vesoft Inc.

## 3.7.9 Index

## • Native index

You can use native indexes together with  ${\tt LOOKUP}$  and  ${\tt MATCH}$  statements.

Statement	Syntax	Example	Description
CREATE INDEX	CREATE {TAG   EDGE} INDEX [IF NOT EXISTS] <index_name> ON {<tag_name></tag_name></index_name>	CREATE TAG INDEX player_index on	Add native indexes for the existing tags, edge types, or properties.
	<pre>  <edge_name>} ([<prop_name_list>]) [COMMENT = '<comment>']</comment></prop_name_list></edge_name></pre>	player()	
SHOW	SHOW CREATE {TAG   EDGE} INDEX	show create tag index	Shows the statement used when creating a
CREATE	<index_name></index_name>	index_2	tag or an edge type. It contains detailed
INDEX			information about the index, such as its associated properties.
SHOW	SHOW {TAG   EDGE} INDEXES	SHOW TAG INDEXES	Shows the defined tag or edge type
INDEXES			indexes names in the current graph space.
DESCRIBE	DESCRIBE {TAG   EDGE} INDEX	DESCRIBE TAG INDEX	Gets the information about the index with
INDEX	<index_name></index_name>	player_index_0	a given name, including the property name (Field) and the property type (Type) of the index.
REBUILD	REBUILD {TAG   EDGE} INDEX	REBUILD TAG INDEX	Rebuilds the created tag or edge type
INDEX	[ <index_name_list>]</index_name_list>	single_person_index	index. If data is updated or inserted before the creation of the index, you must rebuild the indexes <b>manually</b> to make sure that the indexes contain the previously added data.
SHOW	SHOW {TAG   EDGE} INDEX STATUS	SHOW TAG INDEX STATUS	Returns the name of the created tag or
INDEX			edge type index and its status.
STATUS			
DROP	DROP {TAG   EDGE} INDEX [IF EXISTS]	DROP TAG INDEX	Removes an existing index from the
INDEX	<index_name></index_name>	player_index_0	current graph space.

## • Full-text index

Syntax	Example	Description
<pre>SIGN IN TEXT SERVICE [(<elastic_ip:port> [,<username>, <password>]), (<elastic_ip:port>),]</elastic_ip:port></password></username></elastic_ip:port></pre>	SIGN IN TEXT SERVICE (127.0.0.1:9200)	The full-text indexes is implemented based on Elasticsearch. After deploying an Elasticsearch cluster, you can use the SIGN IN statement to log in to the Elasticsearch client.
SHOW TEXT SEARCH CLIENTS	SHOW TEXT SEARCH CLIENTS	Shows text search clients.
SIGN OUT TEXT SERVICE	SIGN OUT TEXT SERVICE	Signs out to the text search clients.
<pre>CREATE FULLTEXT {TAG   EDGE} INDEX <index_name> ON {<tag_name>   <edge_name>} ([<prop_name_list>])</prop_name_list></edge_name></tag_name></index_name></pre>	CREATE FULLTEXT TAG INDEX nebula_index_1 ON player(name)	Creates full-text indexes.
SHOW FULLTEXT INDEXES	SHOW FULLTEXT INDEXES	Show full-text indexes.
REBUILD FULLTEXT INDEX	REBUILD FULLTEXT INDEX	Rebuild full-text indexes.
DROP FULLTEXT INDEX <index_name></index_name>	DROP FULLTEXT INDEX nebula_index_1	Drop full-text indexes.
LOOKUP ON { <tag>   <edge_type>} WHERE <expression> [YIELD <return_list>]</return_list></expression></edge_type></tag>	LOOKUP ON player WHERE FUZZY(player.name, "Tim Dunncan", AUTO, OR) YIELD player.name	Use query options.

# 3.7.10 Subgraph and path statements

Туре	Syntax	Example	Description
GET SUBGRAPH	GET SUBGRAPH [WITH PROP] [ <step_count>  {STEP STEPS}] FROM {<vid>&gt;, <vid>&gt;} [{IN    OUT   BOTH} <edge_type>, <edge_type>]  YIELD [VERTICES AS <vertex_alias>] [,EDGES  AS <edge_alias>]</edge_alias></vertex_alias></edge_type></edge_type></vid></vid></step_count>	GET SUBGRAPH 1 STEPS FROM "player100" YIELD VERTICES AS nodes, EDGES AS relationships	Retrieves information of vertices and edges reachable from the source vertices of the specified edge types and returns information of the subgraph.
FIND PATH	FIND { SHORTEST   ALL   NOLOOP } PATH [WITH PROP] FROM <vertex_id_list> TO <vertex_id_list> OVER <edge_type_list> [REVERSELY   BIDIRECT] [<where clause="">] [UPTO <n> {STEP STEPS}] YIELD path as <alias> [  ORDER BY \$path] [  LIMIT <m>]</m></alias></n></where></edge_type_list></vertex_id_list></vertex_id_list>	FIND SHORTEST PATH FROM "player102" TO "team204" OVER * YIELD path as p	Finds the paths between the selected source vertices and destination vertices. A returned path is like ( <vertex_id>)-[:<edge_type_name>@<rank>]-&gt;(<vertex_id).< td=""></vertex_id).<></rank></edge_type_name></vertex_id>

# 3.7.11 Query tuning statements

Туре	Syntax	Example	Description
EXPLAIN	<pre>EXPLAIN [format="row"   "dot"] <your_ngql_statement></your_ngql_statement></pre>	EXPLAIN format="row" SHOW TAGS EXPLAIN format="dot" SHOW TAGS	Helps output the execution plan of an nGQL statement without executing the statement.
PROFILE	PROFILE [format="row"   "dot"] <your_ngql_statement></your_ngql_statement>	PROFILE format="row" SHOW TAGS EXPLAIN format="dot" SHOW TAGS	Executes the statement, then outputs the execution plan as well as the execution profile.

- 124/1069 - 2023 Vesoft Inc.

## 3.7.12 Operation and maintenance statements

## • BALANCE

Syntax	Description
BALANCE LEADER	Starts a job to balance the distribution of all the storage leaders in graph spaces. It returns the job ID.

## • Job statements

Syntax	Description
SUBMIT JOB COMPACT	Triggers the long-term RocksDB compact operation.
SUBMIT JOB FLUSH	Writes the RocksDB memfile in the memory to the hard disk.
SUBMIT JOB STATS	Starts a job that makes the statistics of the current graph space. Once this job succeeds, you can use the SHOW STATS statement to list the statistics.
SHOW JOB <job_id></job_id>	Shows the information about a specific job and all its tasks in the current graph space. The Meta Service parses a SUBMIT JOB request into multiple tasks and assigns them to the nebula-storaged processes.
SHOW JOBS	Lists all the unexpired jobs in the current graph space.
STOP JOB	Stops jobs that are not finished in the current graph space.
RECOVER JOB	Re-executes the failed jobs in the current graph space and returns the number of recovered jobs.

## • Kill queries

KILL QUERY (session= <session_id>, KILL plan=<plan_id>) QUERY(SESSION=162555</plan_id></session_id>	Terminates the query being executed, and is often used to terminate slow queries.

# 4. nGQL guide

## 4.1 nGQL overview

## 4.1.1 NebulaGraph Query Language (nGQL)

This topic gives an introduction to the query language of NebulaGraph, nGQL.

### What is nGQL

nGQL is a declarative graph query language for NebulaGraph. It allows expressive and efficient graph patterns. nGQL is designed for both developers and operations professionals. nGQL is an SQL-like query language, so it's easy to learn.

nGQL is a project in progress. New features and optimizations are done steadily. There can be differences between syntax and implementation. Submit an issue to inform the NebulaGraph team if you find a new issue of this type. NebulaGraph 3.0 or later releases will support openCypher 9.

### What can nGQL do

- · Supports graph traversals
- · Supports pattern match
- · Supports aggregation
- Supports graph mutation
- · Supports access control
- Supports composite queries
- Supports index
- Supports most openCypher 9 graph query syntax (but mutations and controls syntax are not supported)

## Example data Basketballplayer

Users can download the example data Basketballplayer in NebulaGraph. After downloading the example data, you can import it to NebulaGraph by using the -f option in NebulaGraph Console.



Ensure that you have executed the ADD HOSTS command to add the Storage service to your NebulaGraph cluster before importing the example data. For more information, see Manage Storage hosts.

#### Placeholder identifiers and values

Refer to the following standards in nGQL:

- (Draft) ISO/IEC JTC1 N14279 SC 32 Database\_Languages GQL
- (Draft) ISO/IEC JTC1 SC32 N3228 SQL\_Property\_Graph\_Queries SQLPGQ
- OpenCypher 9

In template code, any token that is not a keyword, a literal value, or punctuation is a placeholder identifier or a placeholder value.

- 126/1069 - 2023 Vesoft Inc.

For details of the symbols in nGQL syntax, see the following table:

Token	Meaning
< >	name of a syntactic element
:	formula that defines an element
[]	optional elements
{ }	explicitly specified elements
1	complete alternative elements
	may be repeated any number of times

#### For example, create vertices in nGQL syntax:

```
INSERT VERTEX [IF NOT EXISTS] [tag_props, [tag_props] ...]
VALUES <vid>: ([prop_value_list])
tag_props:
   tag_name ([prop_name_list])
prop_name_list:
   [prop_name [, prop_name] ...]
prop_value_list:
   [prop_value [, prop_value] ...]
```

#### Example statement:

```
nebula> CREATE TAG IF NOT EXISTS player(name string, age int);
```

#### About openCypher compatibility

NATIVE NGQL AND OPENCYPHER

Native nGQL is the part of a graph query language designed and implemented by NebulaGraph. OpenCypher is a graph query language maintained by openCypher Implementers Group.

The latest release is openCypher 9. The compatible parts of openCypher in nGQL are called openCypher compatible sentences (short as openCypher).



```
nGQL = native nGQL + openCypher compatible sentences
```

IS NGQL COMPATIBLE WITH OPENCYPHER 9 COMPLETELY?

NO.

# Empatibility with openCypher

nGQL is designed to be compatible with part of DQL (match, optional match, with, etc.).

- $\bullet$  It is not planned to be compatible with any DDL, DML, or DCL.
- It is not planned to be compatible with the Bolt Protocol.
- It is not planned to be compatible with APOC and GDS.

Users can search in this manual with the keyword compatibility to find major compatibility issues.

Multiple known incompatible items are listed in NebulaGraph Issues. Submit an issue with the incompatible tag if you find a new issue of this type.

WHAT ARE THE MAJOR DIFFERENCES BETWEEN NGQL AND OPENCYPHER 9?

The following are some major differences (by design incompatible) between nGQL and openCypher.

Category	openCypher 9	nGQL
Schema	Optional Schema	Strong Schema
Equality operator		=
Math exponentiation	Λ	^ is not supported. Use pow(x, y) instead.
Edge rank	No such concept.	edge rank (reference by @)
Statement	-	All DMLs ( $\mbox{\tt CREATE}$ , $\mbox{\tt MERGE}$ , etc.) of $\mbox{\tt openCypher}$ 9.
Label and tag	A label is used for searching a vertex, namely an index of vertex.	A tag defines the type of a vertex and its corresponding properties. It cannot be used as an index.
Pre-compiling and parameterized queries	Support	Parameterized queries are supported, but precompiling is not.



OpenCypher 9 and Cypher have some differences in grammar and licence. For example,

- 1. Cypher requires that **All Cypher statements are explicitly run within a transaction**. While openCypher has no such requirement. And nGQL does not support transactions.
- 2. Cypher has a variety of constraints, including Unique node property constraints, Node property existence constraints, Relationship property existence constraints, and Node key constraints. While OpenCypher has no such constraints. As a strong schema system, most of the constraints mentioned above can be solved through schema definitions (including NOT NULL) in nGQL. The only function that cannot be supported is the UNIQUE constraint.
- 3. Cypher has APoC, while openCypher 9 does not have APoC. Cypher has Blot protocol support requirements, while openCypher 9 does not.

WHERE CAN I FIND MORE NGQL EXAMPLES?

Users can find more than 2500 nGQL examples in the features directory on the NebulaGraph GitHub page.

The features directory consists of .feature files. Each file records scenarios that you can use as nGQL examples. Here is an example:

```
Feature: Basic match
  Background:
    Given a graph with space named "basketballplayer"
  Scenario: Single node
    When executing query:
      MATCH (v:player {name: "Yao Ming"}) RETURN v;
    Then the result should be, in any order, with relax comparison:
      ("player133" :player{age: 38, name: "Yao Ming"})
  Scenario: One step
    When executing query:
      MATCH (v1:player{name: "LeBron James"}) -[r]-> (v2)
      RETURN type(r) AS Type, v2.player.name AS Name
    Then the result should be, in any order:
       Type
"follow"
                   "Ray Allen"
        "serve"
                   "Lakers'
                   "Heat"
        "serve"
Feature: Comparison of where clause
```

```
Background:
Given a graph with space named "basketballplayer"

Scenario: push edge props filter down
When profiling query:
"""

GO FROM "player100" OVER follow
WHERE properties(edge).degree IN [v IN [95,99] WHERE v > 0]
YIELD dst(edge), properties(edge).degree
"""

Then the result should be, in any order:
| follow._dst | follow.degree |
| "player101" | 95
| "player101" | 95
| "player125" | 95
| And the execution plan should be:
| id | name | dependencies | operator info
| 0 | Project | 1 | |
| 1 | GetNeighbors | 2 | {"filter": "(properties(edge).degree IN [v IN [95,99] WHERE (v>0)])"} |
| 2 | Start |
```

The keywords in the preceding example are described as follows.

Keyword	Description
Feature	Describes the topic of the current .feature file.
Background	Describes the background information of the current .feature file.
Given	Describes the prerequisites of running the test statements in the current lifeature file.
Scenario	Describes the scenarios. If there is the <code>@skip</code> before one <code>Scenario</code> , this scenario may not work and do not use it as a working example in a production environment.
When	Describes the nGQL statement to be executed. It can be a executing query or profiling query .
Then	Describes the expected return results of running the statement in the When clause. If the return results in your environment do not match the results described in the .feature file, submit an issue to inform the NebulaGraph team.
And	Describes the side effects of running the statement in the When clause.
@skip	This test case will be skipped. Commonly, the to-be-tested code is not ready.

Welcome to add more tck case and return automatically to the using statements in CI/CD.

DOES IT SUPPORT TINKERPOP GREMLIN?

No. And no plan to support that.

DOES NEBULAGRAPH SUPPORT W3C RDF (SPARQL) OR GRAPHQL?

No. And no plan to support that.

The data model of NebulaGraph is the property graph. And as a strong schema system, NebulaGraph does not support RDF.

NebulaGraph Query Language does not support SPARQL nor GraphQL.

#### 4.1.2 Patterns

Patterns and graph pattern matching are the very heart of a graph query language. This topic will describe the patterns in NebulaGraph, some of which have not yet been implemented.

#### Patterns for vertices

A vertex is described using a pair of parentheses and is typically given a name. For example:

(a)

This simple pattern describes a single vertex and names that vertex using the variable a.

#### Patterns for related vertices

A more powerful construct is a pattern that describes multiple vertices and edges between them. Patterns describe an edge by employing an arrow between two vertices. For example:

(a)-[]->(b)

This pattern describes a very simple data structure: two vertices and a single edge from one to the other. In this example, the two vertices are named as a and b respectively and the edge is directed: it goes from a to b.

This manner of describing vertices and edges can be extended to cover an arbitrary number of vertices and the edges between them, for example:

(a)-[]->(b)<-[]-(c)

Such a series of connected vertices and edges is called a path.

Note that the naming of the vertices in these patterns is only necessary when one needs to refer to the same vertex again, either later in the pattern or elsewhere in the query. If not, the name may be omitted as follows:

(a)-[]->()<-[]-(c)

## Patterns for tags



The concept of tag in nGQL has a few differences from that of label in openCypher. For example, users must create a tag before using it. And a tag also defines the type of properties.

In addition to simply describing the vertices in the graphs, patterns can also describe the tags of the vertices. For example:

(a:User)-[]->(b)

Patterns can also describe a vertex that has multiple tags. For example:

(a:User:Admin)-[]->(b)

## Patterns for properties

Vertices and edges are the fundamental elements in a graph. In nGQL, properties are added to them for richer models.

In the patterns, the properties can be expressed as follows: some key-value pairs are enclosed in curly brackets and separated by commas, and the tag or edge type to which a property belongs must be specified.

- 130/1069 - 2023 Vesoft Inc.

For example, a vertex with two properties will be like:

```
(a:player{name: "Tim Duncan", age: 42})
```

One of the edges that connect to this vertex can be like:

```
(a)-[e:follow{degree: 95}]->(b)
```

#### Patterns for edges

The simplest way to describe an edge is by using the arrow between two vertices, as in the previous examples.

Users can describe an edge and its direction using the following statement. If users do not care about its direction, the arrowhead can be omitted. For example:

```
(a)-[]-(b)
```

Like vertices, edges can also be named. A pair of square brackets will be used to separate the arrow and the variable will be placed between them. For example:

```
(a)-[r]->(b)
```

Like the tags on vertices, edges can also have types. To describe an edge with a specific type, use the pattern as follows:

```
(a)-[r:REL_TYPE]->(b)
```

An edge can only have one edge type. But if we'd like to describe some data such that the edge could have a set of types, then they can all be listed in the pattern, separating them with the pipe symbol | like this:

```
(a)-[r:TYPE1|TYPE2]->(b)
```

Like vertices, the name of an edge can be omitted. For example:

```
(a)-[:REL_TYPE]->(b)
```

### Variable-length pattern

Rather than describing a long path using a sequence of many vertex and edge descriptions in a pattern, many edges (and the intermediate vertices) can be described by specifying a length in the edge description of a pattern. For example:

```
(a)-[*2]->(b)
```

The following pattern describes a graph of three vertices and two edges, all in one path (a path of length 2). It is equivalent to:

```
(a)-[]->()-[]->(b)
```

The range of lengths can also be specified. Such edge patterns are called variable-length edges. For example:

```
(a)-[*3..5]->(b)
```

The preceding example defines a path with a minimum length of 3 and a maximum length of 5.

It describes a graph of either 4 vertices and 3 edges, 5 vertices and 4 edges, or 6 vertices and 5 edges, all connected in a single path.

The lower bound can be omitted. For example, to describe paths of length 5 or less, use:

```
(a)-[*..5]->(b)
```

- 131/1069 - 2023 Vesoft Inc.



The upper bound must be specified. The following are  $\boldsymbol{NOT}$  accepted.

(a)-[\*3..]->(b) (a)-[\*]->(b)

## Assigning to path variables

As described above, a series of connected vertices and edges is called a path. nGQL allows paths to be named using variables. For example:

p = (a)-[\*3..5]->(b)

Users can do this in the MATCH statement.

## 4.1.3 Comments

This topic will describe the comments in nGQL.

# Lacy version compatibility

- $\bullet$  In NebulaGraph 1.x, there are four comment styles: # , -- , // , /\* \*/ .
- Since NebulaGraph 2.x, -- cannot be used as comments.

### **Examples**

In nGQL statement, the backslash  $\ \ \ \ \ \$  in a line indicates a line break.

## OpenCypher compatibility

- In nGQL, you must add a  $\setminus$  at the end of every line, even in multi-line comments  $/^*$  \*/.
- In openCypher, there is no need to use a \ as a line break.

```
/* openCypher style:
The following comment
spans more than
one line */
MATCH (n:label)
RETURN n;

/* nGQL style: \
The following comment
spans more than \
one line */
MATCH (n:tag) \
RETURN n;
```

## 4.1.4 Identifier case sensitivity

#### **Identifiers are Case-Sensitive**

The following statements will not work because they refer to two different spaces, i.e. my\_space and MY\_SPACE.

```
nebula> CREATE SPACE IF NOT EXISTS my_space (vid_type=FIXED_STRING(30));
nebula> use NY_SPACE;
[ERROR (-1005)]: SpaceNotFound:
```

## Keywords and Reserved Words are Case-Insensitive

The following statements are equivalent since show and spaces are keywords.

```
nebula> show spaces;
nebula> SHOW SPACES;
nebula> SHOW spaces;
nebula> show SPACES;
```

#### **Functions are Case-Insensitive**

Functions are case-insensitive. For example, count(), count(), and count() are equivalent.

## 4.1.5 Keywords

Keywords have significance in nGQL. It can be classified into reserved keywords and non-reserved keywords. It is not recommend to use keywords in schema.

If you must use keywords in schema:

- Non-reserved keywords can be used as identifiers without quotes if they are all in lowercase. However, if a non-reserved keyword contains any uppercase letters when used as an identifier, it must be enclosed in backticks (`), for example, `Comment`.
- ullet To use special characters or reserved keywords as identifiers, quote them with backticks such as  ${\tt AND}$ .



Keywords are case-insensitive.

```
nebula> CREATE TAG TAG(name string);
[ERROR (-1004)]: SyntaxError: syntax error near `TAG'

nebula> CREATE TAG `TAG` (name string);
Execution succeeded

nebula> CREATE TAG SPACE(name string);
Execution succeeded

nebula> CREATE TAG 中文(简体 string);
Execution succeeded

nebula> CREATE TAG 中文(简体 string);
Execution succeeded
```

### Reserved keywords

```
ACROSS
ALTER
AND
ASC
ASCENDING
BALANCE
B00L
CASE
CHANGE
CREATE
DATE
DATETIME
DELETE
DESC
DESCENDING
DESCRIBE
DISTINCT
DOUBLE
DOWNLOAD
DROP
DURATION
EDGE
EDGES
EXTSTS
EXPLAIN
FETCH
FIND
FIXED_STRING
FLOAT
FLUSH
FORMAT
FROM
GET
G0
GRANT
IGNORE_EXISTED_INDEX
INDEX
INDEXES
```

```
INGEST
INSERT
INT
INT16
INT32
INT64
INT8
INTERSECT
      IS
LIMIT
      LOOKUP
MAP
MATCH
MINUS
NO
   NO
NOT
NOT_IN
NULL
OF
OFFSET
ON
OR
      ORDER
    OVER
OVERWRITE
PROFILE
PROFILE
PROP
REBUILD
RECOVER
REMOVE
RESTART
RETURN
REVERSELY
REVOKE
SET
SHOW
STEPS
STOP
STRING
SUBMIT
TAG
TIAGS
TIME
TIMESTAMP
TO
UNION
UPDATE
UPSERT
UPTO
USE
VERTEX
VERTEX
VERTICES
WHEN
WHERE
WITH
XOR
YIELD
```

## Non-reserved keywords

```
GROUPS
GUEST
HDFS
HOST
HOSTS
INTO
IS_EMPTY
IS_NOT_EMPTY
IS_NOT_NULL
IS_NULL
JOB
JOBS
   JOBS
KILL
 KILL
LEADER
LISTENER
META
NOLOOP
NONE
NOT_CONTAINS
NOT_ENDS_WITH
NOT_STARTS_WITH
OPTIONAL
OUT
PART
PARTITION_NUM
PARTS
PASSWORD
PATH
   PATH
PLAN
PREFIX
  QUERIES
QUERY
REDUCE
REGEXP
REPLICA_FACTOR
RESET
ROLE
ROLES
SAMPLE
SEARCH
SERVICE
SESSION
SHORTEST
SION
SHORTEST
SINGLE
SKIP
SNAPSHOT
SNAPSHOT
SNAPSHOT
SPACES
  STARTS
STARTS_WITH
STATS
STATUS
STORAGE
   SUBGRAPH
TEXT
TEXT_SEARCH
  THEN
TOP
TTL_COL
TTL_DURATION
UNWIND
   USER
USERS
UUID
  VALUE
VALUES
VID_TYPE
WILDCARD
ZONE
ZONES
FALSE
   TRUE
```

## 4.1.6 nGQL style guide

nGQL does not have strict formatting requirements, but creating nGQL statements according to an appropriate and uniform style can improve readability and avoid ambiguity. Using the same nGQL style in the same organization or project helps reduce maintenance costs and avoid problems caused by format confusion or misunderstanding. This topic will provide a style guide for writing nGQL statements.

# **Umpatibility**

The styles of nGQL and Cypher Style Guide are different.

#### Newline

1. Start a new line to write a clause.

Not recommended:

```
GO FROM "player100" OVER follow REVERSELY YIELD src(edge) AS id;
```

#### Recommended:

```
GO FROM "player100" \
OVER follow REVERSELY \
YIELD src(edge) AS id;
```

2. Start a new line to write different statements in a composite statement.

#### Not recommended:

```
GO FROM "player100" OVER follow REVERSELY YIELD src(edge) AS id | GO FROM $-.id \
OVER serve WHERE properties($^).age > 20 YIELD properties($^).name AS FriendOf, properties($$).name AS Team;
```

#### Recommended:

```
GO FROM "player100" \
OVER follow REVERSELY \
YIELD src(edge) AS id | \
GO FROM $-.id OVER serve \
WHERE properties($^\).age > 20 \
YIELD properties($^\).name AS FriendOf, properties($$).name AS Team;
```

3. If the clause exceeds 80 characters, start a new line at the appropriate place.

## Not recommended:

```
MATCH (v:player{name:"Tim Duncan"})-[e]->(v2) \
WHERE (v2.player.name STARTS WITH "V" AND v2.player.age > 35 AND v2.player.age < v.player.age) \
RETURN v2;
```

## Recommended:

```
MATCH (v:player{name:"Tim Duncan"})-[e]->(v2) \
WHERE (v2.player.name STARTS WITH "Y" AND v2.player.age > 35 AND v2.player.age < v.player.age) \
OR (v2.player.name STARTS WITH "T" AND v2.player.age < 45 AND v2.player.age) \
RETURN v2;
```



If needed, you can also start a new line for better understanding, even if the clause does not exceed 80 characters.

- 138/1069 - 2023 Vesoft Inc.

#### **Identifier naming**

In nGQL statements, characters other than keywords, punctuation marks, and blanks are all identifiers. Recommended methods to name the identifiers are as follows.

1. Use singular nouns to name tags, and use the base form of verbs or verb phrases to form Edge types.

Not recommended:

```
MATCH p=(v:players)-[e:are_following]-(v2) \
RETURN nodes(p);
```

#### Recommended:

```
MATCH p=(v:player)-[e:follow]-(v2) \
RETURN nodes(p);
```

2. Use the snake case to name identifiers, and connect words with underscores (\_) with all the letters lowercase.

Not recommended:

```
MATCH (v:basketballTeam) \
RETURN v;
```

#### Recommended:

```
MATCH (v:basketball_team) \
RETURN v;
```

3. Use uppercase keywords and lowercase variables.

Not recommended:

```
match (V:player) return V limit 5;
```

## Recommended:

```
MATCH (v:player) RETURN v LIMIT 5;
```

#### Pattern

 $1. \ Start \ a \ new \ line \ on \ the \ right \ side \ of \ the \ arrow \ indicating \ an \ edge \ when \ writing \ patterns.$ 

Not recommended:

```
MATCH (v:player{name: "Tim Duncan", age: 42}) \
-[e:follow]->()-[e:serve]->()<--(v2) \
RETURN v, e, v2;
```

#### Recommended:

```
MATCH (v:player{name: "Tim Duncan", age: 42})-[e:follow]-> \
()-[e:serve]->()<--(v2) \
RETURN v, e, v2;
```

2. Anonymize the vertices and edges that do not need to be queried.

Not recommended:

```
MATCH (v:player)-[e:follow]->(v2) \
RETURN v;
```

#### Recommended:

```
MATCH (v:player)-[:follow]->() \
RETURN v;
```

3. Place named vertices in front of anonymous vertices.

Not recommended:

```
MATCH ()-[:follow]->(v) \
RETURN v;
```

#### Recommended:

```
MATCH (v)<-[:follow]-() \
RETURN v;
```

#### String

The strings should be surrounded by double quotes.

Not recommended:

```
RETURN 'Hello Nebula!';
```

### Recommended:

```
RETURN "Hello Nebula!\"123\"";
```



When single or double quotes need to be nested in a string, use a backslash () to escape. For example:

```
RETURN "\"NebulaGraph is amazing,\" the user says.";
```

#### Statement termination

1. End the nGQL statements with an English semicolon (;).

Not recommended:

```
FETCH PROP ON player "player100" YIELD properties(vertex)
```

#### Recommended:

```
FETCH PROP ON player "player100" YIELD properties(vertex);
```

2. Use a pipe (|) to separate a composite statement, and end the statement with an English semicolon at the end of the last line. Using an English semicolon before a pipe will cause the statement to fail.

Not supported:

```
GO FROM "player100" \
OVER follow \
YIELD dst(edge) AS id; | \
GO FROM $-.id \
OVER serve \
YIELD properties($$).name AS Team, properties($^).name AS Player;
```

## Supported:

```
GO FROM "player100" \
OVER follow \
YIELD dst(edge) AS id | \
GO FROM $-.id \
OVER serve \
YIELD properties($$).name AS Team, properties($^).name AS Player;
```

3. In a composite statement that contains user-defined variables, use an English semicolon to end the statements that define the variables. If you do not follow the rules to add a semicolon or use a pipe to end the composite statement, the execution will fail.

Not supported:

```
Svar = G0 FROM "player100" \
OVER follow \
YIELD follow._dst AS id \
G0 FROM $var.id \
```

```
OVER serve \
YIELD $$.team.name AS Team, $^.player.name AS Player;
```

## Not supported:

```
$var = G0 FROM "player100" \
OVER follow \
YIELD follow._dst AS id | \
G0 FROM $var.id \
OVER serve \
YIELD $$.team.name AS Team, $^.player.name AS Player;
```

## Supported:

```
$var = G0 FROM "player100" \
0VER follow \
YIELD follow._dst AS id; \
G0 FROM $var.id \
0VER serve \
YIELD $$.team.name AS Team, $^.player.name AS Player;
```

## 4.2 Data types

## 4.2.1 Numeric types

nGQL supports both integer and floating-point number.

#### Integer

Signed 64-bit integer (INT64), 32-bit integer (INT32), 16-bit integer (INT16), and 8-bit integer (INT8) are supported.

INT64 INT64 or INT -9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807  INT32 INT32 -2,147,483,648 ~ 2,147,483,647  INT16 INT16 -32,768 ~ 32,767  INT8 -128 ~ 127	Туре	Declared keywords	Range
INT16 INT16 -32,768 ~ 32,767	INT64	INT64 or INT	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
	INT32	INT32	-2,147,483,648 ~ 2,147,483,647
INT8 INT8 -128 ~ 127	INT16	INT16	-32,768 ~ 32,767
	INT8	INT8	-128 ~ 127

#### Floating-point number

Both single-precision floating-point format (FLOAT) and double-precision floating-point format (DOUBLE) are supported.

Туре	Declared keywords	Range	Precision
FLOAT	FLOAT	3.4E +/- 38	6~7 bits
DOUBLE	DOUBLE	1.7E +/- 308	15~16 bits

Scientific notation is also supported, such as  $\ 1e2$ ,  $\ 1.1e2$ ,  $\ .3e4$ ,  $\ 1.e4$ , and  $\ -1234E-10$ .



The data type of DECIMAL in MySQL is not supported.

## Reading and writing of data values

When writing and reading different types of data, nGQL complies with the following rules:

pperty Resulted data type
INT64
INT64
INT64
INT64
DOUBLE
DOUBLE

- 142/1069 - 2023 Vesoft Inc.

For example, nGQL does not support setting VID as INT8, but supports setting a certain property type of TAG or Edge type as INT8. When using the nGQL statement to read the property of INT8, the resulted type is INT64.

- Multiple formats are supported:
- Decimal, such as 123456.
- Hexadecimal, such as 0x1e240.
- Octal, such as 0361100.

However, NebulaGraph will parse the written non-decimal value into a decimal value and save it. The value read is decimal. For example, the type of the property score is INT. The value of 0xb is assigned to it through the INSERT statement. If querying the property value with statements such as FETCH, you will get the result 11, which is the decimal result of the hexadecimal 0xb.

 $\bullet$  Round a FLOAT/DOUBLE value when inserting it to an INT column.

## 4.2.2 Boolean

A boolean data type is declared with the  $\mbox{bool}$  keyword and can only take the values  $\mbox{true}$  or  $\mbox{false}$ .

nGQL supports using boolean in the following ways:

- $\bullet$  Define the data type of the property value as a boolean.
- $\bullet$  Use boolean as judgment conditions in the  $\mbox{\sc WHERE}$  clause.

# 4.2.3 String

Fixed-length strings and variable-length strings are supported.

### Declaration and literal representation

The string type is declared with the keywords of:

- STRING: Variable-length strings.
- FIXED\_STRING(<length>): Fixed-length strings. <length> is the length of the string, such as FIXED\_STRING(32).

A string type is used to store a sequence of characters (text). The literal constant is a sequence of characters of any length surrounded by double or single quotes. For example, "Hello, Cooper" or 'Hello, Cooper'.

### String reading and writing

Nebula Graph supports using string types in the following ways:

- Define the data type of VID as a fixed-length string.
- Set the variable-length string as the Schema name, including the names of the graph space, tag, edge type, and property.
- Define the data type of the property as a fixed-length or variable-length string.

For example:

• Define the data type of the property as a fixed-length string

```
nebula> CREATE TAG IF NOT EXISTS t1 (p1 FIXED_STRING(10));
```

• Define the data type of the property as a variable-length string

```
nebula> CREATE TAG IF NOT EXISTS t2 (p2 STRING);
```

When the fixed-length string you try to write exceeds the length limit:

- If the fixed-length string is a property, the writing will succeed, and NebulaGraph will truncate the string and only store the part that meets the length limit.
- If the fixed-length string is a VID, the writing will fail and NebulaGraph will return an error.

## Escape characters

Line breaks are not allowed in a string. Escape characters are supported within strings, for example:

- "\n\t\r\b\f"
- "\110ello world"

## OpenCypher compatibility

There are some tiny differences between openCypher and Cypher, as well as nGQL. The following is what openCypher requires. Single quotes cannot be converted to double quotes.

```
# File: Literals.feature
Feature: Literals

Background:
    Given any graph
Scenario: Return a single-quoted string
    When executing query:
    """

RETURN '' AS literal
    """
```

```
Then the result should be, in any order:
| literal |
| '' | # Note: it should return single-quotes as openCypher required.
And no side effects
```

While Cypher accepts both single quotes and double quotes as the return results. nGQL follows the Cypher way.

# 4.2.4 Date and time types

This topic will describe the DATE, TIME, DATETIME, TIMESTAMP, and DURATION types.

#### Precautions

• While inserting time-type property values with DATE, TIME, and DATETIME, NebulaGraph transforms them to a UTC time according to the timezone specified with the timezone\_name parameter in the configuration files.



To change the timezone, modify the timezone\_name value in the configuration files of all NebulaGraph services.

- date(), time(), and datetime() can convert a time-type property with a specified timezone. For example, datetime("2017-03-04 22:30:40.003000+08:00") or datetime("2017-03-04T22:30:40.003000[Asia/Shanghai]").
- date(), time(), datetime(), and timestamp() all accept empty parameters to return the current date, time, and datetime.
- date(), time(), and datetime() all accept the property name to return a specific property value of itself. For example, date().month returns the current month, while time("02:59:40").minute returns the minutes of the importing time.

### OpenCypher Compatibility

In nGQL:

- · Year, month, day, hour, minute, second, millisecond, and microsecond are supported, while the nanosecond is not supported.
- localdatetime() is not supported.
- Most string time formats are not supported. The exceptions are YYYY-MM-DDThh:mm:ss and YYYY-MM-DD hh:mm:ss.
- ullet The single-digit string time format is supported. For example,  $\time("1:1:1")$ .

### DATE

The DATE type is used for values with a date part but no time part. Nebula Graph retrieves and displays DATE values in the YYYY-MM-DD format. The supported range is -32768-01-01 to 32767-12-31.

The properties of date() include year, month, and day.

### TIME

The TIME type is used for values with a time part but no date part. Nebula Graph retrieves and displays TIME values in hh:mm:ss.msmsususus format. The supported range is 00:00:00.000000 to 23:59:59.999999.

The properties of time() include hour, minute, and second.

### **DATETIME**

The DATETIME type is used for values that contain both date and time parts. Nebula Graph retrieves and displays DATETIME values in YYYY-MM-DDThh:mm:ss.msmsmsususus format. The supported range is -32768-01-01T00:00:00.000000 to 32767-12-31T23:59:59.999999 .

- The properties of datetime() include year, month, day, hour, minute, and second.
- datetime() can convert TIMESTAMP to DATETIME. The value range of TIMESTAMP is 0~9223372036.
- $\bullet$  datetime() supports an int argument. The int argument specifies a timestamp.

- 147/1069 - 2023 Vesoft Inc.

## TIMESTAMP

The TIMESTAMP data type is used for values that contain both date and time parts. It has a range of 1970-01-01T00:00:01 UTC to 2262-04-11T23:47:16 UTC.

TIMESTAMP has the following features:

- Stored and displayed in the form of a timestamp, such as 1615974839, which means 2021-03-17T17:53:59.
- Supported TIMESTAMP querying methods: timestamp and timestamp() function.
- Supported TIMESTAMP inserting methods: timestamp, timestamp() function, and now() function.
- timestamp() function accepts empty arguments to get the current timestamp. It can pass an integer arguments to identify the integer as a timestamp and the range of passed integer is: 0~9223372036 °
- timestamp() function can convert DATETIME to TIMESTAMP, and the data type of DATETIME should be a string.
- The underlying storage data type is int64.



The date and time format string passed into timestamp() cannot include any millisecond and microsecond, but the date and time format string passed into timestamp(datetime()) can include a millisecond and a microsecond.

### **DURATION**

The DURATION data type is used to indicate a period of time. Map data that are freely combined by years, months, days, hours, minutes, and seconds indicates the DURATION.

DURATION has the following features:

- Creating indexes for DURATION is not supported.
- DURATION can be used to calculate the specified time.

## **Examples**

1. Create a tag named date1 with three properties: DATE, TIME, and DATETIME.

```
nebula> CREATE TAG IF NOT EXISTS date1(p1 date, p2 time, p3 datetime);
```

2. Insert a vertex named test1.

```
nebula> INSERT VERTEX date1(p1, p2, p3) VALUES "test1":(date("2021-03-17"), time("17:53:59"), datetime("2017-03-04T22:30:40.003000[Asia/Shanghai]"));
```

3. Query whether the value of property p1 on the test1 tag is 2021-03-17.

4. Return the content of the property pl on testl.

5. Search for vertices with p3 property values less than 2023-01-01700:00:00.000000, and return the p3 values.

6. Create a tag named school with the property of TIMESTAMP.

```
nebula> CREATE TAG IF NOT EXISTS school(name string , found_time timestamp);
```

7. Insert a vertex named DUT with a found-time timestamp of "1988-03-01T08:00:00".

```
# Insert as a timestamp. The corresponding timestamp of 1988-03-01T08:00:00 is 573177600, or 573206400 UTC.
nebula> INSERT VERTEX school(name, found_time) VALUES "DUT":("DUT", 573206400);

# Insert in the form of date and time.
nebula> INSERT VERTEX school(name, found_time) VALUES "DUT":("DUT", timestamp("1988-03-01T08:00:00"));
```

8. Insert a vertex named dut and store time with now() or timestamp() functions.

```
# Use now() function to store time
nebula> INSERT VERTEX school(name, found_time) VALUES "dut":("dut", now());
# Use timestamp() function to store time
nebula> INSERT VERTEX school(name, found_time) VALUES "dut":("dut", timestamp());
```

You can also use WITH statement to set a specific date and time, or to perform calculations. For example:

# 4.2.5 NULL

You can set the properties for vertices or edges to NULL. Also, you can set the NOT NULL constraint to make sure that the property values are NOT NULL. If not specified, the property is set to NULL by default.

### Logical operations with NULL

Here is the truth table for  $\mbox{\sc AND}\,,\,\mbox{\sc OR}\,,\,\mbox{\sc XOR}\,,\,\mbox{\sc and}\,\,\mbox{\sc NOT}\,.$ 

a	b	a AND b	a OR b	a XOR b	NOT a
false	false	false	false	false	true
false	null	false	null	null	true
false	true	false	true	true	true
true	false	false	true	true	false
true	null	null	true	null	false
true	true	true	true	false	false
null	false	false	null	null	null
null	null	null	null	null	null
null	true	null	true	null	null

## OpenCypher compatibility

The comparisons and operations about NULL are different from openCypher. There may be changes later.

COMPARISONS WITH NULL

The comparison operations with NULL are incompatible with openCypher.

OPERATIONS AND RETURN WITH NULL

The NULL operations and RETURN with NULL are incompatible with openCypher.

## **Examples**

USE NOT NULL

Create a tag named player. Specify the property name as NOT NULL.

```
nebula> CREATE TAG IF NOT EXISTS player(name string NOT NULL, age int);
```

Use SHOW to create tag statements. The property name is NOT NULL. The property age is NULL by default.

nebula> SHOW CREATE TAG player;	
+	+
Tag   Create Tag	
Tag   Create Tag	
+	+
-to-desett       CDEATE TAC \  \alpha     \	
"student"   "CREATE TAG `player` (	
name` string NOT NULL,	
	!
age` int64 NULL	
) ttl_duration = 0, ttl_col = """	i
	A .
+	+

Insert the vertex Kobe. The property age can be NULL.

```
nebula> INSERT VERTEX player(name, age) VALUES "Kobe":("Kobe",null);
```

USE NOT NULL AND SET THE DEFAULT

Create a tag named  $\,$  player . Specify the property  $\,$  age  $\,$  as  $\,$  NOT  $\,$  NULL . The default value is  $\,$  18 .

```
nebula> CREATE TAG IF NOT EXISTS player(name string, age int NOT NULL DEFAULT 18);
```

Insert the vertex Kobe. Specify the property name only.

```
nebula> INSERT VERTEX player(name) VALUES "Kobe":("Kobe");
```

Query the vertex Kobe. The property age is 18 by default.

## 4.2.6 Lists

The list is a composite data type. A list is a sequence of values. Individual elements in a list can be accessed by their positions.

A list starts with a left square bracket [ and ends with a right square bracket ]. A list contains zero, one, or more expressions. List elements are separated from each other with commas ( , ). Whitespace around elements is ignored in the list, thus line breaks, tab stops, and blanks can be used for formatting.

## OpenCypher compatibility

A composite data type (i.e. set, map, and list) CANNOT be stored as properties of vertices or edges.

## List operations

You can use the preset list function to operate the list, or use the index to filter the elements in the list.

INDEX SYNTAX

```
[M]
[M.N]
[M..]
[.N]
```

The index of nGQL supports queries from front to back, starting from 0.0 means the first element, 1 means the second element, and so on. It also supports queries from back to front, starting from -1. -1 means the last element, -2 means the penultimate element, and so on.

- [M]: represents the element whose index is M.
- [M..N]: represents the elements whose indexes are greater or equal to M but smaller than N. Return empty when N is O.
- $\bullet$  [M..]: represents the elements whose indexes are greater or equal to M.
- [..N]: represents the elements whose indexes are smaller than N. Return empty when N is O.

# Note

- Return empty if the index is out of bounds, while return normally if the index is within the bound.
- Return empty if  $M \ge N$ .
- $\bullet \ \ When \ querying \ a \ single \ element, \ if \ \ \texttt{M} \ \ is \ null, \ return \ \ \ \texttt{BAD\_TYPE} \ . \ When \ conducting \ a \ range \ query, \ if \ \ \texttt{M} \ \ or \ \ \texttt{N} \ \ is \ null, \ return \ \ \ null \ .$

# Examples

```
| range(1,5)[0..3] |
[1, 2, 3]
# The following query returns the elements whose indexes are greater than 2 in the list [1,2,3,4,5]. nebula> RETURN range(1,5)[3..] AS a;
a
| [4, 5] |
# The following query returns the elements whose indexes are smaller than 3. nebula> WITH list[1, 2, 3, 4, 5] AS a \backslash
       RETURN a[..3] AS r;
| r
| [1, 2, 3] |
# The following query filters the elements whose indexes are greater than 2 in the list [1,2,3,4,5], calculate them respectively, and returns them.
nebula> RETURN [n IN range(1,5) WHERE n > 2 | n + 10] AS a;
| a
| [13, 14, 15] |
# The following query returns the elements from the first to the penultimate (inclusive) in the list [1, 2, 3].
nebula> YIELD list[1, 2, 3][0..-1] AS a;
a
| [1, 2] |
# The following query returns the elements from the first (exclusive) to the third backward in the list [1, 2, 3, 4, 5]. nebula> YIELD list[1, 2, 3, 4, 5][-3..-1] AS a;
a
[3, 4]
\# The following query sets the variables and returns the elements whose indexes are 1 and 2.
nebula> $var = YIELD 1 AS f, 3 AS t; \
        YIELD list[1, 2, 3][$var.f..$var.t] AS a;
| a
| [2, 3] |
# The following query returns empty because the index is out of bound. It will return normally when the index is within the bound.
nebula> RETURN list[1, 2, 3, 4, 5] [0..10] AS a;
| [1, 2, 3, 4, 5] |
nebula> RETURN list[1, 2, 3] [-5..5] AS a;
a
| [1, 2, 3] |
\# The following query returns empty because there is a [0..0]. nebula> RETURN list[1, 2, 3, 4, 5] [0..0] AS a;
la l
[] [
# The following query returns empty because of M \geq N. nebula> RETURN list[1, 2, 3, 4, 5] [3..1] AS a;
| a |
[] [
# When conduct a range query, if `M` or `N` is null, return `null`.
nebula> WITH list[1,2,3] AS a \
        RETURN a[0..null] as r;
| r
| __NULL__ |
```

```
# The following query calculates the elements in the list [1,2,3,4,5] respectively and returns them without the list head.
nebula> RETURN tail([n IN range(1, 5) | 2 * n - 10]) AS a;
| [-6, -4, -2, 0] |
# The following query takes the elements in the list [1,2,3] as true and return.
nebula> RETURN [n IN range(1, 3) WHERE true \mid n] AS r;
| [1, 2, 3] |
# The following query returns the length of the list [1,2,3].
nebula> RETURN size(list[1,2,3]);
| size(list[1,2,3])
| 3
# The following query calculates the elements in the list [92,90] and runs a conditional judgment in a where clause.
nebula> GO FROM "player100" OVER follow WHERE properties(edge).degree NOT IN [x IN [92, 90] | x + $$.player.age] \ YIELD dst(edge) AS id, properties(edge).degree AS degree;
lid
               degree
 "player101" | 95
"player102" | 90
# The following query takes the query result of the MATCH statement as the elements in a list. Then it calculates and returns them. nebula> MATCH p = (n:player{name:"Tim Duncan"})-[:follow]->(m) \
        RETURN [n IN nodes(p) | n.player.age + 100] AS r;
[142, 136]
  [142, 141]
```

## OpenCypher compatibility

• In openCypher, return <code>null</code> when querying a single out-of-bound element. However, in nGQL, return <code>OUT\_OF\_RANGE</code> when querying a single out-of-bound element.

ullet A composite data type (i.e., set, map, and list) **CAN NOT** be stored as properties for vertices or edges.

It is recommended to modify the graph modeling method. The composite data type should be modeled as an adjacent edge of a vertex, rather than its property. Each adjacent edge can be dynamically added or deleted. The rank values of the adjacent edges can be used for sequencing.

 $\bullet$  Patterns are not supported in the list. For example, <code>[(src)-[]->(m) \mid m.name]</code> .

## 4.2.7 Sets

The set is a composite data type. A set is a set of values. Unlike a List, values in a set are unordered and each value must be unique.

A set starts with a left curly bracket { and ends with a right curly bracket }. A set contains zero, one, or more expressions. Set elements are separated from each other with commas ( , ). Whitespace around elements is ignored in the set, thus line breaks, tab stops, and blanks can be used for formatting.

## OpenCypher compatibility

- A composite data type (i.e. set, map, and list) CANNOT be stored as properties of vertices or edges.
- A set is not a data type in openCypher, but in nGQL, users can use the set.

### **Examples**

# 4.2.8 Maps

The map is a composite data type. Maps are unordered collections of key-value pairs. In maps, the key is a string. The value can have any data type. You can get the map element by using <code>map['key']</code>.

A map starts with a left curly bracket { and ends with a right curly bracket } . A map contains zero, one, or more key-value pairs. Map elements are separated from each other with commas ( , ). Whitespace around elements is ignored in the map, thus line breaks, tab stops, and blanks can be used for formatting.

## OpenCypher compatibility

- A composite data type (i.e. set, map, and list) CANNOT be stored as properties of vertices or edges.
- Map projection is not supported.

### **Examples**

# 4.2.9 Type Conversion/Type coercions

Converting an expression of a given type to another type is known as type conversion.

NebulaGraph supports converting expressions explicit to other types. For details, see Type conversion functions.

## **Examples**

# 4.2.10 Geography

Geography is a data type composed of latitude and longitude that represents geospatial information. NebulaGraph currently supports Point, LineString, and Polygon in Simple Features and some functions in SQL-MM 3, such as part of the core geo parsing, construction, formatting, conversion, predicates, and dimensions.

# Type description

A point is the basic data type of geography, which is determined by a latitude and a longitude. For example, "POINT(3 8)" means that the longitude is 3° and the latitude is 8°. Multiple points can form a linestring or a polygon.

Shape	Example	Description
Point	"POINT(3 8)"	Specifies the data type as a point.
LineString	"LINESTRING(3 8, 4.7 73.23)"	Specifies the data type as a linestring.
Polygon	"POLYGON((0 1, 1 2, 2 3, 0 1))"	Specifies the data type as a polygon.

#### **Examples**

For functions about the geography data type, see Geography functions.

```
//Create a Tag to allow storing any geography data type.
nebula> CREATE TAG IF NOT EXISTS any_shape(geo geography);
//Create a Tag to allow storing a point only.
nebula> CREATE TAG IF NOT EXISTS only_point(geo geography(point));
//Create a Tag to allow storing a linestring only.
nebula> CREATE TAG IF NOT EXISTS only_linestring(geo geography(linestring));
//Create a Tag to allow storing a polygon only.
nebula> CREATE TAG IF NOT EXISTS only_polygon(geo geography(polygon));
//Create an Edge type to allow storing any geography data type. nebula> CREATE EDGE IF NOT EXISTS any_shape_edge(geo geography);
//Create a vertex to store the geography of a polygon.
nebula> INSERT VERTEX any_shape(geo) VALUES "103":(ST_GeogFromText("POLYGON((0 1, 1 2, 2 3, 0 1))"));
//Create an edge to store the geography of a polygon.
nebula> INSERT EDGE any_shape_edge(geo) VALUES "201"->"302":(ST_GeogFromText("POLYGON((0 1, 1 2, 2 3, 0 1))"));
//Query the geography of Vertex 103. nebula> FETCH PROP ON any_shape "103" YIELD ST_ASText(any_shape.geo);
| ST ASText(any shape.geo)
| "POLYGON((0 1, 1 2, 2 3, 0 1))"
//Query the geography of the edge which traverses from Vertex 201 to Vertex 302. nebula> FETCH PROP ON any_shape_edge "201"->"302" YIELD ST_ASText(any_shape_edge.geo);
| ST_ASText(any_shape_edge.geo)
| "POLYGON((0 1, 1 2, 2 3, 0 1))" |
//Create an index for the geography of the Tag any_shape and run LOOKUP
nebula> CREATE TAG INDEX IF NOT EXISTS any_shape_geo_index ON any_shape(geo);
nebula> REBUILD TAG INDEX any_shape_geo_index;
nebula> LOOKUP ON any_shape YIELD ST_ASText(any_shape.geo);
| ST_ASText(any_shape.geo)
| "POLYGON((0 1, 1 2, 2 3, 0 1))" |
```

When creating an index for geography properties, you can specify the parameters for the index.

Parameter	Default value	Description
s2_max_level	30	The maximum level of S2 cell used in the covering. Allowed values: $1\sim30$ . Setting it to less than the default means that NebulaGraph will be forced to generate coverings using larger cells.
s2_max_cells	8	The maximum number of S2 cells used in the covering. Provides a limit on how much work is done exploring the possible coverings. Allowed values: $1\sim30$ . You may want to use higher values for odd-shaped regions such as skinny rectangles.



Specifying the above two parameters does not affect the Point type of property. The  $s2\_max\_level$  value of the Point type is forced to be 30.

nebula> CREATE TAG INDEX IF NOT EXISTS any\_shape\_geo\_index ON any\_shape(geo) with (s2\_max\_level=30, s2\_max\_cells=8);

# 4.3 Variables and composite queries

## 4.3.1 Composite queries (clause structure)

Composite queries put data from different queries together. They then use filters, group-bys, or sorting before returning the combined return results.

Nebula Graph supports three methods to run composite queries (or sub-queries):

- (openCypher) Clauses are chained together, and they feed intermediate result sets between each other.
- (Native nGQL) More than one query can be batched together, separated by semicolons (;). The result of the last query is returned as the result of the batch.
- (Native nGQL) Queries can be piped together by using the pipe (||). The result of the previous query can be used as the input of the next query.

## OpenCypher compatibility

In a composite query, **do not** put together openCypher and native nGQL clauses in one statement. For example, this statement is undefined: MATCH ... | 60 ... | YIELD .....

- If you are in the openCypher way (MATCH, RETURN, WITH, etc), do not introduce any pipe or semicolons to combine the sub-clauses.
- If you are in the native nGQL way ( FETCH, 60, LOOKUP, etc), you must use pipe or semicolons to combine the sub-clauses.

## Composite queries are not transactional queries (as in SQL/Cypher)

For example, a query is composed of three sub-queries: A B C,  $A \mid B \mid C$  or A; B; C. In that A is a read operation, B is a computation operation, and C is a write operation. If any part fails in the execution, the whole result will be undefined. There is no rollback. What is written depends on the query executor.



OpenCypher has no requirement of transaction.

### **Examples**

• OpenCypher compatibility statement

- 161/1069 - 2023 Vesoft Inc.

```
UNWIND n AS n1 \
RETURN DISTINCT n1;
```

## • Native nGQL (Semicolon queries)

# • Native nGQL (Pipe queries)

## 4.3.2 User-defined variables

User-defined variables allow passing the result of one statement to another.

### OpenCypher compatibility

In openCypher, when you refer to the vertex, edge, or path of a variable, you need to name it first. For example:

The user-defined variable in the preceding query is  $\ v$ .



In a pattern of a MATCH statement, you cannot use the same edge variable repeatedly. For example, e cannot be written in the pattern  $p=(v1)-[e^*2..2]->(v2)-[e^*2..2]->(v3)$ .

## Native nGQL

User-defined variables are written as \$var\_name . The var\_name consists of letters, numbers, or underline characters. Any other characters are not permitted.

The user-defined variables are valid only at the current execution (namely, in this composite query). When the execution ends, the user-defined variables will be automatically expired. The user-defined variables in one statement **CANNOT** be used in any other clients, executions, or sessions.

You can use user-defined variables in composite queries. Details about composite queries, see Composite queries.



- User-defined variables are case-sensitive.
- To define a user-defined variable in a compound statement, end the statement with a semicolon (;). For details, please refer to the nGQL Style Guide.

# Example

# 4.3.3 Property reference

You can refer to the properties of a vertex or an edge in  $\mbox{WHERE}$  and  $\mbox{YIELD}$  syntax.



This function applies to native nGQL only.

## Property reference for vertex

FOR SOURCE VERTEX

Parameter

pescription

show is used to get the property of the source vertex.

tag\_name is the tag name of the vertex.

prop\_name specifies the property name.

## FOR DESTINATION VERTEX

\$\$.<tag\_name>.<prop\_name>

Parameter	Description
\$\$	is used to get the property of the destination vertex.
tag_name	is the tag name of the vertex.
prop_name	specifies the property name.

# Property reference for edge

FOR USER-DEFINED EDGE PROPERTY

<edge\_type>.<prop\_name>

Parameter	Description
edge_type	is the edge type of the edge.
prop_name	specifies the property name of the edge type.

# FOR BUILT-IN PROPERTIES

Apart from the user-defined edge property, there are four built-in properties in each edge:

Parameter	Description
_src	source vertex ID of the edge
_dst	destination vertex ID of the edge
_type	edge type
_rank	the rank value for the edge

- 164/1069 - 2023 Vesoft Inc.

### **Examples**

The following query returns the name property of the player tag on the source vertex and the age property of the player tag on the destination vertex.

The following query returns the degree property of the edge type follow.

The following query returns the source vertex, the destination vertex, the edge type, and the edge rank value of the edge type follow.

# L Jacy version compatibility

NebulaGraph 2.6.0 and later versions support the new Schema-related functions. Similar statements as the above examples are written as follows in 3.4.1.

```
GO FROM "player100" OVER follow YIELD properties($^).name AS startName, properties($$).age AS endAge;
GO FROM "player100" OVER follow YIELD properties(edge).degree;
GO FROM "player100" OVER follow YIELD src(edge), dst(edge), type(edge);
```

In 3.4.1, NebulaGraph is still compatible with the old syntax.

# 4.4 Operators

# 4.4.1 Comparison operators

NebulaGraph supports the following comparison operators.

Name	Description
<b>=</b>	Assigns a value
+	Addition operator
	Minus operator
*	Multiplication operator
/	Division operator
==	Equal operator
!=, <>	Not equal operator
>	Greater than operator
>=	Greater than or equal operator
<	Less than operator
<=	Less than or equal operator
%	Modulo operator
	Changes the sign of the argument
IS NULL	NULL check
IS NOT NULL	Not NULL check
IS EMPTY	EMPTY check
IS NOT EMPTY	Not EMPTY check

The result of the comparison operation is  $\mbox{ true }$  or  $\mbox{ false }.$ 



- $\bullet \ \ Comparability \ between \ values \ of \ different \ types \ is \ often \ undefined. \ The \ result \ could \ be \ \ \verb|NULL| \ or \ others.$
- EMPTY is currently used only for checking, and does not support functions or operations such as GROUP BY, count(), sum(), max(), hash(), collect(), + or \*.

# OpenCypher compatibility

# Examples

==

String comparisons are case-sensitive. Values of different types are not equal.

- 166/1069 - 2023 Vesoft Inc.

#### Q Note

The equal operator is = in nGQL, while in openCypher it is =.

>

```
nebula> RETURN 3 > 2;

+-----+
| (3-2) |
+-----+
| true |
+-----+

nebula> WITH 4 AS one, 3 AS two \
RETURN one > two AS result;
+-----+
| result |
+-----+
| true |
+------+
```

>=

```
nebula> RETURN 2 >= "2", 2 >= 2;

+------+

| (2>="2") | (2>=2) |

+------+

| __NULL__ | true |
```

<

```
nebula> YIELD 2.0 < 1.9;
+------+
| (2<1.9) |
+------+
| false |
+------+
```

<=

```
nebula> YIELD 0.11 <= 0.11;
+-------+
| (0.11<=0.11) |
+------+
| true |
+------+
```

!=

```
nebula> YIELD 1 != '1';

+-----+
| (1!="1") |
+-----+
| true |
+-----+
```

IS [NOT] NULL

```
nebula> RETURN null IS NULL AS value1, null == null AS value2, null != null AS value3;

+-----+------+
| value1 | value2 | value3 |
+-----+------+
| true | __NULL__ | __NULL__ |
+------+---------+
```

### IS [NOT] EMPTY

# 4.4.2 Boolean operators

NebulaGraph supports the following boolean operators.

Name	Description
AND	Logical AND
NOT	Logical NOT
OR	Logical OR
XOR	Logical XOR

For the precedence of the operators, refer to Operator Precedence.

For the logical operations with  $\ensuremath{\,^{\text{NULL}}}$  , refer to  $\ensuremath{\,^{\text{NULL}}}$  .

# Legacy version compatibility

• Non-zero numbers cannot be converted to boolean values.

# 4.4.3 Pipe operators

Multiple queries can be combined using pipe operators in nGQL.

## OpenCypher compatibility

Pipe operators apply to native nGQL only.

### Syntax

One major difference between nGQL and SQL is how sub-queries are composed.

- In SQL, sub-queries are nested in the query statements.
- In nGQL, the shell style PIPE (|) is introduced into the sub-queries.

### **Examples**

Users must define aliases in the YIELD clause for the reference operator \$- to use, just like \$-.dstid in the preceding example.

### Performance tips

In NebulaGraph, pipes will affect the performance. Take  $A \mid B$  as an example, the effects are as follows:

- 1. Pipe operators operate synchronously. That is, the data can enter the pipe clause as a whole after the execution of clause A before the pipe operator is completed.
- 2. Pipe operators need to be serialized and descrialized, which is executed in a single thread.
- 3. If A sends a large amount of data to [], the entire query request may be very slow. You can try to split this statement.
- a. Send  $\overline{\mbox{\sc A}}$  from the application,
- b. Split the return results on the application,
- c. Send to multiple graphd processes concurrently,
- d. Every graphd process executes part of B.

This is usually much faster than executing a complete  $A \mid B$  with a single graphd process.

# 4.4.4 Reference operators

NGQL provides reference operators to represent a property in a WHERE or YIELD clause, or the output of the statement before the pipe operator in a composite query.

## OpenCypher compatibility

Reference operators apply to native nGQL only.

## Reference operator List

Reference operator	Description
\$^	Refers to a source vertex property. For more information, see Property reference.
\$\$	Refers to a destination vertex property. For more information, see Property reference.
\$-	Refers to the output of the statement before the pipe operator in a composite query. For more information, see <a href="Pipe">Pipe</a> .

## **Examples**

## 4.4.5 Set operators

This topic will describe the set operators, including UNION, UNION ALL, INTERSECT, and MINUS. To combine multiple queries, use these set operators.

All set operators have equal precedence. If a nGQL statement contains multiple set operators, NebulaGraph will evaluate them from left to right unless parentheses explicitly specify another order.



The names and order of the variables defined in the query statements before and after the set operator must be consistent. For example, the names and order of a,b,c in RETURN a,b,c union return a,b,c need to be consistent.

## UNION, UNION DISTINCT, and UNION ALL

```
<left> UNION [DISTINCT | ALL] <right> [ UNION [DISTINCT | ALL] <right> ...]
```

- Operator UNION DISTINCT (or by short UNION) returns the union of two sets A and B without duplicated elements.
- Operator UNION ALL returns the union of two sets A and B with duplicated elements.
- The <left> and <right> must have the same number of columns and data types. Different data types are converted according to the Type Conversion.

### **EXAMPLES**

```
UNION \
        GO FROM "player100" OVER follow YIELD dst(edge);
dst(EDGE)
  "player100"
 "player101"
"player125"
nebula> MATCH (v:player) \
        WITH v.player.name AS v \
RETURN n ORDER BY n LIMIT 3 \
        UNWIND ["Tony Parker", "Ben Simmons"] AS n \setminus
        RETURN n:
| n
  "Amar'e Stoudemire"
  "Aron Baynes"
  "Ben Simmons"
  "Tony Parker"
\# The following statement returns the union of two query results with duplicated elements. nebula> GO FROM "player102" OVER follow YIELD dst(edge) \
        UNION ALL
        GO FROM "player100" OVER follow YIELD dst(edge);
dst(EDGE)
  "player100"
  "player101"
  "player101'
  "player125"
nebula> MATCH (v:player) \
        WITH v.player.name AS n \
RETURN n ORDER BY n LIMIT 3 \
        UNION ALL \
        UNWIND ["Tony Parker", "Ben Simmons"] AS n \setminus
        RETURN n;
l n
  "Amar'e Stoudemire"
"Aron Baynes"
```

```
| "Ben Simmons" |
| "Tony Parker" |
| "Ben Simmons" |
| "Ben Simmons" |
| "UNION can also work with the YIELD statement. The DISTINCT keyword will check duplication by all the columns for every line, and remove duplicated lines if every column is the same.

# UNION can also work with the YIELD statement. The DISTINCT keyword will check duplication by all the columns for every line, and remove duplicated lines if every column is the same.

# UNION can also work with the YIELD statement. The DISTINCT keyword will check duplication by all the columns for every line, and remove duplicated lines if every column is the same.

# UNION can also work with the YIELD statement. The DISTINCT keyword will check duplication by all the columns for every line, and remove duplicated lines if every column is the same.

# UNION can also work with the YIELD statement. The DISTINCT keyword will check duplication by all the columns for every line, and remove duplicated lines if every column is the same.

# UNION can also work with the YIELD statement. The DISTINCT keyword will check duplication by all the columns for every line, and remove duplicated lines if every column is the same.

# UNION can also work with the YIELD statement. The DISTINCT keyword will check duplication by all the columns for every line, and remove duplicated lines if every column is the same.

# UNION can also work with the YIELD statement. The DISTINCT keyword will check duplication by all the columns for every line, and remove duplicated lines if every column is the same.

# UNION can also work with the YIELD statement. The DISTINCT keyword will check duplication by all the columns for every line, and remove duplicated lines if every column is the same.

# UNION can also work with the YIELD statement. The DISTINCT keyword will check duplication by all the columns for every line, and remove duplicated lines if every column is the same.

# UNION can also work with the YIELD statement. The DISTINCT keyword will be columns for every line, and remove duplicated
```

#### INTERSECT

```
<left> INTERSECT <right>
```

- Operator Intersect returns the intersection of two sets A and B (denoted by A  $\cap$  B).
- Similar to UNION, the left and right must have the same number of columns and data types. Different data types are converted according to the Type Conversion.

### EXAMPLE

```
# The following statement returns the intersection of two query results.
nebula> GO FROM "player102" OVER follow \
YIELD dst(edge) AS id, properties(edge).degree AS Degree, properties($$).age AS Age \
        INTERSECT
        GO FROM "player100" OVER follow \
       YIELD dst(edge) AS id, properties(edge).degree AS Degree, properties($$).age AS Age;
| id | Degree | Age |
nebula> MATCH (v:player)-[e:follow]->(v2) \
        WHERE id(v) = "player102"
        RETURN id(v2) As id, e.degree As Degree, v2.player.age AS Age \
       MATCH (v:player)-[e:follow]->(v2) \
WHERE id(v) == "player100" \
        RETURN id(v2) As id, e.degree As Degree, v2.player.age AS Age;
| id | Degree | Age |
+----+
nebula> UNWIND [1,2] AS a RETURN a \
        INTERSECT \
        UNWIND [1,2,3,4] AS a \
        RETURN a:
| a |
1 1 |
```

## MINUS

```
<left> MINUS <right>
```

Operator MINUS returns the subtraction (or difference) of two sets A and B (denoted by A-B). Always pay attention to the order of left and right. The set A-B consists of elements that are in A but not in B.

## EXAMPLE

```
| "player125" |
nebula> GO FROM "player102" OVER follow YIELD dst(edge) AS id\
       GO FROM "player100" OVER follow YIELD dst(edge) AS id;
| id
| "player100"
nebula> MATCH (v:player)-[e:follow]->(v2) \
        WHERE id(v) =="player102"
       RETURN id(v2) AS id\
       MATCH (v:player)-[e:follow]->(v2) \
       WHERE id(v) =="player100"
       RETURN id(v2) AS id;
| id
| "player100" |
nebula> UNWIND [1,2,3] AS a RETURN a \
       MINUS \
        WITH 4 AS a \
       RETURN a;
| a |
| 1 |
 2 |
```

#### Precedence of the set operators and pipe operators

Please note that when a query contains a pipe | and a set operator, the pipe takes precedence. Refer to Pipe for details. The query GO FROM 1 UNION GO FROM 2 | GO FROM 3 is the same as the query GO FROM 1 UNION (GO FROM 2 | GO FROM 3).

#### **EXAMPLES**

```
nebula> GO FROM "player102" OVER follow YIELD follow._dst AS play_dst \
UNION \
GO FROM "team200" OVER serve REVERSELY YIELD serve._dst AS play_dst \
GO FROM $-.play_dst OVER follow YIELD follow._dst AS play_dst;
```

The above query executes the statements in the red bar first and then executes the statement in the green box.

The parentheses can change the execution priority. For example:

```
nebula> (60 FROM "player102" OVER follow \
YIELD dst(edge) AS play_dst \
UNION \
GO FROM "team200" OVER serve REVERSELY \
YIELD src(edge) AS play_dst) \
| GO FROM $-.play_dst OVER follow YIELD dst(edge) AS play_dst;
```

In the above query, the statements within the parentheses take precedence. That is, the UNION operation will be executed first, and its output will be executed as the input of the next operation with pipes.

# 4.4.6 String operators

You can use the following string operators for concatenating, querying, and matching.

Name	Description
+	Concatenates strings.
CONTAINS	Performs searchings in strings.
(NOT) IN	Checks whether a value is within a set of values.
(NOT) STARTS WITH	Performs matchings at the beginning of a string.
(NOT) ENDS WITH	Performs matchings at the end of a string.
Regular expressions	Perform string matchings using regular expressions.



All the string searchings or matchings are case-sensitive.

# **Examples**

+

```
nebula> RETURN 'a' + 'b';

+------+
| ("a"+"b") |
+-----+
| "ab" |
+-----+
| (a+b) |
+-----+
| "ab" |
```

# CONTAINS

The CONTAINS operator requires string types on both left and right sides.

# (NOT) IN

### (NOT) STARTS WITH

### (NOT) ENDS WITH

REGULAR EXPRESSIONS



Regular expressions cannot work with native nGQL statements (60, FETCH, LOOKUP, etc.). Use it in openCypher only (MATCH, WHERE, etc.).

NebulaGraph supports filtering by using regular expressions. The regular expression syntax is inherited from std::regex. You can match on regular expressions by using = regexp'. For example:

# 4.4.7 List operators

NebulaGraph supports the following list operators:

List operator	Description
+	Concatenates lists.
IN	Checks if an element exists in a list.
[]	Accesses an element(s) in a list using the index operator.

# **Examples**

```
nebula> YIELD [1,2,3,4,5]+[6,7] AS myList;
myList
| [1, 2, 3, 4, 5, 6, 7] |
nebula> RETURN size([NULL, 1, 2]);
| size([NULL,1,2]) |
nebula> RETURN NULL IN [NULL, 1];
| (NULL IN [NULL,1]) |
| __NULL__
+-----
nebula> WITH [2, 3, 4, 5] AS numberlist \backslash UNWIND numberlist AS number \backslash
    WITH number \
WHERE number IN [2, 3, 8] \
    RETURN number;
number
| 2
nebula> WITH ['Anne', 'John', 'Bill', 'Diane', 'Eve'] AS names RETURN names[1] AS result;
| result |
| "John" |
```

# 4.4.8 Operator precedence

The following list shows the precedence of nGQL operators in descending order. Operators that are shown together on a line have the same precedence.

- - (negative number)
- !, NOT
- \*,/,%
- - . +
- == , >= , > , <= , < , <> , !=
- AND
- OR , XOR
- = (assignment)

For operators that occur at the same precedence level within an expression, evaluation proceeds left to right, with the exception that assignments evaluate right to left.

The precedence of operators determines the order of evaluation of terms in an expression. To modify this order and group terms explicitly, use parentheses.

## Examples

# OpenCypher compatibility

In openCypher, comparisons can be chained arbitrarily, e.g.,  $x < y \le z$  is equivalent to x < y AND  $y \le z$  in openCypher.

But in nGQL,  $x < y \le z$  is equivalent to  $(x < y) \le z$ . The result of (x < y) is a boolean. Compare it with an integer z, and you will get the final result NULL.

# 4.5 Functions and expressions

## 4.5.1 Built-in math functions

This topic describes the built-in math functions supported by NebulaGraph.

## abs()

abs() returns the absolute value of the argument.

Syntax: abs(<expression>)

- expression : An expression of which the result type is double.
- Result type: Double

## Example:

## floor()

floor() returns the largest integer value smaller than or equal to the argument.(Rounds down)

Syntax: floor(<expression>)

- expression: An expression of which the result type is double.
- Result type: Double

## Example:

```
nebula> RETURN floor(9.9);
+------+
| floor(9.9) |
+------+
| 9.0 |
+------+
```

# ceil()

ceil() returns the smallest integer greater than or equal to the argument.(Rounds up)

Syntax: ceil(<expression>)

- expression : An expression of which the result type is double.
- · Result type: Double

## Example:

#### round()

round() returns the rounded value of the specified number. Pay attention to the floating-point precision when using this function.

Syntax: round(<expression>, <digit>)

- expression: An expression of which the result type is double.
- digit: Decimal digits. If digit is less than 0, round at the left of the decimal point.
- Result type: Double

#### Example:

#### sqrt()

sqrt() returns the square root of the argument.

Syntax: sqrt(<expression>)

- expression : An expression of which the result type is double.
- Result type: Double

### Example:

```
mebula> RETURN sqrt(9);
+------+
| sqrt(9) |
+------+
| 3.0 |
+------+
```

### cbrt()

cbrt() returns the cubic root of the argument.

Syntax: cbrt(<expression>)

- expression : An expression of which the result type is double.
- Result type: Double

### Example:

```
mebula> RETURN cbrt(8);
+-----+
| cbrt(8) |
+-----+
| 2.0 |
+-----+
```

#### hypot()

hypot() returns the hypotenuse of a right-angled triangle.

Syntax: hypot(<expression\_x>,<expression\_y>)

- expression\_x , expression\_y : An expression of which the result type is double. They represent the side lengths x and y of a right triangle.
- Result type: Double

### Example:

```
nebula> RETURN hypot(3,2*2);
+-----+
| hypot(3,(2*2)) |
+-----+
| 5.0 |
+-----+
```

#### pow()

pow() returns the result of  $x^y$ .

Syntax: pow(<expression\_x>,<expression\_y>,)

- $\bullet$   $expression\_x$  : An expression of which the result type is double. It represents the base  $\ x$  .
- expression\_y: An expression of which the result type is double. It represents the exponential y.
- · Result type: Double

#### Example:

```
nebula> RETURN pow(3,3);
+-------+
| pow(3,3) |
+-------+
| 27 |
+------+
```

### exp()

exp() returns the result of  $e^{X}$ .

Syntax: exp(<expression>)

- expression: An expression of which the result type is double. It represents the exponential x.
- Result type: Double

### Example:

### exp2()

 $\exp 2()$  returns the result of  $2^{x}$ .

Syntax: exp2(<expression>)

- expression: An expression of which the result type is double. It represents the exponential x.
- · Result type: Double

#### Example:

```
nebula> RETURN exp2(3);
+------+
| exp2(3) |
+-----+
| 8.0 |
+-----+
```

### log()

log() returns the base-e logarithm of the argument. (\(log\_{e}{N}\))

Syntax: log(<expression>)

- expression: An expression of which the result type is double. It represents the antilogarithm N.
- · Result type: Double

### Example:

### log2()

log2() returns the base-2 logarithm of the argument. (\(log\_{2}{N}\))

Syntax: log2(<expression>)

- expression: An expression of which the result type is double. It represents the antilogarithm N.
- Result type: Double

#### Example:

```
nebula> RETURN log2(8);
+-----+
| log2(8) |
+-----+
| 3.0 |
```

### log10()

log10() returns the base-10 logarithm of the argument. (\(log\_{10}{N}\))

Syntax: log10(<expression>)

- ullet expression : An expression of which the result type is double. It represents the antilogarithm  $\,{
  m N}$  .
- Result type: Double

#### Example:

### sin()

sin() returns the sine of the argument. Users can convert angles to radians using the function radians().

Syntax: sin(<expression>)

• expression: An expression of which the result type is double.

· Result type: Double

#### Example:

#### asin()

asin() returns the inverse sine of the argument. Users can convert angles to radians using the function radians().

Syntax: asin(<expression>)

- $\bullet\,$  expression : An expression of which the result type is double.
- Result type: Double

#### Example:

#### cos()

cos() returns the cosine of the argument. Users can convert angles to radians using the function radians().

Syntax: cos(<expression>)

- expression : An expression of which the result type is double.
- Result type: Double

### Example:

### acos()

acos() returns the inverse cosine of the argument. Users can convert angles to radians using the function radians().

Syntax: acos(<expression>)

- expression: An expression of which the result type is double.
- Result type: Double

```
| 1.0471975511965979 |
+------+
```

#### tan()

tan() returns the tangent of the argument. Users can convert angles to radians using the function radians().

Syntax: tan(<expression>)

- expression: An expression of which the result type is double.
- · Result type: Double

### Example:

### atan()

atan() returns the inverse tangent of the argument. Users can convert angles to radians using the function radians().

Syntax: atan(<expression>)

- expression : An expression of which the result type is double.
- Result type: Double

#### Example:

#### rand()

rand() returns a random floating point number in the range from 0 (inclusive) to 1 (exclusive); i.e.[0,1).

Syntax: rand()

• Result type: Double

### Example:

#### rand32()

rand32() returns a random 32-bit integer in [min, max).

Syntax: rand32(<expression\_min>,<expression\_max>)

- expression\_min: An expression of which the result type is int. It represents the minimum min.
- expression\_max: An expression of which the result type is int. It represents the maximum max.
- Result type: Int
- If you set only one argument, it is parsed as max and min is 0 by default. If you set no argument, the system returns a random signed 32-bit integer.

### Example:

#### rand64()

rand64() returns a random 64-bit integer in [min, max).

Syntax: rand64(<expression\_min>,<expression\_max>)

- expression\_min: An expression of which the result type is int. It represents the minimum min.
- $\bullet$  expression\_max : An expression of which the result type is int. It represents the maximum  $\mbox{\ max}$  .
- Result type: Int
- If you set only one argument, it is parsed as max and min is 0 by default. If you set no argument, the system returns a random signed 64-bit integer.

### Example:

### bit\_and()

bit\_and() returns the result of bitwise AND.

Syntax: bit\_and(<expression\_1>,<expression\_2>)

- $\bullet$   $expression\_1$  ,  $expression\_2$  : An expression of which the result type is int.
- Result type: Int

### Example:

#### bit\_or()

bit\_or() returns the result of bitwise OR.

 $Syntax: \ \ \texttt{bit\_or}(\texttt{<expression\_1>,<expression\_2>})$ 

- expression\_1, expression\_2: An expression of which the result type is int.
- Result type: Int

#### Example:

#### bit\_xor()

bit\_xor() returns the result of bitwise XOR.

Syntax: bit\_xor(<expression\_1>,<expression\_2>)

- $\bullet$  expression\_1 , expression\_2 : An expression of which the result type is int.
- Result type: Int

#### Example:

#### size()

size() returns the number of elements in a list or a map, or the length of a string.

Syntax: size({<expression>|<string>})

- expression: An expression for a list or map.
- string: A specified string.
- Result type: Int

### Example:

```
nebula> RETURN size("basketballplayer") as size;
+-----+
| size |
+-----+
| 16 |
+-----+
```

### range()

 $range \hbox{() returns a list of integers from $[start,end]$ in the specified steps.}$ 

Syntax: range(<expression\_start>,<expression\_end>[,<expression\_step>])

- expression\_start : An expression of which the result type is int. It represents the starting value start .
- expression\_step: An expression of which the result type is int. It represents the step size step, step is 1 by default.
- · Result type: List

### Example:

### sign()

sign() returns the signum of the given number. If the number is 0, the system returns 0. If the number is negative, the system returns -1. If the number is positive, the system returns 1.

Syntax: sign(<expression>)

- expression: An expression of which the result type is double.
- Result type: Int

#### Example:

#### e()

e() returns the base of the natural logarithm, e (2.718281828459045).

Syntax: e()

• Result type: Double

### Example:

### pi()

pi() returns the mathematical constant pi (3.141592653589793).

Syntax: pi()

· Result type: Double

```
| 3.141592653589793 |
+------
```

### radians()

radians() converts angles to radians.

Syntax: radians(<angle>)
• Result type: Double

Example:

Last update: February 19, 2024

### 4.5.2 Aggregating functions

This topic describes the aggregating functions supported by NebulaGraph.

#### avg()

avg() returns the average value of the argument.

Syntax: avg(<expression>)

· Result type: Double

#### Example:

```
nebula> MATCH (v:player) RETURN avg(v.player.age);
+------+
| avg(v.player.age) |
+-----+
| 33.294117647058826 |
+-----+
```

#### count()

count() returns the number of records.

- $\bullet$  (Native nGQL) You can use count() and GROUP BY together to group and count the number of parameters. Use YIELD to return.
- (OpenCypher style) You can use count() and RETURN. GROUP BY is not necessary.

Syntax: count({<expression> | \*})

- count(\*) returns the number of rows (including NULL).
- Result type: Int

```
# The statement in the following example searches for the people whom `player101` follows and people who follow `player101`, i.e. a bidirectional query.
# Group and count the number of parameters.
nebula> GO FROM "player101" OVER follow BIDIRECT \
         YIELD properties($$).name AS Name
        GROUP BY $-.Name YIELD $-.Name, count(*);
| $-.Name
                         | count(*)
 "LaMarcus Aldridge" | 2
  "Tim Duncan"
  "Marco Belinelli"
  "Manu Ginobili"
  "Boris Diaw"
  "Dejounte Murray"
# Count the number of parameters.
nebula> MATCH (v1:player)-[:follow]-(v2:player) \ WHERE id(v1)== "player101" \
         RETURN v2.player.name AS Name, count(*) as cnt ORDER BY cnt DESC;
  "LaMarcus Aldridge" | 2
 "Tim Duncan"
"Boris Diaw"
  "Manu Ginobili"
 "Dejounte Murray"
"Marco Belinelli"
                         1
```

The preceding example retrieves two columns:

- \$-.Name: the names of the people.
- count(\*): how many times the names show up.

Because there are no duplicate names in the basketballplayer dataset, the number 2 in the column count(\*) shows that the person in that row and player101 have followed each other.

```
# a: The statement in the following example retrieves the age distribution of the players in the dataset.
nebula> LOOKUP ON player
         YIELD player.age As playerage \
| GROUP BY $-.playerage \
         YIELD $-.playerage as age, count(*) AS number \
| ORDER BY $-.number DESC, $-.age DESC;
| age | number
 34 | 4
 33 | 4
30 | 4
29 | 4
38 | 3
...
# b: The statement in the following example retrieves the age distribution of the players in the dataset.
nebula> MATCH (n:player) \
         RETURN n.player.age as age, count(*) as number \
ORDER BY number DESC, age DESC;
| age | number |
34 | 4
 33
30
         4
 29 | 4
```

#### max()

max() returns the maximum value.

Syntax: max(<expression>)

 $\bullet$  Result type: Same as the original argument.

### Example:

### min()

min() returns the minimum value.

#### Syntax: min(<expression>)

• Result type: Same as the original argument.

#### Example:

#### collect()

collect() returns a list containing the values returned by an expression. Using this function aggregates data by merging multiple records or values into a single list.

Syntax: collect(<expression>)

· Result type: List

```
nebula> UNWIND [1, 2, 1] AS a \setminus RETURN a;
| a |
+---+
| 1 |
| 2 |
| 1 |
nebula> UNWIND [1, 2, 1] AS a \setminus
       RETURN collect(a);
| collect(a)
[1, 2, 1]
nebula> UNWIND [1, 2, 1] AS a \
    RETURN a, collect(a), size(collect(a));
| a | collect(a) | size(collect(a))
| 2 | [2]
1 | [1, 1]
| collect(q)
| ["d", "c", "b"] |
nebula> WITH [1, 1, 2, 2] AS coll \backslash UNWIND coll AS x \backslash
        WITH DISTINCT x \
       RETURN collect(x) AS ss;
ss
| [1, 2] |
nebula> MATCH (n:player) \
       RETURN collect(n.player.age);
| collect(n.player.age)
| [32, 32, 34, 29, 41, 40, 33, 25, 40, 37, ...
# The following example aggregates all the players' names by their ages.
nebula> MATCH (n:player) \
       RETURN n.player.age AS age, collect(n.player.name);
| age | collect(n.player.name)
```

#### std()

std() returns the population standard deviation.

 $Syntax: \ \mathsf{std}(\mathsf{<\!expression\!>})$ 

• Result type: Double

#### Example:

### sum()

sum() returns the sum value.

Syntax: sum(<expression>)

• Result type: Same as the original argument.

### Example:

### Aggregating example

Last update: February 19, 2024

### 4.5.3 Built-in string functions

This topic describes the built-in string functions supported by NebulaGraph.

#### **Precautions**

- A string type is used to store a sequence of characters (text). The literal constant is a sequence of characters of any length surrounded by double or single quotes.
- ullet Like SQL, the position index of nGQL starts from ullet , while in C language it starts from ullet .

#### strcasecmp()

strcasecmp() compares string a and b without case sensitivity.

Syntax: strcasecmp(<string\_a>,<string\_b>)

- string\_a , string\_b : Strings to compare.
- Result type: Int
- When string\_a = string\_b, the return value is 0. When string\_a > string\_b, the return value is greater than 0. When string\_a < string\_b, the return value is less than 0.

#### Example:

### lower() and toLower()

lower() and toLower() can both returns the argument in lowercase.

Syntax: lower(<string>) , toLower(<string>)

- string: A specified string.
- Result type: String

### Example:

### upper() and toUpper()

upper() and toUpper() can both returns the argument in uppercase.

Syntax: upper(<string>), toUpper(<string>)

- string: A specified string.
- · Result type: String

### length()

length() returns the length of the given string in bytes.

Syntax: length({<string>|<path>})

- string: A specified string.
- path : A specified path represented by a variable.
- Result type: Int

### Example:

### trim()

trim() removes the spaces at the leading and trailing of the string.

Syntax: trim(<string>)

- string: A specified string.
- Result type: String

### Example:

```
nebula> RETURN trim(" basketball player ");
+------+
| trim(" basketball player ") |
+------+
| "basketball player " |
+--------+
```

## Itrim()

ltrim() removes the spaces at the leading of the string.

Syntax: ltrim(<string>)

- string: A specified string.
- Result type: String

```
nebula> RETURN ltrim(" basketball player ");
+------+
| ltrim(" basketball player ") |
+-----+
```

#### rtrim()

rtrim() removes the spaces at the trailing of the string.

Syntax: rtrim(<string>)

- string: A specified string.
- · Result type: String

### Example:

### left()

left() returns a substring consisting of several characters from the leading of a string.

 $Syntax: \ \mathsf{left}(\mathsf{<string>}, \mathsf{<count>})$ 

- string: A specified string.
- count : The number of characters from the leading of the string. If the string is shorter than count , the system returns the string itself.
- Result type: String

### Example:

```
nebula> RETURN left("basketball_player",6);

+-------+
| left("basketball_player",6) |
+------+
| "basket" |
+------+
```

### right()

right() returns a substring consisting of several characters from the trailing of a string.

Syntax: right(<string>,<count>)

- string: A specified string.
- count : The number of characters from the trailing of the string. If the string is shorter than count , the system returns the string itself.
- Result type: String

#### lpad()

lpad() pads a specified string from the left-side to the specified length and returns the result string.

Syntax: lpad(<string>,<count>,<letters>)

- string: A specified string.
- count: The length of the string after it has been left-padded. If the length is less than that of string, only the length of string characters **from front to back** will be returned.
- letters: A string to be padding from the leading.
- · Result type: String

#### Example:

#### rpad()

rpad() pads a specified string from the right-side to the specified length and returns the result string.

Syntax: rpad(<string>,<count>,<letters>)

- string: A specified string.
- count: The length of the string after it has been right-padded. If the length is less than that of string, only the length of string characters **from front to back** will be returned.
- letters: A string to be padding from the trailing.
- · Result type: String

### Example:

#### substr() and substring()

substr() and substring() return a substring extracting count characters starting from the specified position pos of a specified string.

Syntax: substr(<string>,<pos>,<count>) , substring(<string>,<pos>,<count>)

- string: A specified string.
- pos: The position of starting extract (character index). Data type is int.
- count: The number of characters extracted from the start position onwards.
- · Result type: String

EXPLANATIONS FOR THE RETURN OF SUBSTR() AND SUBSTRING()

- If pos is 0, it extracts from the specified string leading (including the first character).
- If pos is greater than the maximum string index, an empty string is returned.
- If pos is a negative number, BAD\_DATA is returned.
- If count is omitted, the function returns the substring starting at the position given by pos and extending to the end of the string.
- If count is 0, an empty string is returned.
- Using NULL as any of the argument of substr() will cause an issue.

# enCypher compatibility

In openCypher, if a is null, null is returned.

#### Example:

### reverse()

reverse() returns a string in reverse order.

Syntax: reverse(<string>)

- string: A specified string.
- · Result type: String

#### replace()

replace() replaces string a in a specified string with string b.

Syntax: replace(<string>,<substr\_a>,<string\_b>)

- string: A specified string.
- substr\_a: String a.
- string\_b : String b.
- Result type: String

#### Example:

#### split()

split() splits a specified string at string b and returns a list of strings.

Syntax: split(<string>,<substr>)

- string: A specified string.
- substr : String b.
- Result type: List

### Example:

### concat()

concat() returns strings concatenated by all strings.

Syntax: concat(<string1>,<string2>,...)

- The function requires at least two or more strings. If there is only one string, the string itself is returned.
- $\bullet$  If any one of the strings is  $\mbox{\scriptsize NULL}$  ,  $\mbox{\scriptsize NULL}$  is returned.
- Result type: String

#### concat\_ws()

concat ws() returns strings concatenated by all strings that are delimited with a separator.

Syntax: concat\_ws(<separator>,<string1>,<string2>,...)

- The function requires at least two or more strings.
- $\bullet$  If the separator is NULL , the <code>concat\_ws()</code> function returns <code>NULL</code> .
- If the separator is not NULL and there is only one string, the string itself is returned.
- If there is a NULL in the strings, NULL is ignored during the concatenation.

#### Example

#### extract()

extract() uses regular expression matching to retrieve a single substring or all substrings from a string.

 $Syntax: \ \, \mathsf{extract}(\mathsf{`string}\mathsf{'},\mathsf{``ergular\_expression}\mathsf{''})$ 

- string: A specified string
- regular\_expression : A regular expression
- Result type: List

## json\_extract()

json\_extract() converts the specified JSON string to a map.

Syntax: extract(<string>)

- string: A specified string, must be JSON string.
- Result type: Map



- Only Bool, Double, Int, String value and NULL are supported.
- Only depth-1 nested Map is supported now. If nested Map depth is greater than 1, the nested item is left as an empty Map().

#### Example:

Last update: February 19, 2024

### 4.5.4 Built-in date and time functions

NebulaGraph supports the following built-in date and time functions:

Function	Description		
int now()	Returns the current timestamp of the system.		
timestamp timestamp()	Returns the current timestamp of the system.		
date date()	Returns the current UTC date based on the current system.		
time time()	Returns the current UTC time based on the current system.		
datetime datetime()	Returns the current UTC date and time based on the current system.		
map duration()	Returns the period of time. It can be used to calculate the specified time.		

For more information, see Date and time types.

### Examples

nehula> RET	URN now(), timestamp(), date(	') time() dateti	me()·	
	-+	.,,	17.7	-+
now()	timestamp()   date()	time()	datetime()	
	-+			
	1640057560   2021-12-21		· ·	
+	-+	+	+	+

Last update: February 19, 2024

- 203/1069 - 2023 Vesoft Inc.

#### 4.5.5 Schema-related functions

This topic describes the schema-related functions supported by NebulaGraph. There are two types of schema-related functions, one for native nGQL statements and the other for openCypher-compatible statements.

#### For nGQL statements

The following functions are available in YIELD and WHERE clauses of nGQL statements.



Since vertex, edge, vertices, edges, and path are keywords, you need to use AS <alias> to set the alias, such as GO FROM "player100" OVER follow YIELD edge AS e; .

#### ID(VERTEX)

id(vertex) returns the ID of a vertex.

Syntax: id(vertex)

· Result type: Same as the vertex ID.

#### Example:

```
nebula> LOOKUP ON player WHERE player.age > 45 YIELD id(vertex);
+------+
| id(VERTEX) |
+-----+
| "player144" |
| "player140" |
+-------+
```

#### PROPERTIES(VERTEX)

properties(vertex) returns the properties of a vertex.

Syntax: properties(vertex)

• Result type: Map

#### Example:

You can also use the property reference symbols ( \$^ and \$\$) instead of the vertex field in the properties() function to get all properties of a vertex.

- \$^ represents the data of the starting vertex at the beginning of exploration. For example, in GO FROM "player100" OVER follow reversely YIELD properties(\$^), \$^ refers to the vertex player100.
- \$\$ represents the data of the end vertex at the end of exploration.

properties(\$^) and properties(\$\$) are generally used in 60 statements. For more information, see Property reference.

## Caution

You can use properties().sproperty\_name to get a specific property of a vertex. However, it is not recommended to use this method to obtain specific properties because the properties() function returns all properties, which can decrease query performance.

PROPERTIES(EDGE)

properties(edge) returns the properties of an edge.

Syntax: properties(edge)

· Result type: Map

#### Example:

## Caution

You can use properties(edge). property\_name> to get a specific property of an edge. However, it is not recommended to use this method to obtain specific properties because the properties(edge) function returns all properties, which can decrease query performance.

#### TYPE(EDGE)

type(edge) returns the edge type of an edge.

Syntax: type(edge)

• Result type: String

### Example:

#### SRC(EDGE)

src(edge) returns the source vertex ID of an edge.

Syntax: src(edge)

• Result type: Same as the vertex ID.

## Note

The semantics of the query for the starting vertex with src(edge) and properties(§^) are different. src(edge) indicates the starting vertex ID of the edge in the graph database, while properties(§^) indicates the data of the starting vertex where you start to expand the graph, such as the data of the starting vertex player100 in the above GO statement.

#### DST(EDGE)

dst(edge) returns the destination vertex ID of an edge.

Syntax: dst(edge)

• Result type: Same as the vertex ID.

#### Example:



dst(edge) indicates the destination vertex ID of the edge in the graph database.

#### RANK(EDGE)

rank(edge) returns the rank value of an edge.

Syntax: rank(edge)

• Result type: Int

### Example:

#### VERTEX

vertex returns the information of vertices, including VIDs, tags, properties, and values. You need to use AS <alias> to set the alias.

Syntax: vertex

### Example:

#### EDGE

edge returns the information of edges, including edge types, source vertices, destination vertices, ranks, properties, and values. You need to use AS <alias> to set the alias.

#### Syntax: edge

#### Example:

#### VERTICES

vertices returns the information of vertices in a subgraph. For more information, see GET SUBGRAPH.

FDGES

edges returns the information of edges in a subgraph. For more information, see GET SUBGRAPH.

PATH

path returns the information of a path. For more information, see FIND PATH.

#### For statements compatible with openCypher

The following functions are available in RETURN and WHERE clauses of openCypher-compatible statements.

ID()

id() returns the ID of a vertex.

Syntax: id(<vertex>)

• Result type: Same as the vertex ID.

### Example:

TAGS() AND LABELS()

tags() and labels() return the Tag of a vertex.

Syntax: tags(<vertex>) , labels(<vertex>)

• Result type: List

## Example:

PROPERTIES()

properties() returns the properties of a vertex or an edge.

Syntax: properties(<vertex\_or\_edge>)

• Result type: Map

#### Example:

#### TYPE()

type() returns the edge type of an edge.

Syntax: type(<edge>)

· Result type: String

#### Example:

#### SRC()

src() returns the source vertex ID of an edge.

Syntax: src(<edge>)

• Result type: Same as the vertex ID.

### Example:

### DST()

 $\mbox{dst()}$  returns the destination vertex ID of an edge.

Syntax: dst(<edge>)

• Result type: Same as the vertex ID.

#### Example:

### STARTNODE()

startNode() visits a path and returns its information of source vertex ID, including VIDs, tags, properties, and values.

Syntax: startNode(<path>)

### Example:

#### ENDNODE()

endNode() visits a path and returns its information of destination vertex ID, including VIDs, tags, properties, and values.

Syntax: endNode(<path>)

### Example:

#### RANK()

rank() returns the rank value of an edge.

Syntax: rank(<edge>)

• Result type: Int

### Example:

Last update: February 19, 2024

### 4.5.6 List functions

This topic describes the list functions supported by NebulaGraph. Some of the functions have different syntax in native nGQL statements and openCypher-compatible statements.

#### **Precautions**

Like SQL, the position index in nGQL starts from 1, while in the C language it starts from 0.

#### General

RANGE()

range() returns the list containing all the fixed-length steps in [start,end].

Syntax: range(start, end [, step])

- $\bullet\,$  step : Optional parameters. step is 1 by default.
- Result type: List

#### Example:

#### REVERSE()

reverse() returns the list reversing the order of all elements in the original list.

Syntax: reverse(<list>)

• Result type: List

#### Example:

#### TAIL()

tail() returns all the elements of the original list, excluding the first one.

Syntax: tail(<list>)

• Result type: List

### Example:

### HEAD()

head() returns the first element of a list.

Syntax: head(<list>)

• Result type: Same as the element in the original list.

#### Example:

LAST()

last() returns the last element of a list.

Syntax: last(<list>)

• Result type: Same as the element in the original list.

#### Example

#### REDUCE()

reduce() applies an expression to each element in a list one by one, chains the result to the next iteration by taking it as the initial value, and returns the final result. This function iterates each element e in the given list, runs the expression on e, accumulates the result with the initial value, and store the new result in the accumulator as the initial value of the next iteration. It works like the fold or reduce method in functional languages such as Lisp and Scala.

# enCypher compatibility

In openCypher, the reduce() function is not defined. nGQL will implement the reduce() function in the Cypher way.

Syntax: reduce(<accumulator> = <initial>, <variable> IN list> | <expression>)

- accumulator : A variable that will hold the accumulated results as the list is iterated.
- initial: An expression that runs once to give an initial value to the accumulator.
- variable: A variable in the list that will be applied to the expression successively.
- list: A list or a list of expressions.
- ullet expression: This expression will be run on each element in the list once and store the result value in the accumulator.
- Result type: Depends on the parameters provided, along with the semantics of the expression.

```
165
163
 34
       | 31
 34
34
        29
         33
                 167
 34
34
        26
                 160
       34
                 168
 34
         37
                 171
nebula> LOOKUP ON player WHERE player.name == "Tony Parker" YIELD id(vertex) AS VertexID \ | GO FROM \S-.VertexID over follow \
        WHERE properties(edge).degree != reduce(totalNum = 5, n IN range(1, 3) | properties($$).age + totalNum + n) \
        {\tt YIELD\ properties(\$\$).name\ AS\ id,\ properties(\$\$).age\ AS\ age,\ properties(edge).degree\ AS\ degree;}
| id
                        | age | degree |
 "Tim Duncan"
                        | 42 | 95
 "LaMarcus Aldridge" | 33 | 90
                        | 41 | 95
 "Manu Ginobili"
```

#### For nGQL statements

KEYS()

keys() returns a list containing the string representations for all the property names of vertices or edges.

Syntax: keys({vertex | edge})

· Result type: List

#### Example:

```
nebula> LOOKUP ON player \
    WHERE player.age > 45 \
    YTELD keys(vertex);
+------+
| keys(VERTEX) |
+------+
| ["age", "name"] |
| ["age", "name"] |
```

LABELS()

labels() returns the list containing all the tags of a vertex.

Syntax: labels(verte)

• Result type: List

#### Example:

#### For statements compatible with openCypher

KEYS()

keys() returns a list containing the string representations for all the property names of vertices, edges, or maps.

Syntax: keys(<vertex\_or\_edge>)

· Result type: List

#### LABELS()

labels() returns the list containing all the tags of a vertex.

Syntax: labels(<vertex>)

• Result type: List

#### Example:

#### NODES()

nodes() returns the list containing all the vertices in a path.

 $Syntax: \ \mathsf{nodes}(\mathsf{<\!path\!>})$ 

• Result type: List

#### Example:

### RELATIONSHIPS()

relationships() returns the list containing all the relationships in a path.

 $Syntax: \ \textit{relationships}(\textit{<path>})$ 

• Result type: List

### Example:

Last update: February 19, 2024

### 4.5.7 Type conversion functions

This topic describes the type conversion functions supported by NebulaGraph.

#### toBoolean()

toBoolean() converts a string value to a boolean value.

Syntax: toBoolean(<value>)

• Result type: Bool

Example:

#### toFloat()

toFloat() converts an integer or string value to a floating point number.

Syntax: toFloat(<value>)

• Result type: Float

Example:

### toString()

toString() converts non-compound types of data, such as numbers, booleans, and so on, to strings.

Syntax: toString(<value>)

• Result type: String

Example:

```
nebula> RETURN toString(9669) AS int2str, toString(null) AS null2str;
+-------+
| int2str | null2str |
+------+
| "9669" | __NULL_ |
+-----+
```

### toInteger()

toInteger() converts a floating point or string value to an integer value.

Syntax: toInteger(<value>)

• Result type: Int

#### Example:

#### toSet()

toSet() converts a list or set value to a set value.

Syntax: toSet(<value>)
• Result type: Set

#### Example:

```
nebula> RETURN toSet(list[1,2,3,1,2]) AS list2set;
+------+
| list2set |
+------+
| {3, 1, 2} |
+------+
```

#### hash()

hash() returns the hash value of the argument. The argument can be a number, a string, a list, a boolean, null, or an expression that evaluates to a value of the preceding data types.

The source code of the hash() function (MurmurHash2), seed ( 0xc70f6907UL), and other parameters can be found in MurmurHash2.h.

For Java, the hash function operates as follows.

```
MurmurHash2.hash64("to_be_hashed".getBytes(),"to_be_hashed".getBytes().length, 0xc70f6907)
```

Syntax: hash(<string>)

· Result type: Int

#### Example:

Last update: February 19, 2024

### 4.5.8 Conditional expressions

This topic describes the conditional functions supported by NebulaGraph.

#### CASE

The CASE expression uses conditions to filter the parameters. nGQL provides two forms of CASE expressions just like openCypher: the simple form and the generic form.

The CASE expression will traverse all the conditions. When the first condition is met, the CASE expression stops reading the conditions and returns the result. If no conditions are met, it returns the result in the ELSE clause. If there is no ELSE clause and no conditions are met, it returns NULL.

THE SIMPLE FORM OF CASE EXPRESSIONS

#### • Syntax

```
CASE <comparer>
WHEN <value> THEN <result>
[WHEN ...]
[ELSE <default>]
END
```

## Caution

Always remember to end the  $\mbox{CASE}$  expression with an  $\mbox{END}$ .

Parameter	Description	
comparer	A value or a valid expression that outputs a value. This value is used to compare with the $\ensuremath{^{\text{value}}}$ .	
value	It will be compared with the comparer. If the value matches the comparer, then this condition is met.	
result	The result is returned by the CASE expression if the value matches the comparer.	
default	The default is returned by the CASE expression if no conditions are met.	

### • Examples

THE GENERIC FORM OF CASE EXPRESSIONS

#### Syntax

```
CASE
WHEN <condition> THEN <result>
[WHEN ...]
[ELSE <default>]
END
```

Parameter	Description
condition	If the condition is evaluated as true, the result is returned by the CASE expression.
result	The result is returned by the CASE expression if the condition is evaluated as true.
default	The default is returned by the CASE expression if no conditions are met.

#### • Examples

DIFFERENCES BETWEEN THE SIMPLE FORM AND THE GENERIC FORM

To avoid the misuse of the simple form and the generic form, it is important to understand their differences. The following example can help explain them.

The preceding 60 query is intended to output Yes when the player's age is above 35. However, in this example, when the player's age is 36, the actual output is not as expected: It is No instead of Yes.

This is because the query uses the CASE expression in the simple form, and a comparison between the values of \$\$.player.age and \$\$.player.age > 35 is made. When the player age is 36:

- The value of \$\$.player.age is 36. It is an integer.
- \$\$.player.age > 35 is evaluated to be true. It is a boolean.

The values of \$\$.player.age and \$\$.player.age > 35 do not match. Therefore, the condition is not met and No is returned.

# coalesce()

coalesce() returns the first not null value in all expressions.

Syntax: coalesce(<expression\_1>[,<expression\_2>...])

• Result type: Same as the original element.

# Example:

# 4.5.9 Predicate functions

Predicate functions return true or false. They are most commonly used in WHERE clauses.

NebulaGraph supports the following predicate functions:

Functions	Description
exists()	Returns $\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
any()	Returns $\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
all()	Returns $\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
none()	Returns $\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
single()	Returns true if the specified predicate holds for exactly one of the elements in the given list. Otherwise, returns false $\cdot$



 $\ensuremath{\text{NULL}}$  is returned if the list is  $\ensuremath{\text{NULL}}$  or all of its elements are  $\ensuremath{\text{NULL}}$  .

# **P**mpatibility

In openCypher, only function exists() is defined and specified. The other functions are implement-dependent.

### Syntax

```
<predicate>(<variable> IN <list> WHERE <condition>)
```

# Examples

# 4.5.10 Geography functions

 $Geography \ functions \ are \ used \ to \ generate \ or \ perform \ operations \ on \ the \ value \ of \ the \ geography \ data \ type.$ 

For descriptions of the geography data types, see Geography.

#### **Descriptions**

Function	Return Type	Description
ST_Point(longitude, latitude)	GEOGRAPHY	Creates the geography that contains a point.
ST_GeogFromText(wkt_string)	GEOGRAPHY	Returns the geography corresponding to the input WKT string.
ST_ASText(geography)	STRING	Returns the WKT string of the input geography.
ST_Centroid(geography)	GEOGRAPHY	Returns the centroid of the input geography in the form of the single point geography.
ST_ISValid(geography)	BOOL	Returns whether the input geography is valid.
ST_Intersects(geography_1, geography_2)	BOOL	Returns whether geography_1 and geography_2 have intersections.
ST_Covers(geography_1, geography_2)	B00L	Returns whether geography_1 completely contains geography_2. If there is no point outside geography_1 in geography_2, return True.
ST_CoveredBy(geography_1, geography_2)	BOOL	Returns whether geography_2 completely contains geography_1.If there is no point outside geography_2 in geography_1, return True.
ST_DWithin(geography_1, geography_2, distance)	BOOL	If the distance between one point (at least) in geography_1 and one point in geography_2 is less than or equal to the distance specified by the distance parameter (measured by meters), return True.
ST_Distance(geography_1, geography_2)	FLOAT	Returns the smallest possible distance (measured by meters) between two non-empty geographies.
S2_CellIdFromPoint(point_geography)	INT	Returns the $S2$ Cell ID that covers the point geography.
S2_CoveringCellIds(geography)	ARRAY <int64></int64>	Returns an array of S2 Cell IDs that cover the input geography.

# Examples

```
nebula> RETURN ST_ISValid(ST_GeogFromText("POINT(3 8)"));
| ST_ISValid(ST_GeogFromText("POINT(3 8)"))
| true
nebula> RETURN ST_Intersects(ST_GeogFromText("LineString(0 1,1 0)"),ST_GeogFromText("LineString(0 0,1 1)"));
ST_Intersects(ST_GeogFromText("LineString(0 1,1 0)"),ST_GeogFromText("LineString(0 0,1 1)"))
nebula> RETURN ST_Covers(ST_GeogFromText("POLYGON((0 0,10 0,10 10,0 10,0 0))"),ST_Point(1,2));
| ST_Covers(ST_GeogFromText("POLYGON((0 0,10 0,10 10,0 10,0 0))"),ST_Point(1,2)) |
nebula> RETURN ST_CoveredBy(ST_Point(1,2),ST_GeogFromText("POLYGON((0 0,10 0,10 10,0 10,0 0))"));
| ST_CoveredBy(ST_Point(1,2),ST_GeogFromText("POLYGON((0 0,10 0,10 10,0 10,0 0))")) |
| true
nebula> RETURN ST_dwithin(ST_GeogFromText("Point(0 0)"),ST_GeogFromText("Point(10 10)"),20000000000.0);
| \  \, \mathsf{ST\_dwithin}(\mathsf{ST\_GeogFromText}("Point(0\ 0)"), \mathsf{ST\_GeogFromText}("Point(10\ 10)"), 20000000000) \  \, | \  \, \mathsf{ST\_dwithin}(\mathsf{ST\_GeogFromText}("Point(10\ 10)"), \mathsf{ST\_GeogFromText}("Point(10\ 10)"), \mathsf{ST\_GeogFromText}("Point
nebula> RETURN ST_Distance(ST_GeogFromText("Point(0 0)"),ST_GeogFromText("Point(10 10)"));
| ST_Distance(ST_GeogFromText("Point(0 0)"),ST_GeogFromText("Point(10 10)"))
| 1.5685230187677438e+06
nebula> RETURN S2_CellIdFromPoint(ST_GeogFromText("Point(1 1)"));
| S2_CellIdFromPoint(ST_GeogFromText("Point(1 1)")) |
| 1153277837650709461
nebula> RETURN S2_CoveringCellIds(ST_GeogFromText("POLYGON((0 1, 1 2, 2 3, 0 1))"));
| S2_CoveringCellIds(ST_GeogFromText("POLYGON((0 1, 1 2, 2 3, 0 1))"))
  [ [1152391494368201343, 1153466862374223872, 1153554823304445952, 1153836298281156608, 1153959443583467520, 1154240918560178176, 1160503736791990272, 1160591697722212352] |
```

# 4.5.11 User-defined functions

# OpenCypher compatibility

 $User\text{-}defined \ functions \ (UDF) \ and \ storage \ processes \ are \ not \ yet \ supported \ nor \ designed \ in \ Nebula Graph \ 3.4.1.$ 

# 4.6 General queries statements

#### 4.6.1 MATCH

The MATCH statement supports searching based on pattern matching.

A MATCH statement defines a search pattern and uses it to match data stored in NebulaGraph and to retrieve them in the form defined in the RETURN clause.

The examples in this topic use the basketballplayer dataset as the sample dataset.

#### **Syntax**

The syntax of MATCH is relatively more flexible compared with that of other query statements such as 60 or LOOKUP. The path type of the MATCH statement is trail. That is, only vertices can be repeatedly visited in the graph traversal. Edges cannot be repeatedly visited. For details, see path. But generally, it can be summarized as follows.

MATCH <pattern> [<clause\_1>] RETURN <output> [<clause\_2>];

- pattern: For the detailed description of patterns, see <u>Patterns</u>. The MATCH statement supports matching one or multiple patterns. Multiple patterns are separated by commas (,). For example: (a)-[]->(b),(c)-[]->(d).
- clause\_1: The WHERE, WITH, UNWIND, and OPTIONAL MATCH clauses are supported, and the MATCH clause can also be used.
- output: Define the output to be returned. You can rename the output column by using AS.
- clause\_2: The ORDER BY and LIMIT clauses are supported.

#### Limitations

#### L Jacy version compatibility

Starting from NebulaGraph version 3.0.0, in order to distinguish the properties of different tags, you need to specify a tag name when querying properties. The original statement RETURN variable\_name.property\_name is changed to RETURN variable\_name.<a href="mailto:retarger-name">retarger-name</a>.

#### Q Note

- Currently the match statement cannot find dangling edges.
- It is not supported to traverse the specified Tag and Edge Type at the same time when there is no index. For example, executing MATCH (v:player)-[e:follow]->() RETURN e LIMIT N an error will occur.

When no index has been created, the MATCH statements is only supported in the following cases. When you get an error executing the MATCH statement, you can create and rebuild the index and then execute the MATCH statement.

- In a valid MATCH statement, the VID of a specific vertex must be specified with the id() function in the WHERE clause. There is no need to create an index.
- When traversing all vertices o r edges with MATCH, such as MATCH (v) RETURN v LIMIT N, MATCH ()-[e]->() RETURN e LIMIT N.
- When traversing all vertices of the specified Tag, such as MATCH (v:player) RETURN v LIMIT N.
- When traversing all edges of the specified Edge Type(edges must have direction), such as MATCH ()-[e:follow]->() RETURN e LIMIT N.

- 224/1069 - 2023 Vesoft Inc.

#### Using patterns in MATCH statements

#### CREATE INDEXES

```
# The following example creates an index on both the name property of the tag player and the edge type follow.
nebula> CREATE TAG INDEX IF NOT EXISTS player_index_1 ON player(name(20));
nebula> CREATE EDGE INDEX IF NOT EXISTS follow_index on follow();
# The following example rebuilds the index.
nebula> REBUILD TAG INDEX player_index_1;
 New Job Id
121
nebula> REBUILD EDGE INDEX follow_index;
New Job Id
| 122
# The following example makes sure the index is rebuilt successfully.
nebula> SHOW JOB 121;
 Job Id(TaskId) | Command(Dest)
                                           Status
                                                          | Start Time
                                                                                          | Stop Time
                                                                                                                          | Error Code
                                             "FINISHED" | 2021-05-27T02:18:02.000000 |
 121
                     "REBUILD TAG INDEX" |
                                                                                           2021-05-27T02:18:02.000000
                                                                                                                            "SUCCEEDED"
                                                           2021-05-27T02:18:02.000000
                                                                                            2021-05-27T02:18:02.000000
                                              "FINISHED"
                                                                                                                            "SUCCEEDED"
 0
                      "storaged1"
                     "storaged0"
                                             "FTNTSHED"
                                                           2021-05-27T02:18:02 000000
                                                                                           2021-05-27T02:18:02 000000
                                                                                                                            "SUCCEEDED"
                                             "FINISHED"
                     "storaged2"
                                                           2021-05-27T02:18:02.000000
                                                                                                                            "SUCCEEDED'
                                                                                            2021-05-27T02:18:02.000000
  "Total:3"
                     "Succeeded:3"
                                             "Failed:0"
                                                           "In Progress:0"
nebula> SHOW JOB 122;
 Job Id(TaskId) | Command(Dest)
                                            Status
                                                                                           | Stop Time
                                                                                                                           | Error Code
                     "REBUILD EDGE INDEX" | "FINISHED" | 2021-05-27T02:18:11.000000 |
 122
                                                                                             2021-05-27T02:18:11.000000
                                                                                                                              "SUCCEEDED"
                                              "FINISHED"
                                                            2021-05-27T02:18:11.000000
                                                                                             2021-05-27T02:18:21.000000
                                                                                                                             "SUCCEEDED"
                     "storaged1"
 1
                     "storaged0"
                                              "FINISHED"
                                                            2021-05-27T02:18:11.000000
                                                                                             2021-05-27T02:18:21.000000
                                                                                                                             "SUCCEEDED"
                      "storaged2"
                                              "FINISHED"
                                                            2021-05-27T02:18:11.000000
                                                                                             2021-05-27T02:18:21.000000
                                                                                                                             "SUCCEEDED"
  "Total:3"
                    "Succeeded:3"
                                             "Failed:0" | "In Progress:0"
```

### MATCH VERTICES

# Lacy version compatibility

As of version 3.0.0, nGQL support MATCH (v) RETURN v LIMIT n, there is no need to create an index. But you must use LIMIT to limit the number of output results.

 $nGQL \ still \ does \ not \ support \ MATCH \ (v) \ RETURN \ v$  .

You can use a user-defined variable in a pair of parentheses to represent a vertex in a pattern. For example: (v).

MATCH TAGS



In NebulaGraph versions earlier than 3.0.0, the prerequisite for matching a tag is that the tag itself has an index or a certain property of the tag has an index. As of version 3.0.0, there is no need to create an index for matching a tag, but you need to use LIMIT to limit the number of output results.

You can specify a tag with :<tag\_name> after the vertex in a pattern.

To match vertices with multiple tags, use colons (:).



It is not yet supported to add property conditions when matching vertices with multiple tags.

For example, the statement match (v1:player:team) where v1.player.name="Tim Duncan" return v1 limit 10; does not work.

MATCH VERTEX PROPERTIES



The prerequisite for matching a vertex property is that the tag itself has an index of the corresponding property. Otherwise, you cannot execute the MATCH statement to match the property.

You can specify a vertex property with {prop\_name>: cyrop\_value>} after the tag in a pattern.

The WHERE clause can do the same thing:

# OpenCypher compatibility

In openCypher 9, = is the equality operator. However, in nGQL, == is the equality operator and = is the assignment operator (as in C++ or Java).

Use the WHERE clause to directly get all the vertices with the vertex property value Tim Duncan.

```
nebula> MATCH (v) \
WITH v, properties(v) as props, keys(properties(v)) as kk \
```

#### MATCH VIDS

You can use the VID to match a vertex. The id() function can retrieve the VID of a vertex.

To match multiple VIDs, use WHERE id(v) IN [vid\_list].

#### MATCH CONNECTED VERTICES

You can use the -- symbol to represent edges of both directions and match vertices connected by these edges.

# Lacy version compatibility

In nGQL 1.x, the -- symbol is used for inline comments. Starting from nGQL 2.x, the -- symbol represents an incoming or outgoing edge.

You can add a > or < to the -- symbol to specify the direction of an edge.

In the following example,  $\rightarrow$  represents an edge that starts from v and points to v2. To v, this is an outgoing edge, and to v2 this is an incoming edge.

To query the properties of the target vertices, use the  $\mbox{\em {\scriptsize CASE}}$  expression.

```
| "Manu Ginobili" |
| "Manu Ginobili" |
| "Spurs" |
| "Dejounte Murray" |
...
```

To extend the pattern, you can add more vertices and edges.

If you do not need to refer to a vertex, you can omit the variable representing it in the parentheses.

#### MATCH PATHS

Connected vertices and edges form a path. You can use a user-defined variable to name a path as follows.

# PrenCypher compatibility

In nGQL, the @ symbol represents the rank of an edge, but openCypher has no such concept.

#### MATCH EDGES

# LenCypher compatibility

In NebulaGraph versions earlier than 3.0.0, the prerequisite for matching a edge is that the edge itself has an index or a certain property of the edge has an index. As of version 3.0.0, there is no need to create an index for matching a edge, but you need to use LIMIT to limit the number of output results and you must specify the direction of the edge.

#### MATCH EDGE TYPES

Just like vertices, you can specify edge types with :-edge\_type> in a pattern. For example: -[e:follow]-.

# FrenCypher compatibility

In NebulaGraph versions earlier than 3.0.0, the prerequisite for matching a edge type is that the edge type itself has an index or a certain property of the edge type has an index. As of version 3.0.0, there is no need to create an index for matching a edge type, but you need to use LIMIT to limit the number of output results and you must specify the direction of the edge.

MATCH EDGE TYPE PROPERTIES



The prerequisite for matching an edge type property is that the edge type itself has an index of the corresponding property. Otherwise, you cannot execute the MATCH statement to match the property.

You can specify edge type properties with {sprop\_name>: sprop\_value>} in a pattern. For example: [e:follow{likeness:95}].

Use the WHERE clause to directly get all the edges with the edge property value 90.

MATCH MULTIPLE EDGE TYPES

The | symbol can help matching multiple edge types. For example: [e:follow|:serve] . The English colon (:) before the first edge type cannot be omitted, but the English colon before the subsequent edge type can be omitted, such as [e:follow|serve] .

#### Q Note

It is not yet supported to add property conditions when matching data with multiple tags and multiple edge types at the same time.

For example, the statement MATCH (v)-[e:follow|serve]->(v2) where v.player.name="Tim Duncan" RETURN e limit 10; does not work. (v) represents a vertex with all its tags.

#### MATCH MULTIPLE EDGES

You can extend a pattern to match multiple edges in a path.

#### MATCH FIXED-LENGTH PATHS

You can use the  $:<edge_type>^*<hop>$  pattern to match a fixed-length path. hop must be a non-negative integer.

If hop is 0, the pattern will match the source vertex of the path.

#### Q Note

When you conditionally filter on multi-hop edges, such as -[e:follow\*2]->, note that the e is a list of edges instead of a single edge.

For example, the following statement is correct from the syntax point of view which may not get your expected query result, because the e is a list without the .degree property.

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*2]->(v2) \
WHERE e.degree > 1 \
RETURN DISTINCT v2 AS Friends;
```

The correct statement is as follows:

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*2]->(v2) \
WHERE ALL(e_ in e WHERE e_.degree > 0) \
RETURN DISTINCT v2 AS Friends;
```

Further, the following statement is for filtering the properties of the first-hop edge in multi-hop edges:

```
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*2]->(v2) \
WHERE e[0].degree > 98 \
RETURN DISTINCT v2 AS Friends;
```

# MATCH VARIABLE-LENGTH PATHS

You can use the :<edge\_type>\*[minHop..maxHop] pattern to match variable-length paths. minHop and maxHop are optional and default to 1 and infinity respectively.



When setting bounds, at least one of minHop and maxHop exists.

# Caution

If maxHop is not set, it may cause the Graph service to OOM, execute this command with caution.

```
nebula > MATCH \ p=(v:player\{name:"Tim \ Duncan"\}) - [e:follow^*] -> (v2) \ \setminus \ P(v) - P(v)
                            RETURN v2 AS Friends;
 | Friends
 | ("player125" :player{age: 41, name: "Manu Ginobili"})
 ("player101" :player{age: 36, name: "Tony Parker"})
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*1..3]->(v2) \
                            RETURN v2 AS Friends;
 | ("player101" :player{age: 36, name: "Tony Parker"})
       ("player125" :player{age: 41, name: "Manu Ginobili"})
 ("player100" :player{age: 42, name: "Tim Duncan"})
nebula> MATCH p=(v:player{name:"Tim Duncan"})-[e:follow*1..]->(v2) \
                            RETURN v2 AS Friends;
 Friends
     ("player125" :player{age: 41, name: "Manu Ginobili"})
       ("player101" :player{age: 36, name: "Tony Parker"})
 | ("player100" :player{age: 42, name: "Tim Duncan"})
```

You can use the DISTINCT keyword to aggregate duplicate results.

If minHop is 0, the pattern will match the source vertex of the path. Compared to the preceding statement, the following example uses 0 as the minHop. So in the following result set, "Tim Duncan" is counted one more time than it is in the preceding result set because it is the source vertex.

MATCH VARIABLE-LENGTH PATHS WITH MULTIPLE EDGE TYPES

You can specify multiple edge types in a fixed-length or variable-length pattern. In this case, hop, minHop, and maxHop take effect on all edge types.

MATCH MULTIPLE PATTERNS

You can separate multiple patterns with commas (,).

MATCH SHORTEST PATHS

The allShortestPaths function can be used to find all shortest paths between two vertices.

The shortestPath function can be used to find a single shortest path between two vertices.

# Retrieve with multiple match

Multiple MATCH can be used when different patterns have different filtering criteria and return the rows that exactly match the pattern.

#### Retrieve with optional match

See OPTIONAL MATCH.



In NebulaGraph, the performance and resource usage of the MATCH statement have been optimized. But we still recommend to use 60, LOOKUP, |, and FETCH instead of MATCH when high performance is required.

# 4.6.2 OPTIONAL MATCH



The feature is still in beta. It will continue to be optimized.

The OPTIONAL MATCH clause is used to search for the pattern described in it. OPTIONAL MATCH matches patterns against your graph database, just like MATCH does. The difference is that if no matches are found, OPTIONAL MATCH will use a null for missing parts of the pattern.

#### **OpenCypher Compatibility**

This topic applies to the openCypher syntax in nGQL only.

#### Limitations

The WHERE clause cannot be used in an OPTIONAL MATCH clause.

#### Example

The example of the use of  ${\tt OPTIONAL}$  MATCH in the MATCH statement is as follows:

```
nebula> MATCH (m)-[]->(n) WHERE id(m)=="player100" \ OPTIONAL MATCH (n)-[]->(l) \
         RETURN id(m), id(n), id(l);
| id(m)
                  id(n)
                                 | id(l)
  "player100"
                   "team204"
                                    NULL
  "player100"
                   "player101"
                                   "team204"
  "player100"
                   "player101"
                                   "team215"
  "player100"
                   "player101"
                                   "player100'
  "player100"
                   "player101"
                                   "player102'
  "player100"
                   "player101"
                                   "player125"
"team204"
                  "player125"
  "player100"
  "player100"
                  "player125"
                                   "player100"
```

Using multiple MATCH instead of OPTIONAL MATCH returns rows that match the pattern exactly. The example is as follows:

```
nebula> MATCH (m)-[]->(n) WHERE id(m)=="player100" \
        MATCH (n)-[]->(l) \
RETURN id(m),id(n),id(l);
| id(m)
                 id(n)
                               | id(l)
 "player100"
                  "player101"
                                  "team204"
                                  "team215"
 "player100"
                  "player101"
 "player100"
                                  "player100"
                  "player101"
  "player100"
                  "player101"
                                  "player102"
  "player100"
                  "player101"
                                  "player125"
                 "player125"
"player125"
  "player100"
 "player100"
                                  "player100"
```

#### **4.6.3 LOOKUP**

The LOOKUP statement traverses data based on indexes. You can use LOOKUP for the following purposes:

- Search for the specific data based on conditions defined by the WHERE clause.
- List vertices with a tag: retrieve the VID of all vertices with a tag.
- · List edges with an edge type: retrieve the source vertex IDs, destination vertex IDs, and ranks of all edges with an edge type.
- Count the number of vertices or edges with a tag or an edge type.

#### OpenCypher compatibility

This topic applies to native nGQL only.

#### Precautions

- Correct use of indexes can speed up queries, but indexes can dramatically reduce the write performance. The performance can be greatly reduced. **DO NOT** use indexes in production environments unless you are fully aware of their influences on your service.
- If the specified property is not indexed when using the LOOKUP statement, NebulaGraph randomly selects one of the available indexes

For example, the tag player has two properties, name and age. Both the tag player itself and the property name have indexes, but the property age has no indexes. When running LOOKUP ON player WHERE player.age = 36 YIELD player.name; NebulaGraph randomly uses one of the indexes of the tag player and the property name.

# Lyacy version compatibility

Before the release 2.5.0, if the specified property is not indexed when using the LOOKUP statement, NebulaGraph reports an error and does not use other indexes.

#### Prerequisites

Before using the LOCKUP statement, make sure that at least one index is created. If there are already related vertices, edges, or properties before an index is created, the user must rebuild the index after creating the index to make it valid.

#### **Syntax**

- WHERE <expression>: filters data with specified conditions. Both AND and OR are supported between different expressions. For more information, see WHERE.
- YIELD: Define the output to be returned. For details, see YIELD.
- DISTINCT: Aggregate the output results and return the de-duplicated result set.
- AS: Set an alias.

#### Limitations of using WHERE in LOOKUP

The WHERE clause in a LOOKUP statement does not support the following operations:

- \$- and \$^.
- Filter rank().
- In relational expressions, operators are not supported to have field names on both sides, such as tagName.prop1> tagName.prop2.
- · Nested AliasProp expressions in operation expressions and function expressions are not supported.
- The XOR operation is not supported.
- String operations other than STARTS WITH are not supported.
- · Graph patterns.

#### Retrieve vertices

The following example returns vertices whose name is Tony Parker and the tag is player.

```
nebula> CREATE TAG INDEX IF NOT EXISTS index_player ON player(name(30), age);
nebula> REBUILD TAG INDEX index_player;
| New Job Id |
nebula> LOOKUP ON player \
        WHERE player.name == "Tony Parker" \
YIELD id(vertex);
| id(VERTEX)
| "player101"
nebula> LOOKUP ON player \
    WHERE player.name == "Tony Parker" \
        YIELD properties(vertex).name AS name, properties(vertex).age AS age;
                 age
| "Tony Parker" | 36 |
nebula> LOOKUP ON player \
        WHERE player.age > 45 \
YIELD id(vertex);
| id(VERTEX)
 "player144"
 "player140"
nebula> LOOKUP ON player \
WHERE player.name STARTS WITH "B" \
        AND player.age IN [22,30] \
        YIELD properties(vertex).name, properties(vertex).age;
| properties(VERTEX).name | properties(VERTEX).age |
 "Blake Griffin"
                            30
nebula> LOOKUP ON player \
         WHERE player.name == "Kobe Bryant"\
        YIELD id(vertex) AS VertexID, properties(vertex).name AS name |\ GO FROM \$-.VertexID OVER serve \
        YIELD $-.name, properties(edge).start_year, properties(edge).end_year, properties($$).name;
              | properties(EDGE).start_year | properties(EDGE).end_year | properties($$).name |
$-.name
 "Kobe Bryant" | 1996
                                                                               "Lakers"
                                                 2016
```

# Retrieve edges

The following example returns edges whose degree is 90 and the edge type is follow.

```
nebula> CREATE EDGE INDEX IF NOT EXISTS index_follow ON follow(degree);
nebula> REBUILD EDGE INDEX index_follow;
62
nebula> LOOKUP ON follow \
        WHERE follow.degree == 90 YIELD edge AS e;
e
| [:follow "player109"->"player125" @0 {degree: 90}} |
| [:follow "player118"->"player120" @0 {degree: 90}} |
| [:follow "player118"->"player131" @0 {degree: 90}] |
nebula> LOOKUP ON follow \
        WHERE follow.degree == 90 \
        YIELD properties(edge).degree;
| SrcVID | DstVID | Ranking | properties(EDGE).degree |
nebula> LOOKUP ON follow \
        WHERE follow.degree == 60 \
        VIELD dst(edge) AS DstVID, properties(edge).degree AS Degree |\
GO FROM $-.DstVID OVER serve \
        YIELD $-.DstVID, properties(edge).start_year, properties(edge).end_year, properties($$).name;
| $-.DstVID | properties(EDGE).start_year | properties(EDGE).end_year | properties($$).name |
 "player105" | 2010
"player105" | 2009
                                                 2018
                                                                                  "Spurs"
                                                                                  "Cavaliers"
  "player105" | 2018
                                                  2019
                                                                                 "Raptors"
```

# List vertices or edges with a tag or an edge type

To list vertices or edges with a tag or an edge type, at least one index must exist on the tag, the edge type, or its property.

For example, if there is a player tag with a name property and an age property, to retrieve the VID of all vertices tagged with player, there has to be an index on the player tag itself, the name property, or the age property.

• The following example shows how to retrieve the VID of all vertices tagged with player.

• The following example shows how to retrieve the source Vertex IDs, destination vertex IDs, and ranks of all edges of the follow edge type.

#### Count the numbers of vertices or edges

The following example shows how to count the number of vertices tagged with player and edges of the follow edge type.

#### O Note

You can also use  $\ensuremath{\mathsf{SHOW}}$  STATS to count the numbers of vertices or edges.

# 4.6.4 GO

60 traverses in a graph with specified filters and returns results.

# OpenCypher compatibility

This topic applies to native nGQL only.

# Syntax

```
<return_list> ::=
    <col_name> [AS <col_alias>] [, <col_name> [AS <col_alias>] ...]
```

• <N> {STEP|STEPS}: specifies the hop number. If not specified, the default value for N is one. When N is zero, NebulaGraph does not traverse any edges and returns nothing.



The path type of the 60 statement is walk, which means both vertices and edges can be repeatedly visited in graph traversal. For more information, see Path.

- M TO N {STEP|STEPS}: traverses from M to N hops. When M is zero, the output is the same as that of M is one. That is, the output of 60 0 TO 2 and 60 1 TO 2 are the same.
- <vertex\_list>: represents a list of vertex IDs separated by commas, or a special place holder \$-.id. For more information, see
  Pipe.
- <edge\_type\_list> : represents a list of edge types which the traversal can go through.
- REVERSELY | BIDIRECT: defines the direction of the query. By default, the 60 statement searches for outgoing edges of <vertex\_list>.

  If REVERSELY is set, 60 searches for incoming edges. If BIDIRECT is set, 60 searches for edges of both directions.
- WHERE <expression>: specifies the traversal filters. You can use the WHERE clause for the source vertices, the edges, and the
  destination vertices. You can use it together with AND, OR, NOT, and XOR. For more information, see WHERE.

Note

- There are some restrictions for the WHERE clause when you traverse along with multiple edge types. For example, WHERE edge1.prop1 > edge2.prop2 is not supported.
- The GO statement is executed by traversing all the vertices and then filtering according to the filter condition.
- YIELD [DISTINCT] <return\_List>: defines the output to be returned. It is recommended to use the Schema-related functions to fill in <return\_List>. src(edge), dst(edge), type(edge)), rank(edge), etc., are currently supported, while nested functions are not. For more information, see YIELD.
- SAMPLE <sample\_list>: takes samples from the result set. For more information, see SAMPLE.
- limit\_by\_list\_clause> : limits the number of outputs during the traversal process. For more information, see LIMIT.
- GROUP BY: groups the output into subgroups based on the value of the specified property. For more information, see GROUP BY. After grouping, you need to use YIELD again to define the output that needs to be returned.
- ORDER BY: sorts outputs with specified orders. For more information, see ORDER BY.

Note

When the sorting method is not specified, the output orders can be different for the same query.

 $\bullet \ \ \text{LIMIT} \ \ [\text{``entropy}, ] \ \ \text{``number\_rows'}]: limits \ the \ number \ of \ rows \ of \ the \ output. For \ more \ information, see \ \underline{\text{LIMIT}}.$ 

#### Examples

```
| "team204" |
# The following example returns the friends of player 102 with 2 hops. nebula> GO 2 STEPS FROM "player102" OVER follow YIELD dst(edge);
dst(EDGE)
  "player101"
  "player125"
  "player100"
  "player102"
  "player125"
YIELD DISTINCT properties($$).name AS team_name, properties(edge).start_year AS start_year, properties($^).name AS player_name;
| team_name
                      | start_year | player_name
                      1997
                                       "Tim Duncan"
  "Spurs'
  "Trail Blazers" | 2006
                                        "LaMarcus Aldridge"
  "Spurs"
                                      "LaMarcus Aldridge"
                      2015
# The following example traverses along with multiple edge types. If there is no value for a property, the output is `NULL`.
nebula> GO FROM "player100" OVER follow, serve \
YIELD properties(edge).degree, properties(edge).start_year;
| properties(EDGE).degree | properties(EDGE).start year
  95
                                 __NULL_
                                   _NULL_
  95
  __NULL__
                                 1997
# The following example returns the neighbor vertices in the incoming direction of player 100.
nebula> GO FROM "player100" OVER follow REVERSELY \
YIELD src(edge) AS destination;
| destination |
  "player101"
| "player102"
# This MATCH query shares the same semantics with the preceding GO query. nebula> MATCH (v)<-[e:follow]- (v2) WHERE id(v) == 'player100' \ RETURN id(v2) AS destination;
destination
  "player101"
  "player102"
# The following example retrieves the friends of player 100 and the teams that they serve. nebula> GO FROM "player100" OVER follow REVERSELY \ YIELD src(edge) AS id | \
         GO FROM $-.id OVER serve \
WHERE properties($^\).age > 20 \
          YIELD properties($^).name AS FriendOf, properties($$).name AS Team;
| FriendOf
                          Team
  "Boris Diaw"
                          | "Spurs'
                            "Jazz"
  "Boris Diaw"
"Boris Diaw"
                          "Suns"
RETURN v2.player.name AS FriendOf, v3.team.name AS Team;
| FriendOf
                          Team
  "Boris Diaw"
                             "Spurs'
  "Boris Diaw"
                             "Jazz"
| "Boris Diaw"
                          i "Suns"
# The following example retrieves the friends of player 100 within 1 or 2 hops. nebula> G0 1 T0 2 STEPS FROM "player100" OVER follow \setminus
```

YIELD dst(edge) AS destination;

```
| destination |
 player101"
 player125"
destination
| "player100"
| "player102"
# The following example the outputs according to age.
nebula> G0 2 STEPS FROM "player100" OVER follow \
    YIELD src(edge) AS src, dst(edge) AS dst, properties($$).age AS age \
    | GROUP BY $-.dst \
    YIELD $-.dst AS dst, collect_set($-.src) AS src, collect($-.age) AS age;
dst
              src
                                                          age
  "player105" | ["player101"] | [41] | "player100" | ["player125", "player101"] | [42, 42] | "player102" | ["player101"] | [33]
src
                | $a.dst | follow._src | follow._dst |
  "player100" | "player125" | "player100" | "player101" | "player100" | "player101" | "player100" | "player125"
# The following example determines if $$.player.name IS NOT EMPTY.
nebula> GO FROM "player100" OVER follow WHERE properties($$).name IS NOT EMPTY YIELD dst(edge);
 | follow._dst
  "player125"
"player101"
```

# 4.6.5 FETCH

The FETCH statement retrieves the properties of the specified vertices or edges.

# **OpenCypher Compatibility**

This topic applies to native nGQL only.

#### Fetch vertex properties

SYNTAX

```
FETCH PROP ON {<tag_name>[, tag_name ...] | *} 
<vid>[, vid ...]
YIELD [DISTINCT] <return_list> [AS <alias>];
```

Parameter	Description
tag_name	The name of the tag.
*	Represents all the tags in the current graph space.
vid	The vertex ID.
YIELD	Define the output to be returned. For details, see $\ensuremath{^{{\tiny{\mbox{\scriptsize{VIELD}}}}}}\xspace$ .
AS	Set an alias.

FETCH VERTEX PROPERTIES BY ONE TAG

Specify a tag in the  $\mbox{\sc FETCH}$  statement to fetch the vertex properties by that tag.

FETCH SPECIFIC PROPERTIES OF A VERTEX

Use a YIELD clause to specify the properties to be returned.

FETCH PROPERTIES OF MULTIPLE VERTICES

Specify multiple VIDs (vertex IDs) to fetch properties of multiple vertices. Separate the VIDs with commas.

FETCH VERTEX PROPERTIES BY MULTIPLE TAGS

Specify multiple tags in the FETCH statement to fetch the vertex properties by the tags. Separate the tags with commas.

```
# The following example creates a new tag t1.
nebula> CREATE TAG IF NOT EXISTS t1(a string, b int);
```

You can combine multiple tags with multiple VIDs in a FETCH statement.

FETCH VERTEX PROPERTIES BY ALL TAGS

Set an asterisk symbol \* to fetch properties by all tags in the current graph space.

#### Fetch edge properties

SYNTAX

```
FETCH PROP ON <edge_type> <src_vid> -> <dst_vid>[@<rank>] [, <src_vid> -> <dst_vid> ...]
YIELD <output>;
```

Parameter	Description
edge_type	The name of the edge type.
src_vid	The VID of the source vertex. It specifies the start of an edge.
dst_vid	The VID of the destination vertex. It specifies the end of an edge.
rank	The rank of the edge. It is optional and defaults to 0. It distinguishes an edge from other edges with the same edge type, source vertex, destination vertex, and rank.
YIELD	Define the output to be returned. For details, see YIELD.

FETCH ALL PROPERTIES OF AN EDGE

The following statement fetches all the properties of the serve edge that connects vertex "player100" and vertex "team204".

FETCH SPECIFIC PROPERTIES OF AN EDGE

Use a YIELD clause to fetch specific properties of an edge.

#### FETCH PROPERTIES OF MULTIPLE EDGES

Specify multiple edge patterns ( $\sc vid > -> \sc vid > [@\sc vid > ]$ ) to fetch properties of multiple edges. Separate the edge patterns with commas.

#### Fetch properties based on edge rank

If there are multiple edges with the same edge type, source vertex, and destination vertex, you can specify the rank to fetch the properties on the correct edge.

#### Use FETCH in composite queries

A common way to use FETCH is to combine it with native nGQL such as 60.

The following statement returns the degree values of the follow edges that start from vertex "player101".

Or you can use user-defined variables to construct similar queries.

For more information about composite queries, see Composite queries (clause structure).

# 4.6.6 SHOW

#### **SHOW CHARSET**

The  $\mbox{SHOW CHARSET}$  statement shows the available character sets.

Currently available types are utf8 and utf8mb4. The default charset type is utf8. NebulaGraph extends the uft8 to support fourbyte characters. Therefore utf8 and utf8mb4 are equivalent.

#### SYNTAX

SHOW CHARSET;

#### EXAMPLE



Parameter	Description
Charset	The name of the character set.
Description	The description of the character set.
Default collation	The default collation of the character set.
Maxlen	The maximum number of bytes required to store one character.

#### **SHOW COLLATION**

The  $\mbox{\scriptsize SHOW}$  COLLATION statement shows the collations supported by NebulaGraph.

Currently available types are:  ${\tt utf8\_bin}$  and  ${\tt utf8mb4\_bin}$ .

- $\bullet$  When the character set is  $\,$  utf8 , the default collate is  $\,$  utf8\_bin .
- $\bullet$  When the character set is  $\tt utf8mb4$  , the default collate is  $\tt utf8mb4\_bin$  .

#### SYNTAX

SHOW COLLATION;

# EXAMPLE



Parameter	Description
Collation	The name of the collation.
Charset	The name of the character set with which the collation is associated.

Last update: February 19, 2024

- 248/1069 - 2023 Vesoft Inc.

# SHOW CREATE SPACE

The  $\mbox{\scriptsize SHOW}$  CREATE SPACE statement shows the creating statement of the specified graph space.

For details about the graph space information, see CREATE SPACE.

SYNTAX

SHOW CREATE SPACE <space\_name>;

#### EXAMPLE

Space   Create Space   Create Space	nebula> SHOW CREATE SPACE basketballplayer;
	Space   Create Space
++	"basketballplayer"   "CREATE SPACE `basketballplayer` (partition_num = 10, replica_factor = 1, charset = utf8, collate = utf8_bin, vid_type = FIXED_STRING(32))"

#### SHOW CREATE TAG/EDGE

The SHOW CREATE TAG statement shows the basic information of the specified tag. For details about the tag, see CREATE TAG.

The SHOW CREATE EDGE statement shows the basic information of the specified edge type. For details about the edge type, see CREATE EDGE.

#### SYNTAX

```
SHOW CREATE {TAG <tag_name> | EDGE <edge_name>};
```

#### EXAMPLES

	HOW CREATE TAG player;
Tag	Create Tag
"player'   	"   "CREATE TAG`player` (   `name` string NULL,   `age` int64 NULL   ) ttl_duration = 0, ttl_col = """
nebula> SF	HOW CREATE EDGE follow;
Edge	Create Edge
"follow'   	"   "CREATE EDGE `follow` (   `degree` int64 NULL   ) ttl_duration = 0, ttl_col = """
+	+

#### **SHOW HOSTS**

The SHOW HOSTS statement shows the cluster information, including the port, status, leader, partition, and version information. You can also add the service type in the statement to view the information of the specific service.

#### SYNTAX

SHOW HOSTS [GRAPH | STORAGE | META];



For a NebulaGraph cluster installed with the source code, the version of the cluster will not be displayed in the output after executing the command  $SHOW HOSTS (GRAPH \mid STORAGE \mid META)$  with the service name.

#### **EXAMPLES**

"storaged0"   9779   "ONLINE"   8	Host	Port		Leader count			Partition distribution	Version
"storaged2"   9779   "ONLINE"   8								
hebula> SHOW HOSTS GRAPH;  Host   Port   Status   Role   Git Info Sha   Version    "graphd"   9669   "ONLINE"   "GRAPH"   "3ba4lbd"   "3.4.1"    "graphd2"   9669   "ONLINE"   "GRAPH"   "3ba4lbd"   "3.4.1"    "graphd2"   9669   "ONLINE"   "GRAPH"   "3ba4lbd"   "3.4.1"    Hebula> SHOW HOSTS STORAGE;  Host   Port   Status   Role   Git Info Sha   Version    "storaged0"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"    "storaged2"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"    "storaged2"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"		1		1 1				
Host	"storaged2" 	9779 -+	"ONLINE"	+	"basketb	allplayer:3, docs:5" 		
Host	andulas CUOM I	IUCIE CD	ADU -					
"graphd"   9669   "ONLINE"   "GRAPH"   "3ba4lbd"   "3.4.1"   "graphd1"   9669   "ONLINE"   "GRAPH"   "3ba4lbd"   "3.4.1"   "graphd2"   9669   "ONLINE"   "GRAPH"   "3ba4lbd"   "3.4.1"    webula> SHOW HOSTS STORAGE;  Host   Port   Status   Role   Git Info Sha   Version    "storaged0"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"   "storaged2"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"   "storaged2"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"			,		+	+		
"graphd"   9669   "ONLINE"   "GRAPH"   "3ba4lbd"   "3.4.1"   "graphd1"   9669   "ONLINE"   "GRAPH"   "3ba4lbd"   "3.4.1"   "graphd2"   9669   "ONLINE"   "GRAPH"   "3ba4lbd"   "3.4.1"    nebula> SHOW HOSTS STORAGE;  Host   Port   Status   Role   Git Info Sha   Version    "storaged0"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"   "storaged1"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"   "storaged2"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"		,						
"graphd2"   9669   "ONLINE"   "GRAPH"   "3ba4lbd"   "3.4.1"    mebula> SHOW HOSTS STORAGE;  Host   Port   Status   Role   Git Info Sha   Version    "storaged0"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"    "storaged1"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"    "storaged2"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"								
hebula> SHOW HOSTS STORAGE;  Host   Port   Status   Role   Git Info Sha   Version    "storaged0"   9779   "ONLINE"   "STORAGE"   "3ba41bd"   "3.4.1"    "storaged1"   9779   "ONLINE"   "STORAGE"   "3ba41bd"   "3.4.1"    "storaged2"   9779   "ONLINE"   "STORAGE"   "3ba41bd"   "3.4.1"				GRAPH"   "3ba4				
Host   Port   Status   Role   Git Info Sha   Version			"ONLTHE"   1	CDVDH.   "3P3V.	1hd"   "	2 / 1"		
"storaged1"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"   "storaged2"   9779   "ONLINE"   "STORAGE"   "3ba4lbd"   "3.4.1"	++-	·				+		
"storaged2"   9779   "ONLINE"   "STORAGE"   "3ba41bd"   "3.4.1"	nebula> SHOW H	HOSTS ST -+   Port	ORAGE; +	Role   G	it Info Sha	-++   Version		
	nebula> SHOW F	HOSTS ST -+   Port -+   9779	ORAGE; +	Role   G	 it Info Sha 3ba41bd"	++   Version   ++   "3.4.1"		
	nebula> SHOW H +	HOSTS ST -+   Port -+   9779   9779	ORAGE; +	Role   G	it Info Sha 3ba41bd" 3ba41bd"	+   Version   +   "3.4.1"     "3.4.1"		
	hebula> SHOW He	HOSTS ST +   Port +   9779   9779   9779	ORAGE; +	Role   G "STORAGE"   ": "STORAGE"   ": "STORAGE"   ":	it Info Sha  3ba41bd" 3ba41bd" 3ba41bd"	+   Version   		
ebula> SHOW HOSTS META;	hebula> SHOW Hebul	HOSTS ST +	ORAGE; +	RoLe   G "STORAGE"   ": "STORAGE"   ": "STORAGE"   ":	it Info Sha 3ba41bd" 3ba41bd" 3ba41bd"	+   Version   		
Host   Port   Status   Role   Git Info Sha   Version	hebula> SHOW Helphonebula> SHOW	HOSTS ST. +	ORAGE;  +	Role   G "STORAGE"   ". "STORAGE"   ". "STORAGE"   ".	it Info Sha 3ba41bd" 3ba41bd" 3ba41bd"	++   Version   ++   "3.4.1"     "3.4.1"     "3.4.1"   ++		
	mebula> SHOW H	HOSTS ST. +	ORAGE;  +	Role   G "STORAGE"   " "STORAGE"   " "STORAGE"   "	it Info Sha 3ba41bd" 3ba41bd" 3ba41bd" o Sha   Ver	++   Version   ++   "3.4.1"     "3.4.1"     "3.4.1"   ++		
Host   Port   Status   Role   Git Info Sha   Version	Host   F	HOSTS ST +	ORAGE;  +	Role   G "STORAGE"   "	it Info Sha 3ba4lbd" 3ba4lbd" 3ba4lbd" o Sha   Ver	++   Version   ++   "3.4.1"     "3.4.1"     "3.4.1"   ++ sion   + 4.1"		

#### **SHOW INDEX STATUS**

The SHOW INDEX STATUS statement shows the status of jobs that rebuild native indexes, which helps check whether a native index is successfully rebuilt or not.

#### SYNTAX

```
SHOW {TAG | EDGE} INDEX STATUS;
```

#### **EXAMPLES**



#### RELATED TOPICS

- Job manager and the JOB statements
- REBUILD NATIVE INDEX

### **SHOW INDEXES**

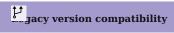
The  $\mbox{\scriptsize SHOW}$  INDEXES statement shows the names of existing native indexes.

SYNTAX

SHOW {TAG | EDGE} INDEXES;

EXAMPLES





In NebulaGraph 2.x, SHOW TAG/EDGE INDEXES only returns  $\mbox{\tt Names}$  .

# SHOW PARTS

The SHOW PARTS statement shows the information of a specified partition or all partitions in a graph space.

### SYNTAX

SHOW PARTS [<part\_id>];

# EXAMPLES

Partition ID	Leader	Peers	Losts
+	+		
1	"192.168.2.1:9779"	"192.168.2.1:9779"	""
2	"192.168.2.2:9779"	"192.168.2.2:9779"	""
3	"192.168.2.3:9779"	"192.168.2.3:9779"	""
4	"192.168.2.1:9779"	"192.168.2.1:9779"	""
5	"192.168.2.2:9779"	"192.168.2.2:9779"	""
6	"192.168.2.3:9779"	"192.168.2.3:9779"	""
7	"192.168.2.1:9779"	"192.168.2.1:9779"	""
8	"192.168.2.2:9779"	"192.168.2.2:9779"	""
9	"192.168.2.3:9779"	"192.168.2.3:9779"	""
10	"192.168.2.1:9779"	"192.168.2.1:9779"	""
+	+	'	+
nebula> SHOW PA	RTS 1.		
	+	+	+
Partition ID	Leader	Peers	Losts
+	+	· +	+
1	"192.168.2.1:9779"	"192.168.2.1:9779"	""
1	+		

The descriptions are as follows.

Parai	neter	Description
Partit	ion ID	The ID of the partition.
Leader		The IP address and the port of the leader.
Peers		The IP addresses and the ports of all the replicas.
Losts		The IP addresses and the ports of replicas at fault.

#### **SHOW ROLES**

The  $\mbox{SHOW ROLES}$  statement shows the roles that are assigned to a user account.

The return message differs according to the role of the user who is running this statement:

- If the user is a GOD or ADMIN and is granted access to the specified graph space, NebulaGraph shows all roles in this graph space except for GOD.
- If the user is a DBA, USER, or GUEST and is granted access to the specified graph space, NebulaGraph shows the user's own role in this graph space.
- If the user does not have access to the specified graph space, NebulaGraph returns PermissionError.

For more information about roles, see Roles and privileges.

#### SYNTAX

```
SHOW ROLES IN <space_name>;
```

### EXAMPLE

```
mebula> SHOW ROLES in basketballplayer;
+-----+
| Account | Role Type |
+-----+
| "user1" | "ADMIN" |
+-----+
```

# SHOW SNAPSHOTS

The  $\mbox{\scriptsize SHOW SNAPSHOTS}$  statement shows the information of all the snapshots.

For how to create a snapshot and backup data, see Snapshot.

ROLE REQUIREMENT

Only the root user who has the  $\mbox{GOD}$  role can use the  $\mbox{SHOW}$  SNAPSHOTS statement.

SYNTAX

```
SHOW SNAPSHOTS;
```

### EXAMPLE

nebula> SHOW SNAPSHOTS;	+	+
Name	Status	Hosts
· ·	"VALID"	"storaged0:9779, storaged1:9779, storaged2:9779"     "storaged0:9779, storaged1:9779, storaged2:9779"   

# SHOW SPACES

The  $\mbox{\scriptsize SHOW}$  SPACES statement shows existing graph spaces in NebulaGraph.

For how to create a graph space, see CREATE SPACE.

SYNTAX

	SHOW SPACES;		
E	EXAMPLE		

nebula> SHOW SPACES;			
++			
Name			
++			
"docs"			
"basketballplayer"			
++			

#### **SHOW STATS**

The SHOW STATS statement shows the statistics of the graph space collected by the latest SUBMIT JOB STATS job.

The statistics include the following information:

- The number of vertices in the graph space
- The number of edges in the graph space
- The number of vertices of each tag
- The number of edges of each edge type



The data returned by SHOW STATS is not real-time. The returned data is collected by the latest SUBMIT JOB STATS job and may include TTL-expired data. The expired data will be deleted and not included in the statistics the next time the Compaction operation is performed.

### PREREQUISITES

You have to run the SUBMIT JOB STATS statement in the graph space where you want to collect statistics. For more information, see SUBMIT JOB STATS.



The result of the SHOW STATS statement is based on the last executed SUBMIT JOB STATS statement. If you want to update the result, run SUBMIT JOB STATS again. Otherwise the statistics will be wrong.

### SYNTAX

SHOW STATS;

# EXAMPLES

```
# Choose a graph space
nebula> USE basketballplayer;
# Start SUBMIT JOB STATS.
nebula> SUBMIT JOB STATS;
| New Job Id |
# Make sure the job executes successfully.
nebula> SHOW JOB 98;
| Job Id(TaskId) | Command(Dest) | Status | Start Time
                                                                                   | Stop Time
                                                                                                                   | Error Code
 98
                    "STATS"
                                      "FINISHED" | 2021-11-01T09:33:21.000000 | 2021-11-01T09:33:21.000000 |
 0
                     "storaged2"
                                      "FINISHED"
"FINISHED"
                                                    2021-11-01T09:33:21.000000 |
2021-11-01T09:33:21.000000 |
                                                                                    2021-11-01T09:33:21.000000
                                                                                                                     "SUCCEEDED"
                                                                                     2021-11-01T09:33:21.000000
                    "storaged0"
                                                                                                                     "SUCCEEDED'
                   | "storaged1" | "FINISHED" | 2021-11-01T09:33
| "Succeeded:3" | "Failed:0" | "In Progress:0"
                                                    2021-11-01T09:33:21.000000
                                                                                     2021-11-01T09:33:21.000000
                                                                                                                     "SUCCEEDED"
 "Total:3"
# Show the statistics of the graph space.
nebula> SHOW STATS;
Type
           Name
                         Count
  "Tag"
           | "player"
  "Tag"
  "Edge"
             "follow"
  "Edge'
              "serve"
                           152
              "vertices"
   "Space
  "Space"
             "edges"
```

# SHOW TAGS/EDGES

The  $\mbox{\scriptsize SHOW}$  TAGS statement shows all the tags in the current graph space.

The  $\mbox{\sc SHOW}$  EDGES statement shows all the edge types in the current graph space.

SYNTAX

SHOW {TAGS | EDGES};

### EXAMPLES

nebula> SHOW TAGS; ++   Name			
"team"   ++ nebula> SHOW EDGES; ++			
Name			
+			

# **SHOW USERS**

The SHOW USERS statement shows the user information.

ROLE REQUIREMENT

Only the root user who has the  $\ensuremath{\mathsf{GOD}}$  role can use the  $\ensuremath{\mathsf{SHOW}}$  USERS statement.

SYNTAX

```
SHOW USERS;
```

# EXAMPLE

#### **SHOW SESSIONS**

When a user logs in to the database, a corresponding session will be created and users can query for session information.

The SHOW SESSIONS statement shows the information of all the sessions. It can also show a specified session with its ID.

#### PRECAUTIONS

- The client will call the API release to release the session and clear the session information when you run exit after the operation ends. If you exit the database in an unexpected way and the session timeout duration is not set via session\_idle\_timeout\_secs in nebula-graphd.conf, the session will not be released automatically. For those sessions that are not automatically released, you need to delete them manually. For details, see KILL SESSIONS.
- ullet SHOW SESSIONS queries the session information of all the Graph services.
- SHOW LOCAL SESSIONS queries the session information of the currently connected Graph service and does not query the session information of other Graph services.
- SHOW SESSION <Session\_Id> queries the session information with a specific session id.

#### SYNTAX

```
SHOW [LOCAL] SESSIONS;
SHOW SESSION <Session_Id>;
```

#### **EXAMPLES**

SessionId		SpaceName	CreateTime	UpdateTime	GraphAddr	Timezone	
1651220858102296	root"	"basketballplayer"	2022-04-29T08:27:38.102296			0	"127.0.0.1"
1651199330300991	root"	"basketballplayer"	2022-04-29T02:28:50.300991	2022-04-29T08:16:28.339038	"127.0.0.1:9669"	0	"127.0.0.1"
1651112899847744	"root"	"basketballplayer"	2022-04-28T02:28:19.847744	2022-04-28T08:17:44.470210	"127.0.0.1:9669"	0	"127.0.0.1"
1651041092662100	root"	"basketballplayer"	2022-04-27T06:31:32.662100	2022-04-27T07:01:25.200978	"127.0.0.1:9669"	0	"127.0.0.1"
1650959429593975	"root"	"basketballplayer"	2022-04-26T07:50:29.593975	2022-04-26T07:51:47.184810	"127.0.0.1:9669"	0	"127.0.0.1"
1650958897679595	root"	""	2022-04-26T07:41:37.679595	2022-04-26T07:41:37.683802	"127.0.0.1:9669"	0	"127.0.0.1"
nebula> SHOW SESSIO		,	+	·	tt	·	÷
SessionId +	UserName   ++	SpaceName	CreateTime	UpdateTime	GraphAddr	Timezone	
1651220858102296	root"	"basketballplayer"	2022-04-29T08:27:38.102296	2022-04-29T08:50:54.254384	"127.0.0.1:9669"	0	"127.0.0.1"

Parameter	Description
SessionId	The session ID, namely the identifier of a session.
UserName	The username in a session.
SpaceName	The name of the graph space that the user uses currently. It is null ( $""$ ) when you first log in because there is no specified graph space.
CreateTime	The time when the session is created, namely the time when the user logs in. The time zone is specified by timezone_name in the configuration file.
UpdateTime	The system will update the time when there is an operation. The time zone is specified by <code>timezone_name</code> in the configuration file.
GraphAddr	The IP address and port of the Graph server that hosts the session.
Timezone	A reserved parameter that has no specified meaning for now.
ClientIp	The IP address of the client.

### **SHOW QUERIES**

The SHOW QUERIES statement shows the information of working queries in the current session.



To terminate queries, see Kill Query.

### PRECAUTIONS

- The SHOW LOCAL QUERIES statement gets the status of queries in the current session from the local cache with almost no latency.
- The SHOW QUERIES statement gets the information of queries in all the sessions from the Meta Service. The information will be synchronized to the Meta Service according to the interval defined by session\_reclaim\_interval\_secs. Therefore the information that you get from the client may belong to the last synchronization interval.

#### SYNTAX

SHOW [LOCAL] QUERIES;

### EXAMPLES

SessionID	ExecutionPlanID			StartTime			
1625463842921750	46	"root"	""192.168.x.x":9669"	2021-07-05T05:44:19.502903	0	"RUNNING"	"SHOW LOCAL QUERIES;"
nebula> SHOW QUERIE	*	+	+	+	.+	+	+
SessionID			Host	StartTime			Query
1625456037718757				2021-07-05T05:51:08.691318			"MATCH p=(v:player)-[*14]-(v2) RETURN v2 AS Friends;"
	tomont voturns the	top 10 qu	eries that have the lon	gest duration.			
nebula> SHOW QUERIE +	S   ORDER BY \$Du	+	+				+
nebula> SHOW QUERIE +	S   ORDER BY \$Du +   ExecutionPlanID	+   User	+   Host	+    StartTime +	DurationInUSec	Status	Query

The descriptions are as follows.

Parameter	Description
SessionID	The session ID.
ExecutionPlanID	The ID of the execution plan.
User	The username that executes the query.
Host	The IP address and port of the Graph server that hosts the session.
StartTime	The time when the query starts.
DurationInUSec	The duration of the query. The unit is microsecond.
Status	The current status of the query.
Query	The query statement.

Last update: February 19, 2024

- 263/1069 - 2023 Vesoft Inc.

# SHOW META LEADER

The SHOW META LEADER statement shows the information of the leader in the current Meta cluster.

For more information about the Meta service, see Meta service.

SYNTAX

SHOW META LEADER;

### EXAMPLE

nebula> SHOW META LEADER;	
+	+
Meta Leader   secs from	last heart beat
+	÷
"127.0.0.1:9559"   3	
+	+

Parameter	Description
Meta Leader	Shows the information of the leader in the Meta cluster, including the IP address and port of the server where the leader is located.
secs from last heart beat	Indicates the time interval since the last heartbeat. This parameter is measured in seconds.

# 4.7 Clauses and options

### 4.7.1 GROUP BY

The GROUP BY clause can be used to aggregate data.

### **OpenCypher Compatibility**

This topic applies to native nGQL only.

You can also use the count() function to aggregate data.

### **Syntax**

The GROUP BY clause groups the rows with the same value. Then operations such as counting, sorting, and calculation can be applied.

The GROUP BY clause works after the pipe symbol (|) and before a YIELD clause.

```
| GROUP BY <var> YIELD <var>, <aggregation_function(var)>
```

The aggregation\_function() function supports avg(), sum(), max(), min(), count(), collect(), and std().

### **Examples**

The following statement finds all the vertices connected directly to vertex "player100", groups the result set by player names, and counts how many times the name shows up in the result set.

The following statement finds all the vertices connected directly to vertex "player100", groups the result set by source vertices, and returns the sum of degree values.

```
nebula> GO FROM "player100" OVER follow \
YIELD src(edge) AS player, properties(edge).degree AS degree \
| GROUP BY $-.player \
YIELD sum($-.degree);
+------+
| sum($-.degree) |
+------+
```

For more information about the sum() function, see Built-in math functions.

#### Implicit GROUP BY

The usage of GROUP BY in the above nGQL statements that explicitly write GROUP BY and act as grouping fields is called explicit GROUP BY, while in openCypher, the GROUP BY is implicit, i.e., GROUP BY groups fields without explicitly writing GROUP BY. The explicit GROUP BY in nGQL is the same as the implicit GROUP BY in openCypher, and nGQL also supports the implicit GROUP BY. For the implicit usage of GROUP BY, see how-to-make-group-by-in-a-cypher-query.

For example, to look up the players over 34 years old with the same length of service, you can use the following statement:

### 4.7.2 LIMIT AND SKIP

The LIMIT clause constrains the number of rows in the output. The usage of LIMIT in native nGQL statements and openCypher compatible statements is different.

- Native nGQL: Generally, a pipe | needs to be used before the LIMIT clause. The offset parameter can be set or omitted directly after the LIMIT statement.
- OpenCypher compatible statements: No pipes are permitted before the LIMIT clause. And you can use SKIP to indicate an offset.



When using LIMIT in either syntax above, it is important to use an ORDER BY clause that constrains the output into a unique order. Otherwise, you will get an unpredictable subset of the output.

#### LIMIT in native nGQL statements

In native nGQL, LIMIT has general syntax and exclusive syntax in 60 statements.

GENERAL LIMIT SYNTAX IN NATIVE NGQL STATEMENTS

In native nGQL, the general LIMIT syntax works the same as in SQL. The LIMIT clause accepts one or two parameters. The values of both parameters must be non-negative integers and be used after a pipe. The syntax and description are as follows:

```
Parameter Description

offset The offset value. It defines the row from which to start returning. The offset starts from 0. The default value is 0, which returns from the first row.

number_rows It constrains the total number of returned rows.
```

### For example:

```
# The following example returns the top 3 rows of data from the result.
nebula> LOOKUP ON player YIELD id(vertex)|\
        LIMIT 3:
id(VERTEX)
 "player100"
   "player101"
 "player102"
# The following example returns the 3 rows of data starting from the second row of the sorted output.
nebula~ GO FROM "player100" OVER follow REVERSELY \
YIELD properties($$).name AS Friend, properties($$).age AS Age \
          ORDER BY $-.Age, $-.Friend \
         | LIMIT 1, 3;
Friend
                      Age
  "Danny Green"
                      31
  "Marco Belinelli"
                       32
```

### LIMIT IN GO STATEMENTS

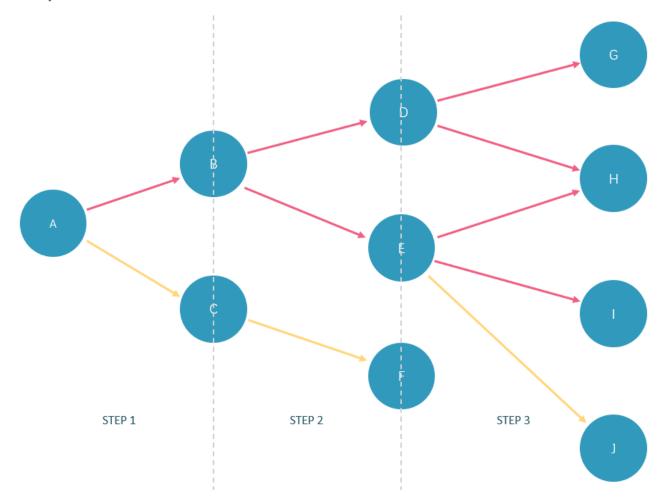
In addition to the general syntax in the native nGQL, the LIMIT in the 60 statement also supports limiting the number of output results based on edges.

Syntax:

```
<go_statement> LIMIT <limit_list>;
```

limit\_list is a list. Elements in the list must be natural numbers, and the number of elements must be the same as the maximum number of STEPS in the 60 statement. The following takes 60 1 TO 3 STEPS FROM "A" OVER \* LIMIT < limit\_list> as an example to introduce this usage of LIMIT in detail.

- The list limit\_list must contain 3 natural numbers, such as 60 1 TO 3 STEPS FROM "A" OVER \* LIMIT [1,2,4].
- 1 in LIMIT [1,2,4] means that the system automatically selects 1 edge to continue traversal in the first step. 2 means to select 2 edges to continue traversal in the second step. 4 indicates that 4 edges are selected to continue traversal in the third step.
- Because 60 1 TO 3 STEPS means to return all the traversal results from the first to third steps, all the red edges and their source and destination vertices in the figure below will be matched by this 60 statement. And the yellow edges represent there is no path selected when the GO statement traverses. If it is not 60 1 TO 3 STEPS but 60 3 STEPS, it will only match the red edges of the third step and the vertices at both ends.



In the basketballplayer dataset, the example is as follows:

```
| "player100" |
| "player100" |
+-------+
```

#### LIMIT in openCypher compatible statements

In openCypher compatible statements such as MATCH, there is no need to use a pipe when LIMIT is used. The syntax and description are as follows:

```
Parameter Description

offset The offset value. It defines the row from which to start returning. The offset starts from 0. The default value is 0, which returns from the first row.

number_rows It constrains the total number of returned rows.
```

Both offset and number\_rows accept expressions, but the result of the expression must be a non-negative integer.



Fraction expressions composed of two integers are automatically floored to integers. For example, 8/6 is floored to 1.

EXAMPLES OF LIMIT

LIMIT can be used alone to return a specified number of results.

EXAMPLES OF SKIP

SKIP can be used alone to set the offset and return the data after the specified position.

EXAMPLE OF SKIP AND LIMIT

SKIP and LIMIT can be used together to return the specified amount of data starting from the specified position.

| "Manu Ginobili" | 41 | +-----

### **4.7.3 SAMPLE**

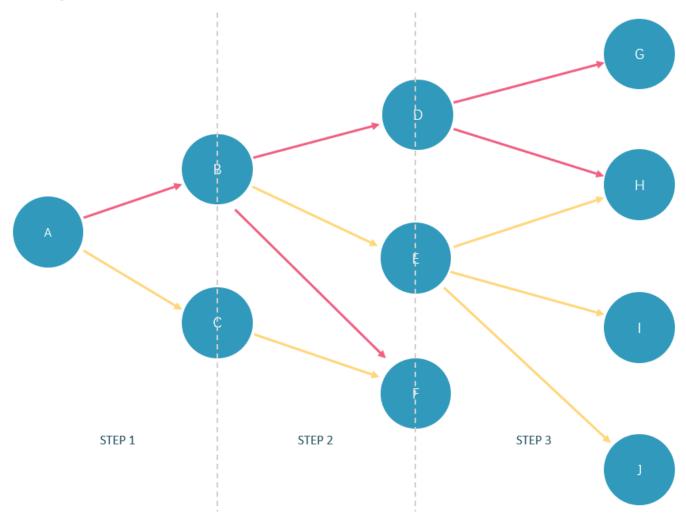
The SAMPLE clause takes samples evenly in the result set and returns the specified amount of data.

SAMPLE can be used in 60 statements only. The syntax is as follows:

```
<go_statement> SAMPLE <sample_list>;
```

sample\_list is a list. Elements in the list must be natural numbers, and the number of elements must be the same as the maximum number of STEPS in the GO statement. The following takes GO 1 TO 3 STEPS FROM "A" OVER \* SAMPLE <sample\_list> as an example to introduce this usage of SAMPLE in detail.

- The list sample\_list must contain 3 natural numbers, such as 60 1 TO 3 STEPS FROM "A" OVER \* SAMPLE [1,2,4].
- 1 in SAMPLE [1,2,4] means that the system automatically selects 1 edge to continue traversal in the first step. 2 means to select 2 edges to continue traversal in the second step. 4 indicates that 4 edges are selected to continue traversal in the third step. If there is no matched edge in a certain step or the number of matched edges is less than the specified number, the actual number will be returned.
- Because 60 1 TO 3 STEPS means to return all the traversal results from the first to third steps, all the red edges and their source and destination vertices in the figure below will be matched by this 60 statement. And the yellow edges represent there is no path selected when the GO statement traverses. If it is not 60 1 TO 3 STEPS but 60 3 STEPS, it will only match the red edges of the third step and the vertices at both ends.



In the basketballplayer dataset, the example is as follows:

```
nebula> GO 3 STEPS FROM "player100" \
OVER * \
```

### 4.7.4 ORDER BY

The ORDER BY clause specifies the order of the rows in the output.

- Native nGQL: You must use a pipe ( | ) and an ORDER BY clause after YIELD clause.
- ullet OpenCypher style: No pipes are permitted. The ORDER BY clause follows a RETURN clause.

There are two order options:

- ASC: Ascending. ASC is the default order.
- DESC: Descending.

### Native nGQL Syntax

```
<YIELD clause>
| ORDER BY <expression> [ASC | DESC] [, <expression> [ASC | DESC] ...];
```



In the native nGQL syntax, \$-. must be used after ORDER BY. But it is not required in releases prior to 2.5.0.

#### EXAMPLES

# OpenCypher Syntax

```
<RETURN clause>
ORDER BY <expression> [ASC | DESC] [, <expression> [ASC | DESC] ...];
```

### EXAMPLES

### Order of NULL values

nGQL lists NULL values at the end of the output for ascending sorting, and at the start for descending sorting.

### **4.7.5 RETURN**

The RETURN clause defines the output of an nGQL query. To return multiple fields, separate them with commas.

RETURN can lead a clause or a statement:

- A RETURN clause can work in openCypher statements in nGQL, such as MATCH or UNWIND.
- A RETURN statement can work independently to output the result of an expression.

#### OpenCypher compatibility

This topic applies to the openCypher syntax in nGQL only. For native nGQL, use VIELD.

RETURN does not support the following openCypher features yet.

• Return variables with uncommon characters, for example:

```
MATCH (`non-english_characters`:player) \
RETURN `non-english_characters`;
```

• Set a pattern in the RETURN clause and return all elements that this pattern matches, for example:

```
MATCH (v:player) \
RETURN (v)-[e]->(v2);
```

#### Map order description

When RETURN returns the map data structure, the order of key-value pairs is undefined.

### Return vertices or edges

Use the RETURN {<vertex\_name> | <edge\_name>} to return vertices and edges all information.

#### Return VIDs

Use the id() function to retrieve VIDs.

#### **Return Tag**

Use the labels() function to return the list of tags on a vertex.

To retrieve the nth element in the labels(v) list, use labels(v)[n-1]. The following example shows how to use labels(v)[0] to return the first tag in the list.

### Return properties

Use the syntax <vertex\_name>.<tag\_name>..property\_name> to return the property of a vertex; use the syntax <edge\_name>..property\_name> to return the property of an edge.

### Return edge type

Use the type() function to return the matched edge types.

#### Return paths

Use RETURN <path\_name> to return all the information of the matched paths.

RETURN VERTICES IN A PATH

Use the nodes() function to return all vertices in a path.

RETURN EDGES IN A PATH

Use the relationships() function to return all edges in a path.

RETURN PATH LENGTH

Use the length() function to return the length of a path.

```
| <("player100" :player{age: 42, name: "Tim Duncan"})-[:serve@0 {end_year: 2016, start_year: 1997}]->("team204" :team{name:
    <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player101" :player{age: 36, name: "Tony
Parker"})>
| <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player125" :player{age: 41, name: "Manu
| <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player101" :player{age: 36, name: "Tony Parker"})-[:serve@0 {end_year: 2018, start_year: 1999}]-
  >("team204" :team{name: "Spurs"})>
| <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player101" :player{age: 36, name: "Tony Parker"})-[:serve@0 {end_year: 2019, start_year: 2018}]->("team215" :team{name: "Hornets"})> | 2 | <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player101" :player{age: 36, name: "Tony Parker"})-[:follow@0 {degree: 95}]->("player100" :player{age: 42, name: "Tony Parker"})-[:follow@0 {degree: 95}]->("player100" :player4ge: 36, name: "Tony Parker"})-[:follow@0 {degree: 95}]->("player4ge: 95, name: "Tony Parker"]-[:follow@0 {degr
                   "Tim Duncan"})>
("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player101" :player{age: 36, name: "Tony Parker"})-[:follow@0 {degree: 90}]->("player102" :player{age: 33,
name: "LaMarcus Aldridge"})> | 2
| <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player101" :player{age: 36, name: "Tony Parker"})-[:follow@0 {degree: 95}]->("player125" :player{age: 41,
                     "Manu Ginobili"})>
| <("player100" :player{age: 42, name: "Tim Duncan"}}-[:follow@0 {degree: 95}]->("player125" :player{age: 41, name: "Manu Ginobili"})-[:serve@0 {end_year: 2018, start_year: 2002}]-
>("team204" :team{name: "Spurs"})> | 2
    <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player125" :player{age: 41, name: "Manu Ginobili"})-[:follow@0 {degree: 90}]->("player100" :player125" :player{age: 42, name: "Manu Ginobili"})-[:follow@0 {degree: 90}]->("player100" :player125" :player
name: "Tim Duncan"})>
                                                                                | 2
```

#### Return all elements

To return all the elements that this pattern matches, use an asterisk (\*).

#### Rename a field

Use the AS <alias> syntax to rename a field in the output.

### Return a non-existing property

If a property matched does not exist, NULL is returned.

### Return expression results

To return the results of expressions such as literals, functions, or predicates, set them in a RETURN clause.

```
nebula> MATCH (v:player{name:"Tony Parker"})-->(v2:player) \
    RETURN DISTINCT v2.player.name, "Hello"+" graphs!", v2.player.age > 35;
| v2.player.name
                           | ("Hello"+" graphs!") | (v2.player.age>35)
| "LaMarcus Aldridge" | "Hello graphs!"
                                                        false
                          | "Hello graphs!"
| "Hello graphs!"
 "Tim Duncan"
"Manu Ginobili"
                                                         true
nebula> RETURN 1+1;
| (1+1) |
| 2
nebula> RETURN 3 > 1;
| (3>1) |
| true
nebula> RETURN 1+1, rand32(1, 5);
| (1+1) | rand32(1,5) |
        | 1
```

### Return unique fields

Use DISTINCT to remove duplicate fields in the result set.

```
# Before using DISTINCT.
nebula> MATCH (v:player{name:"Tony Parker"})--(v2:player) \
    RETURN v2.player.name, v2.player.age;
v2.player.name
                            v2.player.age
  "Manu Ginobili"
                             41
                             36
  "Boris Diaw"
  "Marco Belinelli"
  "Dejounte Murray"
"Tim Duncan"
                             29
  "Tim Duncan" | 42
"LaMarcus Aldridge" | 33
"LaMarcus Aldridge" | 33
# After using DISTINCT.
nebula> MATCH (v:player{name:"Tony Parker"})--(v2:player) \
RETURN DISTINCT v2.player.name, v2.player.age;
| v2.player.name
                             | v2.player.age
  "Manu Ginobili"
                             41
  "Boris Diaw"
                              36
   "Marco Belinelli"
  "Dejounte Murray"
"Tim Duncan"
                             29
  "LaMarcus Aldridge" | 33
```

### 4.7.6 TTL

TTL (Time To Live) specifies a timeout for a property. Once timed out, the property expires.

### **OpenCypher Compatibility**

This topic applies to native nGQL only.

#### Precautions

- You CANNOT modify a property schema with TTL options on it.
- TTL options and indexes have coexistence issues.
- TTL options and indexes CANNOT coexist on a tag or an edge type. If there is an index on a property, you cannot set TTL options on other properties.
- If there are TTL options on a tag, an edge type, or a property, you can still add an index on them.

#### TTL options

The native nGQL TTL feature has the following options.

Option	Description
ttl_col	Specifies the property to set a timeout on. The data type of the property must be $\mbox{int}$ or $\mbox{timestamp}$ .
ttl_duration	Specifies the timeout adds-on value in seconds. The value must be a non-negative int64 number. A property expires if the sum of its value and the ttl_duration value is smaller than the current timestamp. If the ttl_duration value is 0, the property never expires.

### Data expiration and deletion



- When the TTL options are set for a property of a tag or an edge type and the property's value is NULL, the property never expires.
- If a property with a default value of <code>now()</code> is added to a tag or an edge type and the TTL options are set for the property, the history data related to the tag or the edge type will never expire because the value of that property for the history data is the current timestamp.

#### VERTEX PROPERTY EXPIRATION

Vertex property expiration has the following impact.

- If a vertex has only one tag, once a property of the vertex expires, the vertex expires.
- If a vertex has multiple tags, once a property of the vertex expires, properties bound to the same tag with the expired property also expire, but the vertex does not expire and other properties of it remain untouched.

### EDGE PROPERTY EXPIRATION

Since an edge can have only one edge type, once an edge property expires, the edge expires.

#### DATA DELETION

The expired data are still stored on the disk, but queries will filter them out.

NebulaGraph automatically deletes the expired data and reclaims the disk space during the next compaction.



If TTL is disabled, the corresponding data deleted after the last compaction can be queried again.

#### Use TTL options

You must use the TTL options together to set a valid timeout on a property.

SET A TIMEOUT IF A TAG OR AN EDGE TYPE EXISTS

If a tag or an edge type is already created, to set a timeout on a property bound to the tag or edge type, use ALTER to update the tag or edge type.

```
# Create a tag.
nebula> CREATE TAG IF NOT EXISTS t1 (a timestamp);

# Use ALTER to update the tag and set the TTL options.
nebula> ALTER TAG t1 TTL_COL = "a", TTL_DURATION = 5;

# Insert a vertex with tag t1. The vertex expires 5 seconds after the insertion.
nebula> INSERT VERTEX t1(a) VALUES "101":(now());
```

SET A TIMEOUT WHEN CREATING A TAG OR AN EDGE TYPE

Use TTL options in the CREATE statement to set a timeout when creating a tag or an edge type. For more information, see CREATE TAG and CREATE EDGE.

```
# Create a tag and set the TTL options.
nebula> CREATE TAG IF NOT EXISTS t2(a int, b int, c string) TTL_DURATION= 100, TTL_COL = "a";

# Insert a vertex with tag t2. The timeout timestamp is 1648197238 (1648197138 + 100).
nebula> INSERT VERTEX t2(a, b, c) VALUES "102":(1648197138, 30, "Hello");
```

### Remove a timeout

To disable TTL and remove the timeout on a property, you can use the following approaches.

• Drop the property with the timeout.

```
nebula> ALTER TAG t1 DROP (a);
```

• Set ttl\_col to an empty string.

```
nebula> ALTER TAG t1 TTL_COL = "";
```

• Set ttl\_duration to 0. This operation keeps the TTL options and prevents the property from expiring and the property schema from being modified.

```
nebula> ALTER TAG t1 TTL_DURATION = 0;
```

# 4.7.7 WHERE

The WHERE clause filters the output by conditions.

The WHERE clause usually works in the following queries:

- Native nGQL: such as 60 and LOOKUP.
- $\bullet$  OpenCypher syntax: such as  $\mbox{\sc MATCH}$  and  $\mbox{\sc WITH}$  .

### OpenCypher compatibility

Filtering on edge rank is a native nGQL feature. To retrieve the rank value in openCypher statements, use the rank() function, such as MATCH (:player)-[e:follow]->() RETURN rank(e); .

### Basic usage



In the following examples, \$\$ and  $\$^A$  are reference operators. For more information, see Operators.

**DEFINE CONDITIONS WITH BOOLEAN OPERATORS** 

Use the boolean operators NOT, AND, OR, and XOR to define conditions in WHERE clauses. For the precedence of the operators, see  $\frac{1}{2}$ 

#### FILTER ON PROPERTIES

Use vertex or edge properties to define conditions in where clauses.

• Filter on a vertex property:

• Filter on an edge property:

```
nebula> MATCH (v:player)-[e]->() \
    WHERE e.start_year < 2000 \</pre>
        {\tt RETURN\ DISTINCT\ v.player.name,\ v.player.age;}
| v.player.name
                       v.player.age
  "Tony Parker"
                        36
  "Tim Duncan"
                         42
| "Grant Hill"
                        46
nebula> GO FROM "player100" OVER follow \
        WHERE follow.degree > 90 \
        YIELD dst(edge);
dst(EDGE)
  "player101"
| "player125"
```

#### FILTER ON DYNAMICALLY-CALCULATED PROPERTIES

### FILTER ON EXISTING PROPERTIES

#### FILTER ON EDGE RANK

In nGQL, if a group of edges has the same source vertex, destination vertex, and properties, the only thing that distinguishes them is the rank. Use rank conditions in WHERE clauses to filter such edges.

```
# The following example creates test data.
nebula> CREATE SPACE IF NOT EXISTS test (vid_type=FIXED_STRING(30));
nebula> USE test;
nebula> CREATE EDGE IF NOT EXISTS e1(p1 int);
nebula> CREATE TAG IF NOT EXISTS person(p1 int);
nebula> INSERT VERTEX person(p1) VALUES "1":(1);
nebula> INSERT VERTEX person(p1) VALUES "2":(2);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@0:(10);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@1:(11);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@2:(12);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@3:(13);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@4:(14);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@5:(15);
nebula> INSERT EDGE e1(p1) VALUES "1"->"2"@6:(16);
# The following example use rank to filter edges and retrieves edges with a rank greater than 2. nebula> GO FROM "1" \
          OVER e1 \
          WHERE rank(edge) > 2 \
           YIELD src(edge), dst(edge), rank(edge) AS Rank, properties(edge).pl | \
          ORDER BY $-.Rank DESC;
| src(EDGE) | dst(EDGE) | Rank | properties(EDGE).pl
   "1"
                   "2"
                                            15
                  "2"
"2"
  "1"
"1"
                                 | 4
                                            14
                                          13
# Filter edges by rank. Find follow edges with rank equal to 0.
nebula> MATCH (v)-[e:follow]->() \
           WHERE rank(e)==0 \
            RETURN *;
l v
                                                                                   | e
  ("player142" :player{age: 29, name: "Klay Thompson"})
                                                                                     [:follow "player142"->"player117" @0 {degree: 90}]
  ("player139" :player{age: 34, name: "Marc Gasol"})
("player108" :player{age: 36, name: "Boris Diaw"})
                                                                                   | [:follow "player139"->"player138" @0 (degree: 99]|
| [:follow "player108"->"player100" @0 (degree: 80)]
| [:follow "player108"->"player101" @0 (degree: 80)]
("player108" :player{age: 36, name: "Boris Diaw"})
```

### FILTER ON PATTERN

```
nebula> MATCH (v:player{name:"Tim Duncan"})-[e]->(t) \
         WHERE (v)-[e]->(t:team) \
         RETURN (v)-->();
| (v)-->() = (v)--
>()
[<("player100" :player{age: 42, name: "Tim Duncan"})-[:serve@0 {end_year: 2016, start_year: 1997}]->("team204" :team{name: "Spurs"})>, <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player101" :player{age: 36, name: "Tony Parker"})>, <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player125" :player{age: 41,
name: "Manu Ginobili"})>] |
RETURN (v)-->();
| (v)-->() = (v)--
>()
 [<("player100" :player{age: 42, name: "Tim Duncan"})-[:serve@0 {end_year: 2016, start_year: 1997}]->("team204" :team{name: "Spurs"})>, <("player100" :player{age: 42, name: "Tim Duncan"})-
[:follow@0 {degree: 95}]->("player101" :player{age: 36, name: "Tony Parker"})>, <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player125" :player{age: 41, name: "Manu Ginobili"})>] |
| [<("player100" :player{age: 42, name: "Tim Duncan"})-[:serve@0 {end_year: 2016, start_year: 1997}]->("team204" :team{name: "Spurs"})>, <("player100" :player{age: 42, name: "Tim Duncan"})-
[:follow@0 {degree: 95}]->("player101" :player{age: 36, name: "Tony Parker"})>, <("player100" :player{age: 42, name: "Tim Duncan"})-[:follow@0 {degree: 95}]->("player125" :player{age: 41,
name: "Manu Ginobili"})>] |
```

# Filter on strings

Use STARTS WITH, ENDS WITH, or CONTAINS in WHERE clauses to match a specific part of a string. String matching is case-sensitive.

### STARTS WITH

STARTS WITH will match the beginning of a string.

The following example uses STARTS WITH "T" to retrieve the information of players whose name starts with T.

If you use STARTS WITH "t" in the preceding statement, an empty set is returned because no name in the dataset starts with the lowercase t.

### ENDS WITH

ENDS WITH will match the ending of a string.

The following example uses ENDS WITH "r" to retrieve the information of players whose name ends with  $\ r$ .

#### CONTAINS

CONTAINS will match a certain part of a string.

The following example uses CONTAINS "Pa" to match the information of players whose name contains Pa.

NEGATIVE STRING MATCHING

You can use the boolean operator NOT to negate a string matching condition.

### Filter on lists

MATCH VALUES IN A LIST

Use the IN operator to check if a value is in a specific list.

```
| v.player.age |
| v.player.name
 "Ben Simmons"
"Giannis Antetokounmpo"
                               24
 "Kyle Anderson"
"Joel Embiid"
"Kristaps Porzingis"
                               25
                               25
                               23
 "Luka Doncic"
                               20
nebula> LOOKUP ON player \
WHERE player.age IN [25,28] \
YIELD properties(vertex).name, properties(vertex).age;
| properties(VERTEX).name | properties(VERTEX).age
 "Kyle Anderson"
                              25
  "Damian Lillard"
                              28
 "Joel Embiid"
"Paul George"
                              25
  "Ricky Rubio"
                               28
```

MATCH VALUES NOT IN A LIST

Use NOT before IN to rule out the values in a list.

# 4.7.8 YIELD

YIELD defines the output of an nGQL query.

YIELD can lead a clause or a statement:

- $\bullet$  A YIELD clause works in nGQL statements such as 60 , FETCH , or LOOKUP and must be defined to return the result.
- A YIELD statement works in a composite query or independently.

### OpenCypher compatibility

This topic applies to native nGQL only. For the openCypher syntax, use RETURN.

YIELD has different functions in openCypher and nGQL.

• In openCypher, YIELD is used in the CALL[...YIELD] clause to specify the output of the procedure call.



NGQL does not support CALL[...YIELD] yet.

• In nGQL, YIELD works like RETURN in openCypher.



In the following examples, \$\$ and \$- are reference operators. For more information, see Operators.

### **YIELD clauses**

SYNTAX

```
YIELD [DISTINCT] <col> [AS <alias>] [, <col> [AS <alias>] ...];
```

Parameter	Description
DISTINCT	Aggregates the output and makes the statement return a distinct result set.
col	A field to be returned. If no alias is set, col will be a column name in the output.
alias	An alias for $\cot$ . It is set after the keyword AS and will be a column name in the output.

USE A YIELD CLAUSE IN A STATEMENT

• Use YIELD with GO:

- 287/1069 - 2023 Vesoft Inc.

```
| "Manu Ginobili" | 41 |
+-----+
```

• Use YIELD with FETCH:

• Use YIELD with LOOKUP:

#### YIELD statements

SYNTAX

```
YIELD [DISTINCT] <col> [AS <alias>] [, <col> [AS <alias>] ...]
[WHERE <conditions>];
```

Parameter	Description
DISTINCT	Aggregates the output and makes the statement return a distinct result set.
col	A field to be returned. If no alias is set, col will be a column name in the output.
alias	An alias for $\cot$ . It is set after the keyword AS and will be a column name in the output.
conditions	Conditions set in a WHERE clause to filter the output. For more information, see WHERE.

USE A YIELD STATEMENT IN A COMPOSITE QUERY

In a composite query, a YIELD statement accepts, filters, and modifies the result set of the preceding statement, and then outputs it.

The following query finds the players that "player100" follows and calculates their average age.

The following query finds the players that "player101" follows with the follow degrees greater than 90.

The following query finds the vertices in the player that are older than 30 and younger than 32, and returns the de-duplicate

USE A STANDALONE YIELD STATEMENT

A YIELD statement can calculate a valid expression and output the result.

```
nebula> YIELD rand32(1, 6);

| rand32(1, 6) |
| rand32(1, 6) |
| result |
| rand32(1, 6) |
| string1 |
| string2 |
| string2 |
| rand32(1, 6) |
| rand32(1, 6)
```

### 4.7.9 WITH

The WITH clause can retrieve the output from a query part, process it, and pass it to the next query part as the input.

### OpenCypher compatibility

This topic applies to openCypher syntax only.



WITH has a similar function with the Pipe symbol in native nGQL, but they work in different ways. DO NOT use pipe symbols in the openCypher syntax or use WITH in native nGQL statements.

#### Combine statements and form a composite query

Use a WITH clause to combine statements and transfer the output of a statement as the input of another statement.

EXAMPLE 1

The following statement:

- 1. Matches a path.
- 2. Outputs all the vertices on the path to a list with the nodes() function.
- 3. Unwinds the list into rows.
- 4. Removes duplicated vertices and returns a set of distinct vertices.

### EXAMPLE 2

The following statement:

- 1. Matches the vertex with the VID player100.
- 2. Outputs all the tags of the vertex into a list with the labels() function.
- 3. Unwinds the list into rows.
- 4. Returns the output.

```
nebula> MATCH (v) \
WHERE id(v)="player100" \
WITH labels(v) AS tags_unf \
UNWIND tags_unf AS tags_f \
RETURN tags_f;
+-----+
| tags_f |
+-----+
| "player" |
+-----+
```

### Filter composite queries

WITH can work as a filter in the middle of a composite query.

### Process the output before using collect()

Use a WITH clause to sort and limit the output before using collect() to transform the output into a list.

#### Use with RETURN

Set an alias using a WITH clause, and then output the result through a RETURN clause.

### 4.7.10 UNWIND

UNWIND transform a list into a sequence of rows.

UNWIND can be used as an individual statement or as a clause within a statement.

#### **UNWIND** statement

SYNTAX

```
UNWIND <list> AS <alias> <RETURN clause>;
```

EXAMPLES

• To transform a list.

```
nebula> UNWIND [1,2,3] AS n RETURN n;
+---+
| n |
+---+
| 1 |
| 2 |
| 3 |
+---+
```

### **UNWIND** clause

SYNTAX

• The UNWIND clause in native nGQL statements.



To use a UNWIND clause in a native nGQL statement, use it after the | operator and use the \\$- prefix for variables. If you use a statement or clause after the UNWIND clause, use the | operator and use the \\$- prefix for variables.

```
<statement> | UNWIND $-.<var> AS <alias> <|> <clause>;
```

• The UNWIND clause in openCypher statements.

```
<statement> UNWIND <List> AS <alias> <RETURN clause>;
```

EXAMPLES

• To transform a list of duplicates into a unique set of rows using WITH DISTINCT in a UNWIND clause.



WITH DISTINCT is not available in native nGQL statements.

```
| [1, 2, 3] |
```

• To use an UNWIND clause in a MATCH statement.

• To use an UNWIND clause in a GO statement.

• To use an UNWIND clause in a LOOKUP statement.

• To use an UNWIND clause in a FETCH statement.

. To use an UNWIND clause in a GET SUBGRAPH statement.

• To use an UNWIND clause in a FIND PATH statement.

### 4.8 Space statements

### 4.8.1 CREATE SPACE

Graph spaces are used to store data in a physically isolated way in NebulaGraph, which is similar to the database concept in MySQL. The CREATE SPACE statement can create a new graph space or clone the schema of an existing graph space.

### Prerequisites

Only the God role can use the CREATE SPACE statement. For more information, see AUTHENTICATION.

#### **Syntax**

CREATE GRAPH SPACES

Parameter	Description			
IF NOT EXISTS	Detects if the related graph space exists. If it does not exist, a new one will be created. The graph space existence detection here only compares the graph space name (excluding properties).			
<pre><graph_space_name></graph_space_name></pre>	<ol> <li>Uniquely identifies a graph space in a NebulaGraph instance.</li> <li>Space names cannot be modified after they are set.</li> <li>Space names cannot start with a number; they support 1-4 byte UTF-8 encoded characters, including English letters (case sensitive), numbers, Chinese characters, etc., but do not include special characters other than underscores. To use special characters, reserved keywords or starting with a number, quote them with backticks (`) and cannot use periods ( . ). For more information, see Keywords and reserved words. Note: If you name a space in Chinese and encounter a SyntaxError , you need to quote the Chinese characters with backticks (`).</li> </ol>			
partition_num	Specifies the number of partitions in each replica. The suggested value is 20 times (2 times for HDD) the number of the hard disks in the cluster. For example, if you have three hard disks in the cluster, we recommend that you set 60 partitions. The default value is 100.			
replica_factor	Specifies the number of replicas in the cluster. The suggested number is 3 in a production environment and 1 in a test environment. The replica number must be an <b>odd number</b> for the need of quorum-based voting. The default value is 1.			
vid_type	A required parameter. Specifies the VID type in a graph space. Available values are FIXED_STRING(N) and INT64. INT equals to INT64.  `FIXED_STRING( <n>) specifies the VID as a string, while INT64 specifies it as an integer. N represents the maximum length of the VIDs. If you set a VID that is longer than N bytes, NebulaGraph throws an error. Note, for UTF-8 chars, the length may vary in different cases, i.e. a UTF-8 Chinese char is 3 byte, this means 11 Chinese chars(length-33) will exeed a FIXED_STRING(32) vid defination.</n>			
COMMENT	The remarks of the graph space. The maximum length is 256 bytes. By default, there is no comments on a space.			

## Caution

- If the replica number is set to one, you will not be able to load balance or scale out the NebulaGraph Storage Service with the BALANCE statement.
- Restrictions on VID type change and VID length:
- For NebulaGraph v1.x, the type of VIDs can only be INT64, and the String type is not allowed. For NebulaGraph v2.x, both INT64 and FIXED\_STRING(<N>) VID types are allowed. You must specify the VID type when creating a graph space, and use the same VID type in INSERT statements, otherwise, an error message Wrong vertex id type: 1001 occurs.
- The length of the VID should not be longer than N characters. If it exceeds N, NebulaGraph throws The VID must be a 64-bit integer or a string fitting space vertex id length limit.
- If the Host not enough! error appears, the immediate cause is that the number of online storage hosts is less than the value of replica\_factor specified when creating a graph space. In this case, you can use the SHOW HOSTS command to see if the following situations occur:
- For the case where there is only one storage host in a cluster, the value of replica\_factor can only be specified to 1. Or create a graph space after storage hosts are scaled out.
- A new storage host is found, but ADD HOSTS is not executed to activate it. In this case, run SHOW HOSTS to locate the new storage host information and then run ADD HOSTS to activate it. A graph space can be created after there are enough storage hosts.
- For offline storage hosts after running SHOW HOSTS, troubleshooting is needed.

# Lacy version compatibility

For NebulaGraph v2.x before v2.5.0, vid\_type is optional and defaults to FIXED\_STRING(8).

#### O Note

graph\_space\_name, partition\_num, replica\_factor, vid\_type, and comment cannot be modified once set. To modify them, drop the current working graph space with DROP\_SPACE and create a new one with CREATE SPACE.

#### **CLONE GRAPH SPACES**

CREATE SPACE [IF NOT EXISTS] <new\_graph\_space\_name> AS <old\_graph\_space\_name>;

Parameter	Description		
IF NOT EXISTS	Detects if the new graph space exists. If it does not exist, the new one will be created. The graph space existence detection here only compares the graph space name (excluding properties).		
<new_graph_space_name></new_graph_space_name>	The name of the graph space that is newly created. The name of the graph space starts with a letter, supports 1 to 4 bytes UTF-8 encoded characters, such as English letters (case-sensitive), digits, and Chinese characters, but does not support special characters except underscores. For more information, see Keywords and reserved words. When a new graph space is created, the schema of the old graph space <code>sold_graph_space_name&gt;</code> will be cloned, including its parameters (the number of partitions and replicas, etc.), Tag, Edge type and native indexes.		
<pre><old_graph_space_name></old_graph_space_name></pre>	The name of the graph space that already exists.		

#### **Examples**

# The following example creates a graph space with a specified VID type and the maximum length. Other fields still use the default values. nebula> CREATE SPACE IF NOT EXISTS my\_space\_1 (vid\_type=FIXED\_STRING(30));

# The following example creates a graph space with a specified partition number, replica number, and VID type. nebula> CREATE SPACE IF NOT EXISTS my\_space\_2 (partition\_num=15, replica\_factor=1, vid\_type=FIXED\_STRING(30));

# The following example creates a graph space with a specified partition number, replica number, and VID type, and adds a comment on it. nebula> CREATE SPACE IF NOT EXISTS my\_space\_3 (partition\_num=15, replica\_factor=1, vid\_type=FIXED\_STRING(30)) comment="Test the graph space";

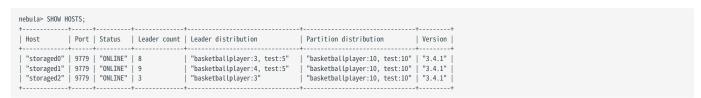
#### Implementation of the operation



Trying to use a newly created graph space may fail because the creation is implemented asynchronously. To make sure the follow-up operations work as expected, Wait for two heartbeat cycles, i.e., 20 seconds. To change the heartbeat interval, modify the heartbeat\_interval\_secs parameter in the configuration files for all services. If the heartbeat interval is too short (i.e., less than 5 seconds), disconnection between peers may happen because of the misjudgment of machines in the distributed system.

### Check partition distribution

On some large clusters, the partition distribution is possibly unbalanced because of the different startup times. You can run the following command to do a check of the machine distribution.



To balance the request loads, use the following command.



### 4.8.2 USE

USE specifies a graph space as the current working graph space for subsequent queries.

### **Prerequisites**

Running the USE statement requires some privileges for the graph space. Otherwise, NebulaGraph throws an error.

#### **Syntax**

```
USE <graph_space_name>;
```

### **Examples**

```
# The following example creates two sample spaces.
nebula> CREATE SPACE IF NOT EXISTS space1 (vid_type=FIXED_STRING(30));
nebula> CREATE SPACE IF NOT EXISTS space2 (vid_type=FIXED_STRING(30));

# The following example specifies space1 as the current working graph space.
nebula> USE space1;

# The following example specifies space2 as the current working graph space. Hereafter, you cannot read any data from space1, because these vertices and edges being traversed have no relevance with space1.
nebula> USE space2;
```

## Caution

You cannot use two graph spaces in one statement.

Different from Fabric Cypher, graph spaces in NebulaGraph are fully isolated from each other. Making a graph space as the working graph space prevents you from accessing other spaces. The only way to traverse in a new graph space is to switch by the USE statement. In Fabric Cypher, you can use two graph spaces in one statement (using the USE + CALL syntax). But in NebulaGraph, you can only use one graph space in one statement.

### 4.8.3 SHOW SPACES

 $\ensuremath{\mathsf{SHOW}}$  Spaces lists all the graph spaces in the NebulaGraph examples.

### Syntax

SHOW SPACES;

### Example



To create graph spaces, see **CREATE SPACE**.

### 4.8.4 DESCRIBE SPACE

 ${\tt DESCRIBE\ SPACE\ returns\ the\ information\ about\ the\ specified\ graph\ space}.$ 

### Syntax

You can use DESC instead of DESCRIBE for short.

```
DESC[RIBE] SPACE <graph_space_name>;
```

The DESCRIBE SPACE statement is different from the SHOW SPACES statement. For details about SHOW SPACES, see SHOW SPACES.

### Example



### 4.8.5 CLEAR SPACE

CLEAR SPACE deletes the vertices and edges in a graph space, but does not delete the graph space itself and the schema information.



It is recommended to execute SUBMIT JOB COMPACT immediately after executing the CLEAR SPACE operation improve the query performance. Note that the COMPACT operation may affect query performance, and it is recommended to perform this operation during low business hours (e.g., early morning).

#### Permission requirements

Only the God role has the permission to run CLEAR SPACE.

#### Caution

- Once cleared, the data CANNOT be recovered. Use CLEAR SPACE with caution.
- · CLEAR SPACE is not an atomic operation. If an error occurs, re-run CLEAR SPACE to avoid data remaining.
- The larger the amount of data in the graph space, the longer it takes to clear it. If the execution fails due to client connection timeout, increase the value of the storage\_client\_timeout\_ms parameter in the Graph Service configuration.
- During the execution of CLEAR SPACE, writing data into the graph space is not automatically prohibited. Such write operations can result in incomplete data clearing, and the residual data can be damaged.



The NebulaGraph Community Edition does not support blocking data writing while allowing CLEAR SPACE.



The NebulaGraph Enterprise Edition supports blocking data writing by setting VARIABLE read\_only=true before running CLEAR SPACE. After the data are cleared successfully, run SET VARIABLE read\_only=false to allow data writing again.

#### **Syntax**

CLEAR SPACE [IF EXISTS] <space\_name>;

Parameter/ Option	Description
IF EXISTS	Check whether the graph space to be cleared exists. If it exists, continue to clear it. If it does not exist, the execution finishes, and a message indicating that the execution succeeded is displayed.  If IF EXISTS is not set and the graph space does not exist, the CLEAR SPACE statement fails to execute, and an error occurs.
space_name	The name of the space to be cleared.

### Example:

CLEAR SPACE basketballplayer;

- 301/1069 - 2023 Vesoft Inc.

#### Data reserved

CLEAR SPACE does not delete the following data in a graph space:

- Tag information.
- Edge type information.
- The metadata of native indexes and full-text indexes.

The following example shows what CLEAR SPACE deletes and reserves.

```
# Enter the graph space basketballplayer.
nebula [(none)]> use basketballplayer;
Execution succeeded
# List tags and Edge types.
nebula[basketballplayer]> SHOW TAGS;
  "player"
Got 2 rows
nebula[basketballplayer]> SHOW EDGES;
Name
  "follow"
| "serve"
Got 2 rows
# Submit a job to make statistics of the graph space.
nebula[basketballplayer]> SUBMIT JOB STATS;
| New Job Id |
| 4
Got 1 rows
# Check the statistics.
nebula[basketballplayer]> SHOW STATS;
| Type | Name
                           Count
            | "player"
                            | 51
  "Tag"
"Edge"
"Edge"
                "team"
                              30
               "follow"
               "serve" | 152
"vertices" | 81
                               152
  "Space"
  "Space" | "edges"
Got 6 rows
# List tag indexes.
nebula[basketballplayer]> SHOW TAG INDEXES;
Index Name
                     | By Tag | Columns
| "player_index_0" | "player" | []
| "player_index_1" | "player" | ["name"]
Got 2 rows
# ----- Dividing line for CLEAR SPACE ----
# Run CLEAR SPACE to clear the graph space basketballplayer.
nebula[basketballplayer]> CLEAR SPACE basketballplayer;
Execution succeeded
# Update the statistics.
nebula[basketballplayer]> SUBMIT JOB STATS;
| New Job Id |
| 5
Got 1 rows
# Check the statistics. The tags and edge types still exist, but all the vertices and edges are gone.nebula[basketballplayer]> SHOW STATS;
Type Name
                            Count
| "Tag"
            | "player" | 0
  "Tag"
                "team"
"Edge"
```

"Edge"   "serve"   0
+
Got 6 rows
# Try to list the tag indexes. They still exist.
nebula[basketballplayer]> SHOW TAG INDEXES;
tt
Index Name   By Tag   Columns
tt
"player_index_0"   "player"   []
"player_index_1"   "player"   ["name"]
tt
Got 2 rows (time spent 523/978 us)

### 4.8.6 DROP SPACE

DROP SPACE deletes the specified graph space and everything in it.



DROP SPACE can only delete the specified logic graph space while retain all the data on the hard disk by modifying the value of auto\_remove\_invalid\_space to false in the Storage service configuration file. For more information, see Storage configuration.



After you execute DROP SPACE, even if the snapshot contains data of the graph space, the data of the graph space cannot be recovered. But if the value of auto\_remove\_invalid\_space is set to false, contact the sales team to recover the data of the graph space.

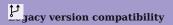
### **Prerequisites**

Only the God role can use the DROP SPACE statement. For more information, see AUTHENTICATION.

#### **Syntax**

DROP SPACE [IF EXISTS] <graph\_space\_name>;

You can use the IF EXISTS keywords when dropping spaces. These keywords automatically detect if the related graph space exists. If it exists, it will be deleted. Otherwise, no graph space will be deleted.



In NebulaGraph versions earlier than 3.1.0, the DROP SPACE statement does not remove all the files and directories from the disk by default.



BE CAUTIOUS about running the DROP SPACE statement.

#### FAQ

Q: Why is my disk space not freed after executing the 'DROP SPACE' statement and deleting a graph space?

A: For NebulaGraph version earlier than 3.1.0, DROP SPACE can only delete the specified logic graph space and does not delete the files and directories on the disk. To delete the files and directories on the disk, manually delete the corresponding file path. The file path is located in <nebula\_graph\_install\_path>/data/storage/nebula/<space\_id> . The <space\_id> can be viewed via DESCRIBE SPACE  $\{space_name\}$ .

Last update: February 19, 2024

- 304/1069 - 2023 Vesoft Inc.

### 4.9 Tag statements

### 4.9.1 CREATE TAG

 $\mbox{\it CREATE TAG}$  creates a tag with the given name in a graph space.

### OpenCypher compatibility

Tags in nGQL are similar to labels in openCypher. But they are also quite different. For example, the ways to create them are different.

- In openCypher, labels are created together with vertices in CREATE statements.
- In nGQL, tags are created separately using CREATE TAG statements. Tags in nGQL are more like tables in MySQL.

### Prerequisites

Running the CREATE TAG statement requires some privileges for the graph space. Otherwise, NebulaGraph throws an error.

### Syntax

To create a tag in a specific graph space, you must specify the current working space with the USE statement.

[TTL\_COL = <prop\_name>]
[COMMENT = '<comment>'];

Parameter	Description
IF NOT EXISTS	Detects if the tag that you want to create exists. If it does not exist, a new one will be created. The tag existence detection here only compares the tag names (excluding properties).
<tag_name></tag_name>	<ol> <li>Each tag name in the graph space must be unique.</li> <li>Tag names cannot be modified after they are set.</li> <li>Tag names cannot start with a number; they support 1-4 byte UTF-8 encoded characters, including English letters (case sensitive), numbers, Chinese characters, etc., but do not include special characters other than underscores. To use special characters, reserved keywords or starting with a number, quote them with backticks (`) and cannot use periods ( .). For more information, see Keywords and reserved words. Note: If you name a tag in Chinese and encounter a SyntaxError, you need to quote the Chinese characters with backticks (`).</li> </ol>
<pre><prop_name></prop_name></pre>	The name of the property. It must be unique for each tag. The rules for permitted property names are the same as those for tag names.
<data_type></data_type>	Shows the data type of each property. For a full description of the property data types, see Data types and Boolean.
NULL \  NOT	Specifies if the property supports $$ NULL $ $ NOT NULL $ $ The default value is $$ NULL $$ DEFAULT $$ must be specified if $$ NOT NULL is set.
DEFAULT	Specifies a default value for a property. The default value can be a literal value or an expression supported by NebulaGraph. If no value is specified, the default value is used when inserting a new vertex.
COMMENT	The remarks of a certain property or the tag itself. The maximum length is 256 bytes. By default, there will be no comments on a tag.
TTL_DURATION	Specifies the life cycle for the property. The property that exceeds the specified TTL expires. The expiration threshold is the TTL_COL value plus the TTL_DURATION. The default value of TTL_DURATION is 0. It means the data never expires.
TTL_COL	Specifies the property to set a timeout on. The data type of the property must be $\[$ int or $\]$ timestamp. A tag can only specify one field as $\[$ TIL_COL $\]$ . For more information on TTL, see $\[$ TTL $\]$ options.

### EXAMPLES

```
mebula> CREATE TAG IF NOT EXISTS player(name string, age int);

# The following example creates a tag with no properties.
nebula> CREATE TAG IF NOT EXISTS no_property();

# The following example creates a tag with a default value.
nebula> CREATE TAG IF NOT EXISTS player_with_default(name string, age int DEFAULT 20);

# In the following example, the TTL of the create_time field is set to be 100 seconds.
nebula> CREATE TAG IF NOT EXISTS woman(name string, age int, \
married bool, salary double, create_time timestamp) \
TTL_DURATION = 100, TTL_COL = "create_time";
```

### Implementation of the operation

Trying to use a newly created tag may fail because the creation of the tag is implemented asynchronously. To make sure the follow-up operations work as expected, Wait for two heartbeat cycles, i.e., 20 seconds.

To change the heartbeat interval, modify the heartbeat\_interval\_secs parameter in the configuration files for all services.

### 4.9.2 DROP TAG

DROP TAG drops a tag with the given name in the current working graph space.

A vertex can have one or more tags.

- If a vertex has only one tag, the vertex **CANNOT** be accessed after you drop it. The vertex will be dropped in the next compaction. But its edges are available, this operation will result in dangling edges.
- If a vertex has multiple tags, the vertex is still accessible after you drop one of them. But all the properties defined by this dropped tag **CANNOT** be accessed.

This operation only deletes the Schema data. All the files or directories in the disk will not be deleted directly until the next compaction.



In NebulaGraph 3.4.1, inserting vertex without tag is not supported by default. If you want to use the vertex without tags, add -- graph\_use\_vertex\_key=true to the configuration files ( nebula-graphd.conf ) of all Graph services in the cluster, and add --use\_vertex\_key=true to the configuration files ( nebula-storaged.conf ) of all Storage services in the cluster.

### Prerequisites

- Running the DROP TAG statement requires some privileges for the graph space. Otherwise, NebulaGraph throws an error.
- Before you drop a tag, make sure that the tag does not have any indexes. Otherwise, the conflict error ( [ERROR (-1005)]: Conflict! ) will be returned when you run the DROP TAG statement. To drop an index, see DROP INDEX.

#### Syntax

DROP TAG [IF EXISTS] <tag\_name>;

- IF NOT EXISTS: Detects if the tag that you want to drop exists. Only when it exists will it be dropped.
- ullet tag\_name: Specifies the tag name that you want to drop. You can drop only one tag in one statement.

### Example

nebula> CREATE TAG IF NOT EXISTS test(p1 string, p2 int);
nebula> DROP TAG test;

Last update: February 19, 2024

- 307/1069 - 2023 Vesoft Inc.

### 4.9.3 ALTER TAG

ALTER TAG alters the structure of a tag with the given name in a graph space. You can add or drop properties, and change the data type of an existing property. You can also set a TTL (Time-To-Live) on a property, or change its TTL duration.

#### Notes

- Running the ALTER TAG statement requires some privileges for the graph space. Otherwise, NebulaGraph throws an error.
- Before you alter properties for a tag, make sure that the properties are not indexed. If the properties contain any indexes, the conflict error [ERROR (-1005)]: Conflict! will occur when you ALTER TAG. For more information on dropping an index, see DROP INDEX.
- The property name must be unique in a tag. If you add a property with the same name as an existing property or a dropped property, the operation fails.

### Syntax

- tag\_name: Specifies the tag name that you want to alter. You can alter only one tag in one statement. Before you alter a tag, make sure that the tag exists in the current working graph space. If the tag does not exist, an error will occur when you alter it.
- Multiple ADD, DROP, and CHANGE clauses are permitted in a single ALTER TAG statement, separated by commas.
- When a property value is set to NOT NULL using ADD or CHANGE, a default value must be specified for the property, that is, the value of DEFAULT must be specified.
- When using CHANGE to modify the data type of a property:
- Only the length of a FIXED\_STRING or an INT can be increased. The length of a STRING or an INT cannot be decreased.
- Only the data type conversions from FIXED\_STRING to STRING and from FLOAT to DOUBLE are allowed.

#### **Examples**

```
nebula> CREATE TAG IF NOT EXISTS tl (pl string, p2 int);
nebula> ALTER TAG tl ADD (p3 int32, fixed_string(10));
nebula> ALTER TAG tl TTL_DURATION = 2, TTL_COL = "p2";
nebula> ALTER TAG tl COMMENT = 'test1';
nebula> ALTER TAG tl ADD (p5 double NOT NULL DEFAULT 0.4 COMMENT 'p5') COMMENT='test2';
// Change the data type of p3 in the TAG tl from INT32 to INT64, and that of p4 from FIXED_STRING(10) to STRING.
nebula> ALTER TAG tl CHANGE (p3 int64, p4 string);
[ERROR(-1005)]: Unsupported!
```

#### Implementation of the operation

Trying to use a newly altered tag may fail because the alteration of the tag is implemented asynchronously. To make sure the follow-up operations work as expected, Wait for two heartbeat cycles, i.e., 20 seconds.

To change the heartbeat interval, modify the heartbeat\_interval\_secs parameter in the configuration files for all services.

### 4.9.4 SHOW TAGS

The  $\mbox{\scriptsize SHOW TAGS}$  statement shows the name of all tags in the current graph space.

You do not need any privileges for the graph space to run the SHOW TAGS statement. But the returned results are different based on role privileges.

### Syntax

```
SHOW TAGS;
```

### Examples



### 4.9.5 DESCRIBE TAG

DESCRIBE TAG returns the information about a tag with the given name in a graph space, such as field names, data type, and so on.

### Prerequisite

 $Running \ the \ {\tt DESCRIBE} \ {\tt TAG} \ statement \ requires \ some \ {\tt privileges} \ for \ the \ graph \ space. \ Otherwise, \ Nebula Graph \ throws \ an \ error.$ 

### Syntax

```
DESC[RIBE] TAG <tag_name>;
```

You can use DESC instead of DESCRIBE for short.

### Example

ebula> DESCRIBE TAG player;			
+	+		
Field   Type   Null   De			
++++	+		
"name"   "string"   "YES"			
"age"   "int64"   "YES"			
	+		

### 4.9.6 DELETE TAG

DELETE TAG deletes a tag with the given name on a specified vertex.

### **Prerequisites**

Running the DELETE TAG statement requires some privileges for the graph space. Otherwise, NebulaGraph throws an error.

### Syntax

```
DELETE TAG <tag_name_List> FROM <VID>;
```

- tag\_name\_list: Specifies the name of the tag. Multiple tags are separated with commas (,). \* means all tags.
- VID: Specifies the VID of the tag to delete.

### Example

# **P**mpatibility

- ullet In openCypher, you can use the statement REMOVE v:LABEL to delete the tag LABEL of the vertex v.
- $\bullet$  Delete TAG and DROP TAG have the same semantics but different syntax. In nGQL, use Delete TAG .

### 4.9.7 Add and delete tags

OpenCypher has the features of SET label and REMOVE label to speed up the process of querying or labeling.

NebulaGraph achieves the same operations by creating and inserting tags to an existing vertex, which can quickly query vertices based on the tag name. Users can also run DELETE TAG to delete some vertices that are no longer needed.

#### **Examples**

For example, in the basketballplayer data set, some basketball players are also team shareholders. Users can create an index for the shareholder tag shareholder for quick search. If the player is no longer a shareholder, users can delete the shareholder tag of the corresponding player by DELETE TAG.



If the index is created after inserting the test data, use the REBUILD TAG INDEX <index\_name\_list>; statement to rebuild the index.

### 4.10 Edge type statements

### 4.10.1 CREATE EDGE

CREATE EDGE creates an edge type with the given name in a graph space.

### OpenCypher compatibility

Edge types in nGQL are similar to relationship types in openCypher. But they are also quite different. For example, the ways to create them are different.

- In openCypher, relationship types are created together with vertices in CREATE statements.
- In nGQL, edge types are created separately using CREATE EDGE statements. Edge types in nGQL are more like tables in MySQL.

### Prerequisites

Running the CREATE EDGE statement requires some privileges for the graph space. Otherwise, NebulaGraph throws an error.

### **Syntax**

To create an edge type in a specific graph space, you must specify the current working space with the USE statement.

[TTL\_COL = <prop\_name>]
[COMMENT = '<comment>'];

Parameter	Description			
IF NOT EXISTS	Detects if the edge type that you want to create exists. If it does not exist, a new one will be created. The edge type existence detection here only compares the edge type names (excluding properties).			
<edge_type_name></edge_type_name>	1. The edge type name must be <b>unique</b> in a graph space.			
	2. Once the edge type name is set, it can not be altered.			
	3. Edge type names cannot start with a number; they support 1-4 byte UTF-8 encoded characters, including			
	English letters (case sensitive), numbers, Chinese characters, etc., but do not include special characters			
	other than underscores. To use special characters, reserved keywords or starting with a number, quote			
	them with backticks (`) and cannot use periods ( . ). For more information, see Keywords and reserved words.			
	Note: If you name an edge type in Chinese and encounter a SyntaxError, you need to quote the Chinese			
	characters with backticks (`).			
<pre><prop_name></prop_name></pre>	The name of the property. It must be unique for each edge type. The rules for permitted property names are the same as those for edge type names.			
<data_type></data_type>	Shows the data type of each property. For a full description of the property data types, see Data types and Boolean.			
NULL \  NOT NULL	Specifies if the property supports $NULL \mid NOT \mid NULL \mid$ . The default value is $NULL \mid$ DEFAULT must be specified if $NOT \mid NULL \mid$ is set.			
DEFAULT	Specifies a default value for a property. The default value can be a literal value or an expression supported by NebulaGraph. If no value is specified, the default value is used when inserting a new edge.			
COMMENT	The remarks of a certain property or the edge type itself. The maximum length is 256 bytes. By default, there will be no comments on an edge type.			
TTL_DURATION	Specifies the life cycle for the property. The property that exceeds the specified TTL expires. The expiration threshold is the TTL_COL value plus the TTL_DURATION. The default value of TTL_DURATION is 0. It means the data never expires.			
TTL_COL	Specifies the property to set a timeout on. The data type of the property must be int or timestamp. An edge type can only specify one field as TTL_COL. For more information on TTL, see TTL options.			

### EXAMPLES

```
nebula> CREATE EDGE IF NOT EXISTS follow(degree int);

# The following example creates an edge type with no properties.
nebula> CREATE EDGE IF NOT EXISTS no_property();

# The following example creates an edge type with a default value.
nebula> CREATE EDGE IF NOT EXISTS follow_with_default(degree int DEFAULT 20);

# In the following example, the TTL of the p2 field is set to be 100 seconds.
nebula> CREATE EDGE IF NOT EXISTS el(p1 string, p2 int, p3 timestamp) \
TTL_DURATION = 100, TTL_COL = "p2";
```

### 4.10.2 DROP EDGE

DROP EDGE drops an edge type with the given name in a graph space.

An edge can have only one edge type. After you drop it, the edge **CANNOT** be accessed. The edge will be deleted in the next compaction.

This operation only deletes the Schema data. All the files or directories in the disk will not be deleted directly until the next compaction.

### **Prerequisites**

- Running the DROP EDGE statement requires some privileges for the graph space. Otherwise, NebulaGraph throws an error.
- Before you drop an edge type, make sure that the edge type does not have any indexes. Otherwise, the conflict error ( [ERROR (-1005)]: Conflict! ) will be returned. To drop an index, see DROP INDEX.

#### Syntax

DROP EDGE [IF EXISTS] <edge\_type\_name>

- IF NOT EXISTS: Detects if the edge type that you want to drop exists. Only when it exists will it be dropped.
- edge\_type\_name: Specifies the edge type name that you want to drop. You can drop only one edge type in one statement.

#### Example

nebula> CREATE EDGE IF NOT EXISTS e1(p1 string, p2 int);
nebula> DROP EDGE e1;

### 4.10.3 ALTER EDGE

ALTER EDGE alters the structure of an edge type with the given name in a graph space. You can add or drop properties, and change the data type of an existing property. You can also set a TTL (Time-To-Live) on a property, or change its TTL duration.

#### Notes

- Running the ALTER EDGE statement requires some privileges for the graph space. Otherwise, NebulaGraph throws an error.
- Before you alter properties for an edge type, make sure that the properties are not indexed. If the properties contain any indexes, the conflict error [ERROR (-1005)]: Conflict! will occur when you ALTER EDGE. For more information on dropping an index, see DROP INDEX
- The property name must be unique in an edge type. If you add a property with the same name as an existing property or a dropped property, the operation fails.
- Only the length of a FIXED\_STRING or an INT can be increased.
- Only the data type conversions from FIXED STRING to STRING and from FLOAT to DOUBLE are allowed.

#### **Syntax**

- edge\_type\_name: Specifies the edge type name that you want to alter. You can alter only one edge type in one statement. Before you alter an edge type, make sure that the edge type exists in the graph space. If the edge type does not exist, an error occurs when you alter it.
- Multiple ADD, DROP, and CHANGE clauses are permitted in a single ALTER EDGE statement, separated by commas.
- When a property value is set to NOT NULL using ADD or CHANGE, a default value must be specified for the property, that is, the value of DEFAULT must be specified.

#### Example

```
nebula> CREATE EDGE e1 ADD (p3 int, p4 string);
nebula> ALTER EDGE e1 TTL_DURATION = 2, TTL_COL = "p2";
nebula> ALTER EDGE e1 COMMENT = 'edge1';
```

### Implementation of the operation

Trying to use a newly altered edge type may fail because the alteration of the edge type is implemented asynchronously. To make sure the follow-up operations work as expected, Wait for two heartbeat cycles, i.e., 20 seconds.

To change the heartbeat interval, modify the heartbeat\_interval\_secs parameter in the configuration files for all services.

### 4.10.4 SHOW EDGES

SHOW EDGES shows all edge types in the current graph space.

You do not need any privileges for the graph space to run the SHOW EDGES statement. But the returned results are different based on role privileges.

### Syntax

SHOW EDGES;

### Example



### 4.10.5 DESCRIBE EDGE

DESCRIBE EDGE returns the information about an edge type with the given name in a graph space, such as field names, data type, and so on.

### Prerequisites

 $Running \ the \ {\tt DESCRIBE} \ {\tt EDGE} \ \ statement \ requires \ some \ {\tt privileges} \ for \ the \ graph \ space. \ Otherwise, \ Nebula Graph \ throws \ an \ error.$ 

### Syntax

```
DESC[RIBE] EDGE <edge_type_name>
```

You can use DESC instead of DESCRIBE for short.

### Example

mahulas DESCRIPE EDGE fallows	
nebula> DESCRIBE EDGE follow;	
++	
Field   Type   Null   Default   Comment	
++	
"degree"   "int64"   "YES"	
++	

### 4.11 Vertex statements

#### 4.11.1 INSERT VERTEX

The INSERT VERTEX statement inserts one or more vertices into a graph space in NebulaGraph.

#### **Prerequisites**

Running the INSERT VERTEX statement requires some privileges for the graph space. Otherwise, NebulaGraph throws an error.

#### **Syntax**

```
INSERT VERTEX [IF NOT EXISTS] [tag_props, [tag_props] ...]
VALUES VID: ([prop_value_list])

tag_props:
    tag_name ([prop_name_list])

prop_name_list:
    [prop_name [, prop_name] ...]

prop_value_list:
    [prop_value [, prop_value] ...]
```

• IF NOT EXISTS detects if the VID that you want to insert exists. If it does not exist, a new one will be inserted.



- IF NOT EXISTS only compares the names of the VID and the tag (excluding properties).
- IF NOT EXISTS will read to check whether the data exists, which will have a significant impact on performance.
- tag\_name denotes the tag (vertex type), which must be created before INSERT VERTEX. For more information, see CREATE TAG.



NebulaGraph 3.4.1 supports inserting vertices without tags.

# mpatibility

In NebulaGraph 3.4.1, inserting vertex without tag is not supported by default. If you want to use the vertex without tags, add -- graph\_use\_vertex\_key=true to the configuration files ( nebula-graphd.conf ) of all Graph services in the cluster, add --use\_vertex\_key=true to the configuration files ( nebula-storaged.conf ) of all Storage services in the cluster. An example of a command to insert a vertex without tag is INSERT VERTEX VALUES "1":();

- $\bullet$  prop\_name\_list contains the names of the properties on the tag.
- WID is the vertex ID. In NebulaGraph 2.0, string and integer VID types are supported. The VID type is set when a graph space is created. For more information, see CREATE SPACE.
- prop\_value\_list must provide the property values according to the prop\_name\_list. When the NOT NULL constraint is set for a given property, an error is returned if no property is given. When the default value for a property is NULL, you can omit to specify the property value. For details, see CREATE TAG.



INSERT VERTEX and CREATE have different semantics.

- The semantics of INSERT VERTEX is closer to that of INSERT in NoSQL (key-value), or UPSERT (UPDATE or INSERT) in SQL.
- When two INSERT statements (neither uses IF NOT EXISTS) with the same VID and TAG are operated at the same time, the latter INSERT will overwrite the former.
- When two INSERT statements with the same VID but different TAGS are operated at the same time, the operation of different tags will not overwrite each other.

Examples are as follows.

| {p1: "shala", p2: 4, p3: \_\_NULL\_\_} |

#### **Examples**

```
# Insert a vertex without tag.
nebula> INSERT VERTEX VALUES "1":();

# The following examples create tag t1 with no property and inserts vertex "10" with no property.
nebula> CREATE TAG IF NOT EXISTS t1();
nebula> INSERT VERTEX t1() VALUES "10":();

nebula> CREATE TAG IF NOT EXISTS t2 (name string, age int);
nebula> INSERT VERTEX t2 (name, age) VALUES "11":("n1", 12);

# In the following example, the insertion fails because "a13" is not int.
nebula> INSERT VERTEX t2 (name, age) VALUES "12":("n1", "a13");

# The following example inserts two vertices at one time.
nebula> INSERT VERTEX t2 (name, age) VALUES "13":("n3", 12), "14":("n4", 8);

nebula> CREATE TAG IF NOT EXISTS t3(p1 int);
nebula> CREATE TAG IF NOT EXISTS t4(p2 string);

# The following example inserts vertex "21" with two tags.
nebula> INSERT VERTEX t3 (p1), t4(p2) VALUES "21": (321, "hello");
```

A vertex can be inserted/written with new values multiple times. Only the last written values can be read.

If you insert a vertex that already exists with IF NOT EXISTS, there will be no modification.

### 4.11.2 DELETE VERTEX

By default, the DELETE VERTEX statement deletes vertices but the incoming and outgoing edges of the vertices.



- NebulaGraph 2.x deletes vertices and their incoming and outgoing edges.
- NebulaGraph 3.4.1 only deletes the vertices, and does not delete the related outgoing and incoming edges of the vertices. At this time, there will be dangling edges by default.

The DELETE VERTEX statement deletes one vertex or multiple vertices at a time. You can use DELETE VERTEX together with pipes. For more information about pipe, see Pipe operator.



- DELETE VERTEX deletes vertices directly.
- DELETE TAG deletes a tag with the given name on a specified vertex.

#### **Syntax**

```
DELETE VERTEX <vid>[, <vid>...] [WITH EDGE];
```

• WITH EDGE: deletes vertices and the related incoming and outgoing edges of the vertices.

### **Examples**

This query deletes the vertex whose ID is "team1".

```
# Delete the vertex whose VID is `team1` but the related incoming and outgoing edges are not deleted.

nebula> DELETE VERTEX "team1";

# Delete the vertex whose VID is `team1` and the related incoming and outgoing edges.

nebula> DELETE VERTEX "team1" WITH EDGE;
```

This query shows that you can use DELETE VERTEX together with pipe to delete vertices.

```
nebula> GO FROM "player100" OVER serve WHERE properties(edge).start_year == "2021" YIELD dst(edge) AS id | DELETE VERTEX $-.id;
```

### Process of deleting vertices

Once NebulaGraph deletes the vertices, all edges (incoming and outgoing edges) of the target vertex will become dangling edges. When NebulaGraph deletes the vertices <code>WITH EDGE</code>, NebulaGraph traverses the incoming and outgoing edges related to the vertices and deletes them all. Then NebulaGraph deletes the vertices.



- Atomic deletion is not supported during the entire process for now. Please retry when a failure occurs to avoid partial deletion, which will cause pendent edges.
- Deleting a supernode takes a lot of time. To avoid connection timeout before the deletion is complete, you can modify the parameter --storage\_client\_timeout\_ms in nebula-graphd.conf to extend the timeout period.

Last update: February 19, 2024

- 323/1069 - 2023 Vesoft Inc.

### 4.11.3 UPDATE VERTEX

The UPDATE VERTEX statement updates properties on tags of a vertex.

In NebulaGraph, UPDATE VERTEX supports compare-and-set (CAS).



An  $\mbox{\sc update}$  vertex statement can only update properties on  $\mbox{\sc ONE}$   $\mbox{\sc TAG}$  of a vertex.

### Syntax

```
UPDATE VERTEX ON <tag_name> <vid>
SET =update_prop>
[WHEN <condition>]
[YIELD <output>]
```

Parameter	Required	Description	Example
ON <tag_name></tag_name>	Yes	Specifies the tag of the vertex. The properties to be updated must be on this tag.	ON player
<vid></vid>	Yes	Specifies the ID of the vertex to be updated.	"player100"
SET <update_prop></update_prop>	Yes	Specifies the properties to be updated and how they will be updated.	SET age = age +1
WHEN <condition></condition>	No	Specifies the filter conditions. If <condition> evaluates to false, the SET clause will not take effect.</condition>	WHEN name == "Tim"
YIELD <output></output>	No	Specifies the output format of the statement.	YIELD name AS

### Example

## 4.11.4 UPSERT VERTEX

The UPSERT statement is a combination of UPDATE and INSERT. You can use UPSERT VERTEX to update the properties of a vertex if it exists or insert a new vertex if it does not exist.



An UPSERT VERTEX statement can only update the properties on ONE TAG of a vertex.

The performance of UPSERT is much lower than that of INSERT because UPSERT is a read-modify-write serialization operation at the partition level.



Don't use UPSERT for scenarios with highly concurrent writes. You can use UPDATE or INSERT instead.

#### **Syntax**

UPSERT VERTEX ON <tag> <vid>
SET <update\_prop>
[WHEN <condition>]
[YIELD <output>]

Parameter	Required	Description	Example
ON <tag></tag>	Yes	Specifies the tag of the vertex. The properties to be updated must be on this tag.	ON player
<vid></vid>	Yes	Specifies the ID of the vertex to be updated or inserted.	"player100"
SET <update_prop></update_prop>	Yes	Specifies the properties to be updated and how they will be updated.	SET age = age +1
WHEN <condition></condition>	No	Specifies the filter conditions.	WHEN name == "Tim"
YIELD <output></output>	No	Specifies the output format of the statement.	YIELD name AS

#### Insert a vertex if it does not exist

If a vertex does not exist, it is created no matter the conditions in the WHEN clause are met or not, and the SET clause always takes effect. The property values of the new vertex depend on:

- How the SET clause is defined.
- Whether the property has a default value.

For example, if:

- The vertex to be inserted will have properties name and age based on the tag player.
- ullet The SET clause specifies that age = 30.

Then the property values in different cases are listed as follows:

Are WHEN conditions met	If properties have default values	Value of name	Value of age
Yes	Yes	The default value	30
Yes	No	NULL	30
No	Yes	The default value	30
No	No	NULL	30

## Here are some examples:

```
// This query checks if the following three vertices exist. The result "Empty set" indicates that the vertices do not exist. nebula> FETCH PROP ON * "player666", "player667", "player668" YIELD properties(vertex);
| properties(VERTEX) |
Empty set
nebula> UPSERT VERTEX ON player "player666" \
          SET age = 30 \
WHEN name == "Joe" \
          YIELD name AS Name, age AS Age;
Name Age
| __NULL__ | 30
nebula> UPSERT VERTEX ON player "player666" \
         SET age = 31 \
WHEN name == "Joe" \
YIELD name AS Name, age AS Age;
| Name | Age |
| __NULL__ | 30 |
nebula> UPSERT VERTEX ON player "player667" \setminus
          SET age = 31 \
YIELD name AS Name, age AS Age;
| Name | Age |
| __NULL__ | 31 |
nebula> UPSERT VERTEX ON player "player668" \
SET name = "Amber", age = age + 1 \
YIELD name AS Name, age AS Age;
| Name | Age
| "Amber" | __NULL__ |
```

In the last query of the preceding examples, since age has no default value, when the vertex is created, age is NULL, and age = age + 1 does not take effect. But if age has a default value, age = age + 1 will take effect. For example:

#### Update a vertex if it exists

If the vertex exists and the WHEN conditions are met, the vertex is updated.

```
nebula> FETCH PROP ON player "player101" YIELD properties(vertex);
```

If the vertex exists and the WHEN conditions are not met, the update does not take effect.

## 4.12 Edge statements

#### 4.12.1 INSERT EDGE

The INSERT EDGE statement inserts an edge or multiple edges into a graph space from a source vertex (given by src\_vid) to a destination vertex (given by dst\_vid) with a specific rank in NebulaGraph.

When inserting an edge that already exists, INSERT EDGE overrides the edge.

## Syntax

```
INSERT EDGE [IF NOT EXISTS] <edge_type> ( <prop_name_list> ) VALUES
<src_vid> -> <dst_vid>[@<rank>] : ( <prop_value_list> )
[, <src_vid> -> <dst_vid>[@<rank>] : ( <prop_value_list> ), ...];
<prop_name_list> ::=
[ <prop_name> [, <prop_name> ] ...]

<prop_value_list> ::=
[ <prop_value> [, <prop_value> ] ...]
```

• IF NOT EXISTS detects if the edge that you want to insert exists. If it does not exist, a new one will be inserted.



- IF NOT EXISTS only detects whether exist and does not detect whether the property values overlap.
- IF NOT EXISTS will read to check whether the data exists, which will have a significant impact on performance.
- <edge\_type> denotes the edge type, which must be created before INSERT EDGE. Only one edge type can be specified in this
  statement.
- prop\_name\_list> is the property name list in the given <edge\_type> .
- src\_vid is the VID of the source vertex. It specifies the start of an edge.
- dst\_vid is the VID of the destination vertex. It specifies the end of an edge.
- rank is optional. It specifies the edge rank of the same edge type. The data type is int. If not specified, the default value is 0. You can insert many edges with the same edge type, source vertex, and destination vertex by using different rank values.

# PenCypher compatibility

OpenCypher has no such concept as rank.

#### **Examples**

```
# The following example creates edge type el with no property and inserts an edge from vertex "10" to vertex "11" with no property.
nebula> CREATE EDGE IF NOT EXISTS el();
nebula> INSERT EDGE el () VALUES "10"->"11":();

# The following example inserts an edge from vertex "10" to vertex "11" with no property. The edge rank is 1.
nebula> INSERT EDGE el () VALUES "10"->"11"@1:();
```

```
nebula> CREATE EDGE IF NOT EXISTS e2 (name string, age int);
nebula> INSERT EDGE e2 (name, age) VALUES "11"->"13":("n1", 1);

# The following example creates edge type e2 with two properties.
nebula> INSERT EDGE e2 (name, age) VALUES \
    "12"->"13":("n1", 1), "13"->"14":("n2", 2);

# In the following example, the insertion fails because "a13" is not int.
nebula> INSERT EDGE e2 (name, age) VALUES "11"->"13":("n1", "a13");
```

An edge can be inserted/written with property values multiple times. Only the last written values can be read.

If you insert an edge that already exists with IF NOT EXISTS, there will be no modification.



- NebulaGraph 3.4.1 allows dangling edges. Therefore, you can write the edge before the source vertex or the destination vertex exists. At this time, you can get the (not written) vertex VID through <edgetype>.\_src or <edgetype>.\_dst (which is not recommended).
- Atomic operation is not guaranteed during the entire process for now. If it fails, please try again. Otherwise, partial writing will occur. At this time, the behavior of reading the data is undefined. For example, if multiple machines are involved in the write operation, only one of the forward and reverse edges of a single edge is written successfully, or only part of the edge is written successfully when multiple edges are inserted. In this case, an error will be returned, so please execute the command again.
- · Concurrently writing the same edge will cause an edge conflict error, so please try again later.
- The inserting speed of an edge is about half that of a vertex. Because in the storaged process, the insertion of an edge involves two tasks, while the insertion of a vertex involves only one task.

## 4.12.2 DELETE EDGE

The DELETE EDGE statement deletes one edge or multiple edges at a time. You can use DELETE EDGE together with pipe operators. For more information, see PIPE OPERATORS.

To delete all the outgoing edges for a vertex, please delete the vertex. For more information, see DELETE VERTEX.

## **Syntax**

```
DELETE EDGE <edge_type> <src_vid> -> <dst_vid>[@<rank>] [, <src_vid> -> <dst_vid>[@<rank>] ...]
```



If no rank is specified, NebulaGraph only deletes the edge with rank 0. Delete edges with all ranks, as shown in the following example.

## **Examples**

```
nebula> DELETE EDGE serve "player100" -> "team204"@0;
```

The following example shows that you can use DELETE EDGE together with pipe operators to delete edges that meet the conditions.

```
nebula> GO FROM "player100" OVER follow \
    WHERE dst(edge) == "player101" \
    YIELD src(edge) AS src, dst(edge) AS dst, rank(edge) AS rank \
    | DELETE EDGE follow $-.src->$-.dst @ $-.rank;
```

## 4.12.3 UPDATE EDGE

The UPDATE EDGE statement updates properties on an edge.

In NebulaGraph, UPDATE EDGE supports compare-and-swap (CAS).

#### **Syntax**

```
UPDATE EDGE ON <edge_type>
<src_vid> -> <dst_vid> [@<rank>]

SET <update_prop>
[WHEN <condition>]
[YIELD <output>]
```

Parameter	Required	Description	Example
ON <edge_type></edge_type>	Yes	Specifies the edge type. The properties to be updated must be on this edge type.	ON serve
<pre><src_vid></src_vid></pre>	Yes	Specifies the source vertex ID of the edge.	"player100"
<dst_vid></dst_vid>	Yes	Specifies the destination vertex ID of the edge.	"team204"
<rank></rank>	No	Specifies the rank of the edge. The data type is $% \left( 1\right) =\left( 1\right) +\left( 1\right) =\left( 1\right) +\left( 1\right) =\left( 1\right) =\left( 1\right) +\left( 1\right) =\left( 1\right) =\left( 1\right) +\left( 1\right) =\left( 1\right) =\left$	10
<pre>SET <update_prop></update_prop></pre>	Yes	Specifies the properties to be updated and how they will be updated.	<pre>SET start_year = start_year +1</pre>
WHEN <condition></condition>	No	Specifies the filter conditions. If <condition> evaluates to false, the SET clause does not take effect.</condition>	WHEN end_year < 2010
YIELD <output></output>	No	Specifies the output format of the statement.	YIELD start_year AS Start_Year

## Example

The following example checks the properties of the edge with the GO statement.

The following example updates the  $\,$  start\_year  $\,$  property and returns the  $\,$  end\_year  $\,$  and the new  $\,$  start\_year  $\,$  .

## 4.12.4 UPSERT EDGE

The UPSERT statement is a combination of UPDATE and INSERT. You can use UPSERT EDGE to update the properties of an edge if it exists or insert a new edge if it does not exist.

The performance of UPSERT is much lower than that of INSERT because UPSERT is a read-modify-write serialization operation at the partition level.



Do not use UPSERT for scenarios with highly concurrent writes. You can use UPDATE or INSERT instead.

## **Syntax**

UPSERT EDGE ON <edge\_type>
<src\_vid> -> <dst\_vid> [@rank]
SET <update\_prop>
[WHEN <condition>]
[YIELD 
[

Parameter	Required	Description	Example
ON <edge_type></edge_type>	Yes	Specifies the edge type. The properties to be updated must be on this edge type.	ON serve
<src_vid></src_vid>	Yes	Specifies the source vertex ID of the edge.	"player100"
<dst_vid></dst_vid>	Yes	Specifies the destination vertex ID of the edge.	"team204"
<rank></rank>	No	Specifies the rank of the edge.	10
SET <update_prop></update_prop>	Yes	Specifies the properties to be updated and how they will be updated.	<pre>SET start_year = start_year +1</pre>
WHEN <condition></condition>	No	Specifies the filter conditions.	WHEN end_year < 2010
YIELD <output></output>	No	Specifies the output format of the statement.	YIELD start_year AS Start_Year

## Insert an edge if it does not exist

If an edge does not exist, it is created no matter the conditions in the WHEN clause are met or not, and the SET clause takes effect. The property values of the new edge depend on:

- How the SET clause is defined.
- Whether the property has a default value.

For example, if:

- The edge to be inserted will have properties start\_year and end\_year based on the edge type serve.
- The SET clause specifies that end\_year = 2021.

Then the property values in different cases are listed as follows:

Are WHEN conditions met	If properties have default values	Value of start_year	Value of end_year
Yes	Yes	The default value	2021
Yes	No	NULL	2021
No	Yes	The default value	2021
No	No	NULL	2021

## Here are some examples:

```
// This example checks if the following three vertices have any outgoing serve edge. The result "Empty set" indicates that such an edge does not exist.
nebula> GO FROM "player666", "player667", "player668" \
        OVER serve
        YIELD properties(edge).start_year, properties(edge).end_year;
| properties(EDGE).start_year | properties(EDGE).end_year |
+-----+
Empty set
nebula> UPSERT EDGE on serve \
    "player666" -> "team200"@0 \
        SET end_year = 2021 \
        WHEN end_year == 2010
        YIELD start_year, end_year;
| start year | end year
| __NULL__ | 2021
nebula> UPSERT EDGE on serve \
        "player666" -> "team200"@0 \
SET end_year = 2022 \
WHEN end_year == 2010 \
        YIELD start_year, end_year;
| start_year | end_year |
| __NULL__ | 2021
nebula> UPSERT EDGE on serve \
    "player667" -> "team200"@0 \
    SET end_year = 2022 \
        YIELD start_year, end_year;
| start_year | end_year
| __NULL__ | 2022
+-----
nebula> UPSERT EDGE on serve \
    "player668" -> "team200"@0 \
        SET start_year = 2000, end_year = end_year + 1 \
        YIELD start_year, end_year;
| start_year | end_year
```

In the last query of the preceding example, since end\_year has no default value, when the edge is created, end\_year is NULL, and end\_year = end\_year + 1 does not take effect. But if end\_year has a default value, end\_year = end\_year + 1 will take effect. For example:

## Update an edge if it exists

If the edge exists and the WHEN conditions are met, the edge is updated.

If the edge exists and the WHEN conditions are not met, the update does not take effect.

## 4.13 Native index statements

#### 4.13.1 Index overview

Indexes are built to fast process graph queries. Nebula Graph supports two kinds of indexes: native indexes and full-text indexes. This topic introduces the index types and helps choose the right index.

#### Native indexes

Native indexes allow querying data based on a given property. Features are as follows.

- There are two kinds of native indexes: tag index and edge type index.
- Native indexes must be updated manually. You can use the REBUILD INDEX statement to update native indexes.
- Native indexes support indexing multiple properties on a tag or an edge type (composite indexes), but do not support indexing across multiple tags or edge types.

OPERATIONS ON NATIVE INDEXES

- CREATE INDEX
- SHOW CREATE INDEX
- SHOW INDEXES
- DESCRIBE INDEX
- REBUILD INDEX
- SHOW INDEX STATUS
- DROP INDEX
- LOOKUP
- MATCH

## Full-text indexes

Full-text indexes are used to do prefix, wildcard, regexp, and fuzzy search on a string property. Features are as follows.

- Full-text indexes allow indexing just one property.
- $\bullet$  Full-text indexes do not support logical operations such as  $\mbox{\sc AND}$  ,  $\mbox{\sc OR}$  , and  $\mbox{\sc NOT}$  .



To do complete string matches, use native indexes.

#### Null values

Indexes do not support indexing null values.

## Range queries

In addition to querying single results from native indexes, you can also do range queries. Not all the native indexes support range queries. You can only do range searches for numeric, date, and time type properties.

## 4.13.2 CREATE INDEX

#### **Prerequisites**

Before you create an index, make sure that the relative tag or edge type is created. For how to create tags or edge types, see CREATE TAG and CREATE EDGE.

For how to create full-text indexes, see Deploy full-text index.

#### Must-read for using indexes

The concept and using restrictions of indexes are comparatively complex. You can use it together with LOOKUP and MATCH statements.

You can use CREATE INDEX to add native indexes for the existing tags, edge types, or properties. They are usually called as tag indexes, edge type indexes, and property indexes.

- Tag indexes and edge type indexes apply to queries related to the tag and the edge type, but do not apply to queries that are based on certain properties on the tag. For example, you can use LOOKUP to retrieve all the vertices with the tag player.
- Property indexes apply to property-based queries. For example, you can use the age property to retrieve the VID of all vertices that meet age == 19.

If a property index  $i_TA$  is created for the property A of the tag T and  $i_T$  for the tag T, the indexes can be replaced as follows (the same for edge type indexes):

- The query engine can use i\_TA to replace i\_T.
- In the MATCH statement, i\_T cannot replace i\_TA for querying properties.
- In the LOOKUP statement, i\_T may replace i\_TA for querying properties.

## Lacy version compatibility

In previous releases, the tag or edge type index in the LOOKUP statement cannot replace the property index for property queries.

Although the same results can be obtained by using alternative indexes for queries, the query performance varies according to the selected index.

## Caution

Indexes can dramatically reduce the write performance. The performance can be greatly reduced. **DO NOT** use indexes in production environments unless you are fully aware of their influences on your service.

Indexes cannot make queries faster. It can only locate a vertex or an edge according to properties or count the number of vertices or edges.

Long indexes decrease the scan performance of the Storage Service and use more memory. We suggest that you set the indexing length the same as that of the longest string to be indexed. The longest index length is 256 bytes.

- 337/1069 - 2023 Vesoft Inc.

If you must use indexes, we suggest that you:

- 1. Import the data into NebulaGraph.
- 2. Create indexes.
- 3. Rebuild indexes.
- 4. After the index is created and the data is imported, you can use LOOKUP or MATCH to retrieve the data. You do not need to specify which indexes to use in a query, NebulaGraph figures that out by itself.



If you create an index before importing the data, the importing speed will be extremely slow due to the reduction in the write performance.

Keep --disable\_auto\_compaction = false during daily incremental writing.

The newly created index will not take effect immediately. Trying to use a newly created index (such as LOOKUP or REBUILD INDEX) may fail and return can't find xxx in the space because the creation is implemented asynchronously. To make sure the follow-up operations work as expected, Wait for two heartbeat cycles, i.e., 20 seconds. To change the heartbeat interval, modify the heartbeat\_interval\_secs in the configuration files for all services.



After creating a new index, or dropping the old index and creating a new one with the same name again, you must REBUILD INDEX. Otherwise, these data cannot be returned in the MATCH and LOOKUP statements.

## Syntax

CREATE {TAG | EDGE} INDEX [IF NOT EXISTS] <index\_name> ON {<tag\_name> | <edge\_name>} ([<prop\_name\_list>]) [COMMENT '<comment>'];

Parameter Description

Parameter	Description	
TAG   EDGE	Specifies the index type that you want to create.	
IF NOT EXISTS	Detects if the index that you want to create exists. If it does not exist, a new one will be created.	
<index_name></index_name>	The name of the index. It must be unique in a graph space. A recommended way of naming is <code>i_tagName_propName</code> .  Index names cannot start with a number. They supports 1 to 4 bytes UTF-8 encoded characters, such as English letters (case-sensitive), numbers, and Chinese characters, but does not support special characters except underscores. To use special characters, reserved keywords, or starting with a number, quote them with backticks. For more information, see <code>Keywords</code> and <code>reserved</code> words.  Note: If you name an index in Chinese and encounter a <code>SyntaxError</code> , you need to quote the Chinese characters with backticks (`).	
<tag_name>   <edge_name></edge_name></tag_name>	Specifies the name of the tag or edge associated with the index.	
<pre><prop_name_list></prop_name_list></pre>	To index a <b>variable-length</b> string property, you must use <code>prop_name(length)</code> to specify the index length. To index a tag or an edge type, ignore the <code>prop_name_list</code> .	
COMMENT	The remarks of the index. The maximum length is 256 bytes. By default, there will be no comments on an index.	

## Create tag/edge type indexes

nebula> CREATE TAG INDEX IF NOT EXISTS player\_index on player();

nebula> CREATE EDGE INDEX IF NOT EXISTS follow index on follow();

- 338/1069 - 2023 Vesoft Inc.

After indexing a tag or an edge type, you can use the LOOKUP statement to retrieve the VID of all vertices with the tag, or the source vertex ID, destination vertex ID, and ranks of all edges with the edge type. For more information, see LOOKUP.

#### Create single-property indexes

```
nebula> CREATE TAG INDEX IF NOT EXISTS player_index_0 on player(name(10));
```

The preceding example creates an index for the name property on all vertices carrying the player tag. This example creates an index using the first 10 characters of the name property.

```
# To index a variable-length string property, you need to specify the index length.

nebula> CREATE TAG IF NOT EXISTS var_string(p1 string);

nebula> CREATE TAG INDEX IF NOT EXISTS var ON var_string(p1(10));

# To index a fixed-length string property, you do not need to specify the index length.

nebula> CREATE TAG IF NOT EXISTS fix_string(p1 FIXED_STRING(10));

nebula> CREATE TAG INDEX IF NOT EXISTS fix ON fix_string(p1);

nebula> CREATE EDGE INDEX IF NOT EXISTS follow_index_0 on follow(degree);
```

#### Create composite property indexes

An index on multiple properties on a tag (or an edge type) is called a composite property index.

nebula> CREATE TAG INDEX IF NOT EXISTS player\_index\_1 on player(name(10), age);



Creating composite property indexes across multiple tags or edge types is not supported.



NebulaGraph follows the left matching principle to select indexes.

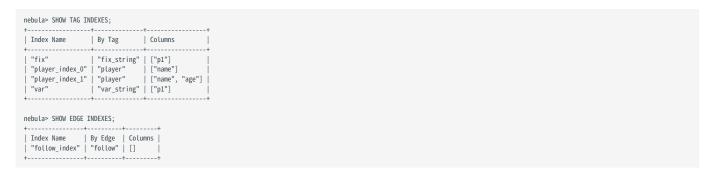
## 4.13.3 SHOW INDEXES

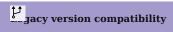
SHOW INDEXES shows the defined tag or edge type indexes names in the current graph space.

## Syntax

SHOW {TAG | EDGE} INDEXES

## Examples





In NebulaGraph 2.x, the  $\mbox{SHOW TAG/EDGE\ INDEXES\ statement\ only\ returns\ Names\ .}$ 

## 4.13.4 SHOW CREATE INDEX

SHOW CREATE INDEX shows the statement used when creating a tag or an edge type. It contains detailed information about the index, such as its associated properties.

#### **Syntax**

```
SHOW CREATE {TAG | EDGE} INDEX <index_name>;
```

## **Examples**

You can run SHOW TAG INDEXES to list all tag indexes, and then use SHOW CREATE TAG INDEX to show the information about the creation of the specified index.



Edge indexes can be queried through a similar approach.



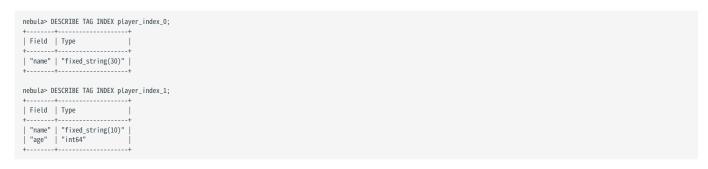
## 4.13.5 DESCRIBE INDEX

DESCRIBE INDEX can get the information about the index with a given name, including the property name (Field) and the property type (Type) of the index.

## Syntax

```
DESCRIBE {TAG | EDGE} INDEX <index_name>;
```

## **Examples**



## 4.13.6 REBUILD INDEX



- If data is updated or inserted before the creation of the index, you must rebuild the indexes **manually** to make sure that the indexes contain the previously added data. Otherwise, you cannot use LOOKUP and MATCH to query the data based on the index. If the index is created before any data insertion, there is no need to rebuild the index.
- When the rebuild of an index is incomplete, queries that rely on the index can use only part of the index and therefore cannot obtain
  accurate results.

You can use REBUILD INDEX to rebuild the created tag or edge type index. For details on how to create an index, see CREATE INDEX.



The speed of rebuilding indexes can be optimized by modifying the rebuild\_index\_part\_rate\_limit and snapshot\_batch\_size parameters in the configuration file. In addition, greater parameter values may result in higher memory and network usage, see Storage Service configurations for details.

## Syntax

```
REBUILD {TAG | EDGE} INDEX [<index_name_list>];

<index_name_list>::=
    [index_name [, index_name] ...]
```

- Multiple indexes are permitted in a single REBUILD statement, separated by commas. When the index name is not specified, all tag or edge indexes are rebuilt.
- After the rebuilding is complete, you can use the SHOW {TAG | EDGE} INDEX STATUS command to check if the index is successfully rebuilt. For details on index status, see SHOW INDEX STATUS.

## Examples



NebulaGraph creates a job to rebuild the index. The job ID is displayed in the preceding return message. To check if the rebuilding process is complete, use the SHOW JOB <job\_id> statement. For more information, see SHOW JOB.

## 4.13.7 SHOW INDEX STATUS

SHOW INDEX STATUS returns the name of the created tag or edge type index and its status of job.

The index status includes:

• QUEUE: The job is in a queue.

• RUNNING: The job is running.

• FINISHED: The job is finished.

• FAILED: The job has failed.

• STOPPED: The job has stopped.

• INVALID: The job is invalid.



For details on how to create an index, see CREATE INDEX.

## Syntax

SHOW {TAG | EDGE} INDEX STATUS;

## Example



## 4.13.8 DROP INDEX

DROP INDEX removes an existing index from the current graph space.

## Prerequisite

Running the DROP INDEX statement requires some privileges of DROP TAG INDEX and DROP EDGE INDEX in the given graph space. Otherwise, NebulaGraph throws an error.

## Syntax

DROP {TAG | EDGE} INDEX [IF EXISTS] <index\_name>;

IF EXISTS: Detects whether the index that you want to drop exists. If it exists, it will be dropped.

## Example

nebula> DROP TAG INDEX player\_index\_0;

## 4.14 Full-text index statements

## 4.14.1 Full-text index restrictions



This topic introduces the restrictions for full-text indexes. Please read the restrictions very carefully before using the full-text indexes.

For now, full-text search has the following limitations:

- Currently, full-text search supports LOOKUP statements only.
- The full-text index name must starts with <code>nebula\_</code> and can contain only numbers, lowercase letters, underscores, and strings no longer than 256 bytes.
- If there is a full-text index on the tag/edge type, the tag/edge type cannot be deleted or modified.
- The type of properties must be STRING or FIXED\_STRING.
- Full-text index can not be applied to search multiple tags/edge types.
- Sorting for the returned results of the full-text search is not supported. Data is returned in the order of data insertion.
- Full-text index can not search properties with value NULL.
- Altering Elasticsearch indexes is not supported at this time.
- $\bullet$  The pipe operator is not supported.
- ullet where clauses supports full-text search only working on single terms.
- Make sure that you start the Elasticsearch cluster and Nebula Graph at the same time. If not, the data writing on the Elasticsearch cluster can be incomplete.
- It may take a while for Elasticsearch to create indexes. If Nebula Graph warns no index is found, wait for the index to take effect (however, the waiting time is unknown and there is no code to check).
- $\bullet$  Nebula Graph clusters deployed with K8s do not support the full-text search feature.

Last update: February 19, 2024

- 347/1069 - 2023 Vesoft Inc.

## 4.14.2 Deploy full-text index

Nebula Graph full-text indexes are powered by Elasticsearch. This means that you can use Elasticsearch full-text query language to retrieve what you want. Full-text indexes are managed through built-in procedures. They can be created only for variable STRING and FIXED\_STRING properties when the listener cluster and the Elasticsearch cluster are deployed.

#### Precaution

Before you start using the full-text index, please make sure that you know the restrictions.

#### Deploy Elasticsearch cluster

To deploy an Elasticsearch cluster, see Kubernetes Elasticsearch deployment or Elasticsearch installation. Currently only 7.x versions of Elasticsearch are supported.



For NebulaGraph 3.4 and later versions, no additional templates need to be created.



The full-text index name must starts with nebula\_.

You can configure the Elasticsearch to meet your business needs. To customize the Elasticsearch, see Elasticsearch Document.

## Sign in to the text search clients

When the Elasticsearch cluster is deployed, use the SIGN IN statement to sign in to the Elasticsearch clients. Multiple elastic\_ip:port pairs are separated with commas. You must use the IPs and the port number in the configuration file for the Elasticsearch.

SYNTAX

```
SIGN IN TEXT SERVICE (<elastic_ip:port>, {HTTP | HTTPS} [,"<username>", "<password>"]) [, (<elastic_ip:port>, ...)];
```

#### EXAMPLE

nebula> SIGN IN TEXT SERVICE (127.0.0.1:9200, HTTP);



Elasticsearch does not have a username or password by default. If you configured a username and password, you need to specify them in the SIGN IN statement.



The Elasticsearch client can only be logged in once, and if there are changes, you need to SIGN OUT and then SIGN IN again, and the client takes effect globally, and multiple graph spaces share the same Elasticsearch client.

## Show text search clients

The SHOW TEXT SEARCH CLIENTS statement can list the text search clients.

- 348/1069 - 2023 Vesoft Inc.

## SYNTAX

SHOW TEXT SEARCH CLIENTS;

## EXAMPLE

## Sign out to the text search clients

The SIGN OUT TEXT SERVICE statement can sign out all the text search clients.

SYNTAX

SIGN OUT TEXT SERVICE;

## EXAMPLE

nebula> SIGN OUT TEXT SERVICE;

## 4.14.3 Deploy Raft Listener for NebulaGraph Storage service

Full-text index data is written to the Elasticsearch cluster asynchronously. The Raft Listener (Listener for short) is a separate process that fetches data from the Storage Service and writes them into the Elasticsearch cluster.

#### Prerequisites

- You have read and fully understood the restrictions for using full-text indexes.
- You have deployed a NebulaGraph cluster.
- You have deploy a Elasticsearch cluster.
- You have prepared at least one extra Storage Server. To use the full-text search, you must run one or more Storage Server as the Raft Listener.

#### **Precautions**

- The Storage Service that you want to run as the Listener must have the same or later release with all the other Nebula Graph services in the cluster.
- For now, you can only add all Listeners to a graph space once and for all. Trying to add a new Listener to a graph space that already has a Listener will fail. To add all Listeners, set them in one statement.

#### **Deployment process**

STEP 1: INSTALL THE STORAGE SERVICE

The Listener process and the storaged process use the same binary file. However, their configuration files and using ports are different. You can install NebulaGraph on all servers that need to deploy a Listener, but only the Storage service can be used. For details, see Install NebulaGraph by RPM or DEB Package.

STEP 2: PREPARE THE CONFIGURATION FILE FOR THE LISTENER

You have to prepare a corresponding configuration file on the machine that you want to deploy a Listener. The file must be named as nebula-storaged-listener.conf and stored in the etc directory. A template is provided for your reference. Note that the file suffix .production should be removed.

- 350/1069 - 2023 Vesoft Inc.

Most configurations are the same as the configurations of Storage Service. This topic only introduces the differences.

Name	Default value	Description
daemonize	true	When set to true, the process is a daemon process.
pid_file	pids/nebula-metad- listener.pid	The file that records the process ID.
meta_server_addrs	-	IP addresses and ports of all Meta services. Multiple Meta services are separated by commas.
local_ip	-	The local IP address of the Listener service.
port	-	The listening port of the RPC daemon of the Listener service.
heartbeat_interval_secs	10	The heartbeat interval of the Meta service. The unit is second (s).
listener_path	data/listener	The WAL directory of the Listener. Only one directory is allowed.
data_path	data	For compatibility reasons, this parameter can be ignored. Fill in the default value $\mbox{\scriptsize data}.$
part_man_type	memory	The type of the part manager. Optional values are $\ensuremath{memory}$ and $\ensuremath{meta}$ .
rocksdb_batch_size	4096	The default reserved bytes for batch operations.
rocksdb_block_cache	4	The default block cache size of BlockBasedTable. The unit is Megabyte (MB).
engine_type	rocksdb	The type of the Storage engine, such as rocksdb, memory, etc.
part_type	simple	The type of the part, such as simple, consensus, etc.



Use real IP addresses in the configuration file instead of domain names or loopback IP addresses such as 127.0.0.1.

## STEP 3: START LISTENERS

Run the following command to start the Listener.

./bin/nebula-storaged --flagfile <listener\_config\_path>/nebula-storaged-listener.conf

\${listener\_config\_path} is the path where you store the Listener configuration file.

STEP 4: ADD LISTENERS TO NEBULAGRAPH

Connect to NebulaGraph and run USE <space> to enter the graph space that you want to create full-text indexes for. Then run the following statement to add a Listener into NebulaGraph.

ADD LISTENER ELASTICSEARCH <listener\_ip:port> [,<listener\_ip:port>, ...]



You must use real IPs for a Listener.

Add all Listeners in one statement completely.

nebula> ADD LISTENER ELASTICSEARCH 192.168.8.5:9789,192.168.8.6:9789;

- 351/1069 - 2023 Vesoft Inc.

## **Show Listeners**

Run the  $\mbox{SHOW LISTENER}$  statement to list all Listeners.

## EXAMPLE



## **Remove Listeners**

 $Run \ the \ {\tt REMOVE \ LISTENER \ ELASTICSEARCH} \ statement \ to \ remove \ all \ Listeners \ in \ a \ graph \ space.$ 

## EXAMPLE

nebula> REMOVE LISTENER ELASTICSEARCH;

## 4.14.4 Full-text indexes

Full-text indexes are used to do prefix, wildcard, regexp, and fuzzy search on a string property.

You can use the WHERE clause to specify the search strings in LOOKUP statements.

#### Prerequisite

Before using the full-text index, make sure that you have deployed a Elasticsearch cluster and a Listener cluster. For more information, see Deploy Elasticsearch and Deploy Listener.

#### Precaution

Before using the full-text index, make sure that you know the restrictions.

#### Natural language full-text search

A natural language search interprets the search string as a phrase in natural human language. The search is case-sensitive and by default prefixes the string with a match. For example, there are three vertices with the tag player. The tag player contains the property name. The name of these three vertices are Kevin Durant, Tim Duncan, and David Beckham. Now that the full-text index of player.name is established, only David Beckham will be queried when using the prefix search statement LOOKUP ON player WHERE PREFIX(player.name, "D");

## Syntax

**CREATE FULL-TEXT INDEXES** 

```
CREATE FULLTEXT {TAG | EDGE} INDEX <index_name> ON {<tag_name> | <edge_name>} ([<prop_name>]);
```

SHOW FULL-TEXT INDEXES

SHOW FULLTEXT INDEXES;

REBUILD FULL-TEXT INDEXES

REBUILD FULLTEXT INDEX;



When there is a large amount of data, rebuilding full-text index is slow, you can modify <code>snapshot\_send\_files=false</code> in the configuration file of Storage <code>service(nebula-storaged.conf)</code>.

DROP FULL-TEXT INDEXES

```
DROP FULLTEXT INDEX <index_name>;
```

USE QUERY OPTIONS

```
LOOKUP ON {<tag> | <edge_type>} WHERE <expression> [YIELD <return_list>];

<expression> ::=
PREFIX | WILDCARD | REGEXP | FUZZY
```

- 353/1069 - 2023 Vesoft Inc.

```
<return_list>
<prop_name> [AS <prop_alias>] [, <prop_name> [AS <prop_alias>] ...]
```

- PREFIX(schema name.prop name, prefix string, row limit, timeout)
- WILDCARD(schema\_name.prop\_name, wildcard\_string, row\_limit, timeout)
- REGEXP(schema\_name.prop\_name, regexp\_string, row\_limit, timeout)
- FUZZY(schema\_name.prop\_name, fuzzy\_string, fuzziness, operator, row\_limit, timeout)
- fuzziness (optional): Maximum edit distance allowed for matching. The default value is AUTO. For other valid values and more information, see Elasticsearch document.
- · operator (optional): Boolean logic used to interpret the text. Valid values are OR (default) and AND.
- row\_timit (optional): Specifies the number of rows to return. The default value is 100.
- $\bullet$  timeout (optional): Specifies the timeout time. The default value is 200 ms .

#### **Examples**

```
// This example creates the graph space
nebula> CREATE SPACE IF NOT EXISTS basketballplayer (partition_num=3,replica_factor=1, vid_type=fixed_string(30));
// This example signs in the text service.
nebula> SIGN IN TEXT SERVICE (127.0.0.1:9200, HTTP);
// This example checks the text service status.
nebula> SHOW TEXT SEARCH CLIENTS:
// This example switches the graph space.
nebula> USE basketballplaver:
// This example adds the listener to the NebulaGraph cluster.
nebula> ADD LISTENER ELASTICSEARCH 192.168.8.5:9789;
// This example checks the listener status. When the status is 'Online', the listener is ready.
// This example creates the tag.
nebula> CREATE TAG IF NOT EXISTS player(name string, age int);
// This example creates the full-text index. The index name starts with "nebula_".
nebula> CREATE FULLTEXT TAG INDEX nebula_index_1 ON player(name);
// This example rebuilds the full-text index
nebula> REBUILD FULLTEXT INDEX;
// This example shows the full-text index.
nebula> SHOW FULLTEXT INDEXES:
                      | Schema Type | Schema Name | Fields |
| "nebula_index_1" | "Tag"
                                       | "player" | "name"
// This example inserts the test data.
// This example Therts the test data.

"Russell Westbrook": ("Russell Westbrook", 30), \
"Chris Paul": ("Chris Paul", 33), \
"Boris Diaw": ("Boris Diaw", 36), \
"David West": ("David West", 38), \
   "Danny Green": ("Danny Green", 31),\
  "Tim Duncan": ("Tim Duncan", 42),
"James Harden": ("James Harden", 29),
"Tony Parker": ("Tony Parker", 36),
"Aron Baynes": ("Aron Baynes", 32),
"Ben Simmons": ("Ben Simmons", 22),
  "Blake Griffin": ("Blake Griffin", 30);
// These examples run test queries.
nebula> LOOKUP ON player WHERE PREFIX(player.name, "B") YIELD id(vertex);
| id(VERTEX)
  "Boris Diaw"
  "Ben Simmons"
  "Blake Griffin"
nebula> LOOKUP ON player WHERE WILDCARD(player.name, "*ri*") YIELD player.name, player.age;
| name
                     age
```

## 4.15 Subgraph and path

## 4.15.1 GET SUBGRAPH

The GET SUBGRAPH statement returns a subgraph that is generated by traversing a graph starting from a specified vertex.

GET SUBGRAPH statements allow you to specify the number of steps and the type or direction of edges during the traversal.

#### Syntax

```
GET SUBGRAPH [WITH PROP] [<step_count> {STEP|STEPS}] FROM {<vid>...}

[{IN | OUT | BOTH} <edge_type>, <edge_type>...]

[WHERE <expression> [AND <expression> ...]]

YIELD {[VERTICES AS <vertex_alias>] [,EDGES AS <edge_alias>]};
```

- WITH PROP shows the properties. If not specified, the properties will be hidden.
- step\_count specifies the number of hops from the source vertices and returns the subgraph from 0 to step\_count hops. It must be a non-negative integer. Its default value is 1.
- vid specifies the vertex IDs.
- edge\_type specifies the edge type. You can use IN, OUT, and BOTH to specify the traversal direction of the edge type. The default is BOTH.
- «WHERE clause» specifies the filter conditions for the traversal, which can be used with the boolean operator AND.
- YIELD defines the output that needs to be returned. You can return only vertices or edges. A column alias must be set.



The path type of GET SUBGRAPH is trail. Only vertices can be repeatedly visited in graph traversal. For more information, see Path.

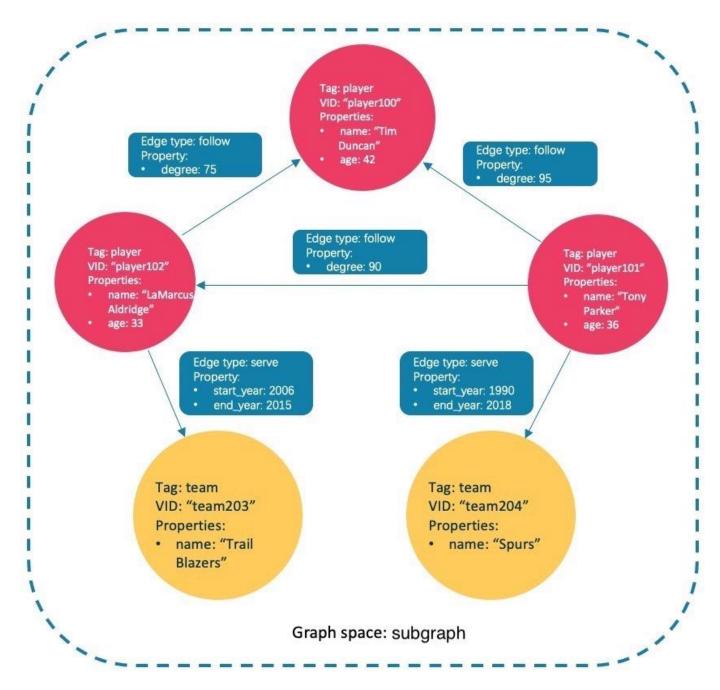
#### Limitations

While using the WHERE clause in a GET SUBGRAPH statement, note the following restrictions:

- $\bullet$   $Only \ support \ the \ \ \ \ AND \ \ operator.$
- $\bullet \ \, \textbf{Only support} \ \, \text{filter destination vertex, the vertex format must be } \$\$. \texttt{tagName.propName} \, .$
- $\bullet$  Support filter edge, the edge format must be  $\mbox{\tt edge\_type.propName}$  .
- Support math functions, aggregate functions, string functions, datetime functions, type conversion functions and general
  functions in list functions.
- Not support aggregate functions, schema-related functions, conditional expression, predicate functions, geography function and user-defined functions.

#### **Examples**

The following graph is used as the sample.



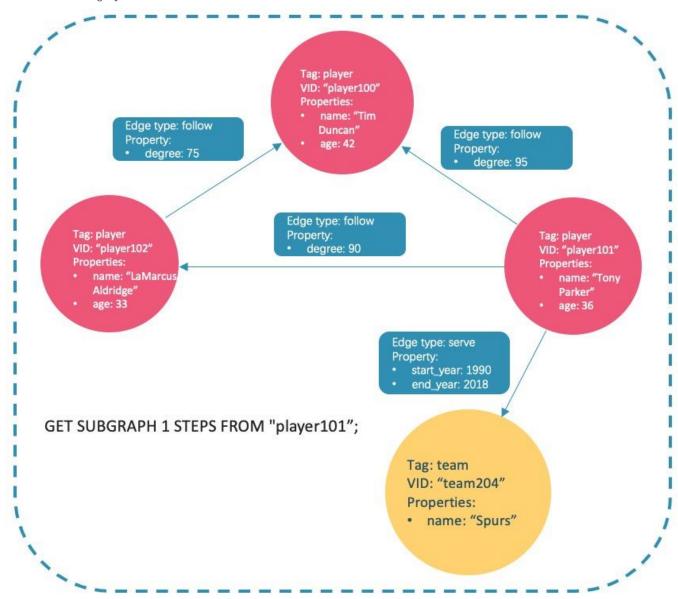
## Insert the test data:

```
nebula> CREATE SPACE IF NOT EXISTS subgraph(partition_num=15, replica_factor=1, vid_type=fixed_string(30));
nebula> CREATE TAG IF NOT EXISTS player(name string, age int);
nebula> CREATE TAG IF NOT EXISTS team(name string);
nebula> CREATE TAG IF NOT EXISTS follow(degree int);
nebula> CREATE EDGE IF NOT EXISTS follow(degree int);
nebula> CREATE EDGE IF NOT EXISTS serve(start_year int);
nebula> INSERT VERTEX player(name, age) VALUES "player100":("Tim Duncan", 42);
nebula> INSERT VERTEX player(name, age) VALUES "player101":("Tony Parker", 36);
nebula> INSERT VERTEX player(name, age) VALUES "player102":("LaMarcus Aldridge", 33);
nebula> INSERT VERTEX team(name) VALUES "ream203":("Trail Blazers"), "team204":("Spurs");
nebula> INSERT EDGE follow(degree) VALUES "player101" -> "player100":(95);
nebula> INSERT EDGE follow(degree) VALUES "player101" -> "player102":(90);
```

```
nebula> INSERT EDGE follow(degree) VALUES "player102" -> "player100":(75);
nebula> INSERT EDGE serve(start_year, end_year) VALUES "player101" -> "team204":(1999, 2018),"player102" -> "team203":(2006, 2015);
```

• This example goes one step from the vertex player101 over all edge types and gets the subgraph.

The returned subgraph is as follows.

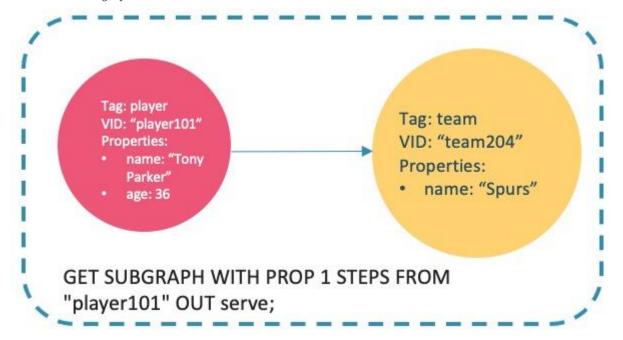


• This example goes one step from the vertex player101 over incoming follow edges and gets the subgraph.

There is no incoming follow edge to player101, so only the vertex player101 is returned.

• This example goes one step from the vertex player101 over outgoing serve edges, gets the subgraph, and shows the property of the edge.

The returned subgraph is as follows.

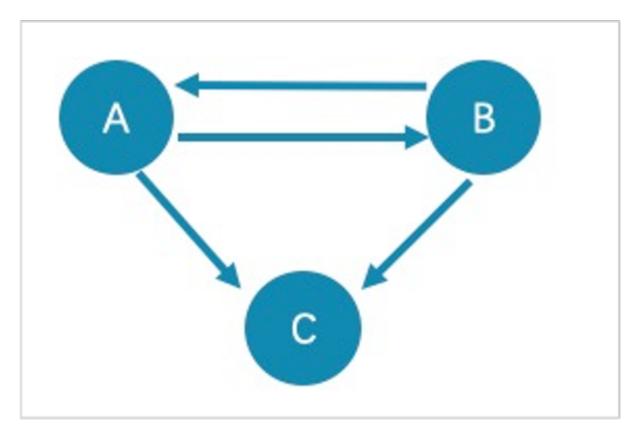


• This example goes two steps from the vertex player101 over follow edges, filters by degree > 90 and age > 30, and shows the properties of edges.

## FAQ

WHY IS THE NUMBER OF HOPS IN THE RETURNED RESULT GREATER THAN STEP\_COUNT?

To show the completeness of the subgraph, an additional hop is made on all vertices that meet the conditions. The following graph is used as the sample.



- The returned paths of GET SUBGRAPH 1 STEPS FROM "A"; are A->B, B->A, and A->C. To show the completeness of the subgraph, an additional hop is made on all vertices that meet the conditions, namely B->C.
- The returned path of GET SUBGRAPH 1 STEPS FROM "A" IN follow; is B->A. To show the completeness of the subgraph, an additional hop is made on all vertices that meet the conditions, namely A->B.

If you only query paths or vertices that meet the conditions, we suggest you use MATCH or GO. The example is as follows.

```
nebula> MATCH p= (v:player) -- (v2) WHERE id(v)=="A" RETURN p;
nebula> GO 1 STEPS FROM "A" OVER follow YIELD src(edge),dst(edge);
```

WHY IS THE NUMBER OF HOPS IN THE RETURNED RESULT LOWER THAN STEP\_COUNT?

The query stops when there is not enough subgraph data and will not return the null value.



### 4.15.2 FIND PATH

The FIND PATH statement finds the paths between the selected source vertices and destination vertices.



To improve the query performance with the FIND PATH statement, you can add the num\_operator\_threads parameter in the nebula-graphd.conf configuration file. The value range of the num\_operator\_threads parameter is [2, 10] and make sure that the value is not greater than the number of CPU cores of the machine where the graphd service is deployed. It is recommended to set the value to the number of CPU cores of the machine where the graphd service is deployed. For more information about the nebula-graphd.conf configuration file, see nebula-graphd.conf.

## Syntax

```
FIND { SHORTEST | ALL | NOLOOP } PATH [WITH PROP] FROM <vertex_id_list> TO <vertex_id_list>

OVER <edge_type_list> [REVERSELY | BIDIRECT]

[<\MHERE clause>] [UPTO <\mi> {STEP|STEPS}]

YIELD path as <alias>
[| ORDER BY $-.path] [| LIMIT <\mi>];

<vertex_id_list> ::=
[vertex_id_list> ::=
[vertex_id_n, vertex_id] ...]
```

- SHORTEST finds the shortest path.
- · ALL finds all the paths.
- NOLOOP finds the paths without circles.
- WITH PROP shows properties of vertices and edges. If not specified, properties will be hidden.
- <vertex\_id\_list> is a list of vertex IDs separated with commas (,). It supports \$- and \$var.
- <edge\_type\_list> is a list of edge types separated with commas (,). \* is all edge types.
- REVERSELY | BIDIRECT specifies the direction. REVERSELY is reverse graph traversal while BIDIRECT is bidirectional graph traversal.
- <WHERE clause> filters properties of edges.
- <N> is the maximum hop number of the path. The default value is 5.
- ullet specifies the maximum number of rows to return.



The path type of FIND PATH is trail. Only vertices can be repeatedly visited in graph traversal. For more information, see Path.

## Limitations

- When a list of source and/or destination vertex IDs are specified, the paths between any source vertices and the destination vertices will be returned.
- There can be cycles when searching all paths.
- FIND PATH only supports filtering properties of edges with WHERE clauses. Filtering properties of vertices and functions are not supported for now.
- FIND PATH is a single-thread procedure, so it uses much memory.

### **Examples**

A returned path is like (<vertex\_id>)-[:<edge\_type\_name>@<rank>]->(<vertex\_id) .

#### FAQ

DOES IT SUPPORT THE WHERE CLAUSE TO ACHIEVE CONDITIONAL FILTERING DURING GRAPH TRAVERSAL?

FIND PATH only supports filtering properties of edges with WHERE clauses, such as WHERE follow.degree is EMPTY or follow.degree >=0.

Filtering properties of vertices is not supported for now.

Last update: February 19, 2024

# 4.16 Query tuning and terminating statements

### 4.16.1 EXPLAIN and PROFILE

EXPLAIN helps output the execution plan of an nGQL statement without executing the statement.

PROFILE executes the statement, then outputs the execution plan as well as the execution profile. You can optimize the queries for better performance according to the execution plan and profile.

#### **Execution Plan**

The execution plan is determined by the execution planner in the NebulaGraph query engine.

The execution planner processes the parsed nGQL statements into actions. An action is the smallest unit that can be executed. A typical action fetches all neighbors of a given vertex, gets the properties of an edge, and filters vertices or edges based on the given conditions. Each action is assigned to an operator that performs the action.

For example, a SHOW TAGS statement is processed into two actions and assigned to a Start operator and a ShowTags operator, while a more complex GO statement may be processed into more than 10 actions and assigned to 10 operators.

#### **Syntax**

EXPLAIN

```
EXPLAIN [format= {"row" | "dot"}] <your_n6QL_statement>;
```

PROFILE

```
PROFILE [format= {"row" | "dot"}] <your_nGQL_statement>;
```

#### **Output formats**

The output of an EXPLAIN or a PROFILE statement has two formats, the default row format and the dot format. You can use the format option to modify the output format. Omitting the format option indicates using the default row format.

- 363/1069 - 2023 Vesoft Inc.

#### The row format

The row format outputs the return message in a table as follows.

#### • EXPLAIN

#### PROFILE

nebula> PROFILE form	ma+="row" CUN	II TACS.		
++	mat- row snor	w 1A03,		
Name				
++				
player				
++				
team				
++				
Got 2 rows (time spe	ent 2038/2728	us)		
Execution Plan				
execution Ptan				
+	+		<del>-</del>	
id   name   de	ependencies	profiling data	operator info	
·÷	÷		· 	
1   ShowTags   0		ver: 0, rows: 1, execTime: 42us, totalTime: 1177us	outputVar: [{"colNames":[],"name":"ShowTags_1","type":"DATASET"}]	
	- 1		inputVar:	
+			+	
0 Start		ver: 0, rows: 0, execTime: 1us, totalTime: 57us	outputVar: [{"colNames":[],"name":"Start_0","type":"DATASET"}]	
+	+		·	

The descriptions are as follows.

Parameter	Description
id	The ID of the operator .
name	The name of the operator.
dependencies	The ID of the operator that the current operator depends on.
profiling data	The content of the execution profile. ver is the version of the operator. rows shows the number of rows to be output by the operator. execTime shows the execution time of action. totalTime is the sum of the execution time, the system scheduling time, and the queueing time.
operator info	The detailed information of the operator.

#### The dot format

You can use the format="dot" option to output the return message in the dot language, and then use Graphviz to generate a graph of the plan.

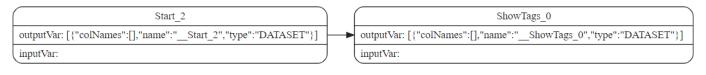


Graphviz is open source graph visualization software. Graphviz provides an online tool for previewing DOT language files and exporting them to other formats such as SVG or JSON. For more information, see Graphviz Online.

```
nebula> EXPLAIN format="dot" SHOW TAGS;
Execution succeeded (time spent 161/665 us)
Execution Plan
```

```
digraph exec_plan {
    rankdir=LR;
    "ShowTags_0"[label="ShowTags_0|outputVar: \[\{\"colNames\":\\[\],\"name\":\"__ShowTags_0\",\"type\":\"DATASET\"\}\]\\|inputVar:\\l", shape=Mrecord];
    "Start_2".>"ShowTags_0";
    "Start_2"[label="Start_2|outputVar: \[\{\"colNames\":\\[\],\"name\":\"__Start_2\",\"type\":\"DATASET\"\}\]\\\|inputVar:\\\", shape=Mrecord];
}
```

The Graphviz graph transformed from the above DOT statement is as follows.



Last update: February 19, 2024

# 4.16.2 Kill queries

KILL QUERY can terminate the query being executed, and is often used to terminate slow queries.



Users with the God role can kill any query. Other roles can only kill their own queries.

### **Syntax**

KILL QUERY (session=<session\_id>, plan=<plan\_id>);

- session\_id: The ID of the session.
- plan\_id: The ID of the execution plan.

The ID of the session and the ID of the execution plan can uniquely determine a query. Both can be obtained through the SHOW QUERIES statement.

# Examples

This example executes KILL QUERY in one session to terminate the query in another session.

nebula> KILL QUERY(SESSION=1625553545984255,PLAN=163);

The query will be terminated and the following information will be returned.

[ERROR (-1005)]: ExecutionPlanId[1001] does not exist in current Session.

Last update: February 19, 2024

- 366/1069 - 2023 Vesoft Inc.

# 4.16.3 Kill sessions

The KILL SESSION command is to terminate running sessions.



- Only the NebulaGraph root user can terminate sessions.
- After executing the KILL SESSION command, all Graph services synchronize the latest session information after 2\* session\_reclaim\_interval\_secs seconds (120 seconds by default).

#### **Syntax**

You can run the KILL SESSION command to terminate one or multiple sessions. The syntax is as follows:

· To terminate one session

KILL {SESSION|SESSIONS} <SessionId>

- {SESSION|SESSIONS}: SESSION or SESSIONS, both are supported.
- <sessionId>: Specifies the ID of one session. You can run the SHOW SESSIONS command to view the IDs of sessions.
- To terminate multiple sessions

```
SHOW SESSIONS
| YIELD $-.SessionId AS sid [WHERE <filter_clause>]
| KILL {SESSION|SESSIONS} $-.sid
```



The KILL SESSION command supports the pipeline operation, combining the SHOW SESSIONS command with the KILL SESSION command to terminate multiple sessions.

- [WHERE <filter\_clause>] :
- Optional, the WHERE clause is used to filter sessions. <filter\_expression> specifies a session filtering expression, for example, WHERE \$-.CreateTime < datetime("2022-12-14T18:00:00"). If the WHERE clause is not specified, all sessions are terminated.
- Filtering conditions in a WHERE clause include: SessionId, UserName, SpaceName, CreateTime, UpdateTime, GraphAddr, Timezone, and ClientIp. You can run the SHOW SESSIONS command to view descriptions of these conditions.
- {SESSION|SESSIONS} : SESSION or SESSIONS, both are supported.



Please use filtering conditions with caution to avoid deleting sessions by mistake.

#### **Examples**

• To terminate one session

nebula> KILL SESSION 1672887983842984

- To terminate multiple sessions
- Terminate all sessions whose creation time is less than 2023-01-05T18:00:00.

```
nebula> SHOW SESSIONS | YIELD $-.SessionId AS sid WHERE $-.CreateTime < datetime("2023-01-05T18:00:00") | KILL SESSIONS $-.sid
```

• Terminates the two sessions with the earliest creation times.

```
nebula> SHOW SESSIONS | YIELD $-.SessionId AS sid, $-.CreateTime as CreateTime | ORDER BY $-.CreateTime ASC | LIMIT 2 | KILL SESSIONS $-.sid
```

• Terminates all sessions created by the username session\_user1.

```
nebula> SHOW SESSIONS | YIELD $-.SessionId as sid WHERE $-.UserName == "session_user1" | KILL SESSIONS $-.sid
```

• Terminate all sessions.

```
nebula> SHOW SESSIONS | YIELD $-.SessionId as sid | KILL SESSION $-.sid

// Or
nebula> SHOW SESSIONS | KILL SESSIONS $-.SessionId
```



When you terminate all sessions, the current session is terminated. Please use it with caution.

Last update: February 19, 2024

- 368/1069 - 2023 Vesoft Inc.

# 4.17 Job manager and the JOB statements

The long-term tasks run by the Storage Service are called jobs, such as COMPACT, FLUSH, and STATS. These jobs can be time-consuming if the data amount in the graph space is large. The job manager helps you run, show, stop, and recover jobs.



All job management commands can be executed only after selecting a graph space.

# 4.17.1 SUBMIT JOB BALANCE DATA



Only available for the NebulaGraph Enterprise Edition.



- Before performing the job, it is recommended to create a snapshot.
- During job execution, do not execute other jobs, such as SUBMIT JOB STATS, REBUILD INDEX, etc.
- During job execution, it is recommended not to write or read data in large batches.

The SUBMIT JOB BALANCE DATA statement starts a job to balance the distribution of storage partitions in the current graph space. It returns the job ID.

For example:

# 4.17.2 SUBMIT JOB COMPACT

The SUBMIT JOB COMPACT statement triggers the long-term RocksDB compact operation in the current graph space.

For more information about compact configuration, see Storage Service configuration.

For example:

# 4.17.3 SUBMIT JOB FLUSH

The SUBMIT JOB FLUSH statement writes the RocksDB memfile in the memory to the hard disk in the current graph space.

For example:

```
nebula> SUBMIT JOB FLUSH;
+-----+
| New Job Id |
+----+
```

#### 4.17.4 SUBMIT JOB STATS

The SUBMIT JOB STATS statement starts a job that makes the statistics of the current graph space. Once this job succeeds, you can use the SHOW STATS statement to list the statistics. For more information, see SHOW STATS.



If the data stored in the graph space changes, in order to get the latest statistics, you have to run SUBMIT JOB STATS again.

### For example:

#### 4.17.5 SUBMIT JOB DOWNLOAD/INGEST

The SUBMIT JOB DOWNLOAD HDFS and SUBMIT JOB INGEST commands are used to import the SST file into NebulaGraph. For detail, see Import data from SST files.

The SUBMIT JOB DOWNLOAD HDFS command will download the SST file on the specified HDFS.

The SUBMIT JOB INGEST command will import the downloaded SST file into NebulaGraph.

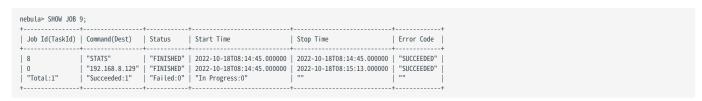
## For example:

#### 4.17.6 SHOW JOB

The Meta Service parses a SUBMIT JOB request into multiple tasks and assigns them to the nebula-storaged processes. The SHOW JOB <job\_id> statement shows the information about a specific job and all its tasks in the current graph space.

job\_id is returned when you run the SUBMIT JOB statement.

#### For example:



The descriptions are as follows.

Parameter	Description
Job Id(TaskId)	The first row shows the job ID and the other rows show the task IDs and the last row shows the total number of job-related tasks.
Command(Dest)	The first row shows the command executed and the other rows show on which storaged processes the task is running. The last row shows the number of successful tasks related to the job.
Status	Shows the status of the job or task. The last row shows the number of failed tasks related to the job. For more information, see Job status.
Start Time	Shows a timestamp indicating the time when the job or task enters the RUNNING phase. The last row shows the number of ongoing tasks related to the job.
Stop Time	Shows a timestamp indicating the time when the job or task gets $$ FINISHED, $$ FAILED, or $$ STOPPED.
Error Code	The error code of job.

### Job status

The descriptions are as follows.

Status	Description
QUEUE	The job or task is waiting in a queue. The Start Time is empty in this phase.
RUNNING	The job or task is running. The Start Time shows the beginning time of this phase.
FINISHED	The job or task is successfully finished. The Stop Time shows the time when the job or task enters this phase.
FAILED	The job or task has failed. The Stop Time shows the time when the job or task enters this phase.
STOPPED	The job or task is stopped without running. The Stop Time shows the time when the job or task enters this phase.
REMOVED	The job or task is removed.

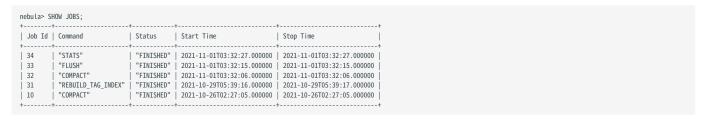
The description of switching the status is described as follows.

# 4.17.7 SHOW JOBS

The  $\mbox{\scriptsize SHOW JOBS}$  statement lists all the unexpired jobs in the current graph space.

The default job expiration interval is one week. You can change it by modifying the <code>job\_expired\_secs</code> parameter of the Meta Service. For how to modify <code>job\_expired\_secs</code>, see Meta Service configuration.

For example:



# 4.17.8 STOP JOB

The STOP JOB <job\_id> statement stops jobs that are not finished in the current graph space.

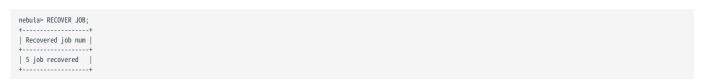
### For example:



# 4.17.9 RECOVER JOB

The RECOVER JOB [<job\_id>] statement re-executes the jobs that status is FAILED or STOPPED in the current graph space and returns the number of recovered jobs. If <job\_id> is not specified, re-execution is performed from the earliest job and the number of jobs that have been recovered is returned.

### For example:



# 4.17.10 FAQ

### How to troubleshoot job problems?

The SUBMIT JOB operations use the HTTP port. Please check if the HTTP ports on the machines where the Storage Service is running are working well. You can use the following command to debug.

curl "http://{storaged-ip}:19779/admin?space={space\_name}&op=compact"

Last update: February 19, 2024

# 5. Deployment and installation

# 5.1 Prepare resources for compiling, installing, and running NebulaGraph

This topic describes the requirements and suggestions for compiling and installing NebulaGraph, as well as how to estimate the resource you need to reserve for running a NebulaGraph cluster.



In addition to installing NebulaGraph with the source code, the Dashboard Enterprise Edition tool is a better and convenient choice for installing Community and Enterprise Edition NebulaGraph. For details, see Deploy Dashboard.

### 5.1.1 About storage devices

NebulaGraph is designed and implemented for NVMe SSD. All default parameters are optimized for the SSD devices and require extremely high IOPS and low latency.

- Due to the poor IOPS capability and long random seek latency, HDD is not recommended. Users may encounter many problems when using HDD.
- Do not use remote storage devices, such as NAS or SAN. Do not connect an external virtual hard disk based on HDFS or Ceph.
- · Do not use RAID.
- Use local SSD devices, or AWS Provisioned IOPS SSD equivalence.

# 5.1.2 About CPU architecture



You can run NebulaGraph Enterprise Edition on ARM, including Apple Mac M1 and Huawei Kunpeng. Contact us for details.



Starting with 3.0.2, you can run containerized NebulaGraph databases on Docker Desktop for ARM macOS or on ARM Linux servers.

## 5.1.3 Requirements for compiling the source code

# Hardware requirements for compiling NebulaGraph

Item	Requirement
CPU architecture	x86_64
Memory	4 GB
Disk	10 GB, SSD

### Supported operating systems for compiling NebulaGraph

For now, we can only compile NebulaGraph in the Linux system. We recommend that you use any Linux system with kernel version 4.15 or above.

- 373/1069 - 2023 Vesoft Inc.



To install NebulaGraph on Linux systems with kernel version lower than required, use RPM/DEB packages or TAR files.

# Software requirements for compiling NebulaGraph

You must have the correct version of the software listed below to compile NebulaGraph. If they are not as required or you are not sure, follow the steps in Prepare software for compiling NebulaGraph to get them ready.

Software	Version	Note
glibc	2.17 or above	You can run lddversion to check the glibc version.
make	Any stable version	-
m4	Any stable version	-
git	Any stable version	-
wget	Any stable version	-
unzip	Any stable version	-
xz	Any stable version	-
readline-devel	Any stable version	-
ncurses-devel	Any stable version	-
zlib-devel	Any stable version	-
g++	8.5.0 or above	You can run gcc -v to check the gcc version.
cmake	3.14.0 or above	You can run cmakeversion to check the cmake version.
curl	Any stable version	-
redhat-lsb-core	Any stable version	-
libstdc++-static	Any stable version	Only needed in CentOS 8+, RedHat 8+, and Fedora systems.
libasan	Any stable version	Only needed in CentOS 8+, RedHat 8+, and Fedora systems.
bzip2	Any stable version	-

Other third-party software will be automatically downloaded and installed to the build directory at the configure (cmake) stage.

#### Prepare software for compiling NebulaGraph

If part of the dependencies are missing or the versions does not meet the requirements, manually install them with the following steps. You can skip unnecessary dependencies or steps according to your needs.

- 1. Install dependencies.
- For CentOS, RedHat, and Fedora users, run the following commands.

```
$ yum update
$ yum install -y make \
                 git \
                 wget
                 unzip \
                 xz \
                 readline-devel \
                 ncurses-devel \
                 zlib-devel \
                 gcc \
                 gcc-c++ \
                 cmake \
                 curl \
                 redhat-lsb-core \
                 bzip2
  // For CentOS 8+, RedHat 8+, and Fedora, install libstdc++-static and libasan as well
$ yum install -y libstdc++-static libasan
```

• For Debian and Ubuntu users, run the following commands.

2. Check if the GCC and cmake on your host are in the right version. See Software requirements for compiling NebulaGraph for the required versions.

```
$ g++ --version
$ cmake --version
```

If your GCC and CMake are in the right versions, then you are all set and you can ignore the subsequent steps. If they are not, select and perform the needed steps as follows.

- 3. If the CMake version is incorrect, visit the CMake official website to install the required version.
- 4. If the G++ version is incorrect, visit the G++ official website or follow the instructions below to to install the required version.
- For CentOS users, run:

```
yum install centos-release-scl
yum install devtoolset-11
scl enable devtoolset-11 'bash'
```

• For Ubuntu users, run:

```
add-apt-repository ppa:ubuntu-toolchain-r/test
apt install gcc-11 g++-11
```

### 5.1.4 Requirements and suggestions for installing NebulaGraph in test environments

# Hardware requirements for test environments

Item	Requirement	
CPU architecture	x86_64	
Number of CPU core	4	
Memory	8 GB	
Disk	100 GB, SSD	

### Supported operating systems for test environments

For now, we can only install NebulaGraph in the Linux system. To install NebulaGraph in a test environment, we recommend that you use any Linux system with kernel version 3.9 or above.

### Suggested service architecture for test environments

Process	Suggested number
metad (the metadata service process)	1
storaged (the storage service process)	1 or more
graphd (the query engine service process)	1 or more

For example, for a single-machine test environment, you can deploy 1 metad, 1 storaged, and 1 graphd processes in the machine.

For a more common test environment, such as a cluster of 3 machines (named as A, B, and C), you can deploy NebulaGraph as follows:

Machine name	Number of metad	Number of storaged	Number of graphd
A	1	1	1
В	None	1	1
С	None	1	1

# 5.1.5 Requirements and suggestions for installing NebulaGraph in production environments

## Hardware requirements for production environments

Item	Requirement
CPU architecture	x86_64
Number of CPU core	48
Memory	256 GB
Disk	2 * 1.6 TB, NVMe SSD

# Supported operating systems for production environments

For now, we can only install NebulaGraph in the Linux system. To install NebulaGraph in a production environment, we recommend that you use any Linux system with kernel version 3.9 or above.

Users can adjust some of the kernel parameters to better accommodate the need for running NebulaGraph. For more information, see kernel configuration.

- 376/1069 - 2023 Vesoft Inc.

# Suggested service architecture for production environments



 $\textbf{DO NOT} \ \text{deploy a single cluster across IDCs} \ (\text{The Enterprise Edtion supports data synchronization between clusters across IDCs}).$ 

Process	Suggested number
metad (the metadata service process)	3
storaged (the storage service process)	3 or more
graphd (the query engine service process)	3 or more

Each metad process automatically creates and maintains a replica of the metadata. Usually, you need to deploy three metad processes and only three.

The number of storaged processes does not affect the number of graph space replicas.

Users can deploy multiple processes on a single machine. For example, on a cluster of 5 machines (named as A, B, C, D, and E), you can deploy NebulaGraph as follows:

Machine name	Number of metad	Number of storaged	Number of graphd
A	1	1	1
В	1	1	1
С	1	1	1
D	None	1	1
E	None	1	1

- 377/1069 - 2023 Vesoft Inc.

# 5.1.6 Capacity requirements for running a NebulaGraph cluster

Users can estimate the memory, disk space, and partition number needed for a NebulaGraph cluster of 3 replicas as follows.

Resource	Unit	How to estimate	Description
Disk space for a cluster	Bytes	the_sum_of_edge_number_and_vertex_number * average_bytes_of_properties * 7.5 * 120%	For more information, see Edge partitioning and storage amplification.
Memory for a cluster	Bytes	[the_sum_of_edge_number_and_vertex_number * 16 + the_number_of_RocksDB_instances * (write_buffer_size * max_write_buffer_number) + rocksdb_block_cache] * 120%	write_buffer_size and max_write_buffer_number are RocksDB parameters. For more information, see MemTable. For details about rocksdb_block_cache, see Memory usage in RocksDB.
Number of partitions for a graph space	-	<pre>the_number_of_disks_in_the_cluster * disk_partition_num_multiplier</pre>	disk_partition_num_multiplier is an integer between 2 and 20 (both including). Its value depends on the disk performance. Use 20 for SSD and 2 for HDD.

• Question 1: Why do I need to multiply by 7.5 in the disk space estimation formula?

Answer: On one hand, the data in one single replica takes up about 2.5 times more space than that of the original data file (csv) according to test values. On the other hand, indexes take up additional space. Each indexed vertex or edge takes up 16 bytes of memory. The hard disk space occupied by the index can be empirically estimated as the total number of indexed vertices or edges \* 50 bytes.

• Question 2: Why do we multiply the disk space and memory by 120%?

Answer: The extra 20% is for buffer.

• Question 3: How to get the number of RocksDB instances?

Answer: Each graph space corresponds to one RocksDB instance and each directory in the --data\_path item in the etc/nebula-storaged.conf file corresponds to one RocksDB instance. That is, the number of RocksDB instances = the number of directories \* the number of graph spaces.



Users can decrease the memory size occupied by the bloom filter by adding  $--enable\_partitioned\_index\_filter=true$  in etc/nebula-storaged.conf. But it may decrease the read performance in some random-seek cases.



Each RocksDB instance takes up about 70M of disk space even when no data has been written yet. One partition corresponds to one RocksDB instance, and when the partition setting is very large, for example, 100, the graph space takes up a lot of disk space after it is created.

Last update: February 19, 2024

- 378/1069 - 2023 Vesoft Inc.

# 5.2 Compile and install Nebula Graph

### 5.2.1 Install NebulaGraph by compiling the source code

Installing NebulaGraph from the source code allows you to customize the compiling and installation settings and test the latest features.

## Prerequisites

• Users have to prepare correct resources described in Prepare resources for compiling, installing, and running NebulaGraph.



Compilation of NebulaGraph offline is not currently supported.

• The host to be installed with NebulaGraph has access to the Internet.

#### Installation steps

- 1. Use Git to clone the source code of NebulaGraph to the host.
- [Recommended] To install NebulaGraph 3.4.1, run the following command.

```
\ git clone --branch release-3.4 https://github.com/vesoft-inc/nebula.git
```

• To install the latest developing release, run the following command to clone the source code from the master branch.

```
$ git clone https://github.com/vesoft-inc/nebula.git
```

2. Go to the nebula/third-party directory, and run the install-third-party.sh script to install the third-party libraries.

```
$ cd nebula/third-party
$ ./install-third-party.sh
```

3. Go back to the nebula directory, create a directory named build, and enter the directory.

```
$ cd ..
$ mkdir build && cd build
```

4. Generate Makefile with CMake.



For more information about CMake variables, see CMake variables.

```
$ cmake -DCMAKE_INSTALL_PREFIX=/usr/local/nebula -DENABLE_TESTING=OFF -DCMAKE_BUILD_TYPE=Release .
```

5. Compile NebulaGraph.



Check Prepare resources for compiling, installing, and running NebulaGraph.

- 379/1069 - 2023 Vesoft Inc.

To speed up the compiling, use the -j option to set a concurrent number N. It should be  $\mbox{\rm N}$ . It should be  $\mbox{\rm CPU}$  core number,  $\mbox{\rm M}$ . It should be  $\mbox{\rm N}$ .

```
$ make -j{N} # E.g., make -j2
```

6. Install NebulaGraph.

```
$ sudo make install
```

7. The configuration files in the etc/ directory (/usr/local/nebula/etc by default) are references. Users can create their own configuration files accordingly. If you want to use the scripts in the script directory to start, stop, restart, and kill the service, and check the service status, the configuration files have to be named as nebula-graph.conf, nebula-metad.conf, and nebula-storaged.conf.

#### Update the master branch

The source code of the master branch changes frequently. If the corresponding NebulaGraph release is installed, update it in the following steps.

- 1. In the nebula directory, run git pull upstream master to update the source code.
- 2. In the nebula/build directory, run make  $-j\{N\}$  and make install again.

#### Next to do

- (Enterprise Edition)Deploy license
- Manage NebulaGraph services

# CMake variables

USAGE OF CMAKE VARIABLES

```
$ cmake -D<variable>=<value> ...
```

The following CMake variables can be used at the configure (cmake) stage to adjust the compiling settings.

CMAKE\_INSTALL\_PREFIX

CMAKE\_INSTALL\_PREFIX specifies the path where the service modules, scripts, configuration files are installed. The default path is /usr/local/nebula.

ENABLE WERROR

ENABLE\_WERROR is ON by default and it makes all warnings into errors. You can set it to OFF if needed.

ENABLE\_TESTING

ENABLE\_TESTING is ON by default and unit tests are built with the NebulaGraph services. If you just need the service modules, set it to OFF.

ENABLE\_ASAN

ENABLE\_ASAN is OFF by default and the building of ASan (AddressSanitizer), a memory error detector, is disabled. To enable it, set ENABLE\_ASAN to ON. This variable is intended for NebulaGraph developers.

- 380/1069 - 2023 Vesoft Inc.

CMAKE\_BUILD\_TYPE

NebulaGraph supports the following building types of MAKE\_BUILD\_TYPE:

Debug

The default value of CMAKE\_BUILD\_TYPE. It indicates building NebulaGraph with the debug info but not the optimization options.

Release

It indicates building NebulaGraph with the optimization options but not the debug info.

• RelWithDebInfo

It indicates building NebulaGraph with the optimization options and the debug info.

MinSizeRel

It indicates building NebulaGraph with the optimization options for controlling the code size but not the debug info.

ENABLE INCLUDE WHAT YOU USE

ENABLE\_INCLUDE\_WHAT\_YOU\_USE is OFF by default. When set to ON and include-what-you-use is installed on the system, the system reports redundant headers contained in the project source code during makefile generation.

NEBULA\_USE\_LINKER

Specifies the program linker on the system. The available values are:

- bfd, the default value, indicates that ld.bfd is applied as the linker.
- Illd, indicates that ld.lld, if installed on the system, is applied as the linker.
- gold, indicates that ld.gold, if installed on the system, is applied as the linker.

CMAKE\_C\_COMPILER/CMAKE\_CXX\_COMPILER

Usually, CMake locates and uses a C/C++ compiler installed in the host automatically. But if your compiler is not installed at the standard path, or if you want to use a different one, run the command as follows to specify the installation path of the target compiler:

```
$ cmake -DCMAKE_C_COMPILER=<path_to_gcc/bin/gcc> -DCMAKE_CXX_COMPILER=<path_to_gcc/bin/g++> ..
$ cmake -DCMAKE_C_COMPILER=<path_to_clang/bin/clang> -DCMAKE_CXX_COMPILER=<path_to_clang/bin/clang++> ..
```

ENABLE\_CCACHE

ENABLE\_CCACHE is ON by default and Ccache (compiler cache) is used to speed up the compiling of NebulaGraph.

To disable ccache, setting ENABLE\_CCACHE to OFF is not enough. On some platforms, the ccache installation hooks up or precedes the compiler. In such a case, you have to set an environment variable export CCACHE\_DISABLE=true or add a line disable=true in ~/.ccache/ccache.conf as well. For more information, see the ccache official documentation.

NEBULA\_THIRDPARTY\_ROOT

NEBULA\_THIRDPARTY\_ROOT specifies the path where the third party software is installed. By default it is /opt/vesoft/third-party.

# Examine problems

If the compiling fails, we suggest you:

- 1. Check whether the operating system release meets the requirements and whether the memory and hard disk space are sufficient.
- 2. Check whether the third-party is installed correctly.
- 3. Use make -j1 to reduce the compiling concurrency.

Last update: February 19, 2024

# 5.2.2 Install NebulaGraph with RPM or DEB package

RPM and DEB are common package formats on Linux systems. This topic shows how to quickly install NebulaGraph with the RPM or DEB package.



The console is not complied or packaged with NebulaGraph server binaries. You can install nebula-console by yourself.



For the Enterprise Edition, please contact us.

# Prerequisites

Wget installed.

### Download the package from cloud service



NebulaGraph is currently only supported for installation on Linux systems, and only CentOS 7.x, CentOS 8.x, Ubuntu 16.04, Ubuntu 18.04, and Ubuntu 20.04 operating systems are supported.

· Download the released version.

#### URL:

```
//Centos 7
https://oss-cdn.nebula-graph.io/package/<release_version>/nebula-graph-<release_version>.el6.x86_64.rpm

//Centos 7
https://oss-cdn.nebula-graph.io/package/<release_version>/nebula-graph-<release_version>.el7.x86_64.rpm

//Centos 8
https://oss-cdn.nebula-graph.io/package/<release_version>/nebula-graph-<release_version>.el8.x86_64.rpm

//Ubuntu 1604
https://oss-cdn.nebula-graph.io/package/<release_version>/nebula-graph-<release_version>.ubuntu1604.amd64.deb

//Ubuntu 1804
https://oss-cdn.nebula-graph.io/package/<release_version>/nebula-graph-<release_version>.ubuntu1804.amd64.deb

//Ubuntu 2004
https://oss-cdn.nebula-graph.io/package/<release_version>/nebula-graph-<release_version>.ubuntu2004.amd64.deb
```

#### For example, download the release package 3.4.1 for Centos 7.5:

```
wget https://oss-cdn.nebula-graph.io/package/3.4.1/nebula-graph-3.4.1.el7.x86_64.rpm
wget https://oss-cdn.nebula-graph.io/package/3.4.1/nebula-graph-3.4.1.el7.x86_64.rpm.sha256sum.txt
```

## Download the release package 3.4.1 for Ubuntu 1804:

```
wget https://oss-cdn.nebula-graph.io/package/3.4.1/nebula-graph-3.4.1.ubuntu1804.amd64.deb
wget https://oss-cdn.nebula-graph.io/package/3.4.1/nebula-graph-3.4.1.ubuntu1804.amd64.deb.sha256sum.txt
```

• Download the nightly version.



- Nightly versions are usually used to test new features. Do not use it in a production environment.
- Nightly versions may not be built successfully every night. And the names may change from day to day.

#### URL:

```
//Centos 7
https://oss-cdn.nebula-graph.io/package/nightly/<yyyy.mm.dd>/nebula-graph-<yyyy.mm.dd>-nightly.el6.x86_64.rpm

//Centos 7
https://oss-cdn.nebula-graph.io/package/nightly/<yyyy.mm.dd>/nebula-graph-<yyyy.mm.dd>-nightly.el7.x86_64.rpm

//Centos 8
https://oss-cdn.nebula-graph.io/package/nightly/<yyyyy.mm.dd>/nebula-graph-<yyyy.mm.dd>-nightly.el8.x86_64.rpm

//Ubuntu 1604
https://oss-cdn.nebula-graph.io/package/nightly/<yyyyy.mm.dd>/nebula-graph-<yyyy.mm.dd>-nightly.ubuntu1604.amd64.deb

//Ubuntu 1804
https://oss-cdn.nebula-graph.io/package/nightly/<yyyyy.mm.dd>/nebula-graph-<yyyy.mm.dd>-nightly.ubuntu1804.amd64.deb

//Ubuntu 2004
https://oss-cdn.nebula-graph.io/package/nightly/<yyyyy.mm.dd>/nebula-graph-<yyyy.mm.dd>-nightly.ubuntu1804.amd64.deb
```

For example, download the Centos 7.5 package developed and built in 2021.11.28:

```
wget https://oss-cdn.nebula-graph.io/package/nightly/2021.11.28/nebula-graph-2021.11.28-nightly.el7.x86_64.rpm wget https://oss-cdn.nebula-graph.io/package/nightly/2021.11.28/nebula-graph-2021.11.28-nightly.el7.x86_64.rpm.sha256sum.txt
```

For example, download the Ubuntu 1804 package developed and built in 2021.11.28:

```
wget https://oss-cdn.nebula-graph.io/package/nightly/2021.11.28/nebula-graph-2021.11.28-nightly.ubuntu1804.amd64.deb
wget https://oss-cdn.nebula-graph.io/package/nightly/2021.11.28/nebula-graph-2021.11.28-nightly.ubuntu1804.amd64.deb.sha256sum.txt
```

# Install NebulaGraph

• Use the following syntax to install with an RPM package.

```
$ sudo rpm -ivh --prefix=<installation_path> <package_name>
```

The option  $\operatorname{--prefix}$  indicates the installation path. The default path is  $\operatorname{/usr/local/nebula/}$ .

For example, to install an RPM package in the default path for the 3.4.1 version, run the following command.

sudo rpm -ivh nebula-graph-3.4.1.el7.x86\_64.rpm

• Use the following syntax to install with a DEB package.

\$ sudo dpkg -i <package\_name>



Customizing the installation path is not supported when installing NebulaGraph with a DEB package. The default installation path is  $\frac{\sqrt{\sqrt{\frac{1}{2}}}}{\sqrt{\frac{1}{2}}}$ 

For example, to install a DEB package for the 3.4.1 version, run the following command.

sudo dpkg -i nebula-graph-3.4.1.ubuntu1804.amd64.deb



The default installation path is  $\mbox{\it /usr/local/nebula/}$  .

### Next to do

- (Enterprise Edition)Deploy license
- Start NebulaGraph
- Connect to NebulaGraph

Last update: February 19, 2024

# 5.2.3 Install NebulaGraph graph with the tar.gz file

You can install NebulaGraph by downloading the tar.gz file.



- NebulaGraph provides installing with the tar.gz file starting from version 2.6.0.
- NebulaGraph is currently only supported for installation on Linux systems, and only CentOS 7.x, CentOS 8.x, Ubuntu 16.04, Ubuntu 18.04, and Ubuntu 20.04 operating systems are supported.

### Installation steps

1. Download the NebulaGraph tar.gz file using the following address.

Before downloading, you need to replace <release\_version> with the version you want to download.

```
//Centos 7
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.el7.x86_64.tar.gz
//checksum
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.el7.x86_64.tar.gz.sha256sum.txt

//Centos 8
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.el8.x86_64.tar.gz
//checksum
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.el8.x86_64.tar.gz.sha256sum.txt

//Ubuntu 1604
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.ubuntu1604.amd64.tar.gz
//checksum
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.ubuntu1604.amd64.tar.gz
//buntu 1804
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.ubuntu1804.amd64.tar.gz
//checksum
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.ubuntu1804.amd64.tar.gz
//buntu 1804
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.ubuntu1804.amd64.tar.gz
//buntu 2004
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.ubuntu1804.amd64.tar.gz
//buntu 2004
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.ubuntu1804.amd64.tar.gz
//buntu 2004
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.ubuntu1804.amd64.tar.gz
//buntu 2004
https://oss-cdn.nebula-graph.com.cn/package/<release_version>/nebula-graph-<release_version>.ubuntu2004.amd64.tar.gz
```

For example, to download the NebulaGraph release-3.4 tar.gz file for CentOS 7.5, run the following command:

```
wget https://oss-cdn.nebula-graph.com.cn/package/3.4.1/nebula-graph-3.4.1.el7.x86_64.tar.gz
```

2. Decompress the tar.gz file to the NebulaGraph installation directory.

```
tar -xvzf <tar.gz_file_name> -C <install_path>
```

- $\bullet$  tar.gz\_file\_name specifies the name of the tar.gz file.
- install\_path specifies the installation path.

For example:

```
tar -xvzf nebula-graph-3.4.1.el7.x86_64.tar.gz -C /home/joe/nebula/install
```

3. Modify the name of the configuration file.

Enter the decompressed directory, rename the files nebula-graphd.conf.default, nebula-metad.conf.default, and nebula-storaged.conf.default in the subdirectory etc, and delete .default to apply the default configuration of NebulaGraph. To modify the configuration, see Configurations.

So far, you have installed NebulaGraph successfully.

- 387/1069 - 2023 Vesoft Inc.

# Next to do

- (Enterprise Edition)Deploy license
- Manage NebulaGraph services

Last update: February 19, 2024

# 5.2.4 Deploy NebulaGraph with Docker Compose

Using Docker Compose can quickly deploy NebulaGraph services based on the prepared configuration file. It is only recommended to use this method when testing functions of NebulaGraph.

#### **Prerequisites**

• You have installed the following applications on your host.

Application	Recommended version	d version Official installation reference	
Docker	Latest	Install Docker Engine	
Docker Compose	Latest	Install Docker Compose	
Git	Latest	Download Git	

- If you are deploying NebulaGraph as a non-root user, grant the user with Docker-related privileges. For detailed instructions, see Manage Docker as a non-root user.
- You have started the Docker service on your host.
- If you have already deployed another version of NebulaGraph with Docker Compose on your host, to avoid compatibility issues, you need to delete the nebula-docker-compose/data directory.

#### Deploy NebulaGraph

1. Clone the 3.4.0 branch of the nebula-docker-compose repository to your host with Git.



The master branch contains the untested code for the latest NebulaGraph development release. **DO NOT** use this release in a production environment.

 $\$  git clone -b release-3.4 https://github.com/vesoft-inc/nebula-docker-compose.git



The x.y version of Docker Compose aligns to the x.y version of NebulaGraph. For the NebulaGraph z version, Docker Compose does not publish the corresponding z version, but pulls the z version of the NebulaGraph image.

2. Go to the nebula-docker-compose directory.

```
$ cd nebula-docker-compose/
```

 $3. \ \mbox{Run}$  the following command to start all the Nebula Graph services.



- Update the NebulaGraph images and NebulaGraph Console images first if they are out of date.
- The return result after executing the command varies depending on the installation directory.

```
[nebula-docker-compose]$ docker-compose up -d
Creating nebuladockercompose_metad0_1 ... done
Creating nebuladockercompose_metad1_1 ... done
Creating nebuladockercompose_metad1_1 ... done
Creating nebuladockercompose_graphd2_1 ... done
Creating nebuladockercompose_graphd_1 ... done
```

- 389/1069 - 2023 Vesoft Inc.

```
Creating nebuladockercompose_graphdl_1 ... done
Creating nebuladockercompose_storaged2_1 ... done
Creating nebuladockercompose_storaged2_1 ... done
Creating nebuladockercompose_storaged1_1 ... done
```

# mpatibility

Starting from NebulaGraph version 3.1.0, nebula-docker-compose automatically starts a NebulaGraph Console docker container and adds the storage host to the cluster (i.e. ADD HOSTS command).



For more information of the preceding services, see NebulaGraph architecture.

#### Connect to NebulaGraph

There are two ways to connect to NebulaGraph:

- Connected with Nebula Console outside the container. Because the external mapping port for the Graph service is also fixed as 9669 in the container's configuration file, you can connect directly through the default port. For details, see Connect to NebulaGraph.
- Log into the container installed NebulaGraph Console, then connect to the Graph service. This section describes this approach.
- 1. Run the following command to view the name of NebulaGraph Console docker container.

```
$ docker-compose ps
Name
Command
State
Ports

nebuladockercompose_console_1 sh -c sleep 3 && Up
nebula-co ...
```

2. Run the following command to enter the NebulaGraph Console docker container.

```
docker exec -it nebuladockercompose_console_1 /bin/sh
/ #
```

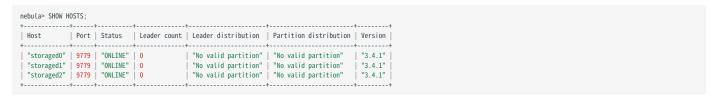
3. Connect to NebulaGraph with NebulaGraph Console.

```
/# ./usr/local/bin/nebula-console -u <user_name> -p <password> --address=graphd --port=9669
```



By default, the authentication is off, you can only log in with an existing username (the default is root) and any password. To turn it on, see Enable authentication.

4. Run the following commands to view the cluster state.



Run exit twice to switch back to your terminal (shell).

#### Check the NebulaGraph service status and ports

Run docker-compose ps to list all the services of NebulaGraph and their status and ports.

- 390/1069 - 2023 Vesoft Inc.



NebulaGraph provides services to the clients through port 9669 by default. To use other ports, modify the docker-compose.yaml file in the nebula-docker-compose directory and restart the NebulaGraph services.

```
$ docker-compose ps
 nebuladockercompose console 1
                                                                                                                                                                                  sh -c sleep 3 &&
                                                                                                                                                                                                                                                                                                                                                               Up
                                                                                                                                                                                  nebula-co
 nebuladockercompose_graphd1_1
                                                                                                                                                                                  /usr/local/nebula/bin/nebu ...
                                                                                                                                                                                                                                                                                                                                                           Up
                                                                                                                                                                                                                                                                                                                                                                                                        {\color{blue}0.0.0.0:49174->19669/tcp, :::49174->19669/tcp, \phantom{0}0.0.0.0:49171->19670/tcp, :::49171->19670/tcp, \phantom{0}0.0.0.0:49177->9669/tcp, \phantom{0}0.0.0.0:49174->19670/tcp, \phantom{0}0.0.0.0:49171->19670/tcp, \phantom{0}0.0.0.0.0:49171->19670/tcp, \phantom{0}0.0.0.0:49171->19670/tcp, \phantom{0}0.0.0.0.0:49171->19670/tcp, \phantom{0}0.0.0.0:49171->19670/tcp, \phantom{0}0.0.0.0.0:49171->19670/tcp, \phantom{0}0.0.0.0.0:49171->19670/tcp, \phantom{0}0.0.0.0
 tcp,:::49177->9669/tcp
 nebuladockercompose_graphd2_1
                                                                                                                                                                                                                                                                                                                                                                                                        0.0.0.0:49175->19669/tcp,:::49175->19669/tcp, 0.0.0.0:49172->19670/tcp,:::49172->19670/tcp, 0.0.0.0:49178->9669/
                                                                                                                                                                                  /usr/local/nebula/bin/nebu ...
 tcn.:::49178->9669/tcp
 nebuladockercompose_graphd_1
                                                                                                                                                                                  /usr/local/nebula/bin/nebu ...
                                                                                                                                                                                                                                                                                                                                                                                                        0.0.0.0:49180->19669/tcp,:::49180->19669/tcp, 0.0.0.0:49179->19670/tcp,:::49179->19670/tcp, 0.0.0.0:9669->9669/
 tcp,:::9669->9669/tcp
 nebuladockercompose metad0 1
                                                                                                                                                                                  /usr/local/nebula/bin/nebu ... Up
                                                                                                                                                                                                                                                                                                                                                                                                        0.0.0.0:49157->19559/tcp,:::49157->19559/tcp, 0.0.0.0:49154->19560/tcp,:::49154->19560/tcp, 0.0.0.0:49160->9559/
 tcp,:::49160->9559/tcp, 9560/tcp
 nebuladockercompose_metad1_1
                                                                                                                                                                                 /usr/local/nebula/bin/nebu ...
                                                                                                                                                                                                                                                                                                                                                                                                        0.0.0.0:49156->19559/tcp,:::49156->19559/tcp, 0.0.0.0:49153->19560/tcp,:::49153->19560/tcp, 0.0.0.0:49159->9559/
 tcp.:::49159->9559/tcp. 9560/tcp
 nebuladockercompose_metad2_1
                                                                                                                                                                                  /usr/local/nebula/bin/nebu ...
                                                                                                                                                                                                                                                                                                                                                                                                        0.0.0.0:49158->19559/tcp,:::49158->19559/tcp, 0.0.0.0:49155->19560/tcp,:::49155->19560/tcp, 0.0.0.0:49161->9559/
 tcp,:::49161->9559/tcp, 9560/tcp
 nebuladockercompose storaged0 1
                                                                                                                                                                                  /usr/local/nebula/bin/nebu ... Up
                                                                                                                                                                                                                                                                                                                                                                                                        0.0.0.0:49166->19779/tcp,:::49166->19779/tcp, 0.0.0.0:49163->19780/tcp,:::49163->19780/tcp, 9777/tcp, 9778/tcp, 0.0.0.0:49163->19780/tcp, 0.0.0.0:49163->19780/tcp, 9778/tcp, 9778/t
 0.0.0:49169->9779/tcp,:::49169->9779/tcp, 9780/tcp
nebuladockercompose_storaged1_1 /usr/local/nebula
0.0.0:49168->9779/tcp,:::49168->9779/tcp, 9780/tcp
                                                                                                                                                                                 /usr/local/nebula/bin/nebu ...
                                                                                                                                                                                                                                                                                                                                                           Un
                                                                                                                                                                                                                                                                                                                                                                                                        0.0.0.0:49165->19779/tcp,:::49165->19779/tcp, 0.0.0.0:49162->19780/tcp,:::49162->19780/tcp, 9777/tcp, 9778/tcp, 0.0.0.0:49162->19780/tcp, 0.0.0.0:49162->19780/tcp, 9777/tcp, 9778/tcp, 0.0.0.0:49162->19780/tcp, 0.0.0.0:49162->19780/tcp, 9778/tcp, 0.0.0.0:49162->19780/tcp, 9778/tcp, 9778/tcp, 0.0.0.0:49162->19780/tcp, 9778/tcp, 9778/tcp, 0.0.0.0:49162->19780/tcp, 9778/tcp, 9778/tcp, 9778/tcp, 0.0.0.0:49162->19780/tcp, 9778/tcp, 9778/tcp, 0.0.0.0.0:49162->19780/tcp, 9778/tcp, 9778/tcp
 nebuladockercompose_storaged2_1 /usr/local/nebula/bin/nebu ...
                                                                                                                                                                                                                                                                                                                                                                                                         0.0.0.0:49167->19779/tcp,:::49167->19779/tcp, 0.0.0.0:49164->19780/tcp,:::49164->19780/tcp, 9777/tcp, 9778/tcp, 0.0.0.0:49164->19780/tcp, 9777/tcp, 9778/tcp, 9778/tcp, 0.0.0.0:49164->19780/tcp, 9777/tcp, 9778/tcp, 0.0.0.0:49164->19780/tcp, 9777/tcp, 9778/tcp, 9778/tcp, 0.0.0.0:49164->19780/tcp, 9778/tcp, 9778/tcp, 0.0.0.0:49164->19780/tcp, 9778/tcp, 
0.0.0:49170->9779/tcp,:::49170->9779/tcp, 9780/tcp
```

If the service is abnormal, you can first confirm the abnormal container name (such as nebuladockercompose\_graphd2\_1).

Then you can execute docker ps to view the corresponding CONTAINER ID (such as 2a6c56c405f5).

```
[nebula-docker-compose]$ docker ps
CONTAINER ID IMAGE
                                                                                                      COMMAND
                                                                                                                                                        CREATED
                                                                                                                                                                                          STATUS
                                                                                                                                                                                                              NAMES
PORTS
2a6c56c405f5 vesoft/nebula-graphd:nightly
                                                                                                                                                                                                                                          0.0.0.0:49230->9669/tcp, 0.0.0.0:49229->19669/tcp, 0.0.0.0:49228->19670/
                                                                                               "/usr/local/nebula/b..."
                                                                                                                                                    36 minutes ago
                                                                                                                                                                                       Up 36 minutes (healthy)
                                                              nebuladockercompose\_graphd2\_1
                                                                                               "./bin/nebula-storag..."
7042e0a8e83d
                             vesoft/nebula-storaged:nightly
                                                                                                                                                                                                                                          9777-9778/tcp, 9780/tcp, 0.0.0.0:49227->9779/tcp, 0.0.0.0:49226->19779/
                                                                                                                                                   36 minutes ago
                                                                                                                                                                                      Up 36 minutes (healthy)
1042evase830 vesoft/nebuta-storaged.nightly "./bin/nebuta-storaged2_1
1Re3ea63ad65 vesoft/nebuta-storaged:nightly "./bin/nebuta-storaged..."
                                                                                                                                                                                                                                          9777-9778/tcp, 9780/tcp, 0.0.0.0:49219->9779/tcp, 0.0.0.0:49218->19779/
                                                                                                                                                   36 minutes ago
                                                                                                                                                                                      Up 36 minutes (healthy)
tcp, 0.0.0.0:49217->19780/tcp nebuladockercompose_storaged0_1
4dcabfe8677a vesoft/nebula-graphd:nightly
                                                                                                "/usr/local/nebula/b..."
                                                                                                                                                   36 minutes ago
                                                                                                                                                                                      Up 36 minutes (healthy)
                                                                                                                                                                                                                                          0.0.0.0:49224->9669/tcp, 0.0.0.0:49223->19669/tcp, 0.0.0.0:49222->19670/
                                                            nebuladockercompose_graphd1_1
tcp
a74054c6ae25 vesoft/nebula-graphd:nightly
                                                                                               "/usr/local/nebula/b..."
                                                                                                                                                    36 minutes ago
                                                                                                                                                                                      Up 36 minutes (healthy)
                                                                                                                                                                                                                                          0.0.0.0:9669->9669/tcp, 0.0.0.0:49221->19669/tcp, 0.0.0.0:49220->19670/
                             nebuladockercompose_graphd_1 vesoft/nebula-storaged:nightly "./bin/nebula-storag..."
880025a3858c
                                                                                                                                                                                                                                          9777-9778/tcp, 9780/tcp, 0.0.0.0:49216->9779/tcp, 0.0.0.0:49215->19779/
                                                                                                                                                    36 minutes ago
                                                                                                                                                                                      Up 36 minutes (healthy)
tcp, 0.0.0.0:49214->19780/tcp nebuladockercompose_storaged1_1
45736a32a23a vesoft/nebula-metad:nightly
                                                                                                 "./bin/nebula-metad ...'
                                                                                                                                                   36 minutes ago
                                                                                                                                                                                      Up 36 minutes (healthy)
                                                                                                                                                                                                                                          9560/tcp, 0.0.0.0:49213->9559/tcp, 0.0.0.0:49212->19559/tcp, 0.0.0.0.0:49212->19559/tcp, 0.0.0.0:49212->19559/tcp, 0.0.0.0
0.0.0:49211->19560/tcp
                                                                               nebuladockercompose_metad0_1
3b2c90eb073e
                             vesoft/nebula-metad:nightly
                                                                                                "./bin/nebula-metad ..."
                                                                                                                                                                                                                                          9560/tcp, 0.0.0.0:49207->9559/tcp, 0.0.0.0:49206->19559/tcp, 0.
                                                                                                                                                    36 minutes ago
                                                                                                                                                                                     Up 36 minutes (healthy)
0.0.0:49205->19560/tcp
                                                                               nebuladockercompose metad2 1
                            vesoft/nebula-metad:nightly
                                                                                                "./bin/nebula-metad ..."
                                                                                                                                                    36 minutes ago
                                                                                                                                                                                      Up 36 minutes (healthy)
                                                                                                                                                                                                                                          0.0.0:49208->19560/tcp
                                                                               nebuladockercompose_metad1_1
```

Use the CONTAINER ID to log in the container and troubleshoot.

```
nebula-docker-compose]$ docker exec -it 2a6c56c405f5 bash [root@2a6c56c405f5 nebula]#
```

#### Check the service data and logs

All the data and logs of NebulaGraph are stored persistently in the nebula-docker-compose/data and nebula-docker-compose/logs directories.

The structure of the directories is as follows:

```
nebula-docker-compose/
   -- docker-compose.yaml
        data
            - meta0
              meta1
              meta2
              storage0
              storage1
              storage2
         Logs
             graph
             graph1
             graph2
             meta0
             meta1
             meta2
```

- 391/1069 - 2023 Vesoft Inc

```
├── storage0
├── storage1
└── storage2
```

#### Stop the NebulaGraph services

You can run the following command to stop the NebulaGraph services:

```
$ docker-compose down
```

The following information indicates you have successfully stopped the NebulaGraph services:

```
Stopping nebuladockercompose console 1
Stopping nebuladockercompose_graphd1_1
                                         ... done
                                          ... done
Stopping nebuladockercompose_graphd_1
Stopping nebuladockercompose graphd2 1
                                          ... done
Stopping nebuladockercompose_storaged1_1 ... done
Stopping nebuladockercompose_storaged0_1 ... done
Stopping nebuladockercompose storaged2 1 ... done
Stopping nebuladockercompose_metad2_1
                                          ... done
Stopping nebuladockercompose_metad0_1
                                         ... done
Stopping nebuladockercompose_metad1_1
Removing nebuladockercompose_console_1
Removing nebuladockercompose graphd1 1
                                         ... done
Removing nebuladockercompose_graphd_1
                                          ... done
Removing nebuladockercompose_graphd2_1
Removing nebuladockercompose_storaged1_1 ... done
Removing nebuladockercompose_storaged0_1 ... done
Removing nebuladockercompose_storaged2_1 ... done
Removing nebuladockercompose_metad2_1
                                         ... done
Removing nebuladockercompose_metad0_1
Removing nebuladockercompose metad1 1
                                             done
Removing network nebuladockercompose_nebula-net
```



The parameter -v in the command docker-compose down -v will **delete** all your local NebulaGraph storage data. Try this command if you are using the nightly release and having some compatibility issues.

## **Modify configurations**

The configuration file of NebulaGraph deployed by Docker Compose is nebula-docker-compose.yaml. To make the new configuration take effect, modify the configuration in this file and restart the service.

For more instructions, see Configurations.

#### FAQ

HOW TO FIX THE DOCKER MAPPING TO EXTERNAL PORTS?

To set the ports of corresponding services as fixed mapping, modify the docker-compose.yaml in the nebula-docker-compose directory. For example:

```
graphd:
    image: vesoft/nebula-graphd:release-3.4
    ...
    ports:
        - 9669:9669
        - 19669
        - 19670
```

9669:9669 indicates the internal port 9669 is uniformly mapped to external ports, while 19669 indicates the internal port 19669 is randomly mapped to external ports.

HOW TO UPGRADE OR UPDATE THE DOCKER IMAGES OF NEBULAGRAPH SERVICES

- 1. In the nebula-docker-compose/docker-compose.yaml file, change all the image values to the required image version.
- 2. In the nebula-docker-compose directory, run docker-compose pull to update the images of the Graph Service, Storage Service, Meta Service, and NebulaGraph Console.

- 3. Run docker-compose up -d to start the NebulaGraph services again.
- 4. After connecting to NebulaGraph with NebulaGraph Console, run SHOW HOSTS GRAPH, SHOW HOSTS STORAGE, or SHOW HOSTS META to check the version of the responding service respectively.

ERROR: TOOMANYREQUESTS WHEN DOCKER-COMPOSE PULL

You may meet the following error.

ERROR: toomanyrequests: You have reached your pull rate limit. You may increase the limit by authenticating and upgrading: https://www.docker.com/increase-rate-limit.

You have met the rate limit of Docker Hub. Learn more on Understanding Docker Hub Rate Limiting.

HOW TO UPDATE THE NEBULAGRAPH CONSOLE CLIENT

The command docker-compose pull updates both the NebulaGraph services and the NebulaGraph Console.

### **Related documents**

- Install and deploy NebulaGraph with the source code
- Install NebulaGraph by RPM or DEB
- Connect to NebulaGraph

Last update: February 19, 2024

- 393/1069 - 2023 Vesoft Inc.

# 5.2.5 Deploy a NebulaGraph cluster with RPM/DEB package on multiple servers

For now, NebulaGraph does not provide an official deployment tool. Users can deploy a NebulaGraph cluster with RPM or DEB package manually. This topic provides an example of deploying a NebulaGraph cluster on multiple servers (machines).

#### Deployment

Machine name	IP address	Number of graphd	Number of storaged	Number of metad
A	192.168.10.111	1	1	1
В	192.168.10.112	1	1	1
С	192.168.10.113	1	1	1
D	192.168.10.114	1	1	None
E	192.168.10.115	1	1	None

#### Prerequisites

- Prepare 5 machines for deploying the cluster.
- Use the NTP service to synchronize time in the cluster.

#### Manual deployment process

INSTALL NEBULAGRAPH

Install NebulaGraph on each machine in the cluster. Available approaches of installation are as follows.

- Install NebulaGraph with RPM or DEB package
- Install NebulaGraph by compiling the source code

ADD A LICENSE (FOR THE ENTERPRISE EDITION ONLY).

- Adding a license is only required when you deploy a NebulaGraph cluster with the Enterprise Edition. For details, see Deploy a license for NebulaGraph Enterprise Edition.
- Skip this step when you deploy a cluster with the Community Edition.

MODIFY THE CONFIGURATIONS

To deploy NebulaGraph according to your requirements, you have to modify the configuration files.

All the configuration files for NebulaGraph, including nebula-graphd.conf, nebula-metad.conf, and nebula-storaged.conf, are stored in the etc directory in the installation path. You only need to modify the configuration for the corresponding service on the machines. The configurations that need to be modified for each machine are as follows.

Machine name	The configuration to be modified
A	nebula-graphd.conf , nebula-storaged.conf , nebula-metad.conf
В	nebula-graphd.conf , nebula-storaged.conf , nebula-metad.conf
С	nebula-graphd.conf , nebula-storaged.conf , nebula-metad.conf
D	nebula-graphd.conf , nebula-storaged.conf
E	nebula-graphd.conf , nebula-storaged.conf

Users can refer to the content of the following configurations, which only show part of the cluster settings. The hidden content uses the default setting so that users can better understand the relationship between the servers in the NebulaGraph cluster.

- 394/1069 - 2023 Vesoft Inc.



The main configuration to be modified is <code>meta\_server\_addrs</code>. All configurations need to fill in the IP addresses and ports of all Meta services. At the same time, <code>local\_ip</code> needs to be modified as the network IP address of the machine itself. For detailed descriptions of the configuration parameters, see:

- Meta Service configurations
- Graph Service configurations
- Storage Service configurations
- · Deploy machine A
- nebula-graphd.conf

• nebula-storaged.conf

• nebula-metad.conf

- 395/1069 - 2023 Vesoft Inc.

# • Deploy machine B

### • nebula-graphd.conf

### • nebula-storaged.conf

### • nebula-metad.conf

### . Deploy machine C

• nebula-graphd.conf

nebula-storaged.conf

nebula-metad.conf

### · Deploy machine D

• nebula-graphd.conf

• nebula-storaged.conf

- . Deploy machine E
- · nebula-graphd.conf

nebula-storaged.conf

#### START THE CLUSTER

Start the corresponding service on **each machine**. Descriptions are as follows.

Machine name	The process to be started	
A	graphd, storaged, metad	
В	graphd, storaged, metad	
С	graphd, storaged, metad	
D	graphd, storaged	
Е	graphd, storaged	

The command to start the NebulaGraph services is as follows.

 $sudo \ /usr/local/nebula/scripts/nebula.service \ start \ \verb|\mathcal| graphd \ | \ storaged \ | \ all > \ | \ start \ | \ st$ 



- · Make sure all the processes of services on each machine are started. Otherwise, you will fail to start NebulaGraph.
- When the graphd process, the storaged process, and the metad process are all started, you can use all instead.
- /usr/local/nebula is the default installation path for NebulaGraph. Use the actual path if you have customized the path. For more information about how to start and stop the services, see Manage NebulaGraph services.

#### CHECK THE CLUSTER STATUS

Install the native CLI client NebulaGraph Console, then connect to any machine that has started the graphd process, run ADD HOSTS command to add storage hosts, and run SHOW HOSTS to check the cluster status. For example:

++	+
"192.168.10.111"   9779   "ONLINE"   0	"No valid partition"   "No valid partition"   "3.4.1"
"192.168.10.112"   9779   "ONLINE"   0	"No valid partition"   "No valid partition"   "3.4.1"
"192.168.10.113"   <mark>9779</mark>   "ONLINE"   <b>0</b>	"No valid partition"   "No valid partition"   "3.4.1"
"192.168.10.114"   9779   "ONLINE"   0	"No valid partition"   "No valid partition"   "3.4.1"
"192.168.10.115"   9779   "ONLINE"   0	"No valid partition"   "No valid partition"   "3.4.1"
+	+

Last update: February 19, 2024

## 5.2.6 Install NebulaGraph with ecosystem tools

You can install the Enterprise Edition and Community Edition of NebulaGraph with the following ecosystem tools:

- NebulaGraph Dashboard Enterprise Edition
- NebulaGraph Operator

#### Installation details

- To install NebulaGraph with NebulaGraph Dashboard Enterprise Edition, see Create a cluster.
- To install NebulaGraph with **NebulaGraph Operator**, see Deploy NebulaGraph clusters with Kubectl or Deploy NebulaGraph clusters with Helm.



Contact our sales (inqury@vesoft.com) to get the installation package for the Enterprise Edition of NebulaGraph.

Last update: February 19, 2024

- 400/1069 - 2023 Vesoft Inc.

## 5.3 Standalone NebulaGraph

Standalone NebulaGraph merges the Meta, Storage, and Graph services into a single process deployed on a single machine. This topic introduces scenarios, deployment steps, etc. of standalone NebulaGraph.



Do not use standalone NebulaGraph in production environments.

## 5.3.1 Background

The traditional NebulaGraph consists of three services, each service having executable binary files and the corresponding process. Processes communicate with each other by RPC. In standalone NebulaGraph, the three processes corresponding to the three services are combined into one process. For more information about NebulaGraph, see Architecture overview.

#### 5.3.2 Scenarios

Small data sizes and low availability requirements. For example, test environments that are limited by the number of machines, scenarios that are only used to verify functionality.

#### 5.3.3 Limitations

- · Single service instance per machine.
- High availability and reliability not supported.

### 5.3.4 Resource requirements

For information about the resource requirements for standalone NebulaGraph, see Software requirements for compiling NebulaGraph.

## 5.3.5 Steps

Currently, you can only install standalone NebulaGraph with the source code. The steps are similar to those of the multi-process NebulaGraph. You only need to modify the step **Generate Makefile with CMake** by adding "-DENABLE\_STANDALONE\_VERSION=on to the command. For example:

cmake -DCMAKE\_INSTALL\_PREFIX=/usr/local/nebula -DENABLE\_TESTING=OFF -DENABLE\_STANDALONE\_VERSION=on -DCMAKE\_BUILD\_TYPE=Release ...

For more information about installation details, see Install NebulaGraph by compiling the source code.

After installing standalone NebulaGraph, see the topic connect to Service to connect to NebulaGraph databases.

### 5.3.6 Configuration file

The path to the configuration file for standalone NebulaGraph is /usr/tocat/nebula/etc by default.

- 401/1069 - 2023 Vesoft Inc.

You can run sudo cat nebula-standalone.conf.default to see the file content. The parameters and the corresponding descriptions in the file are generally the same as the configurations for multi-process NebulaGraph except for the following parameters.

Parameter	Predefined value	Description
meta_port	9559	The port number of the Meta service.
storage_port	9779	The port number of the Storage Service.
meta_data_path	data/meta	The path to Meta data.

You can run commands to check configurable parameters and the corresponding descriptions. For details, see Configurations.

Last update: February 19, 2024

## 5.4 Deploy a license for NebulaGraph Enterprise Edition

NebulaGraph Enterprise Edition requires the user to deploy a license file before starting the Enterprise Edition. This topic describes how to deploy a license file for the Enterprise Edition.



License is a software authorization certificate provided for users of the Enterprise Edition. Users of the Enterprise Edition can contact us to apply for a license file.

### 5.4.1 Precautions

- If the license file is not deployed, NebulaGraph Enterprise Edition cannot be started.
- Do not modify the license file, otherwise the license will become invalid.
- If the license is about to expire, contact us to apply for renewal.
- The transition period after the license expires is 14 days:
- If you start the Enterprise Edition within 30 days before the license expires or on the day the license expires, a log will be printed as a reminder.
- The license can still be used for 14 days after it expires.
- If the license has expired for 14 days, you will not be able to start the Enterprise Edition, and a log will be printed as a reminder.

## 5.4.2 License description

The example of the content of the license file (nebula.license) is as follows:

The license file contains information such as issuedDate and expirationDate. The description is as follows.

Parameter	Description
vendor	The supplier.
organization	The username.
issuedDate	The date that the license is issued.
expirationDate	The date that the license expires.
product	The product type. The product type of NebulaGraph is $\ensuremath{nebula\_graph}$ .
version	The version information.
licenseType	The license type, including enterprise, samtl_bussiness, pro, and individual.
gracePeriod	The buffer time (in days) for the service to continue to be used after the license expires, and the service will be stopped after the buffer period. The trial version of license has no buffer period after expiration and the default value of this parameter is 0.
graphdSpec	The max number of graph services in a cluster. NebulaGraph detects the number of active graph services in real-time. You are unable to connect to the cluster once the max number is reached.
storagedSpec	The max number of storage services in a cluster. NebulaGraph detects the number of active storage services in real-time. You are unable to connect to the cluster once the max number is reached.
clusterCode	The user's hardware information, which is also the unique identifier of the cluster. This parameter is not available in the trial version of the license.

## 5.4.3 Deploy the license

- 1. Contact us to apply for the NebulaGraph Enterprise Edition package.
- 2. Install NebulaGraph Enterprise Edition. The installation method is the same as the Community Edition. See Install NebulaGraph with RPM or DEB package.
- 3. Contact us to apply for the license file nebula. License.
- 4. Upload the license file to all hosts that contain Meta services. The path is in the share/resources/ of each Meta service installation directory.



For the upload address of the license file for ecosystem tools, refer to the document of Ecosystem tools overview.

## 5.4.4 Renew a NebulaGraph Enterprise Edition license

- 1. Email us at inqury@vesoft.com to apply for a new license file nebula. License.
- $2.\ In\ share/resources/\ under\ the\ installation\ directory\ of\ each\ Meta\ service,\ replace\ the\ old\ license\ file\ with\ the\ new\ one.$
- 3. Restart Storage and Graph services. For information about how to restart services, see <u>Start NebulaGraph</u>. If your license expires within the buffer period (14 days by default), you do not have to restart Storage and Graph services.



The Graph and Storage services are automatically stopped when your license expires beyond the buffer period after expiration. To ensure that the service is running properly, please renew your license in time.

- 404/1069 - 2023 Vesoft Inc.

## 5.4.5 View the license

• View the License file directly

You can use cat to view the content of the license file directly. For example: cat share/resources/nebula.license.

• View the License file with HTTP port

When the NebulaGraph cluster is running normally, you can view the license file with the HTTP port (default port is 19559) of the meta service. For example: curl - G "http://192.168.10.101:19559/license".

Last update: February 19, 2024

## 5.5 Manage NebulaGraph Service

NebulaGraph supports managing services with scripts.



You can also manage NebulaGraph with systemd in the NebulaGraph Enterprise Edition.



The two methods are incompatible. It is recommended to use only one method in a cluster.

## 5.5.1 Manage services with script

You can use the nebula.service script to start, stop, restart, terminate, and check the NebulaGraph services.



nebula.service is stored in the /usr/local/nebula/scripts directory by default. If you have customized the path, use the actual path in your environment.

#### **Syntax**

\$ sudo /usr/local/nebula/scripts/nebula.service
[-v] [-c <config\_file\_path>]
<start | stop | restart | kill | status>
<metad | graphd | storaged | all>

Parameter	Description
-V	Display detailed debugging information.
-c	Specify the configuration file path. The default path is $\mbox{\tt /usr/local/nebula/etc/}$ .
start	Start the target services.
stop	Stop the target services.
restart	Restart the target services.
kill	Terminate the target services.
status	Check the status of the target services.
metad	Set the Meta Service as the target service.
graphd	Set the Graph Service as the target service.
storaged	Set the Storage Service as the target service.
all	Set all the NebulaGraph services as the target services.

## 5.5.2 Manage services with systemd

For easy maintenance, NebulaGraph Enterprise Edition supports managing services with systemd. You can start, stop, restart, and check services with systemctl commands.

- 406/1069 - 2023 Vesoft Inc.



- After installing NebulaGraph Enterprise Edition, the .service files required by systemd are located in the etc/unit path in the installation directory. NebulaGraph installed with the RPM/DEB package automatically places the .service files into the path /usr/lib/system/system and the parameter ExecStart is generated based on the specified NebulaGraph installation path, so you can use systemctl commands directly.
- The systematt commands cannot be used to manage the Enterprise Edition cluster that is created with Dashboard of the Enterprise Edition.
- Otherwise, users need to move the .service files manually into the directory /usr/lib/systemd/system, and modify the file path of the parameter ExecStart in the .service files.

### Syntax

\$ systemctl <start | stop | restart | status > <nebula | nebula-metad | nebula-graphd | nebula-storaged>

Parameter	Description	
start	Start the target services.	
stop	Stop the target services.	
restart	Restart the target services.	
status	Check the status of the target services.	
nebula	Set all the NebulaGraph services as the target services.	
nebula-metad	Set the Meta Service as the target service.	
nebula-graphd	Set the Graph Service as the target service.	
nebula-storaged	Set the Storage Service as the target service.	

## 5.5.3 Start NebulaGraph

Run the following command to start NebulaGraph.

```
$ sudo /usr/local/nebula/scripts/nebula.service start all
[INFO] Starting nebula-metad...
[INFO] Done
[INFO] Starting nebula-graphd...
[INFO] Done
[INFO] Starting nebula-storaged...
[INFO] Done
```

Users can also run the following command:

```
$ systemctl start nebula
```

If users want to automatically start NebulaGraph when the machine starts, run the following command:

```
$ systemctl enable nebula
```

## 5.5.4 Stop NebulaGraph



Do not run kill -9 to forcibly terminate the processes. Otherwise, there is a low probability of data loss.

- 407/1069 - 2023 Vesoft Inc.

Run the following command to stop NebulaGraph.

```
$ sudo /usr/local/nebula/scripts/nebula.service stop all
[INFO] Stopping nebula-metad...
[INFO] bone
[INFO] Stopping nebula-graphd...
[INFO] bone
[INFO] Stopping nebula-storaged...
[INFO] Done
```

Users can also run the following command:

```
$ systemctl stop nebula
```

#### 5.5.5 Check the service status

Run the following command to check the service status of NebulaGraph.

```
$ sudo /usr/local/nebula/scripts/nebula.service status all
```

• NebulaGraph is running normally if the following information is returned.

```
INFO] nebula-metad(33fd35e): Running as 29020, Listening on 9559
[IMFO] nebula-graphd(33fd35e): Running as 29095, Listening on 9669
[WARN] nebula-storaged after v3.0.0 will not start service until it is added to cluster.
[WARN] See Manage Storage hosts:ADD HOSTS in https://docs.nebula-graph.io/
[INFO] nebula-storaged(33fd35e): Running as 29147, Listening on 9779
```



After starting NebulaGraph, the port of the nebula-storaged process is shown in red. Because the nebula-storaged process waits for the nebula-metad to add the current Storage service during the startup process. The Storage works after it receives the ready signal. Starting from NebulaGraph 3.0.0, the Meta service cannot directly read or write data in the Storage service that you add in the configuration file. The configuration file only registers the Storage service to the Meta service. You must run the ADD HOSTS command to enable the Meta to read and write data in the Storage service. For more information, see Manage Storage hosts.

• If the returned result is similar to the following one, there is a problem. You may also go to the NebulaGraph community for help.

```
[INFO] nebula-metad: Running as 25600, Listening on 9559
[INFO] nebula-graphd: Exited
[INFO] nebula-storaged: Running as 25646, Listening on 9779
```

Users can also run the following command:

```
$ systemctl status nebula

■ nebula.service
Loaded: loaded (/usr/lib/system/nebula.service; disabled; vendor preset: disabled)
Active: active (exited) since — 2022-03-28 04:13:24 UTC; 1h 47min ago
Process: 21772 ExecStart=/usr/local/ent-nightly/scripts/nebula.service start all (code=exited, status=0/SUCCESS)
Main PID: 21772 (code=exited, status=0/SUCCESS)
Tasks: 325
Memory: 424.5M
CGroup: /system.slice/nebula.service
—21789 /usr/local/ent-nightly/bin/nebula-metad --flagfile /usr/local/ent-nightly/etc/nebula-graphd.conf
—21827 /usr/local/ent-nightly/bin/nebula-graphd --flagfile /usr/local/ent-nightly/etc/nebula-storaged.conf
3月 28 04:13:24 xxxxxxx systemd[1]: Started nebula.service.
...
```

The NebulaGraph services consist of the Meta Service, Graph Service, and Storage Service. The configuration files for all three services are stored in the <code>/usr/local/nebula/etc/</code> directory by default. You can check the configuration files according to the returned result to troubleshoot problems.

#### 5.5.6 Next to do

Connect to NebulaGraph

Last update: February 19, 2024

## 5.6 Connect to NebulaGraph

This topic provides basic instruction on how to use the native CLI client NebulaGraph Console to connect to NebulaGraph.



When connecting to NebulaGraph for the first time, you must register the Storage Service before querying data.

NebulaGraph supports multiple types of clients, including a CLI client, a GUI client, and clients developed in popular programming languages. For more information, see the client list.

### 5.6.1 Prerequisites

- You have started NebulaGraph services.
- The machine on which you plan to run NebulaGraph Console has network access to the Graph Service of NebulaGraph.
- The NebulaGraph Console version is compatible with the NebulaGraph version.



NebulaGraph Console and NebulaGraph of the same version number are the most compatible. There may be compatibility issues when connecting to NebulaGraph with a different version of NebulaGraph Console. The error message incompatible version between client and server is displayed when there is such an issue.

### Steps

1. On the NebulaGraph Console releases page, select a NebulaGraph Console version and click Assets.



It is recommended to select the **latest** version.

- 2. In the **Assets** area, find the correct binary file for the machine where you want to run NebulaGraph Console and download the file to the machine.
- 3. (Optional) Rename the binary file to  $\mbox{\it nebula-console}$  for convenience.



For Windows, rename the file to  ${\tt nebula-console.exe}$  .

4. On the machine to run NebulaGraph Console, grant the execute permission of the nebula-console binary file to the user.



For Windows, skip this step.

\$ chmod 111 nebula-console

5. In the command line interface, change the working directory to the one where the nebula-console binary file is stored.

- 410/1069 - 2023 Vesoft Inc.

- $_{\rm 6.}$  Run the following command to connect to NebulaGraph.
- For Linux or macOS:

```
$ ./nebula-console -addr <ip> -port <port> -u <username> -p <password>
[-t 120] [-e "nGQL_statement" | -f filename.nGQL]
```

• For Windows:

```
> nebula-console.exe -addr <ip> -port <port> -u <username> -p <password>
[-t 120] [-e "nGQL_statement" | -f filename.nGQL]
```

Parameter descriptions are as follows:

Parameter	Description
-h/-help	Shows the help menu.
-addr/-address	Sets the IP address of the Graph service. The default address is 127.0.0.1.
-P/-port	Sets the port number of the graphd service. The default port number is 9669.
-u/-user	Sets the username of your NebulaGraph account. Before enabling authentication, you can use any existing username. The default username is $\ {\sf root}$ .
-p/-password	Sets the password of your NebulaGraph account. Before enabling authentication, you can use any characters as the password.
-t/-timeout	Sets an integer-type timeout threshold of the connection. The unit is millisecond. The default value is 120.
-e/-eval	Sets a string-type nGQL statement. The nGQL statement is executed once the connection succeeds. The connection stops after the result is returned.
-f/-file	Sets the path of an nGQL file. The nGQL statements in the file are executed once the connection succeeds. The result will be returned and the connection stops then.
-enable_ssl	Enables SSL encryption when connecting to NebulaGraph.
-ssl_root_ca_path	Sets the storage path of the certification authority file.
-ssl_cert_path	Sets the storage path of the certificate file.
ssl_private_key_path	Sets the storage path of the private key file.

For information on more parameters, see the  $\operatorname{project}$  repository.

Last update: February 19, 2024

- 411/1069 - 2023 Vesoft Inc.

## 5.7 Manage Storage hosts

Starting from NebulaGraph 3.0.0, setting Storage hosts in the configuration files only registers the hosts on the Meta side, but does not add them into the cluster. You must run the ADD HOSTS statement to add the Storage hosts.



NebulaGraph Cloud clusters add Storage hosts automatically. Cloud users do not need to manually run ADD HOSTS.

## 5.7.1 Add Storage hosts

Add the Storage hosts to a NebulaGraph cluster.

```
ADD HOSTS <ip>:<port> [,<ip>:<port> ...];
ADD HOSTS "<hostname>":<port> [,"<hostname>":<port> ...];
```



- To make sure the follow-up operations work as expected, wait for two heartbeat cycles, i.e., 20 seconds, and then run SHOW HOSTS to check whether the host is online.
- Make sure that the IP address and port number are the same as those in the configuration file. For example, the default IP address and port number in standalone deployment are 127.0.0.1:9779.
- When using a domain name, enclose it in quotation marks, for example, ADD HOSTS "foo-bar":9779.
- Ensure that the storage host to be added is not used by any other cluster, otherwise, the storage adding operation will fail.

### 5.7.2 Drop Storage hosts

Delete the Storage hosts from cluster.



You can not delete an in-use Storage host directly. Delete the associated graph space before deleting the Storage host.

```
DROP HOSTS <ip>:<port> [,<ip>:<port> ...];
DROP HOSTS "<hostname>":<port> [,"+hostname>":<port> ...];
```

Last update: February 19, 2024

- 412/1069 - 2023 Vesoft Inc.

## 5.8 Upgrade

### 5.8.1 Upgrade NebulaGraph to 3.4.1

This topic describes how to upgrade NebulaGraph from version 2.x and 3.x to 3.4.1, taking upgrading from version 2.6.1 to 3.4.1 as an example.

#### Applicable source versions

This topic applies to upgrading NebulaGraph from 2.5.0 and later 2.x, and 3.x versions to 3.4.1. It does not apply to historical versions earlier than 2.5.0, including the 1.x versions.

To upgrade NebulaGraph from historical versions to 3.4.1:

- 1. Upgrade it to the latest 2.5 version according to the docs of that version.
- 2. Follow this topic to upgrade it to 3.4.1.



To upgrade NebulaGraph from versions earlier than 2.0.0 (including the 1.x versions) to 3.4.1, you need to find the date\_time\_zonespec.csv in the share/resources directory of 3.4.1 files, and then copy it to the same directory in the NebulaGraph installation path.

#### Limitations

- Rolling Upgrade is not supported. You must stop all the NebulaGraph services before the upgrade.
- There is no upgrade script. You have to manually upgrade each server in the cluster.
- This topic does not apply to scenarios where NebulaGraph is deployed with Docker, including Docker Swarm, Docker Compose, and K8s.
- You must upgrade the old NebulaGraph services on the same machines they are deployed. **DO NOT** change the IP addresses, configuration files of the machines, and **DO NOT** change the cluster topology.
- Known issues that could cause data loss are listed on GitHub known issues. The issues are all related to altering schema or default values.
- DO NOT use soft links to switch the data directories.
- You must have the sudo privileges to complete the steps in this topic.

### Upgrade influences

· Client compatibility

- 413/1069 - 2023 Vesoft Inc.

After the upgrade, you will not be able to connect to NebulaGraph from old clients. You will need to upgrade all clients to a version compatible with NebulaGraph 3.4.1.

· Configuration changes

A few configuration parameters have been changed. For more information, see the release notes and configuration docs.

• nGQL compatibility

The nGQL syntax is partially incompatible:

- Disable the YIELD clause to return custom variables.
- The YIELD clause is required in the FETCH, GO, LOOKUP, FIND PATH and GET SUBGRAPH statements.
- It is required to specify a tag to query properties of a vertex in a MATCH statement. For example, from return v.name to return v.player.name.
- · Full-text indexes

Before upgrading a NebulaGraph cluster with full-text indexes deployed, you must manually delete the full-text indexes in Elasticsearch, and then run the SIGN IN command to log into ES and recreate the indexes after the upgrade is complete. To manually delete the full-text indexes in Elasticsearch, you can use the curl command curl -XDELETE -u <es\_username>:<es\_password> '<es\_access\_ip>:<port>/<fullindex\_name>' , for example, curl -XDELETE -u elastic:elastic 'http://192.168.8.xxx:9200/nebula\_index\_2534' . If no username and password are set for Elasticsearch, you can omit the -u <es\_username>:<es\_password> part.



There may be other undiscovered influences. Before the upgrade, we recommend that you read the release notes and user manual carefully, and keep an eye on the posts on the forum and issues on Github.

#### Preparations before the upgrade

• Download the package of NebulaGraph 3.4.1 according to your operating system and system architecture. You need the binary files during the upgrade. Find the package on the download page.



You can also get the new binaries from the source code or the RPM/DEB package.

• Locate the data files based on the value of the data\_path parameters in the Storage and Meta configurations, and backup the data files. The default paths are nebula/data/storage and nebula/data/meta.



The old data will not be automatically backed up during the upgrade. You must manually back up the data to avoid data loss.

- Backup the configuration files.
- Collect the statistics of all graph spaces before the upgrade. After the upgrade, you can collect again and compare the results to make sure that no data is lost. To collect the statistics:
- a. Run SUBMIT JOB STATS.
- b. Run SHOW JOBS and record the result.

- 414/1069 - 2023 Vesoft Inc.

#### Upgrade steps

1. Stop all NebulaGraph services.

<nebula\_install\_path>/scripts/nebula.service stop all

nebula\_install\_path indicates the installation path of NebulaGraph.

The storaged progress needs around 1 minute to flush data. You can run nebula.service status all to check if all services are stopped. For more information about starting and stopping services, see Manage services.



If the services are not fully stopped in 20 minutes, stop upgrading and ask for help on the forum or Github.



Starting from version 3.0.0, it is possible to insert vertices without tags. If you need to keep vertices without tags, add -graph\_use\_vertex\_key=true in the configuration file ( nebula-graphd.conf ) of all Graph services within the cluster; and add --use\_vertex\_key=true in
the configuration file ( nebula-storaged.conf ) of all Storage services."

2. In the target path where you unpacked the package, use the binaries in the bin directory to replace the old binaries in the bin directory in the NebulaGraph installation path.

#### Q Note

Update the binary of the corresponding service on each NebulaGraph server.

- 3. Modify the following parameters in all Graph configuration files to accommodate the value range of the new version. If the parameter values are within the specified range, skip this step.
- Set a value in [1,604800] for session\_idle\_timeout\_secs. The recommended value is 28800.
- Set a value in [1,604800] for client\_idle\_timeout\_secs . The recommended value is 28800.

The default values of these parameters in the 2.x versions are not within the range of the new version. If you do not change the default values, the upgrade will fail. For detailed parameter description, see <u>Graph Service Configuration</u>.

4. Start all Meta services.

<nebula\_install\_path>/scripts/nebula-metad.service start

Once started, the Meta services take several seconds to elect a leader.

To verify that Meta services are all started, you can start any Graph server, connect to it through NebulaGraph Console, and run SHOW HOSTS metal and SHOW META LEADER. If the status of Meta services are correctly returned, the services are successfully started.



If the operation fails, stop the upgrade and ask for help on the forum or GitHub.

5. Start all the Graph and Storage services.



If the operation fails, stop the upgrade and ask for help on the forum or GitHub.

6. Connect to the new version of NebulaGraph to verify that services are available and data are complete. For how to connect, see Connect to NebulaGraph.

- 415/1069 - 2023 Vesoft Inc.

Currently, there is no official way to check whether the upgrade is successful. You can run the following reference statements to test the upgrade:

```
nebula> SHOW HOSTS;
nebula> SHOW HOSTS storage;
nebula> SHOW SPACES;
nebula> USE <space_name>
nebula> SHOW PARTS;
nebula> SHOW PARTS;
nebula> SHOW STATS;
nebula> SHOW STATS;
nebula> MATCH (v) RETURN v LIMIT 5;
```

You can also test against new features in version 3.4.1.

### Upgrade failure and rollback

If the upgrade fails, stop all NebulaGraph services of the new version, recover the old configuration files and binaries, and start the services of the old version.

All NebulaGraph clients in use must be switched to the old version.

#### FAQ

CAN I WRITE THROUGH THE CLIENT DURING THE UPGRADE?

A: No. You must stop all NebulaGraph services during the upgrade.

THE SPACE O NOT FOUND WARNING MESSAGE DURING THE UPGRADE PROCESS

When the Space 0 not found warning message appears during the upgrade process, you can ignore it. The space 0 is used to store meta information about the Storage service and does not contain user data, so it will not affect the upgrade.

HOW TO UPGRADE IF A MACHINE HAS ONLY THE GRAPH SERVICE, BUT NOT THE STORAGE SERVICE?

A: You only need to update the configuration files and binaries of the Graph Service.

HOW TO RESOLVE THE ERROR PERMISSION DENIED?

A: Try again with the sudo privileges.

IS THERE ANY CHANGE IN GFLAGS?

A: Yes. For more information, see the release notes and configuration docs.

IS THERE A TOOL OR SOLUTION FOR VERIFYING DATA CONSISTENCY AFTER THE UPGRADE?

A: No. But if you only want to check the number of vertices and edges, run SUBMIT JOB STATS and SHOW STATS after the upgrade, and compare the result with the result that you recorded before the upgrade.

HOW TO SOLVE THE ISSUE THAT STORAGE IS OFFLINE AND LEADER COUNT IS 0?

A: Run the following statement to add the Storage hosts into the cluster manually.

```
ADD HOSTS <ip>:<port>[, <ip>:<port> ...];
```

## For example:

```
ADD HOSTS 192.168.10.100:9779, 192.168.10.101:9779, 192.168.10.102:9779;
```

If the issue persists, ask for help on the forum or GitHub.

WHY THE JOB TYPE CHANGED AFTER THE UPGRADE, BUT JOB ID REMAINS THE SAME?

A: SHOW JOBS depends on an internal ID to identify job types, but in NebulaGraph 2.5.0 the internal ID changed in this pull request, so this issue happens after upgrading from a version earlier than 2.5.0.

Last update: February 19, 2024

- 416/1069 - 2023 Vesoft Inc.

## 5.8.2 Upgrade NebulaGraph Enterprise Edition from version 3.x to 3.4.1

This topic takes the enterprise edition of NebulaGraph v3.1.0 as an example and describes how to upgrade to v3.4.1.

#### Notes

• This upgrade is only applicable for upgrading the enterprise edition of NebulaGraph v3.x (x < 4) to v3.4.1. For upgrading from version 3.4.0 and above to 3.4.1, you can directly replace the binary files for upgrade. For more information, see Upgrade NebulaGraph to 3.4.1 . !!! note

If your version is below 3.0.0, please upgrade to enterprise edition 3.1.0 before upgrading to v3.4.1. For details, see [Upgrade NebulaGraph Enterprise Edition 2.x to 3.1.0](https://docs.nebula-graph.io/3.1.0/4.deployment-and-installation/3.upgrade-nebula-graph/upgrade-nebula-graph-to-latest/).

- The IP address of the machine performing the upgrade operation must be the same as the original machine.
- The remaining disk space on the machine must be at least 1.5 times the size of the original data directory.
- Before upgrading a NebulaGraph cluster with full-text indexes deployed, you must manually delete the full-text indexes in Elasticsearch, and then run the SIGN IN command to log into ES and recreate the indexes after the upgrade is complete.



To manually delete the full-text indexes in Elasticsearch, you can use the curl command curl -XDELETE -u <es\_username>:<es\_password> '<es\_access\_ip>:<port>/<fullindex\_name>' , for example, curl -XDELETE -u elastic:elastic 'http://192.168.8.223:9200/nebula\_index\_2534' . If no username and password are set for Elasticsearch, you can omit the -u <es\_username>:<es\_password> part.

### Steps

1. Contact us to obtain the installation package of the enterprise edition of NebulaGraph v3.4.1 and install it.



The upgrade steps are the same for different installation packages. This article uses the RPM package and the installation directory / usr/local/nebulagraph-ent-3.4 as an example. See Install with RPM packages for specific operations.



Please ensure that the number of storage paths set for the --data\_path parameter in the Meta and Storage service configuration files of the 3.4.1 cluster is the same as that for the --data\_path parameter in the configuration files of the 3.x cluster. Otherwise, the upgraded cluster will not start.

2. Stop the enterprise edition of v3.x services. For details see Manage NebulaGraph services.

Run the nebula service status all command to confirm that all services have been stopped after running the command.

- 417/1069 - 2023 Vesoft Inc.

- 3. In the installation directory of the Enterprise Edition NebulaGraph v3.4.1, run the following commands to upgrade the Storage and Meta services.
- Upgrade the Storage service:

#### Syntax:

sudo ./bin/db\_upgrader --max\_concurrent\_parts<num> --src\_db\_path<source\_storage\_data\_path> --dst\_db\_path<destination\_storage\_data\_path>

Parameter	Description
max_concurrent_parts	Specify the number of partitions to upgrade simultaneously, with the default value being 1. It is recommended to increase the value appropriately based on disk performance.
src_db_path	Specify the absolute path to the source data directory. The following takes the source data directory $/usr/local/nebula-ent-3.1.0/data/storage$ as an example.
dst_db_path	Specify the absolute path to the target data directory. The example target data directory is $\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/$

#### Example:

```
sudo ./bin/db_upgrader --max_concurrent_parts=20 --src_db_path=/usr/local/nebula-ent-3.1.0/data/storage --dst_db_path=/usr/local/nebula-ent-3.4/data/storage
```

If there are multiple source data directories, specify each source data directory and target data directory and run the corresponding command. For example, there are two source data directories /usr/local/nebula-ent-3.1.0/data/storage and /usr/local/nebula-ent-3.1.0/data2/storage, run the following commands:

```
sudo ./bin/db_upgrader --src_db_path=/usr/local/nebula-ent-3.1.0/data/storage --dst_db_path=/usr/local/nebula-ent-3.4/data/storage sudo ./bin/db_upgrader --src_db_path=/usr/local/nebula-ent-3.1.0/data2/storage --dst_db_path=/usr/local/nebula-ent-3.4/data2/storage
```

• Upgrade the Meta service:

### Syntax:

sudo ./bin/meta\_upgrader --src\_meta\_path=<source\_meta\_data\_path> --dst\_meta\_path=<destination\_meta\_data\_path>

Parameter	Description
src_meta_path	Specify the absolute path to the source meta data directory. The following takes the source data directory /usr/local/nebula-ent-3.1.0/data/meta as an example.
dst_meta_path	Specify the absolute path to the target meta data directory. The example target data directory is $\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/\/$

## Example:

```
sudo ./bin/meta_upgrader --src_meta_path=/usr/local/nebula-ent-3.1.0/data/meta --dst_meta_path=/usr/local/nebula-ent-3.4/data/meta
```

If there are multiple source meta data directories, specify each source meta data directory and target meta data directory and run the corresponding command.

After the upgrade, a data directory will be generated in the v3.4.1 installation directory, containing the upgraded data files.

- $4. \ Upload \ the \ license \ file \ to \ the \ \ share/resources \ directory \ under \ the \ v3.4.1 \ installation \ directory.$
- 5. Start and connect to the NebulaGraph v3.4.1 enterprise edition service and verify that the data is correct. The following commands can be used as reference:

```
nebula> SHOW HOSTS;
nebula> SHOW HOSTS storage;
nebula> SHOW SPACES;
nebula> USE <pace_name>
nebula> SHOW PARTS;
nebula> SUBNIT JOB STATS;
nebula> SHOW STATS;
nebula> MATCH (v) RETURN v LIMIT 5;
```

- 418/1069 - 2023 Vesoft Inc.

## **Docker Compose Deployment**



For NebulaGraph deployed using Docker Compose, it is recommended to redeploy the new version and import data.

Last update: February 19, 2024

## 5.9 Uninstall NebulaGraph

This topic describes how to uninstall NebulaGraph.



Before re-installing NebulaGraph on a machine, follow this topic to completely uninstall the old NebulaGraph, in case the remaining data interferes with the new services, including inconsistencies between Meta services.

## 5.9.1 Prerequisite

The NebulaGraph services should be stopped before the uninstallation. For more information, see Manage NebulaGraph services.

## 5.9.2 Step 1: Delete data files of the Storage and Meta Services

If you have modified the data\_path in the configuration files for the Meta Service and Storage Service, the directories where NebulaGraph stores data may not be in the installation path of NebulaGraph. Check the configuration files to confirm the data paths, and then manually delete the directories to clear all data.



For a NebulaGraph cluster, delete the data files of all Storage and Meta servers.

1. Check the Storage Service disk settings. For example:

- 2. Check the Metad Service configurations and find the corresponding metadata directories.
- 3. Delete the data and the directories found in step 2.

## 5.9.3 Step 2: Delete the installation directories



Delete all installation directories, including the <code>cluster.id</code> file in them.

The default installation path is /usr/local/nebula, which is specified by --prefix while installing NebulaGraph.

### Uninstall NebulaGraph deployed with source code

Find the installation directories of NebulaGraph, and delete them all.

## Uninstall NebulaGraph deployed with RPM packages

1. Run the following command to get the NebulaGraph version.

```
$ rpm -qa | grep "nebula"
```

The return message is as follows.

- 420/1069 - 2023 Vesoft Inc.

nebula-graph-3.4.1-1.x86\_64

2. Run the following command to uninstall NebulaGraph.

sudo rpm -e <nebula\_version>

For example:

sudo rpm -e nebula-graph-3.4.1-1.x86\_64

3. Delete the installation directories.

## Uninstall NebulaGraph deployed with DEB packages

1. Run the following command to get the NebulaGraph version.

```
$ dpkg -l | grep "nebula"
```

The return message is as follows.

```
ii nebula-graph 3.4.1 amd64 NebulaGraph Package built using CMake
```

 $2. \ Run \ the \ following \ command \ to \ uninstall \ Nebula Graph.$ 

sudo dpkg -r <nebula\_version>

For example:

sudo dpkg -r nebula-graph

3. Delete the installation directories.

## Uninstall NebulaGraph deployed with Docker Compose

 $1.\ In\ the\ \textit{nebula-docker-compose}\ directory,\ run\ the\ following\ command\ to\ stop\ the\ NebulaGraph\ services.$ 

docker-compose down -v

2. Delete the nebula-docker-compose directory.

Last update: February 19, 2024

- 421/1069 - 2023 Vesoft Inc.

# 6. Configurations and logs

## 6.1 Configurations

## 6.1.1 Configurations

NebulaGraph builds the configurations based on the gflags repository. Most configurations are flags. When the NebulaGraph service starts, it will get the configuration information from Configuration files by default. Configurations that are not in the file apply the default values.



The tuning service for performance, parameters and query statements are provided only in the Enterprise Edition.



- Because there are many configurations and they may change as NebulaGraph develops, this topic will not introduce all configurations. To get detailed descriptions of configurations, follow the instructions below.
- It is not recommended to modify the configurations that are not introduced in this topic, unless you are familiar with the source code and fully understand the function of configurations.

#### L Jacy version compatibility

In the topic of 1.x, we provide a method of using the CONFIGS command to modify the configurations in the cache. However, using this method in a production environment can easily cause inconsistencies of configurations between clusters and the local. Therefore, this method will no longer be introduced starting with version 2.x.

## Get the configuration list and descriptions

Use the following command to get all the configuration information of the service corresponding to the binary file:

```
<binary> --help
```

#### For example:

```
# Get the help information from Meta
$ /usr/local/nebula/bin/nebula-metad --help

# Get the help information from Graph
$ /usr/local/nebula/bin/nebula-graphd --help

# Get the help information from Storage
$ /usr/local/nebula/bin/nebula-storaged --help
```

The above examples use the default storage path /usr/local/nebula/bin/. If you modify the installation path of NebulaGraph, use the actual path to query the configurations.

## **Get configurations**

Use the curl command to get the value of the running configurations.

#### For example:

```
# Get the running configurations from Meta
curl 127.0.0.1:19559/flags
# Get the running configurations from Graph
```

curl 127.0.0.1:19669/flags

# Get the running configurations from Storage curl 127.0.0.1:19779/flags



In an actual environment, use the real host IP address instead of 127.0.0.1 in the above example.

#### Configuration files

CONFIGURATION FILES FOR CLUSTERS INSTALLED FROM SOURCE, WITH AN RPM/DEB PACKAGE, OR A TAR PACKAGE

NebulaGraph provides two initial configuration files for each service, <service\_name>.conf.default and <service\_name>.conf.production. You can use them in different scenarios conveniently. For clusters installed from source and with a RPM/DEB package, the default path is /usr/local/nebula/etc/. For clusters installed with a TAR package, the path is <install\_path>/<tar\_package\_directory>/etc.

The configuration values in the initial configuration file are for reference only and can be adjusted according to actual needs. To use the initial configuration file, choose one of the above two files and delete the suffix default or production to make it valid.



To ensure the availability of services, the configurations of the same service must be consistent, except for the local IP address local\_ip. For example, three Storage servers are deployed in one NebulaGraph cluster. The configurations of the three Storage servers need to be the same, except for the IP address.

The initial configuration files corresponding to each service are as follows.

NebulaGraph service	Initial configuration file	Description
Meta	${\tt nebula-metad.conf.default} \ \ and \ \ {\tt nebula-metad.conf.production}$	Meta service configuration
Graph	${\tt nebula-graphd.conf.default} \ \ and \ \ {\tt nebula-graphd.conf.production}$	Graph service configuration
Storage	${\tt nebula-storaged.conf.default} \ \ and \ \ {\tt nebula-storaged.conf.production}$	Storage service configuration

Each initial configuration file of all services contains <code>local\_config</code>. The default value is <code>true</code>, which means that the NebulaGraph service will get configurations from its configuration files and start it.



It is not recommended to modify the value of <code>local\_config</code> to <code>false</code>. If modified, the NebulaGraph service will first read the cached configurations, which may cause configuration inconsistencies between clusters and cause unknown risks.

CONFIGURATION FILES FOR CLUSTERS INSTALLED WITH DOCKER COMPOSE

For clusters installed with Docker Compose, the configuration file's default installation path of the cluster is <install\_path>/nebula-docker-compose/docker-compose.yaml. The parameters in the command field of the file are the launch parameters for each service.

CONFIGURATION FILES FOR CLUSTERS INSTALLED WITH NEBULAGRAPH OPERATOR

For clusters installed with Kubectl through NebulaGraph Operator, the configuration file's path is the path of the cluster YAML file. You can modify the configuration of each service through the <code>spec.{graphd|storaged|metad}.config</code> parameter.



The services cannot be configured for clusters installed with Helm.

#### Modify configurations

You can modify the configurations of NebulaGraph in the configuration file or use commands to dynamically modify configurations.



Using both methods to modify the configuration can cause the configuration information to be managed inconsistently, which may result in confusion. It is recommended to only use the configuration file to manage the configuration, or to make the same modifications to the configuration file after dynamically updating the configuration through commands to ensure consistency.

MODIFYING CONFIGURATIONS IN THE CONFIGURATION FILE

By default, each NebulaGraph service gets configured from its configuration files. You can modify configurations and make them valid according to the following steps:

- For clusters installed from source, with a RPM/DEB, or a TAR package
- a. Use a text editor to modify the configuration files of the target service and save the modification.
- b. Choose an appropriate time to restart all NebulaGraph services to make the modifications valid.
- · For clusters installed with Docker Compose
- a. In the <install\_path>/nebula-docker-compose/docker-compose.yaml file, modify the configurations of the target service.
- b. In the nebula-docker-compose directory, run the command docker-compose up -d to restart the service involving configuration modifications.
- For clusters installed with Kubectl

For details, see Customize configuration parameters for a NebulaGraph cluster.

DYNAMICALLY MODIFYING CONFIGURATIONS USING COMMAND

You can dynamically modify the configuration of NebulaGraph by using the curl command. For example, to modify the  $wal_ttl$  parameter of the Storage service to 600, use the following command:

```
curl -X PUT -H "Content-Type: application/json" -d'{"wal_ttl":"600"}' -s "http://192.168.15.6:19779/flags"
```

In this command, {"wal\_ttl":"600"} specifies the configuration parameter and its value to be modified, and 192.168.15.6:19779 specifies the IP address and HTTP port number of the Storage service.

# Caution

- The functionality of dynamically modifying configurations is only applicable to prototype verification and testing environments. It is not recommended to use this feature in production environments. This is because when the <code>local\_config</code> value is set to <code>true</code>, the dynamically modified configuration is not persisted, and the configuration will be restored to the initial configuration after the service is restarted.
- Only **part of** the configuration parameters can be dynamically modified. For the specific list of parameters that can be modified, see the description of **Whether supports runtime dynamic modifications** in the respective service configuration.

Last update: February 19, 2024

### 6.1.2 Meta Service configuration

 $Nebula Graph\ provides\ two\ initial\ configuration\ files\ for\ the\ Meta\ Service,\ nebula-metad.conf.default\ and\ nebula-metad.conf.production\ .$  Users can use them in different scenarios conveniently. The default file path is  $\ /usr/local/nebula/etc/\ .$ 

# Caution

- It is not recommended to modify the value of <code>local\_config</code> to <code>false</code>. If modified, the NebulaGraph service will first read the cached configurations, which may cause configuration inconsistencies between clusters and cause unknown risks.
- It is not recommended to modify the configurations that are not introduced in this topic, unless you are familiar with the source code and fully understand the function of configurations.

### How to use the configuration files

To use the initial configuration file, choose one of the above two files and delete the suffix default or production from the initial configuration file for the Meta Service to apply the configurations defined in it.

#### About parameter values

If a parameter is not set in the configuration file, NebulaGraph uses the default value. Not all parameters are predefined. And the predefined parameters in the two initial configuration files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different.



Some parameter values in the configuration file can be dynamically modified during runtime. We label these parameters as **Yes** that supports runtime dynamic modification in this article. When the <code>local\_config</code> value is set to <code>true</code>, the dynamically modified configuration is not persisted, and the configuration will be restored to the initial configuration after the service is restarted. For more information, see <code>Modify configurations</code>.

For all parameters and their current values, see Configurations.

## **Basics configurations**

Name	Predefined value	Description	Whether supports runtime dynamic modifications
daemonize	true	When set to true, the process is a daemon process.	No
pid_file	pids/nebula- metad.pid	The file that records the process ID.	No
timezone_name	-	Specifies the NebulaGraph time zone. This parameter is not predefined in the initial configuration files. You can manually set it if you need it. The system default value is UTC+00:00:00. For the format of the parameter value, see Specifying the Time Zone with TZ. For example,timezone_name=UTC+08:00 represents the GMT+8 time zone.	No
Name	Predefined value	Description	Whether supports runtime dynamic modifications
license_path	share/resources/ nebula.License	Path of the license of the NebulaGraph Enterprise Edition. Users need to deploy a license file before starting the Enterprise Edition. This parameter is required only for the NebulaGraph Enterprise Edition. For details about how to configure licenses for other ecosystem tools, see the deployment documents of the corresponding ecosystem tools.	No



- While inserting property values of time types, NebulaGraph transforms time types (except TIMESTAMP) to the corresponding UTC according to the time zone specified by timezone\_name. The time-type values returned by nGQL queries are all UTC time.
- timezone\_name is only used to transform the data stored in NebulaGraph. Other time-related data of the NebulaGraph processes still uses the default time zone of the host, such as the log printing time.

## Logging configurations

	Predefined value	Description	Whether supports runtime dynamic modifications
log_dir	logs	The directory that stores the Meta Service log. It is recommended to put logs on a different hard disk from the data.	No
minloglevel	0	Specifies the minimum level of the log. That is, log messages at or above this level. Optional values are 0 (INFO), 1 (WARNING), 2 (ERROR), 3 (FATAL). It is recommended to set it to 0 during debugging and 1 in a production environment. If it is set to 4, NebulaGraph will not print any logs.	Yes
v	0	Specifies the detailed level of VLOG. That is, log all VLOG messages less or equal to the level. Optional values are 0, 1, 2, 3, 4, 5. The VLOG macro provided by glog allows users to define their own numeric logging levels and control verbose messages that are logged with the parameter v. For details, see Verbose Logging.	Yes
logbufsecs	0	Specifies the maximum time to buffer the logs. If there is a timeout, it will output the buffered log to the log file. 0 means real-time output. This configuration is measured in seconds.	No
redirect_stdout	true	When set to true, the process redirects the stdout and stderr to separate output files.	No
stdout_log_file	metad- stdout.log	Specifies the filename for the stdout log.	No
stderr_log_file	metad- stderr.log	Specifies the filename for the stderr log.	No
stderrthreshold	3	Specifies the minloglevel to be copied to the stderr log.	No
timestamp_in_logfile_name	true	Specifies if the log file name contains a timestamp. true indicates yes, false indicates no.	No

- 427/1069 - 2023 Vesoft Inc.

## **Networking configurations**

Name	Predefined value	Description	Whether supports runtime dynamic modifications
meta_server_addrs	127.0.0.1:9559	Specifies the IP addresses and ports of all Meta Services. Multiple addresses are separated with commas.	No
local_ip	127.0.0.1	Specifies the local IP for the Meta Service. The local IP address is used to identify the nebula-metad process. If it is a distributed cluster or requires remote access, modify it to the corresponding address.	No
port	9559	Specifies RPC daemon listening port of the Meta service. The external port for the Meta Service is predefined to 9559. The internal port is predefined to port + 1, i.e., 9560. Nebula Graph uses the internal port for multi-replica interactions.	No
ws_ip	0.0.0.0	Specifies the IP address for the HTTP service.	No
ws_http_port	19559	Specifies the port for the HTTP service.	No
ws_storage_http_port	19779	Specifies the Storage service listening port used by the HTTP protocol. It must be consistent with the ws_http_port in the Storage service configuration file. This parameter only applies to standalone NebulaGraph.	No
heartbeat_interval_secs	10	Specifies the default heartbeat interval. Make sure the heartbeat_interval_secs values for all services are the same, otherwise NebulaGraph <b>CANNOT</b> work normally. This configuration is measured in seconds.	Yes



The real IP address must be used in the configuration file. Otherwise, 127.0.0.1/0.0.0.0 cannot be parsed correctly in some cases.

## Storage configurations

Name	Predefined Value	Description	Whether supports runtime dynamic modifications
data_path	data/meta	The storage path for Meta data.	No

## Misc configurations

Name	Predefined Value	Description	Whether supports runtime dynamic modifications
default_parts_num	100	Specifies the default partition number when creating a new graph space.	No
default_replica_factor	1	Specifies the default replica number when creating a new graph space.	No

- 428/1069 - 2023 Vesoft Inc.

## RocksDB options configurations

Name	Predefined Value	Description	Whether supports runtime dynamic modifications
rocksdb_wal_sync	true	Enables or disables RocksDB WAL synchronization. Available values are true (enable) and false (disable).	No

## Black box configurations



The Nebula-BBox configurations are for the Enterprise Edition only.

Name	Predefined Value	Description	Whether supports runtime dynamic modifications
ng_black_box_switch	true	Whether to enable the Nebula-BBox feature.	No
ng_black_box_home	black_box	The name of the directory to store Nebula-BBox file data.	No
ng_black_box_dump_period_seconds	5	The time interval for Nebula-BBox to collect metric data. Unit: Second.	No
ng_black_box_file_lifetime_seconds	1800	Storage time for Nebula-BBox files generated after collecting metric data. Unit: Second.	Yes

Last update: February 19, 2024

- 429/1069 - 2023 Vesoft Inc.

### 6.1.3 Graph Service configuration

NebulaGraph provides two initial configuration files for the Graph Service, <code>nebula-graphd.conf.default</code> and <code>nebula-graphd.conf.production</code>. Users can use them in different scenarios conveniently. The default file path is /usr/local/nebula/etc/.



- It is not recommended to modify the value of <code>local\_config</code> to <code>false</code>. If modified, the NebulaGraph service will first read the cached configurations, which may cause configuration inconsistencies between clusters and cause unknown risks.
- It is not recommended to modify the configurations that are not introduced in this topic, unless you are familiar with the source code and fully understand the function of configurations.

### How to use the configuration files

To use the initial configuration file, choose one of the above two files and delete the suffix default or production from the initial configuration file for the Meta Service to apply the configurations defined in it.

#### About parameter values

If a parameter is not set in the configuration file, NebulaGraph uses the default value. Not all parameters are predefined. And the predefined parameters in the two initial configuration files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different. This topic uses the parameters in <a href="mailto:nebula-metad.configuration">nebula-metad.configuration</a> files are different.



Some parameter values in the configuration file can be dynamically modified during runtime. We label these parameters as **Yes** that supports runtime dynamic modification in this article. When the <code>local\_config</code> value is set to <code>true</code>, the dynamically modified configuration is not persisted, and the configuration will be restored to the initial configuration after the service is restarted. For more information, see <code>Modify configurations</code>.

For all parameters and their current values, see Configurations.

### **Basics configurations**

Name	Predefined value	Description	Whether supports runtime dynamic modifications
daemonize	true	When set to $\ensuremath{true}$ , the process is a daemon process.	No
pid_file	pids/nebula- graphd.pid	The file that records the process ID.	No
enable_optimizer	true	When set to true, the optimizer is enabled.	No
timezone_name	-	Specifies the NebulaGraph time zone. This parameter is not predefined in the initial configuration files. The system default value is UTC+00:00:00. For the format of the parameter value, see Specifying the Time Zone with TZ. For example,timezone_name=UTC+08:00 represents the GMT+8 time zone.	No
local_config	true	When set to true, the process gets configurations from the configuration files.	No



- While inserting property values of time types, NebulaGraph transforms time types (except TIMESTAMP) to the corresponding UTC according to the time zone specified by timezone\_name. The time-type values returned by nGQL queries are all UTC time.
- timezone\_name is only used to transform the data stored in NebulaGraph. Other time-related data of the NebulaGraph processes still uses the default time zone of the host, such as the log printing time.

### Logging configurations

Name	Predefined value	Description	Whether supports runtime dynamic modifications
log_dir	logs	The directory that stores the Meta Service log. It is recommended to put logs on a different hard disk from the data.	No
minloglevel	0	Specifies the minimum level of the log. That is, log messages at or above this level. Optional values are 0 (INFO), 1 (WARNING), 2 (ERROR), 3 (FATAL). It is recommended to set it to 0 during debugging and 1 in a production environment. If it is set to 4, NebulaGraph will not print any logs.	Yes
V	0	Specifies the detailed level of VLOG. That is, log all VLOG messages less or equal to the level. Optional values are 0, 1, 2, 3, 4, 5. The VLOG macro provided by glog allows users to define their own numeric logging levels and control verbose messages that are logged with the parameter v. For details, see Verbose Logging.	Yes
logbufsecs	0	Specifies the maximum time to buffer the logs. If there is a timeout, it will output the buffered log to the log file. 0 means real-time output. This configuration is measured in seconds.	No
redirect_stdout	true	When set to true, the process redirects the stdout and stderr to separate output files.	No
stdout_log_file	graphd- stdout.log	Specifies the filename for the stdout log.	No
stderr_log_file	graphd- stderr.log	Specifies the filename for the stderr log.	No
stderrthreshold	3	Specifies the minloglevel to be copied to the stderr log.	No
timestamp_in_logfile_name	true	Specifies if the log file name contains a timestamp. true indicates yes, false indicates no.	No

## Query configurations

Name	Predefined value	Description	Whether supports runtime dynamic modifications
accept_partial_success	false	When set to false, the process treats partial success as an error. This configuration only applies to read-only requests. Write requests always treat partial success as an error.	Yes
session_reclaim_interval_secs	60	Specifies the interval that the Session information is sent to the Meta service. This configuration is measured in seconds.	Yes
max_allowed_query_size	4194304	Specifies the maximum length of queries. Unit: bytes. The default value is $^{4194304}$ , namely $^{4}MB$ .	Yes

- 432/1069 - 2023 Vesoft Inc.

**Networking configurations** 

Name	Predefined value	Description	Whether supports runtime dynamic modifications
meta_server_addrs	127.0.0.1:9559	Specifies the IP addresses and ports of all Meta Services. Multiple addresses are separated with commas.	No
local_ip	127.0.0.1	Specifies the local IP for the Graph Service. The local IP address is used to identify the nebulagraphd process. If it is a distributed cluster or requires remote access, modify it to the corresponding address.	No
listen_netdev	any	Specifies the listening network device.	No
port	9669	Specifies RPC daemon listening port of the Graph service.	No
reuse_port	false	When set to false, the ${\tt SO\_REUSEPORT}$ is closed.	No
listen_backlog	1024	Specifies the maximum length of the connection queue for socket monitoring. This configuration must be modified together with the <pre>net.core.somaxconn</pre> .	No
client_idle_timeout_secs	28800	Specifies the time to expire an idle connection. The value ranges from 1 to 604800. The default is 8 hours. This configuration is measured in seconds.	No
session_idle_timeout_secs	28800	Specifies the time to expire an idle session. The value ranges from 1 to 604800. The default is 8 hours. This configuration is measured in seconds.	No
num_accept_threads	1	Specifies the number of threads that accept incoming connections.	No
num_netio_threads	0	Specifies the number of networking IO threads. $\ensuremath{\text{0}}$ is the number of CPU cores.	No
num_worker_threads	0	Specifies the number of threads that execute queries. 0 is the number of CPU cores.	No
ws_i p	0.0.0.0	Specifies the IP address for the HTTP service.	No
ws_http_port	19669	Specifies the port for the HTTP service.	No
heartbeat_interval_secs	10	Specifies the default heartbeat interval. Make sure the heartbeat_interval_secs values for all services are the same, otherwise NebulaGraph <b>CANNOT</b> work normally. This configuration is measured in seconds.	Yes
storage_client_timeout_ms	-	Specifies the RPC connection timeout threshold between the Graph Service and the Storage Service. This parameter is not predefined in the initial configuration files. You can manually set it if you need it. The system default value is 60000 ms.	No
enable_record_slow_query	true	Whether to record slow queries. Only available in NebulaGraph Enterprise Edition.	No
slow_query_limit	100	The maximum number of slow queries that can be recorded.	No

- 434/1069 - 2023 Vesoft Inc.

Name	Predefined value	Description	Whether supports runtime dynamic modifications
slow_query_threshold_us	200000	When the execution time of a query exceeds the value, the query is called a slow query. Unit: Microsecond.	No
ws_meta_http_port	19559	Specifies the Meta service listening port used by the HTTP protocol. It must be consistent with the ws_http_port in the Meta service configuration file.	No



The real IP address must be used in the configuration file. Otherwise, 127.0.0.1/0.0.0.0 cannot be parsed correctly in some cases.

## Charset and collate configurations

Name	Predefined value	Description	Whether supports runtime dynamic modifications
default_charset	utf8	Specifies the default charset when creating a new graph space.	No
default_collate	utf8_bin	Specifies the default collate when creating a new graph space.	No

# Authorization configurations

Name	Predefined value	Description	Whether supports runtime dynamic modifications
enable_authorize	false	When set to false, the system authentication is not enabled. For more information, see Authentication.	No
auth_type	password	Specifies the login method. Available values are password, tdap, and cloud.	No

## Memory configurations

Name	Predefined value	Description	Whether supports runtime dynamic modifications
system_memory_high_watermark_ratio	0.8	Specifies the trigger threshold of the high- level memory alarm mechanism. If the system memory usage is higher than this value, an alarm mechanism will be triggered, and NebulaGraph will stop querying. This parameter is not predefined in the initial configuration files.	Yes

- 435/1069 - 2023 Vesoft Inc.

### **Audit configurations**



The audit log is only available in the Enterprise Edition.

For more information about audit log, see Audit log.

### Metrics configurations

| Name | Predefined value | Description | Whether supports runtime dynamic modifications | | - | - | - | | enable\_space\_level\_metrics | false | Enable or disable space-level metrics. Such metric names contain the name of the graph space that it monitors, for example, query\_latency\_us{space=basketballplayer}.avg.3600 . You can view the supported metrics with the curl command. For more information, see Query NebulaGraph metrics. | No|

### Session configurations

Name	Predefined value	Description	Whether supports runtime dynamic modifications
max_sessions_per_ip_per_user	300	The maximum number of active sessions that can be created from a single IP adddress for a single user.	No

### **Experimental configurations**

Name	Predefined value	Description	Whether supports runtime dynamic modifications
enable_experimental_feature	false	Specifies the experimental feature. Optional values are true and false.	No
enable_data_balance	true	Whether to enable the BALANCE DATA feature. Only works when enable_experimental_feature is true.	No

- 436/1069 - 2023 Vesoft Inc.

# Black box configurations



The Nebula-BBox configurations are for the Enterprise Edition only.

Name	Predefined value	Description	Whether supports runtime dynamic modifications
ng_black_box_switch	true	Whether to enable the Nebula-BBox feature.	No
ng_black_box_home	black_box	The name of the directory to store Nebula-BBox file data.	No
ng_black_box_dump_period_seconds	5	The time interval for Nebula-BBox to collect metric data. Unit: Second.	No
ng_black_box_file_lifetime_seconds	1800	Storage time for Nebula-BBox files generated after collecting metric data. Unit: Second.	Yes

# Memory tracker configurations

Name	Predefined value	Description	Whether supports runtime dynamic modifications
memory_tracker_limit_ratio	0.8	The percentage of free memory. When the free memory is lower than this value, NebulaGraph stops accepting queries. Calculated as follows:  Free memory / (Total memory - Reserved memory)  Note: For clusters with a mixed-used environment, the value of memory_tracker_limit_ratio should be set to a lower value. For example, when Graphd is expected to occupy only 50% of memory, the value can be set to less than 0.5.	Yes
memory_tracker_untracked_reserved_memory_mb	50	The reserved memory that is not tracked by the memory tracker. Unit: MB.	Yes
memory_tracker_detail_log	false	Whether to enable the memory tracker log. When the value is true, the memory tracker log is generated.	Yes
memory_tracker_detail_log_interval_ms	60000	The time interval for generating the memory tracker log. Unit: Millisecond.  memory_tracker_detail_log is true when this parameter takes effect.	Yes
memory_purge_enabled	true	Whether to enable the memory purge feature. When the value is true, the memory purge feature is enabled.	Yes
memory_purge_interval_seconds	10	The time interval for the memory purge feature to purge memory. Unit: Second.  This parameter only takes effect if memory_purge_enabled is set to true.	Yes

Last update: February 19, 2024

- 438/1069 - 2023 Vesoft Inc.

### 6.1.4 Storage Service configurations

NebulaGraph provides two initial configuration files for the Storage Service, nebula-storaged.conf.default and nebula-storaged.conf.production. Users can use them in different scenarios conveniently. The default file path is  $\protect\pr$ 

# Caution

- It is not recommended to modify the value of local\_config to false. If modified, the NebulaGraph service will first read the cached configurations, which may cause configuration inconsistencies between clusters and cause unknown risks.
- It is not recommended to modify the configurations that are not introduced in this topic, unless you are familiar with the source code and fully understand the function of configurations.

#### How to use the configuration files

To use the initial configuration file, choose one of the above two files and delete the suffix default or production from the initial configuration file for the Meta Service to apply the configurations defined in it.

#### About parameter values

If a parameter is not set in the configuration file, NebulaGraph uses the default value. Not all parameters are predefined. And the predefined parameters in the two initial configuration files are different. This topic uses the parameters in <code>nebula-metad.conf.default</code>. For parameters that are not included in <code>nebula-metad.conf.default</code>, see <code>nebula-storaged.conf.production</code>.



Some parameter values in the configuration file can be dynamically modified during runtime. We label these parameters as **Yes** that supports runtime dynamic modification in this article. When the <code>local\_config</code> value is set to <code>true</code>, the dynamically modified configuration is not persisted, and the configuration will be restored to the initial configuration after the service is restarted. For more information, see <code>Modify configurations</code>.



The configurations of the Raft Listener and the Storage service are different. For details, see Deploy Raft listener.

For all parameters and their current values, see Configurations.

- 439/1069 - 2023 Vesoft Inc.

## **Basics configurations**

Name	Predefined value	Description	Whether supports runtime dynamic modifications
daemonize	true	When set to true, the process is a daemon process.	No
pid_file	pids/nebula- storaged.pid	The file that records the process ID.	No
timezone_name	UTC+00:00:00	Specifies the NebulaGraph time zone. This parameter is not predefined in the initial configuration files, if you need to use this parameter, add it manually. For the format of the parameter value, see Specifying the Time Zone with TZ. For example,timezone_name=UTC+08:00 represents the GMT+8 time zone.	No
local_config	true	When set to $\mbox{ true}$ , the process gets configurations from the configuration files.	No



- While inserting property values of time types, NebulaGraph transforms time types (except TIMESTAMP) to the corresponding UTC according to the time zone specified by timezone\_name. The time-type values returned by nGQL queries are all UTC.
- timezone\_name is only used to transform the data stored in NebulaGraph. Other time-related data of the NebulaGraph processes still uses the default time zone of the host, such as the log printing time.

- 440/1069 - 2023 Vesoft Inc.

# Logging configurations

Name	Predefined value	Description	Whether supports runtime dynamic modifications
log_dir	logs	The directory that stores the Meta Service log. It is recommended to put logs on a different hard disk from the data.	No
minloglevel	0	Specifies the minimum level of the log. That is, log messages at or above this level. Optional values are 0 (INFO), 1 (WARNING), 2 (ERROR), 3 (FATAL). It is recommended to set it to 0 during debugging and 1 in a production environment. If it is set to 4, NebulaGraph will not print any logs.	Yes
v	0	Specifies the detailed level of VLOG. That is, log all VLOG messages less or equal to the level. Optional values are 0, 1, 2, 3, 4, 5. The VLOG macro provided by glog allows users to define their own numeric logging levels and control verbose messages that are logged with the parameter v. For details, see Verbose Logging.	Yes
logbufsecs	0	Specifies the maximum time to buffer the logs. If there is a timeout, it will output the buffered log to the log file. 0 means real-time output. This configuration is measured in seconds.	No
redirect_stdout	true	When set to true, the process redirects the stdout and stderr to separate output files.	No
stdout_log_file	graphd- stdout.log	Specifies the filename for the stdout log.	No
stderr_log_file	graphd- stderr.log	Specifies the filename for the stderr log.	No
stderrthreshold	3	Specifies the minloglevel to be copied to the stderr log.	No
timestamp_in_logfile_name	true	Specifies if the log file name contains a timestamp. true indicates yes, false indicates no.	No

- 441/1069 - 2023 Vesoft Inc.

# **Networking configurations**

Name	Predefined value	Description	Whether supports runtime dynamic modifications
meta_server_addrs	127.0.0.1:9559	Specifies the IP addresses and ports of all Meta Services. Multiple addresses are separated with commas.	No
local_ip	127.0.0.1	Specifies the local IP for the Storage Service. The local IP address is used to identify the nebulastoraged process. If it is a distributed cluster or requires remote access, modify it to the corresponding address.	No
port	9779	Specifies RPC daemon listening port of the Storage service. The external port for the Meta Service is predefined to 9779. The internal port is predefined to 9777, 9778, and 9780. Nebula Graph uses the internal port for multi-replica interactions.  9777: The port used by the Drainer service, which is only exposed in the Enterprise Edition cluster. 9778: The port used by the Admin service, which receives Meta commands for Storage. 9780: The port used for Raft communication.	No
ws_ip	0.0.0.0	Specifies the IP address for the HTTP service.	No
ws_http_port	19779	Specifies the port for the HTTP service.	No
heartbeat_interval_secs	10	Specifies the default heartbeat interval. Make sure the heartbeat_interval_secs values for all services are the same, otherwise NebulaGraph <b>CANNOT</b> work normally. This configuration is measured in seconds.	Yes



The real IP address must be used in the configuration file. Otherwise, 127.0.0.1/0.0.0.0 cannot be parsed correctly in some cases.

## Raft configurations

Name	Predefined value	Description	Whether supports runtime dynamic modifications
raft_heartbeat_interval_secs	30	Specifies the time to expire the Raft election. The configuration is measured in seconds.	Yes
raft_rpc_timeout_ms	500	Specifies the time to expire the Raft RPC. The configuration is measured in milliseconds.	Yes
wal_ttl	14400	Specifies the lifetime of the RAFT WAL.  The configuration is measured in seconds.	Yes

- 442/1069 - 2023 Vesoft Inc.

# Disk configurations

Name	Predefined value	Description
data_path	data/storage	Specifies the data storage path. Multiple paths are separated with commas.  In NebulaGraph Community Edition, one RocksDB instance corresponds to one path.  In NebulaGraph Enterprise Edition, one RocksDB instance corresponds to one partition.
minimum_reserved_bytes	268435456	Specifies the minimum remaining space of each data storage path. When the value is lower than this standard, the cluster data writing may fail. This configuration is measured in bytes.
rocksdb_batch_size	4096	Specifies the block cache for a batch operation. The configuration is measured in bytes.
rocksdb_block_cache	4	Specifies the block cache for BlockBasedTable. The configuration is measured in megabytes.
disable_page_cache	false	Enables or disables the operating system's page cache for NebulaGraph. By default, the parameter value is fatse and page cache is enabled. If the value is set to true, page cache is disabled and sufficient block cache space must be configured for NebulaGraph.
engine_type	rocksdb	Specifies the engine type.
rocksdb_compression	Lz4	Specifies the compression algorithm for RocksDB. Optional values are no, snappy, $lz4$ , $lz4hc$ , $zlib$ , $bzip2$ , and $zstd$ . This parameter modifies the compression algorithm for each level. If you want to set different compression algorithms for each level, use the parameter rocksdb_compression_per_level .
rocksdb_compression_per_level	\	Specifies the compression algorithm for each level. The priority is higher than <code>rocksdb_compression</code> . For example, <code>no:no:lz4:lz4:snappy:zstd:snappy</code> . You can also not set certain levels of compression algorithms, for example, <code>no:no:lz4:lz4::zstd</code> , level L4 and L6 use the compression algorithm of <code>rocksdb_compression</code> .
enable_rocksdb_statistics	false	When set to false, RocksDB statistics is disabled.
rocksdb_stats_level	kExceptHistogramOrTimers	Specifies the stats level for RocksDB. Optional values are kExceptHistogramOrTimers, kExceptTimers, kExceptDetailedTimers, kExceptTimeForMutex, and kAll.
enable_rocksdb_prefix_filtering	true	When set to true, the prefix bloom filter for RocksDB is enabled. Enabling prefix bloom filter makes the graph traversal faster but occupies more memory.
enable_rocksdb_whole_key_filtering	false	When set to true, the whole key bloom filter for RocksDB is enabled.
rocksdb_filtering_prefix_length	12	Specifies the prefix length for each key. Optional values are $$ 12 $$ and $$ 16 $$ . The configuration is measured in bytes.
<pre>enable_partitioned_index_filter</pre>	false	When set to true, it reduces the amount of memory used by the bloom filter. But in some random-seek situations, it may reduce the read performance. This parameter is not predefined in the initial configuration files, if you need to use this parameter, add it manually.

- 443/1069 - 2023 Vesoft Inc.

# Misc configurations



The configuration snapshot in the following table is different from the snapshot in NebulaGraph. The snapshot here refers to the stock data on the leader when synchronizing Raft.

Name	Predefined value	Description	Whether supports runtime dynamic modifications
query_concurrently	true	Whether to turn on multi-threaded queries. Enabling it can improve the latency performance of individual queries, but it will reduce the overall throughput under high pressure.	Yes
auto_remove_invalid_space	true	After executing DROP SPACE, the specified graph space will be deleted. This parameter sets whether to delete all the data in the specified graph space at the same time. When the value is true, all the data in the specified graph space will be deleted at the same time.	Yes
num_io_threads	16	The number of network I/O threads used to send RPC requests and receive responses.	Yes
num_worker_threads	32	The number of worker threads for one RPC-based Storage service.	Yes
max_concurrent_subtasks	10	The maximum number of concurrent subtasks to be executed by the task manager.	Yes
snapshot_part_rate_limit	10485760	The rate limit when the Raft leader synchronizes the stock data with other members of the Raft group. Unit: bytes/s.	Yes
snapshot_batch_size	1048576	The amount of data sent in each batch when the Raft leader synchronizes the stock data with other members of the Raft group. Unit: bytes.	Yes
rebuild_index_part_rate_limit	4194304	The rate limit when the Raft leader synchronizes the index data rate with other members of the Raft group during the index rebuilding process. Unit: bytes/s.	Yes
rebuild_index_batch_size	1048576	The amount of data sent in each batch when the Raft leader synchronizes the index data with other members of the Raft group during the index rebuilding process. Unit: bytes.	Yes

- 444/1069 - 2023 Vesoft Inc.

### **RocksDB options**

Name	Predefined value	Description	Whether supports runtime dynamic modifications
rocksdb_db_options	()	Specifies the RocksDB database options.	Yes
rocksdb_column_family_options	<pre>{"write_buffer_size":"67108864",  "max_write_buffer_number":"4",  "max_bytes_for_level_base":"268435456"}</pre>	Specifies the RocksDB column family options.	Yes
rocksdb_block_based_table_options	{"block_size":"8192"}	Specifies the RocksDB block based table options.	Yes

The format of the RocksDB option is  $\{"-option_name>": "-option_value>"\}$ . Multiple options are separated with commas.

Supported options of rocksdb\_db\_options and rocksdb\_column\_family\_options are listed as follows.

rocksdb\_db\_options

max\_total\_wal\_size
delete\_obsolete\_files\_period\_micros
max\_background\_jobs
stats\_dump\_period\_sec
compaction\_readahead\_size
writable\_file\_max\_buffer\_size
bytes\_per\_sync
wal\_bytes\_per\_sync
delayed\_write\_rate
avoid\_flush\_during\_shutdown
max\_open\_files
stats\_persist\_period\_sec
stats\_history\_buffer\_size
strict\_bytes\_per\_sync
enable\_rocksdb\_prefix\_filtering
enable\_rocksdb\_whole\_key\_filtering
rocksdb\_filtering\_prefix\_length
num\_compaction\_threads
rate\_limit

• rocksdb\_column\_family\_options

write\_buffer\_size
max\_write\_buffer\_number
levelO\_file\_num\_compaction\_trigger
levelO\_slowdown\_writes\_trigger
levelO\_stop\_writes\_trigger
target\_file\_size\_base
target\_file\_size\_multiplier
max\_bytes\_for\_level\_base
max\_bytes\_for\_level\_multiplier
disable\_auto\_compactions

For more information, see RocksDB official documentation.

# Black box configurations



The Nebula-BBox configurations are for the Enterprise Edition only.

Name	Predefined value	Description	Whether supports runtime dynamic modifications
ng_black_box_switch	true	Whether to enable the Nebula-BBox feature.	No
ng_black_box_home	black_box	The name of the directory to store Nebula-BBox file data.	No
ng_black_box_dump_period_seconds	5	The time interval for Nebula-BBox to collect metric data. Unit: Second.	No
ng_black_box_file_lifetime_seconds	1800	Storage time for Nebula-BBox files generated after collecting metric data. Unit: Second.	Yes

# Memory tracker configurations

Name	Predefined value	Description	Whether supports runtime dynamic modifications
memory_tracker_limit_ratio	0.8	The value of this parameter can be set to (0, 1], 2, and 3. (0, 1]: The percentage of free memory. When the free memory is lower than this value, NebulaGraph stops accepting queries.  Calculated as follows:  Free memory / (Total memory - Reserved memory)  Note: For clusters with a mixed-used environment, the value of memory_tracker_limit_ratio should be set to a lower value. For example, when Graphd is expected to occupy only 50% of memory, the value can be set to less than 0.5.  2: Dynamic Self Adaptive mode.  MemoryTracker dynamically adjusts the available memory based on the system's current available memory.  Note: This feature is experimental. As memory usage cannot be monitored in real time in dynamic adaptive mode, an OOM error may still occur to handle large memory allocations.  3: Disable MemoryTracker.  MemoryTracker only logs memory usage and does not interfere with executions even if the limit is exceeded.	Yes
memory_tracker_untracked_reserved_memory_mb	50	The reserved memory that is not tracked by the memory tracker. Unit: MB.	Yes
memory_tracker_detail_log	false	Whether to enable the memory tracker log. When the value is true, the memory tracker log is generated.	Yes
memory_tracker_detail_log_interval_ms	60000	The time interval for generating the memory tracker log. Unit: Millisecond.  memory_tracker_detail_log is true when this parameter takes effect.	Yes
memory_purge_enabled	true	Whether to enable the memory purge feature. When the value is true, the memory purge feature is enabled.	Yes
memory_purge_interval_seconds	10	The time interval for the memory purge feature to purge memory. Unit: Second. This parameter only takes effect if memory_purge_enabled is set to true.	Yes

- 447/1069 - 2023 Vesoft Inc.

### For super-Large vertices

When the query starting from each vertex gets an edge, truncate it directly to avoid too many neighboring edges on the superlarge vertex, because a single query occupies too much hard disk and memory. Or you can truncate a certain number of edges specified in the <code>Max\_edge\_returned\_per\_vertex</code> parameter. Excess edges will not be returned. This parameter applies to all spaces.

Property name	Default value	Description	Whether supports runtime dynamic modifications
max_edge_returned_per_vertex	2147483647	Specifies the maximum number of edges returned for each dense vertex. Excess edges are truncated and not returned. This parameter is not predefined in the initial configuration files, if you need to use this parameter, add it manually.	No

### Storage configurations for large dataset



One graph space takes up at least about 300 MB of memory.

When you have a large dataset (in the RocksDB directory) and your memory is tight, we suggest that you set the <code>enable\_partitioned\_index\_filter</code> parameter to <code>true</code>. The performance is affected because RocksDB indexes are cached.

Last update: February 19, 2024

- 448/1069 - 2023 Vesoft Inc.

### 6.1.5 Kernel configurations

This topic introduces the Kernel configurations in Nebula Graph.

#### Resource control

**ULIMIT PRECAUTIONS** 

The utimit command specifies the resource threshold for the current shell session. The precautions are as follows:

- The changes made by utimit only take effect for the current session or child process.
- The resource threshold (soft threshold) cannot exceed the hard threshold.
- Common users cannot use commands to adjust the hard threshold, even with sudo.
- To modify the system level or adjust the hard threshold, edit the file /etc/security/limits.conf . This method requires re-login to take effect.

ULIMIT -C

ulimit -c limits the size of the core dumps. We recommend that you set it to unlimited. The command is:

```
ulimit -c unlimited
```

ULIMIT -N

utimit -n limits the number of open files. We recommend that you set it to more than 100,000. For example:

```
ulimit -n 130000
```

#### Memory

VM.SWAPPINESS

vm.swappiness specifies the percentage of the available memory before starting swap. The greater the value, the more likely the swap occurs. We recommend that you set it to 0. When set to 0, the page cache is removed first. Note that when vm.swappiness is 0, it does not mean that there is no swap.

VM.MIN\_FREE\_KBYTES

vm.min\_free\_kbytes specifies the minimum number of kilobytes available kept by Linux VM. If you have a large system memory, we recommend that you increase this value. For example, if your physical memory 128GB, set it to 5GB. If the value is not big enough, the system cannot apply for enough continuous physical memory.

VM.MAX MAP COUNT

vm.max\_map\_count limits the maximum number of vma (virtual memory area) for a process. The default value is 65530. It is enough for most applications. If your memory application fails because the memory consumption is large, increase the vm.max\_map\_count value.

VM.DIRTY \*

These values control the dirty data cache for the system. For write-intensive scenarios, you can make adjustments based on your needs (throughput priority or delay priority). We recommend that you use the system default value.

TRANSPARENT HUGE PAGE

For better delay performance, you must run the following commands to disable the transparent huge pages (THP).

```
root# echo never > /sys/kernel/mm/transparent_hugepage/enabled
root# echo never > /sys/kernel/mm/transparent_hugepage/defrag
root# swapoff -a && swapon -a
```

To prevent THP from being enabled again after the system restarts, you can modify the GRUB configuration file or /etc/rc.local to disable THP automatically upon system startup.

#### Networking

NET.IPV4.TCP\_SLOW\_START\_AFTER\_IDLE

The default value of net.ipv4.tcp\_slow\_start\_after\_idle is 1. If set, the congestion window is timed out after an idle period. We recommend that you set it to 0, especially for long fat scenarios (high latency and large bandwidth).

NET.CORE.SOMAXCONN

net.core.somaxconn specifies the maximum number of connection queues listened by the socket. The default value is 128. For scenarios with a large number of burst connections, we recommend that you set it to greater than 1024.

NET.IPV4.TCP MAX SYN BACKLOG

 $net.ipv4.tcp\_max\_syn\_backlog$  specifies the maximum number of TCP connections in the SYN\_RECV (semi-connected) state. The setting rule for this parameter is the same as that of net.core.somaxconn.

NET.CORE.NETDEV\_MAX\_BACKLOG

net.core.netdev\_max\_backlog specifies the maximum number of packets. The default value is 1000. We recommend that you increase it to greater than 10,000, especially for 10G network adapters.

NET.IPV4.TCP KEEPALIVE \*

These values keep parameters alive for TCP connections. For applications that use a 4-layer transparent load balancer, if the idle connection is disconnected unexpectedly, decrease the values of tcp\_keepalive\_time and tcp\_keepalive\_intvl.

NET.IPV4.TCP RMEM/WMEM

net.ipv4.tcp\_wmem/rmem specifies the minimum, default, and maximum size of the buffer pool sent/received by the TCP socket. For long fat links, we recommend that you increase the default value to bandwidth (GB)  $^*$  RTT (ms).

SCHEDULER

For SSD devices, we recommend that you set scheduler to noop or none. The path is /sys/block/DEV\_NAME/queue/scheduler.

## Other parameters

KERNEL.CORE\_PATTERN

we recommend that you set it to core and set kernel.core\_uses\_pid to 1.

#### **Modify parameters**

SYSCTL

sysctl <conf\_name>

Checks the current parameter value.

sysctl -w <conf\_name>=<value>

Modifies the parameter value. The modification takes effect immediately. The original value is restored after restarting.

sysctl -p [<file\_path>]

 $Loads\ Linux\ parameter\ values\ from\ the\ specified\ configuration\ file.\ The\ default\ path\ is\ \ /etc/sysctl.conf.$ 

PRLIMIT

The prlimit command gets and sets process resource limits. You can modify the hard threshold by using it and the sudo command. For example, prlimit --nofile = 130000 --pid = \$\$ adjusts the maximum number of open files permitted by the current process to 14000. And the modification takes effect immediately. Note that this command is only available in RedHat 7u or higher versions.

Last update: February 19, 2024

## 6.2 Log management

### 6.2.1 Runtime logs

Runtime logs are provided for DBAs and developers to locate faults when the system fails.

**NebulaGraph** uses glog to print runtime logs, uses gflags to control the severity level of the log, and provides an HTTP interface to dynamically change the log level at runtime to facilitate tracking.

### Log directory

The default runtime log directory is /usr/local/nebula/logs/.

If the log directory is deleted while NebulaGraph is running, the log would not continue to be printed. However, this operation will not affect the services. To recover the logs, restart the services.

#### Parameter descriptions

- minloglevel: Specifies the minimum level of the log. That is, no logs below this level will be printed. Optional values are 0 (INFO), 1 (WARNING), 2 (ERROR), 3 (FATAL). It is recommended to set it to 0 during debugging and 1 in a production environment. If it is set to 4, NebulaGraph will not print any logs.
- v: Specifies the detailed level of the log. The larger the value, the more detailed the log is. Optional values are 0, 1, 2, 3.

The default severity level for the metad, graphd, and storaged logs can be found in their respective configuration files. The default path is /usr/local/nebula/etc/.

### Check the severity level

Check all the flag values (log values included) of the current gflags with the following command.

\$ curl <ws\_ip>:<ws\_port>/flags

Parameter	Description
ws_ip	The IP address for the HTTP service, which can be found in the configuration files above. The default value is $127.0.0.1$ .
ws_port	The port for the HTTP service, which can be found in the configuration files above. The default values are 19559 (Meta), 19669 (Graph), and 19779 (Storage) respectively.

### Examples are as follows:

 $\bullet$  Check the current  $\mbox{\ minloglevel}$  in the Meta service:

```
$ curl 127.0.0.1:19559/flags | grep 'minloglevel'
```

• Check the current v in the Storage service:

```
$ curl 127.0.0.1:19779/flags | grep -w 'v'
```

## Change the severity level

Change the severity level of the log with the following command.

- 451/1069 - 2023 Vesoft Inc.

\$ curl -X PUT -H "Content-Type: application/json" -d '{" <key>":<value>[,"<key>":<value>]}' "<ms_ip>:<ms_port>/flags"</ms_port></ms_ip></value></key></value></key>

Parameter	Description
key	The type of the log to be changed. For optional values, see Parameter descriptions.
value	The level of the log. For optional values, see Parameter descriptions.
ws_ip	The IP address for the HTTP service, which can be found in the configuration files above. The default value is $127.0.0.1$ .
ws_port	The port for the HTTP service, which can be found in the configuration files above. The default values are 19559 (Meta), 19669 (Graph), and 19779 (Storage) respectively.

### Examples are as follows:

```
$ curl -X PUT -H "Content-Type: application/json" -d '{"minloglevel":0,"v":3}' "127.0.0.1:19779/flags" # storaged
$ curl -X PUT -H "Content-Type: application/json" -d '{"minloglevel":0,"v":3}' "127.0.0.1:19669/flags" # graphd
$ curl -X PUT -H "Content-Type: application/json" -d '{"minloglevel":0,"v":3}' "127.0.0.1:19559/flags" # metad
```

If the log level is changed while NebulaGraph is running, it will be restored to the level set in the configuration file after restarting the service. To permanently modify it, see Configuration files.

### RocksDB runtime logs

 $RocksDB\ runtime\ logs\ are\ usually\ used\ to\ debug\ RocksDB\ parameters\ and\ stored\ in\ \ /usr/local/nebula/data/storage/nebula/$id/data/LOG\ .$  \$id is the ID of the example.

Last update: February 19, 2024

# 6.2.2 Audit logs

The NebulaGraph audit logs store all operations received by graph service in categories, then provide the logs for users to track specific types of operations as needed.



Only available for the NebulaGraph Enterprise Edition.

# Log categories

Category	Statement	Description
login		Logs the information when the client tries to connect to graph service.
exit	-	Logs the information when the client disconnect from graph service.
ddl	CREATE SPACE, DROP SPACE, CREATE TAG, DROP TAG, ALTER TAG, DELETE TAG, CREATE EDGE, DROP EDGE, ALTER EDGE, CREATE INDEX, DROP INDEX, CREATE FULLTEXT INDEX, DROP FULLTEXT INDEX	Logs the information about DDL statements.
dql	MATCH, LOOKUP, GO, FETCH, GET SUBGRAPH, FIND PATH, UNWIND, GROUP BY, ORDER BY, YIELD, LIMIT, RETURN, REBUILD INDEX, REBUILD FULLTEXT INDEX	Logs the information about DQL statements.
dml	INSERT VERTEX, DELETE VERTEX, UPDATE VERTEX, UPSERT VERTEX, INSERT EDGE, DELETE EDGE, UPDATE EDGE, UPSERT EDGE	Logs the information about DML statements.
dcl	CREATE USER, GRANT ROLE, REVOKE ROLE, CHANGE PASSWORD, ALTER USER, DROP USER, CREATE SNAPSHOT, DROP SNAPSHOT, ADD LISTENER, REMOVE LISTENER, BALANCE, SUBMIT JOB, STOP JOB, RECOVER JOB, ADD DRAINER, REMOVE DRAINER, SIGN IN DRAINER SERVICE, SIGN OUT DRAINER SERVICE, DOWNLOAD HDFS, INGEST	Logs the information about DCL statements.
util	SHOW HOSTS, SHOW USERS, SHOW ROLES, SHOW SNAPSHOTS, SHOW SPACES, SHOW PARTS, SHOW TAGS, SHOW EDGES, SHOW INDEXES, SHOW CREATE SPACE, SHOW CREATE TAG/EDGE, SHOW CREATE INDEXES, SHOW INDEXES STATUS, SHOW LISTENER, SHOW TEXT SEARCH CLIENTS, SHOW DRAINER CLIENTS, SHOW FULLTEXT INDEXES, SHOW CONFIGS, SHOW CHARSET, SHOW COLLATION, SHOW STATS, SHOW SESSIONS, SHOW META LEADER, SHOW DRAINERS, SHOW QUERIES, SHOW JOBS, DESCRIBE INDEX, DESCRIBE EDGE, DESCRIBE TAG, DESCRIBE SPACE, DESCRIBE USER, USE SPACE, SIGN IN TEXT SERVICE, SIGN OUT TEXT SERVICE, EXPLAIN, PROFILE, KILL QUERY	Logs the information about util statements.
unknown	-	Logs the information about unrecognized statements.

## Configure audit logs



After modifying the configuration, you need to restart the graph service to take effect.

Parameter descriptions are as follows:

Parameter	Predefined value	Description
enable_audit	false	Whether or not to enable audit logs.
audit_log_handler	file	Specifies the place where the audit logs will be written. Optional values are file (local file) and es (Elasticsearch). The supported Elasticsearch versions are $7.x$ and $8.x$ .
audit_log_file	./logs/audit/ audit.log	Takes effect only when $\mbox{audit\_log\_handler=file}$ . The path for storing audit logs. The value can be absolute or relative.
audit_log_strategy	synchronous	Sets the method to synchronize audit logs. Takes effect only when audit_log_handler=file. Optional values are asynchronous and synchronous. When asynchronous, log events are cached in memory and do not block the main thread, but may result in missing logs due to insufficient cache. When synchronous, log events are refreshed and synchronized to the file each time.
audit_log_max_buffer_size	1048576	Take effect only when <code>audit_log_handler=file</code> and <code>audit_log_strategy=asynchronous</code> . The size of the memory buffer used for logging. Unit: bytes.
audit_log_format	xmL	Takes effect only when $\mbox{audit\_log\_handler=file}$ . The format of the the audit logs. Optional values are $\mbox{xml}$ , $\mbox{json}$ and $\mbox{csv}$ .
audit_log_es_address	-	Takes effect only when <code>audit_log_handler=es</code> . The address of Elasticsearch server. The format is <code>IP1:port1, IP2:port2,</code>
audit_log_es_user	-	Takes effect only when <code>audit_log_handler=es</code> . The user name of the Elasticsearch.
audit_log_es_password	-	Takes effect only when $\mbox{audit\_log\_handler=es}$ . The user password of the Elasticsearch.
audit_log_es_batch_size	1000	Takes effect only when $\mbox{audit\_log\_handler=es}$ . The number of logs sent to Elasticsearch at one time.
audit_log_exclude_spaces	-	The list of spaces for not tracking. Multiple graph spaces are separated by commas.
audit_log_categories	login,exit	The list of log categories for tracking. Multiple categories are separated by commas.

## Audit logs format

The fields of audit logs are the same for different handlers and formats. For example, when the audit logs are stored in the default path /usr/local/nebula/logs/audit/audit.log and in the format of XML, the fields in the audit logs are described as follows:



If the audit log directory is deleted while NebulaGraph is running, the log would not continue to be printed and this operation will not affect the services. To recover the logs, you should restart the services.

<AUDIT\_RECORD
CATEGORY="util"
TIMESTAMP="2022-04-07 02:31:38"
TERNITAL=""
CONNECTION\_ID="1649298693144580"
CONNECTION\_STATUS="0"
CONNECTION\_MESSAGE=""</pre>

```
USER="root"

CLIENT_HOST="127.0.0.1"

HOST="192.168.8.111"

SPACE=""

QUERY="use basketbaltplayer1"

QUERY_strius="-1005"

QUERY_MESSAGE="SpaceNotFound: "

/>
<AUDIT_RECORD

CATEGORY="util"

TIMESTAMP="2022-04-07 02:31:39"

TERMINAL=""

CONNECTION_STATUS="0"

CONNECTION_STATUS="0"

CLIENT_HOST="127.0.0.1"

HOST="192.168.8.111"

SPACE=""

QUERY_suse basketbaltplayer"

QUERY_suse basketbaltplayer"

QUERY_strius="0"

QUERY_strius="0"

QUERY_MESSAGE=""

VERSE=""

QUERY_STATUS="0"

QUERY_STATUS="0"

QUERY_STATUS="0"

QUERY_MESSAGE=""
```

Field	Description
CATEGORY	The category of the audit logs.
TIMESTAMP	The generation time of the audit logs.
TERMINAL	The reserved field.
CONNECTION_ID	The session ID of the connection.
CONNECTION_STATUS	The status of the connection. $0$ indicates success, and other numbers indicate different error messages.
CONNECTION_MESSAGE	An error message is displayed when the connection fails.
USER	The user name of the NebulaGraph connection.
CLIENT_HOST	The IP address of the client.
HOST	The IP address of the host.
SPACE	The graph space where you perform queries.
QUERY	The query statement.
QUERY_STATUS	The status of the query. $\ 0\ $ indicates success, and other numbers indicate different error messages.
QUERY_MESSAGE	An error message is displayed when the query fails.

Last update: February 19, 2024

# 7. Monitor and metrics

# 7.1 Query NebulaGraph metrics

NebulaGraph supports querying the monitoring metrics through HTTP ports.

### 7.1.1 Metrics structure

Each metric of NebulaGraph consists of three fields: name, type, and time range. The fields are separated by periods, for example, <code>num\_queries.sum.600</code>. Different NebulaGraph services (Graph, Storage, or Meta) support different metrics. The detailed description is as follows.

Field	Example	Description
Metric name	num_queries	Indicates the function of the metric.
Metric type	sum	Indicates how the metrics are collected. Supported types are SUM, AVG, RATE, and the P-th sample quantiles such as P75, P95, P99, and P99.9.
Time range	600	The time range in seconds for the metric collection. Supported values are $5$ , $60$ , $600$ , and $3600$ , representing the last $5$ seconds, $1$ minute, $10$ minutes, and $1$ hour.

#### Space-level metrics

The Graph service supports a set of space-level metrics that record the information of different graph spaces separately.

To enable space-level metrics, set the value of <code>enable\_space\_level\_metrics</code> to true in the Graph service configuration file before starting NebulaGraph. For details about how to modify the configuration, see <a href="Configuration Management">Configuration Management</a>.



Space-level metrics can be queried only by querying all metrics. For example, run <code>curl -6 "http://192.168.8.40:19559/stats"</code> to show all metrics. The returned result contains the graph space name in the form of '{space=space\_name}', such as num\_active\_queries{space=basketballplayer}.sum.5=0.

## 7.1.2 Query metrics over HTTP

#### **Syntax**

curl -G "http://<ip>:<port>/stats?stats=<metric\_name\_list> [&format=json]"

Parameter	Description
ip	The IP address of the server. You can find it in the configuration file in the installation directory.
port	The HTTP port of the server. You can find it in the configuration file in the installation directory. The default ports are 19559 (Meta), 19669 (Graph), and 19779 (Storage).
metric_name_list	The metrics names. Multiple metrics are separated by commas (,).
&format=json	Optional. Returns the result in the JSON format.

- 456/1069 - 2023 Vesoft Inc.



If NebulaGraph is deployed with Docker Compose, run docker-compose ps to check the ports that are mapped from the service ports inside of the container and then query through them.

#### **Examples**

· Query a single metric

Query the query number in the last 10 minutes in the Graph Service.

```
$ curl -G "http://192.168.8.40:19669/stats?stats=num_queries.sum.600"
num_queries.sum.600=400
```

• Query multiple metrics

Query the following metrics together:

- The average heartbeat latency in the last 1 minute.
- The average latency of the slowest 1% heartbeats, i.e., the P99 heartbeats, in the last 10 minutes.

```
$ curl -6 "http://192.168.8.40:19559/stats?stats=heartbeat_latency_us.avg.60,heartbeat_latency_us.p99.600"
heartbeat_latency_us.avg.60=281
heartbeat_latency_us.p99.600=985
```

• Return a JSON result.

Query the number of new vertices in the Storage Service in the last 10 minutes and return the result in the JSON format.

```
$ curl -G "http://192.168.8.40:19779/stats?stats=num_add_vertices.sum.600&format=json"
[{"value":1,"name":"num_add_vertices.sum.600"}]
```

• Query all metrics in a service.

If no metric is specified in the query, NebulaGraph returns all metrics in the service.

```
$ curl -G "http://192.168.8.40:19559/stats"
heartbeat_latency_us.avg.5=304
heartbeat_latency_us.avg.60=308
heartbeat_latency_us.avg.600=299
heartbeat_latency_us.avg.3600=285
heartbeat\_latency\_us.p75.5 = \!\! 652
heartbeat_latency_us.p75.60=669
heartbeat_latency_us.p75.600=651
heartbeat_latency_us.p75.3600=642
heartbeat_latency_us.p95.5=930
heartbeat_latency_us.p95.60=963
heartbeat_latency_us.p95.600=933
heartbeat_latency_us.p95.3600=929
heartbeat_latency_us.p99.5=986
heartbeat_latency_us.p99.60=1409
heartbeat_latency_us.p99.600=989
heartbeat_latency_us.p99.3600=986
num_heartbeats.rate.5=0
num_heartbeats.rate.60=0
num_heartbeats.rate.600=0
num_heartbeats.rate.3600=0
num_heartbeats.sum.5=2
num_heartbeats.sum.60=40
num_heartbeats.sum.600=394
num_heartbeats.sum.3600=2364
```

# 7.1.3 Metric description

# Graph

Parameter	Description
num_active_queries	The number of changes in the number of active queries.  Formula: The number of started queries minus the number of finished queries within a specified time.
num_active_sessions	The number of changes in the number of active sessions. Formula: The number of logged in sessions minus the number of logged out sessions within a specified time. For example, when querying <code>num_active_sessions.sum.5</code> , if there were 10 sessions logged in and 30 sessions logged out in the last 5 seconds, the value of this metric is -20 (10-30).
num_aggregate_executors	The number of executions for the Aggregation operator.
num_auth_failed_sessions_bad_username_password	The number of sessions where authentication failed due to incorrect username and password.
num_auth_failed_sessions_out_of_max_allowed	The number of sessions that failed to authenticate logins because the value of the parameter $FLAG_OUT_OF_MAX_ALLOWED_CONNECTIONS$ was exceeded.
num_auth_failed_sessions	The number of sessions in which login authentication failed.
num_indexscan_executors	The number of executions for index scan operators.
num_killed_queries	The number of killed queries.
num_opened_sessions	The number of sessions connected to the server.
num_queries	The number of queries.
num_query_errors_leader_changes	The number of the raft leader changes due to query errors.
num_query_errors	The number of query errors.
num_reclaimed_expired_sessions	The number of expired sessions actively reclaimed by the server.
num_rpc_sent_to_metad_failed	The number of failed RPC requests that the Graphd service sent to the Metad service.
num_rpc_sent_to_metad	The number of RPC requests that the Graphd service sent to the Metad service.
num_rpc_sent_to_storaged_failed	The number of failed RPC requests that the Graphd service sent to the Storaged service.
num_rpc_sent_to_storaged	The number of RPC requests that the Graphd service sent to the Storaged service.
num_sentences	The number of statements received by the Graphd service.
num_slow_queries	The number of slow queries.
num_sort_executors	The number of executions for the Sort operator.
optimizer_latency_us	The latency of executing optimizer statements.
query_latency_us	The latency of queries.
slow_query_latency_us	The latency of slow queries.
num_queries_hit_memory_watermark	The number of queries reached the memory watermark.

- 458/1069 - 2023 Vesoft Inc.

# Meta

Parameter	Description
commit_log_latency_us	The latency of committing logs in Raft.
commit_snapshot_latency_us	The latency of committing snapshots in Raft.
heartbeat_latency_us	The latency of heartbeats.
num_heartbeats	The number of heartbeats.
num_raft_votes	The number of votes in Raft.
transfer_leader_latency_us	The latency of transferring the raft leader.
num_agent_heartbeats	The number of heartbeats for the AgentHBProcessor.
agent_heartbeat_latency_us	The latency of the AgentHBProcessor.
replicate_log_latency_us	The latency of replicating the log record to most nodes by Raft.
num_send_snapshot	The number of times that Raft sends snapshots to other nodes.
append_log_latency_us	The latency of replicating the log record to a single node by Raft.
append_wal_latency_us	The Raft write latency for a single WAL.
num_grant_votes	The number of times that Raft votes for other nodes.
num_start_elect	The number of times that Raft starts an election.

- 459/1069 - 2023 Vesoft Inc.

# Storage

Parameter	Description
add_edges_latency_us	The latency of adding edges.
add_vertices_latency_us	The latency of adding vertices.
commit_log_latency_us	The latency of committing logs in Raft.
commit_snapshot_latency_us	The latency of committing snapshots in Raft.
delete_edges_latency_us	The latency of deleting edges.
delete_vertices_latency_us	The latency of deleting vertices.
get_neighbors_latency_us	The latency of querying neighbor vertices.
<pre>get_dst_by_src_latency_us</pre>	The latency of querying the destination vertex by the source vertex.
num_get_prop	The number of executions for the GetPropProcessor.
num_get_neighbors_errors	The number of execution errors for the GetNeighborsProcessor.
num_get_dst_by_src_errors	The number of execution errors for the GetDstBySrcProcessor.
get_prop_latency_us	The latency of executions for the GetPropProcessor.
num_edges_deleted	The number of deleted edges.
num_edges_inserted	The number of inserted edges.
num_raft_votes	The number of votes in Raft.
num_rpc_sent_to_metad_failed	The number of failed RPC requests that the Storage service sent to the Meta service.
num_rpc_sent_to_metad	The number of RPC requests that the Storaged service sent to the Metad service.
num_tags_deleted	The number of deleted tags.
num_vertices_deleted	The number of deleted vertices.
num_vertices_inserted	The number of inserted vertices.
transfer_leader_latency_us	The latency of transferring the raft leader.
lookup_latency_us	The latency of executions for the LookupProcessor.
num_lookup_errors	The number of execution errors for the LookupProcessor.
num_scan_vertex	The number of executions for the ScanVertexProcessor.
num_scan_vertex_errors	The number of execution errors for the ScanVertexProcessor.
update_edge_latency_us	The latency of executions for the UpdateEdgeProcessor.
num_update_vertex	The number of executions for the UpdateVertexProcessor.
num_update_vertex_errors	The number of execution errors for the UpdateVertexProcessor.
kv_get_latency_us	The latency of executions for the Getprocessor.
kv_put_latency_us	The latency of executions for the PutProcessor.
kv_remove_latency_us	The latency of executions for the RemoveProcessor.
num_kv_get_errors	The number of execution errors for the GetProcessor.
num_kv_get	The number of executions for the GetProcessor.
num_kv_put_errors	The number of execution errors for the PutProcessor.
num_kv_put	The number of executions for the PutProcessor.

- 461/1069 - 2023 Vesoft Inc.

Parameter	Description
num_kv_remove_errors	The number of execution errors for the RemoveProcessor.
num_kv_remove	The number of executions for the RemoveProcessor.
forward_tranx_latency_us	The latency of transmission.
scan_edge_latency_us	The latency of executions for the ScanEdgeProcessor.
num_scan_edge_errors	The number of execution errors for the ScanEdgeProcessor.
num_scan_edge	The number of executions for the ScanEdgeProcessor.
scan_vertex_latency_us	The latency of executions for the ScanVertexProcessor.
num_add_edges	The number of times that edges are added.
num_add_edges_errors	The number of errors when adding edges.
num_add_vertices	The number of times that vertices are added.
num_start_elect	The number of times that Raft starts an election.
num_add_vertices_errors	The number of errors when adding vertices.
num_delete_vertices_errors	The number of errors when deleting vertices.
append_log_latency_us	The latency of replicating the log record to a single node by Raft.
num_grant_votes	The number of times that Raft votes for other nodes.
replicate_log_latency_us	The latency of replicating the log record to most nodes by Raft.
num_delete_tags	The number of times that tags are deleted.
num_delete_tags_errors	The number of errors when deleting tags.
num_delete_edges	The number of edge deletions.
num_delete_edges_errors	The number of errors when deleting edges
num_send_snapshot	The number of times that snapshots are sent.
update_vertex_latency_us	The latency of executions for the UpdateVertexProcessor.
append_wal_latency_us	The Raft write latency for a single WAL.
num_update_edge	The number of executions for the UpdateEdgeProcessor.
delete_tags_latency_us	The latency of deleting tags.
num_update_edge_errors	The number of execution errors for the UpdateEdgeProcessor.
num_get_neighbors	The number of executions for the GetNeighborsProcessor.
num_get_dst_by_src	The number of executions for the GetDstBySrcProcessor.
num_get_prop_errors	The number of execution errors for the GetPropProcessor.
num_delete_vertices	The number of times that vertices are deleted.
num_lookup	The number of executions for the LookupProcessor.
num_sync_data	The number of times the Storage service synchronizes data from the Drainer.
num_sync_data_errors	The number of errors that occur when the Storage service synchronizes data from the Drainer.
sync_data_latency_us	The latency of the Storage service synchronizing data from the Drainer.

- 462/1069 - 2023 Vesoft Inc.

# **Graph space**



Space-level metrics are created dynamically, so that only when the behavior is triggered in the graph space, the corresponding metric is created and can be queried by the user.

Parameter	Description
num_active_queries	The number of queries currently being executed.
num_queries	The number of queries.
num_sentences	The number of statements received by the Graphd service.
optimizer_latency_us	The latency of executing optimizer statements.
query_latency_us	The latency of queries.
num_slow_queries	The number of slow queries.
num_query_errors	The number of query errors.
num_query_errors_leader_changes	The number of raft leader changes due to query errors.
num_killed_queries	The number of killed queries.
num_aggregate_executors	The number of executions for the Aggregation operator.
num_sort_executors	The number of executions for the Sort operator.
num_indexscan_executors	The number of executions for index scan operators.
num_auth_failed_sessions_bad_username_password	The number of sessions where authentication failed due to incorrect username and password.
num_auth_failed_sessions	The number of sessions in which login authentication failed.
num_opened_sessions	The number of sessions connected to the server.
num_queries_hit_memory_watermark	The number of queries reached the memory watermark.
num_reclaimed_expired_sessions	The number of expired sessions actively reclaimed by the server.
num_rpc_sent_to_metad_failed	The number of failed RPC requests that the Graphd service sent to the Metad service.
num_rpc_sent_to_metad	The number of RPC requests that the Graphd service sent to the Metad service.
num_rpc_sent_to_storaged_failed	The number of failed RPC requests that the Graphd service sent to the Storaged service.
num_rpc_sent_to_storaged	The number of RPC requests that the Graphd service sent to the Storaged service.
slow_query_latency_us	The latency of slow queries.

- 463/1069 - 2023 Vesoft Inc.

# Single process metrics

Graph, Meta, and Storage services all have their own single process metrics.

Parameter	Description
context_switches_total	The number of context switches.
cpu_seconds_total	The CPU usage based on user and system time.
memory_bytes_gauge	The number of bytes of memory used.
open_filedesc_gauge	The number of file descriptors.
read_bytes_total	The number of bytes read.
write_bytes_total	The number of bytes written.

Last update: February 19, 2024

### 7.2 RocksDB statistics

NebulaGraph uses RocksDB as the underlying storage. This topic describes how to collect and show the RocksDB statistics of NebulaGraph.

#### 7.2.1 Enable RocksDB

By default, the function of RocksDB statistics is disabled. To enable RocksDB statistics, you need to:

- 1. Modify the --enable\_rocksdb\_statistics parameter as true in the nebula-storaged.conf file. The default path of the configuration file is /use/local/nebula/etc.
- 2. Restart the service to make the modification valid.

#### 7.2.2 Get RocksDB statistics

Users can use the built-in HTTP service in the storage service to get the following types of statistics. Results in the JSON format are supported.

- All RocksDB statistics.
- Specified RocksDB statistics.

#### 7.2.3 Examples

Use the following command to get all RocksDB statistics:

```
curl -L "http://${storage_ip}:${port}/rocksdb_stats"
```

#### For example:

```
curl -L "http://172.28.2.1:19779/rocksdb_stats"

rocksdb.blobdb.blob.file.bytes.read=0
rocksdb.blobdb.blob.file.bytes.written=0
rocksdb.blobdb.blob.file.bytes.synced=0
...
```

Use the following command to get specified RocksDB statistics:

```
curl -L "http://${storage_ip}:${port}/rocksdb_stats?stats=${stats_name}"
```

For example, use the following command to get the information of rocksdb.bytes.read and rocksdb.block.cache.add.

```
curl -L "http://172.28.2.1:19779/rocksdb_stats?stats=rocksdb.bytes.read,rocksdb.block.cache.add"
rocksdb.block.cache.add=14
rocksdb.bytes.read=1632
```

Use the following command to get specified RocksDB statistics in the JSON format:

```
curl -L "http://${storage_ip}:${port}/rocksdb_stats?stats=${stats_name}&format=json"
```

For example, use the following command to get the information of rocksdb.bytes.read and rocksdb.block.cache.add and return the results in the JSON format.

1

Last update: February 19, 2024

# 7.3 Black-box monitoring

#### 7.3.1 What is black-box monitoring

NebulaGraph black-box monitoring feature is designed to regularly collect and archive data on the operating system and service metrics. When the NebulaGraph service fails, it helps you quickly locate the problem and analyze the cause without a direct network connection.



The black-box monitoring feature is for the NebulaGraph Enterprise Edition only.

#### Note

Black-box monitoring operates as a set of background processes on the server and collects metric data regularly. Currently, only operating system performance metrics are collected (e.g., CPU, Memory, Network IO, and other related metrics). In the future, we will support collecting NebulaGraph service-related metrics. For the description of metrics, see PROC.

#### **Enable black-box monitoring**

The black-box monitoring feature is turned on by default. The black-box directory is created and stored in the NebulaGraph installation directory the first time NebulaGraph is started. Files of collected black-box monitoring data are stored in that directory.

You can disable the black-box monitoring by setting the related parameters in the **Black box configurations** section of the configuration files of all NebulaGraph services. For details about service configurations, see Configurations.

### Black-box monitoring files

DIRECTORY STRUCTURE

In the black\_box directory, the system automatically creates sub-directories. Sub-directories are named with the corresponding process number of each NebulaGraph service running on the current machine. In each sub-directory, by default, a binary file is generated every 5 seconds to record the OS performance metric data during this time. The file name is in the format of black\_box. {timestamp\_id}.log. timestamp\_id is the timestamp when the file is generated.

```
| → 5776 # The service process number | → black_box.1665472900.log # Black-box monitoring file | → black_box.1665472905.log | ... | → 5787 | → black_box.1665728450.log | → black_box.1665728455.log | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ...
```

Each black-box monitoring file has 30 minutes (1800 seconds) of storage by default. Files stored for more than 30 minutes will be automatically deleted.

The generation interval and storage time of black-box monitoring files can be configured in the **Black box configuration** section of the configuration file of each NebulaGraph service. For configuration file details, see Configurations.

VIEWING BLACK-BOX MONITORING FILES

The NebulaGraph Black Box tool allows you to view and analyze data from black-box monitoring binary files, or you can use the tool to convert binary files to CSV files and export them for viewing. For using the NebulaGraph Black Box tool, see Black Box tool Nebula-BBox.

Last update: February 19, 2024

## 7.3.2 Black-box monitoring tool - NebulaGraph Black Box

The black-box monitoring tool NebulaGraph Black Box (short for Nebula-BBox) helps you view black-box monitoring data. This topic introduces how to use Nebula-BBox in Linux.



Nebula-BBox is only available for the NebulaGraph Enterprise Edition.

## Nebula-BBox features

Nebula-BBox provides the following features:

- View monitoring metric data via TUI, Terminal User Interface.
- Export data as CSV files.
- View data in different dimensions.
- View data for one or more metrics.
- View data for a certain time.
- View data from one or more directories or files, or mixed.
- Support Linux, macOS, and Windows systems.

## Version compatibility

The version correspondence between NebulaGraph and Nebula-BBox is as follows.

NebulaGraph	Nebula-BBox
3.4.1	3.4.0

### **Deploy Nebula-BBox**

You can deploy Nebula-BBox with RPM, DEB, or TAR packages, or with Docker. The following example uses RPM packages.

1. Obtain an RPM package.



Contact us to get the Nebula-BBox installation package.

2. Run sudo rpm -i <rpm> to install the package. For example:

sudo rpm -i nebula-bbox-<version>.x86\_64.rpm

Nebula-BBox is installed in the default path  $\mbox{\it /usr/bin/}$  in the form of a binary file  $\mbox{\it nebula-bbox}$  .

## Use Nebula-BBox

Run nebula-bbox -h/--help to view the available commands.

- 469/1069 - 2023 Vesoft Inc.



For Nebula-BBox installed in a non-default path (default path is /usr/bin/), when executing nebula-bbox related commands, it is necessary to specify the installation path of Nebula-BBox. For example, if Nebula-BBox is installed in /usr/bbox, then you need to execute /usr/bbox/nebula-bbox-h.

## VIEW NEBULA-BBOX VERSION

Run nebula-bbox version to view the version information of Nebula-BBox.

VIEW BLACK-BOX MONITORING METRICS

Run nebula-bbox metrics to view all the metrics collected by Nebula-BBox. For details about the description of metrics, see PROC(5).

VIEW BLACK-BOX MONITORING DATA

You can use Nebula-BBox to view black-box monitoring file data. The syntax is as follows:

nebula-bbox view [(-o|--output=)tui|csv] [--metrics name[,name ...]] [flags] (FILE | DIRECTORY ...)

#### Parameters

Parameter	Description
-o ,output	Specifies the output display format. Optional values are tui and csv. The default value is tui when this parameter is not specified.  tui: The output display format is TUI, Terminal User Interface.  csv: The output display format is a CSV file.
metrics	Specifies one or multiple metrics. Optional values can be the metrics returned by running nebula-bbox metrics, and all metrics are displayed when this parameter is not specified.  When specifying multiple metrics, separate them with commas, for examplemetrics <name>,<name></name></name>
flags	You can specify other parameters:output-file: When the value of -o oroutput is csv, you need to specify it to define the storage path and file name of the CSV filestart-time: View metric data from the defined start time to the current timeend-time: Defines the end time to view data collected during a period, used withstart-timeduration: Defines a duration to view data collected during the duration, used withstart-time, not used withend-time.

### Examples

The following examples assume that the NebulaGraph cluster is installed in the default path /usr/local/nebula/ and the black-box monitoring data is stored in the default path /usr/local/nebula/data/bbox/.

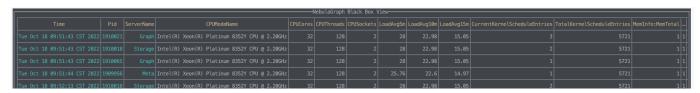


Before you want to specify one or multiple metrics, run nebula-bbox metrics to view all the metrics that you can specify. For details, see the context above.

Cases	Commands
View the data of all metrics by specifying a single file.	nebula-bbox view /usr/local/nebula/black_box/ <pid>/black_box.<timestamp>.log</timestamp></pid>
View the data of all metrics by specifying multiple files.	<pre>nebula-bbox view /usr/local/nebula/black_box/<pid>/black_box.<timestamp1>.log / usr/local/nebula/black_box/<pid>/black_box.<timestamp2>.log</timestamp2></pid></timestamp1></pid></pre>
View the data of all metrics by specifying all files.	nebula-bbox view /usr/local/nebula/black_box
View the data of all metrics by specifying multiple subdirectories.	<pre>nebula-bbox view /usr/local/nebula/black_box/<pid1> /usr/local/nebula/ black_box/<pid2></pid2></pid1></pre>
View the data of all metrics by specifying a subdirectory and a single file.	nebula-bbox view /usr/local/nebula/black_box/ <pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebula/black_box/<pid>/usr/local/nebul</pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid></pid>
View the data of a specified metric in a single file.	<pre>nebula-bbox viewmetrics <name> /usr/local/nebula/black_box/<pid>/ black_box.<timestamp>.log</timestamp></pid></name></pre>
View the data of a specified metric of all files.	nebula-bbox viewmetrics <name> /usr/local/nebula/black_box</name>
View the data of specified metrics of all files in the form of a CSV file.	nebula-bbox viewmetrics <name1>[,<name2>]output csvoutput-file <csv_filename>.csv /usr/local/nebula/black_box</csv_filename></name2></name1>
View the data of multiple specified metrics of all files.	nebula-bbox viewmetrics <name1>[,<name2>] /usr/local/nebula/black_box</name2></name1>
View the data of all metrics of all files from noon September 6, 2022, Beijing time until now.	nebula-bbox viewstart-time "Tue, 06 Sep 2022 12:00:00 +0800" /usr/local/ nebula/black_box The replacement of Tue, 06 Sep 2022 12:00:00 +0800 can be 2022-09-06T12:00:00+08:00 and 2022-09-06 04:00:00 +0800.
View the data of all metrics of all files within one hour starting from noon September 6, 2022, Beijing time.	nebula-bbox viewstart-time "Tue, 06 Sep 2022 12:00:00 +0800"duration 1h / usr/local/nebula/black_box . You can use h $^{\circ}$ m $^{\circ}$ s to specify a duration.
View the data of all metrics of all files from noon September 6, 2022, Beijing time to 13:00 September 6, 2022, Beijing time.	nebula-bbox viewstart-time "2022-09-06 04:00:00 +0800"end-time "2022-09-06 05:00:00 +0800" /usr/local/nebula/black_box

### TUI mode and shortcuts

The TUI mode displays monitoring data in the form of a table. The first line of the table shows the time, service PID based on which metrics are collected, service name, and metric names.



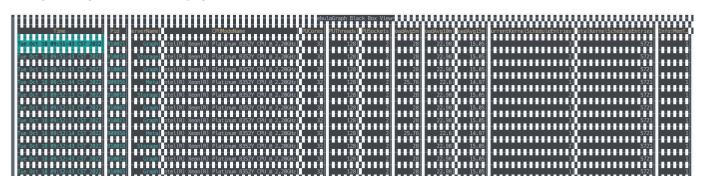
- 471/1069 - 2023 Vesoft Inc.

You can use the following shortcuts to view data in TUI mode.

Shortcut	Description
F1	Displays help.
Left	Move left.
Right	Move right.
Down	Move down.
Up	Move up.
Ctrl-A or Home	Jump to the first column of the current line.
Ctrl-E or End	Jump to the last column of the current line.
Ctrl-T	Jump to the first line.
Ctrl-B	Jump to the last line.
Enter	View the detailed information in a table cell.
Escape	Quit displaying cell details.

FAQ

Q: Why does my TUI interface display as follows?



A: The situation shown above is due to a mismatch in the Linux system character set. Run export LC\_CTYPE="en\_US.UTF-8" to solve the problem.

Last update: February 19, 2024

- 472/1069 - 2023 Vesoft Inc.

# 8. Data security

# 8.1 Authentication and authorization

### 8.1.1 Authentication

NebulaGraph replies on local authentication or LDAP authentication to implement access control.

NebulaGraph creates a session when a client connects to it. The session stores information about the connection, including the user information. If the authentication system is enabled, the session will be mapped to corresponding users.



By default, the authentication is disabled and NebulaGraph allows connections with the username root and any password.

NebulaGraph supports local authentication and LDAP authentication.

### Local authentication

Local authentication indicates that usernames and passwords are stored locally on the server, with the passwords encrypted. Users will be authenticated when trying to visit NebulaGraph.

**ENABLE LOCAL AUTHENTICATION** 

- 1. Modify the nebula-graphd.conf file (/usr/local/nebula/etc/ is the default path) to set the following parameters:
- --enable\_authorize: Set its value to true to enable authentication.

# Note

- By default, the authentication is disabled and NebulaGraph allows connections with the username root and any password.
- You can use the username root and password nebula to log into NebulaGraph after enabling local authentication. This account has the build-in God role. For more information about roles, see Roles and privileges.
- --failed\_login\_attempts: This parameter is optional, and you need to add this parameter manually. Specify the attempts of continuously entering incorrect passwords for a single Graph service. When the number exceeds the limitation, your account will be locked. For multiple Graph services, the allowed attempts are number of services \* failed\_login\_attempts.
- --password\_lock\_time\_in\_secs : This parameter is optional, and you need to add this parameter manually. Specify the time how long your account is locked after multiple incorrect password entries are entered. Unit: second.
- $2. \ Restart \ the \ Nebula Graph \ services. \ For \ how \ to \ restart, see \ \underline{Manage \ Nebula Graph \ services.}$

## LDAP authentication

Lightweight Directory Access Protocol (LDAP) is a lightweight client-server protocol for accessing directories and building a centralized account management system. LDAP authentication and local authentication can be enabled at the same time, but LDAP authentication has a higher priority. If the local authentication server and the LDAP server both have the information of user Amber, NebulaGraph reads from the LDAP server first.

- 473/1069 - 2023 Vesoft Inc.

## ENABLE LDAP AUTHENTICATION



Contact us.

Last update: February 19, 2024

## 8.1.2 User management

User management is an indispensable part of NebulaGraph access control. This topic describes how to manage users and roles.

After enabling authentication, only valid users can connect to NebulaGraph and access the resources according to the user roles.



- By default, the authentication is disabled. NebulaGraph allows connections with the username root and any password.
- Once the role of a user is modified, the user has to re-login to make the new role takes effect.

## CREATE USER

The root user with the GOD role can run CREATE USER to create a new user.

• Syntax

```
CREATE USER [IF NOT EXISTS] <user_name> [WITH PASSWORD '<password>'];
```

- IF NOT EXISTS: Detects if the user name exists. The user will be created only if the user name does not exist.
- user\_name : Sets the name of the user.
- password : Sets the password of the user.
- Syntax with enterprise edition

```
CREATE USER [IF NOT EXISTS] <user_name> [WITH PASSWORD '<password>'][WITH IP WHITELIST <ip_List>];
```

- ip\_list: Sets the IP address whitelist. The user can connect to NebulaGraph only from IP addresses in the list. Use commas to separate multiple IP addresses.
- Example

• Example with enterprise edition

## **GRANT ROLE**

Users with the **GOD** role or the **ADMIN** role can run GRANT ROLE to assign a built-in role in a graph space to a user. For more information about NebulaGraph built-in roles, see Roles and privileges.

• Syntax

```
GRANT ROLE <role_type> ON <space_name> TO <user_name>;
```

• Example

```
nebula> GRANT ROLE USER ON basketballplayer TO user1;
```

## REVOKE ROLE

Users with the **GOD** role or the **ADMIN** role can run REVOKE ROLE to revoke the built-in role of a user in a graph space. For more information about NebulaGraph built-in roles, see Roles and privileges.

• Syntax

```
REVOKE ROLE <role_type> ON <space_name> FROM <user_name>;
```

• Example

```
nebula> REVOKE ROLE USER ON basketballplayer FROM user1;
```

## **DESCRIBE USER**

Users can run DESCRIBE USER to list the roles for a specified user.

• Syntax

```
DESCRIBE USER <user_name>;
DESC USER <user_name>;
```

• Example

### **SHOW ROLES**

Users can run SHOW ROLES to list the roles in a graph space.

• Syntax

```
SHOW ROLES IN <space_name>;
```

Example

## CHANGE PASSWORD

Users can run CHANGE PASSWORD to set a new password for a user. The old password is needed when setting a new one.

• Syntax

```
CHANGE PASSWORD <user_name> FROM '<old_password>' TO '<new_password>';
```

• Example

```
nebula> CHANGE PASSWORD user1 FROM 'nebula' TO 'nebula123';
```

## **ALTER USER**

The root user with the **GOD** role can run ALTER USER to set a new password. The old password is not needed when altering the user.

Syntax

```
ALTER USER <user_name> WITH PASSWORD '<password>';
```

- Example

```
nebula> ALTER USER user2 WITH PASSWORD 'nebula';
```

• Syntax with enterprise edition

```
ALTER USER <user_name> WITH PASSWORD '<password>' [WITH IP WHITELIST <ip_list>];
```

• Example with enterprise edition



When WITH IP WHITELIST is not used, the IP address whitelist is removed and the user can connect to the NebulaGraph by any IP address.

```
nebula> ALTER USER user2 WITH PASSWORD 'nebula' WITH IP WHITELIST 192.168.10.10;
```

## DROP USER

The root user with the **GOD** role can run DROP USER to remove a user.



Removing a user does not close the current session of the user, and the user role still takes effect in the session until the session is closed.

• Syntax

```
DROP USER [IF EXISTS] <user_name>;
```

• Example

```
nebula> DROP USER user1;
```

## **SHOW USERS**

The root user with the  $\mathbf{GOD}$  role can run <code>SHOW USERS</code> to list all the users.

• Syntax

```
SHOW USERS;
```

• Example

Last update: February 19, 2024

- 478/1069 - 2023 Vesoft Inc.

## 8.1.3 Roles and privileges

A role is a collection of privileges. You can assign a role to a user for access control.

### **Built-in roles**

NebulaGraph does not support custom roles, but it has multiple built-in roles:

- GOD
- GOD is the original role with **all privileges** not limited to graph spaces. It is similar to root in Linux and administrator in Windows
- When the Meta Service is initialized, the one and only GOD role user root is automatically created with the password nebula.



Modify the password for root timely for security.

- When the --enable\_authorize parameter in the nebula-graphd.conf file (the default directory is /usr/local/nebula/etc/) is set to true:
- One cluster can only have one user with the GOD role. This user can manage all graph spaces in a cluster.
- Manual authorization of the God role is not supported. Only the root user with the default God role can be used.
- ADMIN
- An ADMIN role can **read and write** both the Schema and the data in a specific graph space.
- An ADMIN role of a graph space can grant DBA, USER, and GUEST roles in the graph space to other users.



Only roles lower than ADMIN can be authorized to other users.

- DBA
- A DBA role can **read and write** both the Schema and the data in a specific graph space.
- $\bullet$  A DBA role of a graph space CANNOT grant roles to other users.
- USER
- A USER role can read and write data in a specific graph space.
- The Schema information is **read-only** to the USER roles in a graph space.
- GUEST
- A GUEST role can **only read** the Schema and the data in a specific graph space.
- BASIC
- $\bullet$  A BASIC role can read the Schema in a specific graph space.
- ( Additional authorization required ) A BASIC role can **read and write** the Tag and Edge Type in a specific graph space.

- 479/1069 - 2023 Vesoft Inc.

# Sterpriseonly

The Basic role is only available in the Enterprise edition.



- $\bullet$  Nebula Graph does not support custom roles. Users can only use the default built-in roles.
- A user can have only one role in a graph space. For authenticated users, see User management.

## Role privileges and allowed nGQL

The privileges of roles and the nGQL statements that each role can use are listed as follows.

Privilege	God	Admin	DBA	User	Guest	Basic	Allowed nGQL
Read space	Y	Y	Y	Y	Y	Y	USE, DESCRIBE SPACE
Read schema	Y	Y	Y	Y	Y	Y	DESCRIBE TAG, DESCRIBE EDGE, DESCRIBE TAG INDEX, DESCRIBE EDGE INDEX
Write schema	Y	Y	Y		Y		CREATE TAG, ALTER TAG, CREATE EDGE, ALTER EDGE, DROP TAG, DELETE TAG, DROP EDGE, CREATE TAG INDEX, CREATE EDGE INDEX, DROP TAG INDEX, DROP EDGE INDEX
Write user	Y						CREATE USER, DROP USER, ALTER USER
Write role	Y	Y					GRANT, REVOKE
Read data	Y	Y	Y	Y	Y	С	GO, SET, PIPE, MATCH, ASSIGNMENT, LOOKUP, YIELD, ORDER BY, FETCH VERTICES, FIND, FETCH EDGES, FIND PATH, LIMIT, GROUP BY, RETURN
Write data	Y	Y	Y	Y		С	INSERT VERTEX, UPDATE VERTEX, INSERT EDGE, UPDATE EDGE, DELETE VERTEX, DELETE EDGES, DELETE TAG
Show operations	Y	Y	Y	Y	Y	Y	SHOW, CHANGE PASSWORD
Job	Y	Y	Y	Y			SUBMIT JOB COMPACT, SUBMIT JOB FLUSH, SUBMIT JOB STATS, STOP JOB, RECOVER JOB, BUILD TAG INDEX, BUILD EDGE INDEX, INGEST, DOWNLOAD
Write space	Y						CREATE SPACE, DROP SPACE, CREATE SNAPSHOT, DROP SNAPSHOT, BALANCE, CONFIG



Only the Enterprise Edition supports fine-grain (Tag/Edge type level) permission management based on Basic roles.



- The results of SHOW operations are limited to the role of a user. For example, all users can run SHOW SPACES, but the results only include the graph spaces that the users have privileges.
- Only the GOD role can run SHOW USERS and SHOW SNAPSHOTS.

- 480/1069 - 2023 Vesoft Inc.

### Basic role(Enterprise Edition)

SYNTAX



The following commands can be executed only after entering the graph space.

· Grant Basic user Tag/Edge Permissions.

```
GRANT { OPTION[,OPTION] } [ TAG { * | <tag>[,...] } | EDGE { * | <edge_type>[, ...] }] TO <user_name>;
OPTION = { READ | WRITE }
```

• Revoke Basic user Tag/Edge Permissions.

```
REVOKE { OPTION[,OPTION] } [ TAG { * | <tag>[,...] } | EDGE { * | <edge_type>[, ...] }] TO <user_name>;
OPTION = { READ | WRITE }
```

· Show Basic user Tag/Edge Permissions.

```
SHOW GRANTS [<user_name>]
```

### **PRECAUTIONS**

- The default Basic role does not have any Tag/Edge read and write permissions.
- Only GOD and ADMIN role users can perform GRANT and REVOKE operations.
- Only allow users to GRANT and REVOKE the Basic role in the specified graph space, and are not allowed to grant authorization to other role users.
- The Basic role cannot insert vertices without a tag.
- Both read and write privileges are required when performing an UPDATE or UPSERT operation.

### EXAMPLES

```
# Create 'test' user
nebula> CREATE USER test WITH PASSWORD 'nebula';
# Grant Basic role permissions for the `test` user
nebula> GRANT ROLE BASIC ON basketballplayer TO test;
# Choose graph space `basketballplayer
nebula> use basketballplayer;
# Grant read and write permissions to `test` user Tag `player` and Edge Type `follow` and `serve
# Granting the user the read and write permissions of the specified Tag/Edge must be after specifying the graph space
nebula> GRANT READ, WRITE TAG player EDGE follow, serve TO test;
# Show `test` user permissions
nebula> > SHOW GRANTS test;
| user | READ(TAG) | READ(EDGE)
                                                   | WRITE(TAG) | WRITE(EDGE)
| "test" | ["player"] | ["follow", "serve"] | ["player"] | ["follow", "serve"] |
# Revoke `test` user all Edge Type read and write permissions nebula> REVOKE READ,WRITE EDGE * FROM test;
# Show 'test' user permissions
nebula> SHOW GRANTS test:
user | READ(TAG) | READ(EDGE) | WRITE(TAG) | WRITE(EDGE)
| "test" | ["player"] | []
                                         | ["player"] | []
# When Basic role users read data without permission, the following error will occur.
nebula> MATCH (v:player)-[:likex]-() RETURN v;
[ERROR (-1008)]: PermissionError: Edge `likex' does not exist or is not readable.
```

# Caution

For Basic role users, an error will be reported for Tag/Edge Type that explicitly specify no read permission, and no errors will be reported for Tag/Edge Types that do not explicitly specify no read permission. During the traverse process, all queries cannot read the unprivileged Tag/Edge Type and its properties. The read permission of the Edge Type can control the expansion behavior of the edge. During the traversal process, if the Edge Type has no permission, it will not be expanded; the read permission of the Tag does not control the expansion behavior of the vertex. Even if the Tag has no permission during the expansion process, also can be expanded.

Last update: February 19, 2024

## 8.1.4 OpenLDAP authentication

This topic introduces how to connect NebulaGraph to the OpenLDAP server and use the DN (Distinguished Name) and password defined in OpenLDAP for authentication.



This feature is supported by the Enterprise Edition only.

### Authentication method

After the OpenLDAP authentication is enabled and users log into NebulaGraph with the account and password, NebulaGraph checks whether the login account exists in the Meta service. If the account exists, NebulaGraph finds the corresponding DN in OpenLDAP according to the authentication method and verifies the password.

OpenLDAP supports two authentication methods: simple bind authentication (SimpleBindAuth) and search bind authentication (SearchBindAuth).

### SIMPLEBINDAUTH

Simple bind authentication splices the login account and the configuration information of Graph services into a DN that can be recognized by OpenLDAP, and then authenticates on OpenLDAP based on the DN and password.

### SEARCHBINDAUTH

Search bind authentication reads the Graph service configuration information and queries whether the uid in the configuration matches the login account. If they match, search bind authentication reads the DN, and then uses the DN and password to verify on OpenLDAP.



Only the uid attribute in OpenLDAP can be used to specify a username for SearchBindAuth.

## Prerequisites

- OpenLDAP is installed.
- The account and password are imported on OpenLDAP.
- The server where OpenLDAP is located has opened the corresponding authentication port.

#### **Procedures**

Take the existing account test2 and password passwdtest2 on OpenLDAP as an example.

1. Connect to NebulaGraph, create and authorize the shadow account test2 corresponding to OpenLDAP.

```
nebula> CREATE USER test2 WITH PASSWORD '';
nebula> GRANT ROLE ADMIN ON basketballplayer TO test2;
```



When creating an account in NebulaGraph, the password can be set arbitrarily.

- 2. Edit the configuration file nebula-graphd.conf (The default path is /usr/local/nebula/etc/):
- SimpleBindAuth (Recommended)

```
# Whether to get the configuration information from the configuration file.
--local_config=true
# Whether to enable authentication.
--enable_authorize=true
# Authentication methods include password, ldap, and cloud.
--auth_type=ldap
# The address of the OpenLDAP server.
--ldap_server=192.168.8.211
# The port of the OpenLDAP server.
--ldap_port=389
# The name of the Schema in OpenLDAP.
--ldap_scheme=ldap
# The prefix of DN.
--ldap_prefix=uid=
# The suffix of DN.
--ldap_suffix=,ou=it,dc=sys,dc=com
```

SearchBindAuth

```
# Whether to get the configuration information from the configuration file.
--local_config=true
# Whether to enable authentication.
--enable_authorize=true
# Authentication methods include password, ldap, and cloud.
--auth_type=ldap
# The address of the OpenLDAP server.
--ldap_server=192.168.8.211
# The port of the OpenLDAP server.
--ldap_port=389
# The name of the Schema in OpenLDAP.
--ldap_scheme=ldap
# The DN that binds the target.
--ldap_basedn=ou=it,dc=sys,dc=com
# The OpenLDAP login username. If anonymous access is supported, this parameter is optional. Otherwise, it is required.
--ldap_bindpasswd=admin
--ldap_bindpasswd=admin
```

- 3. Restart NebulaGraph services to make the new configuration valid.
- 4. Run the login test.

```
$ ./nebula-console --addr 127.0.0.1 --port 9669 -u test2 -p passwdtest2
2021/09/08 03:49:39 [INFO] connection pool is initialized successfully
Welcome to NebulaGraph!
```



After using OpenLDAP for authentication, local users (including root ) cannot log in normally.

Last update: February 19, 2024

- 484/1069 - 2023 Vesoft Inc.

# 8.2 SSL encryption

NebulaGraph supports data transmission with SSL encryption between clients, the Graph service, the Meta service, and the Storage service. This topic describes how to enable SSL encryption.

### 8.2.1 Precaution

Enabling SSL encryption will slightly affect the performance, such as causing operation latency.

## 8.2.2 Parameters

Parameter	Default value	Description
cert_path	-	The path to the PEM certification.
key_path	-	The path to the key certification.
password_path	-	The path to the password file certification.
ca_path	-	The path to the trusted CA file.
enable_ssl	false	Whether to enable SSL encryption.
enable_graph_ssl	false	Whether to enable SSL encryption in the Graph service only.
enable_meta_ssl	false	Whether to enable SSL encryption in the Meta service only.

### 8.2.3 Certificate modes

To use SSL encryption, SSL certificates are required. NebulaGraph supports two certificate modes.

· Self-signed certificate mode

In this mode, users need to make the signed certificate by themselves and set <code>cert\_path</code>, <code>key\_path</code>, and <code>password\_path</code> in the corresponding file according to encryption policies.

• CA-signed certificate mode

In this mode, users need to apply for the signed certificate from a certificate authority and set cert\_path, key\_path, and password\_path in the corresponding file according to encryption policies.

## 8.2.4 Encryption policies

NebulaGraph supports three encryption policies. For details, see Usage explanation.

- Encrypt the data transmission between clients, the Graph service, the Meta service, and the Storage service.

  Add enable\_ssl = true to the configuration files of nebula-graphd.conf, nebula-metad.conf, and nebula-storaged.conf.
- Encrypt the data transmission between clients and the Graph service.

This policy applies to the case that the clusters are set in the same server room. Only the port of the Graph service is open to the outside because other services can communicate over the internal network without encryption. Add <code>enable\_graph\_ssl = true to the configuration file of nebula-graphd.conf</code>.

• Encrypt the data transmission related to the Meta service in the cluster.

This policy applies to transporting classified information to the Meta service. Add  $enable\_meta\_ssl = true$  to the configuration files of nebula-graphd.conf, nebula-metad.conf, and nebula-storaged.conf.

## 8.2.5 Steps

- 1. Ensure the certificate mode and the encryption policy.
- 2. Add the certificate configuration and the policy configuration in corresponding files.

For example, the three configuration files need to be set as follows when using a self-signed certificate and encrypt data transmission between clients, the Graph service, the Meta service, and the Storage service.

```
--cert_path=xxxxxx
--key_path=xxxxxx
--password_path=xxxxxx
--enable_ssl=true
```

3. Set the SSL and the trusted CA in clients. For code examples, see <a href="nebula-test-run.py">nebula-test-run.py</a>.

Last update: February 19, 2024

# 9. Backup & Restore

# 9.1 NebulaGraph BR (Community Edition)

# 9.1.1 What is Backup & Restore

Backup & Restore (BR for short) is a Command-Line Interface (CLI) tool to back up data of graph spaces of NebulaGraph and to restore data from the backup files.

### **Features**

The BR has the following features. It supports:

- Backing up and restoring data in a one-click operation.
- Restoring data in the following backup file types:
- Local Disk (SSD or HDD). It is recommend to use local disk in test environment only.
- Amazon S3 compatible interface, such as Alibaba Cloud OSS, MinIO, Ceph RGW, etc.
- Backing up and restoring the entire NebulaGraph cluster.
- Backing up data of specified graph spaces (experimental).

### Limitations

- Supports NebulaGraph v3.x only.
- Supports full backup, but not incremental backup.
- Currently, NebulaGraph Listener and full-text indexes do not support backup.
- If you back up data to the local disk, the backup files will be saved in the local path of each server. You can also mount the NFS on your host to restore the backup data to a different host.
- During the backup process, both DDL and DML statements in the specified graph spaces are blocked. We recommend that you do the operation within the low peak period of the business, for example, from 2:00 AM to 5:00 AM.
- The backup graph space can be restored to the original cluster only. Cross clusters restoration is not supported. Make sure the number of hosts in the cluster is not changed. Restoring a specified graph space will delete all other graph spaces in the cluster.
- Restoration requires that the number of the storage servers in the original cluster is the same as that of the storage servers in the target cluster and storage server IPs must be the same. Restoring the specified space will clear all the remaining spaces in the cluster.
- During the restoration process, there is a time when NebulaGraph stops running.
- Using BR in a container-based NebulaGraph cluster is not supported.

### How to use BR

To use the BR, follow these steps:

- 1. Install BR.
- 2. Use BR to back up data.
- 3. Use BR to restore data from backup files.

Last update: February 19, 2024

## 9.1.2 Install BR

This topic introduces the installation of BR in bare-metal deployment scenarios. For the deployment of BR in K8s Operator, see Backup and restore data using NebulaGraph Operator.

### Notes

To use the BR (Enterprise Edition) tool, you need to install the NebulaGraph Agent service, which is taken as a daemon for each machine in the cluster that starts and stops the NebulaGraph service, and uploads and downloads backup files. The BR (Enterprise Edition) tool and the Agent plug-in are installed as described below.

## Version compatibility

NebulaGraph	BR	Agent
$3.3.0 \sim 3.4.1$	3.3.0	$0.2.0 \sim 3.4.0$
$3.0.x \sim 3.2.x$	0.6.1	$0.1.0 \sim 0.2.0$

### Install BR with a binary file

1. Install BR.

 $wget\ https://github.com/vesoft-inc/nebula-br/releases/download/v3.3.0/br-3.3.0-linux-amd 64 to 100 to 10$ 

2. Change the binary file name to br.

sudo mv br-3.3.0-linux-amd64 br

3. Grand execute permission to BR.

sudo chmod +x br

4. Run ./br version to check BR version.

[nebula-br]\$ ./br version Nebula Backup And Restore Utility Tool,V-3.3.0

## Install BR with the source code

Before compiling the BR, do a check of these:

- Go 1.14.x or a later version is installed.
- make is installed.

To compile the BR, follow these steps:

1. Clone the nebula-br repository to your machine.

 ${\tt git\ clone\ https://github.com/vesoft-inc/nebula-br.git}$ 

2. Change to the br directory.

cd nebula-br

3. Compile the BR.

make

Users can enter bin/br version on the command line. If the following results are returned, the BR is compiled successfully.

```
[nebula-br]$ bin/br version
NebulaGraph Backup And Restore Utility Tool,V-3.3.0
```

## **Install Agent**

NebulaGraph Agent is installed as a binary file in each machine and serves the BR tool with the RPC protocol.

In **each machine**, follow these steps:

1. Install Agent.

```
wget https://github.com/vesoft-inc/nebula-agent/releases/download/v3.4.0/agent-3.4.0-linux-amd64
```

2. Rename the Agent file to agent .

```
sudo mv agent-3.4.0-linux-amd64 agent
```

3. Add execute permission to Agent.

```
sudo chmod +x agent
```

4. Start Agent.



Before starting Agent, make sure that the Meta service has been started and Agent has read and write access to the corresponding NebulaGraph cluster directory and backup directory.

```
sudo nohup ./agent --agent="<agent_node_ip>:8888" --meta="<metad_node_ip>:9559" > nebula_agent.log 2>&1 &
```

- --agent : The IP address and port number of Agent.
- $\bullet\,$  --meta : The IP address and access port of any Meta service in the cluster.
- --ratelimit: (Optional) Limits the speed of file uploads and downloads to prevent bandwidth from being filled up and making other services unavailable. Unit: Bytes.

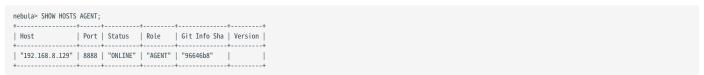
For example:

```
sudo nohup ./agent --agent="192.168.8.129:8888" --meta="192.168.8.129:9559" --ratelimit=1048576 > nebula_agent.log 2>&1 &
```



The IP address format for --agent should be the same as that of Meta and Storage services set in the configuration files. That is, use the real IP addresses or use 127.0.0.1. Otherwise Agent does not run.

5. Log into NebulaGraph and then run the following command to view the status of Agent.



## FAQ

THE ERROR `E\_LIST\_CLUSTER\_NO\_AGENT\_FAILURE

If you encounter E\_LIST\_CLUSTER\_NO\_AGENT\_FAILURE error, it may be due to the Agent service is not started or the Agent service is not registered to Meta service. First, execute SHOW HOSTS AGENT to check the status of the Agent service on all nodes in the cluster, when

the status shows OFFLINE, it means the registration of Agent failed, then check whether the value of the --meta option in the command to start the Agent service is correct.

Last update: February 19, 2024

## 9.1.3 Use BR to back up data

After the BR is installed, you can back up data of the entire graph space. This topic introduces how to use the BR to back up data.

### **Prerequisites**

To back up data with the BR, do a check of these:

- Install BR and Agent and run Agent on each host in the cluster.
- The NebulaGraph services are running.
- If you store the backup files locally, create a directory with the same absolute path on the meta servers, the storage servers, and the BR machine for the backup files and get the absolute path. Make sure the account has write privileges for this directory.



In the production environment, we recommend that you mount Network File System (NFS) storage to the meta servers, the storage servers, and the BR machine for local backup, or use Amazon S3 or Alibaba Cloud OSS for remote backup. When you restore the data from local files, you must manually move these backup files to a specified directory, which causes redundant data and troubles. For more information, see Restore data from backup files.

### Procedure

In the BR installation directory (the default path of the compiled BR is ./bin/br), run the following command to perform a full backup for the entire cluster.



Make sure that the local path where the backup file is stored exists.

\$ ./br backup full --meta <ip\_address> --storage <storage\_path>

## For example:

• Run the following command to perform a full backup for the entire cluster whose meta service address is 192.168.8.129:9559, and save the backup file to /home/nebula/backup/.



If there are multiple metad addresses, you can use any one of them.



If you back up data to a local disk, only the data of the leader metad is backed up by default. So if there are multiple metad processes, you need to manually copy the directory of the leader metad (path <storage\_path>/meta) and overwrite the corresponding directory of other follower meatd processes.

```
$ ./br backup full --meta "192.168.8.129:9559" --storage "local:///home/nebula/backup/"
```

• Run the following command to perform a full backup for the entire cluster whose meta service address is 192.168.8.129:9559, and save the backup file to backup in the br-test bucket of the object storage service compatible with S3 protocol.

```
$ ./br backup full --meta "192.168.8.129:9559" --s3.endpoint "http://192.168.8.129:9000" --storage="s3://br-test/backup/" --s3.access_key=minioadmin --s3.secret_key=minioadmin --s3.region=default
```

The parameters are as follows.

Parameter	Data type	Required	Default value	Description
-h,-help	-	No	None	Checks help for restoration.
debug	-	No	None	Checks for more log information.
log	string	No	"br.log"	Specifies detailed log path for restoration and backup.
meta	string	Yes	None	The IP address and port of the meta service.
space	string	Yes	None	(Experimental feature) Specifies the names of the spaces to be backed up. All spaces will be backed up if not specified. Multiple spaces can be specified, and format isspaces nba_01spaces nba_02.
storage	string	Yes	None	The target storage URL of BR backup data. The format is: \ <schema>://\<path>.  Schema: Optional values are local and s3.  When selecting s3, you need to fill in s3.access_key, s3.endpoint, s3.region, and s3.secret_key.  PATH: The path of the storage location.</path></schema>
 s3.access_key	string	No	None	Sets AccessKey ID.
s3.endpoint	string	No	None	Sets the S3 endpoint URL, please specify the HTTP or HTTPS scheme explicitly.
s3.region	string	No	None	Sets the region or location to upload or download the backup.
 s3.secret_key	string	No	None	Sets SecretKey for AccessKey ID.

# Next to do

After the backup files are generated, you can use the BR to restore them for NebulaGraph. For more information, see Use BR to restore data.

Last update: February 19, 2024

## 9.1.4 Use BR to restore data

If you use the BR to back up data, you can use it to restore the data to NebulaGraph. This topic introduces how to use the BR to restore data from backup files.



During the restoration process, the data on the target NebulaGraph cluster is removed and then is replaced with the data from the backup files. If necessary, back up the data on the target cluster.



The restoration process is performed OFFLINE.

## Prerequisites

To restore data with the BR, do a check of these:

- Install BR and Agent and run Agent on each host in the cluster.
- Download nebula-agent and start the agent service in each cluster(including metad, storaged, graphd) host.
- No application is connected to the target NebulaGraph cluster.
- Make sure that the target and the source NebulaGraph clusters have the same topology, which means that they have exactly the same number of hosts. The number of data folders for each host is consistently distributed.

# Procedures

In the BR installation directory (the default path of the compiled BR is ./br), run the following command to perform a full backup for the entire cluster.

1. Users can use the following command to list the existing backup information:

```
$ ./br show --storage <storage_path>
```

For example, run the following command to list the backup information in the local /home/nebula/backup path.



Or, you can run the following command to list the backup information stored in S3 URL s3://192.168.8.129:9000/br-test/backup.

\$ ./br show --s3.endpoint "http://192.168.8.129:9000" --storage="s3://br-test/backup/" --s3.access\_key=minioadmin --s3.secret\_key=minioadmin --s3.region=default

Parameter	Data type	Required	Default value	Description
-h,-help	-	No	None	Checks help for restoration.
-debug	-	No	None	Checks for more log information.
- log	string	No	"br.log"	Specifies detailed log path for restoration and backup.
storage	string	Yes	None	The target storage URL of BR backup data. The format is: <schema>://<path>.  Schema: Optional values are local and s3.  When selecting s3, you need to fill in s3.access_key, s3.endpoint, s3.region, and s3.secret_key.  PATH: The path of the storage location.</path></schema>
s3.access_key	string	No	None	Sets AccessKey ID.
s3.endpoint	string	No	None	Sets the S3 endpoint URL, please specify the HTTP or HTTPS scheme explicitly.
s3.region	string	No	None	Sets the region or location to upload or download the backup.
s3.secret_key	string	No	None	Sets SecretKey for AccessKey ID.

2. Run the following command to restore data.

```
$ ./br restore full --meta <ip_address> --storage <storage_path> --name <backup_name>
```

For example, run the following command to upload the backup files from the local /home/nebula/backup/ to the cluster where the meta service's address is 192.168.8.129:9559.

```
$ ./br restore full --meta "192.168.8.129:9559" --storage "local:///home/nebula/backup/" --name BACKUP_2021_12_08_18_38_08
```

Or, you can run the following command to upload the backup files from the S3 URL s3://192.168.8.129:9000/br-test/backup.

```
$ ./br restore full --meta "192.168.8.129:9559" --s3.secret_key=minioadmin --s3.secret_key=minioadmin
```

If the following information is returned, the data is restored successfully.

Restore succeed.



If your new cluster hosts' IPs are not all the same as the backup cluster, after restoration, you should run add hosts to add the Storage host IPs in the new cluster one by one.

- 496/1069 - 2023 Vesoft Inc.

The parameters are as follows.

Parameter	Data type	Required	Default value	Description
-h,-help	-	No	None	Checks help for restoration.
-debug	-	No	None	Checks for more log information.
-log	string	No	"br.log"	Specifies detailed log path for restoration and backup.
-meta	string	Yes	None	The IP address and port of the meta service.
-name	string	Yes	None	The name of backup.
storage	string	Yes	None	The target storage URL of BR backup data. The format is: \ <schema>://\<path>.  Schema: Optional values are local and s3.  When selecting s3, you need to fill in s3.access_key, s3.endpoint, s3.region, and s3.secret_key.  PATH: The path of the storage location.</path></schema>
s3.access_key	string	No	None	Sets AccessKey ID.
s3.endpoint	string	No	None	Sets the S3 endpoint URL, please specify the HTTP or HTTPS scheme explicitly.
s3.region	string	No	None	Sets the region or location to upload or download the backup.
s3.secret_key	string	No	None	Sets SecretKey for AccessKey ID.

<sup>3.</sup> Run the following command to clean up temporary files if any error occurred during backup. It will clean the files in cluster and external storage. You could also use it to clean up old backups files in external storage.

\$ ./br cleanup --meta <ip\_address> --storage <storage\_path> --name <backup\_name>

The parameters are as follows.

Parameter	Data type	Required	Default value	Description
-h,-help	-	No	None	Checks help for restoration.
-debug	-	No	None	Checks for more log information.
- log	string	No	"br.log"	Specifies detailed log path for restoration and backup.
-meta	string	Yes	None	The IP address and port of the meta service.
-name	string	Yes	None	The name of backup.
storage	string	Yes	None	The target storage URL of BR backup data. The format is: \ <schema>://\<path>.  Schema: Optional values are local and s3.  When selecting s3, you need to fill in s3.access_key, s3.endpoint, s3.region, and s3.secret_key.  PATH: The path of the storage location.</path></schema>
s3.access_key	string	No	None	Sets AccessKey ID.
s3.endpoint	string	No	None	Sets the S3 endpoint URL, please specify the HTTP or HTTPS scheme explicitly.
s3.region	string	No	None	Sets the region or location to upload or download the backup.
s3.secret_key	string	No	None	Sets SecretKey for AccessKey ID.

Last update: February 19, 2024

# 9.2 NebulaGraph BR (Enterprise Edition)

## 9.2.1 What is Backup Restore (Enterprise Edition)

Backup Restore (BR for short) Enterprise Edition is a Command-Line Interface (CLI) tool. With BR Enterprise Edition, you can back up and restore NebulaGraph data.



The BR Enterprise Edition tool is for NebulaGraph Enterprise Edition only.

#### **Features**

- Backups and restoration of NebulaGraph data with a single command:
- · Support full and incremental backups.
- Support the restoration of a full backup.
- Support restoration across NebulaGraph clusters.
- Cloud and on-premises backup and restoration:
- Local disks (SSD or HDD). It is recommended to use local disks with the shared storage service NFS.
- Cloud services compatible with the Amazon S3 interface, such as Alibaba Cloud OSS, MinIO, Ceph RGW, etc.
- Support to view the progress of backup or restoration.

### Limits

- The version of NebulaGraph Enterprise Edition clusters must be equal to or greater than v3.4.1.
- Listener backups are not supported.
- Full-text indexes data backups are not supported.
- If you back up data to a local disk, the backup files will be saved in the local storage path of each server in a NebulaGraph cluster. You can also mount the NFS on your server to restore the backup data to a different server.
- During the backup process, both DDL and DML statements in a specified graph space are blocked. We recommend that you perform the operation within the low peak period of the business, for example, from 2:00 AM to 5:00 AM.
- Restoration requires that the number of the storage servers in the original cluster is the same as that of the storage servers in the target cluster and storage server IPs must be the same.
- Backing up data of a specified graph space is not supported.
- Using BR Enterprise Edition in a container-based NebulaGraph cluster is not supported.

### Usage process

To use BR Enterprise Edition, follow these steps:

- 1. Install BR (Enterprise Edition)
- 2. Back up data
- 3. Restore data

Last update: February 19, 2024

- 499/1069 - 2023 Vesoft Inc.

## 9.2.2 Install BR (Enterprise Edition)

The BR Enterprise Edition tool is used to back up and restore NebulaGraph Enterprise Edition data. This topic describes how to install this tool.

### Notes

To use the BR (Enterprise Edition) tool, you need to install the NebulaGraph Agent plug-in, which is taken as a daemon for each machine in the cluster that starts and stops the NebulaGraph service, and uploads and downloads backup files. The BR (Enterprise Edition) tool and the Agent plug-in are installed as described below.

## Version compatibility

NebulaGraph Enterprise Edition	<b>BR Enterprise Edition</b>	Agent
3.4.1	3.4.1 \ 3.4.0	3.4.0

## Install BR (Enterprise Edition)

The BR (Enterprise Edition) is a command-line interface (CLI) tool that helps to back up NebulaGraph data or restore the data through backup files.

Follow these steps:

1. Obtain the RPM package.



Contact us to obtain the BR Enterprise Edition installation package.

2. Run sudo rpm -i <rpm> to install the RPM package.

For example, install BR Enterprise Edition with the following command, and the default installation path is /usr/local/br-ent/:

sudo rpm -i nebula-br-ent-<version>.x86\_64.rpm

In the BR Enterprise Edition installation directory, run ./br version to view version information. The following information is returned:

[br-ent]\$ ./br version
Nebula Backup And Restore Utility Tool,V-3.4.1

## Install Agent

NebulaGraph Agent is installed as a binary file in each machine and serves the BR tool with the RPC protocol.

## In **each machine**, follow these steps:

1. Install Agent.

wget https://github.com/vesoft-inc/nebula-agent/releases/download/v3.4.0/agent-3.4.0-linux-amd64

2. Rename the Agent file to agent .

sudo mv agent-3.4.0-linux-amd64 agent

3. Add execute permission to Agent.

sudo chmod +x agent

4. Start Agent.



Before starting Agent, make sure that the Meta service has been started and Agent has read and write access to the corresponding NebulaGraph cluster directory and backup directory.

sudo nohup ./agent --agent="<agent\_node\_ip>:8888" --meta="<metad\_node\_ip>:9559" > nebula\_agent.log 2>&1 &

- --agent : The IP address and port number of Agent.
- --meta: The IP address and access port of any Meta service in the cluster.
- --ratelimit: (Optional) Limits the speed of file uploads and downloads to prevent bandwidth from being filled up and making other services unavailable. Unit: Bytes.

For example:

sudo nohup ./agent --agent="192.168.8.129:8888" --meta="192.168.8.129:9559" --ratelimit=1048576 > nebula\_agent.log 2>&1 &

5. Log into NebulaGraph and then run the following command to view the status of Agent.

Last update: February 19, 2024

- 501/1069 - 2023 Vesoft Inc.

## 9.2.3 Back up data with BR (Enterprise Edition)

You can use BR (Enterprise Edition) to back up NebulaGraph data. You can perform full backups and incremental backups. You can also back up data to your local disks and to cloud storage services compatible with the Amazon S3 interface. This topic introduces how to back up NebulaGraph data.

## Background

- A full backup is a backup of your NebulaGraph database's entire.
- An incremental backup covers all files that have changed or been modified since the last backup was made. The last backup can be either a full backup or an incremental backup.
- For the data directory structure of NebulaGraph, see the (default) path usr/local/nebula-ent/data.

### Notes

- During the process of a full backup, it may take a long time if the amount of the data to be backed up is large.
- During the process of data backup, DDL and DML statements in the specified graph space are blocked. We recommend that you perform the backup operation during the low business peak period.
- The cluster performing an incremental backup needs to be the same as the cluster specified for the previous backup. The storage path of the incremental backup should also be the same as the path of the previous backup.
- Make sure that the time between each incremental backup and the last backup is less than a wal\_ttl time.
- Make sure that Agent has read and write permissions for the corresponding NebulaGraph cluster installation directory and backup directory.

## Prerequisites

- The NebulaGraph services are running.
- You have installed BR Enterprise Edition and Agent, and have started the Agent(s) that are installed in each machine.
- You have created a directory with the same absolute path on the machines where the meta service, storage service, and the BR Enterprise Edition tool are located. Make sure your account has write privileges for this directory.

## Full backups

TO A CLOUD STORAGE SERVICE



You can back up data to the cloud storage services that are compatible with the Amazon S3 interface.

In the directory of the BR Enterprise Edition installation tool, run the following command to back up the entire data of your NebulaGraph database to a cloud storage service:

```
./br backup full --meta <ip_address:port> --s3.access_key <access_key --s3.secret_key <secret_key> --s3.region <region_name> --storage s3://<storage_path> --s3.endpoint <endpoint_url>
```

For example, perform a full backup of the entire cluster where one of the meta service addresses is 192.168.8.129:9559, and back up data to the / directory in the nebula-br-test bucket of Amazon S3.

./br backup full --meta 192.168.8.129:9559 --s3.access\_key QImbbGDjfQExxx --s3.secret\_key dVSJZfl7tnoFq7Z5zt6sfxxxx --s3.region us-east-1 --storage s3://nebula-br-test/ --s3.endpoint http://192.168.8.xxx:9000/

- 502/1069 - 2023 Vesoft Inc.

TO A LOCAL DISK



In the production environment, we recommend that you mount NFS (Network File System) to the machines where the meta service, the storage service, and the BR Enterprise Edition tool are located for local backups, or use Alibaba Cloud OSS, and Amazon S3 for cloud backups. Otherwise, you must manually move these backup files to the specified directory before restoring the data.

For local backups, only the data of the leader mated and the leader partitions are backed up. When the shared storage service is not used and there are multiple metads or the replica number of partitions is greater than 1, you need to manually copy the directory of the backed-up leader metad (<storage\_path>/meta) and overwrite the corresponding directories of other follower metads, and copy the corresponding partition data from the directory of the leader partitions (<storage\_path>/<partition\_id>) to the corresponding directories of other follower partitions. It is not recommended for you to have the copy operation.

In the directory of the BR Enterprise Edition installation tool, run the following command to back up the entire data of your NebulaGraph database to a local disk:



Make sure you have created a directory where your data to be backed up.

```
./br backup full --meta <ip_address:port> --storage local://<storage_path>
```

For example, perform a full backup of the entire cluster where one of the meta server addresses is 192.168.8.129:9559, and back up data to the /backup/ directory on the local disk.

```
./br backup full --meta "192.168.8.129:9559" --storage "local:///backup/"
```

DIRECTORY STRUCTURE

Full backups back up all the data of the leader metad and leader partitions. The structure of a backed-up directory is as follows:



## Incremental backups



Before performing an incremental backup, make sure that the data of the previous full backup or incremental backup is not changed. Otherwise, the incremental backup may fail.

### TO A CLOUD STORAGE SERVICE



You can back up data to the cloud storage services that are compatible with the Amazon S3 interface.

In the directory of the BR Enterprise Edition installation tool, run the following command to back up the incremental data of your NebulaGraph database to a cloud storage service:

```
./br backup incr --meta <ip_address:port> --s3.access_key <access_key> --s3.secret_key <secret_key> --s3.region <region_name> --storage s3://<storage_path> --s3.endpoint <endpoint_url> --base <backup_file_name>
```

For example, based on the file BACKUP\_2022\_08\_11\_09\_11\_07, perform an incremental backup for the cluster where one of the meta server addresses is 192.168.8.129:9559, and back up data to the / directory in the nebula-br-test bucket of Amazon S3.

```
./br backup incr --meta 192.168.8.129:9559 --s3.access_key QImbbGDjfQExxx --s3.secret_key dVSJZfl7tnoFq7Z5zt6sfxxxx --s3.region us-east-1 --storage s3://nebula-br-test/ --s3.endpoint http://192.168.8.xxx:9000/ --base BACKUP_2022_08_11_09_11_07
```

#### TO A LOCAL DISK

In the directory of the BR Enterprise Edition installation tool, run the following command to back up the incremental data of your NebulaGraph database to a local disk:



Make sure that the local path where the backup file is stored exists.

```
./br backup incr --meta <ip_address:port> --storage local://<storage_path> --base <backup_file_name>
```

For example, based on the file BACKUP\_2022\_08\_11\_09\_11\_07, perform an incremental backup for the cluster where one of the meta server addresses is 192.168.8.129:9559, and back up data to the /backup/ directory on the local disk.

```
./br backup incr --meta "192.168.8.129:9559" --storage "local:///backup/" --base BACKUP_2022_08_11_09_11_07
```

## DIRECTORY STRUCTURE

In addition to the leader meta data, for the leader partition data of an existing graph space (graph space ID 1 in the following code block), Incremental backups only back up the wal directory of the leader partition; for the leader partition data of the newly added graph space (graph space ID 4 in the following code block), the full data and wal directories are backed up. The incremental backup directory structure may be different compared to the full backup data structure.

The structure of an incremental backup is as follows:

```
BACKUP_2022_08_12_08_58_23.meta (Backup of metadata, such as host information and the storage path of partitions)
data (Backup of the storage data)
     1 (Graph space ID)
          1 (Partition ID)
              wal (wal directory)
                  00000000000000000671.wal
                   0000000000000000700.wal
                  commitlog.id
     4 (Graph space ID)
              data (data directory)
                   000009.sst
                   CURRENT
                   MANIFEST-000004
                  OPTIONS-000007
              wal (wal directory)
                  000000000000000001.wal
                  commitleg.id
meta (Backup of the meta data)
```

#### Options

Option	Data type	Required	Default value	Description
-h,help	-	No	None	View more information about BR commands.
debug	-	No	None	View more information for BR logs.
log	string	No	"br.log"	The path of BR logs.
concurrency	int	No	None	Used to control the number of concurrent file uploads during data backup. The default value is $5$ .
meta	string	Yes	None	The IP address and port number of any Meta service in the cluster.
storage	string	Yes	None	The storage path of the data to be backed up in the form of <schema>://<storage_path>.  schema: Two options, s3 and local.  When schema is s3, you still need to add options of s3.access_key \ s3.endpoint \ s3.region, and s3.secret_key.  <storage_path>: The storage path.</storage_path></storage_path></schema>
 s3.access_key	string	No	None	The Access Key ID that is used to identify a user.
s3.endpoint	string	No	None	The S3 endpoint URL. You need to specify the HTTP or HTTPS scheme explicitly.
s3.region	string	No	None	The physical location of a data center.
 s3.secret_key	string	No	None	The Access Key Secret that is used to verify the identity of the user.
base	string	Yes	None	The name of the directory of any previous backup.  Based on this backup, perform an incremental backup.

# View backup progress

The log file br.log of BR can be viewed in the installation directory. The log file will record the backup progress and will look like as follows:

```
{"level":"info","msg":"full upload storaged partition finished, progress: 1/20","time":"2023-03-15T02:13:20.946Z"}
{"level":"info","msg":"full upload storaged partition finished, progress: 2/20","time":"2023-03-15T02:13:21.154Z"}
{"level":"info","msg":"full upload storaged partition finished, progress: 3/20","time":"2023-03-15T02:13:21.537Z"}
```

#### What is next?

After your NebulaGraph data is backed up, you can restore the data to NebulaGraph. For details, see Restore data.



DO NOT change the name and path of the backed-up files. Otherwise, data restoration will fail.

Last update: February 19, 2024

- 505/1069 - 2023 Vesoft Inc.

# 9.2.4 Restore data with BR (Enterprise Edition)

For the backed-up NebulaGraph data with the BR tool, you can restore the data. This topic describes how to restore data from backup files.

#### **Notes**

- After the data restoration is executed successfully, the existing data on the target cluster will be deleted and then replaced with the data in the backup directory. It is recommended to back up the data on the target cluster in advance.
- There will be a period of NebulaGraph service unavailability for data restoration, so it is recommended to operate during the low business peak period.

#### Prerequisites

- You have installed BR Enterprise Edition and Agent, and have started the Agent(s) that are installed in each machine.
- $\bullet$  No application is connected to the target Nebula Graph cluster.
- Make sure the number of the storage servers in the original cluster is the same as that in the target cluster.

# Steps

In the BR Enterprise Edition installation directory, follow these steps:

- 1. View backup files.
- List the backup files in a local path.

```
./br show --storage local://<storage_path>
```

For example, you can view the backup files in the local /backup/ path with the following command.

./br show	storage "local:,	//bac	kup/"	-+	+		.+		
į	NAME	į	CREATE TIME	SPACES	F	JLL BACKUP	İ	ALL SPACES	BASE BACKUP NAME
				basketballplayer   basketballplayer,br		rue alse		true true	BACKUP_2022_08_11_06_12_43

• Lists backups in / under the nebula-br-test bucket of the S3 protocol-compliant object storage service.

```
./br show --s3.access_key QImbbGDjfQEYxxxx --s3.secret_key dVSJZfl7tnoFq7Z5zt6sfYnvi63bxxxx --s3.region us-east-1 --storage s3://nebula-br-test/ --s3.endpoint http://192.168.8.xxx:9000/
```

- 2. Restore data.
- · Restore data based on local backups.

```
./br restore full --meta <ip_address> --storage <storage_path> --name <backup_name>
```

For example, restore the BACKUP\_2022\_08\_11\_09\_11\_07 file data from the local /backup/ path to the cluster with the meta address 192.168.8.129:9559:

```
./br restore full --meta "192.168.8.129:9559" --storage "local:///backup/" --name BACKUP_2022_08_11_09_11_07
```

• Restore data based on cloud backups.

Restore the BACKUP\_2022\_08\_12\_07\_37\_02 backup data from the / path under the nebula-br-test bucket of the S3-compliant object storage service to the cluster with the meta address 192.168.8.129:9559, and specify the restoration log path.

```
./br restore full --meta 192.168.8.129:9559 --s3.accesskey QImbbGDjfQEYxxxx --s3.secretkey dVSJZfl7tnoFq7Z5zt6sfYnvi63bxxxx --s3.region us-east-1 --storage s3://nebula-br-test/ --s3.endpoint http://192.168.8.xxx:9000/ --log "3.log" --name BACKUP_2022_08_12_07_37_02
```

If the following information is returned, the data is restored successfully.

Restore succeed.

#### Q Note

If the data restoration fails, BR Enterprise Edition automatically performs a rollback and the cluster's data is automatically recovered back to the pre-restoration data.

3. Run the following command to clean temporary files. This command will clean up temporary files in the cluster or external storage, and you can also use this command to clean up old backup directories. The example is as follows.

- 508/1069 - 2023 Vesoft Inc.



By default, BR automatically cleans up the temporary files when an error occurs during data restoration. If the automatic cleanup fails, you need to execute the command to clean up the temporary files manually.

• Clear a local backup directory.

./br cleanup --meta 192.168.8.129:9559 --storage "local:///backup/" --name BACKUP\_2022\_08\_11\_09\_11\_07

• Clear the backup directory in the cloud storage service.

./br cleanup --meta 192.168.8.129:9559 --s3.accesskey QImbbGDjfQEYxxxx --s3.secretkey dVSJZFL7tnoFq7Z5zt6sfYnvi63bxxxx --s3.region us-east-1 --storage s3://nebula-br-test/ --s3.endpoint http://192.168.8.xxx:9000/ --name BACKUP\_2022\_08\_12\_07\_37\_02

#### Options

The following options are for data restorations.

Option	Data type	Required	Default value	Description
-h,-help	-	No	-	View more information about BR commands.
debug	-	No	None	View more information for BR logs.
log	string	No	"br.log"	The path of BR logs.
concurrency	int	No	None	Used to control the number of concurrent file downloads during data restoration. The default value is ${\bf 5}$ .
meta	string	Yes	None	The IP address and port number of any Meta service in the cluster.
name	string	Yes	None	The name of the backup file.
storage	string	Yes	None	The storage URL of the backup is to be restored. The format is <schema>://<storage_path>. schema: Two options, s3 and local. When schema is s3, you still need to add options of s3.access_key \ s3.endpoint \ s3.region, and s3.secret_key. <storage_path>: The storage path of the backup to be restored.</storage_path></storage_path></schema>
 s3.access_key	string	No	None	The Access Key ID that is used to identify a user.
s3.endpoint	string	No	None	The S3 endpoint URL. You need to specify the HTTP or HTTPS scheme explicitly.
s3.region	string	No	None	The physical location of a data center.
 s3.secret_key	string	No	None	The Access Key Secret that is used to verify the identity of the user.

# View restoration progress

The log file br.log of BR can be viewed in the installation directory. The log file will record the restoration progress and will look like as follows:

```
{"level":"info","msg":"download storaged partition finished, progress: 1/20","time":"2023-03-15T02:16:43.430Z"}
{"level":"info","msg":"download storaged partition finished, progress: 2/20","time":"2023-03-15T02:16:43.431Z"}
{"level":"info","msg":"download storaged partition finished, progress: 3/20","time":"2023-03-15T02:16:43.763Z"}
```

Last update: February 19, 2024

# 9.3 Backup and restore data with snapshots

NebulaGraph supports using snapshots to back up and restore data. When data loss or misoperation occurs, the data will be restored through the snapshot.

#### 9.3.1 Prerequisites

NebulaGraph authentication is disabled by default. In this case, all users can use the snapshot feature.

If authentication is enabled, only the GOD role user can use the snapshot feature. For more information about roles, see Roles and privileges.

#### 9.3.2 Precautions

- To prevent data loss, create a snapshot as soon as the system structure changes, for example, after operations such as ADD HOST, DROP HOST, CREATE SPACE, DROP SPACE, and BALANCE are performed.
- NebulaGraph cannot automatically delete the invalid files created by a failed snapshot task. You have to manually delete them by using DROP SNAPSHOT.
- Customizing the storage path for snapshots is not supported for now. The default path is /usr/local/nebula/data.

#### 9.3.3 Snapshot form and path

NebulaGraph snapshots are stored in the form of directories with names like SNAPSHOT\_2021\_03\_09\_08\_43\_12. The suffix 2021\_03\_09\_08\_43\_12 is generated automatically based on the creation time (UTC).

When a snapshot is created, snapshot directories will be automatically created in the <code>checkpoints</code> directory on the leader Meta server and each Storage server.

To fast locate the path where the snapshots are stored, you can use the Linux command find. For example:

```
$ find |grep 'SNAPSHOT_2021_03_09_08_43_12'
./data/meta2/nebula/0/checkpoints/SNAPSHOT_2021_03_09_08_43_12
./data/meta2/nebula/0/checkpoints/SNAPSHOT_2021_03_09_08_43_12/data
./data/meta2/nebula/0/checkpoints/SNAPSHOT_2021_03_09_08_43_12/data/000081.sst
...
```

#### 9.3.4 Create snapshots

Run CREATE SNAPSHOT to create a snapshot for all the graph spaces based on the current time for NebulaGraph. Creating a snapshot for a specific graph space is not supported yet.



If the creation fails, delete the snapshot and try again.

```
nebula> CREATE SNAPSHOT;
```

#### 9.3.5 View snapshots

To view all existing snapshots, run  $\mbox{\sc Show SNAPSHOTS}$  .

```
| "SNAPSHOT_2021_03_09_09_10_52" | "VALID" | "127.0.0.1:9779" |
+------+
```

The parameters in the return information are described as follows.

Parameter	Description
Name	The name of the snapshot directory. The prefix SMAPSHOT indicates that the file is a snapshot file, and the suffix indicates the time the snapshot was created (UTC).
Status	The status of the snapshot. VALID indicates that the creation succeeded, while INVALID indicates that it failed.
Hosts	IP addresses and ports of all Storage servers at the time the snapshot was created.

### 9.3.6 Delete snapshots

To delete a snapshot with the given name, run DROP SNAPSHOT.

```
DROP SNAPSHOT <snapshot_name>;
```

#### Example:

# 9.3.7 Restore data with snapshots



When you restore data with snapshots, make sure that the graph spaces backed up in the snapshot have not been dropped. Otherwise, the data of the graph spaces cannot be restored.

Currently, there is no command to restore data with snapshots. You need to manually copy the snapshot file to the corresponding folder, or you can make it by using a shell script. The logic implements as follows:

1. After the snapshot is created, the checkpoints directory is generated in the installation directory of the leader Meta server and all Storage servers, and saves the created snapshot. Taking this topic as an example, when there are two graph spaces, the snapshots created are saved in /usr/local/nebula/data/meta/nebula/0/checkpoints, /usr/local/nebula/data/storage/ nebula/3/checkpoints and /usr/local/nebula/data/storage/nebula/4/checkpoints.

```
$ ls /usr/local/nebula/data/meta/nebula/0/checkpoints/
SNAPSHOT_2021_03_09_09_10_52
$ ls /usr/local/nebula/data/storage/nebula/3/checkpoints/
SNAPSHOT_2021_03_09_09_10_52
$ ls /usr/local/nebula/data/storage/nebula/4/checkpoints/
SNAPSHOT_2021_03_09_09_10_52
```

2. To restore the lost data through snapshots, you can take a snapshot at an appropriate time, copy the folders data and wal in the corresponding snapshot directory to its parent directory (at the same level with <code>checkpoints</code>) to overwrite the previous <code>data</code> and <code>wal</code>, and then restart the cluster.



The data and wal directories of all Meta servers should be overwritten at the same time. Otherwise, the new leader Meta server will use the latest Meta data after a cluster is restarted.

Last update: February 19, 2024

# 10. Synchronization & Migration

# 10.1 BALANCE syntax

The BALANCE statements support the load balancing operations of the NebulaGraph Storage services. For more information about storage load balancing and examples for using the BALANCE statements, see Storage load balance.

The BALANCE statements are listed as follows.

Syntax	Description
BALANCE LEADER	Starts a job to balance the distribution of all the storage leaders in graph spaces. It returns the job ID.

For details about how to view, stop, and restart a job, see  ${\tt Job}$  manager and the  ${\tt JOB}$  statements.

Last update: February 19, 2024

- 514/1069 - 2023 Vesoft Inc.

# 10.2 Synchronize between two clusters

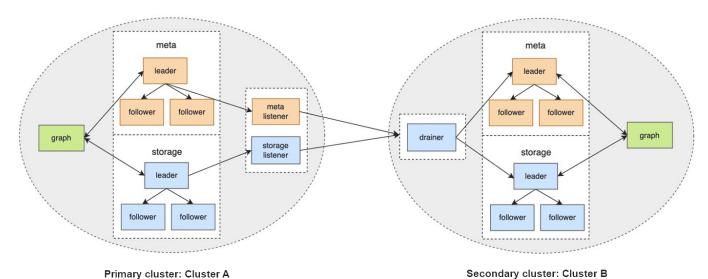
NebulaGraph supports data synchronization from a primary cluster to a secondary cluster in almost real-time. It applies to scenarios such as disaster recovery and load balancing, and helps reduce the risk of data loss and enhance data security.



This feature applies to the Enterprise Edition only.

# 10.2.1 Synchronization workflow

The synchronization works as follows:



- 1. The primary cluster sends any data written into it to the Meta listener or the Storage listener in the form of WALs or snapshots.
- 2. The listener sends the data to the drainer in the form of WALs.
- 3. The drainer sends the data to the partitions of the secondary cluster through the Meta client or the Storage client.

#### 10.2.2 Applicable Scenarios

- Remote disaster recovery: Data synchronization enables cross-data-center or cross-city disaster recovery.
- Data migration: The migration can be implemented by synchronizing data and then switching cluster roles, without stopping the service.
- Read/Write splitting: Enable only writing on the primary cluster and only reading on the secondary cluster to lower the system load, and improve stability and usability.

#### 10.2.3 Precautions

- Make sure that the primary and secondary clusters are deployed in the same NebulaGraph version. Otherwise, the synchronization will fail.
- The synchronization is based on graph spaces, i.e., from one graph space in the primary cluster to another in the secondary cluster.
- About the synchronization topology, NebulaGraph:
- Supports synchronizing from one primary cluster to one secondary cluster, but not multiple primary clusters to one secondary cluster.
- Supports chained synchronization but not synchronization from one primary cluster to multiple secondary clusters directly. An example of chained synchronization is from cluster A to cluster B, and then cluster B to cluster C.
- The synchronization is implemented asynchronously, but with low latency.
- The Meta listener listens to the Meta Service and the Storage listener listens to the Storage Service. Do not mix them up.
- One graph space can have one Meta listener and one to multiple Storage listeners. These listeners can work with one to multiple drainers:
- · One listener with one drainer.
- Multiple listeners with one drainer.
- Multiple listeners with multiple drainers.
- The machines where the listeners and drainers run must have enough disk space to store the WAL or snapshot files.
- If the target graph space in the secondary cluster has data before the synchronization starts, data conflicts or inconsistencies may happen during the synchronization. It is recommended to keep the target graph space empty.
- It is recommended to use the NebulaGraph root user with the God privileges to perform the cluster data synchronization. The required user roles for each synchronization command are different. For details, see the **Role permission requirements** section at the end of this topic.
- During the synchronization, do not perform data recovery (backup recovery and snapshot recovery) operations on the primary cluster at the same time. Otherwise, the synchronization will fail.

# 10.2.4 Prerequisites

- Prepare at least two machines to deploy the primary and secondary clusters, the listeners, and the drainer.

  The listener and drainer can be deployed in a standalone way, or on the machines hosting the primary and secondary clusters.

  The latter way can increase the machine load and decrease the service performance.
- Prepare the license file for the NebulaGraph Enterprise Edition.

#### 10.2.5 Test environment

The test environment for the operation example in this topic is as follows:

- The primary cluster runs on the machine with the IP address 192.168.10.101. The cluster has one nebula-graphd process, one nebula-metad process, and one nebula-storaged process.
- The secondary cluster runs on the machine with the IP address 192.168.10.102. The cluster has one nebula-graphd process, one nebula-metad process, and one nebula-storaged process.



The primary and secondary clusters can have different cluster specifications, such as different numbers of machines, service processes, and data partitions.

- The processes for the Meta and Storage listeners run on the machine with the IP address 192.168.10.103.
- The process for the drainer runs on the machine with the IP address 192.168.10.104.

#### 10.2.6 Steps

#### Step 1: Set up the clusters, listeners, and drainer

1. Install NebulaGraph on all the machines.

For installation instructions, see Install NebulaGraph.

2. Modify the configuration files on all the machines.

#### Q Note

For newly installed services, remove the suffix .default or .production of a configuration template file in the conf directory to make it take effect

- On the primary and secondary cluster machines, modify nebula-graphd.conf, nebula-metad.conf, and nebula-storaged.conf. In all three files, set real IP addresses for local\_ip instead of 127.0.0.1, and set the IP addresses and ports for their own nebula-metad processes as the meta\_server\_addrs values. In nebula-graphd.conf, set enable\_authorize=true.
- On the primary cluster, set --snapshot\_send\_files=false in both the nebula-storaged.conf file and the nebula-metad.conf file.
- On the Meta listener machine, modify nebula-metad-listener.conf. Set the IP addresses and ports of the **primary cluster's** nebula-metad processes for meta\_server\_addrs, and those of the listener process for meta\_sync\_tistener.
- On the Storage listener machine, modify nebula-storaged-listener.conf. Set the IP addresses and ports of the **primary cluster's** nebula-metad processes for meta\_server\_addrs.
- On the drainer machine, modify nebula-drainerd.conf . Set the IP addresses and ports of the **secondary cluster's** nebula-metad processes for meta\_server\_addrs .

For more information about the configurations, see Configurations.

- 3. On the machines of the primary cluster, secondary cluster, and listeners, upload the license files into the share/resources/ directories in the NebulaGraph installation directories.
- 4. Go to the NebulaGraph installation directories on the machines and start the needed services.
- $\bullet$  On the primary and secondary machines, run  $\,$  sudo scripts/nebula.service start all .
- $\bullet \ \ On \ the \ \ Meta \ listener \ machine, \ run \ \ sudo \ bin/nebula-metad \ --flag file \ etc/nebula-metad-listener.conf.$
- On the Storage listener machine, run sudo bin/nebula-storaged --flagfile etc/nebula-storaged-listener.conf.
- On the drainer machine, run sudo scripts/nebula-drainerd.service start.
- 5. Log into the primary cluster, add the Storage hosts, and check the status of the listeners.

- 517/1069 - 2023 Vesoft Inc.

6. Log into the secondary cluster, add the Storage hosts, and check the status of the drainer.

# Step 2: Set up the synchronization

1. Log into the primary cluster and create a graph space  $\,$  basketballplayer .

```
nebula> CREATE SPACE basketballplayer(partition_num=15, \
    replica_factor=1, \
    vid_type=fixed_string(30));
```

2. Use the graph space basketballplayer and register the drainer service.



To register multiple drainer services, run the command such as SIGN IN DRAINER SERVICE(192.168.8.x:9889),(192.168.8.x:9889).

3. Configure the listener service.

PartId	Type	Host	SpaceName	Status
0	"SYNC"	""192.168.10.103":9569"	   "replication basketballplayer"	"ONLINE"
1	"SYNC"	""192.168.10.103":9789"	"replication basketballplayer"	"ONLINE"
2	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
3	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
4	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
5	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
6	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
7	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
8	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
9	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
10	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
11	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
12	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
13	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
14	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
15	"SYNC"	""192.168.10.103":9789"	"replication_basketballplayer"	"ONLINE"
+		++	+	+



To configure multiple storage listener services, run the command such as ADD LISTENER SYNC META 192.168.10.xxx:9569 STORAGE 192.168.10.xxx: 9789,192.168.10.xxx:9789 TO SPACE replication\_basketballplayer.

4. Log into the secondary cluster and create graph space replication\_basketballplayer.

```
nebula> CREATE SPACE replication_basketballplayer(partition_num=15, \
    replica_factor=1, \
    vid_type=fixed_string(30));
```

5. Use replication\_basketballplayer and configure the drainer service.



To configure multiple drainer services, run the command such as ADD DRAINER 192.168.8. $\times$ :9889,192.168.8. $\times$ :9889.

6. Set the target graph space replication\_basketballplayer as read-only to avoid data inconsistency.



This step only sets the target graph space, not other graph spaces.

# Step 3: Validate the data

1. Log into the primary cluster, create the schema, and insert data.

```
nebula> USE basketballplayer;
nebula> CREATE TAG player(name string, age int);
nebula> CREATE EDGE follow(degree int);
nebula> INSERT VERTEX player(name, age) VALUES "player100":("Tim Duncan", 42);
nebula> INSERT VERTEX player(name, age) VALUES "player101":("Tony Parker", 36);
nebula> INSERT EDGE follow(degree) VALUES "player101" -> "player100":(95);
```

2. Log into the secondary cluster and validate the data.

```
nebula> USE replication_basketballplayer;
nebula> SUBMIT JOB STATS;
nebula> SHOW STATS;
Type
          Name
                      Count
 "Tag"
            "player"
                       1 2
  "Edge"
            "follow"
  "Space"
            "vertices"
                        2
  "Space" | "edges"
nebula> FETCH PROP ON player "player100" \
       YIELD properties(vertex);
| properties(VERTEX)
| {age: 42, name: "Tim Duncan"}
nebula> GO FROM "player101" OVER follow \
       YIELD dst(edge);
| dst(EDGE)
| "player100"
```

# 10.2.7 Stop/Restart data synchronization

The listener continuously sends the WALs to the drainer during data synchronization.

To stop data synchronization, run the STOP SYNC command. The listener stops sending data to the drainer.

To restart data synchronization, run the RESTART SYNC command. The listener sends the data accumulated during the period when the synchronization is stopped to the drainer. If the WALs are lost, the listener pulls the snapshot from the primary cluster and synchronizes data again.

#### 10.2.8 View the status of inter-cluster data synchronization

When data is written to the primary cluster, you can check the status of inter-cluster data synchronization and tell whether data synchronization is normal.

# Check the status of synchronized data in the primary cluster

You can execute the SHOW SYNC STATUS command in the primary cluster to view the status of the data sent from the primary cluster to the secondary cluster. SHOW SYNC STATUS gets the information of data synchronization status between clusters in real-time, and sends synchronized data to the secondary cluster only when the primary cluster has written successfully.

Examples are as follows.

```
nebula> INSERT VERTEX player(name,age) VALUES "player102":("LaMarcus Aldridge", 33); nebula> INSERT VERTEX player(name,age) VALUES "player102":("LaMarcus Aldridge", 33);
nebula= INSERT VERTEX player(name,age) VALUES "player103":("Rudy Gay", 32);
nebula= INSERT VERTEX player(name,age) VALUES "player104":("Marco Belinelli", 32);
// Check the status of data synchronization in the current cluster (the returned result indicates that data is being sent to the secondary cluster).
nebula> SHOW SYNC STATUS;
  PartId | Sync Status | LogId Lag | Time Latency
  0
              "ONLTNE"
                               1 0
                                               1 0
  1
               "ONLINE"
                               0
                                                0
  2
              "ONLINE"
  3
               "ONLINE"
                                                 0
              "ONLINE"
                               0
                                               0
```

5	"ONLINE"	1	46242122
6	"ONLINE"	0	0
7	"ONLINE"	0	0
8	"ONLINE"	0	0
9	"ONLINE"	0	0
10	"ONLINE"	0	0
11	"ONLINE"	0	0
12	"ONLINE"	0	0
13	"ONLINE"	0	0
14	"ONLINE"	0	0
15	"ONLINE"	0	0
+	+	+	+
PartId	HOW SYNC STATU 	+   LogId Lag	Time Latence
0	"ONLINE"	+   0	+   0
1	"ONLINE"	0	0
2	"ONLINE"	0	0
3	"ONLINE"	0	0
4	"ONLINE"	0	0
5	"ONLINE"	0	0
6	"ONLINE"	0	0
7	"ONLINE"	0	0
8	"ONLINE"	0	0
9	"ONLINE"	0	0
10	"ONLINE"	0	0
	"ONLINE"	0	0
11			
12	"ONLINE"	0	0
12   13	"ONLINE"	0	0
12		1	

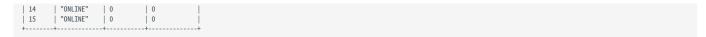
After executing the SHOW SYNC STATUS command, the parameters in the returned result are described as follows.

Parameter	Description
PartId	The partition ID in the specified graph space in the primary cluster. The Meta data to be synchronized by Meta listener is located in the partition 0. The Storage data to be synchronized by Storage listener is located in other partitions.
Sync Status	Indicates the status of the listener service.  When the listener is ONLINE, it continuously sends data to the drainer service.  When the listener is OFFLINE, it stops sending data to the drainer.
LogId Lag	Indicates the difference between Log IDs, that is how many logs are still sent to the secondary cluster from the corresponding partition of the primary cluster.  The value 0 indicates that there are no logs to be sent in the corresponding partition of the primary cluster.
Time Latency	The difference between the timestamp in the WAL of the last log to be sent and the timestamp in the WAL of the last log that has been sent in the corresponding partition of the primary cluster.  The value 0 indicates that data has been sent to the secondary cluster.  Unit: Millisecond.

# Check the status of synchronized data in the secondary cluster

In the secondary cluster, run SHOW DRAINER SYNC STATUS to view the status of synchronizing data to the Meta and Storage services in the secondary cluster.

nobulas C	HOW DRAINER SYN	IC CTATUC.		
			+	-+
PartId	Sync Status	LogId Lag	Time Latency	
+	++	·	+	-
0	ONLINE"	0	0	
1	ONLINE"	0	0	
2	ONLINE"	0	0	
3	ONLINE"	0	0	
4	"ONLINE"	0	0	
5	"ONLINE"	0	0	
6	"ONLINE"	0	0	
7	"ONLINE"	0	0	1
8	"ONLINE"	0	0	ĺ
9	"ONLINE"	0	0	1
10	"ONLINE"	0	0	1
11	"ONLINE"	0	0	ĺ
12	"ONLINE"	0	0	ĺ
13	ONLINE"	0	0	



After executing SHOW DRAINER SYNC STATUS, the parameters in the returned result are described as follows.

Parameter	Description
PartId	The partition ID in the specified graph space in the primary cluster. The partition $0$ is where the Meta data to be synchronized is located. The Storage data is located in other partitions.
Sync Status	Indicates the status of the drainer service.  When drainer is ONLINE, it continuously sends WAL to metaClient/storageClient in the secondary cluster for data synchronization.  When drainer is OFFLINE, it stops sending WAL to metaClient/storageClient in the secondary cluster for data synchronization.
LogId Lag	Indicates the difference between Log IDs, that is how many logs are still sent to metaClient/storageClient from the corresponding drainer partition in the secondary cluster.  The value 0 indicates that there are no logs to be synchronized in the corresponding drainer partition.
Time Latency	The difference between the timestamp in the WAL of the newest log received by the corresponding drainer partition between the timestamp in the WAL of the last log that has been synchronized to <code>metaClient/storageClient</code> in the secondary cluster.  The value <code>0</code> indicates that drainer partition data has been sent to <code>metaClient/storageClient</code> .  Unit: Millisecond.

# 10.2.9 Switch between primary and secondary clusters

To migrate data or implement disaster recovery, manually switch between the primary and secondary clusters.

#### O Note

Before the switching, set up a listener for the new primary cluster, and a drainer for the new secondary cluster. In the following example, the listener has IP address 192.168.10.105 and drainer 192.168.10.106.

1. Log into the old primary cluster and set the working graph space as read-only to avoid data inconsistency.

```
nebula> USE basketballplayer;
nebula> SET VARIABLES read_only=true;
```

- 2. Check whether the data in the old primary cluster has been synchronized to the old secondary cluster. Make sure the data in the old primary cluster has been synchronized to the old secondary cluster.
- a. In the old primary cluster, view the status of the data sent from the old primary cluster to the old secondary cluster.

```
nebula> SHOW SYNC STATUS
```

a. Log into the old secondary cluster and then view the status of synchronizing data to the Meta and Storage services.

```
nebula> USE replication_basketballplayer;
nebula> SHOW DRAINER SYNC STATUS;
```

When the values of LogId Lag and Time Latency in the returned results in both old primary and secondary clusters are 0, the data synchronization is complete.

3. In the old secondary cluster, disable read-only for the working graph space.

nebula> SET VARIABLES read\_only=false;



If there is business data to be written, you can now write the business data to the old secondary cluster (the new primary cluster).

4. In the old secondary cluster, remove the old drainer service.

```
nebula> REMOVE DRAINER;
```

5. Log into the old primary cluster, disable read-only, sign out the drainer, and remove the listener.

```
nebula> USE basketballplayer;
// Disable read-only for the working graph space, otherwise adding drainer fails.
nebula> SET VARIABLES read_only=false;
nebula> SIGN OUT DRAINER SERVICE;
nebula> REMOVE ITSTENER SYNC:
```

6. In the old primary cluster, change the old primary cluster to the new secondary cluster by adding the new drainer service and setting the working graph space as read-only.



Ensure that the new drainer service is deployed and started for the new secondary cluster.

```
nebula> ADD DRAINER 192.168.10.106:9889;
nebula> SET VARIABLES read only=true;
```

7. Log into the old secondary cluster and change the old secondary cluster to the new primary cluster.



Ensure that the new meta listener and storage listener services are deployed and started for the new primary cluster.

```
nebula> SIGN IN DRAINER SERVICE(192.168.10.106:9889);
nebula> ADD LISTENER SYNC META 192.168.10.105:9569 STORAGE 192.168.10.105:9789 TO SPACE basketballplayer;
```

The primary-secondary cluster switch is now complete.

- 524/1069 - 2023 Vesoft Inc.

#### 10.2.10 Role permission requirements

The required user roles for each synchronization command are different. The required roles for each command are as follows (A check mark indicates that the role has the permission).

Command	God	Admin	DBA	User	Guest
SIGN IN / SIGN OUT DRAINER SERVICE	$\checkmark$				
ADD / REMOVE LISTENER SYNC	$\checkmark$	$\sqrt{}$	$\checkmark$		
SHOW DRAINER CLIENTS	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
SHOW LISTENER SYNC	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$
ADD / REMOVE DRAINER	$\checkmark$	$\checkmark$	$\checkmark$		
SET VARIABLES read_only	$\checkmark$				
SHOW DRAINERS	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$

# 10.2.11 FAQ

#### Can the pre-existent data in the primary cluster be synchronized to the secondary cluster?

Yes. After receiving the WAL from the listener, if the drainer finds that the data to be updated does not exist in the secondary cluster, it starts the synchronization of the complete data set.

#### Will the pre-existent data in the secondary cluster affect the synchronization?

If the pre-existent data in the secondary cluster is a subset of the data in the primary cluster, the data in the primary and secondary clusters will eventually become consistent through synchronization. If there is any pre-existent data (not a subset of the data in the primary cluster) in the secondary cluster before the synchronization, the data may be lost after the synchronization. It is recommended to use a secondary cluster without data for synchronization.

### Will the pre-existent schema information in the secondary cluster affect the synchronization?

The pre-existent schema information must not conflict with the schema of the primary cluster. Otherwise, it will be overwritten, and related data in the secondary cluster might become invalid.

#### Should the number of machines, replicas, and partitions in the primary and secondary clusters be the same?

No. The synchronization is based on graph spaces, not other elements such as partitions and replicas. The primary and secondary clusters do not need to have the exact specifications.

#### Does altering the schema in the primary cluster affect the synchronization?

Altering the schema may increase the synchronization latency.

The schema data is synchronized through the Meta listener, while the vertex/edge data is through the Storage listener. When synchronizing the vertex/edge data, the system checks the schema version of the data. If the system finds that the version number of the schema is greater than that in the secondary cluster, it pauses the vertex/edge data update, and updates the schema data first.

- 525/1069 - 2023 Vesoft Inc.

#### How to deal with synchronization failures?

Fix the problems in the cluster, and then the synchronization will be automatically restored.

- If problems have happened in the primary cluster, the synchronization continues when the problems are fixed and the primary cluster restarts.
- If problems have happened in the secondary cluster, listeners, or drainers, when the problems are fixed, the services that had the problems will receive the WALs accumulated from its upstream and the synchronization will continue working. If the faulty machine is replaced with a new one, all the data of the synchronization services on the faulty machine must be copied to the new machine. Otherwise, the synchronization of the complete data set will start automatically.

#### How to check the data synchronization status and progress?

You can run SHOW SYNC STATUS to check the status of the data sent by the primary cluster and run SHOW DRAINER SYNC STATUS to check the status of the data received by the secondary cluster. If all the data is sent successfully from the primary cluster and all the data is received successfully by the secondary cluster, the data synchronization is completed.

#### My WAL log files has expired and will it affect the cluster synchronization?

Expired WAL files (beyond the time set by --wal-ttl) will cause unsynchronization of cluster data. You can manually add --snapshot\_send\_files=false to the configuration files of the Meta and Storage services to synchronize data. After updating the configuration files, you need to restart the services. For more information about the configuration files, see Configuration Files.

Last update: February 19, 2024

# 11. Practices

# 11.1 Compaction

This topic gives some information about compaction.

In NebulaGraph, Compaction is the most important background process and has an important effect on performance.

Compaction reads the data that is written on the hard disk, then re-organizes the data structure and the indexes, and then writes back to the hard disk. The read performance can increase by times after compaction. Thus, to get high read performance, trigger compaction (full compaction) manually when writing a large amount of data into Nebula Graph.



Note that compaction leads to long-time hard disk IO. We suggest that users do compaction during off-peak hours (for example, early morning).

NebulaGraph has two types of compaction: automatic compaction and full compaction.

### 11.1.1 Automatic compaction

Automatic compaction is automatically triggered when the system reads data, writes data, or the system restarts. The read performance can increase in a short time. Automatic compaction is enabled by default. But once triggered during peak hours, it can cause unexpected IO occupancy that has an unwanted effect on the performance.

### 11.1.2 Full compaction

Full compaction enables large-scale background operations for a graph space such as merging files, deleting the data expired by TTL. This operation needs to be initiated manually. Use the following statements to enable full compaction:



We recommend you to do the full compaction during off-peak hours because full compaction has a lot of IO operations.

nebula> USE <your\_graph\_space>;
nebula> SUBMIT JOB COMPACT;

The preceding statement returns the job ID. To show the compaction progress, use the following statement:

nebula> SHOW JOB <job\_id>;

- 527/1069 - 2023 Vesoft Inc.

### 11.1.3 Operation suggestions

These are some operation suggestions to keep Nebula Graph performing well.

- After data import is done, run SUBMIT JOB COMPACT.
- Run SUBMIT JOB COMPACT periodically during off-peak hours (e.g. early morning).
- To control the write traffic limitation for compactions, set the following parameter in the nebula-storaged.conf configuration file.



This parameter limits the rate of all writes including normal writes and compaction writes.

```
# Limit the write rate to 20MB/s.
--rocksdb_rate_Limit=20 (in MB/s)
```

# 11.1.4 FAQ

# "Where are the logs related to Compaction stored?"

By default, the logs are stored under the LOG file in the  $/usr/local/nebula/data/storage/nebula/{1}/data/$  directory, or similar to LOG.old. 1625797988509303 . You can find the following content.

** Compaction Stats [default] **																		
Level	Files	Size	Score	Read(GB)	Rn(GB)	Rnp1(GB)	Write(GB)	Wnew(GB)	Moved(GB)	W-Amp	Rd(MB/s)	Wr(MB/s)	Comp(sec)	CompMergeCPU(sec)	Comp(cnt)	Avg(sec)	KeyIn K	CeyDrop
LO	2/0	2.46 KB	0.5	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.53	0.51	2	0.264	0	0
Sum	2/0	2.46 KB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.53	0.51	2	0.264	0	0
Int	0/0	0.00 KB	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.00	0	0.000	0	0

If the number of LO files is large, the read performance will be greatly affected and compaction can be triggered.

#### "Can I do full compactions for multiple graph spaces at the same time?"

Yes, you can. But the IO is much larger at this time and the efficiency may be affected.

#### "How much time does it take for full compactions?"

When <code>rocksdb\_rate\_limit</code> is set to 20, you can estimate the full compaction time by dividing the hard disk usage by the <code>rocksdb\_rate\_limit</code>. If you do not set the <code>rocksdb\_rate\_limit</code> value, the empirical value is around 50 MB/s.

#### "Can I modify --rocksdb\_rate\_Limit dynamically?"

No, you cannot.

#### "Can I stop a full compaction after it starts?"

No, you cannot. When you start a full compaction, you have to wait till it is done. This is the limitation of RocksDB.

Last update: February 19, 2024

- 528/1069 - 2023 Vesoft Inc.

# 11.2 Storage load balance

You can use the BALANCE statement to balance the distribution of partitions and Raft leaders, or clear some Storage servers for easy maintenance. For details, see BALANCE.



The BALANCE commands migrate data and balance the distribution of partitions by creating and executing a set of subtasks. **DO NOT** stop any machine in the cluster or change its IP address until all the subtasks finish. Otherwise, the follow-up subtasks fail.

# 11.2.1 Balance partition distribution



Only available for the NebulaGraph Enterprise Edition.



If the current graph space already has a BALANCE DATA job in the FAILED status, you can restore the FAILED job, but cannot start a new BALANCE DATA job. If the job continues to fail, manually stop it, and then you can start a new one.

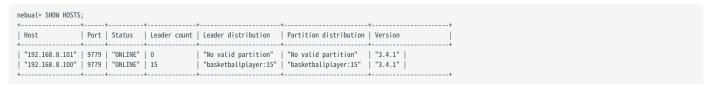
The BALANCE DATA commands starts a job to balance the distribution of storage partitions in the current graph space by creating and executing a set of subtasks.

- 529/1069 - 2023 Vesoft Inc.

#### **Examples**

After you add new storage hosts into the cluster, no partition is deployed on the new hosts.

1. Run SHOW HOSTS to check the partition distribution.



2. Enter the graph space basketballplayer, and execute the command BALANCE DATA to balance the distribution of storage partitions.

```
nebula> USE basketballplayer;
nebula> BALANCE DATA;
+------+
| New Job Id |
+-------+
| 25 |
+-------+
```

3. The job ID is returned after running BALANCE DATA. Run SHOW JOB <job\_id> to check the status of the job.

nebula> SHOW JOB 25;					
Job Id(spaceId:partId)			Start Time	Stop Time	State
25	"DATA_BALANCE" "Succeeded:0"	"FINISHED"		2023-01-17T06:24:35.000000   "Invalid:0"	

4. When all the subtasks succeed, the load balancing process finishes. Run SHOW HOSTS again to make sure the partition distribution is balanced.



BALANCE DATA does not balance the leader distribution. For more information, see Balance leader distribution.

nebula> SHOW HOSTS	5;					
+	+	+	+	+	+	+
Host	Port   Status	Leader count	Leader distribution	Partition distribution	Version	
+	+	+	+	+	+	+
"192.168.8.101"	9779   "ONLINE"	7	"basketballplayer:7"	"basketballplayer:7"	"3.4.1"	
"192.168.8.100"	9779   "ONLINE"	8	"basketballplayer:8"	"basketballplayer:8"	"3.4.1"	
+	+	+	+	+	+	+

If any subtask fails, run RECOVER JOB <job\_id> to recover the failed jobs. If redoing load balancing does not solve the problem, ask for help in the NebulaGraph community.

# Stop data balancing

To stop a balance job, run  $STOP JOB < job_id>$ .

- If no balance job is running, an error is returned.
- If a balance job is running, Job stopped is returned.



STOP JOB <job\_id> does not stop the running subtasks but cancels all follow-up subtasks. The status of follow-up subtasks is set to INVALID. The status of ongoing subtasks is set to SUCCEEDED or FAILED based on the result. You can run the SHOW JOB <job\_id> command to check the stopped job status.

Once all the subtasks are finished or stopped, you can run RECOVER JOB <  $job_id$ > again to balance the partitions again, the subtasks continue to be executed in the original state.

#### Restore a balance job

To restore a balance job in the FAILED or STOPPED status, run RECOVER JOB <job\_id>.

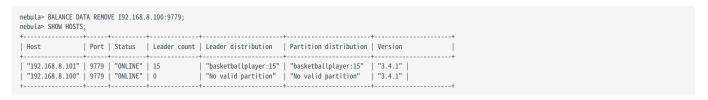


For a STOPPED BALANCE DATA job, NebulaGraph detects whether the same type of FAILED jobs or FINISHED jobs have been created since the start time of the job. If so, the STOPPED job cannot be restored. For example, if chronologically there are STOPPED job1, FINISHED job2, and STOPPED Job3, only job3 can be restored, and job1 cannot.

#### Migrate partition

To migrate specified partitions and scale in the cluster, you can run BALANCE DATA REMOVE <ip:port>:<port>:..].

For example, to migrate the partitions in server 192.168.8.100:9779, the command as following:





This command migrates partitions to other storage hosts but does not delete the current storage host from the cluster. To delete the Storage hosts from cluster, see Manage Storage hosts.

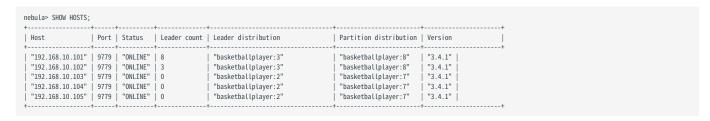
# 11.2.2 Balance leader distribution

To balance the raft leaders, run BALANCE LEADER.

#### Example

nebula> BALANCE LEADER;

Run  $\mbox{SHOW HOSTS}$  to check the balance result.





In NebulaGraph 3.4.1, switching leaders will cause a large number of short-term request errors (Storage Error E\_RPC\_FAILURE). For solutions, see FAQ.

Last update: February 19, 2024

- 531/1069 - 2023 Vesoft Inc.

# 11.3 Graph data modeling suggestions

This topic provides general suggestions for modeling data in NebulaGraph.



The following suggestions may not apply to some special scenarios. In these cases, find help in the NebulaGraph community.

### 11.3.1 Model for performance

There is no perfect method to model in Nebula Graph. Graph modeling depends on the questions that you want to know from the data. Your data drives your graph model. Graph data modeling is intuitive and convenient. Create your data model based on your business model. Test your model and gradually optimize it to fit your business. To get better performance, you can change or redesign your model multiple times.

#### Design and evaluate the most important queries

Usually, various types of queries are validated in test scenarios to assess the overall capabilities of the system. However, in most production scenarios, there are not many types of frequently used queries. You can optimize the data model based on key queries selected according to the Pareto (80/20) principle.

#### Full-graph scanning avoidance

Graph traversal can be performed after one or more vertices/edges are located through property indexes or VIDs. But for some query patterns, such as subgraph and path query patterns, the source vertex or edge of the traversal cannot be located through property indexes or VIDs. These queries find all the subgraphs that satisfy the query pattern by scanning the whole graph space which will have poor query performance. NebulaGraph does not implement indexing for the graph structures of subgraphs or paths.

#### No predefined bonds between Tags and Edge types

Define the bonds between Tags and Edge types in the application, not NebulaGraph. There are no statements that could get the bonds between Tags and Edge types.

### Tags/Edge types predefine a set of properties

While creating Tags or Edge types, you need to define a set of properties. Properties are part of the NebulaGraph Schema.

#### Control changes in the business model and the data model

Changes here refer to changes in business models and data models (meta-information), not changes in the data itself.

Some graph databases are designed to be Schema-free, so their data modeling, including the modeling of the graph topology and properties, can be very flexible. Properties can be re-modeled to graph topology, and vice versa. Such systems are often specifically optimized for graph topology access.

NebulaGraph 3.4.1 is a strong-Schema (row storage) system, which means that the business model should not change frequently. For example, the property Schema should not change. It is similar to avoiding ALTER TABLE in MySQL.

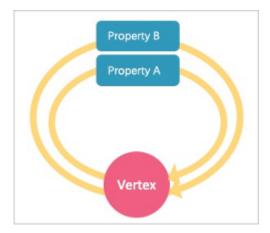
On the contrary, vertices and their edges can be added or deleted at low costs. Thus, the easy-to-change part of the business model should be transformed to vertices or edges, rather than properties.

For example, in a business model, people have relatively fixed properties such as age, gender, and name. But their contact, place of visit, trade account, and login device are often changing. The former is suitable for modeling as properties and the latter as vertices or edges.

- 532/1069 - 2023 Vesoft Inc.

#### Set temporary properties through self-loop edges

As a strong Schema system, NebulaGraph does not support List-type properties. And using ALTER TAG costs too much. If you need to add some temporary properties or List-type properties to a vertex, you can first create an edge type with the required properties, and then insert one or more edges that direct to the vertex itself. The figure is as follows.



To retrieve temporary properties of vertices, fetch from self-loop edges. For example:

Operations on loops are not encapsulated with any syntactic sugars and you can use them just like those on normal edges.

# About dangling edges

A dangling edge is an edge that only connects to a single vertex and only one part of the edge connects to the vertex.

In NebulaGraph 3.4.1, dangling edges may appear in the following two cases.

- 1. Insert edges with INSERT EDGE statement before the source vertex or the destination vertex exists.
- 2. Delete vertices with DELETE VERTEX statement and the WITH EDGE option is not used. At this time, the system does not delete the related outgoing and incoming edges of the vertices. There will be dangling edges by default.

Dangling edges may appear in NebulaGraph 3.4.1 as the design allow it to exist. And there is no MERGE statement like openCypher has. The existence of dangling edges depends entirely on the application level. You can use GO and LOOKUP statements to find a dangling edge, but cannot use the MATCH statement to find a dangling edge.

#### Examples:

```
// Insert an edge that connects two vertices which do not exist in the graph. The source vertex's ID is '11'. The destination vertex's ID is'13'.

nebula> CREATE EDGE IF NOT EXISTS el (name string, age int);
nebula> INSERT EDGE el (name, age) VALUES "11"->"13":("n1", 1);
```

#### Breadth-first traversal over depth-first traversal

- NebulaGraph has lower performance for depth-first traversal based on the Graph topology, and better performance for breadth-first traversal and obtaining properties. For example, if model A contains properties "name", "age", and "eye color", it is recommended to create a tag person and add properties <code>name</code>, <code>age</code>, and <code>eye\_color</code> to it. If you create a tag <code>eye\_color</code> and an edge type <code>has</code>, and then create an edge to represent the eye color owned by the person, the traversal performance will not be high.
- The performance of finding an edge by an edge property is close to that of finding a vertex by a vertex property. For some databases, it is recommended to re-model edge properties as those of the intermediate vertices. For example, model the pattern (src)-[edge {P1, P2}]->(dst) as (src)-[edge1]->(i\_node {P1, P2})-[edge2]->(dst). With NebulaGraph 3.4.1, you can use (src)-[edge {P1, P2}]->(dst) directly to decrease the depth of the traversal and increase the performance.

### **Edge directions**

To query in the opposite direction of an edge, use the following syntax:

```
(dst)<-[edge]-(src) or GO FROM dst REVERSELY.
```

If you do not care about the directions or want to query against both directions, use the following syntax:

```
(src)-[edge]-(dst) or GO FROM src BIDIRECT.
```

Therefore, there is no need to insert the same edge redundantly in the reversed direction.

# Set tag properties appropriately

Put a group of properties that are on the same level into the same tag. Different groups represent different concepts.

### Use indexes correctly

Using property indexes helps find VIDs through properties, but can lead to great performance reduction. Only use an index when you need to find vertices or edges through their properties.

# Design VIDs appropriately

See VID.

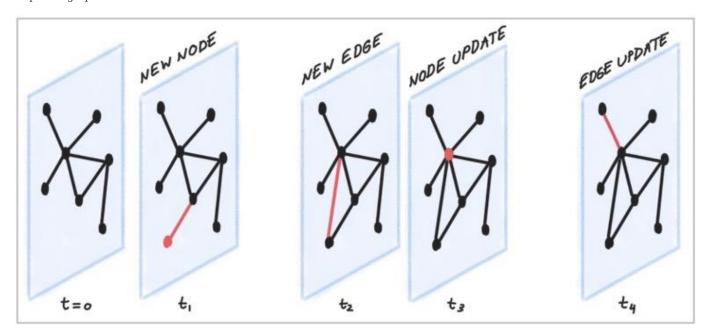
#### Long texts

Do not use long texts to create edge properties. Edge properties are stored twice and long texts lead to greater write amplification. For how edges properties are stored, see Storage architecture. It is recommended to store long texts in HBase or Elasticsearch and store its address in NebulaGraph.

# 11.3.2 Dynamic graphs (sequence graphs) are not supported

In some scenarios, graphs need to have the time information to describe how the structure of the entire graph changes over time.

The Rank field on Edges in NebulaGraph 3.4.1 can be used to store time in int64, but no field on vertices can do this because if you store the time information as property values, it will be covered by new insertion. Thus NebulaGraph does not support sequence graphs.



# 11.3.3 Free graph data modeling tools

### arrows.app

 $1.\ https://blog.twitter.com/engineering/en_us/topics/insights/2021/temporal-graph-networks \ \boldsymbol{\leftarrow}$ 

Last update: February 19, 2024

- 535/1069 - 2023 Vesoft Inc.

# 11.4 System design suggestions

### 11.4.1 QPS or low-latency first

- NebulaGraph 3.4.1 is good at handling small requests with high concurrency. In such scenarios, the whole graph is huge, containing maybe trillions of vertices or edges, but the subgraphs accessed by each request are not large (containing millions of vertices or edges), and the latency of a single request is low. The concurrent number of such requests, i.e., the QPS, can be huge.
- On the other hand, in interactive analysis scenarios, the request concurrency is usually not high, but the subgraphs accessed by each request are large, with thousands of millions of vertices or edges. To lower the latency of big requests in such scenarios, you can split big requests into multiple small requests in the application, and concurrently send them to multiple graphd processes. This can decrease the memory used by each graphd process as well. Besides, you can use NebulaGraph Algorithm for such scenarios.

#### 11.4.2 Data transmission and optimization

- Read/write balance. NebulaGraph fits into OLTP scenarios with balanced read/write, i.e., concurrent write and read. It is not suitable for OLAP scenarios that usually need to write once and read many times.
- Select different write methods. For large batches of data writing, use SST files. For small batches of data writing, use INSERT.
- Run COMPACTION and BALANCE jobs to optimize data format and storage distribution at the right time.
- NebulaGraph 3.4.1 does not support transactions and isolation in the relational database and is closer to NoSQL.

### 11.4.3 Query preheating and data preheating

Preheat on the application side:

- The Grapd process does not support pre-compiling queries and generating corresponding query plans, nor can it cache previous query results.
- The Storagd process does not support preheating data. Only the LSM-Tree and BloomFilter of RocksDB are loaded into memory at startup.
- Once accessed, vertices and edges are cached respectively in two types of LRU cache of the Storage Service.

Last update: February 19, 2024

- 536/1069 - 2023 Vesoft Inc.

# 11.5 Execution plan

NebulaGraph 3.4.1 applies rule-based execution plans. Users cannot change execution plans, pre-compile queries (and corresponding plan cache), or accelerate queries by specifying indexes.

To view the execution plan and executive summary, see EXPLAIN and PROFILE.

Last update: February 19, 2024

# 11.6 Processing super vertices

#### 11.6.1 Principle introduction

In graph theory, a super vertex, also known as a dense vertex, is a vertex with an extremely high number of adjacent edges. The edges can be outgoing or incoming.

Super vertices are very common because of the power-law distribution. For example, popular leaders in social networks (Internet celebrities), top stocks in the stock market, Big Four in the banking system, hubs in transportation networks, websites with high clicking rates on the Internet, and best sellers in E-commerce.

In NebulaGraph 3.4.1, a vertex and its properties form a key-value pair, with its VID and other meta information as the key. Its Out-Edge Key-Value and In-Edge Key-Value are stored in the same partition in the form of LSM-trees in hard disks and caches.

Therefore, directed traversals from this vertex and directed traversals ending at this vertex both involve either a large number of sequential IO scans (ideally, after Compaction or a large number of random IO (frequent writes to the vertex and its ingoing and outgoing edges).

As a rule of thumb, a vertex is considered dense when the number of its edges exceeds 10,000. Some special cases require additional consideration.



In NebulaGraph 3.4.1, there is not any data structure to store the out/in degree for each vertex. Therefore, there is no direct method to know whether it is a super vertex or not. You can try to use Spark to count the degrees periodically.

#### Indexes for duplicate properties

In a property graph, there is another class of cases similar to super vertices: a **property has a very high duplication rate**, i.e., many vertices with the same tag but different VIDs have identical property and property values.

Property indexes in NebulaGraph 3.4.1 are designed to reuse the functionality of RocksDB in the Storage Service, in which case indexes are modeled as keys with the same prefix. If the lookup of a property fails to hit the cache, it is processed as a random seek and a sequential prefix scan on the hard disk to find the corresponding VID. After that, the graph is usually traversed from this vertex, so that another random read and sequential scan for the corresponding key-value of this vertex will be triggered. The higher the duplication rate, the larger the scan range.

For more information about property indexes, see How indexing works in NebulaGraph.

Usually, special design and processing are required when the number of duplicate property values exceeds 10,000.

#### Suggested solutions

SOLUTIONS AT THE DATABASE END

- 1. Truncation: Only return a certain number (a threshold) of edges, and do not return other edges exceeding this threshold.
- 2. Compact: Reorganize the order of data in RocksDB to reduce random reads and increase sequential reads.

#### SOLUTIONS AT THE APPLICATION END

Break up some of the super vertices according to their business significance:

• Delete multiple edges and merge them into one.

For example, in the transfer scenario  $(Account_A)-[TRANSFER]->(Account_B)$ , each transfer record is modeled as an edge between account A and account B, then there may be tens of thousands of transfer records between  $(Account_A)$  and  $(Account_B)$ .

In such scenarios, merge obsolete transfer details on a daily, weekly, or monthly basis. That is, batch-delete old edges and replace them with a small number of edges representing monthly total and times. And keep the transfer details of the latest month.

• Split an edge into multiple edges of different types.

For example, in the (Airport)<-[DEPART]-(Flight) scenario, the departure of each flight is modeled as an edge between a flight and an airport. Departures from a big airport might be enormous.

According to different airlines, divide the DEPART edge type into finer edge types, such as DEPART\_CEAIR, DEPART\_CSAIR, etc. Specify the departing airline in queries (graph traversal).

• Split vertices.

For example, in the loan network (person)-[BORROW]->(bank), large bank A will have a very large number of loans and borrowers. In such scenarios, you can split the large vertex A into connected sub-vertices A1, A2, and A3.

```
(Person1)-[BORROW]->(BankA1), (Person2)-[BORROW]->(BankA2), (Person2)-[BORROW]->(BankA3); (BankA1)-[BELONGS_T0]->(BankA2)-[BELONGS_T0]->(BankA2)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BELONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BankA3)-[BeLONGS_T0]->(BeLONGS_T0]->(BeLONGS_T0]->(BeLONGS_T0]->(BeLONGS_T0]->(BeLONGS_T0]->(BeLONGS_T0]->(BeLONGS_T0]->(BeLONGS_T0]->(BeL
```

A1, A2, and A3 can either be three real branches of bank A, such as Beijing branch, Shanghai branch, and Zhejiang branch, or three virtual branches set up according to certain rules, such as A1: 1-1000, A2: 1001-10000 and A3: 10000+ according to the number of loans. In this way, any operation on A is converted into three separate operations on A1, A2, and A3.

Last update: February 19, 2024

# 11.7 Enable AutoFDO for NebulaGraph

The AutoFDO can analyze the performance of an optimized program and use the program's performance information to guide the compiler to re-optimize the program. This document will help you to enable the AutoFDO for NebulaGraph.

More information about the AutoFDO, please refer AutoFDO Wiki.

# 11.7.1 Resource Preparations

#### **Install Dependencies**

· Install perf

```
sudo apt-get update
sudo apt-get install -y linux-tools-common \
linux-tools-generic \
linux-tools-'uname -r'
```

· Install autofdo tool

```
sudo apt-get update
sudo apt-get install -y autofdo
```

Or you can compile the autofdo tool from source.

#### NebulaGraph Binary with Debug Version

For how to build NebulaGraph from source, please refer to the official document: Install NebulaGraph by compiling the source code. In the configure step, replace CMAKE\_BUILD\_TYPE=Release with CMAKE\_BUILD\_TYPE=RelWithDebInfo as below:

```
$ cmake -DCMAKE_INSTALL_PREFIX=/usr/local/nebula -DENABLE_TESTING=OFF -DCMAKE_BUILD_TYPE=RelWithDebInfo ..
```

# 11.7.2 Prepare Test Data

In our test environment, we use NebulaGraph Bench to prepare the test data and collect the profile data by running the *FindShortestPath*, *Go1Step*, *Go2Step*, *Go3Step*, *InsertPersonScenario* 5 scenarios.



You can use your *TopN* queries in your production environment to collect the profile data, the performance can gain more in your environment.

# 11.7.3 Prepare Profile Data

#### Collect Perf Data For AutoFdo Tool

1. After the test data preparation work done. Collect the perf data for different scenarios. Get the pid of storaged, graphd, metad.

```
$ nebula.service status all
[INFO] nebula-metad: Running as 305422, Listening on 9559
[INFO] nebula-graphd: Running as 305516, Listening on 9669
[INFO] nebula-storaged: Running as 305707, Listening on 9779
```

2. Start the **perf record** for nebula-graphd and nebula-storaged.

```
perf record -p 305516,305707 -b -e br_inst_retired.near_taken:pp -o ~/FindShortestPath.data
```

- 540/1069 - 2023 Vesoft Inc.



Because the nebula-metad service contribution percent is small compared with nebula-graphd and nebula-storaged services. To reduce effort, we didn't collect the perf data for nebula-metad service.

3. Start the benchmark test for FindShortestPath scenario.

```
cd NebulaGraph-Bench
python3 run.py stress run -s benchmark -scenario find_path.FindShortestPath -a localhost:9669 --args='-u 100 -i 100000'
```

- 4. After the benchmark finished, end the **perf record** by Ctrl + c.
- Repeat above steps to collect corresponding profile data for the rest Go1Step, Go2Step, Go3Step and InsertPersonScenario scenarios.

#### Create Gcov File

```
create_gcov --binary=$NEBULA_HOME/bin/nebula-storaged \
--profile=~/FindShortestPath.data \
--gcow=~/FindShortestPath-storaged.gcov \
-gcov_version=1

create_gcov --binary=$NEBULA_HOME/bin/nebula-graphd \
--profile=~/FindShortestPath.data \
--gcov=~/FindShortestPath.graphd.gcov \
-gcov_version=1
```

Repeat for Go1Step, Go2Step, Go3Step and InsertPersonScenario scenarios.

### Merge the Profile Data

```
profile_merger ~/FindShortestPath-graphd.gcov \
~/FindShortestPath-storaged.gcov \
~/golstep-storaged.gcov \
~/golstep-graphd.gcov \
~/go2step-storaged.gcov \
~/go2step-graphd.gcov \
~/go2step-storaged.gcov \
~/go3step-storaged.gcov \
~/go3step-storaged.gcov \
~/go3step-storaged.gcov \
~/go3step-storaged.gcov \
~/go3step-storaged.gcov \
~/go3step-storaged.gcov \
~/lnsertPersonScenario-storaged.gcov \
~/InsertPersonScenario-graphd.gcov
```

You will get a merged profile which is named fbdata.afdo after that.

# 11.7.4 Recompile GraphNebula Binary with the Merged Profile

 $Recompile \ the \ Graph Nebula \ Binary \ by \ passing \ the \ profile \ with \ compile \ option \ \ \ \ -fauto-profile \ .$ 

```
diff --git a/cmake/nebula/GeneralCompilerConfig.cmake b/cmake/nebula/GeneralCompilerConfig.cmake
@@ -20,6 +20,8 @@ add_compile_options(-Winon-virtual-dtor)
add_compile_options(-Woverloaded-virtual)
add_compile_options(-Wignored-qualifiers)
+add_compile_options(-fauto-profile=~/fbdata.afdo)
```



When you use multiple fbdata.afdo to compile multiple times, please remember to make clean before re-compile, baucase only change the fbdata.afdo will not trigger re-compile.

- 541/1069 - 2023 Vesoft Inc.

# 11.7.5 Performance Test Result

# Hardware & Software Environment

Key	Value
CPU Processor#	2
Sockets	2
NUMA	2
CPU Type	Intel(R) Xeon(R) Platinum 8380 CPU @ 2.30GHz
Cores per Processor	40C80T
Cache	L1 data: 48KB L1 i: 32KB L2: 1.25MB per physical core L3: shared 60MB per processor
Memory	Micron DDR4 3200MT/s 16GB16Micron DDR4 3200MT/s 16GB16
SSD Disk	INTEL SSDPE2KE016T8
SSD R/W Sequential	3200 MB/s (read) / 2100 MB/s(write)
Nebula Version	$master\ with\ commit\ id\ 51d84a4ed7d2a032a337e3b996c927e3bc5d1415$
Kernel	4.18.0-408.el8.x86_64

- 542/1069 - 2023 Vesoft Inc.

# Test Results

Scenario	Average Latency(LiB)	Default Binary	Optimized Binary with AutoFDO	P95 Latency (LiB)	Default Binary	Optimized Binary with AutoFDO
FindShortestPath	1	8072.52	7260.10	1	22102.00	19108.00
	2	8034.32	7218.59	2	22060.85	19006.00
	3	8079.27	7257.24	3	22147.00	19053.00
	4	8087.66	7221.39	4	22143.00	19050.00
	5	8044.77	7239.85	5	22181.00	19055.00
	STDDEVP	20.57	17.34	STDDEVP	41.41	32.36
	Mean	8063.71	7239.43	Mean	22126.77	19054.40
	STDDEVP/ Mean	0.26%	0.24%	STDDEVP/ Mean	0.19%	0.17%
	Opt/Default	100.00%	10.22%	Opt/ Default	100.00%	13.89%
Go1Step	1	422.53	418.37	1	838.00	850.00
	2	432.37	402.44	2	866.00	815.00
	3	437.45	407.98	3	874.00	836.00
	4	429.16	408.38	4	858.00	838.00
	5	446.38	411.32	5	901.00	837.00
	STDDEVP	8.02	5.20	STDDEVP	20.63	11.30
	Mean	433.58	409.70	Mean	867.40	835.20
	STDDEVP/ Mean	1.85%	1.27%	STDDEVP/ Mean	2.38%	1.35%
	Opt/Default	100.00%	5.51%	Opt/ Default	100.00%	3.71%
Go2Step	1	2989.93	2824.29	1	10202.00	9656.95
	2	2957.22	2834.55	2	10129.00	9632.40
	3	2962.74	2818.62	3	10168.40	9624.70
	4	2992.39	2817.27	4	10285.10	9647.50
	5	2934.85	2834.91	5	10025.00	9699.65
	STDDEVP	21.53	7.57	STDDEVP	85.62	26.25
	Mean	2967.43	2825.93	Mean	10161.90	9652.24
	STDDEVP/ Mean	0.73%	0.27%	STDDEVP/ Mean	0.84%	0.27%
	Opt/Default	100.00%	4.77%	Opt/ Default	100.00%	5.02%
Go3Step	1	93551.97	89406.96	1	371359.55	345433.50
	2	92418.43	89977.25	2	368868.00	352375.20
	3	92587.67	90339.25	3	365390.15	356198.55

- 544/1069 - 2023 Vesoft Inc.

Scenario	Average Latency(LiB)	Default Binary	Optimized Binary with AutoFDO	P95 Latency (LiB)	Default Binary	Optimized Binary with AutoFDO
	4	93371.64	92458.95	4	373578.15	365177.75
	5	94046.05	89943.44	5	373392.25	352576.00
	STDDEVP	609.07	1059.54	STDDEVP	3077.38	6437.52
	Mean	93195.15	90425.17	Mean	370517.62	354352.20
	STDDEVP/ Mean	0.65%	1.17%	STDDEVP/ Mean	0.83%	1.82%
	Opt/Default	100.00%	2.97%	Opt/ Default	100.00%	4.36%
InsertPerson	1	2022.86	1937.36	1	2689.00	2633.45
	2	1966.05	1935.41	2	2620.45	2555.00
	3	1985.25	1953.58	3	2546.00	2593.00
	4	2026.73	1887.28	4	2564.00	2394.00
	5	2007.55	1964.41	5	2676.00	2581.00
	STDDEVP	23.02	26.42	STDDEVP	57.45	82.62
	Mean	2001.69	1935.61	Mean	2619.09	2551.29
	STDDEVP/ Mean	1.15%	1.37%	STDDEVP/ Mean	2.19%	3.24%
	Opt/Default	100.00%	3.30%	Opt/ Default	100.00%	2.59%

Last update: February 19, 2024

# 11.8 Best practices

NebulaGraph is used in a variety of industries. This topic presents a few best practices for using NebulaGraph. For more best practices, see Blog.

### 11.8.1 Scenarios

- Use cases
- User review
- Performance

# 11.8.2 Kernel

- $\bullet$  What is a graph database and what are its use cases Definition, examples & trends
- NebulaGraph Source Code Explained: Variable-Length Pattern Matching
- Adding a Test Case for NebulaGraph
- BDD-Based Integration Testing Framework for NebulaGraph: Part I
- BDD-Based Integration Testing Framework for NebulaGraph: Part II
- Understanding Subgraph in NebulaGraph
- Full-Text Indexing in NebulaGraph

# 11.8.3 Ecosystem tool

- Validating Import Performance of NebulaGraph Importer
- Ecosystem Tools: NebulaGraph Dashboard for Monitoring
- Visualizing Graph Data with NebulaGraph Explorer

Last update: February 19, 2024

- 546/1069 - 2023 Vesoft Inc.

# 12. Client

# 12.1 Clients overview

NebulaGraph supports multiple types of clients for users to connect to and manage the NebulaGraph database.

- NebulaGraph Console: the native CLI client
- NebulaGraph CPP: the NebulaGraph client for C++
- NebulaGraph Java: the NebulaGraph client for Java
- NebulaGraph Python: the NebulaGraph client for Python
- NebulaGraph Go: the NebulaGraph client for Golang



For now, only NebulaGraph Java is thread-safe.



The following clients can also be used to connect to and manage NebulaGraph, but there is no uptime guarantee.

- NebulaGraph PHP
- NebulaGraph Node
- NebulaGraph .net
- NebulaGraph JDBC
- $\bullet \ NebulaGraph \ Carina \ \ (Python \ ORM)$
- NORM (Golang ORM)
- Graph-Ocean (Java ORM)
- NebulaGraph Ngbatis (Java ORM in a MyBatis fashion)

Last update: February 19, 2024

- 547/1069 - 2023 Vesoft Inc.

# 12.2 NebulaGraph Console

NebulaGraph Console is a native CLI client for NebulaGraph. It can be used to connect a NebulaGraph cluster and execute queries. It also supports special commands to manage parameters, export query results, import test datasets, etc.

# 12.2.1 Obtain NebulaGraph Console

You can obtain NebulaGraph Console in the following ways:

- Download the binary file from the GitHub releases page.
- Compile the source code to obtain the binary file. For more information, see Install from source code.

# 12.2.2 NebulaGraph Console functions

#### Connect to NebulaGraph

To connect to NebulaGraph with the nebula-console file, use the following syntax:

```
<path_of_console> -addr <ip> -port <port> -u <username> -p <password>
```

path\_of\_console indicates the storage path of the NebulaGraph Console binary file.

Parameter descriptions are as follows:

Parameter	Description
-h/-help	Shows the help menu.
-addr/-address	Sets the IP address of the Graph service. The default address is 127.0.0.1.
-P/-port	Sets the port number of the graphd service. The default port number is 9669.
-u/-user	Sets the username of your NebulaGraph account. Before enabling authentication, you can use any existing username. The default username is $root$ .
-p/-password	Sets the password of your NebulaGraph account. Before enabling authentication, you can use any characters as the password.
-t/-timeout	Sets an integer-type timeout threshold of the connection. The unit is millisecond. The default value is 120.
-e/-eval	Sets a string-type nGQL statement. The nGQL statement is executed once the connection succeeds. The connection stops after the result is returned.
-f/-file	Sets the path of an nGQL file. The nGQL statements in the file are executed once the connection succeeds. The result will be returned and the connection stops then.
-enable_ssl	Enables SSL encryption when connecting to NebulaGraph.
-ssl_root_ca_path	Sets the storage path of the certification authority file.
-ssl_cert_path	Sets the storage path of the certificate file.
ssl_private_key_path	Sets the storage path of the private key file.

For information on more parameters, see the project repository.

For example, to connect to the Graph Service deployed on 192.168.10.8, run the following command:

```
./nebula-console -addr 192.168.10.8 -port 9669 -u root -p thisisapassword
```

- 548/1069 - 2023 Vesoft Inc.

### Manage parameters

You can save parameters for parameterized queries.



- Setting a parameter as a VID in a query is not supported.
- Parameters are not supported in SAMPLE clauses.
- Parameters are deleted when their sessions are released.
- The command to save a parameter is as follows:

```
nebula> :param <param_name> => <param_value>;
```

The example is as follows:

 $\bullet$  The command to view the saved parameters is as follows:

```
nebula> :params;
```

• The command to view the specified parameters is as follows:

```
nebula> :params <param_name>;
```

• The command to delete a specified parameter is as follows:

```
nebula> :param <param_name> =>;
```

# **Export query results**

 $Export\ query\ results,\ which\ can\ be\ saved\ as\ a\ CSV\ file,\ DOT\ file,\ and\ a\ format\ of\ Profile\ or\ Explain.$ 



- The exported file is stored in the working directory, i.e., what the linux command pwd shows.
- This command only works for the next query statement.
- You can copy the contents of the DOT file and paste them in GraphvizOnline to generate a visualized execution plan.
- The command to export a csv file is as follows:

```
nebula> :CSV <file_name.csv>;
```

• The command to export a DOT file is as follows:

```
nebula> :dot <file_name.dot>
```

The example is as follows:

```
nebula> :dot a.dot
nebula> PROFILE FORMAT="dot" GO FROM "player100" OVER follow;
```

• The command to export a PROFILE or EXPLAIN format is as follows:

```
nebula> :profile <file_name>;
```

or

nebula> :explain <file\_name>;



The text file output by the above command is the preferred way to report issues in GitHub and execution plans in forums, and for graph query tuning because it has more information and is more readable than a screenshot or CSV file in Studio.

The example is as follows:

```
nebula> :profile go FROM "player102" OVER serve YIELD dst(edge);
nebula> :profile profile.dot
nebula> :PROFILE FORMAT="dot" GO FROM "player102" OVER serve YIELD dst(edge);
nebula> :explain explain.log
nebula> EXPLAIN GO FROM "player102" OVER serve YIELD dst(edge);
```

### Import a testing dataset

The testing dataset is named basketballplayer. To view details about the schema and data, use the corresponding SHOW command.

The command to import a testing dataset is as follows:

```
nebula> :play basketballplayer
```

# Run a command multiple times

To run a command multiple times, use the following command:

```
nebula> :repeat N
```

The example is as follows:

```
nebula> :repeat 3
nebula> GO FROM "player100" OVER follow YIELD dst(edge);
```

- 550/1069 - 2023 Vesoft Inc.

```
| dst(EDGE) |
| "player101" |
| "player25" |
| dst(EDGE) |
| "player102" |
| "player101" |
| dst(EDGE) |
| dst(EDG
```

### Sleep

This command will make NebulaGraph Console sleep for N seconds. The schema is altered in an async way and takes effect in the next heartbeat cycle. Therefore, this command is usually used when altering schema. The command is as follows:

```
nebula> :sleep N
```

# Disconnect NebulaGraph Console from NebulaGraph

You can use :EXIT or :QUIT to disconnect from NebulaGraph. For convenience, NebulaGraph Console supports using these commands in lower case without the colon (":"), such as quit.

The example is as follows:

```
nebula> :QUIT

Bye root!
```

Last update: February 19, 2024

# 12.3 NebulaGraph CPP

NebulaGraph CPP is a C++ client for connecting to and managing the NebulaGraph database.

### 12.3.1 Limitations

You have installed C++ and GCC 4.8 or later versions.

# 12.3.2 Compatibility with NebulaGraph

NebulaGraph version	NebulaGraph CPP version
3.3.0	3.3.0
3.1.0 ~ 3.2.x	3.0.2
3.0.0	3.0.0
2.6.x	2.5.0
2.5.x	2.5.0
2.0.x	2.0.0

### 12.3.3 Install NebulaGraph CPP

This document describes how to install NebulaGraph CPP with the source code.

### Prerequisites

- You have prepared the correct resources.
- You have installed C++ and GCC version is:  $\{10.1.0 \mid 9.3.0 \mid 9.2.0 \mid 9.1.0 \mid 8.3.0 \mid 7.5.0 \mid 7.1.0\}$ . For details, see the gcc\_preset\_versions parameter.

#### Steps

- 1. Clone the NebulaGraph CPP source code to the host.
- (Recommended) To install a specific version of NebulaGraph CPP, use the Git option --branch to specify the branch. For example, to install v3.4.0, run the following command:

```
$ git clone --branch release-3.4 https://github.com/vesoft-inc/nebula-cpp.git
```

• To install the daily development version, run the following command to download the source code from the master branch:

```
$ git clone https://github.com/vesoft-inc/nebula-cpp.git
```

2. Change the working directory to nebula-cpp.

```
$ cd nebula-cpp
```

3. Create a directory named build and change the working directory to it.

```
$ mkdir build && cd build
```

4. Generate the makefile file with CMake.

- 552/1069 - 2023 Vesoft Inc.

#### Q Note

The default installation path is /usr/local/nebula. To modify it, add the -DCMAKE\_INSTALL\_PREFIX=<installation\_path> option while running the following command.

\$ cmake -DCMAKE\_BUILD\_TYPE=Release ...



If G++ does not support C++ 11, add the option <code>-DDISABLE\_CXX11\_ABI=ON</code> .

5. Compile NebulaGraph CPP.

To speed up the compiling, use the -j option to set a concurrent number N. It should be  $\mbox{CPU}\$  core number,  $\mbox{frac}\$  the  $\mbox{memory}\$  size(GB) $\$ {2})\\).

\$ make -j{N}

6. Install NebulaGraph CPP.

\$ sudo make install

7. Update the dynamic link library.

\$ sudo ldconfig

#### 12.3.4 Use NebulaGraph CPP

Compile the CPP file to an executable file, then you can use it. The following steps take using SessionExample.cpp for example.

- 1. Use the example code to create the SessionExample.cpp file.
- 2. Run the following command to compile the file.

\$ LIBRARY\_PATH=<library\_folder\_path>:\$LIBRARY\_PATH g++ -std=c++11 SessionExample.cpp -I<include\_folder\_path> - lnebula\_graph\_client -o session\_example

- library\_folder\_path: The storage path of the NebulaGraph dynamic libraries. The default path is /usr/local/nebula/lib64.
- include\_folder\_path: The storage of the NebulaGraph header files. The default path is /usr/local/nebula/include.

For example:

\$ LIBRARY\_PATH=/usr/local/nebula/lib64:\$LIBRARY\_PATH g++ -std=c++11 SessionExample.cpp -I/usr/local/nebula/include -lnebula\_graph\_client -o session\_example

## 12.3.5 Core of the example code

Nebula CPP clients provide both Session Pool and Connection Pool methods to connect to NebulaGraph. Using the Connection Pool method requires users to manage session instances by themselves.

· Session Pool

For more details about all the code, see SessionPoolExample.

• Connection Pool

For more details about all the code, see SessionExample.

Last update: February 19, 2024

# 12.4 NebulaGraph Java

NebulaGraph Java is a Java client for connecting to and managing the NebulaGraph database.

### 12.4.1 Prerequisites

You have installed Java 8.0 or later versions.

# 12.4.2 Compatibility with NebulaGraph

NebulaGraph version	NebulaGraph Java version
3.3.0	3.3.0
3.0.0 ~ 3.2.0	3.0.0
2.6.x	2.6.1
2.0.x	2.0.0
2.0.0-rc1	2.0.0-rc1

### 12.4.3 Download NebulaGraph Java

• (Recommended) To install a specific version of NebulaGraph Java, use the Git option --branch to specify the branch. For example, to install v3.4.0, run the following command:

\$ git clone --branch release-3.4 https://github.com/vesoft-inc/nebula-java.git

• To install the daily development version, run the following command to download the source code from the master branch:

\$ git clone https://github.com/vesoft-inc/nebula-java.git

# 12.4.4 Use NebulaGraph Java



We recommend that each thread uses one session. If multiple threads use the same session, the performance will be reduced.

When importing a Maven project with tools such as IDEA, set the following dependency in  $\left.\text{pom.xml}\right.$ 



3.0.0-SNAPSHOT indicates the daily development version that may have unknown issues. We recommend that you replace 3.0.0-SNAPSHOT with a released version number to use a table version.

<dependency>
<groupId>com.vesoft</groupId>
<artifactId>client</artifactId>
<version>3.0.0-SNAPSHOT</version>
</dependency>

If you cannot download the dependency for the daily development version, set the following content in pom.xml. Released versions have no such issue.

- 554/1069 - 2023 Vesoft Inc.

```
<repositories>
<repository>
  <id>snapshots</id>
  <url>https://oss.sonatype.org/content/repositories/snapshots/</url>
  </repository>
  </repositories>
```

If there is no Maven to manage the project, manually download the JAR file to install NebulaGraph Java.

# Core of the example code

The NebulaGraph Java client provides both Connection Pool and Session Pool modes, using Connection Pool requires the user to manage session instances.

• Session Pool

For all the code, see GraphSessionPoolExample.

• Connection Pool

For all the code, see GraphClientExample.

Last update: February 19, 2024

# 12.5 NebulaGraph Python

NebulaGraph Python is a Python client for connecting to and managing the NebulaGraph database.

# 12.5.1 Prerequisites

You have installed Python 3.6 or later versions.

# 12.5.2 Compatibility with NebulaGraph

NebulaGraph version	NebulaGraph Python version
3.3.0	3.3.0
3.1.0 ~ 3.2.x	3.1.0
3.0.0 ~ 3.0.2	3.0.0
2.6.x	2.6.0
2.0.x	2.0.0
2.0.0-rc1	2.0.0rc1

# 12.5.3 Install NebulaGraph Python

## Install NebulaGraph Python with pip

\$ pip install nebula3-python==<version>

## Install NebulaGraph Python from the source code

- 1. Clone the NebulaGraph Python source code to the host.
- (Recommended) To install a specific version of NebulaGraph Python, use the Git option --branch to specify the branch. For example, to install v3.4.0, run the following command:

\$ git clone --branch release-3.4 https://github.com/vesoft-inc/nebula-python.git

• To install the daily development version, run the following command to download the source code from the master branch:

\$ git clone https://github.com/vesoft-inc/nebula-python.git

2. Change the working directory to nebula-python.

\$ cd nebula-python

3. Run the following command to install NebulaGraph Python.

\$ pip install .

- 556/1069 - 2023 Vesoft Inc.

# 12.5.4 Core of the example code

NebulaGraph Python clients provides Connection Pool and Session Pool methods to connect to NebulaGraph. Using the Connection Pool method requires users to manage sessions by themselves.

### • Session Pool

For details about all the code, see SessinPoolExample.py.

For limitations of using the Session Pool method, see Example of using session pool.

#### • Connection Pool

For details about all the code, see Example.

Last update: February 19, 2024

# 12.6 NebulaGraph Go

NebulaGraph Go is a Golang client for connecting to and managing the NebulaGraph database.

### 12.6.1 Prerequisites

You have installed Golang 1.13 or later versions.

# 12.6.2 Compatibility with NebulaGraph

NebulaGraph version	NebulaGraph Go version
3.3.0	3.3.0
3.2.x	3.2.0
3.1.0	3.1.0
3.0.0 ~ 3.0.2	3.0.0
2.6.x	2.6.0
2.0.x	2.0.0-GA

# 12.6.3 Download NebulaGraph Go

• (Recommended) To install a specific version of NebulaGraph Go, use the Git option --branch to specify the branch. For example, to install v3.4.0, run the following command:

\$ git clone --branch release-3.4 https://github.com/vesoft-inc/nebula-go.git

• To install the daily development version, run the following command to download the source code from the master branch:

\$ git clone https://github.com/vesoft-inc/nebula-go.git

# 12.6.4 Install or update

Run the following command to install or update NebulaGraph Go:

\$ go get -u -v github.com/vesoft-inc/nebula-go/v3@v3.4.0

## 12.6.5 Core of the example code

The NebulaGraph GO client provides both Connection Pool and Session Pool, using Connection Pool requires the user to manage the session instances.

• Session Pool

For details about all the code, see <a href="mailto:seesion\_pool\_example.go">seesion\_pool\_example.go</a>.

For limitations of using Session Pool, see Usage example.

• Connection Pool

For all the code, see graph\_client\_basic\_example and graph\_client\_goroutines\_example.

Last update: February 19, 2024

# 13. NebulaGraph Studio

# 13.1 About NebulaGraph Studio

# 13.1.1 What is NebulaGraph Studio

NebulaGraph Studio (Studio in short) is a browser-based visualization tool to manage NebulaGraph. It provides you with a graphical user interface to manipulate graph schemas, import data, and run nGQL statements to retrieve data. With Studio, you can quickly become a graph exploration expert from scratch. You can view the latest source code in the NebulaGraph GitHub repository, see <a href="nebula-studio">nebula-studio</a> for details.



You can also try some functions online in Studio.

#### Released versions

In addition to deploying Studio with RPM-based, DEB-based, or Tar-based package, or with Docker. You can also deploy Studio with Helm in the Kubernetes cluster. For more information, see <u>Deploy Studio</u>.

The functions of the above four deployment methods are the same and may be restricted when using Studio. For more information, see Limitations.

#### **Features**

Studio can easily manage NebulaGraph data, with the following functions:

- On the **Schema** page, you can use the graphical user interface to create the space, Tag, Edge Type, Index, and view the statistics on the graph. It helps you quickly get started with NebulaGraph.
- On the Import page, you can operate batch import of vertex and edge data with clicks, and view a real-time import log.
- On the Console page, you can run nGQL statements and read the results in a human-friendly way.

## Scenarios

You can use Studio in one of these scenarios:

- You have a dataset, and you want to explore and analyze data in a visualized way. You can use Docker Compose to deploy NebulaGraph and then use Studio to explore or analyze data in a visualized way.
- You are a beginner of nGQL (NebulaGraph Query Language) and you prefer to use a GUI rather than a command-line interface (CLI) to learn the language.

#### Authentication

Authentication is not enabled in NebulaGraph by default. Users can log into Studio with the root account and any password.

When NebulaGraph enables authentication, users can only sign into Studio with the specified account. For more information, see Authentication.

- 560/1069 - 2023 Vesoft Inc.

# Version compatibility



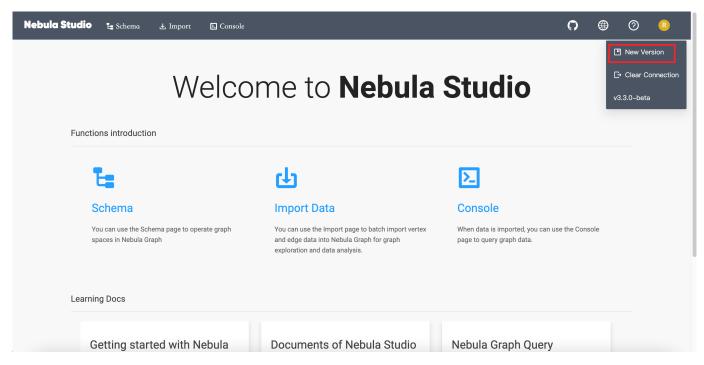
The Studio version is released independently of the NebulaGraph core. The correspondence between the versions of Studio and the NebulaGraph core, as shown in the table below.

NebulaGraph version	Studio version
3.4.0 ~ 3.4.1	3.6.0 \ 3.5.1 \ 3.5.0
3.3.0	3.5.1 \ 3.5.0
$3.0.0\sim3.2.0$	3.4.1 \ 3.4.0
3.1.0	3.3.2
3.0.0	3.2.x
2.6.x	3.1.x
2.6.x	3.1.x
2.0 & 2.0.1	2.x
1.x	1.x

### Check updates

Studio is in development. Users can view the latest releases features through Changelog.

To view the Changelog, on the upper-right corner of the page, click the version and then New version.



Last update: February 19, 2024

# 13.1.2 Limitations

This topic introduces the limitations of Studio.

#### Architecture

For now, Studio v3.x supports x86\_64 architecture only.

### Upload data

Only CSV files without headers can be uploaded, but no limitations are applied to the size and store period for a single file. The maximum data volume depends on the storage capacity of your machine.

# nGQL statements

On the  $\pmb{Console}$  page of Docker-based and RPM-based Studio v3.x, all the nGQL syntaxes except these are supported:

- USE <space\_name> : You cannot run such a statement on the **Console** page to choose a graph space. As an alternative, you can click a graph space name in the drop-down list of **Current Graph Space**.
- You cannot use line breaks (\). As an alternative, you can use the Enter key to split a line.

## Browser

We recommend that you use the latest version of Chrome to get access to Studio.

Last update: February 19, 2024

- 562/1069 - 2023 Vesoft Inc.

# 13.2 Deploy and connect

# 13.2.1 Deploy Studio

This topic describes how to deploy Studio locally by RPM, DEB, tar package and Docker.

#### **RPM-based Studio**

PREREQUISITES

Before you deploy RPM-based Studio, you must confirm that:

- The NebulaGraph services are deployed and started. For more information, see NebulaGraph Database Manual.
- The Linux distribution is CentOS, install Lsof.
- Before the installation starts, the following ports are not occupied.

Port	Description
7001	Web service provided by Studio.

INSTALL

1. Select and download the RPM package according to your needs. It is recommended to select the latest version. Common links are as follows:

Installation package	Checksum	NebulaGraph version
nebula-graph-studio-3.6.0.x86_64.rpm	nebula-graph-studio-3.6.0.x86_64.rpm.sha256	3.4.1

2. Use sudo rpm -i <rpm\_name> to install RPM package.

```
$ sudo rpm -i nebula-graph-studio-3.6.0.x86_64.rpm
```

You can also install it to the specified path using the following command:

```
$ sudo rpm -i nebula-graph-studio-3.6.0.x86_64.rpm --prefix=<path>
```

When the screen returns the following message, it means that the PRM-based Studio has been successfully started.

```
Start installing NebulaGraph Studio now...
NebulaGraph Studio has been installed.
NebulaGraph Studio started automatically.
```

3. When Studio is started, use http://<ip address>:7001 to get access to Studio.

If you can see the Config Server page on the browser, Studio is started successfully.



#### UNINSTALL

You can uninstall Studio using the following command:

\$ sudo rpm -e nebula-graph-studio-3.6.0.x86\_64

If these lines are returned, PRM-based Studio has been uninstalled.

NebulaGraph Studio removed, bye~

# EXCEPTION HANDLING

If the automatic start fails during the installation process or you want to manually start or stop the service, use the following command:

• Start the service manually

\$ bash /usr/local/nebula-graph-studio/scripts/rpm/start.sh

• Stop the service manually

\$ bash /usr/local/nebula-graph-studio/scripts/rpm/stop.sh

If you encounter an error bind EADDRINUSE 0.0.0.0:7001 when starting the service, you can use the following command to check port 7001 usage.

\$ lsof -i:7001

If the port is occupied and the process on that port cannot be terminated, you can use the following command to change Studio service port and restart the service.

//Open the configuration file \$ vi etc/studio-api.yam //Change the port Port: 7001 // Modify this port number and change it to any //Restart service \$ systemctl restart nebula-graph-studio.service

#### **DEB-based Studio**

PREREQUISITES

Before you deploy DEB-based Studio, you must do a check of these:

- The NebulaGraph services are deployed and started. For more information, see NebulaGraph Database Manual.
- The Linux distribution is Ubuntu.
- Before the installation starts, the following ports are not occupied.

Port	Description
7001	Web service provided by Studio

 $\bullet$  The path  $\mbox{/usr/lib/systemd/system}$  exists in the system. If not, create it manually.

INSTALL

1. Select and download the DEB package according to your needs. It is recommended to select the latest version. Common links are as follows:

Installation package	Checksum	NebulaGraph version
nebula-graph-studio-3.6.0.x86_64.deb	$nebula-graph-studio-3.6.0.x86\_64.deb.sha256$	3.4.1

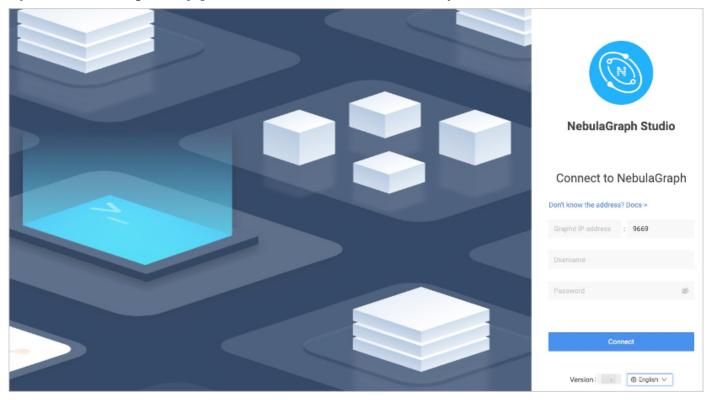
2. Use sudo dpkg -i <deb\_name> to install DEB package.

For example, install Studio 3.6.0, use the following command:

\$ sudo dpkg -i nebula-graph-studio-3.6.0.x86\_64.deb

3. When Studio is started, use http://<ip address>:7001 to get access to Studio.

If you can see the **Config Server** page on the browser, Studio is started successfully.



#### UNINSTALL

You can uninstall Studio using the following command:

\$ sudo dpkg -r nebula-graph-studio

# tar-based Studio

#### PREREQUISITES

Before you deploy tar-based Studio, you must do a check of these:

- The NebulaGraph services are deployed and started. For more information, see NebulaGraph Database Manual.
- Before the installation starts, the following ports are not occupied.

Port	Description	
7001	Web service provided by Studio	

INSTALL AND DEPLOY

1. Select and download the tar package according to your needs. It is recommended to select the latest version. Common links are as follows:

Installation package	Studio version
nebula-graph-studio-3.6.0.x86_64.tar.gz	3.6.0

2. Use tar -xvf to decompress the tar package.

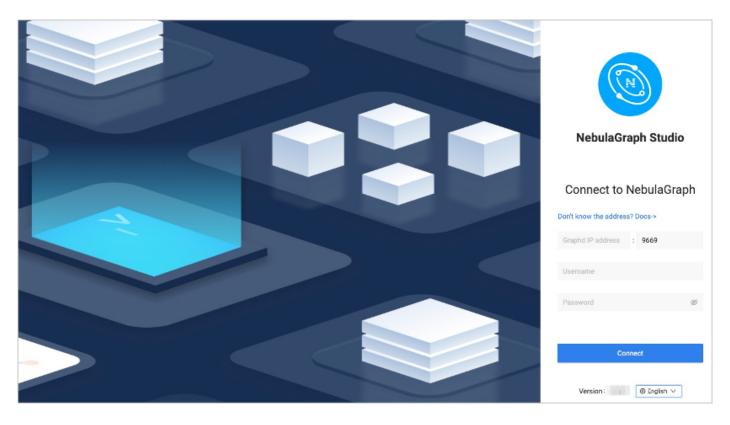
```
$ tar -xvf nebula-graph-studio-3.6.0.x86_64.tar.gz
```

3. Deploy and start nebula-graph-studio.

```
$ cd nebula-graph-studio
$ ./server
```

4. When Studio is started, use http://<ip address>:7001 to get access to Studio.

If you can see the **Config Server** page on the browser, Studio is started successfully.



STOP SERVICE

You can use kill pid to stop the service:

\$ kill \$(lsof -t -i :7001) #stop nebula-graph-studio

# Docker-based Studio

PREREQUISITES

Before you deploy Docker-based Studio, you must do a check of these:

- The NebulaGraph services are deployed and started. For more information, see NebulaGraph Database Manual.
- On the machine where Studio will run, Docker Compose is installed and started. For more information, see <a href="Docker Compose Documentation">Docker Compose Documentation</a>.
- $\bullet$  Before the installation starts, the following ports are not occupied.

Port	Description
7001	Web service provided by Studio

- 567/1069 - 2023 Vesoft Inc.

### PROCEDURE

To deploy and start Docker-based Studio, run the following commands. Here we use NebulaGraph v3.4.1 for demonstration:

1. Download the configuration files for the deployment.

# Installation package NebulaGraph version nebula-graph-studio-3.6.0.tar.gz 3.4.1

2. Create the nebula-graph-studio-3.6.0 directory and decompress the installation package to the directory.

\$ mkdir nebula-graph-studio-3.6.0 -zxvf nebula-graph-studio-3.6.0.gz -C nebula-graph-studio-3.6.0

3. Change to the nebula-graph-studio-3.6.0 directory.

\$ cd nebula-graph-studio-3.6.0

4. Pull the Docker image of Studio.

\$ docker-compose pull

5. Build and start Docker-based Studio. In this command, |-d is to run the containers in the background.

\$ docker-compose up -d

If these lines are returned, Docker-based Studio v3.x is deployed and started.

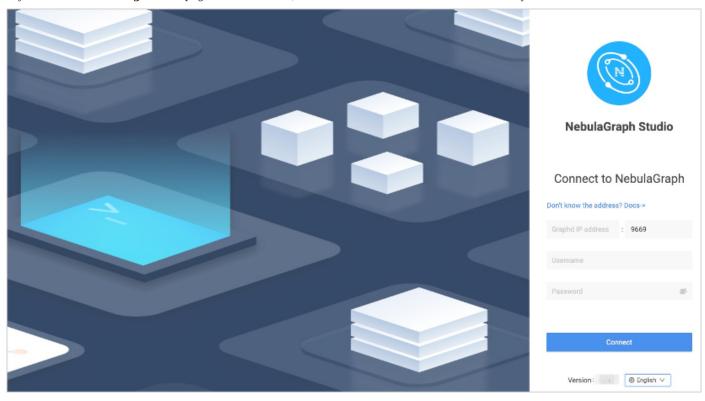
Creating docker\_web\_1 ... done

6. When Docker-based Studio is started, use http://<ip address>:7001 to get access to Studio.

#### Q Note

Run ifconfig or ipconfig to get the IP address of the machine where Docker-based Studio is running. On the machine running Docker-based Studio, you can use <a href="http://localhost:7001">http://localhost:7001</a> to get access to Studio.

If you can see the Config Server page on the browser, Docker-based Studio is started successfully.



### Helm-based Studio

This section describes how to deploy Studio with Helm.

PREREQUISITES

Before installing Studio, you need to install the following software and ensure the correct version of the software:

Software	Requirement
Kubernetes	>= 1.14
Helm	>= 3.2.0

INSTALL

1. Use Git to clone the source code of Studio to the host.

\$ git clone https://github.com/vesoft-inc/nebula-studio.git

2. Make the nebula-studio directory the current working directory.

bash

\$ cd nebula-studio

3. Assume using release name: my-studio, installed Studio in Helm Chart.

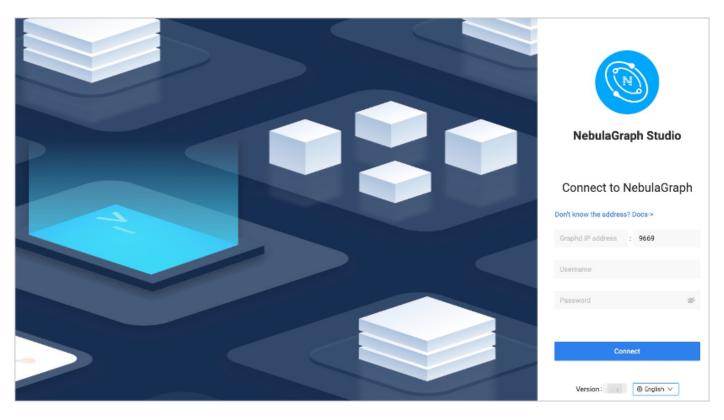
\$ helm upgrade --install my-studio --set service.type=NodePort --set service.port=30070deployment/helm

The configuration parameters of the Helm Chart are described below.

Parameter	Default value	Description
replicaCount	0	The number of replicas for Deployment.
image.nebula Studio.name	vesoft/nebula-graph- studio	The image name of nebula-graph-studio.
image.nebulaStudio.version	v3.6.0	The image version of nebula-graph-studio.
service.type	ClusterIP	The service type, which should be one of ${\tt NodePort}$ , ${\tt ClusterIP}$ , and ${\tt LoadBalancer}$ .
service.port	7001	The expose port for nebula-graph-studio's web.
service.nodePort	32701	The proxy port for accessing nebula-studio outside kubernetes cluster.
resources.nebulaStudio	{}	The resource limits/requests for nebula-studio.
persistent. storage Class Name	пп	The name of storage Class. The default value will be used if not specified. $ \\$
persistent.size	5Gi	The persistent volume size.

 $4. \ When \ Studio \ is \ started, \ use \ \ \verb|http://<node_address>: 30070/ \ to \ get \ access \ to \ Studio.$ 

If you can see the **Config Server** page on the browser, Studio is started successfully.



UNINSTALL

\$ helm uninstall my-studio

# Next to do

On the Config Server page, connect Docker-based Studio to NebulaGraph. For more information, see Connect to NebulaGraph.

Last update: February 19, 2024

- 571/1069 - 2023 Vesoft Inc.

# 13.2.2 Connect to NebulaGraph

After successfully launching Studio, you need to configure to connect to NebulaGraph. This topic describes how Studio connects to the NebulaGraph database.

### Prerequisites

Before connecting to the NebulaGraph database, you need to confirm the following information:

- The NebulaGraph services and Studio are started. For more information, see Deploy Studio.
- ullet You have the local IP address and the port used by the Graph service of NebulaGraph. The default port is ullet 9669 .
- You have a NebulaGraph account and its password.

# Procedure

To connect Studio to NebulaGraph, follow these steps:

 $_{\hbox{1. Type }}$  http://<ip\_address>:7001 in the address bar of your browser.

The following login page shows that Studio is successfully connected to NebulaGraph.



- 2. On the **Config Server** page of Studio, configure these fields:
- Graphd IP address: Enter the IP address of the Graph service of NebulaGraph. For example, 192.168.10.100.

# Note

- When NebulaGraph and Studio are deployed on the same machine, you must enter the IP address of the machine, instead of 127.0.0.1 or localhost.
- When connecting to a NebulaGraph database on a new tab, a new session will overwrite the sessions of the old TAB. If you need to log in to multiple NebulaGraph databases simultaneously, you can use a different browser or non-trace mode.
- $\bullet$   $\boldsymbol{Port}:$  The port of the Graph service. The default port is  $\,\,^{9669}$  .
- Username and Password: Fill in the log in account according to the authentication settings of NebulaGraph.
- If authentication is not enabled, you can use root and any password as the username and its password.
- If authentication is enabled and no account information has been created, you can only log in as GOD role and use root and nebula as the username and its password.
- If authentication is enabled and different users are created and assigned roles, users in different roles log in with their accounts and passwords.
- 3. After the configuration, click the **Connect** button.



One session continues for up to 30 minutes. If you do not operate Studio within 30 minutes, the active session will time out and you must connect to NebulaGraph again.

- 574/1069 - 2023 Vesoft Inc.

A welcome page is displayed on the first login, showing the relevant functions according to the usage process, and the test datasets can be automatically downloaded and imported.

To visit the welcome page, click .



# Next to do

When Studio is successfully connected to NebulaGraph, you can do these operations:

- Create a schema on the **Console** page or on the **Schema** page.
- Batch import data on the Import page.
- Execute nGQL statements on the Console page.
- Design the schema visually on the **Schema drafting** page.



The permissions of an account determine the operations that can be performed. For details, see Roles and privileges.

LOG OUT

If you want to reset NebulaGraph, you can log out and reconfigure the database.

Click the user profile picture in the upper right corner, and choose Log out.

Last update: February 19, 2024

- 575/1069 -2023 Vesoft Inc.

# 13.3 Quick start

# 13.3.1 Design a schema

To manipulate graph data in NebulaGraph with Studio, you must have a graph schema. This article introduces how to design a graph schema for NebulaGraph.

A graph schema for NebulaGraph must have these essential elements:

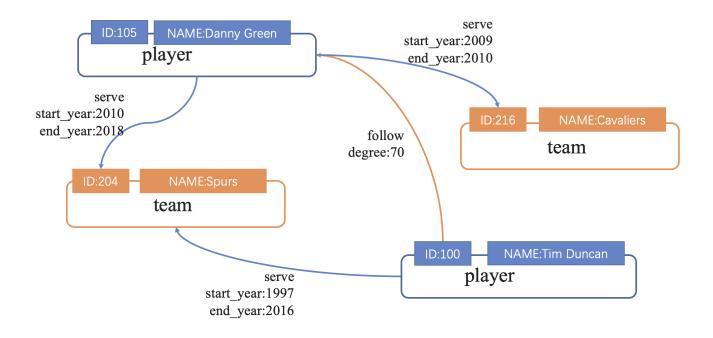
- Tags (namely vertex types) and their properties.
- Edge types and their properties.

In this article, you can install the sample data set basketballplayer and use it to explore a pre-designed schema.

This table gives all the essential elements of the schema.

Element	Name	Property name (Data type)	Description
Tag	player	<ul><li>name (string)</li><li>age (int)</li></ul>	Represents the player.
Tag	team	- name (string)	Represents the team.
Edge type	serve	<pre>- start_year ( int ) - end_year ( int )</pre>	Represent the players behavior.  This behavior connects the player to the team, and the direction is from player to team.
Edge type	follow	- degree (int)	Represent the players behavior.  This behavior connects the player to the player, and the direction is from a player to a player.

This figure shows the relationship (serve/follow) between a player and a team.



## 13.3.2 Create a schema

To batch import data into NebulaGraph, you must have a graph schema. You can create a schema on the **Console** page or on the **Schema** page of Studio.



- Users can use nebula-console to create a schema. For more information, see NebulaGraph Manual and Get started with NebulaGraph.
- Users can use the Schema drafting function to design schema visually. For more information, see Schema drafting.

#### **Prerequisites**

To create a graph schema on Studio, you must do a check of these:

- Studio is connected to NebulaGraph.
- Your account has the privilege of GOD, ADMIN, or DBA.
- The schema is designed.
- A graph space is created.



If no graph space exists and your account has the GOD privilege, you can create a graph space on the **Console** page. For more information, see CREATE SPACE.

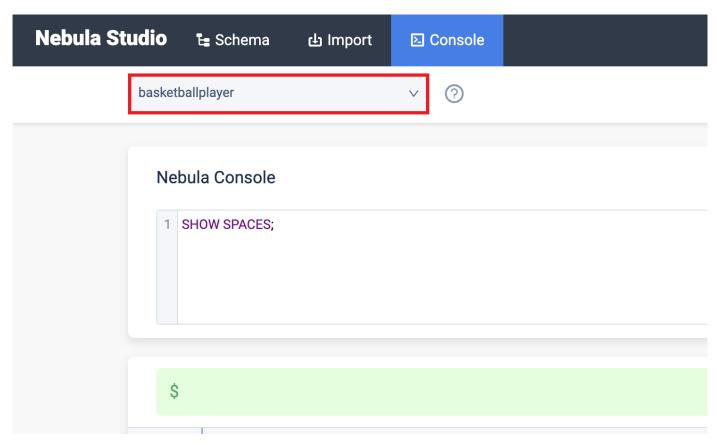
## Create a schema with Schema

- 1. Create tags. For more information, see Operate tags.
- 2. Create edge types. For more information, see Operate edge types.

## Create a schema with Console

- 1. In the toolbar, click the **Console** tab.
- $2. \ In \ the \ \textbf{Current Graph Space} \ field, \ choose \ a \ graph \ space \ name. \ In \ this \ example, \ \textbf{basketballplayer} \ is \ used.$

- 578/1069 - 2023 Vesoft Inc.



3. In the input box, enter these statements one by one and click the button  ${\bf Run}$ .

```
// To create a tag named "player", with two property
nebula> CREATE TAG player(name string, age int);

// To create a tag named "team", with one property
nebula> CREATE TAG team(name string);

// To create an edge type named "follow", with one properties
nebula> CREATE EDGE follow(degree int);

// To create an edge type named "serve", with two properties
nebula> CREATE EDGE serve(start_year int, end_year int);
```

If the preceding statements are executed successfully, the schema is created. You can run the statements as follows to view the schema.

```
// To list all the tags in the current graph space
nebula> SHOW TAGS;

// To list all the edge types in the current graph space
nebula> SHOW EDGES;

// To view the definition of the tags and edge types
DESCRIBE TAG player;
DESCRIBE TAG team;
DESCRIBE EDGE follow;
DESCRIBE EDGE follow;
```

If the schema is created successfully, in the result window, you can see the definition of the tags and edge types.

## Next to do

When a schema is created, you can import data.

## 13.3.3 Import data

After CSV files of data and a schema are created, you can use the **Import** page to batch import vertex and edge data into NebulaGraph for graph exploration and data analysis.

## Prerequisites

To batch import data, do a check of these:

- Studio is connected to NebulaGraph.
- A schema is created.
- CSV files meet the demands of the Schema.
- Your account has privilege of GOD, ADMIN, DBA, or USER.

## Procedure

Before importing data, you need to upload the file first and then create the import task.

## **Upload files**

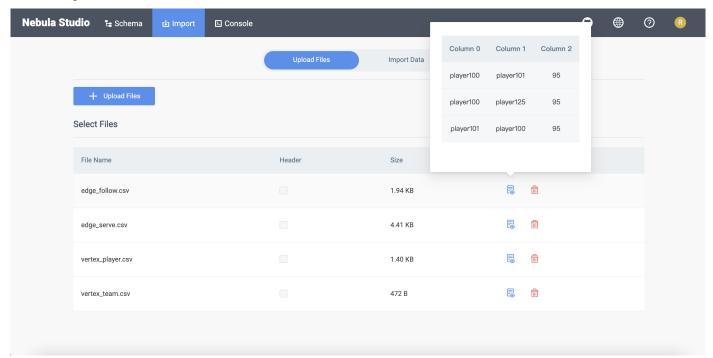
To upload files, follow these steps:

- 1. In the toolbar, click the **Import** tab.
- 2. On the **Upload Files** page, click the **Upload Files** button and then choose CSV files. In this example, edge\_serve.csv, edge\_follow.csv, vertex\_player.csv, and vertex\_team.csv are chosen.



You can choose multiple CSV files at the same time. The CSV file used in this article can be downloaded in the Design a schema.

3. After uploading, you can click the button in the **Operations** column to preview the file content, or click the delete the uploaded file.



## Import Data

To batch import data, follow these steps:

- $_{
  m 1.}$  In the toolbar, click the  ${f Import}$  tab.
- 2. In **Import** tab, click the **Import Data**.
- 3. On the  $\bf Import\ Data$  page, click +  $\bf New\ Import$  button to complete these operations:

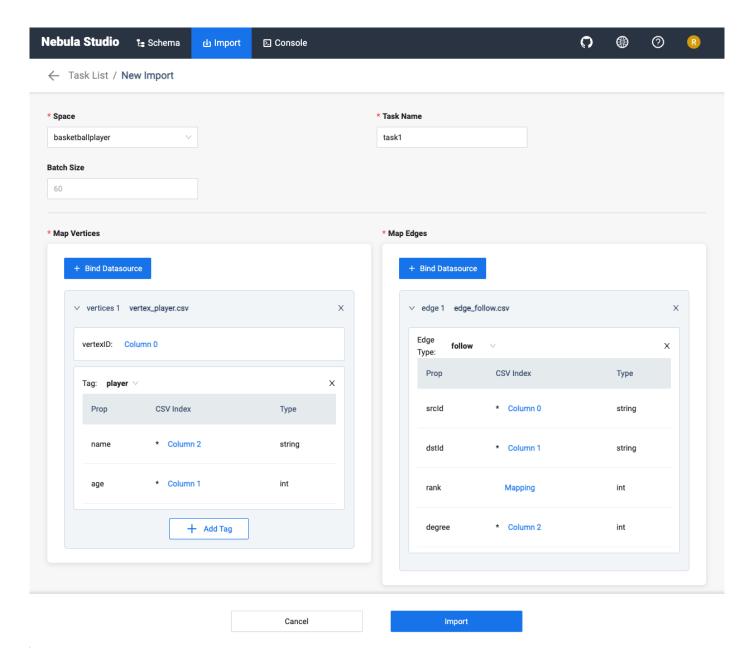
- 583/1069 - 2023 Vesoft Inc.

# Caution

users can click **Import Template** to download the example configuration file example.yaml, and upload the configuration file after configuration. The configuration mode is similar to that of NebulaGraph Importer, but all file paths for configuration files in the template retain the filename only. And make sure all CSV data files are uploaded before importing the YAML file.

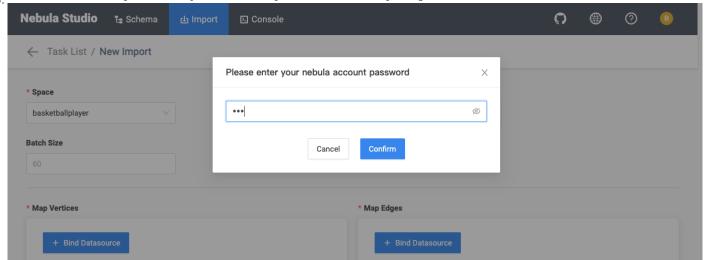
- Select a graph space.
- Fill in the task name.
- (Optional) Fill in the batch size.
- In the **Map Vertices** section, click the **+ Bind Datasource** button, select bind source file in the dialog box, and click the **Confirm** button, the vertex\_player.csv file is chosen.
- In the **vertices 1** drop-down list, click **Select CSV Index**, and select the column where vertexID is located in the pop-up dialog box.
- Click the + Add Tag button and click the icon on the right. In the displayed property list, bind the source data for the tag property. In this example, player is used for the vertex\_player.csv file. For the player tag, choose Column 1 for the age property, and choose Column 2 for the name property.
- In the **Map Edges** section, click the **+ Bind Datasource** button, select bind source file in the dialog box, and click the **Confirm** button, the edge\_follow.csv file is chosen.
- In the vertices 1 drop-down list, click Select Edge Type. In this example, follow is chosen.
- Based on the edge type property, select the corresponding data column from the edge\_follow.csv file. **srcId** and **dstId** are the VIDs of the source vertex and destination vertex of an edge. In this example, **srcId** must be set to the VIDs of the player and **dstId** must be set to the VIDs of another player. **Rank** is optional.

- 585/1069 - 2023 Vesoft Inc.

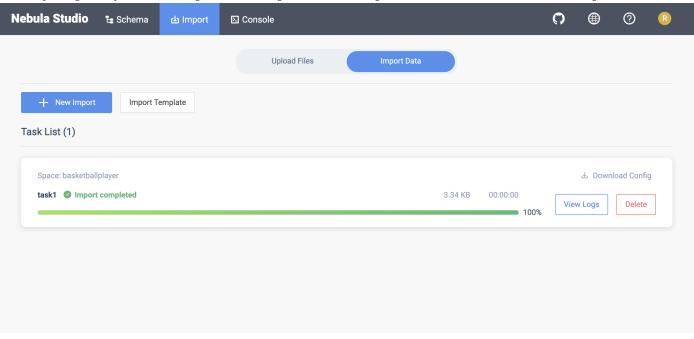


4. After completing the settings, click the **Import** button.

5. You need to enter the password of your NebulaGraph account before importing data.

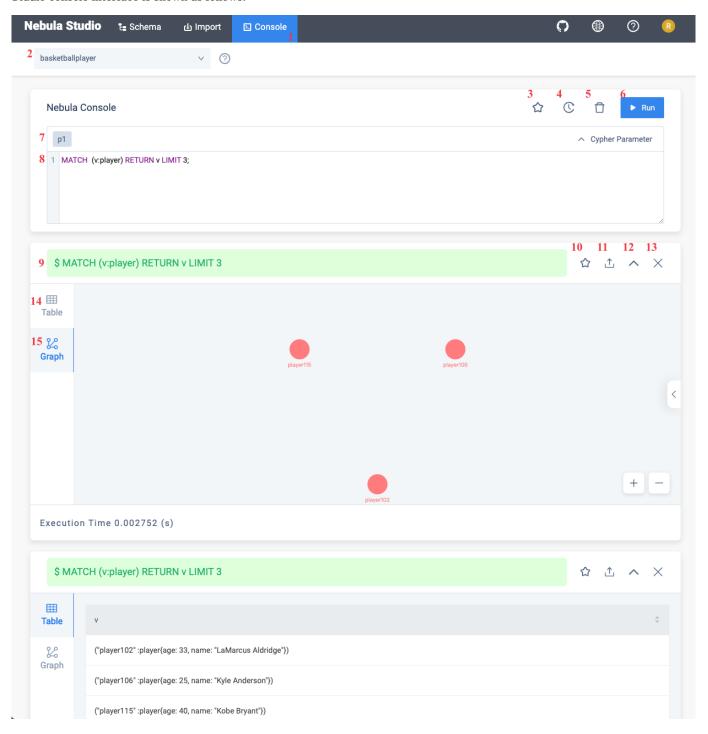


6. After importing data, you can view logs, download logs, download configuration files, and delete tasks on the Import Data tab.



## 13.3.4 Console

Studio console interface is shown as follows.



The following table lists various functions on the console interface.

number	function	descriptions
1	toolbar	Click the <b>Console</b> tab to enter the console page.
2	select a space	Select a space in the Current Graph Space list.  descriptions: Studio does not support running the USE <space_name> statements directly in the input box.</space_name>
3	favorites	Click the button to expand the favorites, click one of the statements, and the input box will automatically enter the statement.
4	history list	Click button representing the statement record. In the statement running record list, click one of the statements, and the statement will be automatically entered in the input box. The list provides the record of the last 15 statements.
5	clean input box	Click $\Box$ button to clear the content entered in the input box.
6	run	After inputting the nGQL statement in the input box, click button to indicate the operation to start running the statement.
7	custom parameters display	Click the button to expand the custom parameters for parameterized query. For details, see Manage parameters.
8	input box	After inputting the nGQL statements, click the button to run the statement. You can input multiple statements and run them at the same time by using the separator;, and also use the symbol // to add comments.
9	statement running status	After running the nGQL statement, the statement running status is displayed. If the statement runs successfully, the statement is displayed in green. If the statement fails, the statement is displayed in red.
10	add to favorites	Click the button to save the statement as a favorite, the button for the favorite statement is colored in yellow exhibit.
11	export CSV file or PNG file	After running the nGQL statement to return the result, when the result is in <b>Table</b> window, click the button to export as a CSV file. Switch to the <b>Graph</b> window and click the button to save the results as a CSV file or PNG image export.
12	expand/hide execution results	Click the button to hide the result or click button to expand the result.
13	close execution results	Click the $\times$ button to close the result returned by this nGQL statement.
14	Table window	Display the result from running nGQL statement. If the statement returns results, the window displays the results in a table.
15	<b>Graph</b> window	Display the result from running nGQL statement. If the statement returns the complete vertex-edge result, the window displays the result as a graph . Click the button on the right to view the overview panel.

## 13.3.5 Use Schema

#### Operate graph spaces

When Studio is connected to NebulaGraph, you can create or delete a graph space. You can use the **Console** page or the **Schema** page to do these operations. This article only introduces how to use the **Schema** page to operate graph spaces in NebulaGraph.

#### **PREREQUISITES**

To operate a graph space on the **Schema** page of Studio, you must do a check of these:

- Studio is connected to NebulaGraph.
- Your account has the authority of GOD. It means that:
- If the authentication is enabled in NebulaGraph, you can use root and any password to sign in to Studio.
- If the authentication is disabled in NebulaGraph, you must use root and its password to sign in to Studio.

CREATE A GRAPH SPACE

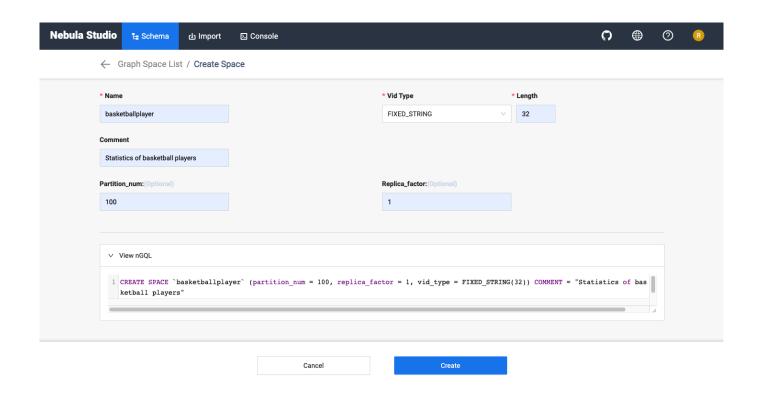
- 1. In the toolbar, click the **Schema** tab.
- 2. In the **Graph Space List** page, click **Create Space**, do these settings:
- Name: Specify a name to the new graph space. In this example, basketballplayer is used. The name must be distinct in the database.
- Vid Type: The data types of VIDs are restricted to FIXED\_STRING(<N>) or INT64. A graph space can only select one VID type. In this example, FIXED\_STRING(32) is used. For more information, see VID.
- **Comment**: Enter the description for graph space. The maximum length is 256 bytes. By default, there will be no comments on a space. But in this example, Statistics of basketball players is used.
- **Optional Parameters**: Set the values of partition\_num and replica\_factor respectively. In this example, these parameters are set to 100 and 1 respectively. For more information, see <a href="CREATE SPACE syntax">CREATE SPACE syntax</a>.

In the Equivalent to the following nGQL statement panel, you can see the statement equivalent to the preceding settings.

```
CREATE SPACE basketballplayer (partition_num = 100, replica_factor = 1, vid_type = FIXED_STRING(32)) COMMENT = "Statistics of basketball players"
```

3. Confirm the settings and then click the + Create button. If the graph space is created successfully, you can see it on the graph space list.

- 590/1069 - 2023 Vesoft Inc.

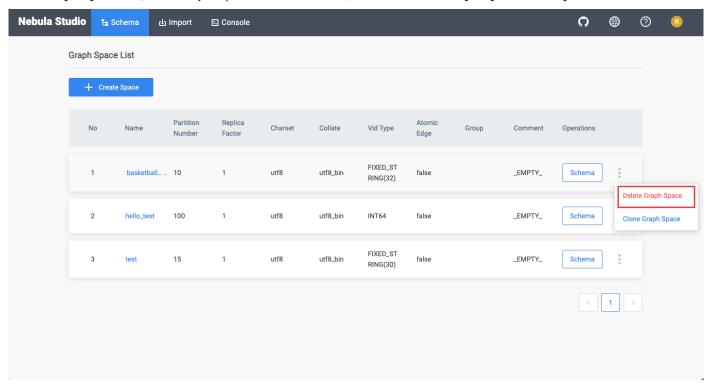


DELETE A GRAPH SPACE



Deleting the space will delete all the data in it, and the deleted data cannot be restored if it is not backed up.

- 1. In the toolbar, click the **Schema** tab.
- 2. In the Graph Space List, find the space you want to be deleted, and click Delete Graph Space in the Operation column.



3. On the dialog box, confirm the information and then click  $\mathbf{OK}$ .

NEXT TO DO

After a graph space is created, you can create or edit a schema, including:

- Operate tags
- Operate edge types
- Operate indexes

#### Operate tags

After a graph space is created in NebulaGraph, you can create tags. With Studio, you can use the **Console** page or the **Schema** page to create, retrieve, update, or delete tags. This topic introduces how to use the **Schema** page to operate tags in a graph space only.

PREREQUISITES

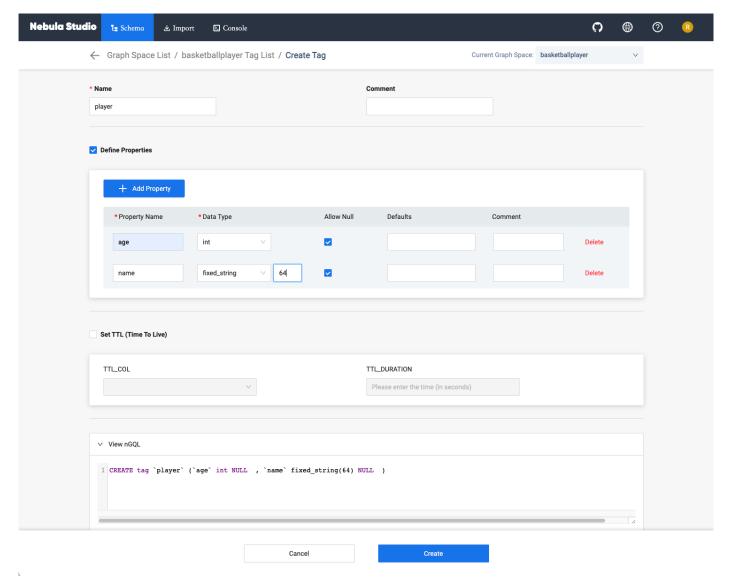
To operate a tag on the **Schema** page of Studio, you must do a check of these:

- Studio is connected to NebulaGraph.
- A graph space is created.
- Your account has the authority of GOD, ADMIN, or DBA.

CDEATE A TAG

- 1. In the toolbar, click the **Schema** tab.
- 2. In the Graph Space List page, find a graph space and then click its name or click Schema in the Operations column.
- 3. In the **Current Graph Space** field, confirm the name of the graph space. If necessary, you can choose another name to change the graph space.
- 4. Click the **Tag** tab and click the **+ Create** button.
- 5. On the **Create** page, do these settings:
- Name: Specify an appropriate name for the tag. In this example, course is specified.
- Comment (Optional): Enter the description for tag.
- Define Properties (Optional): If necessary, click + Add Property to do these settings:
- Enter a property name.
- · Select a data type.
- Select whether to allow null values..
- (Optional) Enter the default value.
- (Optional) Enter the description.
- Set TTL (Time To Live) (Optional): If no index is set for the tag, you can set the TTL configuration: In the upper left corner of the Set TTL panel, click the check box to expand the panel, and configure TTL\_COL and TTL\_ DURATION (in seconds). For more information about both parameters, see TTL configuration.
- 6. When the preceding settings are completed, in the **Equivalent to the following nGQL statement** panel, you can see the nGQL statement equivalent to these settings.

- 593/1069 - 2023 Vesoft Inc.



7. Confirm the settings and then click the **+ Create** button.

When the tag is created successfully, the **Define Properties** panel shows all its properties on the list.

EDIT A TAG

- 1. In the toolbar, click the **Schema** tab.
- 2. In the **Graph Space List** page, find a graph space and then click its name or click **Schema** in the **Operations** column.
- 3. In the **Current Graph Space** field, confirm the name of the graph space. If necessary, you can choose another name to change the graph space.

4.



Click the  ${f Tag}$  tab, find a tag and then click the button

in the  $\boldsymbol{Operations}$  column.

- 5. On the **Edit** page, do these operations:
- To edit a Comment: Click **Edit** on the right of Comment.
- To edit a property: On the **Define Properties** panel, find a property, click **Edit**, and then change the data type or the default value.
- To delete a property: On the **Define Properties** panel, find a property, click **Delete**.
- To add more properties: On the **Define Properties** panel, click the **Add Property** button to add a new property.
- To set the TTL configuration: In the upper left corner of the **Set TTL** panel, click the check box and then set TTL.
- To delete the TTL configuration: When the **Set TTL** panel is expanded, in the upper left corner of the panel, click the check box to delete the configuration.
- To edit the TTL configuration: On the **Set TTL** panel, click **Edit** and then change the configuration of TTL\_COL and TTL\_DURATION (in seconds).



The problem of coexistence of TTL and index, see [TTL]((../../3.ngql-quide/8.clauses-and-options/ttl-options.md).

DELETE A TAG



Confirm the impact before deleting the tag. The deleted data cannot be restored if it is not backup.

- 1. In the toolbar, click the **Schema** tab.
- 2. In the Graph Space List page, find a graph space and then click its name or click Schema in the Operations column.
- 3. In the **Current Graph Space** field, confirm the name of the graph space. If necessary, you can choose another name to change the graph space.
- 4. Click the **Tag** tab, find an tag and then click the button in the **Operations** column.
- 5. Click OK to confirm delete a tag in the pop-up dialog box.

NEXT TO DO

After the tag is created, you can use the **Console** page to insert vertex data one by one manually or use the **Import** page to bulk import vertex data.

Last update: February 19, 2024

- 595/1069 - 2023 Vesoft Inc.

#### Operate edge types

After a graph space is created in NebulaGraph, you can create edge types. With Studio, you can choose to use the **Console** page or the **Schema** page to create, retrieve, update, or delete edge types. This topic introduces how to use the **Schema** page to operate edge types in a graph space only.

#### PREREQUISITES

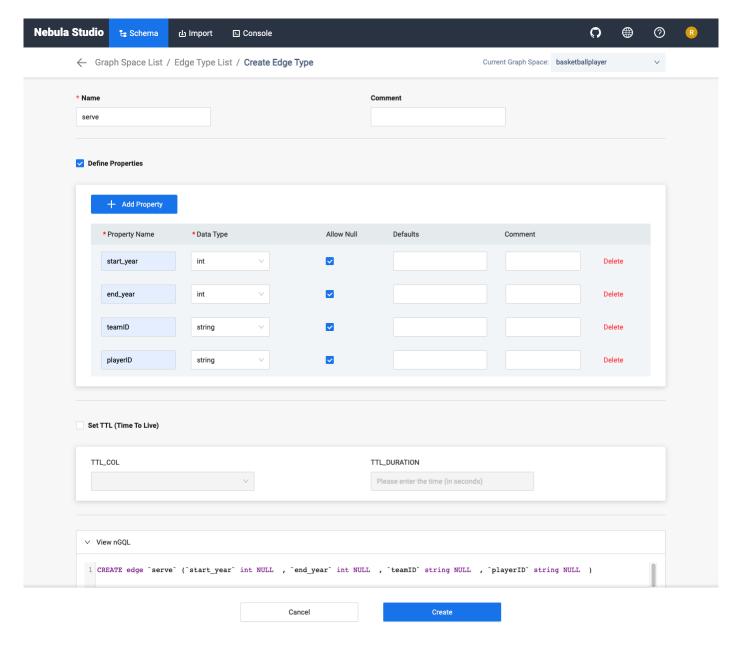
To operate an edge type on the **Schema** page of Studio, you must do a check of these:

- Studio is connected to NebulaGraph.
- A graph space is created.
- Your account has the authority of GOD, ADMIN, or DBA.

CREATE AN EDGE TYPE

- 1. In the toolbar, click the **Schema** tab.
- 2. In the Graph Space List page, find a graph space and then click its name or click Schema in the Operations column.
- 3. In the **Current Graph Space** field, confirm the name of the graph space. If necessary, you can choose another name to change the graph space.
- 4. Click the **Edge Type** tab and click the **+ Create** button.
- 5. On the Create Edge Type page, do these settings:
- Name: Specify an appropriate name for the edge type. In this example, serve is used.
- Comment (Optional): Enter the description for edge type.
- Define Properties (Optional): If necessary, click + Add Property to do these settings:
- Enter a property name.
- · Select a data type.
- Select whether to allow null values..
- (Optional) Enter the default value.
- (Optional) Enter the description.
- Set TTL (Time To Live) (Optional): If no index is set for the edge type, you can set the TTL configuration: In the upper left corner of the Set TTL panel, click the check box to expand the panel, and configure TTL\_COL and TTL\_ DURATION (in seconds). For more information about both parameters, see TTL configuration.
- 6. When the preceding settings are completed, in the **Equivalent to the following nGQL statement** panel, you can see the nGQL statement equivalent to these settings.

- 596/1069 - 2023 Vesoft Inc.



7. Confirm the settings and then click the **+ Create** button.

When the edge type is created successfully, the  $\bf Define\ Properties$  panel shows all its properties on the list.

EDIT AN EDGE TYPE

- 1. In the toolbar, click the **Schema** tab.
- 2. In the **Graph Space List** page, find a graph space and then click its name or click **Schema** in the **Operations** column.
- 3. In the **Current Graph Space** field, confirm the name of the graph space. If necessary, you can choose another name to change the graph space.

4.



Click the  $\mathbf{Edge}$   $\mathbf{Type}$  tab, find an edge type and then click the button

in the **Operations** column.

- 5. On the **Edit** page, do these operations:
- To edit a comment: Click **Edit** on the right of Comment.
- To edit a property: On the **Define Properties** panel, find a property, click **Edit**, and then change the data type or the default value.
- To delete a property: On the **Define Properties** panel, find a property, click **Delete**.
- To add more properties: On the **Define Properties** panel, click the **Add Property** button to add a new property.
- To set the TTL configuration: In the upper left corner of the **Set TTL** panel, click the check box and then set TTL.
- To delete the TTL configuration: When the **Set TTL** panel is expanded, in the upper left corner of the panel, click the check box to delete the configuration.
- To edit the TTL configuration: On the **Set TTL** panel, click **Edit** and then change the configuration of TTL\_COL and TTL\_DURATION (in seconds).



For information about the coexistence problem of TTL and index, see [TTL]((./../3.nggl-guide/8.clauses-and-options/ttl-options.md).

DELETE AN EDGE TYPE



Confirm the impact before deleting the Edge type. The deleted data cannot be restored if it is not backup.

- 1. In the toolbar, click the **Schema** tab.
- 2. In the Graph Space List page, find a graph space and then click its name or click Schema in the Operations column.
- 3. In the **Current Graph Space** field, confirm the name of the graph space. If necessary, you can choose another name to change the graph space.
- 4. Click the **Edge Type** tab, find an edge type and then click the button  $\Box$  in the **Operations** column.
- 5. Click  $\mathbf{OK}$  to confirm in the pop-up dialog box.

NEXT TO DO

After the edge type is created, you can use the **Console** page to insert edge data one by one manually or use the **Import** page to bulk import edge data.

Last update: February 19, 2024

- 598/1069 - 2023 Vesoft Inc.

#### **Operate Indexes**

You can create an index for a Tag and/or an Edge type. An index lets traversal start from vertices or edges with the same property and it can make a query more efficient. With Studio, you can use the **Console** page or the **Schema** page to create, retrieve, and delete indexes. This topic introduces how to use the **Schema** page to operate an index only.



You can create an index when a Tag or an Edge Type is created. But an index can decrease the write speed during data import. We recommend that you import data firstly and then create and rebuild an index. For more information, see <u>Index overview</u>.

#### **PREREQUISITES**

To operate an index on the **Schema** page of Studio, you must do a check of these:

- Studio is connected to NebulaGraph.
- A graph Space, Tags, and Edge Types are created.
- Your account has the authority of GOD, ADMIN, or DBA.

CREATE AN INDEX

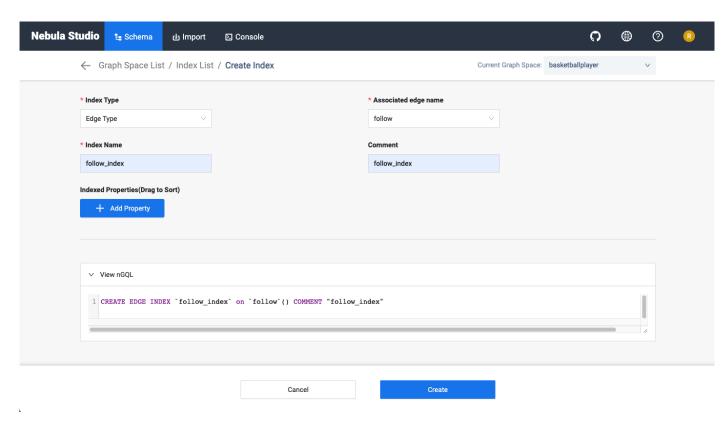
- 1. In the toolbar, click the **Schema** tab.
- 2. In the Graph Space List page, find a graph space and then click its name or click Schema in the Operations column.
- 3. In the **Current Graph Space** field, confirm the name of the graph space. If necessary, you can choose another name to change the graph space.
- 4. Click the **Index** tab and then click the **+ Create** button.
- 5. On the **Create** page, do these settings:
- Index Type: Choose to create an index for a tag or for an edge type. In this example, Edge Type is chosen.
- Associated tag name: Choose a tag name or an edge type name. In this example, follow is chosen.
- Index Name: Specify a name for the new index. In this example, follow\_index is used.
- Comment (Optional): Enter the description for index.
- Indexed Properties (Optional): Click Add property, and then, in the dialog box, choose a property. If necessary, repeat this step to choose more properties. You can drag the properties to sort them. In this example, degree is chosen.



The order of the indexed properties has an effect on the result of the LOOKUP statement. For more information, see nGQL Manual.

6. When the settings are done, the **Equivalent to the following nGQL statement** panel shows the statement equivalent to the settings.

- 599/1069 - 2023 Vesoft Inc.



7. Confirm the settings and then click the + Create button. When an index is created, the index list shows the new index.

VIEW INDEXES

- 1. In the toolbar, click the **Schema** tab.
- 2. In the **Graph Space List** page, find a graph space and then click its name or click **Schema** in the **Operations** column.
- 3. In the **Current Graph Space** field, confirm the name of the graph space. If necessary, you can choose another name to change the graph space.
- 4. Click the Index tab, in the upper left corner, choose an index type, Tag or  $Edge\ Type$ .
- 5. In the list, find an index and click its row. All its details are shown in the expanded row.

REBUILD INDEXES

- 1. In the toolbar, click the **Schema** tab.
- 2. In the **Graph Space List** page, find a graph space and then click its name or click **Schema** in the **Operations** column.
- 3. In the **Current Graph Space** field, confirm the name of the graph space. If necessary, you can choose another name to change the graph space.
- 4. Click the Index tab, in the upper left corner, choose an index type, Tag or  $Edge\ Type$ .
- 5. Click the Index tab, find an index and then click the button Rebuild in the Operations column.



For more Information, see REBUILD INDEX.

#### DELETE AN INDEX

To delete an index on **Schema**, follow these steps:

- 1. In the toolbar, click the **Schema** tab.
- 2. In the **Graph Space List** page, find a graph space and then click its name or click **Schema** in the **Operations** column.
- 3. In the **Current Graph Space** field, confirm the name of the graph space. If necessary, you can choose another name to change the graph space.
- 4. Click the **Index** tab, find an index and then click the button in the **Operations** column.
- 5. Click  $\mathbf{OK}$  to confirm in the pop-up dialog box.

Last update: February 19, 2024

- 601/1069 - 2023 Vesoft Inc.

#### View Schema

Users can visually view schemas in NebulaGraph Studio.

STEPS

- 1. In the toolbar, click the **Schema** tab.
- 2. In the Graph Space List page, find a graph space and then click its name or click Schema in the Operations column.
- 3. Click View Schema tab and click the Get Schema button.

OTHER OPERATIONS

In the **Graph Space List** page, find a graph space and then perform the following operations in the **Operations** column:

- View Schema DDL: Displays schema creation statements for the graph space, including graph spaces, tags, edge types, and indexes.
- Clone Graph Space: Clones the schema of the graph space to a new graph space.
- Delete Graph pace: Deletes the graph space, including the schema and all vertices and edges.

## 13.3.6 Schema drafting

Studio supports the schema drafting function. Users can design their schemas on the canvas to visually display the relationships between vertices and edges, and apply the schema to a specified graph space after the design is completed.

#### **Features**

- · Design schema visually.
- Applies schema to a specified graph space.
- Export the schema as a PNG image.

## Entry

At the top navigation bar, click  ${}^{f y}$  .

## Design schema

The following steps take designing the schema of the basketballplayer dataset as an example to demonstrate how to use the schema drafting function.

- 1. At the upper left corner of the page, click New.
- 2. Create a tag by selecting the appropriate color tag under the canvas. You can hold down the left button and drag the tag into the canvas.
- 3. Click the tag. On the right side of the page, you need to fill in the name of the tag as player, and add two properties name and age.
- 4. Create a tag again. The name of the tag is team, and the property is name.
- 5. Connect from the anchor point of the tag player to the anchor point of the tag team. Click the generated edge, fill in the name of the edge type as serve, and add two properties start\_year and end\_year.
- 6. Connect from an anchor point of the tag player to another one of its own. Click the generated edge, fill in the name of the edge type as follow, and add a property degree.
- 7. After the design is complete, click at the top of the page to change the name of the draft, and then click at the top right corner to save the draft.



## Apply schema

- 1. Select the draft that you want to import from the Draft list on the left side of the page, and then click Apply to Space at the upper right corner.
- 2. Import the schema to a new or existing space, and click Confirm.



- For more information about the parameters for creating a graph space, see CREATE SPACE.
- If the same schema exists in the graph space, the import operation fails, and the system prompts you to modify the name or change the graph space.

## Modify schema

Select the schema draft that you want to modify from the **Draft list** on the left side of the page. Click at the upper right corner after the modification.



The graph space to which the schema has been applied will not be modified synchronously.

#### Delete schema

Select the schema draft that you want to delete from the Draft list on the left side of the page, click X at the upper right corner of the thumbnail, and confirm to delete it.

## **Export Schema**

Click  $\hat{\mathbf{L}}$  at the upper right corner to export the schema as a PNG image.

Last update: February 19, 2024

- 604/1069 -2023 Vesoft Inc.

## 13.4 Troubleshooting

## 13.4.1 Connecting to the database error

#### Problem description

According to the connect Studio operation, it prompts failed.

#### Possible causes and solutions

You can troubleshoot the problem by following the steps below.

STEP1: CONFIRM THAT THE FORMAT OF THE HOST FIELD IS CORRECT

You must fill in the IP address (graph\_server\_ip) and port of the NebulaGraph database Graph service. If no changes are made, the port defaults to 9669. Even if NebulaGraph and Studio are deployed on the current machine, you must use the local IP address instead of 127.0.0.1, localhost or 0.0.0.0.

STEP2: CONFIRM THAT THE USERNAME AND PASSWORD ARE CORRECT

If authentication is not enabled, you can use root and any password as the username and its password.

If authentication is enabled and different users are created and assigned roles, users in different roles log in with their accounts and passwords.

STEP3: CONFIRM THAT NEBULAGRAPH SERVICE IS NORMAL

Check NebulaGraph service status. Regarding the operation of viewing services:

- If you compile and deploy NebulaGraph on a Linux server, refer to the NebulaGraph service.
- If you use NebulaGraph deployed by Docker Compose and RPM, refer to the NebulaGraph service status and ports.

If the NebulaGraph service is normal, proceed to Step 4 to continue troubleshooting. Otherwise, please restart NebulaGraph service.



If you used docker-compose up -d to satrt NebulaGraph before, you must run the docker-compose down to stop NebulaGraph.

STEP4: CONFIRM THE NETWORK CONNECTION OF THE GRAPH SERVICE IS NORMAL

Run a command (for example, telnet 9669) on the Studio machine to confirm whether NebulaGraph's Graph service network connection is normal.

If the connection fails, check according to the following steps:

- If Studio and NebulaGraph are on the same machine, check if the port is exposed.
- If Studio and NebulaGraph are not on the same machine, check the network configuration of the NebulaGraph server, such as firewall, gateway, and port.

If you cannot connect to the NebulaGraph service after troubleshooting with the above steps, please go to the NebulaGraph forum for consultation.

Last update: February 19, 2024

- 605/1069 - 2023 Vesoft Inc.

#### 13.4.2 Cannot access to Studio

#### Problem description

I follow the document description and visit 127.0.0.1:7001 or 0.0.0.0:7001 after starting Studio, why can't I open the page?

#### Possible causes and solutions

You can troubleshoot the problem by following the steps below.

STEP1: CONFIRM SYSTEM ARCHITECTURE

It is necessary to confirm whether the machine where the Studio service is deployed is of  $x86\_64$  architecture. Currently, Studio only supports  $x86\_64$  architecture.

STEP2: CHECK IF THE STUDIO SERVICE STARTS NORMALLY

- For Studio deployed with RPM or DEB packages, use systematl status nebula-graph-studio to see the running status.
- For Studio deployed with tar package, use sudo Lsof -i:7001 to check port status.
- For Studio deployed with docker, use docker-compose ps to see the running status. Run docker-compose ps to check if the service has started normally.

If the service is normal, the return result is as follows. Among them, the State column should all be displayed as Up.

```
Name
                                 Command
                                                       State
                                                                           Ports
nebula-web-docker_client_1
                               ./nebula-go-api
                                                                       0.0.0.0:32782->8080/tcp
nebula-web-docker_importer_1 nebula-importer --port=569 ...
                                                               Up
                                                                       0.0.0.0:32783->5699/tcp
nebula-web-docker_nginx_1
                               /docker-entrypoint.sh ngin ...
                                                                       0.0.0.0:7001->7001/tcp, 80/tcp
                                                               Up
nebula-web-docker web 1
                              docker-entrypoint.sh npm r
                                                                       0.0.0.0:32784->7001/tcp
```

If the above result is not returned, stop Studio and restart it first. For details, refer to Deploy Studio.

#### !!! note

```
If you used `docker-compose up -d` to satrt NebulaGraph before, you must run the `docker-compose down` to stop NebulaGraph.
```

STEP3: CONFIRM ADDRESS

If Studio and the browser are on the same machine, users can use <code>localhost:7001</code>, <code>127.0.0.1:7001</code> or <code>0.0.0.0:7001</code> in the browser to access Studio.

If Studio and the browser are not on the same machine, you must enter <studio\_server\_ip>:7001 in the browser. Among them, studio\_server\_ip refers to the IP address of the machine where the Studio service is deployed.

STEP4: CONFIRM NETWORK CONNECTION

Run curl <studio\_server\_ip>:7001 -I to confirm if it is normal. If it returns HTTP/1.1 200 OK, it means that the network is connected normally.

If the connection is refused, check according to the following steps:

If the connection fails, check according to the following steps:

- If Studio and NebulaGraph are on the same machine, check if the port is exposed.
- If Studio and NebulaGraph are not on the same machine, check the network configuration of the NebulaGraph server, such as firewall, gateway, and port.

If you cannot connect to the NebulaGraph service after troubleshooting with the above steps, please go to the NebulaGraph forum for consultation.

## 13.4.3 FAQ

# Why can't I use a function?

If you find that a function cannot be used, it is recommended to troubleshoot the problem according to the following steps:

- 1. Confirm that NebulaGraph is the latest version. If you use Docker Compose to deploy the NebulaGraph database, it is recommended to run docker-compose pull && docker-compose up -d to pull the latest Docker image and start the container.
- 2. Confirm that Studio is the latest version. For more information, refer to  $\underline{\mathsf{check}}\, \underline{\mathsf{updates}}.$
- 3. Search the nebula forum, nebula and nebula-studio projects on the GitHub to confirm if there are already similar problems.
- 4. If none of the above steps solve the problem, you can submit a problem on the forum.

## 14. NebulaGraph Dashboard Community Edition

## 14.1 What is NebulaGraph Dashboard Community Edition

NebulaGraph Dashboard Community Edition (Dashboard for short) is a visualization tool that monitors the status of machines and services in NebulaGraph clusters. This topic introduces Dashboard Community Edition. For details of Dashboard Enterprise Edition, refer to What is NebulaGraph Dashboard Enterprise Edition.



Dashboard Enterprise Edition adds features such as visual cluster creation, batch import of clusters, fast scaling, etc. For more information, see <a href="Pricing">Pricing</a>.

#### 14.1.1 Features

Dashboard monitors:

- The status of all the machines in clusters, including CPU, memory, load, disk, and network.
- The information of all the services in clusters, including the IP addresses, versions, and monitoring metrics (such as the number of queries, the latency of queries, the latency of heartbeats, and so on).
- The information of clusters, including the information of services, partitions, configurations, and long-term tasks.
- Set how often the metrics page refreshes.

## 14.1.2 Scenarios

You can use Dashboard in one of the following scenarios:

- You want to monitor key metrics conveniently and quickly, and present multiple key information of the business to ensure the business operates normally.
- You want to monitor clusters from multiple dimensions (such as the time, aggregate rules, and metrics).
- · After a failure occurs, you need to review it and confirm its occurrence time and unexpected phenomena.

### 14.1.3 Precautions

The monitoring data will be retained for 14 days by default, that is, only the monitoring data within the last 14 days can be queried.



The monitoring service is supported by Prometheus. The update frequency and retention intervals can be modified. For details, see Prometheus.

- 609/1069 - 2023 Vesoft Inc.

## 14.1.4 Version compatibility

The version correspondence between NebulaGraph and Dashboard Community Edition is as follows.

NebulaGraph version	Dashboard version
3.4.0 ~ 3.4.1	3.4.0 \ 3.2.0
3.3.0	3.2.0
2.5.0 ~ 3.2.0	3.1.0
$2.5.x \sim 3.1.0$	1.1.1
2.0.1~2.5.1	1.0.2
2.0.1~2.5.1	1.0.1

## 14.1.5 Release note

## Release

Last update: February 19, 2024

- 610/1069 - 2023 Vesoft Inc.

## 14.2 Deploy Dashboard Community Edition

This topic will describe how to deploy NebulaGraph Dashboard in detail.

To download and compile the latest source code of Dashboard, follow the instructions on the nebula dashboard GitHub page.

## 14.2.1 Prerequisites

Before you deploy Dashboard, you must confirm that:

- The NebulaGraph services are deployed and started. For more information, see NebulaGraph Database Manual.
- Before the installation starts, the following ports are not occupied.
- 9200
- 9100
- 9090
- 8090
- 7003
- The node-exporter is installed on the machines to be monitored. For details on installation, see Prometheus document.

## 14.2.2 Steps

- 1. Download the tar packagenebula-dashboard-3.4.0.x86\_64.tar.gz as needed.
- 2. Run tar -xvf nebula-dashboard-3.4.0.x86\_64.tar.gz to decompress the installation package.
- 3. Modify the config.yaml file in nebula-dashboard.

The configuration file contains the configurations of four dependent services and configurations of clusters. The descriptions of the dependent services are as follows.

Service	Default port	Description
nebula-http- gateway	8090	Provides HTTP ports for cluster services to execute nGQL statements to interact with the NebulaGraph database.
nebula-stats- exporter	9200	Collects the performance metrics in the cluster, including the IP addresses, versions, and monitoring metrics (such as the number of queries, the latency of queries, the latency of heartbeats, and so on).
node-exporter	9100	Collects the source information of nodes in the cluster, including the CPU, memory, load, disk, and network.
prometheus	9090	The time series database that stores monitoring data.

The descriptions of the configuration file are as follows.

```
port: 7003  # Web service port.
gateway:
    ip: hostIP  # The IP of the machine where the Dashboard is deployed.
port: 8090
https: false  # Whether to enable HTTPS.
runnode: dev  # Program running mode, including dev, test, and prod. It is used to distinguish between different running environments generally.
stats-exporter:
    ip: hostIP  # The IP of the machine where the Dashboard is deployed.
nebulaPort: 9200
https: false  # Whether to enable HTTPS.
node-exporter:
    - ip: nebulaHostIP_1  # The IP of the machine where the NebulaGraph is deployed.
port: 9100
https: false  # Whether to enable HTTPS.
# - ip: nebulaHostIP_2
```

```
# port: 9100
# https: false
    ip: hostIP # The IP of the machine where the Dashboard is deployed.
    prometheusPort: 9090
    https: false # Whether to enable HTTPS.

scrape_interval: 5s # The interval for collecting the monitoring data, which is 1 minute by default.

evaluation_interval: 5s # The interval for running alert rules, which is 1 minute by default.
  # Cluster node info
  nebula-cluster:
    name: 'default' # Cluster name
    metad:
       - name: metad0
        endpointIP: nebulaMetadIP # The IP of the machine where the Meta service is deployed.
         port: 9559
         endpointPort: 19559
    # - name: metad1
    # endpointIP: nebulaMetadIP
        endpointPort: 19559
    graphd:
        name: graphd0
        endpointTP: nebulaGraphdIP # The IP of the machine where the Graph service is deployed. port: 9669
         endpointPort: 19669
    # - name: graphd1
# endpointIP: nebulaGraphdIP
        port: 9669
endpointPort: 19669
        name: storaged0
         endpointIP: nebulaStoragedIP # The IP of the machine where the Storage service is deployed.
         port: 9779
    endpointPort: 19779
# - name: storaged1
        endpointIP: nebulaStoragedIP
        port: 9779
         endpointPort: 19779
```

4. Run ./dashboard.service start all to start the services.

#### **Deploy Dashboard with Docker Compose**

If you are deploying Dashboard using docker, you should also modify the configuration file <code>config.yaml</code>, and then run <code>docker-compose</code> up -d to start the container.



If you change the port number in config.yaml, the port number in docker-compose.yaml needs to be consistent as well.

Run docker-compose stop to stop the container.

## 14.2.3 Manage services in Dashboard

You can use the dashboard.service script to start, restart, stop, and check the Dashboard services.

 $\label{lem:substant} $$ \sc '' = \sc '' - \sc$ 

Parameter	Description
dashboard_path	Dashboard installation path.
-v	Display detailed debugging information.
-h	Display help information.
start	Start the target services.
restart	Restart the target services.
stop	Stop the target services.
status	Check the status of the target services.
prometheus	Set the prometheus service as the target service.
webserver	Set the webserver Service as the target service.
exporter	Set the exporter Service as the target service.
gateway	Set the gateway Service as the target service.
all	Set all the Dashboard services as the target services.



To view the Dashboard version, run the command  $\,$  ./dashboard.service  $\,$  -version .

# 14.2.4 Next to do

### Connect to Dashboard

Last update: February 19, 2024

- 613/1069 -2023 Vesoft Inc.

# 14.3 Connect Dashboard

After Dashboard is deployed, you can log in and use Dashboard on the browser.

### 14.3.1 Prerequisites

- The Dashboard services are started. For more information, see Deploy Dashboard.
- We recommend you to use the Chrome browser of the version above 89. Otherwise, there may be compatibility issues.

#### 14.3.2 Procedures

- 1. Confirm the IP address of the machine where the Dashboard service is installed. Enter <IP>:7003 in the browser to open the login page.
- 2. Enter the username and the passwords of the NebulaGraph database.
- If authentication is enabled, you can log in with the created accounts.
- If authentication is not enabled, you can only log in using root as the username and random characters as the password.

  To enable authentication, see Authentication.
- 3. Select the NebulaGraph version to be used.
- 4. Click Login.

Last update: February 19, 2024

- 614/1069 - 2023 Vesoft Inc.

# 14.4 Dashboard

 $Nebula Graph\ Dashboard\ consists\ of\ three\ parts:\ Machine,\ Service,\ and\ Management.\ This\ topic\ will\ describe\ them\ in\ detail.$ 

# 14.4.1 Overview



# 14.4.2 Machine

Click Machine->Overview to enter the machine overview page.

On this page, you can view the variation of CPU, Memory, Load, Disk, and Network In/Out quickly.

- By default, you can view the monitoring data for a maximum of 14 days. You can also select a time range or quickly select the latest 1 hour, 6 hours, 12 hours, 1 day, 3 days, 7 days, or 14 days.
- By default, you can view the monitoring data of all the instances in clusters. You can select the instances you want to view in the **instance** box.
- By default, the monitoring information page will not be updated automatically. You can set the update frequency of the monitoring information page globally or click the button to update the page manually.

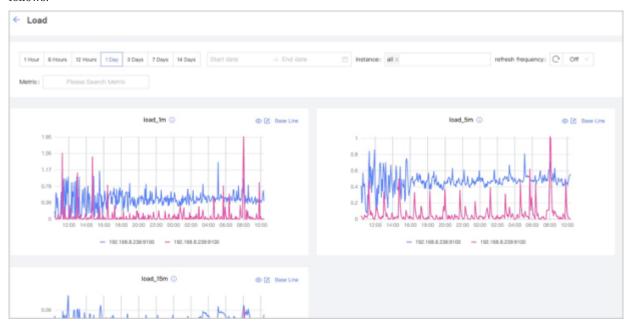


To set a base line, click the

button

To view the detailed monitoring information, click the follows.

button. In this example, select  $\ensuremath{\,\text{Load}\,}$  for details. The figure is as



- You can set the monitoring time range, instance, update frequency and base line.
- You can search for or select the target metric. For details about monitoring metrics, see Metrics.
- You can temporarily hide nodes that you do not need to view.

**o** 

You can click the

button to view the detailed monitoring information.

### 14.4.3 Service

Click **Service->Overview** to enter the service overview page.

On this page, you can view the information of Graph, Meta, and Storage services quickly. In the upper right corner, the number of normal services and abnormal services will be displayed.

- 616/1069 - 2023 Vesoft Inc.

#### Q Note

In the Service page, only two monitoring metrics can be set for each service, which can be adjusted by clicking the Set up button.

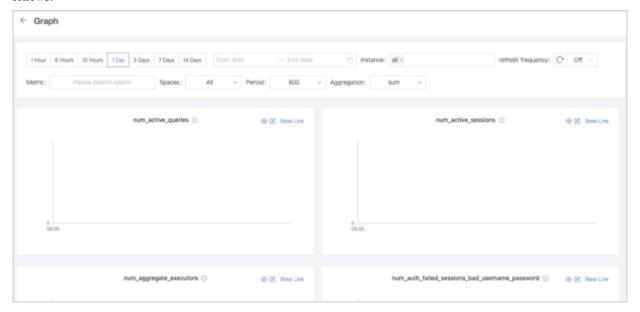
- By default, you can view the monitoring data for a maximum of 14 days. You can also select a time range or quickly select the latest 1 hour, 6 hours, 12 hours, 1 day, 3 days, 7 days, or 14 days.
- By default, you can view the monitoring data of all the instances in clusters. You can select the instances you want to view in the **instance** box.
- By default, the monitoring information page will not be updated automatically. You can set the update frequency of the monitoring information page globally or click the button to update the page manually.
- You can view the status of all the services in a cluster.

•



To view the detailed monitoring information, click the follows.

button. In this example, select  $\ensuremath{\mbox{\sf Graph}}$  for details. The figure is as



- You can set the monitoring time range, instance, update frequency, period, aggregation and base line.
- $\bullet \ \ \text{You can search for or select the target metric. For details of monitoring metrics, see $$Monitor parameter. }$
- You can temporarily hide nodes that you do not need to view.

0

You can click the button to view the detailed monitoring information.

• The Graph service supports a set of space-level metrics. For more information, see the following section **Graph space**.

### **Graph space**



Before using graph space metrics, you need to set <code>enable\_space\_level\_metrics</code> to true in the Graph service. For details, see [Graph Service configurations](../5.configurations-and-logs/1.configurations/3.graph-config.md.

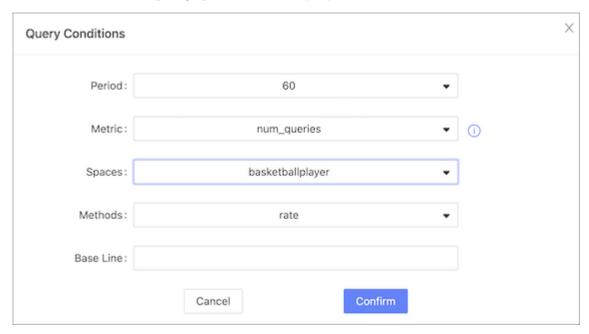
# Prace-level metric incompatibility

If a graph space name contains special characters, the corresponding metric data of that graph space may not be displayed.

The service monitoring page can also monitor graph space level metrics. Only when the behavior of a graph space metric is triggered, you can specify the graph space to view information about the corresponding graph space metric.

Space graph metrics record the information of different graph spaces separately. Currently, only the Graph service supports a set of space-level metrics.

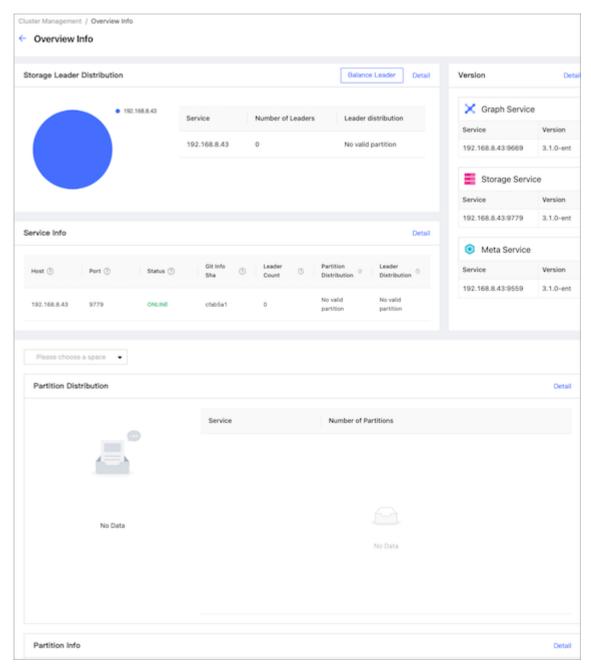
For information about the space graph metrics, see Graph space.



### 14.4.4 Management

### Overview info

On the **Overview Info** page, you can see the information of the NebulaGraph cluster, including Storage leader distribution, Storage service details, versions and hosts information of each NebulaGraph service, and partition distribution and details.



### STORAGE LEADER DISTRIBUTION

In this section, the number of Leaders and the Leader distribution will be shown.

- Click the **Balance Leader** button in the upper right corner to distribute Leaders evenly and quickly in the NebulaGraph cluster. For details about the Leader, see <u>Storage Service</u>.
- Click **Detail** in the upper right corner to view the details of the Leader distribution.

### VERSION

In this section, the version and host information of each NebulaGraph service will be shown. Click **Detail** in the upper right corner to view the details of the version and host information.

#### SERVICE INFORMATION

In this section, the information on Storage services will be shown. The parameter description is as follows:

Parameter	Description
Host	The IP address of the host.
Port	The port of the host.
Status	The host status.
Git Info Sha	The commit ID of the current version.
Leader Count	The number of Leaders.
Partition Distribution	The distribution of partitions.
Leader Distribution	The distribution of Leaders.

Click **Detail** in the upper right corner to view the details of the Storage service information.

#### PARTITION DISTRIBUTION

Select the specified graph space in the upper left corner, you can view the distribution of partitions in the specified graph space. You can see the IP addresses and ports of all Storage services in the cluster, and the number of partitions in each Storage service.

Click **Detail** in the upper right corner to view more details.

#### PARTITION INFORMATION

In this section, the information on partitions will be shown. Before viewing the partition information, you need to select a graph space in the upper left corner. The parameter description is as follows:

Parameter	Description
Partition ID	The ID of the partition.
Leader	The IP address and port of the leader.
Peers	The IP addresses and ports of all the replicas.
Losts	The IP addresses and ports of faulty replicas.

Click **Detail** in the upper right corner to view details. You can also enter the partition ID into the input box in the upper right corner of the details page to filter the shown data.

### Config

It shows the configuration of the NebulaGraph service. NebulaGraph Dashboard Community Edition does not support online modification of configurations for now.

### 14.4.5 Others

In the lower left corner of the page, you can:

- Sign out
- Switch between Chinese and English
- View the current Dashboard release
- View the user manual and forum
- Fold the sidebar

Last update: February 19, 2024

# 14.5 Metrics

This topic will describe the monitoring metrics in NebulaGraph Dashboard.

# 14.5.1 Machine



- $\bullet$  All the machine metrics listed below are for the Linux operating system.
- The default unit in **Disk** and **Network** is byte. The unit will change with the data magnitude as the page displays. For example, when the flow is less than 1 KB/s, the unit will be Bytes/s.
- For versions of Dashboard Community Edition greater than v1.0.2, the memory occupied by Buff and Cache will not be counted in the memory usage.

# CPU

Parameter	Description
cpu_utilization	The percentage of used CPU.
cpu_idle	The percentage of idled CPU.
cpu_wait	The percentage of CPU waiting for IO operations.
cpu_user	The percentage of CPU used by users.
cpu_system	The percentage of CPU used by the system.

### Memory

Parameter	Description
memory_utilization	The percentage of used memory.
memory_used	The memory space used (not including caches).
memory_free	The memory space available.

# Load

Parameter	Description
load_1m	The average load of the system in the last 1 minute.
Load_5m	The average load of the system in the last 5 minutes.
load_15m	The average load of the system in the last 15 minutes.

- 622/1069 - 2023 Vesoft Inc.

### Disk

Parameter	Description
disk_used_percentage	The disk utilization percentage.
disk_used	The disk space used.
disk_free	The disk space available.
disk_readbytes	The number of bytes that the system reads in the disk per second.
disk_writebytes	The number of bytes that the system writes in the disk per second.
disk_readiops	The number of read queries that the disk receives per second.
disk_writeiops	The number of write queries that the disk receives per second.
inode_utilization	The percentage of used inode.

### Network

Parameter	Description
network_in_rate	The number of bytes that the network card receives per second.
network_out_rate	The number of bytes that the network card sends out per second.
network_in_errs	The number of wrong bytes that the network card receives per second.
network_out_errs	The number of wrong bytes that the network card sends out per second.
network_in_packets	The number of data packages that the network card receives per second.
network_out_packets	The number of data packages that the network card sends out per second.

# 14.5.2 Service

### Period

The period is the time range of counting metrics. It currently supports 5 seconds, 60 seconds, 600 seconds, and 3600 seconds, which respectively represent the last 5 seconds, the last 1 minute, the last 10 minutes, and the last 1 hour.

### Metric methods

Parameter	Description
rate	The average rate of operations per second in a period.
sum	The sum of operations in the period.
avg	The average latency in the cycle.
P75	The 75th percentile latency.
P95	The 95th percentile latency.
P99	The 99th percentile latency.
P999	The 99.9th percentile latency.



Dashboard collects the following metrics from the NebulaGraph core, but only shows the metrics that are important to it.

- 623/1069 - 2023 Vesoft Inc.

# Graph

Parameter	Description
num_active_queries	The number of changes in the number of active queries.  Formula: The number of started queries minus the number of finished queries within a specified time.
num_active_sessions	The number of changes in the number of active sessions. Formula: The number of logged in sessions minus the number of logged out sessions within a specified time. For example, when querying <code>num_active_sessions.sum.5</code> , if there were 10 sessions logged in and 30 sessions logged out in the last 5 seconds, the value of this metric is -20 (10-30).
num_aggregate_executors	The number of executions for the Aggregation operator.
num_auth_failed_sessions_bad_username_password	The number of sessions where authentication failed due to incorrect username and password.
num_auth_failed_sessions_out_of_max_allowed	The number of sessions that failed to authenticate logins because the value of the parameter $FLAG_OUT_OF_MAX_ALLOWED_CONNECTIONS$ was exceeded.
num_auth_failed_sessions	The number of sessions in which login authentication failed.
num_indexscan_executors	The number of executions for index scan operators.
num_killed_queries	The number of killed queries.
num_opened_sessions	The number of sessions connected to the server.
num_queries	The number of queries.
num_query_errors_leader_changes	The number of the raft leader changes due to query errors.
num_query_errors	The number of query errors.
num_reclaimed_expired_sessions	The number of expired sessions actively reclaimed by the server.
num_rpc_sent_to_metad_failed	The number of failed RPC requests that the Graphd service sent to the Metad service.
num_rpc_sent_to_metad	The number of RPC requests that the Graphd service sent to the Metad service.
num_rpc_sent_to_storaged_failed	The number of failed RPC requests that the Graphd service sent to the Storaged service. $ \begin{tabular}{ll} \hline \end{tabular} $
num_rpc_sent_to_storaged	The number of RPC requests that the Graphd service sent to the Storaged service.
num_sentences	The number of statements received by the Graphd service.
num_slow_queries	The number of slow queries.
num_sort_executors	The number of executions for the Sort operator.
optimizer_latency_us	The latency of executing optimizer statements.
query_latency_us	The latency of queries.
slow_query_latency_us	The latency of slow queries.
num_queries_hit_memory_watermark	The number of queries reached the memory watermark.

- 624/1069 - 2023 Vesoft Inc.

# Meta

Parameter	Description
commit_log_latency_us	The latency of committing logs in Raft.
<pre>commit_snapshot_latency_us</pre>	The latency of committing snapshots in Raft.
heartbeat_latency_us	The latency of heartbeats.
num_heartbeats	The number of heartbeats.
num_raft_votes	The number of votes in Raft.
transfer_leader_latency_us	The latency of transferring the raft leader.
num_agent_heartbeats	The number of heartbeats for the AgentHBProcessor.
agent_heartbeat_latency_us	The latency of the AgentHBProcessor.
replicate_log_latency_us	The latency of replicating the log record to most nodes by Raft.
num_send_snapshot	The number of times that Raft sends snapshots to other nodes.
append_log_latency_us	The latency of replicating the log record to a single node by Raft.
append_wal_latency_us	The Raft write latency for a single WAL.
num_grant_votes	The number of times that Raft votes for other nodes.
num_start_elect	The number of times that Raft starts an election.

- 625/1069 - 2023 Vesoft Inc.

# Storage

Parameter	Description
add_edges_latency_us	The latency of adding edges.
add_vertices_latency_us	The latency of adding vertices.
commit_log_latency_us	The latency of committing logs in Raft.
commit_snapshot_latency_us	The latency of committing snapshots in Raft.
delete_edges_latency_us	The latency of deleting edges.
delete_vertices_latency_us	The latency of deleting vertices.
get_neighbors_latency_us	The latency of querying neighbor vertices.
get_dst_by_src_latency_us	The latency of querying the destination vertex by the source vertex.
num_get_prop	The number of executions for the GetPropProcessor.
num_get_neighbors_errors	The number of execution errors for the GetNeighborsProcessor.
num_get_dst_by_src_errors	The number of execution errors for the GetDstBySrcProcessor.
get_prop_latency_us	The latency of executions for the GetPropProcessor.
num_edges_deleted	The number of deleted edges.
num_edges_inserted	The number of inserted edges.
num_raft_votes	The number of votes in Raft.
num_rpc_sent_to_metad_failed	The number of failed RPC requests that the Storage service sent to the Meta service.
num_rpc_sent_to_metad	The number of RPC requests that the Storaged service sent to the Metad service.
num_tags_deleted	The number of deleted tags.
num_vertices_deleted	The number of deleted vertices.
num_vertices_inserted	The number of inserted vertices.
transfer_leader_latency_us	The latency of transferring the raft leader.
lookup_latency_us	The latency of executions for the LookupProcessor.
num_lookup_errors	The number of execution errors for the LookupProcessor.
num_scan_vertex	The number of executions for the ScanVertexProcessor.
num_scan_vertex_errors	The number of execution errors for the ScanVertexProcessor.
update_edge_latency_us	The latency of executions for the UpdateEdgeProcessor.
num_update_vertex	The number of executions for the UpdateVertexProcessor.
num_update_vertex_errors	The number of execution errors for the UpdateVertexProcessor.
kv_get_latency_us	The latency of executions for the Getprocessor.
kv_put_latency_us	The latency of executions for the PutProcessor.
kv_remove_latency_us	The latency of executions for the RemoveProcessor.
num_kv_get_errors	The number of execution errors for the GetProcessor.
num_kv_get	The number of executions for the GetProcessor.
num_kv_put_errors	The number of execution errors for the PutProcessor.
num_kv_put	The number of executions for the PutProcessor.

- 627/1069 - 2023 Vesoft Inc.

Parameter	Description
num_kv_remove_errors	The number of execution errors for the RemoveProcessor.
num_kv_remove	The number of executions for the RemoveProcessor.
forward_tranx_latency_us	The latency of transmission.
scan_edge_latency_us	The latency of executions for the ScanEdgeProcessor.
num_scan_edge_errors	The number of execution errors for the ScanEdgeProcessor.
num_scan_edge	The number of executions for the ScanEdgeProcessor.
scan_vertex_latency_us	The latency of executions for the ScanVertexProcessor.
num_add_edges	The number of times that edges are added.
num_add_edges_errors	The number of errors when adding edges.
num_add_vertices	The number of times that vertices are added.
num_start_elect	The number of times that Raft starts an election.
num_add_vertices_errors	The number of errors when adding vertices.
num_delete_vertices_errors	The number of errors when deleting vertices.
append_log_latency_us	The latency of replicating the log record to a single node by Raft.
num_grant_votes	The number of times that Raft votes for other nodes.
replicate_log_latency_us	The latency of replicating the log record to most nodes by Raft.
num_delete_tags	The number of times that tags are deleted.
num_delete_tags_errors	The number of errors when deleting tags.
num_delete_edges	The number of edge deletions.
num_delete_edges_errors	The number of errors when deleting edges
num_send_snapshot	The number of times that snapshots are sent.
update_vertex_latency_us	The latency of executions for the UpdateVertexProcessor.
append_wal_latency_us	The Raft write latency for a single WAL.
num_update_edge	The number of executions for the UpdateEdgeProcessor.
delete_tags_latency_us	The latency of deleting tags.
num_update_edge_errors	The number of execution errors for the UpdateEdgeProcessor.
num_get_neighbors	The number of executions for the GetNeighborsProcessor.
num_get_dst_by_src	The number of executions for the GetDstBySrcProcessor.
num_get_prop_errors	The number of execution errors for the GetPropProcessor.
num_delete_vertices	The number of times that vertices are deleted.
num_lookup	The number of executions for the LookupProcessor.
num_sync_data	The number of times the Storage service synchronizes data from the Drainer.
num_sync_data_errors	The number of errors that occur when the Storage service synchronizes data from the Drainer.
sync_data_latency_us	The latency of the Storage service synchronizing data from the Drainer.

- 628/1069 - 2023 Vesoft Inc.

# **Graph** space



Space-level metrics are created dynamically, so that only when the behavior is triggered in the graph space, the corresponding metric is created and can be queried by the user.

Parameter	Description
num_active_queries	The number of queries currently being executed.
num_queries	The number of queries.
num_sentences	The number of statements received by the Graphd service.
optimizer_latency_us	The latency of executing optimizer statements.
query_latency_us	The latency of queries.
num_slow_queries	The number of slow queries.
num_query_errors	The number of query errors.
num_query_errors_leader_changes	The number of raft leader changes due to query errors.
num_killed_queries	The number of killed queries.
num_aggregate_executors	The number of executions for the Aggregation operator.
num_sort_executors	The number of executions for the Sort operator.
num_indexscan_executors	The number of executions for index scan operators.
num_auth_failed_sessions_bad_username_password	The number of sessions where authentication failed due to incorrect username and password.
num_auth_failed_sessions	The number of sessions in which login authentication failed.
num_opened_sessions	The number of sessions connected to the server.
num_queries_hit_memory_watermark	The number of queries reached the memory watermark.
num_reclaimed_expired_sessions	The number of expired sessions actively reclaimed by the server.
num_rpc_sent_to_metad_failed	The number of failed RPC requests that the Graphd service sent to the Metad service.
num_rpc_sent_to_metad	The number of RPC requests that the Graphd service sent to the Metad service.
num_rpc_sent_to_storaged_failed	The number of failed RPC requests that the Graphd service sent to the Storaged service.
num_rpc_sent_to_storaged	The number of RPC requests that the Graphd service sent to the Storaged service.
slow_query_latency_us	The latency of slow queries.

- 629/1069 - 2023 Vesoft Inc.

# Single process metrics

Graph, Meta, and Storage services all have their own single process metrics.

Parameter	Description
context_switches_total	The number of context switches.
cpu_seconds_total	The CPU usage based on user and system time.
memory_bytes_gauge	The number of bytes of memory used.
open_filedesc_gauge	The number of file descriptors.
read_bytes_total	The number of bytes read.
write_bytes_total	The number of bytes written.

Last update: February 19, 2024

- 630/1069 - 2023 Vesoft Inc.

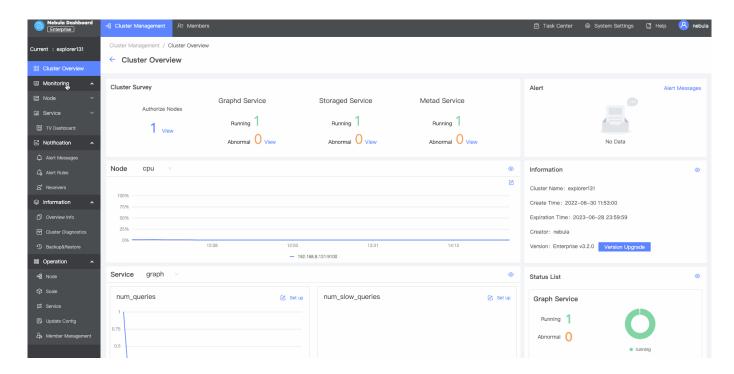
# 15. NebulaGraph Dashboard Enterprise Edition

# 15.1 What is NebulaGraph Dashboard Enterprise Edition

NebulaGraph Dashboard Enterprise Edition (Dashboard for short) is a visualization tool that monitors and manages the status of machines and services in NebulaGraph clusters. This topic introduces Dashboard Enterprise Edition. For more information, see What is NebulaGraph Dashboard Community Edition.



You can also try some functions online in Dashboard.



### 15.1.1 Features

- Create a NebulaGraph cluster of a specified version, import nodes in batches, scale out NebulaGraph services with one click
- Import clusters, balance data, scale out or in on the visualization interface.
- Manage clusters, and view the operation log of clusters.
- Start, stop, and restart services on the visualization interface.
- Update the configuration of Storage services and Graph services in clusters quickly.
- Set how often the metrics page refreshes.
- Monitor the information of all the services in clusters, including the IP address, version, and monitoring metrics (such as the number of queries, the latency of queries, and the latency of heartbeats).
- Monitor the status of all the machines in clusters, including CPU, memory, load, disk, and network.
- Monitor the information of clusters, including the information of services, partitions, configurations, and long-term tasks.
- Set notifications based on the monitoring information.

- 631/1069 - 2023 Vesoft Inc.

# 15.1.2 Scenarios

- You want a visualized operation and maintenance monitoring platform for large-scale clusters.
- You want to monitor key metrics conveniently and quickly, and present multiple key information of the business to ensure that the business can be operated normally.
- You want to monitor clusters from multiple dimensions (such as the time, aggregate rules, and metrics).
- You want to review the failure after it occurs, confirm when it happened, and view its associated phenomena.

### 15.1.3 Precautions

- The monitoring data will be retained for 14 days by default, that is, only the monitoring data within the last 14 days can be queried.
- The version of NebulaGraph must be 2.5.0 or later.
- It is recommend to use the latest version of Chrome to access Dashboard.
- It is recommend to use the official installation package to create or import clusters.



The monitoring feature is supported by Prometheus. The update frequency and retention intervals can be modified. For details, see Prometheus.

### 15.1.4 Version compatibility

The version correspondence between NebulaGraph and Dashboard Enterprise Edition is as follows.

NebulaGraph version	Dashboard version
3.4.0 ~ 3.4.1	3.4.2, 3.4.1, 3.4.0, 3.2.4, 3.2.3, 3.2.2, 3.2.1, 3.2.0
3.3.0	3.2.4, 3.2.3, 3.2.2, 3.2.1, 3.2.0
2.5.0 ~ 3.2.0	3.1.2, 3.1.1, 3.1.0
$2.5.x \sim 3.1.0$	3.0.4
2.5.1 ~ 3.0.0	1.1.0
2.0.1 ~ 2.6.1	1.0.2
2.0.1 ~ 2.6.1	1.0.1
2.0.1 ~ 2.6.1	1.0.0

# 15.1.5 Video

• NebulaGraph Dashboard (Enterprise Edition) Intro Demo(5 minutes 25 seconds)

Last update: February 19, 2024

- 632/1069 - 2023 Vesoft Inc.

# 15.2 Deploy Dashboard Enterprise Edition

This topic will introduce how to deploy Dashboard Enterprise Edition in detail.

### 15.2.1 Prerequisites

Before deploying Dashboard Enterprise Edition, you must do a check of these:

- Select and download Dashboard Enterprise Edition of the correct version. For information about the version correspondence between Dashboard Enterprise Edition and NebulaGraph, see Version compatibility.
- MySQL and SQLite are supported to store Dashboard metadata. To use MySQL, make sure that the environment of MySQL is ready and a MySQL database named as dashboard is create. Make sure the default character set of the database is utf8.



To store Dashboard metadata using SQLite, there is no need to configure the SQLite environment because SQLite is built-in to Dashboard

• Before the installation starts, the following ports are not occupied.

Port	Description
7005	The port through which Dashboard Enterprise Edition provides the web service.
9090	The port of the prometheus service.
9200	The port of the nebula-stats-exporter service.
9093	The port of the Alertmanager service, used to receive Prometheus alerts and then send them to Dashboard.
9100	The port of the node-exporter service. The node-exporter is aumatically deployed on the target machine after a cluster is created or imported. It is used to collect the source information of machines in the cluster, including the CPU, memory, load, disk, and network.

• The license is ready.



The license is only available in the Enterprise Edition. To obtain the license, apply for NebulaGraph Dashboard Free Trial.

# 15.2.2 Deploy Dashboard Enterprise Edition with TAR

### Installation

1. Select and download the TAR package according to your needs. It is recommended to select the latest version.



You can apply online for Dashboard Enterprise Edition free trial. Contact us to purchase. For features of Dashboard Enterprise Edition, see Pricing.

2. Use tar -xzvf to decompress the TAR package.

- 633/1069 - 2023 Vesoft Inc.

tar -xzvf nebula-dashboard-ent-<version>.linux-amd64.tar.gz -C <install\_path>

#### For example:

tar -xzvf nebula-dashboard-ent-3.4.2.linux-amd64.tar.gz -C /usr/local/

3. Edit vim /usr/local/nebula-dashboard-ent/etc/config.yaml to modify the configuration.

```
# Specifies the address segment that can access Dashboard.
MaxBytes: 1073741824 # The maximum content length of an Http request that can be accepted. The default value is 1048576. Value range: 0 ~ 8388608.
Timeout: 15000 # Timeout duration of the access
             # Whether to enable the Debug mode.
  Enable: false
        # Dashboard run log settings
Log:
KeepDays: 7 # The number of days for keeping log.

Mode: console # The save mode of logs, including console and file. console means the service logs are logged in webserver.log. file means the service logs are logged in access.log, error.log, severe.log, slow.log, and stat.log respectively.
  Dialect: sqlite # The database type used to store metadata. Only support SQLite and MySQL currently. The default value is SQLite.
  AutoMigrate: true # Whether to automatically create a database table. Defaults to true
  Host: 127.0.0.1 # The IP address of the connected MySQL database.
Port: 3306 # The port of the connected MySQL database.
  Username: root # The username to log in MySQL
  Password: nebula # The password to log in MySQL.
Name: dashboard # The name of the corresponding database.
# Information about the exporter port
Exporter:
  NodePort: 9100 # The port of the node-exporter service
  NebulaPort: 9200 # The port of the nebula-stats-exporter service.
# Information of services
Proxy:
  PrometheusAddr: 127.0.0.1:9090 # The IP address and port of the prometheus service.
  AlertmanagerAddr: 127.0.0.1:9093 # The IP address and port of the Alertmanager service.
# Information of the sender's Email used to invite LDAP accounts.
Mail:
  Host: smtp.office365.com # The SMTP server address.
  Port: 587 # The port number of the SMTP server Username: "" # The SMTP server account name.
  Password: "" # The SMTP server password.
# System information
System:
  WebAddress: http://127.0.0.1:7005
# The external access for Dashboard. It can be set as a hostname, used for interface callbacks. For example, the invitee who is invited by mail can use this link to access Dashboard.
  MessageStore: 90 # It sets the number of days to keep alert messages, the value of which is 90 by default.
# LDAP information
LDAP:
  Server: ldap://127.0.0.1 # The LDAP server address.
  BindDN: cn=admin,dc=vesoft,dc=com # The LDAP login username
BindPassword: "" # The LDAP login password.
  BaseDN: dc=vesoft,dc=com # Set the path to query user data.
  UserFilter: "&(objectClass=*)" # Set a filter to LDAP search queries
  EmailKey: mail # Set the field name used to restore email in LDAP.
```

4. Copy the license file to the nebula-dashboard-ent directory.

```
cp -r <license> <dashboard_path>
```

#### For example:

```
cp -r nebula.license /usr/local/nebula-dashboard-ent
```

#### 5. Start Dashboard.

You can use the following command to start the Dashboard with one click.

```
cd /usr/local/nebula-dashboard-ent/scripts
sudo ./dashboard.service start all
```

Or execute the following commands to start prometheus, webserver, nebula-stats-exporter and alertmanager services to start Dashboard.

```
cd scripts
sudo ./dashboard.service start prometheus # Start prometheus service
sudo ./dashboard.service start webserver # Start webserver service
sudo ./dashboard.service start exporter # Start nebula-stats-exporter service
sudo ./dashboard.service start alertmanager # Start alertmanager service
```



If you change the configuration file after starting Dashboard, you can run dashboard.service restart all in the scripts directory to synchronize the changes to the Dashboard client page.

#### 15.2.3 Deploy Dashboard Enterprise Edition with RPM

#### Installation

1. Download an RPM package.



You can apply online for Dashboard Enterprise Edition free trial. Contact us to purchase. For features of Dashboard Enterprise Edition, see Pricing.

2. Run sudo rpm -i <rpm> to install the RPM package.

For example, run the following command to install Dashboard Enterprise Edition. Installation path is /usr/local/nebula-dashboard-ent by default.

```
sudo rpm -i nebula-dashboard-ent-<version>.x86 64.rpm
```

You can also run the following command to specify the installation path.

```
sudo rpm -i nebula-dashboard-ent-xxx.rpm --prefix=<path>
```

3. Copy the license file to the nebula-dashboard-ent directory.

```
cp -r <license> <dashboard_path>
```

#### For example:

```
cp -r nebula.license /usr/local/nebula-dashboard-ent
```

4. (Optional) Run the following commands to view the status of and start all the services.

```
sudo systemctl list-dependencies nebula-dashboard.target # View the status of all the services.
sudo systemctl start nebula-dashboard.target # Start all the services.
```

You can also view, start, and stop a single service. For example:

```
sudo\ systemctl\ \{status|stop|start\}\ \{nbd-prometheus.service|nbd-alert-manager.service|nbd-stats-exporter.service|nbd-webserver.service\}
```

5. (Optional) To configure recipients of cluster alert notifications and to configure LDAP accounts, run vim /usr/local/nebula-dashboard-ent/etc/config.yaml and add the following settings.

```
# Information of the sender's Email used to invite LDAP accounts.
mail:
host: smtp.office365.com # The SMTP server address.
port: 587 # The port number of the SMTP server.
username: "" # The SMTP server account name.
password: "" # The SMTP server password.
# System information
system:
webAddress: http://127.0.0.1:7005 # The address to access Dashboard for the invitee who is invited by mail.
messageStore: 90 # It sets the number of days to keep alert messages, the value of which is 90 by default.
# LDAP information
Idap:
server: ldap://127.0.0.1 # The LDAP server address.
bindDN: cn=admin,dc=vesoft,dc=com # The LDAP login username.
bindPassword: "" # The LDAP login password.
baseDN: dc=vesoft,dc=com # Set the path to query user data.
userFilter: "&(objectClass=")" # Set a filter to LDAP search queries.
emailKey: mail # Set the field name used to restore email in LDAP.
```

#### Uninstallation

To uninstall Dashboard Enterprise Edition deployed with RPM, run the following command.

```
sudo rpm -e <package_name>
```

### 15.2.4 Deploy Dashboard Enterprise Edition with DEB

#### Installation

1. Download a DEB package.



You can apply online for Dashboard Enterprise Edition free trial. Contact us to purchase. For features of Dashboard Enterprise Edition, see Pricing.

2. Install the package.

sudo dpkg -i <package\_name>



Custom installation paths are not supported when installing Dashboard Enterprise Edition with DEB. The default installation path is / usr/local/nebula-dashboard-ent/.

For example, to install the DEB package of the 3.4.2 version:

```
sudo dpkg -i nebula-dashboard-ent-3.4.2.ubuntu1804.amd64.deb
```

3. Copy the license file to the nebula-dashboard-ent directory.

```
cp -r <license> <dashboard_path>
```

### For example:

```
cp -r nebula.license /usr/local/nebula-dashboard-ent
```

4. (Optional) Run the following commands to view the status of and start all the services.

```
sudo systemctl list-dependencies nebula-dashboard.target # View the status of all the services.
sudo systemctl start nebula-dashboard.target # Start all the services.
```

You can also view, start, and stop a single service. For example:

```
sudo systemctl {status|stop|start} {nbd-prometheus.service|nbd-alert-manager.service|nbd-stats-exporter.service|nbd-webserver.service}
```

5. (Optional) To configure the sender's email address used to invite LDAP and OAuth2.0 accounts and configure the duration for storing alert messages, run vim /usr/local/nebula-dashboard-ent/etc/config.yaml and add the following settings.

```
# Information of the sender's Email used to invite LDAP accounts.
mail:
host: smtp.office365.com # The SMTP server address.
port: $87 # The port number of the SMTP server.
username: "" # The SMTP server account name.
password: "" # The SMTP server password.
# System information
system:
webAddress: http://127.0.0.1:7005 # The address to access Dashboard for the invitee who is invited by mail.
messageStore: 90 # It sets the number of days to keep alert messages, the value of which is 90 by default.
```

- 636/1069 - 2023 Vesoft Inc.

#### Uninstallation

To uninstall Dashboard Enterprise Edition, run the following command.

```
sudo dpkg -r <package_name>
```

### 15.2.5 Manage services in Dashboard

You can use the dashboard.service script to start, restart, stop, and check the Dashboard services.

Parameter	Description
dashboard_path	Dashboard installation path.
-v	Display detailed debugging information.
-h	Display help information.
start	Start the target services.
restart	Restart the target services.
stop	Stop the target services.
status	Check the status of the target services.
prometheus	Set the prometheus Service as the target service.
webserver	Set the webserver Service as the target service.
exporter	Set the exporter Service as the target service.
gateway	Set the gateway Service as the target service.
all	Set all the Dashboard services as the target services.



To view the Dashboard version, run the command ./dashboard.service -version.

### Examples

Dashboard is installed in the current directory, and you can use the following commands to manage services.

```
sudo /dashboard/scripts/dashboard.service start all #Start Dashboard.
sudo /dashboard/scripts/dashboard.service stop all #Stop Dashboard.
sudo /dashboard/scripts/dashboard.service status all #Check Dashboard status.
sudo /dashboard/scripts/dashboard.service restart all #Restart Dashboard.
```

### 15.2.6 View logs

You can view the Dashboard Enterprise Edition logs in the logs path.

For example:

cat logs/prometheus.log

The descriptions of the log files are as follows.

Log file	Description
alertmanager.log	Alertmanager service log.
nebula-stats-exporter.log	nebula-stats-exporter service log.
prometheus.log	Prometheus service log.
br	Backup and restore service log.
webserver.log	Dashboard service log. It takes effect only when the $\ensuremath{Log.Mode}$ in the Dashboard configuration is console .
access.log	$Access \ log. \ It \ takes \ effect \ only \ when \ the \ {\tt Log.Mode} \ in \ the \ {\tt Dashboard} \ configuration \ is \ {\tt file} \ .$
error.log	Error $\log$ . It takes effect only when the $\log$ Mode in the Dashboard configuration is file.
severe.log	Severe log. It takes effect only when the $\ensuremath{Log}.\ensuremath{Mode}$ in the Dashboard configuration is file.
slow.log	Slow log. It takes effect only when the $\ensuremath{Log.Mode}$ in the Dashboard configuration is $\ensuremath{file}$ .
stat.log	Statistic log. It takes effect only when the $\ensuremath{Log.Mode}$ in the Dashboard configuration is $\ensuremath{file}$ .

# 15.2.7 Next to do

### Connect to Dashboard

Last update: February 19, 2024

- 638/1069 - 2023 Vesoft Inc.

### 15.3 Connect Dashboard

After Dashboard is deployed, you can log in and use Dashboard on the browser.

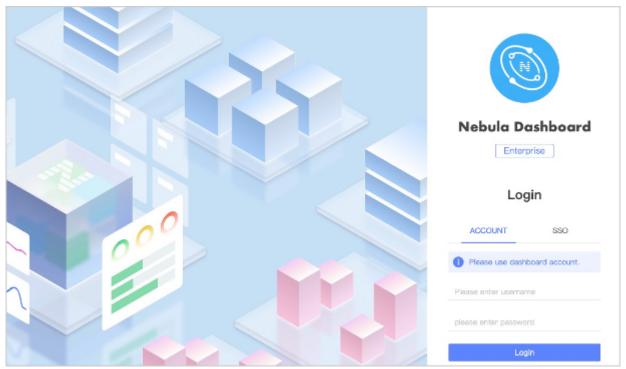
### 15.3.1 Prerequisites

- The Dashboard services are started. For more information, see Deploy Dashboard.
- We recommend you to use the Chrome browser of the version above 89. Otherwise, there may be compatibility issues.

#### 15.3.2 Procedures

1. Confirm the IP address of the machine where the Dashboard is installed. Enter http://<ip\_address>:7005 in the browser to open the login page.

If the following login interface is shown in the browser, then you have successfully deployed and started Dashboard.





When logging into the NebulaGraph Dashboard Enterprise Edition for the first time, the content of *END USER LICENSE AGREEMENT* is displayed on the login page. Please read it and then click **I Agree**.

2. Log into Dashboard with the default account name nebula and password nebula.



You can create LDAP, OAuth2.0 or general accounts after log into Dashboard. For more information about the Dashboard account, see Authority Management.

Last update: February 19, 2024

# 15.4 NebulaGraph Dashboard Enterprise Edition license

A license is a software authorization certificate used to authorize the use of a software product. When deploying NebulaGraph Dashboard Enterprise Edition, you need to add a license to start it. This document describes the license information on NebulaGraph Dashboard Enterprise Edition.

#### 15.4.1 Precautions

- If the license file is not deployed, NebulaGraph Dashboard Enterprise Edition cannot be started.
- Do not modify the license file, otherwise the license will become invalid.
- If the license is about to expire, contact us to apply for renewal.
- The transition period after the license expires is 14 days:
- If you start the Enterprise Edition within 30 days before the license expires or on the day the license expires, a log will be printed as a reminder.
- The license can still be used for 14 days after it expires.
- If the license has expired for 14 days, you will not be able to start the Enterprise Edition, and a log will be printed as a reminder.

### 15.4.2 Obtain a NebulaGraph Dashboard Enterprise Edition license

Contact us to apply for a NebulaGraph Dashboard Enterprise Edition license.



You can apply online for a 30-day free trial of NebulaGraph Dashboard Enterprise Edition.

### 15.4.3 License description

NebulaGraph Dashboard Enterprise Edition license is a file named nebula. License that contains the following information:

- 640/1069 - 2023 Vesoft Inc.

The license file contains information such as issuedDate and expirationDate. The description is as follows.

Parameter	Description
vendor	The supplier.
organization	The username.
issuedDate	The date that the license is issued.
expirationDate	The date that the license expires.
product	$The \ product \ type. \ The \ product \ type \ of \ Nebula Graph \ Dashboard \ Enterprise \ Edition \ is \ \ nebula\_graph\_dashboard \ .$
version	The version information.
licenseType	The license type (a reserved parameter), including enterprise, samtl_bussiness, pro, and individual.
gracePeriod	The buffer time (in days) for the service to continue to be used after the license expires, and the service will be stopped after the buffer period. The trial version of license has no buffer period after expiration and the default value of this parameter is 0.
clusterCode	The user's hardware information, which is also the unique identifier of the cluster. This parameter is not available in the trial version of the license.

### 15.4.4 Use a NebulaGraph Dashboard Enterprise Edition license

For how to use a NebulaGraph Dashboard Enterprise Edition license, see Deploy NebulaGraph Dashboard Enterprise Edition.

# 15.4.5 Renew a NebulaGraph Dashboard Enterprise Edition license

Follow the steps below to renew your NebulaGraph Dashboard Enterprise Edition license.

- 1. Contact us to apply for a new NebulaGraph Dashboard Enterprise Edition license file nebula. License.
- 2. In the NebulaGraph Dashboard Enterprise Edition installation directory, such as /usr/local/nebula-explorer, replace the old license file with the new one.



You cannot log into NebulaGraph Dashboard Enterprise Edition once the license expires. To avoid business interruptions, please renew your license in time.

Last update: February 19, 2024

- 641/1069 - 2023 Vesoft Inc.

# 15.5 Create and import clusters

# 15.5.1 Create clusters

This topic introduces how to create clusters using Dashboard.

# Steps

You can create a cluster following these steps:

- 1. At the top of the Dashboard page, click the  ${\bf Cluster\ Management}$  button.
- 2. On the **Cluster management** page, click **Create cluster**.

- $_{
  m 3.}$  On the  ${f Create\ cluster}$  page, fill in the following:
- Enter a **Cluster Name**, up to 15 characters for each name.
- Choose a NebulaGraph version to install.



Only one Enterprise Edition of NebulaGraph is provided for you to choose from on the **Create cluster** page. To install other versions of NebulaGraph, you can download or upload the corresponding installer package on the **Package Management** page. For details, see Package management.

• Click Upload License.

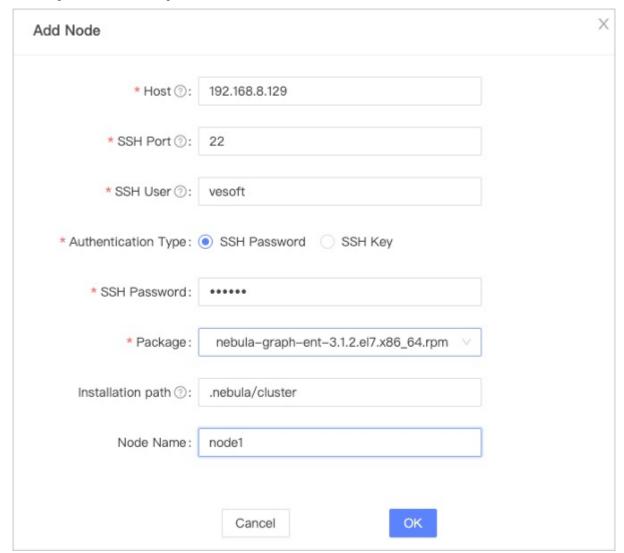


For the creation of a Community version of NebulaGraph, skip this step to upload the License file.

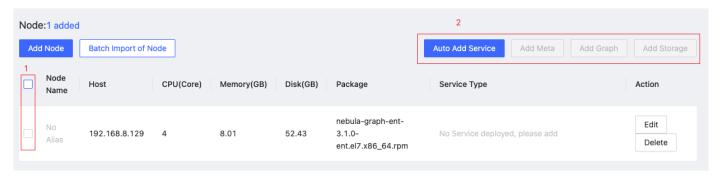
- 645/1069 - 2023 Vesoft Inc.

- Add nodes. Enter the following information, the Host, SSH port, SSH user, authentication type, NebulaGraph package, etc.

  The authentication type is described as follows:
- SSH Password: Enter the password of the SSH user.
- SSH Key: Click **Upload** and select the private key file of the node. You need to generate the secret key files on the node to be added and send the private key file to the current computer (not the machine where Dashboard is deployed). If the passphrase is set, this parameter is also required.



- Import nodes in batches. The information of each node is required. To import nodes in batches, you need to choose the installation package and click **download the CSV template**. Fill in the template and upload it. Ensure that the node is correct, otherwise, upload failure may happen.
- 4. Select the node and add the service you need in the upper right corner. To create a cluster, you need to add 3 types of services to the node. If not familiar with the NebulaGraph architecture, click **Auto add service**.



- 5. (Optional) Edit the port and HTTP port of the meta, graph, and storage services, and then click **OK**.
- 6. Click Create Cluster. Make sure the configuration is correct and there is no conflict between nodes, and then click Confirm.
- 7. If a cluster with the status of installing appears in the list on the cluster management page, you need to wait for 3 to 10 minutes until the status changes to healthy, that is, the cluster is created successfully. If the service status is unhealthy, it means that there is an abnormal service in the cluster, click **Detail** for more information.

#### Next to do

After the cluster is successfully created, you can operate on the cluster. For details, see Cluster operations.

Last update: February 19, 2024

# 15.5.2 Import clusters

This topic introduces how to import clusters using Dashboard. The current version only supports importing clusters deployed by the official DEB or RPM packages and clusters created by Dashboard. Currently, importing clusters deployed by Docker and Kubernetes is not supported.

# Steps



In the same cluster, the service versions need to be unified. Importing NebulaGraph examples from different versions in the same cluster is not supported.

1. In the configuration files of each service, change the IP in <meta|graph|storage>\_server\_addrs and local\_ip to the server's IP, and then start NebulaGraph.

For details, see Configurations and Manage NebulaGraph services.

- 2. On the Cluster management page, click Import cluster.
- 3. On the Import cluster page, enter the information of Connect to NebulaGraph.
- Graphd Host: :n. In this example, the IP is 192.168.8.157:9669.
- Username: The account to connect to NebulaGraph. In this example, the username is vesoft.
- Password: The password to connect to NebulaGraph. In this example, the password is nebula.



By default, authentication is disabled in NebulaGraph. Therefore, you can use root as the username and any password to connect to NebulaGraph. When authentication is enabled in NebulaGraph, you need to use the specified username and password to connect to NebulaGraph. For details of authentication, see NebulaGraph manual.

- 4. On the NebulaGraph connection panel, fill in the following:
- Enter the cluster name, 15 characters at most. In this example, the cluster name is create\_1027, and choose whether to use sudo to connect to the cluster.



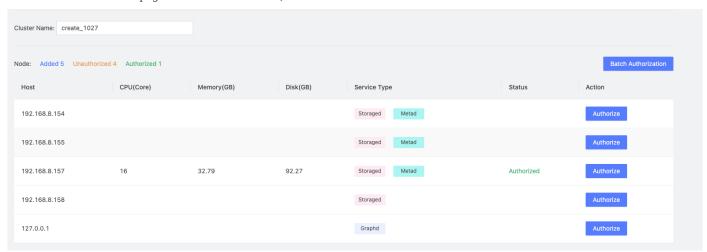
If your SSH account does not have permission for the NebulaGraph cluster, you can use sudo to connect to it.

• Authorize the node. The SSH username and password of each node are required, and choose to run sudo or not.



If your SSH account has no permission to operate NebulaGraph, but can execute sudo commands without password, set **use sudo** to **yes**.

- **Batch authorization** requires uploading the CSV file. Edit the authentication information of each node according to the downloaded CSV file. Ensure that the node information is correct, otherwise upload failure may happen.
- If the node status on the page becomes authorized, the node authentication is successful.



 $5.\ Ensure$  that all nodes are authorized successfully. Click  $\mathbf{Import}$   $\mathbf{cluster}.$ 

# Next to do

After the cluster is successfully imported, you can operate the cluster. For details, see Overview.

Last update: February 19, 2024

# 15.6 Cluster management

#### 15.6.1 Cluster overview

This topic introduces the **Overview** page of Dashboard.

At the top of the Dashboard page, click **Cluster Management**, and then click **Detail** on the right of the cluster management page to check the overview of a specified cluster.

#### Overview

The **Overview** page has the following parts:

- · Cluster survey
- Alert
- Information
- Node
- Status list
- Service

#### CLUSTER SURVEY

In this part, you can view the number of nodes as well as the number of running and abnormal services of Graphd, Storaged, and Metad. You can click the **View** button to quickly check the abnormal service and node.

#### ALERT

In the **Alert** section, the system displays the five most recently triggered alert messages according to their severity levels (emergency > critical > warning).

In the right upper corner, click Alert Messages to view alert messages. For more information about alerts, see Alerts.

#### INFORMATION

In this part, you can view the information of Cluster Name, Creation Time, Expiration Time, Creator, and Version.

- Cluster Name: The name of the cluster.
- Creation Time: The time when the cluster is created.
- $\bullet$   $\mathbf{Expiration}$   $\mathbf{Time}:$  The time when the cluster license expires.



The parameter **Expiration Time** is displayed only if the created or imported cluster is an Enterprise Edition cluster.

- Creator: The Dashboard account that is used to create the cluster.
- **Version**: The version of NebulaGraph installed in the cluster. The **Version Upgrade** button is displayed on the right to go to the page of version upgrade.



to view the cluster details, including name, creation time, account

name, version, and the role of the account name.

# (S) terpriseonly

For Enterprise Edition, there is a **License** section.

- Displays the license details of the cluster, including the usage status, the organization, the creation time, the expiration time, the cluster versions supported by the license, and the license type.
- Supports uploading new licenses. When the license expires, you cannot perform operations on the current cluster. Click **Upload License** to upload a new license.

#### NODE

• You can view the information of node monitoring quickly and change the displayed information. By default, the CPU information will be shown.



You can click

on the page to insert a base line.



You can click

to jump to the detailed node monitoring page.

#### TOLLOUTATO

This part uses pie charts to visually display the running status of nodes.

#### SERVICE

• By default, the information of query\_latency\_us and slow\_query\_latency\_us will be shown.



You can click

**Set up** to insert a base line.



You can click

View to jump to the detailed service monitoring page.

Last update: February 19, 2024

# 15.6.2 Cluster monitoring

This topic introduces node monitoring, service monitoring, graph space monitoring, and TV Dashboard.

At the top navigation bar of the Dashboard Enterprise Edition page, click **Cluster Management**, and click **Detail** at the right of the target cluster. **Monitoring** at the left navigation bar contains **Node**, **Service** and **TV Dashboard**.

#### Node

Click Monitoring->Node->Overview to enter the node monitoring overview page.

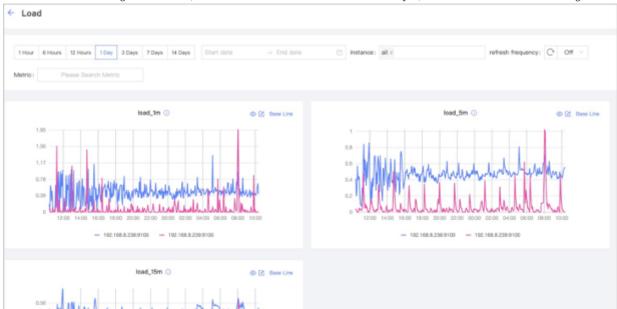
button.

On this page, you can view the variation of CPU, Memory, Load, Disk, and Network In/Out quickly.

- By default, you can view the monitoring data of the maximum of 14 days. You can also select a time range or quickly select latest 1 hour, 6 hours, 12 hours, 1 day, 3 days, 7 days, or 14 days.
- By default, you can view the monitoring data of all the instances in clusters. You can select the instances you want to view in the **instance** box.
- By default, the monitoring information page will not be updated automatically. You can set the update frequency of the monitoring information page globally or click the button to update the page manually.
  - To set a base line, click the

To view the detailed monitoring information, click the

button. In this example, select Load for details. The figure is as



follows.

- You can set the monitoring time range, instance, update frequency and base line.
- You can search for or select the target metric. For details about monitoring metrics, see Monitor parameter.
- You can temporarily hide nodes that you do not need to view.

You can click the

button to view the detailed monitoring information.

- 654/1069 - 2023 Vesoft Inc.

#### Service

Click Monitoring->Service->Overview to enter the service monitoring overview page.

On this page, you can view the information of Graph, Meta, and Storage services quickly. In the upper right corner, the number of normal services and abnormal services will be displayed.



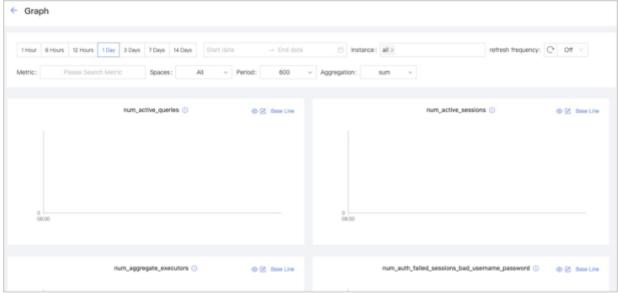
In the current **Service** page of the Enterprise Edition, only two monitoring metrics can be set for each service, which can be adjusted by clicking the **Set up** button.

- By default, you can view the monitoring data of the maximum of 14 days. You can also select a time range or quickly select latest 1 hour, 6 hours, 12 hours, 1 day, 3 days, 7days, or 14 days.
- By default, you can view the monitoring data of all the instances in clusters. You can select the instances you want to view in the **instance** box.
- By default, the monitoring information page will not be updated automatically. You can set the update frequency of the monitoring information page globally or click the button to update the page manually.
- You can view the status of all the services in cluster.

To view the detailed monitoring information, click the



button. In this example, select Graph for details. The figure is as



- follows.
- You can set the monitoring time range, instance, update frequency, period, aggregation and base line.
- You can search for or select the target metric. For details of monitoring metrics, see Monitor parameter.
- You can temporarily hide nodes that you do not need to view.

**©** 

You can click the button to view the detailed monitoring information.

• The Graph service supports a set of space-level metrics. For more information, see the following section **Graph space**.

- 655/1069 - 2023 Vesoft Inc.

**GRAPH SPACE** 



Before using graph space metrics, you need to set enable\_space\_level\_metrics to true in the Graph service. For details, see Update config.

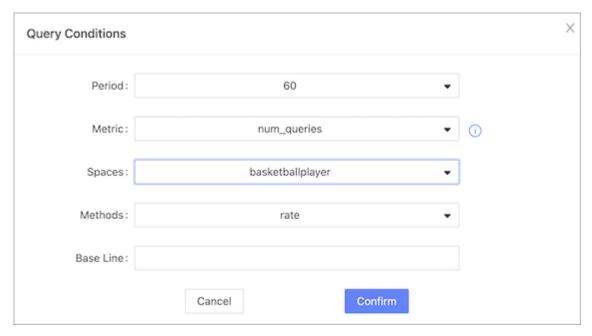


If a graph space name contains special characters, the corresponding metric data of that graph space may not be displayed.

The service monitoring page can also monitor graph space level metrics. Only when the behavior of a graph space metric is triggered, you can specify the graph space to view information about the corresponding graph space metric.

Space graph metrics record the information of different graph spaces separately. Currently, only the Graph service supports a set of space-level metrics.

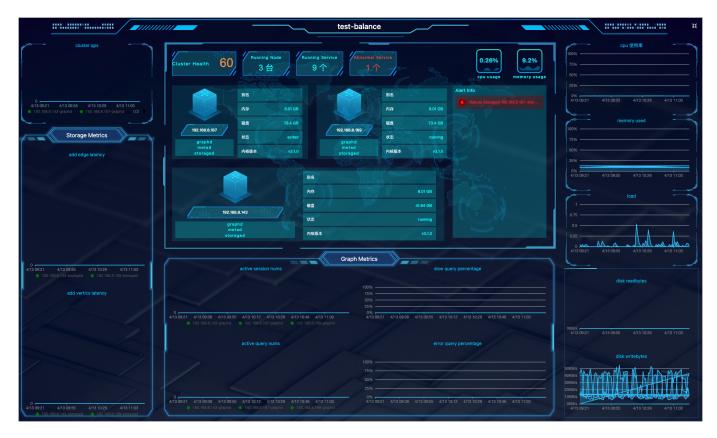
For information about the space graph metrics, see Space graph.



# TV Dashboard

The TV Dashboard feature helps users understand the health status of the cluster and the information of services and nodes at a glance.

Click Monitoring->TV Dashboard to enter the TV Dashboard page.



Screen area	Information displayed
Upper middle area	<ol> <li>The health degree of your cluster. The system scores the health of your cluster. For more information, see the following note.</li> <li>The information and number of running nodes, the number of running services and abnormal services in the cluster.</li> <li>CPU and memory usage of the node at the current time.</li> <li>Alert notifications. The system displays the 5 most recently triggered alert messages based on their severity level (emergency&gt;critical&gt;warning). For more information, Monitoring alerts.</li> </ol>
Lower middle area	Monitoring information of 4 Graph service metrics at different periods. The 4 metrics are: 1. num_active_sessions 2. num_slow_queries 3. num_active_queries 4. num_query_errors
Left side of the area	<ol> <li>QPS (Query Per Second) of your cluster.</li> <li>The monitoring information of 2 Storage service metrics at different periods. The two metrics are: add_edges_latency_us,add_vertices_latency_us.</li> </ol>
Right side of the area	The node-related metrics information at different periods. Metrics include:  1. cpu_utilization  2. memory_utilization  3. load_1m  4. disk_readbytes  5. disk_writebytes

For more information about the monitoring metrics, see Metrics.



Cluster scoring rules are as follows:

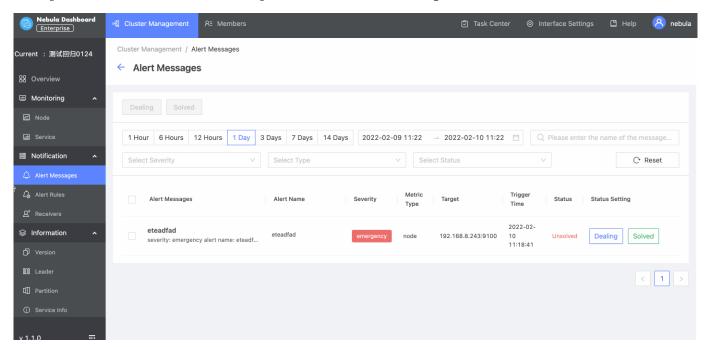
- The maximum score is 100; The minimum score is 13.
- When 100≥Health Degree≥80, the score is blue; When 80>Health Degree≥60, the score is yellow; When Health Degree<60, the score is yellow.
- $\bullet \ Algorithm: (1-number \ of \ abnormal \ services/total \ number \ of \ services)*100\%.$
- Except for the appearance of the first emergency level alert that deducts 40 points, 10 points are deducted for each of the other emergency level alerts and other levels of alerts.

Last update: February 19, 2024

# 15.6.3 Notification

NebulaGraph Dashboard Enterprise Edition notifies on monitoring metrics. You can view alert messages, set alert rules, and set alert receivers.

At the top navigation bar of the Dashboard Enterprise Edition page, click **Cluster Management**, and click **Detail** at the right of the target cluster. **Notification** at the left navigation bar contains **Alert Messages**, **Alert Rules** and **Receivers**.



#### Alert messages

Alert messages will pop up in the upper right corner of the page, you can do the following operations:

- Click the View button to go to the Notification->Alert Messages page to view detailed alert information.
- Click the Mute buttons, the alert rule will not be triggered again for 2 hours.

You can perform the following operations on the **Alert Messages** page:

- You can search for alert messages by message name.
- You can filter alert messages by date and time, and period. Available periods are 1 hour, 6 hours, 12 hours, 1 day, 3 days, 7 days, and 14 days.
- You can filter alert messages by severity, type, and status. Click Reset to empty all filtering results.
- You can set the processing status of alert messages. The status is unsolved by default, and you can set the status to Dealing or Solved.

Alert messages cannot be deleted. In the nebula-dashboard-ent/config/config.yaml file, messageStore sets the number of days to keep alert messages, the value of which is 90 by default. For more information about the configuration file, see <a href="Deploy Dashboard">Deploy Dashboard</a>.

# Alert rules

Before receiving alert messages, you need to set alert rules. Alert rules include custom rules and build-in rules.

#### **CREATE CUSTOM RULES**

Follow the below steps to create a custom rule.

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click **Cluster Management**, and then on the right side of the target cluster, click **Detail**.
- 2. On the left side of the Cluster Management page, click Notification->Alert Rules.
- 3. On the Alert Rules page, click Custom Rules, and then click Create Rule at the top right of the page.
- 4. Set alert rules.
- a. On the Basic Information tab, set alert name, severity, and frequency.

Parameter	Description
Alert Name	Set a name for an alert rule. The name can only contain lowercase letters, numbers, and hyphens (-), and must begin and end with a lowercase letter or number. The name contains up to 253 characters.
Severity	Set a severity level for an alert rule. The severity level includes <code>emergency</code> , <code>critical</code> , and <code>warning</code> .
Alert Frequency	Set how often an alert message is sent. Unit: Minute (Min).

b. On the **Condition** tab, set metric type, rule, and alert duration.

Parameter	Description
Metric Type	Set a metric type. Metric type includes the node metric type and the service type (graphd, storaged, metad).
Metric Rule	Click + Add condition to set metric rules for a node or a service. It supports adding composite conditions (like the usage of AND). For more information, see Monitoring metrics.
Alert duration	Set how long an alert lasts before the alert message is triggered. Unit: Minute (Min).

- c. On the **Rules Overview** tab, check the overall rules.
- d. On the Message Settings tab, you can see the rule summary and rule details, and then click Submit.



DO NOT modify the rule details unless you are clear of the consequences.

#### VIEW CUSTOM RULES

On the Custom Rules, you can do the following operations.

- Search for alert rules and filter alert rules by severity, type, metric, and status.
- $\bullet$  Click  $\boldsymbol{Reset}$  to empty all filtering results.
- Turn on or off the alert rule you set. The status of an alert rule that has been turned on is **active**. The status of an alert rule that has been turned off is **disable**.

EDIT CUSTOM RULES

In the **Custom Rules** list, select the target rule, and then click the edit icon to edit the rule.

DELETE CUSTOM RULES

In the **Custom Rules** list, select the target rule, click the delete icon to delete the rule.

- 660/1069 - 2023 Vesoft Inc.

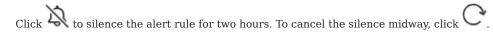
#### **BUILT-IN RULES**

The built-in rules are the default rules in Dashboard Enterprise Edition. You can enable or disable the built-in rules. The status of a built-in rule that has been turned on is **active**. The status of a built-in rule that has been turned off is **disable**.



Built-in rules cannot be edited or deleted.

MUTE ALERT RULES



# Receiver configuration

Alerts can be configured to send notifications to receivers. You can set the email address of the receiver who receives alert notifications. You can also view your Webhook URL and whether the webhook is enabled or not. For more information about the Webhook, see Notification Endpoint.

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, Click **Cluster Management**, and on the right side of the target cluster, click **Detail**.
- 2. On the left-side navigation bar of the Cluster Management page, click Notification->Receivers.
- 3. On the **Receivers** page,
- Click Mail and input the email of the receiver who receives alert notifications and then click Add.
- Click Webhook and see your Webhook URL and whether the webhook is enabled or not.

Last update: February 19, 2024

- 661/1069 - 2023 Vesoft Inc.

# 15.6.4 Information

# Information overview

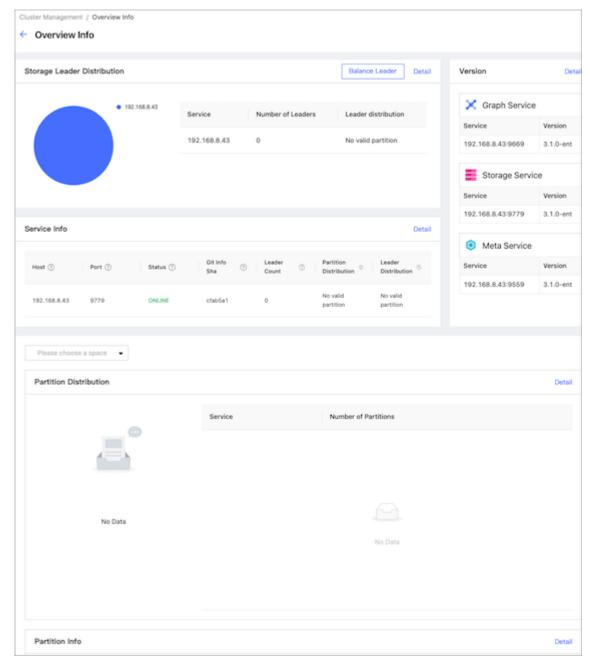
On the **Overview Info** page, you can see the information of the NebulaGraph cluster, including Storage leader distribution, Storage service details, versions and hosts information of each NebulaGraph service, and partition distribution and details.

ENTRY

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click Cluster Management.
- 2. On the right side of the target cluster, click  ${f Detail}.$
- 3. On the left-side navigation bar of the page, click **Information->Overview Info**.



Before viewing the cluster information, you need to select any online Graph service address, enter the account to log in to NebulaGraph (not the Dashboard login account), and the corresponding password.



# STORAGE LEADER DISTRIBUTION

In this section, the number of Leaders and the Leader distribution will be shown.

- Click the **Balance Leader** button in the upper right corner to distribute Leaders evenly and quickly in the NebulaGraph cluster. For details about the Leader, see <u>Storage Service</u>.
- Click **Detail** in the upper right corner to view the details of the Leader distribution.

# VERSION

In this section, the version and host information of each NebulaGraph service will be shown. Click **Detail** in the upper right corner to view the details of the version and host information.

#### SERVICE INFORMATION

In this section, the information on Storage services will be shown. The parameter description is as follows:

Parameter	Description
Host	The IP address of the host.
Port	The port of the host.
Status	The host status.
Git Info Sha	The commit ID of the current version.
Leader Count	The number of Leaders.
Partition Distribution	The distribution of partitions.
Leader Distribution	The distribution of Leaders.

Click **Detail** in the upper right corner to view the details of the Storage service information.

#### PARTITION DISTRIBUTION

Select the specified graph space in the upper left corner, and then you can perform the following operations:

- View the distribution of partitions in the specified graph space. You can see the IP addresses and ports of all Storage services in the cluster, and the number of partitions in each Storage service.
- Click Balance Data to evenly distribute the partitions in the specified graph space.
- Click **Balance Data Remove** to migrate the partitions in the specified Storage service and distribute them evenly to the other Storage services in the cluster. The system will guide you to select the host IP where the specified Storage service is located.

Click **Detail** in the upper right corner to view more details.

#### PARTITION INFORMATION

In this section, the information on partitions will be shown. Before viewing the partition information, you need to select a graph space in the upper left corner. The parameter description is as follows:

Parameter	Description
Partition ID	The ID of the partition.
Leader	The IP address and port of the leader.
Peers	The IP addresses and ports of all the replicas.
Losts	The IP addresses and ports of faulty replicas.

Click **Detail** in the upper right corner to view details. You can also enter the partition ID into the input box in the upper right corner of the details page to filter the shown data.

Last update: February 19, 2024

# Cluster diagnositics

The cluster diagnostics feature in Dashboard Enterprise Edition is to locate and analyze the current cluster problems within a specified time range and summarize the diagnostic results and cluster monitoring information to web-based diagnostic reports.

#### FEATURES

- Diagnostic reports allow you to troubleshoot the current cluster problems within a specified time range.
- · Quickly understand the basic information of the nodes, services, service configurations, and query sessions in the cluster.
- · Based on the diagnostic reports, you can make operation and maintenance recommendations and cluster alerts.

#### ENTRY

- 1. In the top navigation bar of the Dashboard Enterprise Edition page, click Cluster Management.
- 2. On the right side of the target cluster, click **Detail**.
- 3. In the left navigation bar, click Information->Cluster Diagnostics.

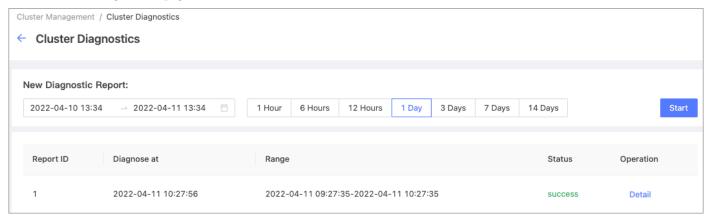
CREATE DIAGNOSTIC REPORTS

1. Select a time range for diagnostics. You can customize the time range or set the range by selecting time intervals, including 1 Hour, 6 Hours, 12 Hours, 1 Day, 3 Days, 7 Days, and 14 Days.



Note that the end time of the diagnostic range you set cannot be longer than the current time. If the end time is longer than the current time, the end time will be set to the current time.

2. On the Cluster Diagnostics page, click Start.



3. Wait for the diagnostic report to be generated. When the diagnostic status is changed to **success** from **generating**, the diagnostic report is ready.

# VIEW DIAGNOSTIC REPORTS

In the diagnostic report list, you can view the diagnostic reports by clicking **Detail** on the right side of the target report. You can also download the diagnostic report in PDF format.

A diagnostic report contains the following information:

- Diagnosis Result
- Basic Info
- Load Info
- Network
- Session
- Service Info
- Configuration Info

# Diagnosis Result

• When the following parameters are abnormal, the corresponding information is displayed in the **Diagnosis Result** section, including the parameter name, type, severity, and details.

Parameter	Description
num_queries_hit_memory_watermark	The total number of nGQL statements that reach the memory high-water mark during execution.
graphd_down	Graph services stopped running.
storaged_down	Storage services stopped running.
metad_down	Meta services stopped running.
node-exporter down	The service used to collect data from the node stopped running.

ullet When no abnormality is diagnosed, **no diagnostic information** is displayed in the diagnostic result.

- 666/1069 - 2023 Vesoft Inc.

#### Basic Info

# **Basic Info**

# Report Time Range

START_TIME	END_TIME
2022-03-30 03:12:29	2022-03-30 03:12:29

# Node Info

ноѕт	INSTANCE	CPU (Core)	MEMORY (GB)	DISK (GB)
192.168.8.129	metad*1 graphd*1 storaged*1	4	8.01	52.43

# Service Info

NAME	ТҮРЕ	ноѕт	PORT	HTTP_PORT	STATUS
192.168.8.129-metad	metad	192.168.8.129	9559	19559	running
192.168.8.129-graphd	graphd	192.168.8.129	9669	19669	running
192.168.8.129-storaged	storaged	192.168.8.129	9779	19779	running

# Leader Distribution

Storage Service	Number of Leaders	Leader Distribution
192.168.8.129:9779	0	

- $\bullet$   $\bf Report\ Time\ Range:$  Displays the time range of the diagnostic report.
- Node Info: Displays the basic information of the node, including the node IP, number of services, CPU, memory, and disk.

Parameter	Description
HOST	The IP address of the node.
INSTANCE	The number of NebulaGraph services deployed on this node. Such as: $metad*1 graphd*1 storaged*1$ .
CPU	The number of CPU cores. Unit: Core.
MEMORY	The memory size of the node. Unit: GB.
DISK	The disk size of the node. Unit: GB.

- Service Info: Displays the type, node IP, HTTP port, and operational status of each NebulaGraph service.
- $\bullet$   $\bf Leader$   $\bf Distribution:$  Displays the distribution of Leaders in Storage services.

Parameter	Description
Storage Service	Displays the access addresses for Storage services.
Number of Leaders	Displays the number of Leaders in the corresponding Storage service.
Leader Distribution	Displays the number of Leader distributions for different space graphs in the corresponding Storage service.

#### Load Info

# **Load Info CPU** Utilization 192.168.8.129:9100 0.30% 0.30% 0.30% Memory Utilization 192.168.8.129:9100 11.62% 11.62% 11.62% Disk Utilization **192.168.8.129:9100** 67.56% 67.56% 67.56% |-- /dev/sda1 67.56% 67.56% 67.56%

Displays the load information of the node, including the average value (AVG), maximum value (MAX), minimum value (MIN) of the following metrics of the node within the time range:

- Memory Utilization: Displays the node memory usage in %.
- CPU Utilization: Displays the node CPU usage in %.
- Disk Utilization: Displays the total disk utilization of the node and the utilization of each disk in the node in %.

Network

Network			
NetworkOut			
Node	AVG (Bytes/s)	MAX (Bytes/s)	MIN (Bytes/s)
▼ 192.168.8.129:9100	11882.18	11882.18	11882.18
eth0	11882.18	11882.18	11882.18
NetworkIN			
Node	AVG (Bytes/s)	MAX (Bytes/s)	MIN (Bytes/s)
▼ 192.168.8.129:9100	8208.09	8208.09	8208.09
eth0	8208.09	8208.09	8208.09

Displays the network traffic information of all nodes in the cluster, including the average (AVG), maximum (MAX), and minimum (MIN) values of the following metrics:

- **NetworkOut**: Displays the magnitude of network outflow speed for each node in the cluster, and the magnitude of outflow speed for each NIC in each node. Unit: Bytes/s.
- **NetworkIn**: Shows the magnitude of network inflow speed for each server node in the cluster and the magnitude of inflow speed for each NIC in each node. Unit: Bytes/s.

# Session

# Sessions Total num\_opened\_sessions num\_auth\_failed\_sessions num\_active\_sessions num\_reclaimed\_expired\_sessions

Displays the session-related information for all Graph services in the cluster.

Parameter	Description
num_opened_sessions	The number of sessions connected to the server.
num_auth_failed_sessions	The number of sessions in which login authentication failed.
num_active_sessions	The number of changes in active sessions.
num_reclaimed_expired_sessions	The number of expired sessions actively reclaimed by the server.

Service Info

Displays metrics related to the stability of each service in the cluster.

# · Graph:

#### Graph

This table shows metrics related to the stability of the Graph service. TOTAL is the total count of the metric; P75 is the 75th percentile latency; P95 is the 95th percentile latency; P99 is he 99th percentile latency.

METRIC_NAME	TOTAL	ERROR	P75	P95	P99	P999
slow_queries	0		-2147.483648s	-2147.483648s	-2147.483648s	-2147.483648s
num_killed_queries	0					
num_queries_hit_memory_watermark	0					
num_rpc_sent_to_metad	0	0				
query	0	0	-2147.483648s	-2147.483648s	-2147.483648s	-2147.483648s

# Parameter Description

METRIC\_NAME query: The number of all queries.

slow\_queries : The number of slow queries.
num\_killed\_queries : The number of killed queries.

 $\verb|num_queries_hit_memory_watermark|: The total number of nGQL statements that reach the memory high-water and the statement of nGQL statements that reach the memory high-water and the statement of nGQL statement of nGQL statements that reach the memory high-water and number of nGQL statements that reach the memory high-water and number of nGQL statements that reach the memory high-water and number of nGQL statements that reach the memory high-water and number of nGQL statements that reach the memory high-water and number of nGQL statements that reach the memory high-water and number of nGQL statements that reach the memory high-water and number of nGQL statements that reach the memory high-water and number of nGQL statements that reach the memory high-water and number of nGQL statements that reach the memory high-water and number of nGQL statements that reach the number of nGQL statements that number of nGQL statements the number of nGQL statements that number of nGQL statements the number of nGQL statements$ 

mark during execution.

num\_rpc\_sent\_to\_metad: The number of RPC requests that the Graphd service sent to the Metad service.

# • Meta:

# Meta

This table shows metrics related to the stability of the Meta service. TOTAL is the total count of the metric; P75 is the 75th percentile latency; P95 is the 95th percentile latency; P99 is the 99.9th percentile latency.

METRIC_NAME	TOTAL	P75	P95	P99
heartbeat	0	-2147.483648s	-2147.483648s	-2147.483648s

# Parameter Description

 ${\tt METRIC\_NAME} \qquad \qquad {\tt heartbeat}: The \ number \ of \ heartbeats.$ 

# • Storage:

# Storage

This table shows metrics related to the stability of the Storage service. TOTAL is the total count of the metric; P75 is the 75th percentile latency; P95 is the 95th percentile latency; P99 is he 99th percentile latency; P999 is the 99.9th percentile latency.

METRIC_NAME	TOTAL	ERROR	P75	P95	P99
delete_edges	0	0	-2147.483648s	-2147.483648s	-2147.483648s
delete_tags	0	0	-2147.483648s	-2147.483648s	-2147.483648s
num_rpc_sent_to_metad	0	0			
delete_vertices	0	0	-2147.483648s	-2147.483648s	-2147.483648s

# Parameter Description

METRIC\_NAME delete\_vertices: The number of deleted vertices.

delete\_edges : The number of deleted edges.
delete\_tags : The number of deleted tags.

num\_rpc\_sent\_to\_metad: The number of RPC requests that the Storaged service sent to the Metad service.

The descriptions of other parameters are as follows:

Parameter	Description	
TOTAL	The total number of times this monitoring metric is executed.	
ERROR	The number of errors that occurred.	
P75	The 75th percentile latency.	
P95	The 95th percentile latency.	
P99	The 99th percentile latency.	
P999	The 99.9th percentile latency.	

# Configuration Info

 $Lists\ all\ configuration\ information\ for\ Graph,\ Meta,\ and\ Storage\ services\ in\ the\ current\ cluster.$ 

For information about the configurations of each service in NebulaGraph, see Configurations.

Last update: February 19, 2024

- 671/1069 - 2023 Vesoft Inc.

#### Job management

Users can manage the jobs in a specified graph space through the Dashboard, including viewing, stopping, and recovering jobs, and supports viewing the details of a single job.



How to run jobs, see Job manager and the JOB statements.

#### PREREQUISITES

- The job management feature is available in NebulaGraph Enterprise Edition 3.4.0 and above versions.
- The job management feature is available in NebulaGraph Community Edition 3.3.0 and above versions.

ENTRY

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click Cluster Management.
- 2. On the right side of the target cluster, click **Detail**.
- 3. On the left-side navigation bar of the page, click Information->Job Management.
- 4. Select any online Graph service address, enter the account to log in to NebulaGraph (not the Dashboard login account), and the corresponding password.
- 5. Select the target graph space at the upper left corner of the page.

VIEW JOB

After you select the graph space, the page will display all the job information that has not expired by default. You can quickly find jobs through the filter box at the top of the page as follows:

- Select a job status for filtering. The status includes QUEUE, RUNNING, FINISHED, FAILED, STOPPED, and SUCCEEDED. For the status description, see Job manager and the JOB statements.
- Select a time range for filtering. You can view the job information of the maximum of 7 days by default. You can also select a time range or quickly select latest 12 hours, 1 day, 3 days, or 7 days.
- Select a Job ID or Command for filtering and enter what you want to search for.
- By default, the job information page will not be updated automatically. You can set the update frequency of the job information page globally or click the button to update the page manually.
- Click Detail in the Operation column on the right side of the target job to view more information, including Task ID , Host , Error Code , etc.

STOP JOB

Click Stop Job in the Operation column on the right side of the target job to stop an unfinished job. After clicking, the status of the job becomes STOPPED.

RECOVER JOB

Click Recover Job in the Operation column on the right side of the target job to recover the job whose status is FAILED or STOPPED. After clicking, the status of the job becomes RUNNING.



- If there are multiple BALANCE DATA jobs in STOPPED status, only the latest one can be recovered.
- The completed job can not be recovered.

Last update: February 19, 2024

# Audit log

The NebulaGraph audit logs store and categorize all operations performed on the Graph service. Dashboard Enterprise Edition allows you to quickly view audit logs.



Only when the cluster you created or imported is the Enterprise Edition, this feature is available.

ENTRY

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click Cluster Management.
- 2. On the right side of the target cluster, click **Detail**.
- 3. On the left-side navigation bar of the page, click Information->Audit Log.



- To use the audit log for the first time, you need to jump to the **Config Management** page as prompts to enable the audit log and restart the graph service.
- For the description of audit log parameters, see Configure audit logs.

#### VIEW AUDIT LOG

In the upper corner of the page, you can filter services or search for the log name. Click View Log in the Operation column.

- Support copying all logs in the window with one click.
- Support copying the log file path.
- ullet Support **Tail Mode** and **Range Mode** to view logs. You need to click **Refresh** after setting.
- Support searching logs by keywords (at least 3 characters).

Last update: February 19, 2024

- 673/1069 - 2023 Vesoft Inc.

# **Runtime log**

DBAs and developers can use runtime logs to investigate and identify issues when the system malfunctions. Dashboard Enterprise Edition allows you to quickly view runtime logs.

ENTRY

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click **Cluster Management**.
- 2. On the right side of the target cluster, click **Detail**.
- 3. On the left-side navigation bar of the page, click **Information->Runtime Log**.



For the description of runtime log parameters, see Runtime log.

VIEW RUNTIME LOG

In the upper corner of the page, you can filter services or search for the log name. Click **View Log** in the **Operation** column.

- Support copying all logs in the window with one click.
- Support copying the log file path.
- Support Tail Mode and Range Mode to view logs. You need to click Refresh after setting.
- Support searching logs by keywords (at least 3 characters).

Last update: February 19, 2024

- 674/1069 - 2023 Vesoft Inc.

# 15.6.5 Operation

#### Node

On this page, the information of all nodes will be shown, including the cluster name, Host(SSH\_User), CPU (Core), etc. Users can add nodes, view node monitoring, and manage services on the node.

ENTRY

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click Cluster Management.
- 2. On the right side of the target cluster, click **Detail**.
- 3. On the left-side navigation bar of the page, click **Operation->Node**.

ADD NODE

Click **Add Node** and enter the following information, the Host, SSH port, SSH user, authentication type, NebulaGraph package, etc., and click **OK**.

The authentication type is described as follows:

- SSH Password: Enter the password of the SSH user.
- SSH Key: Click **Upload** and select the private key file of the node. You need to generate the secret key files on the node to be added and send the private key file to the current computer (not the machine where Dashboard is deployed). If the passphrase is set, this parameter is also required.



After a node is added, data is not automatically imbalanced. You need to select the target graph space on the Overview Info page and then perform the Balance Data and Balance Leader operations.

OTHER NODE OPERATIONS



Click th

button to view the process name, service type, status, and runtime directory of the corresponding node.

- Click **Node Monitoring** to jump to the detailed node monitoring page. For more information, see Cluster monitoring.
- Click **Service Management** to jump to the service management page.
- Click **Edit Node** to modify the node settings.
- If a node has no service, you can **Delete Node**. For details about how to delete a service, see section **Scale** below.

Last update: February 19, 2024

- 675/1069 - 2023 Vesoft Inc.

#### Scale

On **Scale** page, you can **add node** and **import node in batches** quickly, and add **Graph services** and **Storage services** to the existing nodes.



Only when the cluster you created or imported is the Enterprise Edition, this feature is available.

ENTRY

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click Cluster Management.
- 2. On the right side of the target cluster, click **Detail**.
- 3. On the left-side navigation bar of the page, click **Operation->Scale**.

STEPS

Add node

See Node.



After a node is added, data is not automatically imbalanced. You need to select the target graph space on the Overview Info page and then perform the Balance Data and Balance Leader operations.

Batch import of node

Download and fill in the CSV template file, then upload the file and select the installation package. Click  $\mathbf{OK}$  to import nodes in batches.

Modify services

- 1. Select the nodes in the node list. Click the service to be added in the upper right corner of the list, or click  $\mathbf{X}$  on the label of the service to be deleted in the **Service type** column in the list.
- 2. Confirm the modified service in the **Service** display area below. You can modify the port, HTTP port, and HTTP2 port of the newly added service.



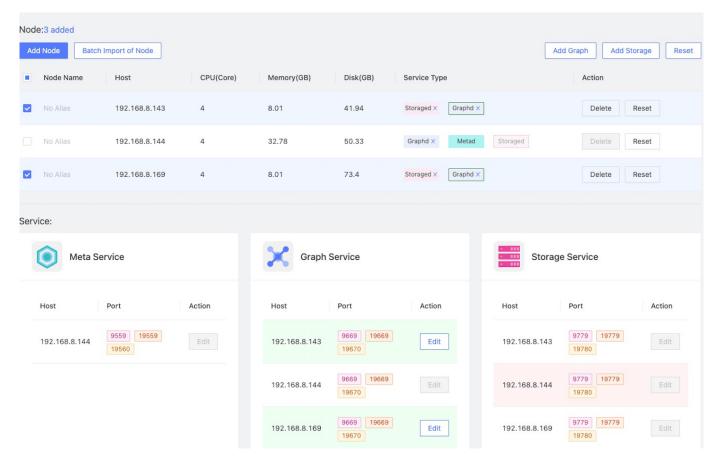
Green indicates services that will be added soon, and red indicates services that will be removed.

3. Click **OK** at the bottom of the page.



- Currently, you can dynamically scale Storaged and Graphd services through Dashboard. The Metad service cannot be scaled. When scaling a cluster, it is recommended to back up data in advance so that data can be rolled back when scaling fails. For more information, see FAQ.
- Make sure that services of the same type are not deployed on the same node, and that at least one of each type of service is deployed in the cluster.
- Before removing the storage service, you must migrate the data stored on the node. You need to perform the Balance Data Remove
  operation on the Overview Info page.

In this example, storage services with nodes 192.168.8.143 and 192.168.8.167 are added, and Graph services with node 192.168.8.169 are deleted. If the box is dotted and the service name is greyed, it means the service is removed. If the box is solid, it means the service is newly added.



Reset

Click the **Reset** button to cancel all uncommitted operations and restore to the initial state.

Last update: February 19, 2024

#### Service

On **Service** page, you can view the host, path, and status of the services, and start, stop, kill, or restart the services. In addition, you can easily and quickly view the contents of the log file.

ENTRY

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click Cluster Management.
- 2. On the right side of the target cluster, click **Detail**.
- 3. On the left-side navigation bar of the page, click **Operation->Service**.

STEPS



If you click **Stop/Restart**, the running task will be stopped instantly, which may cause data inconsistency. It is recommended to perform this operation during the low peak period of the business.

- Locate the target service and perform the related operation in the **Operation** column.
- Select multiple services and perform batch operations at the upper corner of the page.
- Click the icon to quickly view the service monitoring information.
- When synchronizing data, you can view and manage related services on the **Dependency** page. For details about data synchronization, see Synchronize between two clusters.

Last update: February 19, 2024

- 678/1069 - 2023 Vesoft Inc.

# Update config

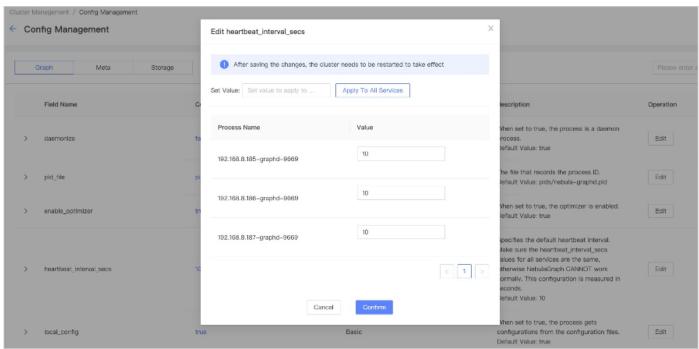
On Update Config page, you can view and modify the service configuration files.

ENTRY

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click Cluster Management.
- 2. On the right side of the target cluster, click **Detail**.
- 3. On the left-side navigation bar of the page, click **Operation->Update Config**.

STEPS

- 1. Select the type of service whose configuration you want to modify.
- 2. Locate the configuration to be modified and click **Edit** in the **Operation** column.
- 3. In the pop-up dialog box, you can modify the **Value** individually. They can also be modified uniformly at the top, and you need to click **Apply To All Services** after modification.



4. Click **Confirm** after the modification is complete.



You need to restart the corresponding service in the Service page after the configuration modification. For details, see Service.

Last update: February 19, 2024

- 679/1069 - 2023 Vesoft Inc.

#### Member management

**Member Management** page shows only the cluster creator account (owner role) by default. The account with the owner role can add and delete the cluster administrator (operator role).

ENTRY

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click Cluster Management.
- 2. On the right side of the target cluster, click **Detail**.
- 3. On the left-side navigation bar of the page, click **Operation->Member Management**.

STEPS

• Add the cluster administrator: Click the search box at the top left. Select the target account that you want to add to be the administrator of the cluster in the drop-down list, and then click **Add**.



The accounts of cluster members must be included in Dashboard accounts. For information about how to create an account, see Authority management.

- Delete the cluster administrator: Click in the operation column on the right of the cluster administrator account, and then click **Confirm**.
- Transfer the owner role: Click **Transfer** in the operation column on the right of the owner role. Select the target account that you want to be transferred, and then click **Confirm**.

Last update: February 19, 2024

- 680/1069 - 2023 Vesoft Inc.

#### Version upgrade

NebulaGraph Dashboard Enterprise Edition supports upgrading the version of the existing NebulaGraph cluster.



- During the upgrade, the cluster will replace binary files. The upgrade speed is fast, but the cluster will still be stopped and restarted.
- · Automatic rollback is not supported. Users can manually upgrade the cluster again when the upgrade failed.
- The upgrade cannot be stopped or canceled.



- Only supports upgrading the NebulaGraph cluster that version greater than **3.0.0** to the version equal to or lower than **3.2.1**. To upgrade to **3.3.0**, see manual upgrade.
- Do not support upgrading clusters across the major version.
- The community edition can be upgraded to the enterprise edition by uploading and verifying licenses, and the enterprise edition can be upgraded to the community edition.
- The cluster can be upgraded to a minor version in the current major version, including a smaller version than the current minor version.
- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click Cluster Management.
- 2. On the right side of the target cluster, click **Detail**.
- 3. On the left-side navigation bar of the page, click **Operation->Version Upgrade**.
- 4. On the Version Upgrade page, confirm Current NebulaGraph version and select the upgrade version.



If you do not find the suitable version, click **Package Management** to download or upload the required version installation package. For details, see <u>Package management</u>.

1. Click **Upload License** to upload the NebulaGraph Dashboard license (skip this step if upgrading a NebulaGraph cluster of the Community Edition).



This step is for NebulaGraph Enterprise Edition clusters.

1. Click Next to perform the upgrade check, and then click Next.

The cluster will be shut down during the upgrade and automatically restart the services after the upgrade. You can use the **diagnostics report** to help you judge whether the timing to upgrade is suitable.

2. Confirm the upgrade information again, including **Cluster Name**, **Current NebulaGraph Version**, and **Upgrade NebulaGraph Version**, and then click **Upgrade**. Users can view the upgrade task information in task center, the task type is version update.

Last update: February 19, 2024

#### Back up and restore NebulaGraph data

To prevent data loss due to operational errors or system failures, NebulaGraph offers the Backup & Restore (BR) tool to help users back up and restore graph data. Dashboard Enterprise Edition integrates BR capabilities and offers simple UIs that allow users to perform data backup and restore operations in just a few steps. This document describes how to use Dashboard Enterprise Edition to backup and restore NebulaGraph data.

#### LIMITS

• Currently, Dashboard only supports backup data to cloud storage services compatible with the S3 protocol (e.g. OSS, MinIO, Ceph RGW, etc.) and does not support local backups.



To back up data to a local device, see What is Backup & Restore.

- Backup and restoration of space-level data are not supported.
- Backup data can only be restored to the original cluster, and cannot be restored across clusters.
- Breakpoint moving of backup and restore data is not supported.
- Currently, only the logs generated by backup and restore operations are supported.

#### PREREQUISITES

- A cluster is created with Dashboard.
- A cloud storage service that is compatible with the S3 protocol is activated and a storage bucket is created. For details, see the documentation for the corresponding cloud storage service.

STEPS

Entry

- 1. In the top navigation bar, click  ${\bf Cluster\ Management}.$
- 2. On the right side of the target cluster, click **Detail**.
- 3. In the left navigation bar, click **Operation->Backup&Restore**.

- 682/1069 - 2023 Vesoft Inc.

Full backup

Data is backed up to the cloud storage service by creating a backup file as follows.

- 1. On the  ${\bf Backup\&Restore}$  page, click the  ${\bf Backup\ List}$  tab.
- 2. In the upper right corner of the page, click S3 Service Settings.
- 3. Fill in the configuration information for the corresponding cloud storage service and click **Submit**.

Parameter	Description
s3.access_key	The Access Key ID that is used to identify a user. For example, ${\tt AKIAI44QH8DHBxxxxx}$ .
s3.endpoint	The domain URL of the entry point for the cloud storage service. For example, $https://s3.us-east-2.amazonaws.com$ . The URL containing $bucket_name$ is not supported, such as $https://\{bucket_name\}.s3.us-west-2.amazonaws.com$ .
s3.region	The physical location of a data center. For example, us-east-1.
s3.secret_key	The Access Key Secret that is used to verify the identity of the user. For example, $je7MtGbClwBF/2Zp9Utk/h3yCoxxxx$ .
storage path	The data storage path which $\mathbf{must}$ $\mathbf{start}$ $\mathbf{with}$ $\mathbf{s3}$ . For example, $\mathbf{s3}$ ://br-test/backup/ .

The following configurations are examples for Alibaba Cloud Object Storage Service and Amazon S3:

• For Amazon S3:

* s3.access_key:	LTAI5tEwhrem
* s3.endpoint:	https://s3.us-west-2.amazonaws.com/
* s3.region:	us-west-2
* s3.secret_key:	MfjNFKNf56Y
* storage path:	s3://nebula-br-test/

• For Alibaba Cloud Object Storage Service:





To back up data to OSS, you need to replace oss with s3 for the OSS storage path. For example, change the original OSS path oss:// nebula-br-test/ to s3://nebula-br-test/.

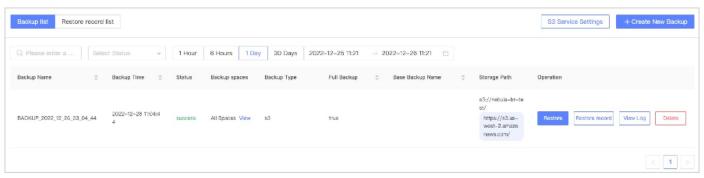
- 684/1069 - 2023 Vesoft Inc.

- 4. In the upper right corner of the page, click Create New Backup.
- 5. On the Create New Backup page, choose Full backup.
- 6. Click **Environment check** to check whether the relevant configurations are working properly, and then click **Submit**. Environment check includes:
- Your NebulaGraph cluster is running.
- The access key to log onto the storage service has not expired.
- The status of business traffic. It only checks if the QPS of your business is 0. When QPS is not 0, you are prompted to back up data during off-peak hours.



You are unable to submit the backup when your cluster works abnormally or the access key to the storage service has expired.

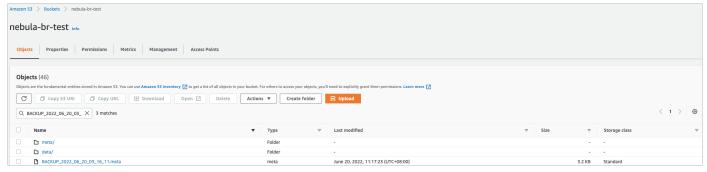
7. View the created backup file in the backup list.



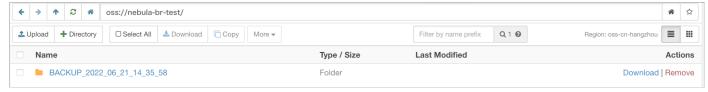
Q Note

You are unable to perform a new backup until the previous backup is completed.

- 8. Check if the created backup file exists in the storage service. Successfully created backup files are stored to the storage path set above, like s3://nebula-br-test.
- Amazon S3:



• Alibaba Cloud Object Storage Service:



# Sanger

Do not modify the file name and storage path of backup files, otherwise, the backup data cannot be restored to the cluster.

Incremental backup

Users can perform incremental backup based on existing backup files. Incremental backup only covers all files that have changed or been modified since the last backup was made.

- 1. On the Backup&Restore page, click the Backup List tab.
- 2. In the upper right corner of the page, click Create New Backup.
- 3. On the Create New Backup page, choose Incremental backup.
- 4. Choose Base Backup Name.



New data will be backed up based on this base backup. Make sure that the data of the base backup is not changed. Otherwise, the incremental backup may fail.

5. Click Environment check to check whether the relevant configurations are working properly, and then click Submit.

Environment check includes:

- Your NebulaGraph cluster is running.
- The access key to log onto the storage service has not expired.
- The status of business traffic. It only checks if the QPS of your business is 0. When QPS is not 0, you are prompted to back up data during off-peak hours.



You are unable to submit the backup when your cluster works abnormally or the access key to the storage service has expired.

Restore data

You can restore the backed-up data stored in the cloud storage service to the original cluster.

# Caution

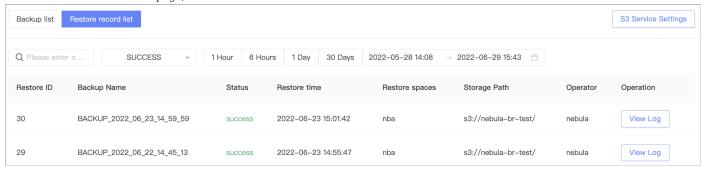
- Before restoring the data, please make sure that the name and storage path of the backup file stored in the cloud storage service are not changed, otherwise, the data restoration will fail.
- During the data restoration process, all data in the cluster is removed and replaced with the data in the backup file.
- The restoration process is executed offline, and you cannot perform other operations during the data restoration process.

Follow the steps below to restore data.

- 1. On the **Backup&Restore** page, click the **Backup list** tab.
- 2. To the right of the target backup file, click **Restore**.
- 3. Click Environment check, and when the environment check is passed, click Submit.

Environment check includes:

- · Your NebulaGraph cluster is running.
- The access key to log onto the storage service has not expired.
- No business website traffic.
- 4. On the Restore record list page, view restoration records.



- Restoration records cannot be deleted.
- The list page displays restoration records created within 30 days.
- The list page displays the restoration ID, backup file name, status, time, graph space, storage path, operator, and the log generated by the restoration operation.
- $\bullet$  The restoration status includes  $\mbox{ running}\,,\,\mbox{ success}\,,\,\mbox{and failed}\,.$



You're unable to restore the backup data until the previous restoration is complete.

• You can filter restoration records by creation time and status, or search backup file names for restoration records.

View backup/restore progress

After the backup or restoration has started, you can click **View Log** in the **Operation** column of the corresponding task to view the progress.

Last update: February 19, 2024

# 15.6.6 Operation record

This topic shows how to use the operation record feature in NebulaGraph Dashboard.

At the top navigation bar of the Dashboard Enterprise Edition page, click **Cluster Management**, and click **Detail** at the right of the target cluster, and on the left-side navigation bar, click **Operation Record** to enter the operation history page.

On the **Operation record** page, you can check the operation records of the latest 1 hour, 6 hours, 1 day, 3 days, 7days, or 14 days. You can also view who runs what operation on which cluster at what time.

Last update: February 19, 2024

# 15.6.7 Other settings

The following shows other settings in NebulaGraph Dashboard.

At the top navigation bar of the Dashboard Enterprise Edition page, click **Cluster Management**, click **Details** on the right side of the target cluster, and on the left-side navigation bar, click **Other Settings** to enter the other settings page.

- Information: shows the cluster name, the creation time, the creator, and the owner of the current cluster.
- Unbind: Unbind a cluster and remove its information from the platform. The unbound cluster info will be removed and no operations will be done on cluster services or NebulaGraph data.



To unbind a cluster, enter the cluster name first.

• Delete: Delete a cluster and remove its information from the platform. Deleting the cluster will stop its service and unbind the cluster info, but retain its NebulaGraph data. Be cautious when you delete a cluster.



To delete a cluster, enter the cluster name first

Last update: February 19, 2024

- 689/1069 - 2023 Vesoft Inc.

# 15.7 Authority management

You can log into NebulaGraph Dashboard Enterprise Edition with different types of accounts. Different accounts have different permissions. This article introduces account types, roles, and permissions.



You need to configure the related protocols before using LDAP accounts or OAuth2.0 accounts. For details, see Single sign-on.

# 15.7.1 Account types

Once you log into Dashboard Enterprise Edition using the initialized account name  $\[$ nebula  $\]$ nebula and password  $\[$ nebula  $\]$ nebula  $\[$ nebula  $\]$ nebula and password  $\[$ nebula  $\]$ nebula  $\[$ nebula  $\]$ nebula  $\[$ nebula  $\]$ nebula  $\]$ nebula  $\[$ nebula  $\]$ nebula  $\[$ nebula  $\]$ nebula  $\]$ nebula  $\[$ nebula  $\]$ nebula  $\]$ nebula  $\[$ nebula  $\]$ nebul

#### LDAP accounts

Dashboard Enterprise Edition enables you to log into it with your enterprise account by accessing LDAP (Lightweight Directory Access Protocol).

#### OAuth2.0 accounts



The feature is still in beta. It will continue to be optimized.

Dashboard Enterprise Edition enables you to use access\_token to authorize the third-party applications to access the protected information based on OAuth2.0.

#### General accounts

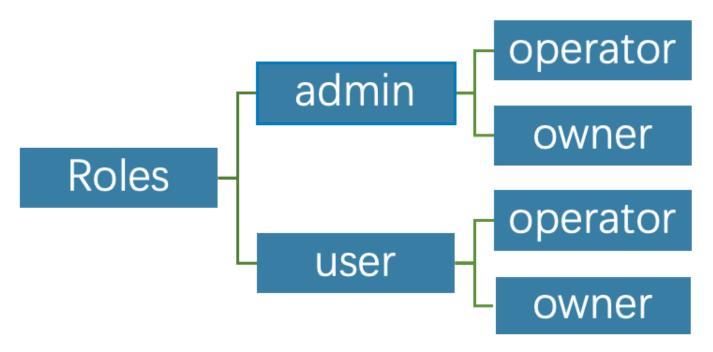
Dashboard Enterprise Edition enables you to create local accounts.

# 15.7.2 Account roles

You can set different roles for your accounts. Roles are different in permissions. There are two types of account roles in Dashboard Enterprise Edition: system roles ( admin and user ) and cluster roles ( owner and operator ).

The relationship between system roles and cluster roles and their descriptions are as follows.

- 690/1069 - 2023 Vesoft Inc.



#### **System roles:**

Roles	Permission	Description
admin	<ol> <li>Create accounts.</li> <li>Modify the role of an existing account.</li> <li>Perform platform settings, system-level alert settings.</li> <li>Delete accounts.</li> </ol>	<ol> <li>There can be multiple admin roles, i.e. system administrators.</li> <li>An admin is the operator of all clusters by default, i.e. an admin can manage all clusters.</li> <li>An admin can assign a user to be the operator of a cluster.</li> <li>Displayed in the cluster member list by default. An owner cannot remove an admin unless the admin is converted to user, and the system will automatically remove the admin from the cluster member list.</li> </ol>
user	<ol> <li>Has read-only permissions for the system dimension.</li> <li>After an admin creates a new account with the user role, the user account cannot view any clusters if the corresponding cluster is not assigned to the account.</li> <li>Can create clusters and become the owner of the clusters.</li> </ol>	<ol> <li>General role.</li> <li>There can be multiple user roles.</li> </ol>

# Cluster roles:

Roles	Permission	Description
operator	1. Scale clusters.	1. The cluster operator.
	2. Set cluster alerts.	2. There can be multiple operator roles in a cluster.
	3. Manage cluster nodes.	
	4. Manage cluster services.	
owner	1. Have all the permissions of operator.	1. The cluster owner.
	2. Unbind and delete clusters.	2. There can only be one owner in a cluster.
	3. Add and remove accounts with operator roles.	
	4. Transfer the owner role.	

- 691/1069 - 2023 Vesoft Inc.

#### 15.7.3 Create accounts

Accounts with admin roles can create other accounts. The steps are as follows:

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click Authority, and click Create.
- 2. Select one method and input information to create an account, and click OK.
- Invite (LDAP or OAuth2.0 accounts): Set the invitee's account type, enterprise email and role. After the invitee clicks the **Accept** button in the email to activate the account, the invitee needs to click **Login** to automatically jump to the Dashboard Enterprise Edition login page. The invitee can log into Dashboard with his/her enterprise email account and password.



Automatic registration is also supported after LDAP is enabled. When you enter an unregistered account in LDAP mode on the login page, the Dashboard automatically registers the account, but the role permission is user.

Create Account (general accounts): Set the login name, password, and role for the new account. For information about roles, see
the above content.

#### 15.7.4 View accounts

The created accounts are displayed on the Authority page.

- You can view the username, account type, role, associated cluster, and create time of accounts.
- Account Type: Includes ldap, oauth2.0 and platform. platform is a general account.
- Role: Displays the role of an account, including admin and user. For more information about roles, see the above content.
- **Associated Clusters**: Displays all the clusters that can be operated by an account. If the cluster was created by the account, the associated cluster has the owner tag.
- You can search for accounts in the search box, and filter accounts by selecting an associated cluster.

### 15.7.5 Other operations

- In the **Action** column on the **Authority** page, click lacksquare to edit account information.
- In the **Action** column on the **Authority** page, click to delete an account.

Last update: February 19, 2024

- 692/1069 - 2023 Vesoft Inc.

### 15.8 Task Center

NebulaGraph Dashboard Enterprise Edition allows you to view the progress of running tasks and the information of ended tasks.

#### 15.8.1 Precautions

- The running tasks can not be canceled.
- The historical tasks cannot be deleted.

### 15.8.2 Task types

Currently the task center supports the following types of tasks:

- install: Create clusters.
- scale: Scale clusters.
- version update: Update NebulaGraph versions.
- package upload: Upload NebulaGraph installation packages.
- package download: Download NebulaGraph installation packages.
- package deploy: Deploy the installation package when adding a node.

### 15.8.3 Entry

At the top navigation bar of the Dashboard Enterprise Edition page, click Task Center to view task information.

### 15.8.4 Running tasks

Click the tab **Running Task** to view the progress of the running tasks.

- Click a task name to view the ID, node name, type, create time, and operator of the running task.
- Clink Task information to view task details.

### 15.8.5 Task history

Click Task History to view all ended tasks.

- You can filter historical tasks by status, type, date, and time.
- On the right side of the target historical task, click **Task information** to view task details, and click **Logs** to view task execution logs.

Last update: February 19, 2024

- 693/1069 - 2023 Vesoft Inc.

# 15.9 System settings

### 15.9.1 System settings

On the **System Settings** page, you can modify the page title, logo image, and cover image, and also switch language or help prompts.

### Entry

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click **System Settings**.
- 2. On the left-side navigation bar of the page, click System Settings.

### Steps

- After modifying the page title, logo image, and cover image, click Save.
- Change the display language. Currently, only Chinese and English are supported.
- Turn on or off help tips. An example of tips is as follows.



Last update: February 19, 2024

# 15.9.2 Notification endpoint

On Notification Endpoint page, you can set the notification mail and webhook.

### Entry

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click System Settings.
- 2. On the left-side navigation bar of the page, click Notification Endpoint.

### Steps

MAIL

Dashboard supports sending and receiving alert messages for all clusters via E-mail.

• You need to set the following parameters to send alert messages.

Parameter	Description
SMTP Server Address	The SMTP server address corresponding to your mail box.
Port	The port number of the SMTP server corresponding to your mailbox. $\label{eq:smtp}$
Use SSL	Check the box to enable SSL for encrypted datatransmission.
SMTP User Name	The SMTP server account name.
SMTP Password	The SMTP server password.
Sender Email	The email address of the one who sent you the email.

 $\bullet$  You need to set a receiver to receive alert messages.

Parameter	Description
Receiver	Set the email address to receive alert messages. This email address will receive alert messages for all clusters created on Dashboard.

#### WEBHOOK

Dashboard supports configuring Webhook to bring all cluster alert messages into third-party projects.

On the left-side navigation bar of the **Interface Settings** page, click **Notification Endpoints->Webhook** to input the **Webhook URL** used to receive alert messages. You can turn on or off the Webhook feature at the top right of the page.

Last update: February 19, 2024

- 695/1069 - 2023 Vesoft Inc.

### 15.9.3 Single sign-on

NebulaGraph Dashboard Enterprise Edition supports general accounts, LDAP accounts, and OAuth2.0 accounts. This article introduces how to configure the protocols of LDAP and OAuth2.0.



- · After the configuration is complete, you can create the account and activate the invitation. For details, see Authority management.
- You can quickly switch on or off LDAP or OAuth2.0 in the left navigation bar.

#### LDAP configuration

**ENTRY** 

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click System Settings.
- 2. On the left-side navigation bar of the page, click Single Sign-on->LDAP.

CONFIGURATION DESCRIPTION

Parameter	Example	Description
LDAP Server Address	ldap://192.168.10.100	The LDAP server address.
Bind DN	cn=admin,dc=vesoft,dc=com	The LDAP login username.
Password	123456	The LDAP login password.
Base DN	dc=vesoft,dc=com	Set the path to query user data.
User Filter	&(objectClass=*)	Set a filter to LDAP search queries.
Email Key	mail	Set the field name used to restore email in LDAP.

#### INSTRUCTION

After LDAP is enabled, you can register an LDAP account in two ways:

- Email invitation: When creating an account on the **Members** page, you can invite others to register by email. The advantage is that you can set the role permissions of the account.
- Automatic registration: When you enter an unregistered account in LDAP mode on the login page, the Dashboard automatically registers the account, but the <u>role permission</u> is <u>user</u>.

### OAuth2.0 configuration



The feature is still in beta. It will continue to be optimized.

ENTRY

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click System Settings.
- 2. On the left-side navigation bar of the page, click **Single Sign-on->OAuth2.0**.

- 696/1069 - 2023 Vesoft Inc.

### CONFIGURATION DESCRIPTION

Parameter	Example	Description
ClientID	4953xxx- mmnoge13xx.apps.googleusercontent.com	The application's ClientId.
ClientSecret	GOCxxx-xaytomFexxx	The application's ClientSecret.
RedirectURL	http://dashboard.vesoft-inc.com/login	The URL that redirects to Dashboard.
AuthURL	https://accounts.google.com/o/oauth2/auth	The URL used for authentication.
TokenURL	https://oauth2.googleapis.com/token	The URL used to get the access_token.
UserInfoURL	https://www.googleapis.com/oauth2/v1/ userinfo	The URL used to get the user information.
Username Key	email	The key of user name.
Organization	vesoft company	The organization name.
Requested scopes for OAuth	email	Scope of OAuth permissions. The scope of permissions needs to be a subset of the scope configured by the vendor's OAuth2.0 platform, otherwise, the request will fail. Make sure the Username Key is accessible within the requested scope.

### INSTRUCTION

After OAuth2.0 is enabled, you can invite others to register by email.

Last update: February 19, 2024

- 697/1069 - 2023 Vesoft Inc.

### 15.9.4 Package management

NebulaGraph Dashboard Enterprise Edition supports managing NebulaGraph installation packages, such as downloading the community edition installation packages or manually uploading the installation packages.

#### **Precautions**

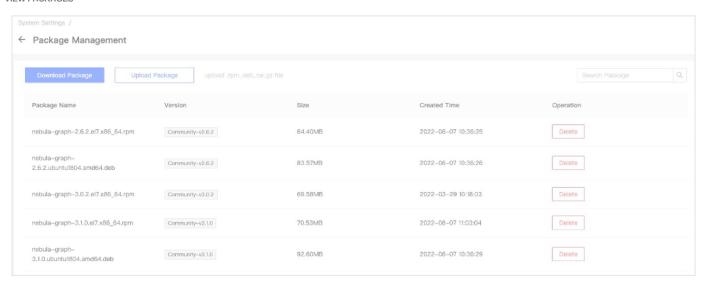
- Only the admin user can manage the installation package.
- Do not support downloading enterprise edition installation packages. For downloading Enterprise Edition packages, please contact us.

#### Entry

- 1. At the top navigation bar of the Dashboard Enterprise Edition page, click System Settings.
- 2. On the left-side navigation bar of the page, click Package Management.

#### Steps

#### VIEW PACKAGES



The list of existing installation packages are displayed on the right-side, showing the package name, version, size, and created time.

Users can filter packages through the search box in the upper right corner.

DOWNLOAD PACKAGES

1. Click Download Package, select the installation package you want to download. The description are as follows:



If you download an existing installation package, the system will prompt you to overwrite the existing installation package.

- Version: Supports stable versions later than v2.5. It is recommended to use the latest version.
- Platform: Supports CentOS 7/8 and Ubuntu 1604/1804/2004.
- Package Type: Supports RPM, DEB and tar.gz.
- 2. Click **Download**.

Users can view the download task information in task center, the task type is package download. If the task status is success, users can return to the **Package Management** page to view the newly downloaded installation package.

UPLOAD PACKAGES

If the required installation package is not listed in the downloaded list, users can manually upload installation packages, such as upload an enterprise edition installation package.

Click **Upload Package**, select the local installation package you want to upload. The package type can be RPM, DEB, or tar.gz. View the upload progress on the upper of the page and wait until the upload is complete.

Users can view the upload task information in task center, the task type is package upload. If the task status is success, users can return to the **Package Management** page to view the newly uploaded installation package.

DELETE PACKAGES

In the operation column of the target installation package, click **Delete** and confirm.

#### FAQ

HOW TO RESOLVE THE ERROR REQUEST ENTITY TOO LARGE?

If users use Nginx as the reverse proxy, the default limit for uploaded files is 1 MB. Add client\_max\_body\_size 200m; to the http{} section of nginx.conf, that means the file of up to 200 MB is allowed to be uploaded.

Last update: February 19, 2024

# 15.10 Metrics

This topic will describe the monitoring metrics in NebulaGraph Dashboard.

# 15.10.1 Machine



- $\bullet$  All the machine metrics listed below are for the Linux operating system.
- The default unit for **Disk** and **Network** is byte. The unit changes with the data magnitude as the page displays. For example, when the flow is less than 1 KB/s, the unit is Bytes/s.
- For all versions of Dashboard Enterprise Edition, the memory occupied by Buff and Cache will not be counted in the memory usage.

### CPU

Parameter	Description
cpu_utilization	The percentage of used CPU.
cpu_idle	The percentage of idled CPU.
cpu_wait	The percentage of CPU waiting for IO operations.
cpu_user	The percentage of CPU used by users.
cpu_system	The percentage of CPU used by the system.

### Memory

Parameter	Description
memory_utilization	The percentage of used memory.
memory_used	The memory space used (not including caches).
memory_free	The memory space available.

### Load

Parameter	Description
load_1m	The average load of the system in the last 1 minute.
Load_5m	The average load of the system in the last 5 minutes.
load_15m	The average load of the system in the last 15 minutes.

- 700/1069 - 2023 Vesoft Inc.

### Disk

Parameter	Description
disk_used_percentage	The disk utilization percentage.
disk_used	The disk space used.
disk_free	The disk space available.
disk_readbytes	The number of bytes that the system reads in the disk per second.
disk_writebytes	The number of bytes that the system writes in the disk per second.
disk_readiops	The number of read queries that the disk receives per second.
disk_writeiops	The number of write queries that the disk receives per second.
inode_utilization	The percentage of used inode.

### Network

Parameter	Description
network_in_rate	The number of bytes that the network card receives per second.
network_out_rate	The number of bytes that the network card sends out per second.
network_in_errs	The number of wrong bytes that the network card receives per second.
network_out_errs	The number of wrong bytes that the network card sends out per second.
network_in_packets	The number of data packages that the network card receives per second.
network_out_packets	The number of data packages that the network card sends out per second.

# 15.10.2 Service

### Period

The period is the time range of counting metrics. It currently supports 5 seconds, 60 seconds, 600 seconds, and 3600 seconds, which respectively represent the last 5 seconds, the last 1 minute, the last 10 minutes, and the last 1 hour.

### Metric methods

Parameter	Description
rate	The average rate of operations per second in a period.
sum	The sum of operations in the period.
avg	The average latency in the cycle.
P75	The 75th percentile latency.
P95	The 95th percentile latency.
P99	The 99th percentile latency.
P999	The 99.9th percentile latency.



Dashboard collects the following metrics from the NebulaGraph core, but only shows the metrics that are important to it.

- 701/1069 - 2023 Vesoft Inc.

# Graph

Parameter	Description
num_active_queries	The number of changes in the number of active queries.  Formula: The number of started queries minus the number of finished queries within a specified time.
num_active_sessions	The number of changes in the number of active sessions. Formula: The number of logged in sessions minus the number of logged out sessions within a specified time. For example, when querying <code>num_active_sessions.sum.5</code> , if there were 10 sessions logged in and 30 sessions logged out in the last 5 seconds, the value of this metric is -20 (10-30).
num_aggregate_executors	The number of executions for the Aggregation operator.
num_auth_failed_sessions_bad_username_password	The number of sessions where authentication failed due to incorrect username and password.
num_auth_failed_sessions_out_of_max_allowed	The number of sessions that failed to authenticate logins because the value of the parameter $FLAG_OUT_OF_MAX_ALLOWED_CONNECTIONS$ was exceeded.
num_auth_failed_sessions	The number of sessions in which login authentication failed.
num_indexscan_executors	The number of executions for index scan operators.
num_killed_queries	The number of killed queries.
num_opened_sessions	The number of sessions connected to the server.
num_queries	The number of queries.
num_query_errors_leader_changes	The number of the raft leader changes due to query errors.
num_query_errors	The number of query errors.
num_reclaimed_expired_sessions	The number of expired sessions actively reclaimed by the server.
num_rpc_sent_to_metad_failed	The number of failed RPC requests that the Graphd service sent to the Metad service.
num_rpc_sent_to_metad	The number of RPC requests that the Graphd service sent to the Metad service.
num_rpc_sent_to_storaged_failed	The number of failed RPC requests that the Graphd service sent to the Storaged service.
num_rpc_sent_to_storaged	The number of RPC requests that the Graphd service sent to the Storaged service.
num_sentences	The number of statements received by the Graphd service.
num_slow_queries	The number of slow queries.
num_sort_executors	The number of executions for the Sort operator.
optimizer_latency_us	The latency of executing optimizer statements.
query_latency_us	The latency of queries.
slow_query_latency_us	The latency of slow queries.
num_queries_hit_memory_watermark	The number of queries reached the memory watermark.

- 702/1069 - 2023 Vesoft Inc.

# Meta

Parameter	Description
commit_log_latency_us	The latency of committing logs in Raft.
<pre>commit_snapshot_latency_us</pre>	The latency of committing snapshots in Raft.
heartbeat_latency_us	The latency of heartbeats.
num_heartbeats	The number of heartbeats.
num_raft_votes	The number of votes in Raft.
transfer_leader_latency_us	The latency of transferring the raft leader.
num_agent_heartbeats	The number of heartbeats for the AgentHBProcessor.
agent_heartbeat_latency_us	The latency of the AgentHBProcessor.
replicate_log_latency_us	The latency of replicating the log record to most nodes by Raft.
num_send_snapshot	The number of times that Raft sends snapshots to other nodes.
append_log_latency_us	The latency of replicating the log record to a single node by Raft.
append_wal_latency_us	The Raft write latency for a single WAL.
num_grant_votes	The number of times that Raft votes for other nodes.
num_start_elect	The number of times that Raft starts an election.

- 703/1069 - 2023 Vesoft Inc.

# Storage

Parameter	Description
add_edges_latency_us	The latency of adding edges.
add_vertices_latency_us	The latency of adding vertices.
commit_log_latency_us	The latency of committing logs in Raft.
<pre>commit_snapshot_latency_us</pre>	The latency of committing snapshots in Raft.
delete_edges_latency_us	The latency of deleting edges.
delete_vertices_latency_us	The latency of deleting vertices.
get_neighbors_latency_us	The latency of querying neighbor vertices.
get_dst_by_src_latency_us	The latency of querying the destination vertex by the source vertex.
num_get_prop	The number of executions for the GetPropProcessor.
num_get_neighbors_errors	The number of execution errors for the GetNeighborsProcessor.
num_get_dst_by_src_errors	The number of execution errors for the GetDstBySrcProcessor.
get_prop_latency_us	The latency of executions for the GetPropProcessor.
num_edges_deleted	The number of deleted edges.
num_edges_inserted	The number of inserted edges.
num_raft_votes	The number of votes in Raft.
num_rpc_sent_to_metad_failed	The number of failed RPC requests that the Storage service sent to the Meta service.
num_rpc_sent_to_metad	The number of RPC requests that the Storaged service sent to the Metad service.
num_tags_deleted	The number of deleted tags.
num_vertices_deleted	The number of deleted vertices.
num_vertices_inserted	The number of inserted vertices.
transfer_leader_latency_us	The latency of transferring the raft leader.
lookup_latency_us	The latency of executions for the LookupProcessor.
num_lookup_errors	The number of execution errors for the LookupProcessor.
num_scan_vertex	The number of executions for the ScanVertexProcessor.
num_scan_vertex_errors	The number of execution errors for the ScanVertexProcessor.
update_edge_latency_us	The latency of executions for the UpdateEdgeProcessor.
num_update_vertex	The number of executions for the UpdateVertexProcessor.
num_update_vertex_errors	The number of execution errors for the UpdateVertexProcessor.
kv_get_latency_us	The latency of executions for the Getprocessor.
kv_put_latency_us	The latency of executions for the PutProcessor.
kv_remove_latency_us	The latency of executions for the RemoveProcessor.
num_kv_get_errors	The number of execution errors for the GetProcessor.
num_kv_get	The number of executions for the GetProcessor.
num_kv_put_errors	The number of execution errors for the PutProcessor.
num_kv_put	The number of executions for the PutProcessor.

- 705/1069 - 2023 Vesoft Inc.

Parameter	Description
num_kv_remove_errors	The number of execution errors for the RemoveProcessor.
num_kv_remove	The number of executions for the RemoveProcessor.
forward_tranx_latency_us	The latency of transmission.
scan_edge_latency_us	The latency of executions for the ScanEdgeProcessor.
num_scan_edge_errors	The number of execution errors for the ScanEdgeProcessor.
num_scan_edge	The number of executions for the ScanEdgeProcessor.
scan_vertex_latency_us	The latency of executions for the ScanVertexProcessor.
num_add_edges	The number of times that edges are added.
num_add_edges_errors	The number of errors when adding edges.
num_add_vertices	The number of times that vertices are added.
num_start_elect	The number of times that Raft starts an election.
num_add_vertices_errors	The number of errors when adding vertices.
num_delete_vertices_errors	The number of errors when deleting vertices.
append_log_latency_us	The latency of replicating the log record to a single node by Raft.
num_grant_votes	The number of times that Raft votes for other nodes.
replicate_log_latency_us	The latency of replicating the log record to most nodes by Raft.
num_delete_tags	The number of times that tags are deleted.
num_delete_tags_errors	The number of errors when deleting tags.
num_delete_edges	The number of edge deletions.
num_delete_edges_errors	The number of errors when deleting edges
num_send_snapshot	The number of times that snapshots are sent.
update_vertex_latency_us	The latency of executions for the UpdateVertexProcessor.
append_wal_latency_us	The Raft write latency for a single WAL.
num_update_edge	The number of executions for the UpdateEdgeProcessor.
delete_tags_latency_us	The latency of deleting tags.
num_update_edge_errors	The number of execution errors for the UpdateEdgeProcessor.
num_get_neighbors	The number of executions for the GetNeighborsProcessor.
num_get_dst_by_src	The number of executions for the GetDstBySrcProcessor.
num_get_prop_errors	The number of execution errors for the GetPropProcessor.
num_delete_vertices	The number of times that vertices are deleted.
num_lookup	The number of executions for the LookupProcessor.
num_sync_data	The number of times the Storage service synchronizes data from the Drainer.
num_sync_data_errors	The number of errors that occur when the Storage service synchronizes data from the Drainer.
sync_data_latency_us	The latency of the Storage service synchronizing data from the Drainer.

- 706/1069 - 2023 Vesoft Inc.

# **Graph space**



Space-level metrics are created dynamically, so that only when the behavior is triggered in the graph space, the corresponding metric is created and can be queried by the user.

Parameter	Description
num_active_queries	The number of queries currently being executed.
num_queries	The number of queries.
num_sentences	The number of statements received by the Graphd service.
optimizer_latency_us	The latency of executing optimizer statements.
query_latency_us	The latency of queries.
num_slow_queries	The number of slow queries.
num_query_errors	The number of query errors.
num_query_errors_leader_changes	The number of raft leader changes due to query errors.
num_killed_queries	The number of killed queries.
num_aggregate_executors	The number of executions for the Aggregation operator.
num_sort_executors	The number of executions for the Sort operator.
num_indexscan_executors	The number of executions for index scan operators.
num_auth_failed_sessions_bad_username_password	The number of sessions where authentication failed due to incorrect username and password.
num_auth_failed_sessions	The number of sessions in which login authentication failed.
num_opened_sessions	The number of sessions connected to the server.
num_queries_hit_memory_watermark	The number of queries reached the memory watermark.
num_reclaimed_expired_sessions	The number of expired sessions actively reclaimed by the server.
num_rpc_sent_to_metad_failed	The number of failed RPC requests that the Graphd service sent to the Metad service.
num_rpc_sent_to_metad	The number of RPC requests that the Graphd service sent to the Metad service.
num_rpc_sent_to_storaged_failed	The number of failed RPC requests that the Graphd service sent to the Storaged service.
num_rpc_sent_to_storaged	The number of RPC requests that the Graphd service sent to the Storaged service.
slow_query_latency_us	The latency of slow queries.

- 707/1069 - 2023 Vesoft Inc.

# Single process metrics

Graph, Meta, and Storage services all have their own single process metrics.

Parameter	Description
context_switches_total	The number of context switches.
cpu_seconds_total	The CPU usage based on user and system time.
memory_bytes_gauge	The number of bytes of memory used.
open_filedesc_gauge	The number of file descriptors.
read_bytes_total	The number of bytes read.
write_bytes_total	The number of bytes written.

Last update: February 19, 2024

- 708/1069 - 2023 Vesoft Inc.

# 15.11 FAQ

This topic lists the frequently asked questions for using NebulaGraph Dashboard. You can use the search box in the help center or the search function of the browser to match the questions you are looking for.

### 15.11.1 "What are Cluster, Node, and Service?"

- Cluster: refers to a group of systems composed of nodes where multiple NebulaGraph services are located.
- Node: refers to the physical or virtual machine hosting NebulaGraph services.
- Service: refers to NebulaGraph services, including Metad, Storaged, and Graphd services.

#### 15.11.2 "What is the cluster status?"

The status of a cluster is as follows:

- installing: The cluster is being created. The process will take about 3 to 10 minutes.
- healthy: All services in the cluster are healthy.
- unhealthy: There is an unhealthy service in the cluster service.

### 15.11.3 "Why authorizing nodes?"

Managing clusters requires the SSH information of the corresponding node. Therefore, you need to have at least an SSH account and the corresponding password with executable permissions before performing operations on Dashboard.

### 15.11.4 "What is scaling?"

NebulaGraph is a distributed graph database that supports dynamic scaling services at runtime. Therefore, you can dynamically scale Storaged and Graphd services through Dashboard. The Metad service cannot be scaled.

### 15.11.5 "Why cannot operate on the Metad service?"

The Metad service stores the metadata of the NebulaGraph database. Once the Metad service fails to function, the entire cluster may break down. Besides, the amount of data processed by the Metad service is not much, so it is not recommended to scale the Metad service. And we directly disabled operating on the Metad service in Dashboard to prevent the cluster from being unavailable due to the misoperation of users.

### 15.11.6 "What impact will the scaling have on the data?"

- Scale out the Storaged service: Dashboard will create and start the Storaged service on the specified machine, which will not affect the existing data. You can choose to perform Balance Leader in the Storage Leader Distribution area and Balance Data in the Partition Distribution area on the **Information->Overview Info** page according to your own needs.
- Scale in the Storaged service: Dashboard will not scale in Storage services until you execute Balance Data Remove to migrate all the partitions from the specified Storage service to other Storage services in the Partition Distribution area on the **Information**>Overview Info page.
- Scaling the Graphd service will not affect the data.

# 15.11.7 "Why Dashboard Enterprise Edition cannot be started?"

- Make sure that the license file is copied to the Dashboard directory and sudo ./dashboard.service start all is executed.
- Make sure that the license is not expired.

- 709/1069 - 2023 Vesoft Inc.

You can also execute cat logs/webserver.log in the Dashboard directory to view the startup information of each module. If the above conditions are met but Dashboard still cannot be started, go to NebulaGraph Official Forum for consultation.

### 15.11.8 "Can I add the NebulaGraph installation package manually?"

You can add the installation package manually in Dashboard. To download the system and RPM/DEB package you need, see How to download NebulaGraph and add the package to nebula-dashboard-ent/download/nebula-graph. And you can select the added package for deployment when creating and scaling out a cluster.

### 15.11.9 Why does it prompt "SSH connection error" when importing a cluster?

If **Service Host** shows 127.0.0.1, and your Dashboard and NebulaGraph are deployed on the same machine when authorizing service hosts, the system will prompt "SSH connection error". You need to change the Host IP of each service to the real machine IP in the configuration files of all NebulaGraph services. For more information, see Configuration management.

If you import a cluster deployed with Docker, it also prompts "SSH connection error". Dashboard does not support importing a cluster deployed with Docker.

Last update: February 19, 2024

- 710/1069 - 2023 Vesoft Inc.

# 16. NebulaGraph Explorer

# 16.1 What is NebulaGraph Explorer

NebulaGraph Explorer (Explorer in short) is a browser-based visualization tool. It is used with the NebulaGraph core to visualize interaction with graph data. Even if there is no experience in graph database, you can quickly become a graph exploration expert.



- To purchase the NebulaGraph Explorer, contact us.
- New users can apply for a 30-day trial. You can also try some functions online in Explorer.



#### 16.1.1 Scenarios

You can use Explorer in one of these scenarios:

- You need to quickly find neighbor relationships from complex relationships, analyze suspicious targets, and display graph data in a visual manner.
- For large-scale data sets, the data needs to be filtered, analyzed, and explored in a visual manner.

#### 16.1.2 Features

Explorer has these features:

- Easy to use: Explorer can be deployed in simple steps.
- User-friendly: Explorer uses simple visual interaction, no need to conceive nGQL sentences, easy to realize graph exploration.
- Flexible: Explorer supports querying data through VID, Tag, and Subgraph.
- Exploration operations: Explorer supports exploration operations on multiple vertices, querying the common neighbors of multiple vertices, and querying the path between the source vertex and the destination vertex.
- Various display: Explorer supports modifying the color and icon of the vertex in the canvas to highlight key nodes. Data can also be displayed in different modes.
- Data storage: Data on a canvas can be stored and exported.
- Inline frame: Explorer supports embedding the canvas on third-party pages.

### 16.1.3 Authentication

Authentication is not enabled in NebulaGraph by default. Users can log into Studio with the root account and any password.

When NebulaGraph enables authentication, users can only sign into Studio with the specified account. For more information, see Authentication.

- 711/1069 - 2023 Vesoft Inc.

# 16.1.4 Version compatibility



Explorer is released separately, not synchronized with NebulaGraph. And the version naming of Explorer is different from that of NebulaGraph. The version correspondence between NebulaGraph and Explorer is as follows.

NebulaGraph version	Explorer version
3.4.0 ~ 3.4.1	3.4.0 \ 3.2.1 \ 3.2.0
3.3.0	3.2.1, 3.2.0
3.1.0 ~ 3.2.x	3.1.0
3.0.0 ~ 3.1.0	3.0.0
2.5.x ~ 3.0.0	2.2.0
2.6.x	2.1.0
2.5.x	2.0.0

### 16.1.5 Video

• NebulaGraph Explorer Intro Demo(5 minutes 22 seconds)

Last update: February 19, 2024

- 712/1069 - 2023 Vesoft Inc.

# 16.2 Deploy and connect

### 16.2.1 Deploy Explorer

This topic describes how to deploy Explorer locally by RPM, DEB and tar packages.

#### **Precautions**

The Dag Controller installation package is built in Explorer starting from version 3.2.0, which provides graph computing services. The user can decide whether or not to start the Dag Controller service. If the Dag Controller service is not started, the **Workflow** menu in Explorer will appear gray and cannot be clicked.

### Prerequisites

Before deploying Explorer, you must check the following information:

- The NebulaGraph services are deployed and started. For more information, see NebulaGraph Database Manual.
- Before the installation starts, the following ports are not occupied.

Port	Description
7002	Web service provided by Explorer



By default, Explorer uses the port 7002. You can modify the httpport in the conf/app.conf file in the installation directory and restart the service.

- The Linux distribution is CentOS.
- The license is ready.



 $\label{license} \mbox{License is only available in the Enterprise Edition. To obtain the license, apply for $NebulaGraph Explorer Free Trial. $$$ 

• The HDFS services are deployed if graph computing is required. The namenode uses port 8020 by default, and the datanode uses port 50010 by default.



If the HDFS port is unavailable, the connection timeout message may be displayed.

### RPM-based deployment

INSTALLATION

1. Select and download the RPM package according to your needs. It is recommended to select the latest version.

- 713/1069 - 2023 Vesoft Inc.

# Sterpriseonly

You can apply online for Explorer free trial. Contact us to purchase. For features of Explorer, see Pricing.

2. Use sudo rpm -i <rpm> to install RPM package.

 $For example, use the following command to install Explorer. The default installation path is \ {\it usr/local/nebula-explorer}.$ 

```
sudo rpm -i nebula-explorer-<version>.x86_64.rpm
```

You can also install it to the specified path using the following command:

```
sudo rpm -i nebula-explorer-<version>.x86_64.rpm --prefix=<path>
```

3. Copy the license to the installation path.

```
sudo cp -r <license> <explorer_path>
```

For example:

```
sudo cp -r nebula.license /usr/local/nebula-explorer
```

- 4. (Optional) Configure the Dag Controller. See the Configure Dag Controller section below.
- 5. Enter the folder nebula-explorer, start the service using the following command.

```
cd nebula-explorer

# Start Explorer.
sudo ./scripts/start.sh

# (Optional) Start Dag Controller.
sudo ./dag-ctrl/scripts/start.sh
```

START AND STOP

You can use SystemCTL to start and stop the service.

```
systemctl status nebula-explorer #Check the status
systemctl stop nebula-explorer #Stop the service
systemctl start nebula-explorer #Start the service
```

You can also start or stop the service manually using the following command in the installation directory.

```
sudo ./scripts/start.sh #Start Explorer
sudo ./scripts/stop.sh #Stop Explorer
sudo ./dag-ctrl/scripts/start.sh #Start Dag Controller
sudo ./dag-ctrl/scripts/stop.sh #Stop Dag Controller
```

UNINSTALLATION

You can uninstall Explorer using the following command:

```
sudo rpm -e nebula-graph-explorer-<version>.x86_64
```

### **DEB-based deployment**

INSTALLATION

1. Select and download the RPM package according to your needs. It is recommended to select the latest version. Common links are as follows:



You can apply online for Explorer free trial. Contact us to purchase. For features of Explorer, see Pricing.

2. Run sudo dpkg -i <package\_name> to unpack the DEB package.

For example, run the following command to install Explorer (The default installation path is  $\protect\ensuremath{\mathsf{Just/local/nebula-explorer}}$ ).

sudo dpkg -i nebula-explorer-3.4.0.x86\_64.deb



You cannot customize the installation path of Explorer when installing a DEB package.

3. Copy the license to the Explorer installation path.

```
Sudo cp -r <license> <explorer_path>
```

#### For example:

Sudo cp -r nebula.license /usr/local/nebula-explorer

- 4. (Optional) Configure the Dag Controller. See the Configure Dag Controller section below.
- 5. Enter the folder nebula-explorer, start the service using the following command.

bash

- cd nebula-explorer
- # Start Explorer.
- sudo ./lib/start.sh
- # (Optional) Start Dag Controller.
- sudo ./dag-ctrl/scripts/start.sh

VIEW THE STATUS

sudo systemctl status nebula-explorer.service

STOP THE SERVICE

sudo systemctl stop nebula-explorer.service

UNINSTALLATION

Run the following command to uninstall Explorer:

sudo dpkg -r nebula-explorer

# TAR-based deployment

INSTALLATION

1. Select and download the TAR package according to your needs. It is recommended to select the latest version. Common links are as follows:



Explorer is only available in the Enterprise Edition. Click Pricing to see more.

2. Use tar -xvf to decompress the TAR package.

tar -xvf nebula-explorer-<version>.tar.gz

3. Copy the license to the nebula-explorer directory.

cp -r <license> <explorer\_path>

For example:

- 715/1069 - 2023 Vesoft Inc.

```
cp -r nebula.license /usr/local/nebula-explorer
```

- 4. (Optional) Configure the Dag Controller. See the Configure Dag Controller section below.
- 5. Enter the folder nebula-explorer, start the service using the following command.

```
cd nebula-explorer

# Start Explorer and Dag Controller.
sudo ./scripts/start.sh

# Start Explorer separately.
sudo nohup ./nebula-explorer-server > explorer.log 2>&1 &
```

STOP SERVICE

You can use kill pid to stop the service.

```
kill $(lsof -t -i :7002)
```

#### **Configure Dag Controller**

Dag Controller is a task scheduling tool that can schedule the jobs which type is DAG (directed acyclic graph). The job consists of multiple tasks to form a directed acyclic graph, and there is a dependency between the tasks.

The Dag Controller can perform complex graph computing with NebulaGraph Analytics. For example, the Dag Controller sends an algorithm request to NebulaGraph Analytics, which saves the result to NebulaGraph or HDFS. The Dag Controller then takes the result as input to the next algorithmic task to create a new task.

STEPS

1. Complete the SSH password-free configurations so that the Dag Controller machine can log directly into the NebulaGraph Analytics machines and all machines within the NebulaGraph Analytics cluster can connect directly to each other without passwords.

For example, the user in the machine A (Dag Controller) log directly into machine B-1 in the NebulaGraph Analytics cluster over SSH without passwords. Run the following commands on the machine A:

```
//Press Enter to execute the default option to generate the key.
ssh-keygen -t rsa
//After the public key file of machine A is installed to the user of the machine B-1, you can log into the machine B-1 from the machine A without passwords.
ssh-copy-id -i -/.ssh/id_rsa.pub -8_user>@<B_IP>
```

In the same way, complete the SSH password-free configurations so that the user in the machine A can log directly into the machine B-2, B-3, etc. and all machines within the NebulaGraph Analytics cluster can connect directly to each other without passwords.

2. Run eval \$(ssh-agent) on the Dag Controller machine to start the ssh-agent, then run ssh-add ~/.ssh/id\_rsa to give the private key to the ssh-agent to manage.



ssh-agent is a key manager that manages multiple keys and provides proxies for other programs that need to use SSH key pairs.

3. Configure the username and port of the NebulaGraph Analytics in the file dag-ctrl-api.yaml, the file path is dag-ctrl/etc/dag-ctrl-api.yaml. If there are multiple machines, ensure that the usernames and ports are the same.

```
# configuration name
Name: task-api
Host: 0.0.0.0
                  # The IP address of Dag Controller.
Port: 9002
                  # The port of Dag Controller
Timeout: 60000
                 # he timeout duration of HTTP interface requests.
Log:
                  # The parameters related to log printing. For more Information, see https://go-zero.dev/cn/docs/blog/tool/logx/
  Mode: file
                  # The log printing method
                  # The maximum number of days to keep logs
  KeepDays: 7
                  # The output path of the log file
  Path: logs
  Level: info
                 # The log printing level
 Compress: false # Whether the log needs to be compressed
```

```
# The user name and SSH port of the NebulaGraph Analytics machine.
UserName: vesoft
Port: 22
# The parallel thread pool sizes of the tasks and jobs.
 Sleep: 3
            \mbox{\#} Check every 3 seconds for any outstanding jobs
          # Up to 3 jobs can be executed in parallel.
Size: 3
TaskPool:
CheckStatusSleep: 1
                       # Check the task status every second.
            # Up to 10 tasks can be executed in parallel.
Size: 10
VarDataListMaxSize: 100  # If HDFS columns are read, the number is limited to 100 at a time.
# Other
 Enable: false # Whether to enable Debugging.
# The key for the Explorer to communicate with the Dag Controller. No modification is required.
     ---BEGIN RSA PRIVATE KEY----
 MIICXAIBAAKBgQDcR0keIMmmV...
     --END RSA PRIVATE KEY----
RsaPubKey:
     ---BEGIN RSA PUBLIC KEY----
 MIGJAoGBANxHSR4gyaZX7uet7...
```

4. Configure the algorithm file path (exec\_file) only in the file tasks.yaml, the file path of which is dag-ctrl/etc/tasks.yaml. Currently, all exec\_file parameters are set to the path of the run\_algo.sh file.



- The algorithm files are provided by NebulaGraph Analytics. Please find the scripts directory under the installation path of NebulaGraph Analytics above. All algorithm files are in this directory.
- If there are multiple machines, ensure that the algorithm file paths are the same.
- The other parameters are the execution parameters of the algorithms and are configured later on the visual workflow page.

exec\_file: /home/xxx/nebula-analytics/scripts/run\_algo.sh

#### Next to do

Connect to Explorer

Last update: February 19, 2024

- 717/1069 - 2023 Vesoft Inc.

### 16.2.2 Connect to NebulaGraph

After successfully launching Explorer, you need to configure to connect to NebulaGraph. You can connect directly to NebulaGraph by default. To ensure data security, OAuth2.0 authentication is also supported. You can connect to NebulaGraph only after the authentication is passed.

### **Prerequisites**

Before connecting to the NebulaGraph database, you need to confirm the following information:

- The NebulaGraph services and Explorer are started. For more information, see Deploy Explorer.
- You have the local IP address and the port used by the Graph service of NebulaGraph. The default port is 9669.
- You have a NebulaGraph account and its password.
- We recommend you to use the Chrome browser of the version above 89. Otherwise, there may be compatibility issues.

### OAuth2.0 Configuration



The feature is still in beta. It will continue to be optimized.



If you want to connect directly to NebulaGraph, see **Procedure** below.

To enable OAuth2.0 authentication, modify the configuration file in the Explorer installation directory. The path is config/app-config.yaml.

- 718/1069 - 2023 Vesoft Inc.

The descriptions of the OAuth configuration are as follows.

Parameter	Example	Description
Enable	false	Enable or disable OAuth2.0 authentication.
ClientID	4953xxx- mmnoge13xx.apps.googleusercontent.com	The application's ClientId.
ClientSecret	GOCxxx-xaytomFexxx	The application's ClientSecret.
RedirectURL	http://dashboard.vesoft-inc.com/login	The URL that redirects to Dashboard.
AuthURL	https://accounts.google.com/o/oauth2/auth	The URL used for authentication.
TokenURL	https://oauth2.googleapis.com/token	The URL used to get the access_token.
UserInfoURL	https://www.googleapis.com/oauth2/v1/userinfo	The URL used to get the user information.
UsernameKey	email	The key of the user name.
Organization	vesoft company	The organization name.
TokenName	oauth_token	The name of the token in the cookie.
Scope	email	Scope of OAuth permissions. The scope of permissions needs to be a subset of the scope configured by the vendor's OAuth2.0 platform, otherwise, the request will fail. Make sure the UsernameKey is accessible within the requested scope.
AvatarKey	picture	The key of the avatar in the user information.

After the configuration is complete, restart the Explorer service. The OAuth authentication is displayed on the login page. You can continue to connect to NebulaGraph only after the authentication is passed.

- 719/1069 - 2023 Vesoft Inc.

# Procedure

To connect Explorer to NebulaGraph, follow these steps:

 $_{\hbox{1. Type }}$  http://<ip\_address>:7002 in the address bar of your browser.

The following login page shows that Explorer is successfully connected to NebulaGraph.



# Note

When logging into NebulaGraph Explorer for the first time, the content of END USER LICENSE AGREEMENT is displayed on the login page. Please read it and then click I agree.

- 2. On the **Config Server** page of Explorer, configure these fields:
- Graphd IP address: Enter the IP address of the Graph service of NebulaGraph. For example, 192.168.10.100.

# Note

- When NebulaGraph and Explorer are deployed on the same machine, you must enter the IP address of the machine, instead of 127.0.0.1 or localhost.
- When connecting a NebulaGraph database on a new tab, The new session will overwrite the sessions of the old TAB. If you need to log in to multiple NebulaGraph databases at the same time, you can use different browsers or non-trace mode.
- Port: The port of the Graph service. The default port is 9669.
- Username and Password: Fill in the log in account according to the authentication settings of NebulaGraph.
- $\bullet$  If authentication is not enabled, you can use root and any password as the username and its password.
- If authentication is enabled and no account information has been created, you can only log in as GOD role and use root and nebula as the username and its password.
- If authentication is enabled and different users are created and assigned roles, users in different roles log in with their accounts and passwords.
- 3. After the configuration, click the  ${f Login}$  button.



One session continues for up to 30 minutes. If you do not operate Explorer within 30 minutes, the active session will time out and you must connect to NebulaGraph again.

- 721/1069 - 2023 Vesoft Inc.

A welcome page is displayed on the first login, showing the relevant functions according to the usage process, and the test datasets can be automatically downloaded and imported.

To visit the welcome page, click ? -> Beginner's Guide.

# Clear connection

When Explorer is still connected to a NebulaGraph database, on the upper right corner of the page, select Connect.

After that, if the **configuration database** page is displayed on the browser, it means that Explorer has successfully disconnected from the NebulaGraph.

# 16.2.3 NebulaGraph Explorer license

A license is a software authorization certificate used to authorize the use of a software product. When deploying NebulaGraph Explorer, you need to add a license to start it. This document describes the license information on NebulaGraph Explorer.

#### **Precautions**

- If the license file is not deployed, NebulaGraph Explorer cannot be started.
- Do not modify the license file, otherwise the license will become invalid.
- If the license is about to expire, contact us to apply for renewal.
- The transition period after the license expires is 14 days:
- If you start the Enterprise Edition within 30 days before the license expires or on the day the license expires, a log will be printed as a reminder.
- The license can still be used for 14 days after it expires.
- If the license has expired for 14 days, you will not be able to start the Enterprise Edition, and a log will be printed as a reminder.

### Obtain a NebulaGraph Explorer license

Contact us to apply for a NebulaGraph Explorer license.



You can apply online for a 30-day free trial of NebulaGraph Explorer.

## License description

NebulaGraph Explorer license is a file named nebula. License that contains the following information:

The license file contains information such as issuedDate and expirationDate. The description is as follows.

Parameter	Description
vendor	The supplier.
organization	The username.
issuedDate	The date that the license is issued.
expirationDate	The date that the license expires.
product	The product type. The product type of NebulaGraph Explorer is nebula_graph_explorer.
version	The version information.
licenseType	The license type (a reserved parameter), including enterprise, samtl_bussiness, pro, and individual.
gracePeriod	The buffer time (in days) for the service to continue to be used after the license expires, and the service will be stopped after the buffer period. The trial version of license has no buffer period after expiration and the default value of this parameter is 0.
clusterCode	The user's hardware information, which is also the unique identifier of the cluster. This parameter is not available in the trial version of the license.

# Use a NebulaGraph Explorer license

For how to use a NebulaGraph Explorer license, see Deploy NebulaGraph Explorer.

# Renew a NebulaGraph Explorer license

Follow the steps below to renew your NebulaGraph Explorer license.

- 1. Contact us to apply for a new NebulaGraph Explorer license file nebula. License.
- 2. In the NebulaGraph Explorer installation directory, such as /usr/local/nebula-explorer, replace the old license file with the new one.



You cannot log into NebulaGraph Explorer once the license expires. To avoid business interruptions, please renew your license in time.

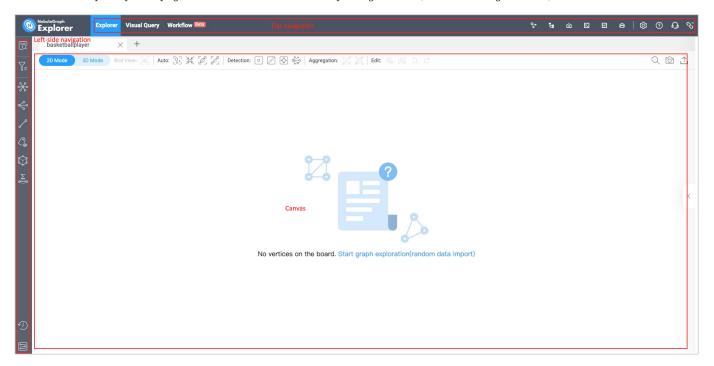
Last update: February 19, 2024

- 724/1069 - 2023 Vesoft Inc.

# 16.3 Page overview

This topic introduces the NebulaGraph Explorer page to help you learn more about NebulaGraph Explorer's functions.

The NebulaGraph Explorer page consists of three modules top navigation bar, left-side navigation bar, and canvas.



# 16.3.1 Top navigation bar

Icon/ Element	Description
Explorer	Visually explore and analyze data. For more information, see Start querying, Vertex Filter, Graph exploration and Graph algorithm.
Visual Query	Visually construct scenarios for data queries. For more information, see Visual Query.
Workflow	Visually construct custom workflows for complex graph computing. The <b>Workflow</b> page can be displayed only when <b>Workflow</b> is enabled in . For more information, see Workflow overview.
<b>℃</b>	Users can design their schemas on the canvas to visually display the relationships between vertices and edges. For more information, see Schema drafting.
ŧ:	Manage NebulaGraph database graph spaces. For more information, see Create a schema.
ጥ	Bulk import of data into NebulaGraph. For more information, see Import data.
入	Query the NebulaGraph data with nGQL statements. For more information, see Console.
	The template of the nGQL. For details, see $nGQL$ template.
8	Manage the users in NebulaGraph database. For more information, see Database user Management $\circ$
<b>®</b>	Global Settings. You can set the language of the Explorer page, enable Beta functions, and the maximum number of canvas query results.
<b>②</b>	Guide and help you in using NebulaGraph.
$\Omega$	Feedback page. You can report troubles, submit suggestions, participate in research, or contact the NebulaGraph team.
E	Show the connection information and version information. You can change passwords and log out.

# 16.3.2 Left-side navigation bar



After logging into Explorer, select a graph space and click on it to unlock query and exploration functions in the left-side navigation bar. For more information, see Choose graph spaces.

- 726/1069 - 2023 Vesoft Inc.

Click the icons in the left-side navigation bar to import, analyze, and explore graph data. The descriptions of the icons are as follows:

Icon	Description
Q	Enter VIDs or tags to query data. For more information, see Ways to query data.
√≡	Search for target vertexes displayed on the canvas. For more information, see Filter vertices.
¥	Perform explorations on the vertices on the canvas by setting edge directions, steps, and filtering conditions. For more information, see Graph exploration.
~	Select at least two vertices on the canvas to search for their common neighbors. For more information, see Graph exploration.
2	Find all paths, the shortest path, and the non-loop paths from the source to the destination vertex. For more information, see <u>Graph exploration</u> .
2	Choose whether to display the properties of vertices or edges on the canvas. For more information, see Graph exploration.
<b>©</b>	Perform graph computing based on the vertexes and edges on the canvas. For more Information see Graph computing.
o <u></u> o	Perform property calculation based on the aggregated edges on the canvas. For more Information see Property calculation $\circ$
D	View historical snapshots. For more information, see Canvas snapshots.
	View all graph spaces. Click a graph space to create a canvas corresponding to it. For more information, see Choose graph spaces.

# 16.3.3 Canvas



After logging into Explorer, select a graph space and click on it to enter the canvas page. For more information, see Choose graph spaces.

Graph data can be displayed visually on a canvas. The canvas consists of the following parts:

- Tabs on the Top
- Visualization modes
- Data storage
- Search box
- Layouts
- Minimap
- Data overview

For more information, see Canvas overview.

# 16.4 Database management

### 16.4.1 Schema drafting

Explorer supports the schema drafting function. Users can design their schemas on the canvas to visually display the relationships between vertices and edges, and apply the schema to a specified graph space after the design is completed.

### Features

- Design schema visually.
- Applies schema to a specified graph space.
- Export the schema as a PNG image.

### Entry

At the top navigation bar, click  $^{f y}$  .

### Design schema

The following steps take designing the schema of the basketballplayer dataset as an example to demonstrate how to use the schema drafting function.

- 1. At the upper left corner of the page, click **New**.
- 2. Create a tag by selecting the appropriate color tag under the canvas. You can hold down the left button and drag the tag into the canvas.
- 3. Click the tag. On the right side of the page, you need to fill in the name of the tag as player, and add two properties name and age.
- 4. Create a tag again. The name of the tag is team, and the property is name.
- 5. Connect from the anchor point of the tag player to the anchor point of the tag team. Click the generated edge, fill in the name of the edge type as serve, and add two properties start\_year and end\_year.
- 6. Connect from an anchor point of the tag player to another one of its own. Click the generated edge, fill in the name of the edge type as follow, and add a property degree.
- 7. After the design is complete, click at the top of the page to change the name of the draft, and then click at the top right corner to save the draft.

- 728/1069 - 2023 Vesoft Inc.



### Apply schema

- 1. Select the draft that you want to import from the **Draft list** on the left side of the page, and then click **Apply to Space** at the upper right corner.
- 2. Import the schema to a new or existing space, and click **Confirm**.



- $\bullet$  For more information about the parameters for creating a graph space, see CREATE SPACE.
- If the same schema exists in the graph space, the import operation fails, and the system prompts you to modify the name or change the graph space.

### Modify schema

Select the schema draft that you want to modify from the **Draft list** on the left side of the page. Click at the upper right corner after the modification.



The graph space to which the schema has been applied will not be modified synchronously.

# Delete schema

Select the schema draft that you want to delete from the Draft list on the left side of the page, click X at the upper right corner of the thumbnail, and confirm to delete it.

# **Export Schema**

Click  $extstyle\Delta$  at the upper right corner to export the schema as a PNG image.

- 729/1069 - 2023 Vesoft Inc.

### 16.4.2 Create a schema

Explorer allows you to create a schema by using GUI.



- Users can use the Schema drafting function to design schema visually. For more information, see Schema drafting.
- Users can directly execute nGQL commands on the console to manage the schema.

### **Prerequisites**

- Your account has the privilege of GOD, ADMIN, or DBA. For more information, see Roles and privileges .
- The schema is designed.



If no graph space exists and your account has the GOD privilege, you can create a graph space on the Console page.

#### **Entry**

At the top navigation bar, click

## Create graph space

- 1. Click Create Space.
- 2. Set parameters. For descriptions of the parameters, see CREATE SPACE.
- 3. Click Create.

## Create Tag or Edge type

- $1. \ In \ the \ \textbf{Graph Space List} \ page, \ find \ a \ graph \ space \ and \ then \ click \ its \ name \ or \ click \ \textbf{Schema} \ in \ the \ \textbf{Operations} \ column.$
- 2. Click the  $\boldsymbol{Tag}$  or  $\boldsymbol{Edge}$   $\boldsymbol{Type}$  tab and click the  $\boldsymbol{Create}$  button.
- 3. Set parameters. For descriptions of the parameters, see  $\frac{\text{CREATE TAG}}{\text{CREATE EDGE}}$ .



If no index is set for the tag, you can set the TTL configuration. For more information, see TTL configuration.

## 4. Click Create.

In the Tag and Edge Type lists, you can perform modification and deletion operations.

- 731/1069 - 2023 Vesoft Inc.

#### Create index



- Before creating an index, ensure that the associated Tag or Edge type has been created.
- The index can decrease the write speed during data import. We recommend that you import data first and then create and rebuild an index. For more information, see Index overview.
- 1. In the Graph Space List page, find a graph space and then click its name or click Schema in the Operations column.
- 2. Click the **Index** tab and click the **Create** button.
- 3. Set parameters. For descriptions of the parameters, see CREATE INDEX.



The order of the indexed properties has an effect on the result of the LOOKUP statement. For more information, see LOOKUP.

4. Click Create.

In the Index list, you can rebuild or delete the index.

#### View statistics

- 1. In the Graph Space List page, find a graph space and then click its name or click Schema in the Operations column.
- 2. Click Statistics tab and click the Refresh button.

### View schema



To display this function, you need to enable **View Schema** in **②**.



- 1. In the Graph Space List page, find a graph space and then click its name or click Schema in the Operations column.
- 2. Click View Schema tab and click the Refresh button.

### Other operations

In the **Graph Space List** page, find a graph space and then perform the following operations in the **Operations** column:

- View Schema DDL: Displays schema creation statements for the graph space, including graph spaces, tags, edge types, and indexes.
- Clone Graph Space: Clone the schema of the graph space to a new graph space.
- Delete Graph pace: Delete the graph space, including the schema and all vertices and edges.

Last update: February 19, 2024

- 732/1069 -2023 Vesoft Inc.

# 16.4.3 Import data

Explorer allows you to import data in CSV format into NebulaGraph using GUI.

# Prerequisites

- CSV files meet the demands of the Schema.
- Your account has the privilege of GOD, ADMIN, or DBA. For more information, see Roles and privileges .

# Entry

At the top navigation bar, click

# Steps

UPLOAD FILES

1. In the Upload Files tab, click the Upload Files button and then choose CSV files.



You can choose multiple CSV files at the same time.

2. After uploading, you can click the button in the **Operations** column to preview the file content.

IMPORT DATA

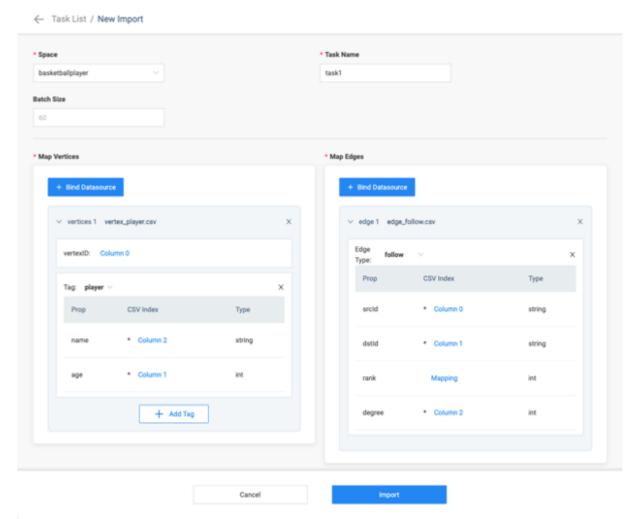
1. In the **Import Data** tab, click + **New Import** button to complete these operations:

- 733/1069 - 2023 Vesoft Inc.

# Caution

users can click **Import Template** to download the example configuration file example.yaml, and upload the configuration file after configuration. The configuration mode is similar to that of NebulaGraph Importer, but all file paths for configuration files in the template retain the filename only. And make sure all CSV data files are uploaded before importing the YAML file.

- Space: The name of the graph space that you want to import data from.
- Task Name: Automatically generated by default and can be modified.
- Batch Size (Optional): The number of rows of imported data per batch.
- Map Vertices: Click the Bind Datasource button, then select bind source file in the dialog box, and click the Confirm button.
- In the **vertexID** item of the **vertices 1** drop-down list, click **Select CSV Index**, and select the data source for VID in the pop-up dialog box.
- Click the **Add Tag** button, then click **Select Tag** in the newly created tab and select the Tag you want to associate. In the property list, select the data source for the property.
- $\bullet$   $\mathbf{Map}$   $\mathbf{Edges}:$  Similar to the operation of the  $\mathbf{Map}$   $\mathbf{Vertices}.$



2. After completing the settings, click the Import button and enter the password of your NebulaGraph account.

On the  ${f Import\ Data}$  tab, you can view logs, download logs, download configuration files, and delete tasks.

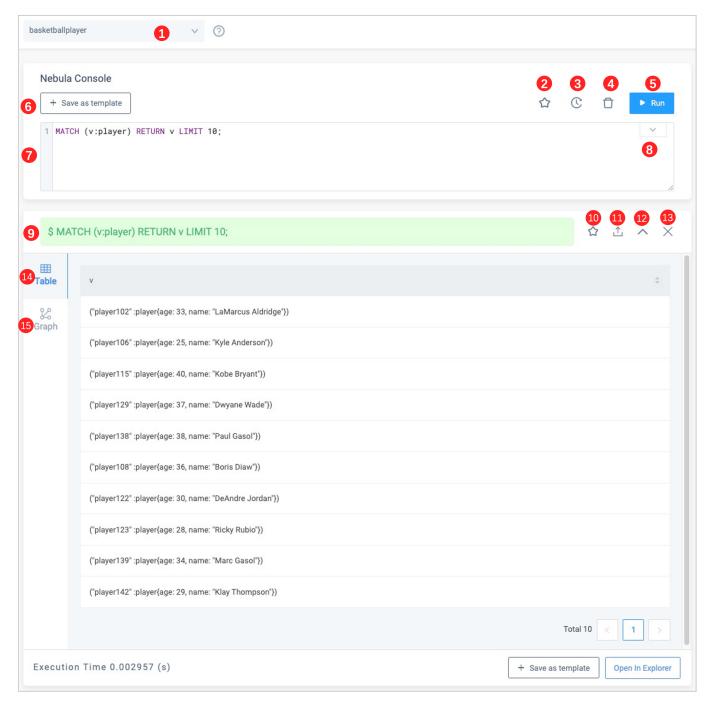
# 16.4.4 Explorer console

Explorer console allows you to enter nGQL statements and visualize the query results. This topic describes the console page.

# Entry

At the top navigation bar, click

## Overview



The following table lists the functions on the console page.

number	function	descriptions
1	select a space	Select a space in the Current Graph Space list. The USE <space_name> statement in the console is not supported to switch graph spaces.</space_name>
2	favorites	Click the button to expand the favorites, click one of the statements, and the input box will automatically enter the statement.
3	history list	Click button representing the statement record. In the statement running record list, click one of the statements, and the statement will be automatically entered in the input box. The list provides the record of the last 15 statements.
4	clean input box	Click $\Box$ button to clear the content entered in the input box.
5	run	After inputting the nGQL statement in the input box, click button to indicate the operation to start running the statement.
6	save as template	Save the nGQL statement entered in the input box as a template. For details, see $nGQL$ template.
7	input box	After inputting the nGQL statements, click the button to run the statement. You can input multiple statements and run them at the same time by using the separator ; , and also use the symbol // to add comments.
8	custom parameters display	Click the button to expand the custom parameters for parameterized query. For details, see Manage parameters.
9	statement running status	After running the nGQL statement, the statement running status is displayed. If the statement runs successfully, the statement is displayed in green. If the statement fails, the statement is displayed in red.
10	add to favorites	Click the button to save the statement as a favorite, the button for the favorite statement is colored in yellow exhibit.
11	export CSV file or PNG file	After running the nGQL statement to return the result, when the result is in <b>Table</b> window, click the button to export as a CSV file. Switch to the <b>Graph</b> window and click the button to save the results as a CSV file or PNG image export.
12	expand/hide execution results	Click the button to hide the result or click button to expand the result.
13	close execution results	Click the $\times$ button to close the result returned by this nGQL statement.
14	Table window	Display the result from running nGQL statement. If the statement returns results, the window displays the results in a table.
15	<b>Graph</b> window	Display the result from running nGQL statement. If the statement returns the complete vertex-edge result, the window displays the result as a graph . Click the button on the right to view the overview panel.

# 16.4.5 nGQL template

NebulaGraph Explorer supports saving the commonly nGQL statement as a template for yourself or others. The text in the nGQL statement supports parameterization, and parameter values can be filled in as needed.

### Prerequisites

The schema has been created in the NebulaGraph database.

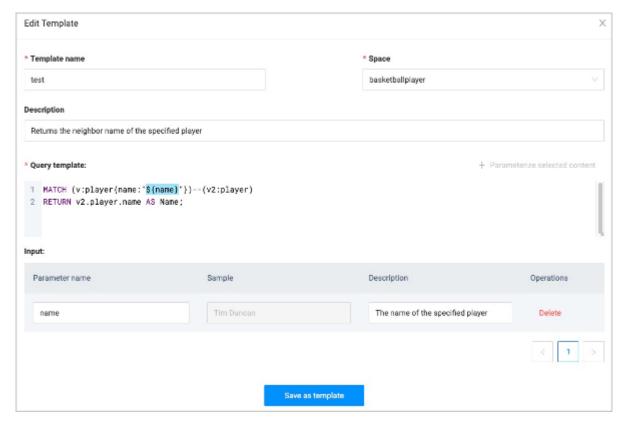
# Entry

At the top navigation bar, click  $\square$ .

### Create new template

1. Click + New Template, and set the parameters as follows.

- 737/1069 - 2023 Vesoft Inc.



Parameter	Example	Description
Template name	test	The name of the template.
Space	basketballplayer	The graph space to which the template applies.
Description	Returns the neighbor name of the specified player	Describes the function of the template.
Query template	MATCH (v:player{name:"\$ {name}"})(v2:player) RETURN v2.player.name AS Name;	nGQL template. You can select the text you want to parameterize, click + parameterize selected content on the right, and set the parameter name and description. In the example, \${name} is parameterized text. In actual use, you can fill in a name such as Tim Duncan.  You can add comments in a single line using //.
Input	-	Displays parameterized text content. You can edit or delete it.

Note

Click + Save as template on the upper left corner of the console page to use the entered query statement as a template statement automatically.

# $2. \ Click \ \textbf{Save as template}.$

# Other Operations

- ${}^{\raisebox{3.5pt}{\text{\circlick}}}$  on the right of the target template to modify the template context.
- · Click on the right of the target template to automatically jump to the console and enter the template.
- $\dot{}$  Click  $\dot{\overline{\mathbb{Q}}}$  on the right of the target template to delete the template.

- 738/1069 - 2023 Vesoft Inc.

- The filter box in the upper right corner allows you to filter templates for a specified graph space.
- The search box in the upper right corner allows you to search the template name.

# Use template

- (Recommended) Use templates on the graph exploration page. For details, see Start querying.
- \* Click on the template list page to automatically jump to the console and enter the template. You need to modify the parameterized text.

# 16.4.6 Database user management

NebulaGraph Explorer supports managing the users in the NebulaGraph database, including creating users, deleting users, changing passwords, etc.

#### **Prerequisites**

The user who logs in to Explorer must have permissions for related operations. For example, users with <code>God</code> permission can perform all operations, and users with <code>Admin</code> permission can authorize the permission of a graph space within their permission to other users. For details about role privileges, see Roles and privileges.

# Entry

At the top navigation bar, click  $m{8}$  .



# Create user



Only the root user can create users.

1. In the tab **User list**, click **Create User** and set the following parameters.

Parameters	Description
Account	The user name.
Password	The password corresponding to the user name.
IP Whitelist	The user can connect to NebulaGraph only from IP addresses in the list. Use commas to separate multiple IP addresses. Only NebulaGraph Enterprise Edition supports the parameter.



Click Add in the upper left corner to create users in batches.

# 2. Click **Confirm**.

#### Authorize user

- 1. Switch the tab to **Authorization**, and select the name of the graph space that you want to authorize to a user in the upper left corner. The page shows all users (except root user) who have permission on the graph space.
- 2. Click **Grant Role** and set the following parameters.

Parameters	Description
Username Set the user name to be authorized. If you log in as the root user, select the user from the drop-dow you log in with the Admin permission, fill in the user name manually.	
Role	Select the role to be authorized from the drop-down menu. For details about role privileges, see Roles and privileges.

#### 3. Click Confirm.

### Other operations in the user list



Only the root user can view the User List.

- View: View the user permissions in each space.
- Edit: Change the password and IP whitelist of the user. You do not need to provide the old password when changing the password. If the user is not root, you can change the password in on the upper right corner of the page.
- Delete User: Only the root user can delete other users.
- Search user: Search for the account by keyword.

# Other operations in the authorization

- Edit: Change the role of the user.
- Revoke Role: Revoke the role of the user.
- Search user: Search for the account by keyword.



After a user is modified or revoked, the modification takes effect only after the user logs in next time.

Last update: February 19, 2024

- 741/1069 - 2023 Vesoft Inc.

# 16.5 Graph explorer

# 16.5.1 Choose graph spaces

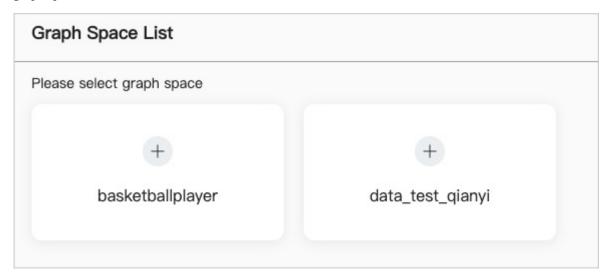
You must first choose a graph space and then query and analyze data with Explorer. This topic introduces how to choose a graph space.

### Prerequisite

You have connected to Explorer. For details, see Connect to Explorer.

#### Steps

After connecting to Explorer, the system automatically displays the graph space selection page. You only need to select the target graph space.



If you want to select a graph space again, follow the below steps to choose one.

1. In the navigation bar on the left side of the Explorer page, click the graph space icon  $\blacksquare$ .

2. Choose the target graph space.



You can select the same or different graph spaces multiple times, and each selection creates a new canvas.

Last update: February 19, 2024

- 742/1069 - 2023 Vesoft Inc.

# 16.5.2 Start querying

To explore graph data, users need to query some initial data, and based on these initial data, can further analysis and filtering. This topic describes how to query initial data.

### Prerequisites

Select a target graph space before querying data. For more information, see Choose graph spaces.



For versions of NebulaGraph below 3.0.0, you need to create an index before querying data. For more information, see Create an index.

# Steps

Click the **Start** icon to query target data on the Explorer page. The queried data will be displayed on the canvas. You have the following ways to query data:

- Query by VID
- · Query by Tag
- · Query Subgraph
- · Query by template

QUERY BY VID

You can enter VIDs to query the target vertices.

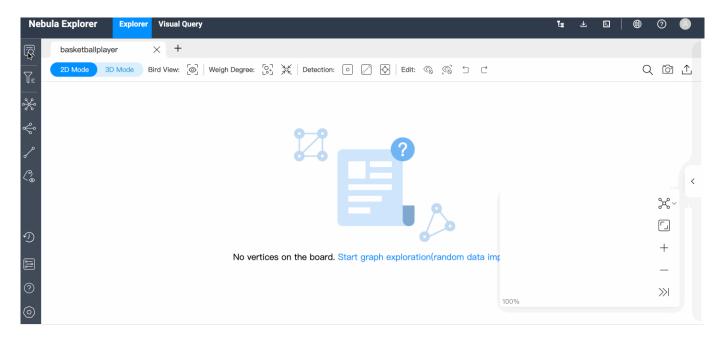
There are three ways to generate VIDs: Manual input, Random import, and File import.



Only one VID per row is supported in the querying area. Press Enter to separate the VIDs.

The following GIF shows how to query data using the  $\,$  basketballplayer  $\,$  graph space and related data.

- 743/1069 - 2023 Vesoft Inc.



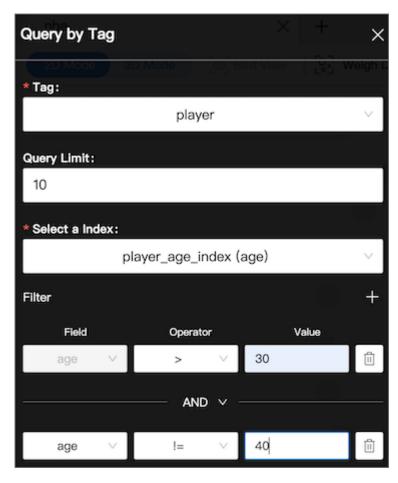
#### QUERY BY TAG

You can select the tag and corresponding index to query the target vertices, and set the number of results limit or filter conditions.



Make sure that the corresponding tags and indexes exist in the graph space when querying by tag. For more information, Create tags and Create indexes.

The following example queries 10 players whose age is greater than 30 years old and not equal to 40 years old.



QUERY SUBGRAPH

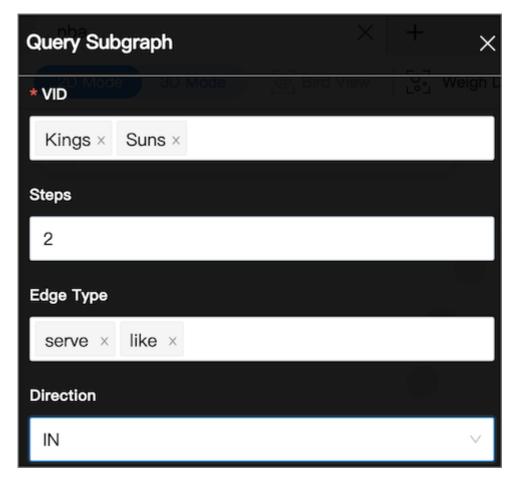
When querying subgraphs, you can specify the number of steps, edge types, and the direction of inflow and outflow of the subgraph. VID is mandatory. The default value of optional steps is 1, and the default value of optional edge type is all.



When multiple VIDs are entered, the VIDs are separated by the Enter key.

The following is an example of VIDs Kings and Suns, step number 2, and incoming edge types with a VID value of 101, the number of steps of 4, and edge types of server and like.

- 745/1069 - 2023 Vesoft Inc.



QUERY BY TEMPLATE

You can select the created nGQL template, and set the parameter value.



- When the returned result is vertices, they will be displayed on the canvas.
- $\bullet$  When the returned result is not vertices, they will be displayed in table format. For example, return player name, age, etc.

For more information, see nGQL template.

### 16.5.3 Vertex Filter

The Vertex Filter helps you filter the vertices and edges displayed on the canvas. You can filter data by tag only or by one or more sets of filter conditions

#### Prerequisite

Make sure that there are vertices on the canvas. For more information, see Start query.

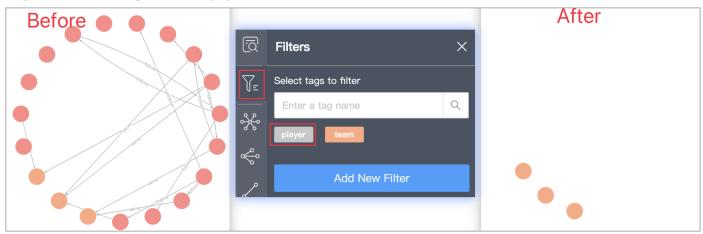
#### **Notes**

- When filtering vertices and associated edges by Tag:
- All the tags in the graph space are displayed on the **Filters** panel.
- The selected tag turns gray, and the vertices and associated edges of the corresponding tag are hidden.
- For multi-tag vertices, if any of its tags is selected, the vertices are hidden.
- You can enter a tag name in the search box to search for tags.
- When filtering vertices and associated edges by filter conditions.
- Each set of filter conditions is only for the data with the target tag. The filtering conditions include Tag, Property, Operator, and Value. If the conditions are met, and the corresponding vertices will be automatically selected. If the conditions are not met, the corresponding vertices can be set to be **hidden** or **turning gray**. The vertices with other tags are not affected.
- The filtering priority by **Tag** is the highest. If the filter conditions include a selected tag (in gray), the corresponding data will not be displayed on the canvas.
- Each time you perform **Vertex Filter**, only one tag can be selected. If you want to filter data based on more tags, conduct **Add New Filter** multiple times.
- The same tag cannot be filtered multiple times. Only the result of the first filtering is displayed.

# Example

EXAMPLE 1 FILTER VERTICES ON THE CANVAS WITH THE TAG PLAYER

- 1. In the left navigation bar, click Vertex Filter  $\sqrt{\epsilon}$ .
- 2. On the Filters panel, click player.
- 3. Only vertices with the tag team are displayed on the canvas.



The orange vertices filtered out in the above figure are the vertices with the tag team.

### EXAMPLE 2 FILTER PLAYERS OLDER THAN 33 YEARS OLD

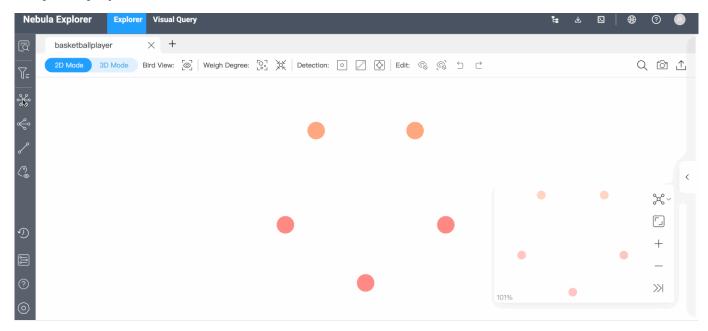
- 1. In the left navigation bar, click **Vertex Filter**  $\sqrt{\epsilon}$ .
- 2. Click **Add New Filter**, and set filter conditions (The values in the example are player, age, >, and age).
- 3. Click  ${f Grayscale}$  to gray the vertices that do not meet the filter conditions.
- 4. Turn on the **Apply Filter** button.



# 16.5.4 Graph exploration

The graph exploration can be performed from the following four aspects:

- Expand
- Common Neighbor
- · Search for Path
- Inspect Property



### Prerequisite

Make sure that there are vertices on the canvas. For more information, see Start querying.

# Expand

1. In the navigation bar on the left side of the page, click to open the **Expand** panel. You can set expansion conditions on the panel, including edge type, direction, vertex style, steps or filter, as described below.

Parameter	Description
Edge type	All edges in the graph space are displayed and selected by default.
Direction	Define the edge direction for the selected vertices, including $\mbox{Outgoing}$ , $\mbox{Incoming}$ , and $\mbox{Bidirect}$ .
Vertex Style	Group by vertex tag: The target vertices are displayed in the same color as the corresponding tag. Custom Style: You can customize the color of the target vertices.
Steps	Single: Customize the number of steps from the selected vertex to the target vertex.  Range: Customize the step range from the selected vertex to the target vertex.
Filter	Query target vertices by filtering conditions.

2. Select the vertex you want to expand, either by holding down the right mouse to select or by holding down the Shift key and clicking on multiple vertexes on the canvas, and then click the Expand button in the **Expand** panel. For a single vertex, you can double-click the left mouse on the vertex to expand.



The system saves the current configurations on the panel. When you double-click or right-click on a vertex for exploration, the exploration will be performed based on the saved configurations.

#### **Common Neighbor**

In the navigation bar on the left side of the page, click to open the **Common Neighbor** panel. You can select two or more vertices either by holding down the right mouse to select or by holding down the Shift key and clicking on multiple vertexes on the canvas and query their common neighbors. When the selected vertices have no common neighbor, the default returns \*\*There is no data.

#### Search for Path

In the navigation bar on the left side of the page, click to open the **Search Path** panel. You can set the edge type, direction, query type or filter, as described below.

Parameter	Description
Edge Type	All edges in the graph space are displayed and selected by default.
Direction	Define the edge direction for the selected vertices, including Outgoing, Incoming, and Bidirect.
Query Type	All path: Request for vertices and edges in all paths from the source vertex to the destination vertex.  Shortest Path: Request for vertices and edges in the shortest path from the source vertex to the destination vertex.  NoLoop Path: Request for vertices and edges in non-loop paths from the source vertex to the destination vertex.
Steps	Customize the number of steps from the source vertex to the destination vertex.
Filter	Query target vertices by filtering conditions.

2. Hold down the Shift key and left-click to select two vertexes on the canvas. The first selected vertex is the source and the second is the destination vertex by default. Then click **Find Path** in the **Search Path** window.

### Inspect Property

In the navigation bar on the left side of the page, click to open the **Inspect Property** panel. Properties of vertices or edges can be hidden or displayed on the canvas.



- Vertex properties are displayed on the canvas only when the zoom ratio is greater than 90%, and properties are automatically hidden when the zoom ratio is less than 90%.
- Edge properties are displayed on the canvas only when the zoom ratio is greater than 100%, and properties are automatically hidden when the zoom ratio is less than 100%.

# 16.5.5 Graph computing

To better mine and analyze the graph data, users can perform graph computing based on the vertexes and edges in the canvas and view the graph computing results directly.



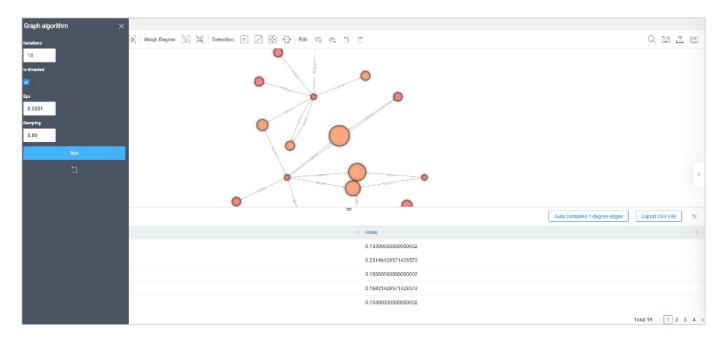
This function only performs graph computing for existing vertexes in the canvas. If you need to perform complex graph computing, it is recommended to use Workflow to perform complex visual graph computing.

## Prerequisites

Ensure the canvas has the vertex and edge data needed for the graph calculation. For details, see Start querying.

# Steps

- 1. In the navigation bar on the left side of the page, click  $^{\textcircled{5}}$  button to open **Graph algorithm** panel.
- 2. Select the algorithm and set related parameters. For more Information about algorithm and parameter, see Algorithm overview •
- 3. Click Run and the result pops up from below the canvas.
- 4. On the result page, you can do the following operations:
- Click **Auto complete 1-degree edges** to completes the one-degree path relationship between all vertexes in the canvas.
- Click **Export CSV File** to download the graph computing result file in CSV format.



# 16.5.6 Property calculation

When there are a large number of vertices in the canvas, to enhance the readability and analyzability of the graph, edges with the same start vertex, end vertex and edge type can be aggregated. The aggregated edges can be computed and displayed based on their properties.

# Prerequisites

There were aggregated edges on the canvas.

#### **Precautions**

- Currently, only summation is supported.
- Only properties of type INT can be aggregated.
- Users can select multiple Edge types for aggregation separately.
- Users can select multiple properties for aggregation separately.
- An edge can display only one aggregation result. You can hover over the aggregated edge to see all the results.

### Steps

METHOD 1

1.

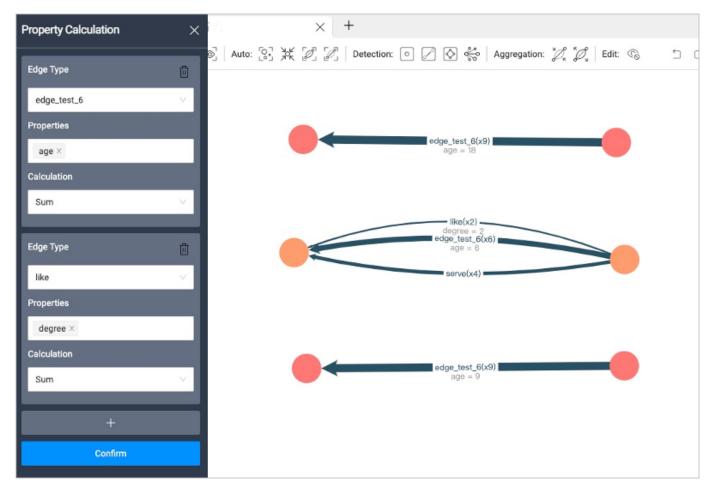


In the left navigation bar, click

to open the **Property Calculation** panel.

- 2. Click + and set the edge type, properties and calculation. You can select multiple attributes to be aggregated separately.
- 3. Click Confirm •

Click ullet to add more edge types for property calculation.



### METHOD 2

- 1. Right-click the aggregated edge on the canvas and select **Property Calculation**.
- $2. \ \mbox{Set}$  the properties and calculation.
- 3. Click **Confirm**.

# 16.6 Visual Query

The Visual Query feature uses a visual representation to express related requests. It allows you to create query scenarios to look up the desired data and view the corresponding statements. You can construct visual query statements by simply dragging and dropping, and then the system displays the query results on the query panel.



The Visual Query feature is not compatible with NebulaGraph versions below 3.0.0.

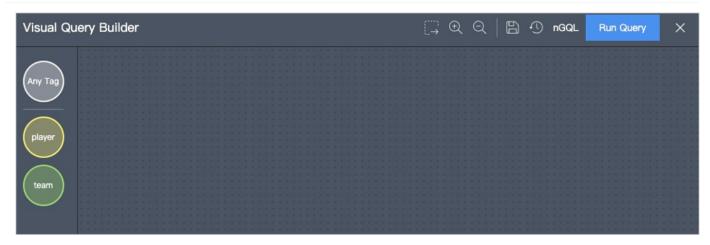


Currently, the Visual Query feature is still in beta.

# 16.6.1 Prerequisite

- You have choosen a graph space. For details, see Choose graph spaces.
- You have created indexes for particular queries. For details, see MATCH precautions and CREATE INDEX.

# 16.6.2 Page elements



At the top of the Explorer page, click **Visual Query** to enter the visual query page. On the left side of the **Visual Query** page, all the Tag(s) corresponding to the graph space (e.g.player and team) and the Tag named **Any Tag** are displayed. You can query vertices without tags by the Tag named **Any Tag**.



Any Tag can also be used to query the vertex without tags.

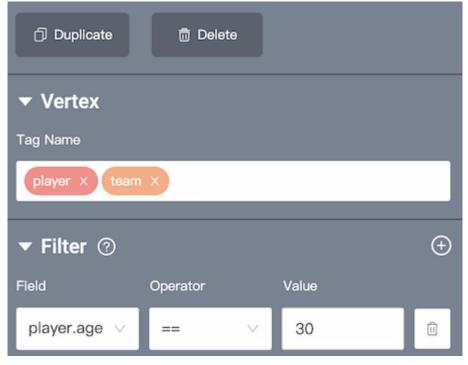
- 755/1069 - 2023 Vesoft Inc.

On the page, the descriptions of other icons are as follows.

Icon/ Element	Description
$[\rightarrow]$	The selected vertices and edges are the results to be queried. Double-click on the query pattern frame to limit the number of queries (with a priority higher than the value of the maximum number of returns in the global settings). Only querying edges is not supported.
$\oplus$	Zoom in on the query page.
$\bigcirc$	Zoom out on the query page.
	Save the current query graph. The saved graph is cached in the browser.
<b>(</b> )	View all stored query graphs. Up to 10 recently saved visual graphs are displayed. Click any of the stored graphs to display them on the visual query page.
nGQL	Click $\mathbf{nGQL}$ to view the statement corresponding to the query pattern.
Run Query	Click <b>Run Query</b> to display the query results visually on the canvas.

# 16.6.3 Steps

- 1. Drag several target tags from the left side of the **Visual Query** page to the canvas to create the corresponding vertices.
- 2. Click a vertex, hold down the left mouse button on the anchor point at the edge of the vertex, and drag it to another vertex to create the corresponding edge.
- 3. Set a vertex by clicking it. The descriptions of configuration options are as follows.



• Tag Name: Set zero, one, or multiple tags.

#### Q Note

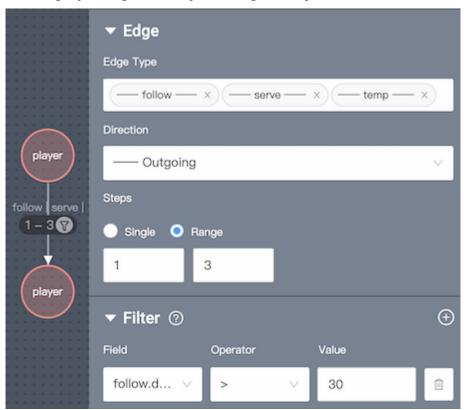
One vertex can have zero or multiple tags:

- When 0 tag is set, query the vertex without tags.
- When 1 tag is set, query the vertex with that tag.
- When multiple tags are set, query the vertex that has all the tags you set.
- Filter: Add one or more sets of filter conditions, including vertex properties, operators, and property values.

# Note

When setting multiple tags in the Tag Name dialog box, it is not supported to set filter conditions to query data.

4. Set an edge by clicking. The descriptions configuration options are as follows.



• Edge Type: Set one or multiple edge types.



One edge have one and only one edge type:

- When one edge type is set, query the edge with that edge type.
- When multiple edge types are set, query the edge that has any of the edge types you set.
- Direction: Set the edge direction between two vertices, including Outgoing, Incoming, and Bidirect.
- Single: Set a fixed-length path.
- Range: Set a variable-length.
- Filter: Add one or more sets of filter conditions, including edge properties, operators, and property values.

- 757/1069 - 2023 Vesoft Inc.

#### O Note

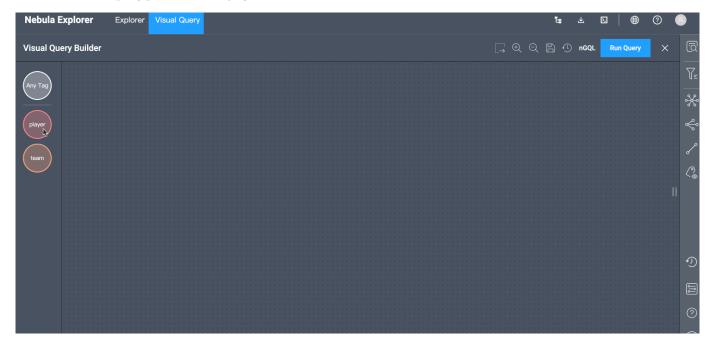
When setting multiple edge types in the Edge Type dialog box, it is not supported to set filter conditions to query data.

- 5. After the query scenarios (pattern) is created, click and select the result you want to return.
- 6. Click Run Query on the upper right corner of the Visual Query page to display the query results on the canvas.

# 16.6.4 Examples

### Example 1

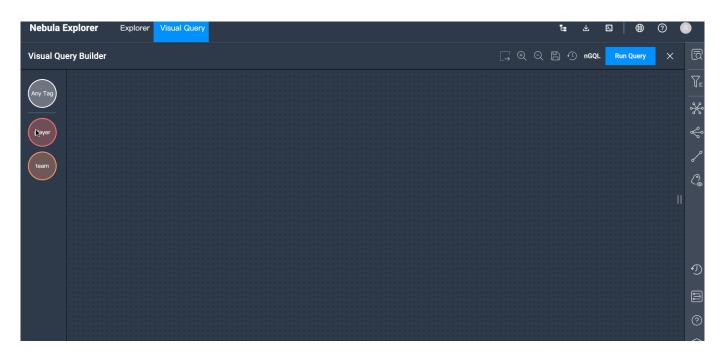
Find out players who follow each other with Yao Ming and older than 35, and which teams these players are loyal to, and limit the number of the query patterns of the players and teams to 6.



- 1. Create a query pattern by dragging and dropping Tags to the panel (2 players and 1 team).
- 2. Configure filter conditions.
- a. Set the filter condition for the first player to  $\mbox{ player.name} = \mbox{Yao Ming} \, .$
- b. Set the edge type of the edge between the first and second vertices of the tag player to follow, set the direction to Bidirect, and the steps to 1.
- c. Set the filter condition for the second player to player.age > 35.
- d. Set the edge type of the edge between the second player and the team to serve, the direction to Outgoing, and the steps to 1.
- e.  $\subset$  Click  $\longrightarrow$  to select the second player, the team, and the serve edge between them.
- f. Click the  $\bf Query~\bf Pattern$  frame, and set the  $\bf Limit~\bf Number$  to  $\ \ \bf 6$  .
- 3. Click  ${\bf Run\ Query},$  and the system displays 6 query patterns on the canvas.

### Example 2

Find out what teams two mutually-following players are loyal to and query for all players on that team who are older than 30.



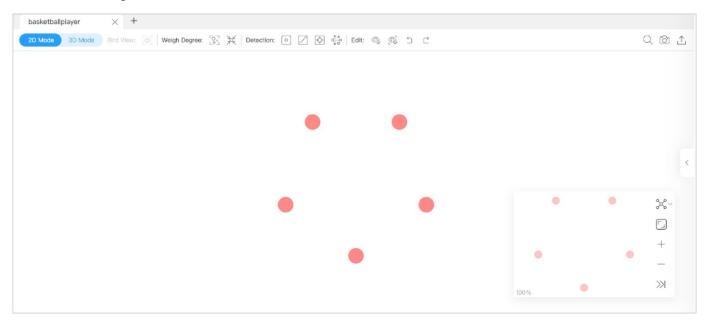
- 1. Create a query pattern by dragging and dropping Tags to the panel (3 players and 1 team).
- 2. Configure filter conditions.
- a. Set the edge type of the edge between the first and second players to follow, set the direction to Bidirect, and the steps to 1.
- b. Set the edge type of the edge between the first player and the team to serve, the direction to Outgoing, and the steps to 1.
- c. Set the edge type of the edge between the second player and the team to serve, the direction to Outgoing, and the steps to 1.
- d. Set the filter conditions for the third player to player.age > 30.
- e. Set the edge type of the edge between the third player and the team to serve, the direction to Outgoing, and the steps to 1.
- Click  $\longrightarrow$  to select the third player, the team, and the serve edge between them.
- 3. Click Run Query.

# 16.7 Canvas

### 16.7.1 Canvas overview

You can visually explore data on a canvas. This topic introduces the composition of a canvas and its related functions.

Canvas overview diagram:



### Tabs on the Top

Click the plus sign to add a new canvas. You can have operations on multiple canvases simultaneously.



- Canvas data on different canvases can come from the same graph space or from different graph spaces.
- $\bullet$  You can customize the name of a canvas except for the canvas in the left-most tab.

### Visualization modes

Graph data can be visually explored in 2D mode and 3D mode. For more information, Visualization modes.

### Data storage

Graph data on the current canvas can be stored by creating snapshots or exporting canvas data as images or CSV files.

- 760/1069 - 2023 Vesoft Inc.

At the top right of the page, you can:

- $^{\bullet}$  Click  $^{\fill \fill \fill}$  to create a snapshot. For more information, see Canvas snapshots.
- Click and then click **Export CSV File** to store canvas data as CSV files.
- · Click  $\stackrel{f L}{\Box}$  and then click **Export PNG File** to store canvas data as images.

### Search box

# Layouts

Explorer provides 6 layouts to show the relationship between the data on a canvas.

Force	Dagre	Circular	Grid	Neural Network	Radial
×	8	\$	<b>%</b>	<b>%</b>	38/6
fraud_detection	× +				
V=	Mode Bird View: [6] We	eigh Degree: O Detection:	○ Ø Edit: ⊗	<u>%</u> 5 C	Q 🖆 <u>1</u>
* *					
\$**					
₹		P I I I I I I I I I I I I I I I I I I I			<
		hair and hai	Cash in dest		. »»
<b>D</b>		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		•	
		9000	500		. +
Selected Vertices	101 Selected Edges 0 ^			101%	· »

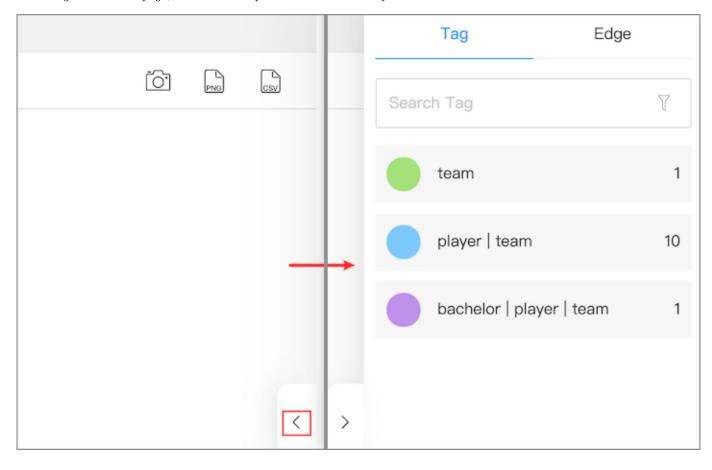
# Minimap

You can display the vertices on a canvas on full screen. You can also collapse the minimap, zoom in or zoom out the canvass, etc. The percentage of a canvas graph to the total is displayed in the lower-left corner of the minimap.



# Data overview

On the right side of the page, click to expand the data overview panel.



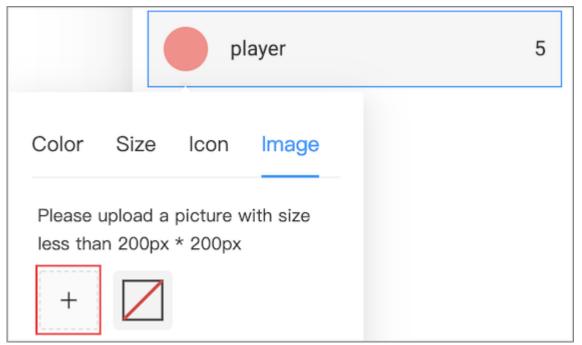
On the data overview panel, you are enabled to:

- See the number of tags and edge types, and the number of the corresponding vertices and edges on a canvas.
- Click the color icon of the tags or edge types to customize the color and size. You can also customize the icons and images of the tags.

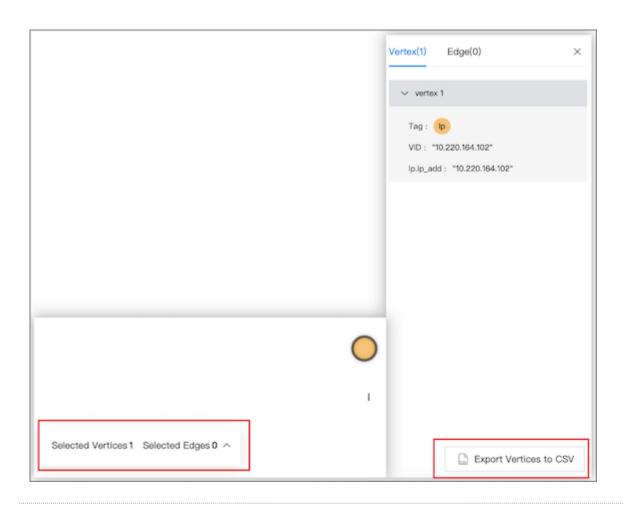


You can only change colors in batches in the data overview panel. Right-click a single vertex on a canvas to manually modify the style of the vertex.

• Upload images to personalize the style of the vertices in the canvas, and the uploaded images are stored in the browser. To store uploaded images permanently, save the canvas data as a snapshot. For details, see Manage snapshots.



Select vertices and edges on the canvas, and then click **Selected Vertices (number) Selected Edges (number)** in the lower left corner to view the detailed information of the vertices and edges. You can export the data as a CSV file.

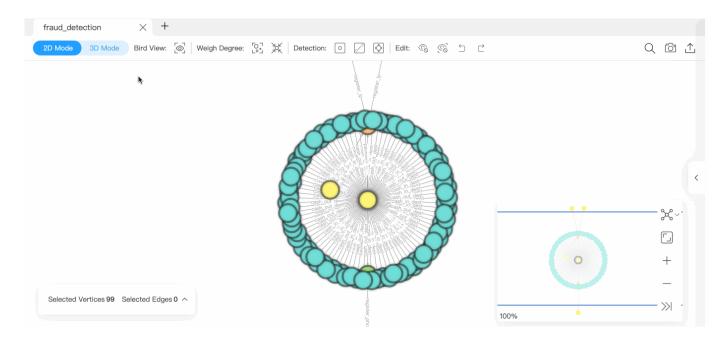


# 16.7.2 Visualization modes

Explorer provides  $\mathbf{2D}$  and  $\mathbf{3D}$  visualization modes for you to explore data. 2D enables you to operate on graph data and view data information. 3D lets you explore graph data from a different perspective. The 3D is suitable for cases with a large amount of data or situations requiring presentations.

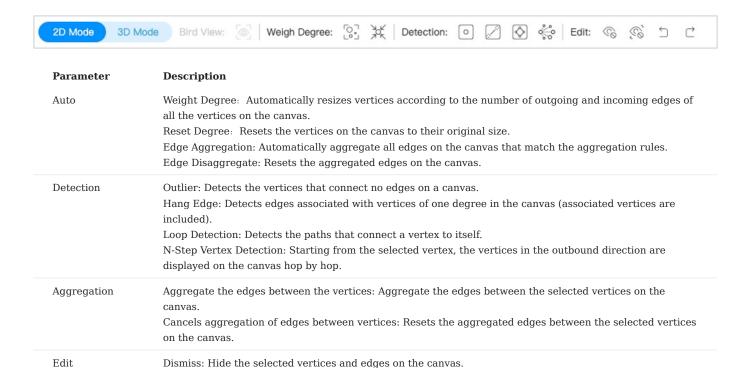
#### Q Note

In 3D mode, operations on graph data are unavailable.



# 2D mode

Exploration of the data on a canvas is possible in 2D mode.





It is recommended to limit the number of vertices and edges displayed on the canvas to no more than 5000 in 2D mode, otherwise, it may stuck when rendering.

Dismiss Others: Hide the unselected vertices and edges on the canvas.

For more information about the operations available in 2D mode, see Canvas.

Undo: Undo the action in the previous step.

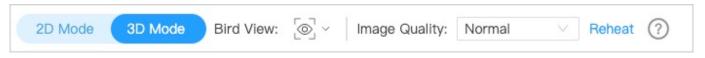
Redo: Restore the action that was previously undone.

### EDGE AGGREGATION DESCRIPTION

When there are a large number of vertices in the canvas, to enhance the readability and analyzability of the graph, edges with the same start vertex, end vertex and edge type can be aggregated to make the relationship between vertices clearer.

- Edge aggregation automatically displays the number of aggregated edges.
- Edge aggregation supports the calculation of properties in it. For details, see Property calculation.
- Hovering over the aggregated edge displays the edge type, the number of aggregated edges, edge properties, and property values. If the property calculation was performed, the result is also displayed.
- In addition to canceling edge aggregation in the upper bar, you can also double-click the aggregated edge or right-click the aggregated edge and select **disaggregate**.

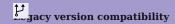
# 3D mode



- 766/1069 - 2023 Vesoft Inc.

At the top left of the page, toggle the view button to switch to 3D mode. 3D mode allows you to switch back to 2D mode and does not influence operations in 2D.

Parameter	Description
Bird View	Shows a bird view of all the data in the current graph space. By default, displays data for up to 20,000
	vertices and 2,000 edges in the current graph space. Click $\begin{tabular}{c} \begin{tabular}{c} ta$
Image Quality	High: Vertices are displayed in the form of balls with better light and shadow effects.  Normal: Vertices are represented in a circle format and support a large amount of data.
Reheat	Disperses the distance between vertices when the vertices overlap.



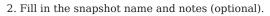
For versions of NebulaGraph below 3.0.0, you need to create an index before using the Bird View feature. For more information, see Create an index.

### 16.7.3 Canvas snapshots

Explorer provides a snapshot feature that lets you store the visualized canvas data so that the data can be restored when your browser is opened again.

### Create snapshots

1. In the upper right corner of a canvas page, click the camera icon  $\Box$ 



### 3. Click submit.



Created snapshots are stored on the snapshot list page. For more information, see below.

### Historical snapshots



Up to 50 snapshots can be stored in the snapshot list currently.

In the left navigation bar of the Explorer page, click to enter the Snapshot page. You can switch graph spaces and view the historical snapshots of the corresponding graph space. You can also import snapshots to a canvas, download canvas snapshots to your local drive, and delete snapshots.

Under the **Operation** column to the right of the target snapshot, you are enabled to:

- · Click to import a historical snapshot to a new canvas.
- Click to download a snapshot in JSON format locally.
- · Click to delete a snapshot.

At the top left of the **Snapshot** page, click **Import Snapshot** to import previously downloaded files in JSON format to the **Snapshot** page for sharing the snapshot data offline. The system automatically places the imported snapshots in the corresponding graph space based on the graph space information recorded in the JSON file.

Last update: February 19, 2024

- 768/1069 - 2023 Vesoft Inc.

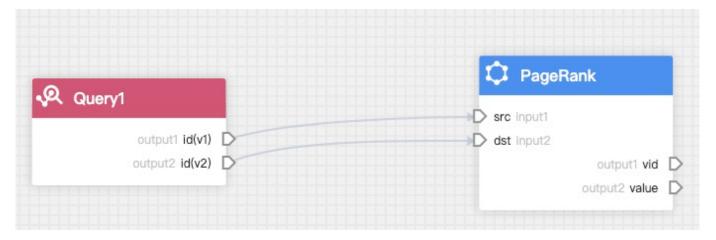
# 16.8 Workflow

### 16.8.1 Workflow overview

NebulaGraph Explorer supports visual and complex graph computing with custom workflows.

### **Background**

NebulaGraph Explorer provides multiple components, including graph query and graph computing components. Users can combine these components based on the scheduling tool Dag Controller for free. For example, using the output of a graph query component as an input to a graph computing component. The whole process is a directed acyclic workflow.



Instantiate the workflow when performing graph computing. The instantiated component is called **task**, and the instantiated workflow is called **job**. A job can consist of multiple tasks. The NebulaGraph Explorer sends the job to NebulaGraph Analytics for graph computing, and you can view the result in the job list.

### Features

- Add, view, modify, delete, compare, clone and rename workflows.
- A workflow supports one query component and multiple graph computing components. You can search for, add, configure, and rename component.
- View the lists, progresses, results and logs of the jobs, and rerun jobs.
- Search for workflows or jobs.

### Precautions

• The **Workflow** page can be displayed only when **Workflow** is enabled in



- Additional deployment of the Dag Controller and the NebulaGraph Analytics is required to use a workflow. For details, see NebulaGraph Analytics and Deploy Explorer.
- The input to the graph query component can only be the nGQL.
- The results of a graph query component can only be stored in the HDFS, which is convenient to be called by multiple algorithms.
- The input to the graph computing component can be the specified data in the NebulaGraph or HDFS, or can depend on the results of the graph query component. If an input depends on the results of the previous graph query component, the graph computing component must be fully connected to the graph query component, that is, the white output anchors of the previous graph query component are all connected to the white input anchors of the graph compute component.
- The parameters of some algorithms can also depend on the upstream components.
- The result of the graph computing components can be stored in the NebulaGraph or HDFS, but not all algorithm results are suitable to be stored in NebulaGraph. Some algorithms can only be saved in HDFS when configuring the save results page.

### Algorithm description

See Algorithm description.

Last update: February 19, 2024

- 770/1069 - 2023 Vesoft Inc.

# 16.8.2 Prepare resources

You must prepare your environment for running a workflow, including NebulaGraph configurations, HDFS configurations, and NebulaGraph Analytics configurations.

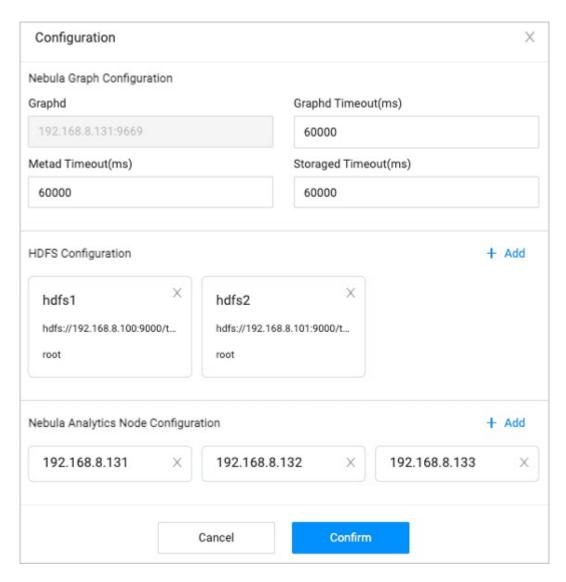
# Prerequisites

- NebulaGraph Analytics 3.4.0 or later have been deployed. For details, see NebulaGraph Analytics.
- Dag Controller have been deployed and started. For details, see Deploy Explorer.

### Steps

- 1. At the top of the Explorer page, click  $\mathbf{Workflow}$ .
- 2. In the **Workflows** tab, click **Configuration**.
- 3. Configure the following resources:

- 771/1069 - 2023 Vesoft Inc.



Туре	Description
NebulaGraph Configuration	The access address of the graph service that executes a graph query or to which the graph computing result is written. The default address is the address that you use to log into Explorer and can not be changed. You can set timeout periods for three services.
HDFS Configuration	The HDFS address that stores the result of the graph query or graph computing. Click <b>Add</b> to add a new address, you can set the HDFS name, HDFS path (fs.defaultFS), and HDFS username. You can configure the save path, such as hdfs://192.168.8.100:9000/test. The configuration takes effect only after the HDFS client is installed on the machine where the Analytics is installed.
NebulaGraph Analytics Configuration	The NebulaGraph Analytics address that performs the graph computing. Click $\mathbf{Add}$ to add a new address.

# 4. Click **Confirm**.

Last update: February 19, 2024

- 772/1069 - 2023 Vesoft Inc.

# 16.8.3 Workflow example

This topic describes how to create a simple workflow.

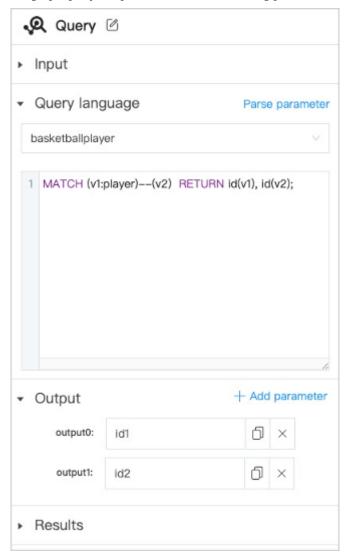
# Prerequisites

- $\bullet$  The data source is ready. The data source can be data in NebulaGraph or CSV files on HDFS.
- The resource has been configured.

# Add workflow

With the result of the MATCH statement MATCH (v1:player)--(v2) RETURN id(v1), id(v2); as the input of the PageRank algorithm, the following will introduce how to create a simple workflow.

- $_{
  m 1.}$  At the top of the Explorer page, click **Workflow**.
- 2. In the Workflows tab, click New workflow to enter the process canvas page.
- 3. In the component library list on the left side of the process canvas page, select **Query->Query** and drag it onto the canvas. Click the graph query component and set the following parameters in the configuration panel on the right side.

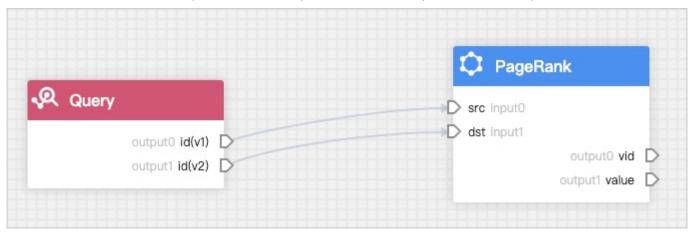


Parameters	Description
Query	Click to modify the component name to identify the component.
Input	Set custom parameters that can be used for parameterized query. Click <b>Add parameter</b> to add more custom parameters.
Query language	Select the graph space to execute the nGQL statement and fill in the nGQL statement. Click <b>Parse Parameter</b> to display the returned column name in the <b>Output</b> .
Output	The column name returned by parsing the query language. You can change the name, which is equivalent to aliasing the column with $\mbox{AS}$ .
Results	Set the saving project of the result. To call the results expediently for other algorithms, the results of the graph query component can only be saved in the HDFS.

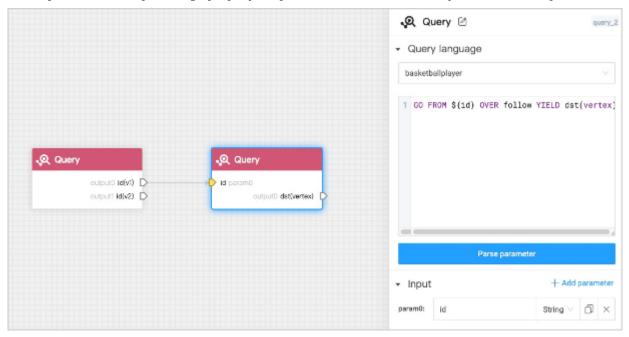
#### Q Note

The connection anchors are shown in yellow, indicating that it is optional and can be set by user or provided by any other component.

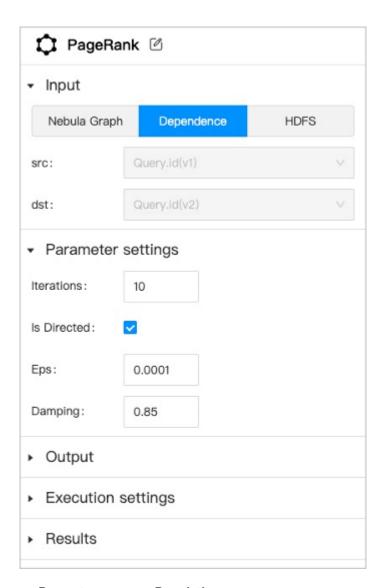
4. In the component library list on the left side of the process canvas page, select **Node importance->PageRank** and drag it onto the canvas. Connect the anchor output0 to the anchor input0 and the anchor output1 to the anchor input1.



If you use multiple graph query components in series, you need to add parameterized text. For example, if you fill in the statement GO FROM \${id} OVER follow YIELD dst(vertex) and click Parse parameter, a yellow anchor for \${id} will appear in the graph query component. The output anchor of the previous graph query component can be connected to the yellow anchor as input.



 $5. \ Click \ the \ graph \ computing \ component \ and \ set \ the \ following \ parameters \ in \ the \ configuration \ panel \ on \ the \ right \ side.$ 



Parameters	Description
PageRank	Click to modify the component name to identify the component.
Input	Three data sources are supported as input.  NebulaGraph: Users must select one graph space and corresponding edge types.  Dependence: The system will automatically recognize the data source according to the connection of the anchor.  HDFS: Users must select HDFS and fill in the relative path of the data source file.
Parameter settings	Set the parameters of the graph algorithm. The parameters of different algorithms are different. Some parameters can be obtained from any upstream component where the anchor are shown in yellow.
Output	Display the column name of the graph computing results. The name can not be modified.
Execution settings	Machine num: The number of machines executing the algorithm.  Processes: The total number of processes executing the algorithm. Allocate these processes equally to each machine based on the number of machines.  Threads: How many threads are started per process.
Results	Set the restoration path of the results in HDFS or NebulaGraph.  HDFS: The save path is automatically generated based on the job and task ID.  NebulaGraph: Tags need to be created beforehand in the corresponding graph space to store the results.  For more information about the properties of the tag, see Algorithm overview.  Some algorithms can only be saved in the HDFS.

6.	Click	next to the automatically generated workflow name at the upper left corner of the canvas page to modify the workflow
		click <b>Run</b> at the upper right corner of the canvas page. The job page is automatically displayed to show the job
	progress.	You can view the result after the job is completed. For details, see Job management.

Q Note

When you click **Run**, the workflow will be automatically saved. If you do not perform graph computing and only make modifications, click to save the modification, or click to save the workflow as a new workflow.

Last update: February 19, 2024

- 778/1069 - 2023 Vesoft Inc.

# 16.8.4 Workflow management

This topic describes how to manage workflows, including view, modify, rename, clone, delete, and compare workflows.

### Steps

- 1. At the top of the Explorer page, click **Workflow**.
- 2. In the Workflows tab, users can view all saved workflows. The list displays Workflow name, Created time, Update time, and Algorithm.
- At the top of the list page, click **Comparison** and select two workflows or different historical versions of the same workflow for code comparison.
- At the top of the list page, users can search the workflow by keywords in the search box.
- In the **Operation** column of the list page, users an perform the following operations:
- Run: Instantiate the workflow directly as a job and execute the job.
- Open: Open a workflow to view and modify the workflow. After modifying the workflow, click to save the modification or click to save the workflow as a new workflow.
- View jobs: Jump to the job list to view all the jobs instantiated by this workflow.
- : Users can view the workflow code, rename the workflow, clone the workflow, and delete the workflow.

Last update: February 19, 2024

- 779/1069 - 2023 Vesoft Inc.

# 16.8.5 Job management

This topic describes how to view the lists, progresses, results, logs of the jobs and rerun jobs.

### Steps

- 1. At the top of the Explorer page, click **Workflow**.
- 2. In the Jobs tab, users can view all the jobs. The page displays Job ID, Job name, Status, CREATE time, End time and Workflow version.
- At the top of the list page, click **Comparison** and select two workflows or different jobs of the same workflow for code comparison.
- At the top of the list page, users can filter the workflow and version in the filter box.
- At the top of the list page, users can search the job by keywords in the search box.
- In the **Operation** column of the list page, users an perform the following operations:
- View in Explorer: For successfully executed jobs, users can select the graph space and the component to view the output of the component. Users can export the results to a CSV file.
- Rerun: For failed executed jobs, users can rerun the job.
- **Open**: Users can rerun the job and view the results and logs of the job. Users can also jump to the corresponding workflow for editing (the workflow is the latest version).

Last update: February 19, 2024

- 780/1069 - 2023 Vesoft Inc.

### 16.8.6 Workflow API

### Workflow API overview

NebulaGraph Explorer provides some APIs for using workflow.

The supported APIs are as follows:

- Add a new job
- Get a list of all jobs
- Get a list of jobs for a specified workflow
- · Query details for a specified job
- Cancel a running job
- Get the result data of a specified task

#### REQUEST METHOD

Users can use curl to call APIs to achieve corresponding functions.

The format is as follows:

```
curl <options> http://<explorer_address>:<explorer_port>/<api_path>?{<body>}
```

- <explorer\_address> : The access address of the NebulaGraph Explorer.
- <explorer\_port> : The access port of the NebulaGraph Explorer.
- <api\_path> : The call path of APIs. For example: api-open/v1/jobs .
- <body>: The body parameters that needs to be supplied when calling APIs.

GET AUTHORIZATION TOKEN

Token information verification is required when calling an API. Run the following command to get the authorization token.

```
curl -i -X POST -H "Content-Type: application/json" -H "Authorization: Bearer <account_base64_encode>" -d '{"address":"<nebula_address>","port":<nebula_port>}' http://
<explorer_address>:<explorer_port>/api-open/v1/connect
```

- <account\_base64\_encode>: The character string of the base64 encoded NebulaGraph account and password. Take the username root and password 123 as an example, the serialized string is ["root", "123"]. After the encoding, the result is WyJyb290IiwiMTIzILO=.
- <nebula\_address> : The access address of the NebulaGraph.
- <nebula\_port> : The access port of the NebulaGraph.
- <explorer\_port> : The access port of the NebulaGraph Explorer.

### Example:

```
curl -i -X POST -H "Content-Type: application/json" -H "Authorization: Bearer WyJyb290IwiMTIzIlo=" -d '{"address":"192.168.8.111","port":9669}' http://192.168.8.145:7002/api-open/v1/connect
```

### Response:

```
HTTP/1.1 200 OK
Content-Type: application/json
Set-Cookie: explorer_token=eyJhbxxx; Path=/; # Max-Age=259200; HttpOnly
Traceparent: 00-1c3f55cdbf8le13a233led88155ce0bf-2b97474943563f20-# 00
Date: Thu, 14 Jul 2022 06:47:01 GMT
Content-Length: 54
{
```

```
"code": 0,
    "data": {
        "success": true
},
    "message": "Success"
}
```

Note the following parameters:

- explorer\_token: The authorization token.
- Max-Age: Token validity time. Unit: second. The default value is 259,200 seconds, that is 3 days. You can change the default validity time in the config/app-config.yaml file in the installation directory.

### RESPONSE

• If an API is called successfully, the system returns the following information:

```
{
    code: 0,
    message: 'Success',
    data: <ResponseData> //Return the results based on the API.
}
```

• If an API is called failed, the system returns the corresponding common error code. For example:

```
{
    code: 40004000,
    message: '<ErrBadRequest>', //Display the error information.
}
```

For descriptions of common error codes, see the following sections.

#### Common error codes

Error code	Information	Description
40004000	ErrBadRequest	Request error.
40004001	ErrParam	Request parameter error.
40104000	ErrUnauthorized	Request authorization error.
40104001	ErrSession	Login session error.
40304000	ErrForbidden	Request denied.
40404000	ErrNotFound	Requested resource does not exist.
50004000	ErrInternalServer	Internal service error.
50004001	ErrInternalDatabase	Database error.
50004002	ErrInternalController	Controller error.
50004003	ErrInternalLicense	Certificate verification error.
90004000	ErrUnknown	Unknown error.

- 782/1069 - 2023 Vesoft Inc.

### Job/Task status code

Status code	Description
0	Preparing
1	Running
2	Success
3	Failed
4	Interrupted
5	Stopping

# Add a new job

This topic describes how to use an API to add a new job.

API PATH

api-open/v1/workflows/<workflow\_id>/jobs

<workflow\_id> : The workflow ID. See request parameters below.

REQUEST PARAMETERS

Path parameters

Parameters	Туре	If required	Default value	Example	Description
workflow_id	number	yes	-	4216617528	The workflow ID. The system instantiates a specified workflow as a job. The ID can be viewed in the upper left corner of the specified workflow page.

### Headers parameters

Para	meters	3 1	If required	Default value	Example	Description
Conter	nt-Type	string	yes	-	application/ json	The content type.
explor	rer_token	string	yes	-		The authorization token that is used to verify account information. For details, see Workflow API overview.

Body parameters



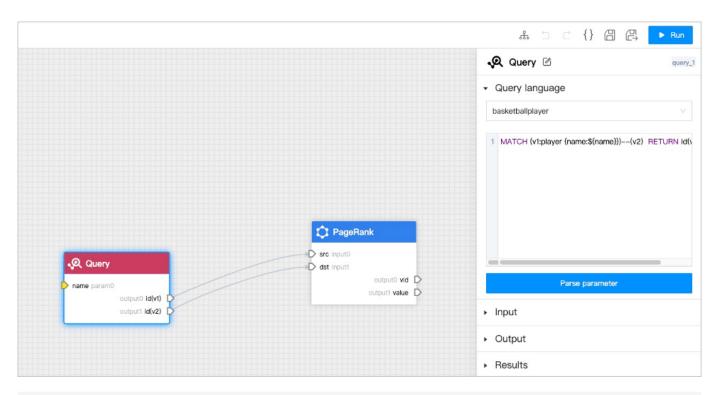
Users must ensure the rationality and correctness of the user-defined input parameters. Otherwise, the operation will fail.

Parameters	Туре	If required	Default value	Example	Description
input	object	no	-	-	The user-defined input parameters.
- task_id	object	no	-	query_1	The task ID. Users can view the ID in the upper right corner of the component settings page. A task can set multiple parameters represented by key-value pairs.
- param_name: param_value	string: {string or number}	no	-	param0: player100	param_name is the parameter key, that is, the parameter name.  param_value is the parameter value.

Request example

The following is an example of using the user-defined input parameter name in an nGQL statement. Pass in the parameter value Tim Duncan when creating a job.

- 784/1069 - 2023 Vesoft Inc.



curl -i -X POST -H "Content-Type: application/json" -H "Cookie: "explorer\_token=eyJhbxxx"" -d '{"input":{"query\_1":{"name":"Tim Duncan"}}}' http://192.168.8.145:7002/api-open/v1/workflows/4216617528/jobs

### RESPONSE PARAMETERS

Parameters	Туре	Example	Description
code	number	0	The result code of the request. Return 0 if the request is successful, and return an error code if the request is unsuccessful. For details, see Workflow API overview.
message	string	Success	The result information of the execution.
data	object	-	The list of returned data.
- id	string	107	The ID of the new job.

# Response example

```
{
    "cookie": [],
    "cortent-Type": "application/json",
    "Traceparent": "00-lbal28615cdc2226c921973a689e9flb-7630b12963494672-00",
    "Date": "Fri, 15 Jul 2022 07:19:25 GMT",
    "Content-Length": "48"
}

{
    "code": 0,
    "data": {
        id": 107
    },
    "message": "Success"
}
```

# Get a list of all jobs

This topic describes how to use an API to get a list of all jobs.

API PATH

api-open/v1/jobs

REQUEST PARAMETERS

Path parameters

None.

Headers parameters

Parameters	Туре	If required	Default value	Example	Description
Content-Type	string	yes	-	application/ json	The content type.
explorer_token	string	yes	-	eyJhbxxx	The authorization token that is used to verify account information. For details, see Workflow API overview.

### Body parameters

Parameters	Туре	If required	Default value	Example	Description
filter	object	no	-	-	The filter settings.
- name	string	no	-	workflow_q745a_20220715092236	The job name.
- status	number	no	-	2	The job status code. For details, see Workflow API overview.
- fromCreateTime	number	no	-	1657848036000	Start time stamp. Filtering based on the job creation time.
- toCreateTime	number	no	-	1657848157000	End time stamp. Filtering based on the job creation time.
- orderByCreateTime	string	no	desc	-	Sorting mode. The available value are desc and asc.
pageSize	number	no	10	-	The number of entries to return on each page.
page	number	no	1	-	The number of the page to return.

### Request example



The content after jobs? is the body parameter, and the content of filter is the result of URL encoding. The original content of filter was { "status": 2, "orderByCreateTime": "asc"}.

```
curl -i -X GET -H "Content-Type: application/json" -H "Cookie: "explorer_token=eyJhbxxxx"" http://192.168.8.145:7002/api-open/v1/jobs?
filter=%7B%20%22status%22%34%20%2C%20%20%20crderByCreateTime%22%3A%20%22asc%22%7D&pageSize=10&pageSi
```

### RESPONSE PARAMETERS

Parameters	Туре	Example	Description		
code	number	0	The result code of the request. Return 0 if the request is successful, and return an error code if the request is unsuccessful. For details, see Workflow API overview.		
message	string	Success	The result information of the execution.		
data	object	-	The list of returned data.		
- total	number	2	The total number of records.		
- Page	number	1	The number of the page to return.		
- PageSize	number	10	The number of entries to return on each page.		
- items	object	-	The list of record details.		
- id	number	105	The job ID.		
- name	string	workflow_q745a_20220715090915	The job name.		
- workflowId	string	4216617528	The workflow ID.		
- workflowName	string	workflow_q745a	The workflow name.		
- status	number	2	The job status code. For details, see Workflow API overview.		
- runBeginTime	number	1657847358000	The start time of the job execution.		
- runEndTime	number	1657847364000	The end time of the job execution.		
- createTime	number	1657847355906	The creation time of the job.		

### Response example

```
"status": 2,
    "runBeginTime": 1657847358000,
    "runEndTime": 1657847358000,
    "createTime": 165784735906
},

{
    "id": 106,
    "name": "workflow_q745a_20220715092236",
    "workflowId": "4216617528",
    "workflowMame": "workflow_q745a",
    "status": 2,
    "runBeginTime": 1657848157000,
    "runEndTime": 1657848150000,
    "createTime": 165784815000
},
    "total": 2,
    "PageSize": 10
},
    "message": "Success"
}
```

# Get a list of jobs for a specified workflow

This topic describes how to use an API to get the list of jobs for a specified workflow.

API PATH

api-open/v1/workflows/<workflow\_id>/jobs

<workflow\_id> : The workflow ID. See request parameters below.

REQUEST PARAMETERS

Path parameters

Parameters	Туре	If required	Default value	Example	Description
workflow_id	number	yes	-	4216617528	The workflow ID. The system instantiates a specified workflow as a job. The ID can be viewed in the upper left corner of the specified workflow page.

# Headers parameters

Parameters	Туре	If required	Default value	Example	Description
Content-Type	string	yes	-	application/ json	The content type.
explorer_token	string	yes	-	eyJhbxxx	The authorization token that is used to verify account information. For details, see Workflow API overview.

- 789/1069 - 2023 Vesoft Inc.

### Body parameters

Parameters	Туре	If required	Default value	Example	Description
filter	object	no	-	-	The filter settings.
- name	string	no	-	workflow_q745a_20220715092236	The job name.
- status	number	no	-	2	The job status code. For details, see Workflow API overview.
- fromCreateTime	number	no	-	1657848036000	Start time stamp. Filtering based on the job creation time.
- toCreateTime	number	no	-	1657848157000	End time stamp. Filtering based on the job creation time.
- orderByCreateTime	string	no	desc	-	Sorting mode. The available value are desc and asc.
pageSize	number	no	10	-	The number of entries to return on each page.
page	number	no	1	-	The number of the page to return.

Request example



The content after jobs? is the body parameter, and the content of filter is the result of URL encoding. The original content of filter was  $\{"status": 2, "fromCreateTime": 1657874100000\}$ .

curl -i -X GET -H "Content-Type: application/json" -H "Cookie: "explorer\_token=eyJhbxxxx"" http://192.168.8.145:7002/api-open/v1/workflows/4216617528/jobs?filter=%7B%22status%22%3A%202%2C%20%20%22fromCreateTime%22%3A%201657874100000%7D&pageSize=10&page=1

- 790/1069 - 2023 Vesoft Inc.

### RESPONSE PARAMETERS

Parameters	Туре	Example	Description
code	number	0	The result code of the request. Return 0 if the request is successful, and return an error code if the request is unsuccessful. For details, see Workflow API overview.
message	string	Success	The result information of the execution.
data	object	-	The list of returned data.
- total	number	2	The total number of records.
- Page	number	1	The number of the page to return.
- PageSize	number	10	The number of entries to return on each page.
- items	object	-	The list of record details.
- id	number	105	The job ID.
- name	string	workflow_q745a_20220715090915	The job name.
- workflowId	string	4216617528	The workflow ID.
- workflowName	string	workflow_q745a	The workflow name.
- status	number	2	The job status code. For details, see Workflow API overview.
- runBeginTime	number	1657847358000	The start time of the job execution.
- runEndTime	number	1657847364000	The end time of the job execution.
- createTime	number	1657847355906	The creation time of the job.

# Response example

# Query details for a specified job

This topic describes how to use an API to query details for a specified job.

API PATH

api-open/v1/jobs/<job\_id>

<job\_id>: The job ID. See request parameters below.

REQUEST PARAMETERS

Path parameters

]	Parameters	Туре	If required	Default value	Example	Description
	job_id	number	yes	-	1964	The job ID. It can be queried through the API Get a list of all jobs or viewed on the job list page.

Headers parameters

Parameters	Туре	If required	Default value	Example	Description
Content-Type	string	yes	-	application/ json	The content type.
explorer_token	string	yes	-	eyJhbxxx	The authorization token that is used to verify account information. For details, see Workflow API overview.

Body parameters

None.

Request example

curl -i -X GET -H "Content-Type: application/json" -H "Cookie: "explorer\_token=eyJhbxxx"" http://192.168.8.145:7002/api-open/v1/jobs/1964

- 792/1069 - 2023 Vesoft Inc.

#### RESPONSE PARAMETERS

Parameters	Туре	Example	Description
code	number	0	The result code of the request. Return 0 if the request is successful, and return an error code if the request is unsuccessful. For details, see Workflow API overview.
message	string	Success	The result information of the execution.
data	object	-	The list of returned data.
- id	number	1964	The job ID.
- name	string	workflow_xkkjf_20220712103332	The job name.
- workflowId	string	3992429968	The workflow ID.
- workflowName	string	workflow_xkkjf	The workflow name.
- status	number	2	The job status code. For details, see Workflow API overview.
- tasks	object	-	The task details.
- (id)	string	f93dea90fc3a11ecac7e6da0662c195b	The task ID.
- name	string	BFS	The task name.
- runBeginTime	datetime	2022-07-12T10:33:35+08:00	The start time of the task execution.
- runEndTime	datetime	2022-07-12T10:33:38+08:00	The end time of the task execution.
- status	number	2	The task status code. For details, see Workflow API overview.

#### Response example

```
{
    "cookie": [],
    "Content-Type": "application/json",
    "Traceparent": "00-3db1rc3fd9e0a4c3824973471523d214-4384705e523dce83-00",
    "Date": "Fri, 15 Jul 2022 09:08:20 GMT",
    "Content-Length": "400"
}
{
    "code": 0,
    "data": {
        "id": 1964,
        "name": "workflow_xkkjf_20220712103332",
        "workflowlare": "sopy2429968",
        "workflowlare": "workflow_xkkjf",
        "stats": 2,
        "tasks": [
        {
            "id": "f93dea90f3allecac7e6da0662c195b",
            "name": "BFS",
            "runBeginTime": "2022-07-12T10:33:38+08:00",
            ""runBeginTime": "2022-07-12T10:33:38+08:00",
            "runBeginTime": 1657593215000,
            "createTime": 1657593215000,
            "createTime": 1657593215005
},
    "message": "Success"
}
```

#### Cancel a running job

This topic describes how to use an API to cancel a running job.

API PATH

api-open/v1/jobs/<job\_id>/cancel

<job\_id>: The job ID. See request parameters below.

REQUEST PARAMETERS

Path parameters

]	Parameters	Туре	If required	Default value	Example	Description
	job_id	number	yes	-	1964	The job ID. It can be queried through the API Get a list of all jobs or viewed on the job list page.

Headers parameters

Parameters	Туре	If required	Default value	Example	Description
Content-Type	string	yes	-	application/x- www-form- urlencoded	The content type.
explorer_token	string	yes	-	eyJhbxxx	The authorization token that is used to verify account information. For details, see Workflow API overview.

Body parameters

None.

Request example

```
curl -i -X PUT -H "Content-Type: application/x-www-form-urlencoded" -H "Cookie: "explorer_token=eyJhbxxxx"" http://192.168.8.145:7002/api-open/v1/jobs/30600/cancel
```

#### RESPONSE PARAMETERS

Parameters	Туре	Example	Description
code	number	0	The result code of the request. Return 0 if the request is successful, and return an error code if the request is unsuccessful. For details, see Workflow API overview.
message	string	Success	The result information of the execution.
data	object	-	The list of returned data.
- success	bool	true	Whether the job was canceled successfully.

#### Response example

```
{
    "cookie": [],
    "Content-Type": "application/json",
    "Traceparent": "00-8b4b47413a211d9b5e0839aadc712052-4a98bae37fe5948a-00",
    "Date": "Mon, 18 Jul 2022 01:45:08 GMT",
    "Content-Length": "54"
}
{
    "code": 0,
    "data": {
```

```
"success": true
},
"message": "Success"
}
```

#### Get the result data of a specified task

This topic describes how to use an API to get the result data of a specified task.

#### API PATH

api-open/v1/jobs/<job\_id>/tasks/<task\_id>/sample\_result

- <job\_id>: The job ID. See request parameters below.
- <task\_id>: The task ID. See request parameters below.

#### REQUEST PARAMETERS

#### Path parameters

Parameters	Туре	If required	Default value	Example	Description
job_id	number	yes	-	29987	The job ID. It can be queried through the API Get a list of all jobs or viewed on the job list page.
task_id	number	yes	-	8c171f70fb6f11ecac7e6da0662c195b	The task ID. It can be queried through the API Query details for a specified job or viewed in the upper right corner of the specified job page by clicking the component.

## Headers parameters

Parameters	Туре	If required	Default value	Example	Description
Content-Type	string	yes	-	application/x- www-form- urlencoded	The content type.
explorer_token	string	yes	-	eyJhbxxx	The authorization token that is used to verify account information. For details, see Workflow API overview.

## Body parameters

Parameters	Туре	If required	Default value	Example	Description
limit	number	yes	10	-	Limit the number of rows to return results.

## Request example

curl -i -X GET -H "Content-Type: application/x-www-form-urlencoded" -H "Cookie: "explorer\_token=eyJhbxxx"" http://192.168.8.145:7002/api-open/v1/jobs/29987/tasks/8c171f70fb6f1lecac7e6da0662c195b/sample\_result?limit=1000

- 796/1069 - 2023 Vesoft Inc.

#### RESPONSE PARAMETERS

Parameters	Туре	Example	Description
code	number	0	The result code of the request. Return 0 if the request is successful, and return an error code if the request is unsuccessful. For details, see Workflow API overview.
message	string	Success	The result information of the execution.
data	object	-	The list of returned data.
- items	list	-	The list of detailed results.
- result	string	"player110","0.150000"	Depending on the algorithm, the result could be 2 or 3 columns.

#### Response example

#### 16.9 Inline frame

NebulaGraph Explorer supports inline frame (iframe), which can be used to embed canvases into third-party pages. This topic describes how to embed a canvas.

#### 16.9.1 Prerequisites

The Explorer has been installed.

#### 16.9.2 Precautions

- Embedded Explorer pages only access the corresponding graph space by default, so some pages and features are not displayed. For example, the upper navigation bar and some left-navigation-bar features are hidden. If you need to access multiple graph spaces, you can embed them separately on multiple pages.
- Language switching is not supported. The default language is Chinese.

#### 16.9.3 Steps

1. Modify the configuration file config/app-config.yaml in the installation directory of Explorer. The following parameters need to be modified.

```
# Uncomment the CertFile and KeyFile parameters.
CertFile: "./config/NebulaGraphExplorer.crt"
KeyFile: "./config/NebulaGraphExplorer.key"

# Modify the value of IframeMode.Enable to true.
IframeMode:
    Enable: true
# You can set the URI whitelist of the window. By default, no URI is restricted.
# Origins:
# - "http://192.168.8.8"
```

2. Use the command opensst in the directory config to generate a self-signed certificate. The following is an example.

```
openssl req -newkey rsa:4096 -x509 -sha512 -days 365 -nodes -subj "/CN=NebulaGraphExplorer.com" -out NebulaGraphExplorer.crt -keyout NebulaGraphExplorer.key
```

- · -newkey: The secret key is automatically generated when a certificate request or self-signed certificate is generated.
- -x509 : Generates a self-signed certificate.
- -sha512 : Specifies the algorithm of the message digest.
- · -days: The number of days that the certificate generated with parameter -x509 is valid.
- -nodes: Outputs the secret key without encryption.
- -subj : Sets the subject of the request.
- $\bullet\,$  -out : Specifies the name of the generated certificate request or self-signed certificate.
- -keyout : Specifies the name of the automatically generated secret key.
- 3. Embed the Explorer page by using iframe on a third-party page. The work needs to be developed by yourself.
- 4. On the parent page, pass the login message through the postMessage method in the following format:

```
{ type: 'NebulaGraphExploreLogin', data: { authorization: 'MyJyb290IiwibmVidWxhIl0=', host: '192.168.8.240:9669',
```

space: 'basketballplayer'
} }

- $\bullet$  type: The method type must be <code>NebulaGraphExploreLogin</code> .
- data:
- authorization: NebulaGraph accounts and passwords were formed into an array and serialized, then Base64 encoded. The array format is ['account', 'password']. The example is['root', 'nebula']. The encoded result is WyJyb290IiwibmVidWxhIlO=.
- host: The graph service address of NebulaGraph.
- space: The name of the target graph space.
- 5. Start the Explorer service.



If the Explorer is installed by RPM/DEB package, run the command sudo ./nebula-explorer-server & °

./scripts/start.sh

6. Check whether the embedded Explorer page is displayed on the third-party page. For example, the first page displays the graph space basketballplayer, and the second and third pages display other graph spaces.



# 16.10 Basic operations and shortcuts

This topic lists the basic operations and shortcuts supported in Explorer.

## 16.10.1 Basic operations

Operation	Description
Move a canvas	Hold down left click and drag the canvas.
Zoom in or out the canvas	Use the mouse wheel to zoom in or out.
Select one single vertex or edge	Left-click a vertex or an edge.
Select multiple vertices and edges	Hold Shift and left-click vertices and edges.
Batch selection	Hold down right click and frame vertices and edges; Or Hold Shift and hold down left click, and then frame vertices and edges.
Move selected vertices	Left-click the selected vertices and then move them.

## 16.10.2 Shortcuts

Operation	Description
Enter	Expand
Shift + '-'	Zoom out
Shift + '+'	Zoom in
Shift + 'l'	Display
Ctrl/Cmd + 'z'	Undo
Ctrl/Cmd + Shift + 'z'	Redo
Ctrl/Cmd + 'a'	Select all vertices.
Selected + 'Backspace'	Hide the selected elements.
Selected + Shift + 'Backspace'	Hide the unselected elements.

Last update: February 19, 2024

- 800/1069 - 2023 Vesoft Inc.

## 16.11 FAQ

This topic lists the frequently asked questions for using NebulaGraph Explorer. You can use the search box in the help center or the search function of the browser to match the questions you are looking for.

#### 16.11.1 Will the Dag Controller service crash if the Graph service returns too much result data?

The Dag Controller service only provides scheduling capabilities and will not crash, but the NebulaGraph Analytics service may crash due to insufficient memory when writing too much data to HDFS or NebulaGraph, or reading too much data from HDFS or NebulaGraph.

#### 16.11.2 Can I continue a job from a failed task?

Not supported. You can only re-execute the entire job.

#### 16.11.3 How can I speed it up if a task result is saved slowly or data is transferred slowly between tasks?

The Dag Controller contains graph query components and graph computing components. Graph queries send requests to a graph service for queries, so the graph queries can only be accelerated by increasing the memory of the graph service. Graph computing is performed on distributed nodes provided by NebulaGraph Analytics, so graph computing can be accelerated by increasing the size of the NebulaGraph Analytics cluster.

#### 16.11.4 The HDFS server cannot be connected and the task status is running.

Set the timeout period for HDFS connections as follows:

### 16.11.5 How to resolve the error Err:dial unix: missing address?

Modify the configuration file dag-ctrl/etc/dag-ctrl-api.yaml to configure the UserName of the SSH.

#### 16.11.6 How to resolve the error bash: /home/xxx/nebula-analytics/scripts/run\_algo.sh: No such file or directory?

 $Modify \ the \ configuration \ file \ \ dag-ctrl/etc/tasks. \textit{yaml to configure the algorithm execution path parameter } \ \ exec\_file.$ 

# 16.11.7 How to resolve the error /lib64/libm.so.6: version 'GLIBC\_2.29' not found (required by /home/vesoft/jdk-18.0.1/jre/lib/amd64/server/libjvm.so)?

Because the operating system version does not support JDK18, the command YUM cannot download GLIBC\_2.29, you can install JDK1.8. Does not forget to change the JDK address in nebula-analytics/scripts/set\_env.sh.

# 16.11.8 How to resolve the error handshake failed: ssh: unable to authenticate, attempted methods [none publickey], no supported methods remain?

Reconfigure the permissions to 744 on the folder .ssh and 600 on the file  $.ssh/authorized\_keys$  .

16.11.9 How to resolve the error There are 0 NebulaGraph Analytics available. clusterSize should be less than or equal to it?

Check according to the following procedure:

1. Check whether the configuration of SSH password-free login between nodes is successful. You can run the ssh <user\_name>@<node\_ip> command on the Dag Controller machine to check whether the login succeeds.



If the Dag Controller and Analytics are on the same machine, you also need to configure SSH password-free login.

- 2. Check the configuration file of the Dag Controller.
- Check whether the SSH user in etc/dag-ctrl-api.yaml is the same as the user who starts the Dag Controller service and the user who configs SSH password-free login.
- Check whether the algorithm path in etc/tasks.yaml is correct.
- Check whether Hadoop and Java paths in scripts/set\_env.sh are correct.
- 3. Restart the Dag Controller for the settings to take effect.

16.11.10 How to resolve the error no available namenodes: dial tcp xx.xx.xx:8020: connect: connection timed out?

Check whether the HDFS namenode port 8020 is open.

16.11.11 How to resolve the error org.apache.hadoop.net.ConnectTimeoutException: 60000 millis timeout?

Check whether the HDFS datanode port 50010 is open.

If the port is not opened, an error similar to the following may be reported:

- Check failed: false close hdfs-file failed
- org.apache.hadoop.ipc.RemoteException(java.io.IOException): File /analytics/xx/tasks/analytics\_xxx/xxx.csv could only be replicated to 0 nodes instead of minReplication

#### 16.11.12 How to resolve the error

broadcast.hpp:193] Check failed: (size\_t)recv\_bytes >= sizeof(chunk\_tail\_t) recv message too small: 0?

The amount of data to be processed is too small, but the number of compute nodes and processes is too large. Smaller <code>clusterSize</code> and <code>processes</code> need to be set when submitting jobs.

Last update: February 19, 2024

- 802/1069 - 2023 Vesoft Inc.

## 17. NebulaGraph Importer

## 17.1 NebulaGraph Importer

NebulaGraph Importer (Importer) is a standalone tool for importing data from CSV files into NebulaGraph. Importer can read the local CSV file and then import the data into the NebulaGraph database.

#### 17.1.1 Scenario

Importer is used to import the contents of a local CSV file into the NebulaGraph.

## 17.1.2 Advantage

- Lightweight and fast: no complex environment can be used, fast data import.
- Flexible filtering: You can flexibly filter CSV data through configuration files.

#### 17.1.3 Release note

#### Release

### 17.1.4 Prerequisites

Before using NebulaGraph Importer, make sure:

- NebulaGraph service has been deployed. There are currently three deployment modes:
- Deploy NebulaGraph with Docker Compose
- Install NebulaGraph with RPM or DEB package
- $\bullet$  Install Nebula Graph by compiling the source code
- $\bullet \ Schema \ is \ created \ in \ Nebula Graph, including \ space, \ Tag \ and \ Edge \ type, \ or \ set \ by \ parameter \ \ \verb|clientSettings.postStart.commands.|$
- Golang environment has been deployed on the machine running the Importer. For details, see Build Go environment.

#### 17.1.5 Steps

Configure the YAML file and prepare the CSV file to be imported to use the tool to batch write data to NebulaGraph.

#### Download binary package and run

- 1. Download the binary package directly and add execute permission to it.
- 2. Start the service.

```
$ ./<binary_package_name> --config <yaml_config_file_path>
```

#### Source code compile and run

1. Clone repository.

\$ git clone -b release-3.4 https://github.com/vesoft-inc/nebula-importer.git

- 803/1069 - 2023 Vesoft Inc.



Use the correct branch. NebulaGraph 2.x and 3.x have different RPC protocols.

2. Access the directory nebula-importer.

```
$ cd nebula-importer
```

3. Compile the source code.

```
$ make build
```

4. Start the service.

```
$ ./nebula-importer --config <yaml_config_file_path>
```



For details about the YAML configuration file, see configuration file description at the end of topic.

#### No network compilation mode

If the server cannot be connected to the Internet, it is recommended to upload the source code and various dependency packages to the corresponding server for compilation on the machine that can be connected to the Internet. The operation steps are as follows:

1. Clone repository.

```
$ git clone -b release-3.4 https://github.com/vesoft-inc/nebula-importer.git
```

2. Use the following command to download and package the dependent source code.

```
$ cd nebula-importer
$ go mod vendor
$ cd .. && tar -zcvf nebula-importer.tar.gz nebula-importer
```

- 3. Upload the compressed package to a server that cannot be connected to the Internet.
- 4. Unzip and compile.

```
$ tar -zxvf nebula-importer.tar.gz
$ cd nebula-importer
$ go build -mod vendor cmd/importer.go
```

#### Run in Docker mode

Instead of installing the Go locale locally, you can use Docker to pull the <a href="image">image</a> of the NebulaGraph Importer and mount the local configuration file and CSV data file into the container. The command is as follows:

```
$ docker run --rm -ti \
    --network=host \
    -v <config_file>:<config_file> \
    -v <csv_data_dir>:<csv_data_dir> \
    vesoft/nebula-importer:<version>
    --config <config_file>
```

- <config\_file> : The absolute path to the local YAML configuration file.
- $\bullet$   $\mbox{-}\mbox{csv\_data\_dir>}$  : The absolute path to the local CSV data file.
- <version>: NebulaGraph 2.x Please fill in 'v3'.



A relative path is recommended. If you use a local absolute path, check that the path maps to the path in the Docker.

#### 17.1.6 Configuration File Description

NebulaGraph Importer uses configuration(nebula-importer/examples/v2/example.yaml) files to describe information about the files to be imported, the NebulaGraph server, and more. You can refer to the example configuration file: Configuration without Header/Configuration with Header. This section describes the fields in the configuration file by category.



If users download a binary package, create the configuration file manually.

#### **Basic configuration**

The example configuration is as follows:

```
version: v2
description: example
removeTempFiles: false
```

Parameter	Default value	Required	Description
version	v2	Yes	Target version of the configuration file.
description	example	No	Description of the configuration file.
removeTempFiles	false	No	Whether to delete temporarily generated logs and error data files.

#### Client configuration

The client configuration stores the configurations associated with NebulaGraph.

UPDATE CONFIGS storage:wal\_ttl=86400;
UPDATE CONFIGS storage:rocksdb\_column\_family\_options = { disable\_auto\_compactions = false };

Parameter	Default value	Required	Description
clientSettings.retry	3	No	Retry times of nGQL statement execution failures.
clientSettings.concurrency	10	No	Number of NebulaGraph client concurrency.
clientSettings.channelBufferSize	128	No	Cache queue size per NebulaGraph client.
clientSettings.space	-	Yes	Specifies the NebulaGraph space to import the data into. Do not import multiple spaces at the same time to avoid performance impact.
clientSettings.connection.user	-	Yes	NebulaGraph user name.
clientSettings.connection.password	-	Yes	The password for the NebulaGraph user name.
clientSettings.connection.address	-	Yes	Addresses and ports for all Graph services.
clientSettings.postStart.commands	-	No	Configure some of the operations to perform after connecting to the NebulaGraph server, and before inserting data.
clientSettings.postStart.afterPeriod	-	No	The interval, between executing the above commands and executing the insert data command, such as 8s.
clientSettings.preStop.commands	-	No	Configure some of the actions you performed before disconnecting from the NebulaGraph server.

#### File configuration

File configuration Stores the configuration of data files and logs, and details about the Schema.

FILE AND LOG CONFIGURATION

```
workingDir: ./data/
LogPath: ./err/test.log
files:
    path: ./student.csv
    failDataPath: ./err/student.csv
batchSize: 128
    Limit: 10
    inOrder: false
    type: csv
    csv:
    withHeader: false
    withLabel: false
```

delimiter:	","
lazyQuotes:	false

Parameter	Default value	Required	Description
workingDir	-	No	If you have multiple directories containing data with the same file structure, you can use this parameter to switch between them. For example, the value of path and failDataPath of the configuration below will be automatically changed to ./data/student.csv and ./data/err/student . If you change workingDir to ./data1, the path will be changed accordingly. The param can be either absolute or relative.
logPath	-	No	Path for exporting log information, such as errors during import.
files.path	-	Yes	Path for storing data files. If a relative path is used, the path is merged with the current configuration file directory. You can use an asterisk (*) for fuzzy matching to import multiple files with similar names, but the files need to be the same structure.
files.failDataPath	-	Yes	Insert the failed data file storage path, so that data can be written later.
files.batchSize	128	No	The number of statements inserting data in a batch.
files.limit	-	No	Limit on the number of rows of read data.
files.inOrder	-	No	Whether to insert rows in the file in order. If the value is set to $\mbox{ false}$ , the import rate decreases due to data skew.
files.type	-	Yes	The file type.
files.csv.withHeader	false	Yes	Whether there is a header.
files.csv.withLabel	false	Yes	Whether there is a label.
files.csv.delimiter	п, п	Yes	Specifies the delimiter for the CSV file. A string delimiter that supports only one character.
lazyQuotes	false	No	If lazyQuotes is true, a quote may appear in an unquoted field and a non-doubled quote may appear in a quoted field.

#### SCHEMA CONFIGURATION

Schema configuration describes the Meta information of the current data file. Schema types are vertex and edge. Multiple vertexes or edges can be configured at the same time.

## • vertex configuration

```
nullable: true
nullValue: "__NULL__"
```

Parameter	Default value	Required	Description
files.schema.type	-	Yes	Schema type. Possible values are $\ensuremath{\text{vertex}}$ and $\ensuremath{\text{edge}}$ .
files.schema.vertex.vid.index	-	No	The vertex ID corresponds to the column number in the $\ensuremath{CSV}$
files.schema.vertex.vid.function	-	No	Functions to generate the VIDs. Currently, we only support fu
files.schema.vertex.vid.prefix	-	No	Add prefix to the original vid. When function is specified also function .
files.schema.vertex.tags.name	-	Yes	Tag name.
files.schema.vertex.tags.props.name	-	Yes	Tag property name, which must match the Tag property in th
files.schema.vertex.tags.props.type	-	Yes	Property data type, supporting bool, int, float, double, string, time, timestamp, date, datetime, geogram geography(polygon).
files.schema.vertex.tags.props.index	-	No	Property corresponds to the sequence number of the column
files.schema.vertex.tags.props.nullable	false	No	Whether this prop property can be ${\tt NULL}{\tt J}$ , optional values is ${\tt tr}$
files.schema.vertex.tags.props.nullValue	нн	No	Ignored when nullable is false. The property is set to NULL w
files.schema.vertex.tags.props.alternativeIndices	-	No	Ignored when nullable is false. When the property value is no according to the index sequence.
files.schema.vertex.tags.props.defaultValue	-	No	Ignored when nullable is false. The property default value, whalternativeIndices are nullValue.



The sequence numbers of the columns in the CSV file start from 0, that is, the sequence numbers of the first column are 0, and the sequence numbers of the second column are 1.

## • edge configuration

```
schema:
type: edge
edge:
name: follow
srcVID:
index: 0
function: hash
dstVID:
index: 1
function:
rank:
index
                  rank:
index: 2
props:
- name: grade
```

type: int index: 3

Parameter	Default value	Required	Description
files.schema.type	-	Yes	Schema type. Possible values are $\ensuremath{\text{vertex}}$ and $\ensuremath{\text{edge}}$ .
files.schema.edge.name	-	Yes	Edge type name.
files.schema.edge.srcVID.index	-	No	The data type of the starting vertex ID of the edge.
files.schema.edge.srcVID.function	-	No	Functions to generate the source vertex. Currently, we only support function $\ensuremath{hash}.$
files.schema.edge.dstVID.index	-	No	The destination vertex ID of the edge corresponds to the column number in the CSV file.
files.schema.edge.dstVID.function	-	No	Functions to generate the destination vertex. Currently, we only support function $\ensuremath{hash}$ .
files.schema.edge.rank.index	-	No	The rank value of the edge corresponds to the column number in the CSV file.
files.schema.edge.props.name	-	Yes	The Edge Type property name must match the Edge Type property in the NebulaGraph.
files.schema.edge.props.type	-	Yes	Property data type, supporting bool, int, float, double, timestamp, string, and geo.
files.schema.edge.props.index	-	No	Property corresponds to the sequence number of the column in the CSV file.

## 17.1.7 About the CSV file header

According to whether the CSV file has a header or not, the Importer needs to make different Settings on the configuration file. For relevant examples and explanations, please refer to:

- Configuration without Header
- Configuration with Header

## 17.2 Configuration with Header

For a CSV file with header, you need to set withHeader to true in the configuration file, indicating that the first behavior in the CSV file is the header. The header content has special meanings.



If the CSV file contains headers, the Importer will parse the Schema of each row of data according to the headers and ignore the vertex or edge settings in the YAML file.

#### 17.2.1 Sample files

The following is an example of a CSV file with header:

sample of vertex

Example data for student\_with\_header.csv:

```
:VID(string),student.name:string,student.age:int,student.gender:string
student100,Monica,16,female
student101,Mike,18,male
student102,Jane,17,female
```

The first column is the vertex ID, followed by the properties <code>name</code> , <code>age</code> , and <code>gender</code> .

· sample of edge

Example data for follow\_with\_header.csv:

```
:SRC_VID(string),:DST_VID(string),:RANK,follow.degree:double student100,student101,0,92.5 student101,student100,1,85.6 student101,student102,2,93.2 student100,student102,1,96.2
```

The first two columns are the start vertex ID and destination vertex ID, respectively. The third column is rank, and the fourth column is property degree.

#### 17.2.2 Header format description

The header defines the start vertex, the destination vertex, the rank, and some special functions by keywords as follows:

- :VID (mandatory): Vertex ID. Need to use :VID(type) form to set data type, for example :VID(string) or :VID(int).
- :SRC\_VID (mandatory): The start vertex ID of the edge. The data type needs to be set in the form :SRC\_VID(type) .
- $\bullet \ \ \text{:DST\_VID} \ (mandatory) \text{: The destination vertex ID of the edge. The data type needs to be set in the form} \ \ \ \text{:DST\_VID} \ (type) \ .$
- :RANK (optional): The rank value of the edge.
- :IGNORE (optional): Ignore this column when inserting data.
- :LABEL (optional): Insert (+) or delete (-) the row. Must be column 1. For example:

```
:LABEL,
+,
-,
```

- 810/1069 - 2023 Vesoft Inc.

#### Q Note

All columns except the :LABEL column can be sorted in any order, so for larger CSV files, the user has the flexibility to set the header to select the desired column.

For Tag or Edge type properties, the format is <tag\_name/edge\_name>.cypep\_type>, described as follows:

- <tag\_name/edge\_name> : Tag or Edge type name.
- prop\_name> : property name.
- $\bullet \ \verb|\| \mathsf{sprop_type}| : property \ type. \ Support \ \mathsf{bool} \ , \ \mathsf{int} \ , \ \mathsf{float} \ , \ \mathsf{double} \ , \ \mathsf{timestamp} \ \ \mathsf{and} \ \ \mathsf{string} \ , \ \mathsf{default} \ \ \mathsf{string} \ .$

Such as student.name:string, follow.degree:double.

#### 17.2.3 Sample configuration

```
version: v2
description: example
# Whether to delete temporarily generated logs and error data files.
removeTempFiles: false
  # Retry times of nGQL statement execution failures.
  # Number of NebulaGraph client concurrency.
  concurrency: 10
  # Cache queue size per NebulaGraph client.
  channelBufferSize: 128
  # Specifies the NebulaGraph space to import the data into.
  space: student
  # Connection information
  connection:
   user: root
    password: nebula
    address: 192.168.*.13:9669
    # Configure some of the operations to perform after connecting to the NebulaGraph server, and before inserting data.
    commands: |
DROP SPACE IF EXISTS student;
      CREATE SPACE IF NOT EXISTS student(partition_num=5, replica_factor=1, vid_type=FIXED_STRING(20));
     USE student;
CREATE TAG student(name string, age int,gender string);
CREATE EDGE follow(degree int);
    # The interval between the execution of the above command and the execution of the insert data command.
    # Configure some of the actions you performed before disconnecting from the NebulaGraph server.
    commands: |
# Path of the error log file.
logPath: ./err/test.log
# CSV file Settings
    # Path for storing data files. If a relative path is used, the path is merged with the current configuration file directory. The first data file in this example is vertex data.
  - path: ./student_with_header.csv
    # Insert the failed data file storage path, so that data can be written later.
    # The number of statements inserting data in a batch.
    # Limit on the number of rows of read data.
    # Whether to insert rows in the file in order. If the value is set to false, the import rate decreases due to data skew.
    # File type. Currently, only CSV files are supported.
    type: csv
```



The data type of the vertex ID must be the same as the data type of the statement in clientSettings.postStart.commands that creates the graph space.

## 17.3 Configuration without Header

For CSV files without header, you need to set withHeader to false in the configuration file, indicating that the CSV file contains only data (excluding the header of the first row). You may also need to set the data type and corresponding columns.

#### 17.3.1 Sample files

The following is an example of a CSV file without header:

· sample of vertex

Example data for student\_without\_header.csv:

```
student100,Monica,16,female
student101,Mike,18,male
student102,Jane,17,female
```

The first column is the vertex ID, followed by the properties name, age, and gender.

· sample of edge

Example data for follow\_without\_header.csv:

```
student100,student101,0,92.5
student101,student100,1,85.6
student101,student102,2,93.2
student100,student102,1,96.2
```

The first two columns are the start vertex ID and destination vertex ID, respectively. The third column is rank, and the fourth column is property degree.

## 17.3.2 Sample configuration

```
version: v2
description: example
# Whether to delete temporarily generated logs and error data files.
removeTempFiles: false
clientSettings:
 # Retry times of nGQL statement execution failures.
 # Number of NebulaGraph client concurrency
 concurrency: 10
 # Cache queue size per NebulaGraph client.
 channelBufferSize: 128
 # Specifies the NebulaGraph space to import the data into.
 space: student
 # Connection information.
   password: nebula
    address: 192.168.*.13:9669
    # Configure some of the operations to perform after connecting to the NebulaGraph server, and before inserting data.
     DROP SPACE IF EXISTS student;
      CREATE SPACE IF NOT EXISTS student(partition_num=5, replica_factor=1, vid_type=FIXED_STRING(20));
     CREATE TAG student(name string, age int,gender string);
     CREATE EDGE follow(degree int)
    # The interval between the execution of the above command and the execution of the insert data command
    # Configure some of the actions you performed before disconnecting from the NebulaGraph server
    commands:
```

```
# Path of the error log file.
logPath: ./err/test.log
# CSV file Settings.
files:
    # Path for storing data files. If a relative path is used, the path is merged with the current configuration file directory. The first data file in this example is vertex data.
  - path: ./student_without_header.csv
    # Insert the failed data file storage path, so that data can be written later.
    failDataPath: ./err/studenterr
    # The number of statements inserting data in a batch.
    # Limit on the number of rows of read data.
    limit: 10
    # Whether to insert rows in the file in order. If the value is set to false, the import rate decreases due to data skew.
    # File type. Currently, only CSV files are supported.
    type: csv
    csv:
# Whether there is a header.
      withHeader: false
       # Whether there is a LABEL.
      withLabel: false
       # Specifies the delimiter for the CSV file. A string delimiter that supports only one character
       delimiter: ","
      # Schema type. Possible values are vertex and edge.
       type: vertex
       vertex:
         # Vertex ID Settings.
           # The vertex ID corresponds to the column number in the CSV file. Columns in the CSV file are numbered from 0.
            # The data type of the vertex ID. The optional values are int and string, corresponding to INT64 and FIXED_STRING in the NebulaGraph, respectively.
         # Tag Settings.
        tags:
# Tag name.
           - name: student
             # property Settings in the Tag.
             props:
# property name.
                 # Property data type.
                 # Property corresponds to the sequence number of the column in the CSV file.
                  index: 1
                 type: int index: 2
                  name: gender
                 type: string index: 3
    # The second data file in this example is edge data.
path: ./follow_without_header.csv
failDataPath: ./err/followerr
batchSize: 10
     limit: 10
    inOrder: true
type: csv
    csv:
withHeader: false
withLabel: false
       # The type of Schema is edge.
       type: edge
       edge:
        # Edge type name.
        name: follow
         # Whether to include rank.
         withRanking: true
         # Start vertex ID setting.
         srcVID:
           # Data type.
```

```
type: string

# The start vertex ID corresponds to the sequence number of a column in the CSV file.
index: 0

# Destination vertex ID.
dstVID:
type: string
index: 1

# rank setting.
rank:
# Rank Indicates the rank number of a column in the CSV file. If index is not set, be sure to set the rank value in the third column. Subsequent columns set each property in turn.

index: 2

# Edge Type property Settings.
props:
# property name.
- name: degree

# Data type.
type: double

# Property corresponds to the sequence number of the column in the CSV file.
index: 3
```

## Note

- The sequence numbers of the columns in the CSV file start from 0, that is, the sequence numbers of the first column are 0, and the sequence numbers of the second column are 1.
- The data type of the vertex ID must be the same as the data type of the statement in clientSettings.postStart.commands that creates the graph space.
- If the index field is not specified, the CSV file must comply with the following rules:
- In the vertex data file, the first column must be the vertex ID, followed by the properties, and must correspond to the order in the configuration file.
- In the side data file, the first column must be the start vertex ID, the second column must be the destination vertex ID, if withRanking is true, the third column must be the rank value, and the following columns must be properties, and must correspond to the order in the configuration file.

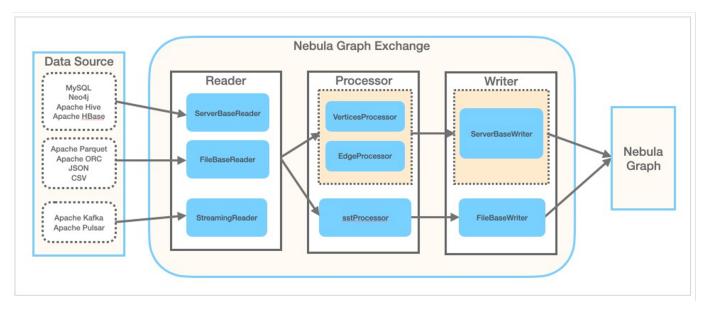
# 18. NebulaGraph Exchange

#### 18.1 Introduction

#### 18.1.1 What is NebulaGraph Exchange

NebulaGraph Exchange (Exchange) is an Apache Spark $^{\text{m}}$  application for bulk migration of cluster data to NebulaGraph in a distributed environment, supporting batch and streaming data migration in a variety of formats.

Exchange consists of Reader, Processor, and Writer. After Reader reads data from different sources and returns a DataFrame, the Processor iterates through each row of the DataFrame and obtains the corresponding value based on the mapping between fields in the configuration file. After iterating through the number of rows in the specified batch, Writer writes the captured data to the NebulaGraph at once. The following figure illustrates the process by which Exchange completes the data conversion and migration.



## **Editions**

Exchange has two editions, the Community Edition and the Enterprise Edition. The Community Edition is open source developed on GitHub. The Enterprise Edition supports not only the functions of the Community Edition but also adds additional features. For details, see Comparisons.

#### Scenarios

Exchange applies to the following scenarios:

- Streaming data from Kafka and Pulsar platforms, such as log files, online shopping data, activities of game players, information on social websites, financial transactions or geospatial services, and telemetry data from connected devices or instruments in the data center, are required to be converted into the vertex or edge data of the property graph and import them into the NebulaGraph database.
- Batch data, such as data from a time period, needs to be read from a relational database (such as MySQL) or a distributed file system (such as HDFS), converted into vertex or edge data for a property graph, and imported into the NebulaGraph database.
- A large volume of data needs to be generated into SST files that NebulaGraph can recognize and then imported into the NebulaGraph database.
- The data saved in NebulaGraph needs to be exported.



Exporting the data saved in NebulaGraph is supported by Exchange Enterprise Edition only.

#### **Advantages**

Exchange has the following advantages:

- High adaptability: It supports importing data into the NebulaGraph database in a variety of formats or from a variety of sources, making it easy to migrate data.
- SST import: It supports converting data from different sources into SST files for data import.
- SSL encryption: It supports establishing the SSL encryption between Exchange and NebulaGraph to ensure data security.
- Resumable data import: It supports resumable data import to save time and improve data import efficiency.



Resumable data import is currently supported when migrating Neo4j data only.

- Asynchronous operation: An insert statement is generated in the source data and sent to the Graph service. Then the insert operation is performed.
- Great flexibility: It supports importing multiple Tags and Edge types at the same time. Different Tags and Edge types can be from different data sources or in different formats.
- Statistics: It uses the accumulator in Apache Spark™ to count the number of successful and failed insert operations.
- Easy to use: It adopts the Human-Optimized Config Object Notation (HOCON) configuration file format and has an object-oriented style, which is easy to understand and operate.

- 817/1069 - 2023 Vesoft Inc.

## Version compatibility

The correspondence between the NebulaGraph Exchange version (the JAR version), the NebulaGraph core version and the Spark version is as follows.

Exchange version	NebulaGraph version	Spark version
$nebula-exchange\_spark\_3.0-3.0-SNAPSHOT.jar$	nightly	3.3.x \ 3.2.x \ 3.1.x \ 3.0.x
$nebula-exchange\_spark\_2.4-3.0-SNAPSHOT.jar$	nightly	2.4.x
$nebula-exchange\_spark\_2.2-3.0-SNAPSHOT.jar$	nightly	2.2.x
$nebula-exchange\_spark\_3.0-3.4.0.jar$	3.x.x	3.3.x \ 3.2.x \ 3.1.x \ 3.0.x
$nebula-exchange\_spark\_2.4-3.4.0.jar$	3.x.x	2.4.x
$nebula-exchange\_spark\_2.2-3.4.0.jar$	3.x.x	2.2.x
$nebula-exchange\_spark\_3.0-3.3.0.jar$	3.x.x	3.3.x \ 3.2.x \ 3.1.x \ 3.0.x
$nebula-exchange\_spark\_2.4-3.3.0.jar$	3.x.x	2.4.x
$nebula-exchange\_spark\_2.2-3.3.0.jar$	3.x.x	2.2.x
$nebula-exchange\_spark\_3.0-3.0.0.jar$	3.x.x	3.3.x \ 3.2.x \ 3.1.x \ 3.0.x
$nebula-exchange\_spark\_2.4-3.0.0.jar$	3.x.x	2.4.x
$nebula-exchange\_spark\_2.2-3.0.0.jar$	3.x.x	2.2.x
nebula-exchange-2.6.3.jar	2.6.1 \ 2.6.0	2.4.x
nebula-exchange-2.6.2.jar	2.6.1 ` 2.6.0	2.4.x
nebula-exchange-2.6.1.jar	2.6.1 ` 2.6.0	2.4.x
nebula-exchange-2.6.0.jar	2.6.1 \ 2.6.0	2.4.x
nebula-exchange-2.5.2.jar	2.5.1 \ 2.5.0	2.4.x
nebula-exchange-2.5.1.jar	2.5.1 \ 2.5.0	2.4.x
nebula-exchange-2.5.0.jar	2.5.1 \ 2.5.0	2.4.x
nebula-exchange-2.1.0.jar	2.0.1 \ 2.0.0	2.4.x
nebula-exchange-2.0.1.jar	2.0.1 \ 2.0.0	2.4.x
nebula-exchange-2.0.0.jar	2.0.1 \ 2.0.0	2.4.x

- 818/1069 - 2023 Vesoft Inc.

#### Data source

Exchange 3.4.0 supports converting data from the following formats or sources into vertexes and edges that NebulaGraph can recognize, and then importing them into NebulaGraph in the form of nGQL statements:

- Data stored in HDFS or locally:
- Apache Parquet
- Apache ORC
- JSON
- CSV
- Apache HBase<sup>™</sup>
- · Data repository:
- Hive
- MaxCompute
- Graph database: Neo4j (Client version 2.4.5-M1)
- · Relational database:
- MySQL
- PostgreSQL
- Oracle
- Column database: ClickHouse
- Stream processing software platform: Apache Kafka®
- Publish/Subscribe messaging platform: Apache Pulsar 2.4.5
- JDBC

In addition to importing data as nGQL statements, Exchange supports generating SST files for data sources and then importing SST files via Console.

In addition, Exchange Enterprise Edition also supports exporting data to a CSV file or another graph space using NebulaGraph as data sources.

#### Release note

## Release

Last update: February 19, 2024

- 819/1069 - 2023 Vesoft Inc.

## 18.1.2 Limitations

This topic describes some of the limitations of using Exchange 3.x.

JAR packages are available in two ways: compile them yourself or download them from the Maven repository.

If you are using NebulaGraph 1.x, use NebulaGraph Exchange 1.x.

#### **Environment**

Exchange 3.x supports the following operating systems:

- CentOS 7
- macOS

#### Software dependencies

To ensure the healthy operation of Exchange, ensure that the following software has been installed on the machine:

- Java version 1.8
- Scala version 2.10.7, 2.11.12, or 2.12.10
- Apache Spark. The requirements for Spark versions when using Exchange to export data from data sources are as follows. In the following table, Y means that the corresponding Spark version is supported, and N means not supported.



Use the correct Exchange JAR file based on the Spark version. For example, for Spark version 2.4, use nebula-exchange\_spark\_2.4-3.4.0.jar.

Data source	Spark 2.2	Spark 2.4	Spark 3
CSV file	Y	N	Y
JSON file	Y	Y	Y
ORC file	Y	Y	Y
Parquet file	Y	Y	Y
HBase	Y	Y	Y
MySQL	Y	Y	Y
PostgreSQL	Y	Y	Y
Oracle	Y	Y	Y
ClickHouse	Y	Y	Y
Neo4j	N	Y	N
Hive	Y	Y	Y
MaxCompute	N	Y	N
Pulsar	N	Y	Untested
Kafka	N	Y	Untested
NebulaGraph	N	Y	N

- 820/1069 - 2023 Vesoft Inc.

 $\label{thm:eq:hadoop} \mbox{ Distributed File System (HDFS) needs to be deployed in the following scenarios:}$ 

- Migrate HDFS data
- Generate SST files

## 18.2 Get Exchange

This topic introduces how to get the JAR file of NebulaGraph Exchange.

#### 18.2.1 Download the JAR file directly

The JAR file of Exchange Community Edition can be downloaded directly.

To download Exchange Enterprise Edition, get NebulaGraph Enterprise Edition Package first.

#### 18.2.2 Get the JAR file by compiling the source code

You can get the JAR file of Exchange Community Edition by compiling the source code. The following introduces how to compile the source code of Exchange.



You can get Exchange Enterprise Edition in NebulaGraph Enterprise Edition Package only.

#### Prerequisites

- Install Maven.
- Install the correct version of Apache Spark. Exporting data from different sources requires different Spark versions. For more information, see Software dependencies.

#### 18.2.3 Steps

1. Clone the repository nebula-exchange in the / directory.

```
git clone -b release-3.4 https://github.com/vesoft-inc/nebula-exchange.git
```

2. Switch to the directory nebula-exchange.

cd nebula-exchange

- 3. Package NebulaGraph Exchange. Run the following command based on the Spark version:
- For Spark 2.2:

```
mvn clean package -Dmaven.test.skip=true -Dgpg.skip -Dmaven.javadoc.skip=true \
-pl nebula-exchange_spark_2.2 -am -Pscala-2.11 -Pspark-2.2
```

• For Spark 2.4:

```
mvn clean package -Dmaven.test.skip=true -Dgpg.skip -Dmaven.javadoc.skip=true \
-pl nebula-exchange_spark_2.4 -am -Pscala-2.11 -Pspark-2.4
```

• For Spark 3.0:

```
mvn clean package -Dmaven.test.skip=true -Dgpg.skip -Dmaven.javadoc.skip=true \
-pl nebula-exchange_spark_3.0 -am -Pscala-2.12 -Pspark-3.0
```

After the compilation is successful, you can find the <code>nebula-exchange\_spark\_x.x-release-3.4.jar</code> file in the <code>nebula-exchange\_spark\_x.x/target/directory. x.x</code> indicates the Spark version, for example, 2.4.

- 822/1069 - 2023 Vesoft Inc.



The JAR file version changes with the release of the NebulaGraph Java Client. Users can view the latest version on the Releases page.

When migrating data, you can refer to configuration file target/classes/application.conf .

## Failed to download the dependency package

If downloading dependencies fails when compiling:

- Check the network settings and ensure that the network is normal.
- Modify the mirror part of Maven installation directory Libexec/conf/settings.xml:

<mirror>
<id>alimaven</id>
<mirror0f>
<name>aliyun avven</name>
<url>http://maven.aliyun.com/nexus/content/repositories/central/</url>
</mirror>

## 18.3 Exchange configurations

#### 18.3.1 Options for import

After editing the configuration file, run the following commands to import specified source data into the NebulaGraph database.

• First import

```
<spark_install_path>/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange <nebula-exchange-2.x.y.jar_path> -c <application.conf_path>
```

· Import the reload file

If some data fails to be imported during the first import, the failed data will be stored in the reload file. Use the parameter -r to import the reload file.

<spark\_install\_path>/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-2.x.y.jar\_path> -c <application.conf\_path> -r "<reload\_file\_path>"



The version number of a JAR file is subject to the name of the JAR file that is actually compiled.



If users use the yarn-cluster mode to submit a job, see the following command, **especially the two '--conf' commands in the example**.

```
$$PARK_HOME/bin/spark-submit --master yarn-cluster \
--class com.vesoft.nebula.exchange.Exchange \
--files application.conf \
--conf spark.driver.extraClassPath=./ \
--conf spark.devecutor.extraClassPath=./ \
nebula-exchange-3.4.0.jar \
-c application.conf
```

The following table lists command parameters.

Parameter	Required	Default value	Description
class	Yes	-	Specify the main class of the driver.
master	Yes	-	Specify the URL of the master process in a Spark cluster. For more information, see $\frac{1}{2}$
-c / config	Yes	-	Specify the path of the configuration file.
-h /hive	No	false	Indicate support for importing Hive data.
-D /dry	No	false	Check whether the format of the configuration file meets the requirements, but it does not check whether the configuration items of tags and edges are correct. This parameter cannot be added when users import data.
-r /reload	No	-	Specify the path of the reload file that needs to be reloaded.

For more Spark parameter configurations, see Spark Configuration.

## 18.3.2 Parameters in the configuration file

This topic describes how to configure the file application.conf when users use NebulaGraph Exchange.

Before configuring the application.conf file, it is recommended to copy the file name application.conf and then edit the file name according to the file type of a data source. For example, change the file name to <code>csv\_application.conf</code> if the file type of the data source is CSV.

The application.conf file contains the following content types:

- Spark configurations
- Hive configurations (optional)
- NebulaGraph configurations
- Vertex configurations
- Edge configurations

#### Spark configurations

This topic lists only some Spark parameters. For more information, see Spark Configuration.

Parameter	Туре	Default value	Required	Description
spark.app.name	string	-	No	The drive name in Spark.
spark.driver.cores	int	(1)	No	The number of CPU cores used by a driver, only applicable to a cluster mode.
spark.driver.maxResultSize	string	16	No	The total size limit (in bytes) of the serialized results of all partitions in a single Spark operation (such as collect). The minimum value is 1M, and 0 means unlimited.
spark.executor.memory	string	1G	No	The amount of memory used by a Spark driver which can be specified in units, such as 512M or 1G.
spark.cores.max	int	16	No	The maximum number of CPU cores of applications requested across clusters (rather than from each node) when a driver runs in a coarse-grained sharing mode on a standalone cluster or a Mesos cluster. The default value is spark.deploy.defaultCores on a Spark standalone cluster manager or the value of the infinite parameter (all available cores) on Mesos.

## Hive configurations (optional)

Users only need to configure parameters for connecting to Hive if Spark and Hive are deployed in different clusters. Otherwise, please ignore the following configurations.

Parameter	Туре	Default value	Required	Description
hive.warehouse	string	-	Yes	The warehouse path in HDFS. Enclose the path in double quotes and start with hdfs://.
hive.connectionURL	string	-	Yes	The URL of a JDBC connection. For example, "jdbc:mysql:// 127.0.0.1:3306/hive_spark? characterEncoding=UTF-8".
hive.connectionDriverName	string	"com.mysql.jdbc.Driver"	Yes	The driver name.
hive.connectionUserName	list[string]	-	Yes	The username for connections.
hive.connectionPassword	list[string]	-	Yes	The account password.

## NebulaGraph configurations

Parameter	Туре	Default value	Required	Description
nebula.address.graph	list[string]	["127.0.0.1:9669"]	Yes	The addresses of all Graph services, including IPs and ports, separated by commas (,). Example: ["ip1:port1","ip2:port2","ip3:port3"].
nebula.address.meta	list[string]	["127.0.0.1:9559"]	Yes	The addresses of all Meta services, including IPs and ports, separated by commas (,). Example: ["ip1:port1","ip2:port2","ip3:port3"].
nebula.user	string	-	Yes	The username with write permissions for NebulaGraph.
nebula.pswd	string	-	Yes	The account password.
nebula.space	string	-	Yes	The name of the graph space where data needs to be imported.
nebula.ssl.enable.graph	bool	false	Yes	Enables the SSL encryption between Exchange and Graph services. If the value is true, the SSL encryption is enabled and the following SSL parameters take effect. If Exchange is run on a multi-machine cluster, you need to store the corresponding files in the same path on each machine when setting the following SSL-related paths.
nebula.ssl.sign	string	са	Yes	Specifies the SSL sign. Optional values are ca and self.
nebula.ssl.ca.param.caCrtFilePath	string	Specifies the storage path of the CA certificate. It takes effect when the value of nebula.ssl.sign is ca.		
nebula.ssl.ca.param.crtFilePath	string	"/path/ crtFilePath"	Yes	Specifies the storage path of the CRT certificate. It takes effect when the value of nebula.ssl.sign is ca.
nebula.ssl.ca.param.keyFilePath	string	"/path/ keyFilePath"	Yes	Specifies the storage path of the key file. It takes effect when the value of nebula.ssl.sign is ca.
nebula.ssl.self.param.crtFilePath	string	"/path/ crtFilePath"	Yes	Specifies the storage path of the CRT certificate. It takes effect when the value of nebula.ssl.sign is self.
nebula.ssl.self.param.keyFilePath	string	"/path/ keyFilePath"	Yes	Specifies the storage path of the key file. It takes effect when the value of nebula.ssl.sign is self.
nebula.ssl.self.param.password	string	"nebula"	Yes	Specifies the storage path of the password. It takes effect when the value of nebula.ssl.sign is self.
nebula.path.local	string	"/tmp"	No	

- 829/1069 - 2023 Vesoft Inc.

Parameter	Туре	Default value	Required	<b>Description</b> The local SST file path which needs to be set when users import SST files.
nebula.path.remote	string	"/sst"	No	The remote SST file path which needs to be set when users import SST files.
nebula.path.hdfs.namenode	string	"hdfs://name_node: 9000"	No	The NameNode path which needs to be set when users import SST files.
nebula.connection.timeout	int	3000	No	The timeout set for Thrift connections. Unit: ms.
nebula.connection.retry	int	3	No	Retries set for Thrift connections.
nebula.execution.retry	int	3	No	Retries set for executing nGQL statements.
nebula.error.max	int	32	No	The maximum number of failures during the import process. When the number of failures reaches the maximum, the Spark job submitted will stop automatically.
nebula.error.output	string	/tmp/errors	No	The path to output error logs. Failed nGQL statement executions are saved in the error log.
nebula.rate.limit	int	1024	No	The limit on the number of tokens in the token bucket when importing data.
nebula.rate.timeout	int	1000	No	The timeout period for getting tokens from a token bucket. Unit: milliseconds.



NebulaGraph doesn't support vertices without tags by default. To import vertices without tags, enable vertices without tags in the NebulaGraph cluster and then add parameter <code>nebula.enableTagless</code> to the Exchange configuration with the value <code>true</code>. For example:

```
nebula: {
    address: {
        graph: ["127.0.0.1:9669"]
        meta: ["127.0.0.1:9559"]
    }
    user: root
    pswd: nebula
    space: test
    enableTagless: true
    .....
}
```

## Vertex configurations

For different data sources, the vertex configurations are different. There are many general parameters and some specific parameters. General parameters and specific parameters of different data sources need to be configured when users configure vertices.

## GENERAL PARAMETERS

		Default value	Required	Description
tags.name	string	-	Yes	The tag name defined in NebulaGraph.
tags.type.source	string	-	Yes	Specify a data source. For example, $\ensuremath{csv}.$
tags.type.sink	string	client	Yes	Specify an import method. Optional values are $% \left( 1\right) =\left( 1\right) +\left( 1$
tags.fields	list[string]	-	Yes	The header or column name of the column corresponding to properties. If there is a header or a column name, please use that name directly. If a CSV file does not have a header, use the form of [_c0, _c1, _c2] to represent the first column, the second column, the third column, and so on.
tags.nebula.fields	list[string]	-	Yes	Property names defined in NebulaGraph, the order of which must correspond to tags.fields. For example, [_c1, _c2] corresponds to [name, age], which means that values in the second column are the values of the property name, and values in the third column are the values of the property age.
tags.vertex.field	string	-	Yes	The column of vertex IDs. For example, when a CSV file has no header, users can use _c0 to indicate values in the first column are vertex IDs.
tags.batch	int	256	Yes	The maximum number of vertices written into NebulaGraph in a single batch.
tags.partition	int	32	Yes	The number of Spark partitions.

# SPECIFIC PARAMETERS OF PARQUET/JSON/ORC DATA SOURCES

Parameter	Туре	Default value	Required	Description
tags.path	string	-	Yes	The path of vertex data files in HDFS. Enclose the path in double quotes and start with ${\tt hdfs://}$ .

# SPECIFIC PARAMETERS OF CSV DATA SOURCES

Parameter	Туре	Default value	Required	Description
tags.path	string	-	Yes	The path of vertex data files in HDFS. Enclose the path in double quotes and start with $hdfs://$ .
tags.separator	string	,	Yes	The separator. The default value is a comma (,). For special characters, such as the control character ^A, you can use ASCII octal \001 or UNICODE encoded hexadecimal \u00001, for the control character ^B, use ASCII octal \002 or UNICODE encoded hexadecimal \u00002, for the control character ^C, use ASCII octal \003 or UNICODE encoded hexadecimal \u00003.
tags.header	bool	true	Yes	Whether the file has a header.

## SPECIFIC PARAMETERS OF HIVE DATA SOURCES

Parameter	Туре	Default value	Required	Description
tags.exec	string	-	Yes	The statement to query data sources. For example, select name,age from mooc.users.

# SPECIFIC PARAMETERS OF MAXCOMPUTE DATA SOURCES

Parameter	Туре	Default value	Required	Description
tags.table	string	-	Yes	The table name of the MaxCompute.
tags.project	string	-	Yes	The project name of the MaxCompute.
tags.odpsUrl	string	-	Yes	The odpsUrl of the MaxCompute service. For more information about odpsUrl, see <a href="Endpoints">Endpoints</a> .
tags.tunnelUrl	string	-	Yes	The tunnelUrl of the MaxCompute service. For more information about tunnelUrl, see Endpoints.
tags.accessKeyId	string	-	Yes	The accessKeyId of the MaxCompute service.
tags.accessKeySecret	string	-	Yes	The accessKeySecret of the MaxCompute service.
tags.partitionSpec	string	-	No	Partition descriptions of MaxCompute tables.
tags.sentence	string	-	No	Statements to query data sources. The table name in the SQL statement is the same as the value of the table above.

## SPECIFIC PARAMETERS OF NEO4J DATA SOURCES

Parameter	Туре	Default value	Required	Description
tags.exec	string	-	Yes	Statements to query data sources. For example: match (n:label) return n.neo4j-field-0.
tags.server	string	"bolt:// 127.0.0.1:7687"	Yes	The server address of Neo4j.
tags.user	string	-	Yes	The Neo4j username with read permissions.
tags.password	string	-	Yes	The account password.
tags.database	string	-	Yes	The name of the database where source data is saved in Neo $4$ j.
tags.check_point_path	string	/tmp/test	No	The directory set to import progress information, which is used for resuming transfers. If not set, the resuming transfer is disabled.

# SPECIFIC PARAMETERS OF MYSQL/POSTGRESQL DATA SOURCES

Parameter	Туре	Default value	Required	Description
tags.host	string	-	Yes	The MySQL/PostgreSQL server address.
tags.port	string	-	Yes	The MySQL/PostgreSQL server port.
tags.database	string	-	Yes	The database name.
tags.table	string	-	Yes	The name of a table used as a data source.
tags.user	string	-	Yes	The MySQL/PostgreSQL username with read permissions.
tags.password	string	-	Yes	The account password.
tags.sentence	string	-	Yes	Statements to query data sources. For example: "select teamid, name from team order by teamid".

## SPECIFIC PARAMETERS OF ORACLE DATA SOURCES

Parameter	Туре	Default value	Required	Description
tags.url	string	-	Yes	The Oracle server address.
tags.driver	string	-	Yes	The Oracle driver address.
tags.user	string	-	Yes	The Oracle username with read permissions.
tags.password	string	-	Yes	The account password.
tags.table	string	-	Yes	The name of a table used as a data source.
tags.sentence	string	-	Yes	Statements to query data sources. For example: "select playerid, name, age from player".

## SPECIFIC PARAMETERS OF CLICKHOUSE DATA SOURCES

Parameter	Туре	Default value	Required	Description
tags.url	string	-	Yes	The JDBC URL of ClickHouse.
tags.user	string	-	Yes	The ClickHouse username with read permissions.
tags.password	string	-	Yes	The account password.
tags.numPartition	string	-	Yes	The number of ClickHouse partitions.
tags.sentence	string	-	Yes	Statements to query data sources.

## SPECIFIC PARAMETERS OF HBASE DATA SOURCES

Parameter	Туре	Default value	Required	Description
tags.host	string	127.0.0.1	Yes	The Hbase server address.
tags.port	string	2181	Yes	The Hbase server port.
tags.table	string	-	Yes	The name of a table used as a data source.
tags.columnFamily	string	-	Yes	The column family to which a table belongs.

# SPECIFIC PARAMETERS OF PULSAR DATA SOURCES

Parameter	Туре	Default value	Required	Description
tags.service	string	"pulsar:// localhost: 6650"	Yes	The Pulsar server address.
tags.admin	string	"http:// localhost: 8081"	Yes	The admin URL used to connect pulsar.
<pre>tags.options.<topic\  topics\ ="" topicspattern=""></topic\ ></pre>	string	-	Yes	Options offered by Pulsar, which can be configured by choosing one from topic, topics, and topicsPattern.
tags.interval.seconds	int	10	Yes	The interval for reading messages. Unit: seconds.

# SPECIFIC PARAMETERS OF KAFKA DATA SOURCES

Parameter	Туре	Default value	Required	Description
tags.service	string	-	Yes	The Kafka server address.
tags.topic	string	-	Yes	The message type.
tags.interval.seconds	int	10	Yes	The interval for reading messages. Unit: seconds.

# SPECIFIC PARAMETERS FOR GENERATING SST FILES

Parameter	Туре	Default value	Required	Description
tags.path	string	-	Yes	The path of the source file specified to generate SST files.
tags.repartitionWithNebula	bool	true	No	Whether to repartition data based on the number of partitions of graph spaces in NebulaGraph when generating the SST file. Enabling this function can reduce the time required to DOWNLOAD and INGEST SST files.



Specific parameters of NebulaGraph are used for exporting NebulaGraph data, which is supported by Exchange Enterprise Edition only.

Parameter	Data type	Default value	Required	Description
tags.path	string	"hdfs:// namenode: 9000/path/ vertex"	Yes	Specifies the storage path of the CSV file. You need to set a new path and Exchange will automatically create the path you set. If you store the data to the HDFS server, the path format is the same as the default value, such as "hdfs://192.168.8.177:9000/vertex/player". If you store the data to the local, the path format is "file:///path/vertex", such as "file:///home/nebula/vertex/player". If there are multiple Tags, different directories must be set for each Tag.
tags.noField	bool	false	Yes	If the value is true, only VIDs will be exported, not the property data. If the value is false, VIDs and the property data will be exported.
tags.return.fields	list	[]	Yes	Specifies the properties to be exported. For example, to export the name and age, you need to set the parameter value to ["name", "age"]. This parameter only takes effect when the value of tags.noField is false.

# Edge configurations

For different data sources, configurations of edges are also different. There are general parameters and some specific parameters. General parameters and specific parameters of different data sources need to be configured when users configure edges.

For the specific parameters of different data sources for edge configurations, please refer to the introduction of specific parameters of different data sources above, and pay attention to distinguishing tags and edges.

## GENERAL PARAMETERS

Parameter	Туре	Default value	Required	Description
edges.name	string	-	Yes	The edge type name defined in NebulaGraph.
edges.type.source	string	-	Yes	The data source of edges. For example, $\ensuremath{csv}$ .
edges.type.sink	string	client	Yes	The method specified to import data. Optional values are client and SST.
edges.fields	list[string]	-	Yes	The header or column name of the column corresponding to properties. If there is a header or column name, please use that name directly. If a CSV file does not have a header, use the form of [_c0, _c1, _c2] to represent the first column, the second column, the third column, and so on.
edges.nebula.fields	list[string]	-	Yes	Edge names defined in NebulaGraph, the order of which must correspond to edges.fields. For example, [_c2, _c3] corresponds to [start_year, end_year], which means that values in the third column are the values of the start year, and values in the fourth column are the values of the end year.
edges.source.field	string	-	Yes	The column of source vertices of edges. For example, _c0 indicates a value in the first column that is used as the source vertex of an edge.
edges.target.field	string	-	Yes	The column of destination vertices of edges. For example, _c0 indicates a value in the first column that is used as the destination vertex of an edge.
edges.ranking	int	-	No	The column of rank values. If not specified, all rank values are $\ensuremath{\text{0}}$ by default.
edges.batch	int	256	Yes	The maximum number of edges written into NebulaGraph in a single batch.
edges.partition	int	32	Yes	The number of Spark partitions.

# SPECIFIC PARAMETERS FOR GENERATING SST FILES

Parameter	Туре	Default value	Required	Description
edges.path	string	-	Yes	The path of the source file specified to generate SST files.
edges.repartitionWithNebula	bool	true	No	Whether to repartition data based on the number of partitions of graph spaces in NebulaGraph when generating the SST file. Enabling this function can reduce the time required to DOWNLOAD and INGEST SST files.

## SPECIFIC PARAMETERS OF NEBULAGRAPH

Parameter	Туре	Default value	Required	Description
edges.path	string	"hdfs:// namenode: 9000/path/ edge"	Yes	Specifies the storage path of the CSV file. You need to set a new path and Exchange will automatically create the path you set. If you store the data to the HDFS server, the path format is the same as the default value, such as "hdfs://192.168.8.177:9000/edge/follow". If you store the data to the local, the path format is "file:///path/edge", such as "file:///home/nebula/edge/follow". If there are multiple Edges, different directories must be set for each Edge.
edges.noField	bool	false	Yes	If the value is true, source vertex IDs, destination vertex IDs, and ranks will be exported, not the property data. If the vaue is false, ranks, source vertex IDs, destination vertex IDs, ranks, and the property data will be exported.
edges.return.fields	list	D .	Yes	Specifies the properties to be exported. For example, to export start_year and end_year, you need to set the parameter value to ["start_year","end_year"]. This parameter only takes effect when the value of edges.noField is false.

# 18.4 Use NebulaGraph Exchange

## 18.4.1 Import data from CSV files

This topic provides an example of how to use Exchange to import NebulaGraph data stored in HDFS or local CSV files.

To import a local CSV file to NebulaGraph, see NebulaGraph Importer.

#### Data set

This topic takes the basketballplayer dataset as an example.

#### **Environment**

This example is done on MacOS. Here is the environment configuration information:

- Hardware specifications:
- CPU: 1.7 GHz Quad-Core Intel Core i7
- Memory: 16 GB
- Spark: 2.4.7, stand-alone
- Hadoop: 2.9.2, pseudo-distributed deployment
- NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

#### **Prerequisites**

Before importing data, you need to confirm the following information:

- $\bullet$  Nebula Graph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- $\bullet$  The user name and password with write permission to Nebula Graph.
- Exchange has been compiled, or download the compiled .jar file directly.
- Spark has been installed.
- · Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- If files are stored in HDFS, ensure that the Hadoop service is running normally.
- If files are stored locally and NebulaGraph is a cluster architecture, you need to place the files in the same directory locally on each machine in the cluster.

- 838/1069 - 2023 Vesoft Inc.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: PROCESS CSV FILES

Confirm the following information:

1. Process CSV files to meet Schema requirements.



Exchange supports uploading CSV files with or without headers.

2. Obtain the CSV file storage path.

```
STEP 3: MODIFY CONFIGURATION FILES
```

After Exchange is compiled, copy the conf file target/classes/application.conf to set CSV data source configuration. In this example, the copied file is called csv\_application.conf. For details on each configuration item, see Parameters in the configuration file.

```
{
  # Spark configuration
  spark: {
    app: {
        name: NebulaGraph Exchange 3.4.0
    }
    driver: {
        cores: 1
        maxResultSize: 16
    }
    executor: {
        memory:16
    }
    cores: {
```

```
max: 16
# NebulaGraph configuration
nebula: {
  address:{
     # Specify the IP addresses and ports for Graph and Meta services.
    # If there are multiple addresses, the format is "ip1:port","ip2:port","ip3:port". # Addresses are separated by commas.
    graph:["127.0.0.1:9669"]
     # the address of any of the meta services.
# if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
    meta:["127.0.0.1:9559"]
  # The account entered must have write permission for the NebulaGraph space.
  user: root
  pswd: nebula
  # Fill in the name of the graph space you want to write data to in the NebulaGraph.
   space: basketballplayer
  connection: {
    timeout: 3000
     retry: 3
  execution: {
    retry: 3
  error: {
    max· 32
    output: /tmp/errors
  rate: {
     limit: 1024
     timeout: 1000
# Processing vertexes
  # Set the information about the Tag player.
     # Specify the Tag name defined in NebulaGraph.
     name: player
     type: {
       # Specify the data source file format to CSV.
      source: csv
      # Specify how to import the data into NebulaGraph: Client or SST.
      sink: client
     # Specify the path to the CSV file.
    # If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example: "hdfs://ip:port/xx/xx".
# If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example: "file:///tmp/xx.csv".
path: "hdfs://192.168.*.*:9000/data/vertex_player.csv"
     # If the CSV file does not have a header, use [_c0, _c1, _c2, ..., _cn] to represent its header and indicate the columns as the source of the property values. # If the CSV file has headers, use the actual column names.
     fields: [_c1, _c2]
     # Specify the column names in the player table in fields, and their corresponding values are specified as properties in the NebulaGraph. # The sequence of fields and nebula.fields must correspond to each other.
     nebula.fields: [age, name]
     # Specify a column of data in the table as the source of vertex VID in the NebulaGraph.
     # The value of vertex must be the same as the column names in the above fields or csv.fields.
     # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
     vertex: {
      field: c0
       # policy:hash
     # The delimiter specified. The default value is comma.
     # If the CSV file has a header, set the header to true.
     # If the CSV file does not have a header, set the header to false. The default value is false.
    header: false
     # The number of data written to NebulaGraph in a single batch.
    batch: 256
     # The number of Spark partitions.
    partition: 32
  # Set the information about the Tag Team.
    # Specify the Tag name defined in NebulaGraph.
     name: team
     type: {
      # Specify the data source file format to CSV.
```

```
source: csv
      # Specify how to import the data into NebulaGraph: Client or SST.
      sink: client
    # Specify the path to the CSV file.
    # If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example: "hdfs://ip:port/xx/xx".
    # If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example: "file:///tmp/xx.csv" path: "hdfs://192.168.*.*:9000/data/vertex_team.csv"
    # If the CSV file does not have a header, use [_c0, _c1, _c2, ..., _cn] to represent its header and indicate the columns as the source of the property values. # If the CSV file has headers, use the actual column names.
    fields: [_c1]
    # Specify the column names in the player table in fields, and their corresponding values are specified as properties in the NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [name]
    # Specify a column of data in the table as the source of VIDs in the NebulaGraph.
# The value of vertex must be the same as the column names in the above fields or csv.fields.
     # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    vertex: {
      field:_c0
      # policy:hash
    # The delimiter specified. The default value is comma.
    separator: ",
    # If the CSV file has a header, set the header to true.
# If the CSV file does not have a header, set the header to false. The default value is false.
    header: false
     # The number of data written to NebulaGraph in a single batch.
    batch: 256
    # The number of Spark partitions.
    partition: 32
  # If more vertexes need to be added, refer to the previous configuration to add them.
# Processing edges
edges: [
  # Set the information about the Edge Type follow.
    # Specify the Edge Type name defined in NebulaGraph.
    name: follow
    type: {
      # Specify the data source file format to CSV.
      source: csv
      # Specify how to import the data into NebulaGraph: Client or SST.
      sink: client
    # Specify the path to the CSV file.
    # If the file is stored in HDFs, use double quotation marks to enclose the file path, starting with hdfs://. For example: "hdfs://ip:port/xx/xx".

# If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example: "file:///tmp/xx.csv".

path: "hdfs://192.168.*.*:9000/data/edge_follow.csv"
    # If the CSV file does not have a header, use [_c0, _c1, _c2, ..., _cn] to represent its header and indicate the columns as the source of the property values. # If the CSV file has headers, use the actual column names.
    fields: [ c2]
    # Specify the column names in the edge table in fields, and their corresponding values are specified as properties in the NebulaGraph.
    # The sequence of fields and nebula fields must correspond to each other.
    nebula.fields: [degree]
    # Specify a column as the source for the source and destination vertexes.
     # The value of vertex must be the same as the column names in the above fields or csv.fields.
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    source: {
      field: _c0
    target: {
      field: _c1
    # The delimiter specified. The default value is comma.
    separator: ",
    # Specify a column as the source of the rank (optional).
    #ranking: rank
    # If the CSV file has a header, set the header to true.
     # If the CSV file does not have a header, set the header to false. The default value is false.
    header: false
    # The number of data written to NebulaGraph in a single batch.
    batch: 256
```

```
# The number of Spark partitions.
# Set the information about the Edge Type serve.
  # Specify the Edge Type name defined in NebulaGraph.
  type: {
   # Specify the data source file format to CSV.
    # Specify how to import the data into NebulaGraph: Client or SST.
   sink: client
 # Specify the path to the CSV file.
  # If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example: "hdfs://ip:port/xx/xx".
 # If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example: "file:///tmp/xx.csv".path: "hdfs://192.168.*.*:9000/data/edge_serve.csv"
 # If the CSV file does not have a header, use [_c0, _c1, _c2, ..., _cn] to represent its header and indicate the columns as the source of the property values. # If the CSV file has headers, use the actual column names.
  fields: [_c2,_c3]
  # Specify the column names in the edge table in fields, and their corresponding values are specified as properties in the NebulaGraph.
  # The sequence of fields and nebula.fields must correspond to each other
  nebula.fields: [start_year, end_year]
  # Specify a column as the source for the source and destination vertexes.
# The value of vertex must be the same as the column names in the above fields or csv.fields.
  # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
  source: {
    field: _c0
  target: {
    field: _c1
  # The delimiter specified. The default value is comma
  separator: ",
  # Specify a column as the source of the rank (optional).
  #ranking: _c5
  # If the CSV file has a header, set the header to true.
  # If the CSV file does not have a header, set the header to false. The default value is false.
  # The number of data written to NebulaGraph in a single batch.
  batch: 256
  # The number of Spark partitions.
  partition: 32
# If more edges need to be added, refer to the previous configuration to add them.
```

STEP 4: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import CSV data into NebulaGraph. For descriptions of the parameters, see Options for import.

```
${$PARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.4.0.jar_path> -c <csv_application.conf_path>
```



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

# For example:

\${\$PARK\_HOWE}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/classes/csv\_application.conf

You can search for batchSuccess.<tag\_name/edge\_name> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

## STEP 5: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

LOOKUP ON player YIELD id(vertex);

Users can also run the SHOW STATS command to view statistics.

STEP 6: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

# 18.4.2 Import data from JSON files

This topic provides an example of how to use Exchange to import NebulaGraph data stored in HDFS or local JSON files.

#### Data set

This topic takes the basketballplayer dataset as an example. Some sample data are as follows:

• player

```
{"id":"player100", "age":42, "name":"Tim Duncan"}
{"id":"player101", "age":36, "name":"Tony Parker"}
{"id":"player102", "age":33, "name":"LuMarcus Aldridge"}
{"id":"player103", "age":32, "name":"Rudy Gay"}
...
```

• team

```
{"id":"team200","name":"Warriors"}
{"id":"team201","name":"Nuggets"}
...
```

• follow

```
{"src":"player100","dst":"player101","degree":95}
{"src":"player101","dst":"player102","degree":90}
...
```

• serve

```
{"src":"player100","dst":"team204","start_year":"1997","end_year":"2016"}
{"src":"player101","dst":"team204","start_year":"1999","end_year":"2018"}
...
```

# Environment

This example is done on MacOS. Here is the environment configuration information:

- · Hardware specifications:
- CPU: 1.7 GHz Quad-Core Intel Core i7
- Memory: 16 GB
- Spark: 2.4.7, stand-alone
- Hadoop: 2.9.2, pseudo-distributed deployment
- NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

#### **Prerequisites**

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- Exchange has been compiled, or download the compiled .jar file directly.
- · Spark has been installed.
- · Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- If files are stored in HDFS, ensure that the Hadoop service is running properly.
- If files are stored locally and NebulaGraph is a cluster architecture, you need to place the files in the same directory locally on each machine in the cluster.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: PROCESS JSON FILES

Confirm the following information:

- 1. Process JSON files to meet Schema requirements.
- 2. Obtain the JSON file storage path.

#### STEP 3: MODIFY CONFIGURATION FILES

After Exchange is compiled, copy the conf file target/classes/application.conf to set JSON data source configuration. In this example, the copied file is called <code>json\_application.conf</code>. For details on each configuration item, see Parameters in the configuration file.

```
# Spark configuration
spark: {
  app: {
    name: NebulaGraph Exchange 3.4.0
  driver: {
    cores: 1
    maxResultSize: 1G
  executor: {
     memory:1G
  cores: {
    max: 16
# NebulaGraph configuration
nebula: {
    # Specify the IP addresses and ports for Graph and all Meta services.
    # If there are multiple addresses, the format is "ip1:port","ip2:port","ip3:port". # Addresses are separated by commas.
    graph:["127.0.0.1:9669"]
    # the address of any of the meta services.
    # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
    meta:["127.0.0.1:9559"]
  # The account entered must have write permission for the NebulaGraph space.
  user: root
  pswd: nebula
  # Fill in the name of the graph space you want to write data to in the NebulaGraph.
  space: basketballplayer
  connection: {
    timeout: 3000
   retry: 3
  execution: {
    retry: 3
  error: {
    max: 32
    output: /tmp/errors
  rate: {
    limit: 1024
    timeout: 1000
# Processing vertexes
tags: [ # Set the information about the Tag player.
    # Specify the Tag name defined in NebulaGraph.
    name: player
      # Specify the data source file format to JSON.
     # Specify how to import the data into NebulaGraph: Client or SST.
    # Specify the path to the JSON file.
    # If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx".
    # If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example, "file://tmp/xx.json"
    path: "hdfs://192.168.*.*:9000/data/vertex_player.json"
    # Specify the key name in the JSON file in fields, and its corresponding value will serve as the data source for the properties specified in the NebulaGraph.
    # If multiple column names need to be specified, separate them by commas.
    fields: [age,name]
    # Specify the column names in the player table in fields, and their corresponding values are specified as properties in the NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [age, name]
    # Specify a column of data in the table as the source of vertex VID in the NebulaGraph.
    # The value of vertex must be the same as that in the JSON file.
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
      field:id
    }
```

```
# The number of data written to NebulaGraph in a single batch.
    # The number of Spark partitions.
    partition: 32
  # Set the information about the Tag Team.
    # Specify the Tag name defined in NebulaGraph.
    type: {
      # Specify the data source file format to JSON.
     source: ison
     # Specify how to import the data into NebulaGraph: Client or SST.
     sink: client
    # Specify the path to the JSON file.
    # If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx".
   # If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example, "file:///tmp/xx.json". path: "hdfs://192.168.*.*:9000/data/vertex_team.json"
    # Specify the key name in the JSON file in fields, and its corresponding value will serve as the data source for the properties specified in the NebulaGraph.
    # If multiple column names need to be specified, separate them by commas.
    fields: [name]
    # Specify the column names in the player table in fields, and their corresponding values are specified as properties in the NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [name]
    # Specify a column of data in the table as the source of vertex VID in the NebulaGraph. # The value of vertex must be the same as that in the JSON file.
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    vertex: {
     field:id
    }
    # The number of data written to NebulaGraph in a single batch.
    # The number of Spark partitions.
   partition: 32
 # If more vertexes need to be added, refer to the previous configuration to add them.
# Processing edges
edges: [
  # Set the information about the Edge Type follow.
    # Specify the Edge Type name defined in NebulaGraph.
    type: {
     # Specify the data source file format to JSON.
     source: json
     # Specify how to import the data into NebulaGraph: Client or SST.
     sink: client
    # Specify the path to the JSON file.
# If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx".
    # If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example, "file:///tmp/xx.json".path: "hdfs://192.168.*.*:9900/data/edge_follow.json"
    # Specify the key name in the JSON file in fields, and its corresponding value will serve as the data source for the properties specified in the NebulaGraph.
    # If multiple column names need to be specified, separate them by commas.
    fields: [degree]
    # Specify the column names in the edge table in fields, and their corresponding values are specified as properties in the NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [degree]
    # Specify a column as the source for the source and destination vertexes.
    # The value of vertex must be the same as that in the JSON file
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    source: {
   field: src
    target: {
     field: dst
    # (Optional) Specify a column as the source of the rank.
    #ranking: rank
    # The number of data written to NebulaGraph in a single batch.
    batch: 256
```

```
# The number of Spark partitions.
  # Set the information about the Edge Type serve.
    # Specify the Edge type name defined in NebulaGraph.
    type: {
     # Specify the data source file format to JSON.
     source: json
      # Specify how to import the data into NebulaGraph: Client or SST.
     sink: client
    # Specify the path to the JSON file.
    # If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx"
   # If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example, "file:///tmp/xx.json".path: "hdfs://192.168.*.*:9000/data/edge_serve.json"
    # Specify the key name in the JSON file in fields, and its corresponding value will serve as the data source for the properties specified in the NebulaGraph.
    # If multiple column names need to be specified, separate them by commas
    fields: [start_year,end_year]
    # Specify the column names in the edge table in fields, and their corresponding values are specified as properties in the NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other
    nebula.fields: [start_year, end_year]
    # Specify a column as the source for the source and destination vertexes.
# The value of vertex must be the same as that in the JSON file.
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    source: {
      field: src
    target: {
     field: dst
    # (Optional) Specify a column as the source of the rank.
    #ranking: _c5
    # The number of data written to NebulaGraph in a single batch.
   batch: 256
    # The number of Spark partitions
   partition: 32
# If more edges need to be added, refer to the previous configuration to add them.
```

STEP 4: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import JSON data into NebulaGraph. For a description of the parameters, see Options for import.

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange=3.4.0.jar\_path> -c <json\_application.conf\_path>



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

#### For example:

```
${$PARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/classes/json_application.conf
```

You can search for batchSuccess.<tag\_name/edge\_name> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 5: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

```
LOOKUP ON player YIELD id(vertex);
```

Users can also run the  ${\mbox{\scriptsize SHOW STATS}}$  command to view statistics.

STEP 6: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

# 18.4.3 Import data from ORC files

This topic provides an example of how to use Exchange to import NebulaGraph data stored in HDFS or local ORC files.

To import a local ORC file to NebulaGraph, see NebulaGraph Importer.

#### Data set

This topic takes the basketballplayer dataset as an example.

#### Environment

This example is done on MacOS. Here is the environment configuration information:

- Hardware specifications:
- CPU: 1.7 GHz Quad-Core Intel Core i7
- Memory: 16 GB
- Spark: 2.4.7, stand-alone
- Hadoop: 2.9.2, pseudo-distributed deployment
- NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

#### **Prerequisites**

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- Exchange has been compiled, or download the compiled .jar file directly.
- Spark has been installed.
- Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- If files are stored in HDFS, ensure that the Hadoop service is running properly.
- If files are stored locally and NebulaGraph is a cluster architecture, you need to place the files in the same directory locally on each machine in the cluster.

- 850/1069 - 2023 Vesoft Inc.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: PROCESS ORC FILES

Confirm the following information:

- 1. Process ORC files to meet Schema requirements.
- 2. Obtain the ORC file storage path.

STEP 3: MODIFY CONFIGURATION FILES

After Exchange is compiled, copy the conf file target/classes/application.conf to set ORC data source configuration. In this example, the copied file is called orc\_application.conf. For details on each configuration item, see Parameters in the configuration file.

```
# Addresses are separated by commas.
    graph:["127.0.0.1:9669"]
      the address of any of the meta services.
    # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
    meta:["127.0.0.1:9559"]
  # The account entered must have write permission for the NebulaGraph space.
 user: root
pswd: nebula
  # Fill in the name of the graph space you want to write data to in the NebulaGraph.
  space: basketballplayer
  connection: {
    timeout: 3000
    retry: 3
  execution: {
   retry: 3
 error: {
    max: 32
   output: /tmp/errors
    limit: 1024
    timeout: 1000
# Processing vertexes
tags: [
  # Set the information about the Tag player.
    name: player
      # Specify the data source file format to ORC.
      # Specify how to import the data into NebulaGraph: Client or SST.
   # Specify the path to the ORC file.
# If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx".
# If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example, "file:///tmp/xx.orc".
path: "hdfs://192.168.*.*:9000/data/vertex_player.orc"
    # Specify the key name in the ORC file in fields, and its corresponding value will serve as the data source for the properties specified in the NebulaGraph.
    # If multiple values need to be specified, separate them with commas.
    fields: [age,name]
    # Specify the property names defined in NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [age, name]
    # Specify a column of data in the table as the source of VIDs in the NebulaGraph.
    # The value of vertex must be consistent with the field in the ORC file.
# Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
      field:id
    # The number of data written to NebulaGraph in a single batch.
    # The number of Spark partitions.
  # Set the information about the Tag team.
    # Specify the Tag name defined in NebulaGraph.
    name: team
    type: {
      # Specify the data source file format to ORC.
      source: orc
      # Specify how to import the data into NebulaGraph: Client or SST.
      sink: client
    # Specify the path to the ORC file.
    # If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx".
   # If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example, "file:///tmp/xx.orc" path: "hdfs://192.168.*.*:9000/data/vertex_team.orc"
    # Specify the key name in the ORC file in fields, and its corresponding value will serve as the data source for the properties specified in the NebulaGraph.
    # If multiple values need to be specified, separate them with commas.
    fields: [name]
    # Specify the property names defined in NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [name]
```

```
# Specify a column of data in the table as the source of VIDs in the NebulaGraph.
    # The value of vertex must be consistent with the field in the ORC file.
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    vertex: {
      field:id
    }
    # The number of data written to NebulaGraph in a single batch.
    # The number of Spark partitions.
   partition: 32
  # If more vertexes need to be added, refer to the previous configuration to add them.
# Processing edges
  # Set the information about the Edge Type follow.
    # Specify the Edge Type name defined in NebulaGraph.
    name: follow
    type: {
       # Specify the data source file format to ORC.
      source: orc
      # Specify how to import the data into NebulaGraph: Client or SST.
      sink: client
    # Specify the path to the ORC file.
    # If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx". # If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example, "file:///tmp/xx.orc". path: "hdfs://192.168.*.*:9000/data/edge_follow.orc"
    # Specify the key name in the ORC file in fields, and its corresponding value will serve as the data source for the properties specified in the NebulaGraph.
     # If multiple values need to be specified, separate them with commas.
    fields: [degree]
    # Specify the property names defined in NebulaGraph.
     # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [degree]
    # Specify a column as the source for the source and destination vertexes.
     # The value of vertex must be consistent with the field in the ORC file.
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    source: {
      field: src
    target: {
      field: dst
    # (Optional) Specify a column as the source of the rank.
    #ranking: rank
    # The number of data written to NebulaGraph in a single batch.
    # The number of Spark partitions.
    partition: 32
  # Set the information about the Edge type serve.
    # Specify the Edge type name defined in NebulaGraph.
    type: {
       # Specify the data source file format to ORC.
      source: orc
       # Specify how to import the data into NebulaGraph: Client or SST.
      sink: client
    # Specify the path to the ORC file.
    # If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx".
    # If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example, "file:///tmp/xx.orc" path: "hdfs://192.168.*.*:9000/data/edge_serve.orc"
    # Specify the key name in the ORC file in fields, and its corresponding value will serve as the data source for the properties specified in the NebulaGraph. # If multiple values need to be specified, separate them with commas.
     fields: [start_year,end_year]
    # Specify the property names defined in NebulaGraph.
     # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [start_year, end_year]
    # Specify a column as the source for the source and destination vertexes. # The value of vertex must be consistent with the field in the ORC file.
```

```
# Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
source: {
    field: src
}
target: {
    field: dst
}

# (Optional) Specify a column as the source of the rank.
#ranking: _c5

# The number of data written to NebulaGraph in a single batch.
batch: 256

# The number of Spark partitions.
partition: 32
}

# If more edges need to be added, refer to the previous configuration to add them.
}
```

STEP 4: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import ORC data into NebulaGraph. For a description of the parameters, see Options for import.

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <-nebula-exchange<-3.4.0.jar\_path> -c <orc\_application.conf\_path>



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

## For example:

\${\$PARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.4.0.jar -c /root/nebula-exchange/nebula-exchange/target/classes/orc\_application.conf

You can search for batchSuccess.<ag\_name/edge\_name> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 5: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

```
LOOKUP ON player YIELD id(vertex);
```

Users can also run the SHOW STATS command to view statistics.

STEP 6: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

# 18.4.4 Import data from Parquet files

This topic provides an example of how to use Exchange to import NebulaGraph data stored in HDFS or local Parquet files.

To import a local Parquet file to NebulaGraph, see NebulaGraph Importer.

#### Data set

This topic takes the basketballplayer dataset as an example.

#### Environment

This example is done on MacOS. Here is the environment configuration information:

- Hardware specifications:
- CPU: 1.7 GHz Quad-Core Intel Core i7
- Memory: 16 GB
- Spark: 2.4.7, stand-alone
- Hadoop: 2.9.2, pseudo-distributed deployment
- NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

#### **Prerequisites**

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- Exchange has been compiled, or download the compiled .jar file directly.
- Spark has been installed.
- Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- If files are stored in HDFS, ensure that the Hadoop service is running properly.
- If files are stored locally and NebulaGraph is a cluster architecture, you need to place the files in the same directory locally on each machine in the cluster.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: PROCESS PARQUET FILES

Confirm the following information:

- 1. Process Parquet files to meet Schema requirements.
- 2. Obtain the Parquet file storage path.

```
STEP 3: MODIFY CONFIGURATION FILES
```

After Exchange is compiled, copy the conf file target/classes/application.conf to set Parquet data source configuration. In this example, the copied file is called parquet\_application.conf. For details on each configuration item, see Parameters in the configuration file.

```
# Specify the IP addresses and ports for Graph and all Meta services.
# If there are multiple addresses, the format is "ip1:port","ip2:port","ip3:port".
    # Addresses are separated by commas.
    graph:["127.0.0.1:9669"]
    # the address of any of the meta services.
    # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
    meta:["127.0.0.1:9559"]
  # The account entered must have write permission for the NebulaGraph space.
  pswd: nebula
  # Fill in the name of the graph space you want to write data to in the NebulaGraph.
  space: basketballplayer
  connection: {
    timeout: 3000
    retry: 3
  execution: {
   retry: 3
  error: {
   max: 32
    output: /tmp/errors
 rate: {
    limit: 1024
    timeout: 1000
# Processing vertexes
  # Set the information about the Tag player.
    # Specify the Tag name defined in NebulaGraph.
    name: player
    type: {
     # Specify the data source file format to Parquet.
      # Specifies how to import the data into NebulaGraph: Client or SST.
      sink: client
    # Specify the path to the Parquet file.
# If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx"
    # If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example, "file:///tmp/xx.parquet".
    path: "hdfs://192.168.*.13:9000/data/vertex player.parquet"
    # Specify the key name in the Parquet file in fields, and its corresponding value will serve as the data source for the properties specified in the NebulaGraph.
    # If multiple values need to be specified, separate them with commas.
    fields: [age,name]
    # Specify the property name defined in NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [age, name]
    # Specify a column of data in the table as the source of VIDs in the NebulaGraph.
    # The value of vertex must be consistent with the field in the Parquet file.
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    vertex: {
     field:id
    # The number of data written to NebulaGraph in a single batch.
    batch: 256
    # The number of Spark partitions.
    partition: 32
  # Set the information about the Tag team
    # Specify the Tag name defined in NebulaGraph.
    name: team
    type: {
      # Specify the data source file format to Parquet.
     source: parquet
     # Specifies how to import the data into NebulaGraph: Client or SST.
     sink: client
    # Specify the path to the Parquet file.
    # If the file is stored locally, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx".
# If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example, "file:///tmp/xx.parquet".
    path: "hdfs://192.168.11.13:9000/data/vertex_team.parquet"
    # Specify the key name in the Parquet file in fields, and its corresponding value will serve as the data source for the properties specified in the NebulaGraph.
    # If multiple values need to be specified, separate them with commas.
    fields: [name]
```

```
# Specify the property name defined in NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [name]
    # Specify a column of data in the table as the source of VIDs in the NebulaGraph.
    # The value of vertex must be consistent with the field in the Parquet file.
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
      field:id
    }
    # The number of data written to NebulaGraph in a single batch.
    # The number of Spark partitions.
   partition: 32
  # If more vertexes need to be added, refer to the previous configuration to add them.
# Processing edges
edges: [
  # Set the information about the Edge Type follow.
    # Specify the Edge Type name defined in NebulaGraph.
    name: follow
    type: {
      # Specify the data source file format to Parquet.
      source: parquet
      # Specifies how to import the data into NebulaGraph: Client or SST.
      sink: client
    # Specify the path to the Parquet file.
    # If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx".
   # If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example, "file:///tmp/xx.parquet".path: "hdfs://192.168.11.13:9000/data/edge_follow.parquet"
    # Specify the key name in the Parquet file in fields, and its corresponding value will serve as the data source for the properties specified in the NebulaGraph.
    # If multiple values need to be specified, separate them with commas.
    fields: [degree]
    # Specify the property name defined in NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.nebula.fields: [degree]
    # Specify a column as the source for the source and destination vertexes.
# The values of vertex must be consistent with the fields in the Parquet file.
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    source: {
      field: src
    target: {
    # (Optional) Specify a column as the source of the rank.
    # The number of data written to NebulaGraph in a single batch.
    # The number of Spark partitions.
    partition: 32
  # Set the information about the Edge type serve.
    # Specify the Edge type name defined in NebulaGraph.
    name: serve
      # Specify the data source file format to Parquet.
      # Specifies how to import the data into NebulaGraph: Client or SST.
    # Specify the path to the Parquet file.
# If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx"
    # If the file is stored locally, use double quotation marks to enclose the file path, starting with file://. For example, "file:///tmp/xx.parquet".
    path: "hdfs://192.168.11.13:9000/data/edge_serve.parquet"
    # Specify the key name in the Parquet file in fields, and its corresponding value will serve as the data source for the properties specified in the NebulaGraph.
    # If multiple values need to be specified, separate them with commas.
    fields: [start_year,end_year]
    # Specify the property name defined in NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [start_year, end_year]
```

```
# Specify a column as the source for the source and destination vertexes.
# The values of vertex must be consistent with the fields in the Parquet file.
# Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
source: {
    field: src
    }
    target: {
        field: dst
    }

# (Optional) Specify a column as the source of the rank.
#ranking: _c5

# The number of data written to NebulaGraph in a single batch.
batch: 256

# The number of Spark partitions.
partition: 32
}

# If more edges need to be added, refer to the previous configuration to add them.
}
```

STEP 4: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import Parquet data into NebulaGraph. For a description of the parameters, see Options for import.

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <-nebula-exchange<-3.4.0.jar\_path> -c <parquet\_application.conf\_path>



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

## For example:

\${\$PARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.4.0.jar -c /root/nebula-exchange/nebula-exchange/target/classes/parquet\_application.conf

You can search for batchSuccess.<tag\_name/edge\_name> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 5: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

```
LOOKUP ON player YIELD id(vertex);
```

Users can also run the **SHOW STATS** command to view statistics.

STEP 6: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see  $\underline{\text{Index overview}}$ .

# 18.4.5 Import data from HBase

This topic provides an example of how to use Exchange to import NebulaGraph data stored in HBase.

#### Data set

This topic takes the basketballplayer dataset as an example.

In this example, the data set has been stored in HBase. All vertexes and edges are stored in the player, team, follow, and serve tables. The following are some of the data for each table.

```
hbase(main):002:0> scan "player"
                                            COLUMN+CELL
player100
                                            column=cf:age, timestamp=1618881347530, value=42
                                            column=cf:name, timestamp=1618881354604, value=Tim Duncan
 player100
                                            column=cf:age, timestamp=1618881369124, value=36 column=cf:name, timestamp=1618881379102, value=Tony Parker
 player101
 player101
                                            column=cf:age, timestamp=1618881386987, value=33
 player102
 player102
player103
                                            column=cf:name, timestamp=1618881393370, value=LaMarcus Aldridge
column=cf:age, timestamp=1618881402002, value=32
                                            column=cf:name, timestamp=1618881407882, value=Rudy Gay
 player103
hbase(main):003:0> scan "team"
                                            COLUMN+CELL
 team200
                                            column=cf:name, timestamp=1618881445563, value=Warriors
 team201
                                            column=cf:name, timestamp=1618881453636, value=Nuggets
hbase(main):004:0> scan "follow"
                                            COLUMN+CELL
                                            column=cf:degree, timestamp=1618881804853, value=95 column=cf:dst_player, timestamp=1618881791522, value=player101 column=cf:degree, timestamp=1618881824685, value=90
player100
 player100
 player101
 player101
                                            column=cf:dst_player, timestamp=1618881816042, value=player102
hbase(main):005:0> scan "serve"
 player100
                                            column=cf:end_year, timestamp=1618881899333, value=2016
 player100
player100
                                            column=cf:start_year, timestamp=1618881890117, value=1997
column=cf:teamid, timestamp=1618881875739, value=team204
```

# Environment

This example is done on MacOS. Here is the environment configuration information:

- Hardware specifications:
- CPU: 1.7 GHz Quad-Core Intel Core i7
- Memory: 16 GB
- Spark: 2.4.7, stand-alone
- Hadoop: 2.9.2, pseudo-distributed deployment
- HBase: 2.2.7
- $\bullet \ \ Nebula Graph: 3.4.1. \ Deploy \ \ Nebula Graph \ \ with \ \ Docker \ Compose.$

#### **Prerequisites**

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- Exchange has been compiled, or download the compiled .jar file directly.
- · Spark has been installed.
- · Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- The Hadoop service has been installed and started.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

```
STEP 2: MODIFY CONFIGURATION FILES
```

After Exchange is compiled, copy the conf file target/classes/application.conf to set HBase data source configuration. In this example, the copied file is called hbase\_application.conf. For details on each configuration item, see Parameters in the configuration file.

```
{
  # Spark configuration
spark: {
  app: {
    name: NebulaGraph Exchange 3.4.0
  }
  driver: {
```

```
cores: 1
    maxResultSize: 1G
  cores: {
    max: 16
# NebulaGraph configuration
    # Specify the IP addresses and ports for Graph and all Meta services.
    # If there are multiple addresses, the format is "ip1:port", "ip2:port", "ip3:port".
    # Addresses are separated by commas.
    graph:["127.0.0.1:9669"]
    # the address of any of the meta services.
    # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
  # The account entered must have write permission for the NebulaGraph space.
  pswd: nebula
  # Fill in the name of the graph space you want to write data to in the NebulaGraph.
  space: basketballplayer
  connection: {
    timeout: 3000
    retry: 3
  execution: {
    retry: 3
  error: {
    max: 32
    output: /tmp/errors
  rate: {
    limit: 1024
    timeout: 1000
# Processing vertexes
tags: [
  # Set information about Tag player.
# If you want to set RowKey as the data source, enter rowkey and the actual column name of the column family.
    # The Tag name in NebulaGraph.
    name: player
      # Specify the data source file format to HBase.
      source: hbase
      # Specify how to import the data into NebulaGraph: Client or SST.
      sink: client
    host:192.168.*.*
    port:2181
    table:"player"
columnFamily:"cf"
    # Specify the column names in the player table in fields, and their corresponding values are specified as properties in the NebulaGraph. # The sequence of fields and nebula.fields must correspond to each other.
    # If multiple column names need to be specified, separate them by commas.
    fields: [age,name]
nebula.fields: [age,name]
    # Specify a column of data in the table as the source of vertex VID in the NebulaGraph. # For example, if rowkey is the source of the VID, enter rowkey.
        field:rowkev
     # Number of pieces of data written to NebulaGraph in a single batch.
    batch: 256
    # Number of Spark partitions
    partition: 32
  # Set Tag Team information.
    type: {
  source: hbase
      sink: client
    host:192.168.*.*
    port:2181
    table:"team'
    columnFamily:"cf"
    fields: [name]
    nebula.fields: [name]
    vertex:{
        field:rowkev
```

```
batch: 256
    partition: 32
]
# Processing edges
edges: [
   # Set the information about the Edge Type follow.
     # The corresponding Edge Type name in NebulaGraph.
     name: follow
       # Specify the data source file format to HBase.
       source: hbase
       # Specify how to import the Edge type data into NebulaGraph.
# Specify how to import the data into NebulaGraph: Client or SST.
       sink: client
    host:192.168.*.*
     port:2181
    table:"follow"
columnFamily:"cf'
     # Specify the column names in the follow table in fields, and their corresponding values are specified as properties in the NebulaGraph. # The sequence of fields and nebula.fields must correspond to each other.
     # If multiple column names need to be specified, separate them by commas.
     fields: [degree]
nebula.fields: [degree]
     # In source, use a column in the follow table as the source of the edge's source vertex.
# In target, use a column in the follow table as the source of the edge's destination vertex.
     source:{
          field:rowkev
     target:{
          field:dst_player
     # (Optional) Specify a column as the source of the rank.
     #ranking: rank
     # The number of data written to NebulaGraph in a single batch.
      # The number of Spark partitions.
   \ensuremath{\text{\#}} Set the information about the Edge Type serve.
     type: {
  source: hbase
       sink: client
     host:192.168.*.*
     port:2181
    table:"serve"
columnFamily:"cf"
     fields: [start_year,end_year]
     nebula.fields: [start_year,end_year]
     source:{
          field:rowkey
     target:{
        field:teamid
     # (Optional) Specify a column as the source of the rank.
     #ranking: rank
    batch: 256
    partition: 32
```

STEP 3: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import HBase data into NebulaGraph. For descriptions of the parameters, see Options for import.

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.4.0.jar_path> -c <hbase_application.conf_path>
```



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

## For example:

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.4.0.jar -c /root/nebula-exchange/nebula-exchange/target/classes/hbase\_application.conf

You can search for batchSuccess.<tag\_name/edge\_name> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 4: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

LOOKUP ON player YIELD id(vertex);

Users can also run the SHOW STATS command to view statistics.

STEP 5: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

# 18.4.6 Import data from MySQL/PostgreSQL

This topic provides an example of how to use Exchange to export MySQL data and import to NebulaGraph. It also applies to exporting data from PostgreSQL into NebulaGraph.

#### Data set

This topic takes the basketballplayer dataset as an example.

In this example, the data set has been stored in MySQL. All vertexes and edges are stored in the player, team, follow, and serve tables. The following are some of the data for each table.



### **Environment**

This example is done on MacOS. Here is the environment configuration information:

• Hardware specifications:

• CPU: 1.7 GHz Quad-Core Intel Core i7

• Memory: 16 GB

• Spark: 2.4.7, stand-alone

• Hadoop: 2.9.2, pseudo-distributed deployment

• MySQL: 8.0.23

• NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

#### **Prerequisites**

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- Exchange has been compiled, or download the compiled .jar file directly.
- · Spark has been installed.
- · Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- The Hadoop service has been installed and started.

#### **Steps**

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: MODIFY CONFIGURATION FILES

After Exchange is compiled, copy the conf file target/classes/application.conf to set MySQL data source configuration. In this case, the copied file is called <code>mysql\_application.conf</code>. For details on each configuration item, see Parameters in the configuration file.

```
{
  # Spark configuration
  spark: {
  app: {
    name: NebulaGraph Exchange 3.4.0
  }
  driver: {
    cores: 1
    maxResultSize: 16
```

```
cores: {
# NebulaGraph configuration
nebula: {
   # Specify the IP addresses and ports for Graph and Meta services.
    # If there are multiple addresses, the format is "ip1:port", "ip2:port", "ip3:port".
   # Addresses are separated by commas. graph:["127.0.0.1:9669"]
    # the address of any of the meta services.
    # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
    meta:["127.0.0.1:9559"]
  # The account entered must have write permission for the NebulaGraph space.
  user: root
  pswd: nebula
  # Fill in the name of the graph space you want to write data to in the NebulaGraph.
  space: basketballplayer
  connection: {
   timeout: 3000
    retry: 3
  execution: {
  retry: 3
  error: {
   max: 32
   output: /tmp/errors
  rate: {
    limit: 1024
    timeout: 1000
# Processing vertexes
tags: [
  # Set the information about the Tag player.
    # The Tag name in NebulaGraph.
    name: player
    type: {
    # Specify the data source file format to MySQL.
      source: mysql
# Specify how to import the data into NebulaGraph: Client or SST.
    host:192.168.*.*
    port:3306
database:"basketball"
   table:"player"
user:"test"
    sentence:"select playerid, age, name from player order by playerid;"
    # Specify the column names in the player table in fields, and their corresponding values are specified as properties in the NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    # If multiple column names need to be specified, separate them by commas.
   fields: [age,name]
nebula.fields: [age,name]
    # Specify a column of data in the table as the source of VIDs in the NebulaGraph.
    vertex: {
      field:playerid
    # The number of data written to NebulaGraph in a single batch.
   batch: 256
    # The number of Spark partitions.
    partition: 32
  # Set the information about the Tag Team.
    type: {
      source: mysql
      sink: client
    host:192.168.*.*
    port:3306
    database:"basketball"
    table:"team"
    user:"test"
    password:"123456" sentence:"select teamid, name from team order by teamid;"
    fields: [name]
    nebula.fields: [name]
```

```
vertex: {
       field: teamid
    hatch: 256
   partition: 32
# Processing edges
  # Set the information about the Edge Type follow.
    # The corresponding Edge Type name in NebulaGraph.
    name: follow
     type: {
       # Specify the data source file format to MySQL.
       source: mysql
       # Specify how to import the Edge type data into NebulaGraph.
# Specify how to import the data into NebulaGraph: Client or SST.
       sink: client
    host:192.168.*.*
    port:3306
    database:"basketball"
table:"follow"
user:"test"
     password:"123456"
     sentence: "select src_player,dst_player,degree from follow order by src_player;"
     # Specify the column names in the follow table in fields, and their corresponding values are specified as properties in the NebulaGraph. # The sequence of fields and nebula.fields must correspond to each other.
     # If multiple column names need to be specified, separate them by commas.
     fields: [degree]
     nebula.fields: [degree]
     # In source, use a column in the follow table as the source of the edge's source vertex.
# In target, use a column in the follow table as the source of the edge's destination vertex.
     source: {
       field: src_player
     target: {
       field: dst_player
    # (Optional) Specify a column as the source of the rank. #ranking: rank
     # The number of data written to NebulaGraph in a single batch.
    batch: 256
     # The number of Spark partitions.
    partition: 32
   # Set the information about the Edge Type serve.
     type: {
       source: mysql
      sink: client
    host:192.168.*.*
     port:3306
     database:"basketball"
    table:"serve"
user:"test"
     password:"123456"
    sentence: "select playerid,teamid,start_year,end_year from serve order by playerid;" fields: [start_year,end_year] nebula.fields: [start_year,end_year]
     source: {
       field: playerid
    target: {
      field: teamid
    batch: 256
    partition: 32
```

#### STEP 3: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import MySQL data into NebulaGraph. For a description of the parameters, see Options for import.

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.4.0.jar\_path> -c <mysql\_application.conf\_path>



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

## For example:

\${\$PARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.4.0.jar -c /root/nebula-exchange/nebula-exchange/target/classes/mysql\_application.conf

You can search for batchSuccess.<tag\_name/edge\_name> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 4: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

LOOKUP ON player YIELD id(vertex);

Users can also run the **SHOW STATS** command to view statistics.

STEP 5: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

# 18.4.7 Import data from Oracle

This topic provides an example of how to use Exchange to export Oracle data and import to NebulaGraph.

#### Data set

This topic takes the basketballplayer dataset as an example.

In this example, the data set has been stored in Oracle. All vertexes and edges are stored in the player, team, follow, and serve tables. The following are some of the data for each table.



# Environment

This example is done on MacOS. Here is the environment configuration information:

- Hardware specifications:
- CPU: 1.7 GHz Quad-Core Intel Core i7
- Memory: 16 GB
- Spark: 2.4.7, stand-alone
- Hadoop: 2.9.2, pseudo-distributed deployment
- NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

#### **Prerequisites**

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- Exchange has been compiled, or download the compiled .jar file directly.
- · Spark has been installed.
- · Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- The Hadoop service has been installed and started.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: MODIFY CONFIGURATION FILES

After Exchange is compiled, copy the conf file target/classes/application.conf to set Oracle data source configuration. In this case, the copied file is called oracle\_application.conf. For details on each configuration item, see Parameters in the configuration file.

```
{
  # Spark configuration
  spark: {
  app: {
    name: NebulaGraph Exchange 3.4.0
  }
  driver: {
    cores: 1
    maxResultSize: 16
```

```
cores: {
# NebulaGraph configuration
nebula: {
    # Specify the IP addresses and ports for Graph and Meta services.
    # If there are multiple addresses, the format is "ip1:port", "ip2:port", "ip3:port".
    # Addresses are separated by commas. graph:["127.0.0.1:9669"]
    # the address of any of the meta services.
    # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
    meta:["127.0.0.1:9559"]
  # The account entered must have write permission for the NebulaGraph space.
  user: root
  pswd: nebula
  # Fill in the name of the graph space you want to write data to in the NebulaGraph.
  space: basketballplayer
  connection: {
    timeout: 3000
    retry: 3
  execution: {
 retry: 3
  error: {
    max· 32
    output: /tmp/errors
  rate: {
    limit: 1024
    timeout: 1000
# Processing vertexes
tags: [
  # Set the information about the Tag player.
    # The Tag name in NebulaGraph.
    name: player
    type: {
     # Specify the data source file format to Oracle.
      source: oracle # Specify how to import the data into NebulaGraph: Client or SST.
    url:"jdbc:oracle:thin:@host:1521:db"
driver: "oracle.jdbc.driver.OracleDriver"
user: "root"
    password: "123456"
table: "basketball.player"
    sentence: "select playerid, name, age from player"
    # Specify the column names in the player table in fields, and their corresponding values are specified as properties in the NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
# If multiple column names need to be specified, separate them by commas.
    fields: [age,name]
    nebula.fields: [age,name]
     # Specify a column of data in the table as the source of VIDs in the NebulaGraph.
    vertex: {
      field:playerid
    # The number of data written to NebulaGraph in a single batch.
    batch: 256
    # The number of Spark partitions.
    partition: 32
  # Set the information about the Tag Team.
    name: team
      source: oracle
      sink: client
    url:"jdbc:oracle:thin:@host:1521:db"
driver: "oracle.jdbc.driver.OracleDriver"
user: "root"
    password: "123456"
    table: "basketball.team"
    sentence: "select teamid, name from team"
     fields: [name]
    nebula.fields: [name]
    vertex: {
      field: teamid
```

```
batch: 256
    partition: 32
# Processing edges
edges: [
# Set the information about the Edge Type follow.
    # The corresponding Edge Type name in NebulaGraph.
    name: follow
      # Specify the data source file format to Oracle.
      source: oracle
      # Specify how to import the Edge type data into NebulaGraph.
      # Specify how to import the data into NebulaGraph: Client or SST.
      sink: client
    url:"jdbc:oracle:thin:@host:1521:db"
    driver: "oracle.jdbc.driver.OracleDriver"
user: "root"
    password: "123456"
    table: "basketball.follow" sentence: "select src_player, dst_player, degree from follow"
    # Specify the column names in the follow table in fields, and their corresponding values are specified as properties in the NebulaGraph. # The sequence of fields and nebula.fields must correspond to each other.
    # If multiple column names need to be specified, separate them by commas.
    fields: [degree]
nebula.fields: [degree]
    # In source, use a column in the follow table as the source of the edge's source vertex.
     # In target, use a column in the follow table as the source of the edge's destination vertex.
    source: {
      field: src player
    target: {
      field: dst_player
    # (Optional) Specify a column as the source of the rank.
    #ranking: rank
    # The number of data written to NebulaGraph in a single batch.
    batch: 256
    # The number of Spark partitions.
    partition: 32
  # Set the information about the Edge Type serve.
    name: serve
      source: oracle
      sink: client
    url:"jdbc:oracle:thin:@host:1521:db"
    driver: "oracle.jdbc.driver.OracleDriver"
user: "root"
    password: "123456"
    table: "basketball.serve"
    sentence: "select playerid, teamid, start_year, end_year from serve"
    fields: [start_year,end_year]
nebula.fields: [start_year,end_year]
    source: {
      field: playerid
    target: {
      field: teamid
    batch: 256
    partition: 32
```

STEP 3: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import Oracle data into NebulaGraph. For a description of the parameters, see Options for import.

```
${SPARK_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.4.0.jar_path> -c <oracle_application.conf_path>
```



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

## For example:

\${\$PARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.4.0.jar -c /root/nebula-exchange/nebula-exchange/target/classes/oracle\_application.conf

You can search for batchSuccess.<tag\_name/edge\_name> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 4: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

LOOKUP ON player YIELD id(vertex);

Users can also run the SHOW STATS command to view statistics.

STEP 5: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

## 18.4.8 Import data from ClickHouse

This topic provides an example of how to use Exchange to import data stored on ClickHouse into NebulaGraph.

#### Data set

This topic takes the basketballplayer dataset as an example.

### Environment

This example is done on MacOS. Here is the environment configuration information:

- Hardware specifications:
- CPU: 1.7 GHz Quad-Core Intel Core i7
- Memory: 16 GB
- Spark: 2.4.7, stand-alone
- Hadoop: 2.9.2, pseudo-distributed deployment
- ClickHouse: docker deployment yandex/clickhouse-server tag: latest(2021.07.01)
- NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

## Prerequisites

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- Exchange has been compiled, or download the compiled .jar file directly.
- Spark has been installed.
- Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- The Hadoop service has been installed and started.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: MODIFY CONFIGURATION FILES

After Exchange is compiled, copy the conf file target/classes/application.conf to set ClickHouse data source configuration. In this example, the copied file is called clickhouse\_application.conf. For details on each configuration item, see Parameters in the configuration file.

```
# Spark configuration
   app: {
      name: NebulaGraph Exchange 3.4.0
   driver: {
     maxResultSize: 1G
   cores: {
     max: 16
# NebulaGraph configuration
 nebula: {
    address:{
      # Specify the IP addresses and ports for Graph and Meta services.
# If there are multiple addresses, the format is "ip1:port","ip2:port","ip3:port".
      # Addresses are separated by commas.
      graph:["127.0.0.1:9669"]
# the address of any of the meta services.
      # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
      meta:["127.0.0.1:9559"]
    # The account entered must have write permission for the NebulaGraph space.
   user: root
pswd: nebula
    .
# Fill in the name of the graph space you want to write data to in the NebulaGraph.
    space: basketballplayer
    connection: {
      timeout: 3000
```

```
retry: 3
  execution: {
    retry: 3
  error: {
    max: 32
    output: /tmp/errors
  rate: {
    limit: 1024
    timeout: 1000
}
# Processing vertexes
  # Set the information about the Tag player.
    name: player
    type: {
    # Specify the data source file format to ClickHouse.
      source: clickhouse
      # Specify how to import the data of vertexes into NebulaGraph: Client or SST.
      sink: client
    # JDBC URL of ClickHouse
    url:"jdbc:clickhouse://192.168.*.*:8123/basketballplayer"
    password: "123456"
    # The number of ClickHouse partitions numPartition:"5"
    sentence:"select * from player"
    # Specify the column names in the player table in fields, and their corresponding values are specified as properties in the NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
# If multiple column names need to be specified, separate them by commas.
    fields: [name,age]
nebula.fields: [name,age]
    # Specify a column of data in the table as the source of vertex VID in the NebulaGraph.
    vertex: {
   # policy:hash
      field:playerid
    # The number of data written to NebulaGraph in a single batch.
    batch: 256
    # The number of Spark partitions.
    partition: 32
  # Set the information about the Tag Team.
    name: team
      source: clickhouse
      sink: client
    url:"jdbc:clickhouse://192.168.*.*:8123/basketballplayer"
    user:"user"
password:"123456"
numPartition:"5"
    sentence:"select * from team"
    fields: [name]
    nebula.fields: [name]
    vertex: {
      field:teamid
    batch: 256
    partition: 32
# Processing edges
  # Set the information about the Edge Type follow.
  \{ \begin{tabular}{ll} \# \begin{tabular}{ll} \# \begin{tabular}{ll} The corresponding Edge Type name in NebulaGraph. \end{tabular}
    name: follow
    type: {
    # Specify the data source file format to ClickHouse.
      source: clickhouse
      \mbox{\# Specify how to import the data into NebulaGraph: Client or SST.}
      sink: client
    # JDBC URL of ClickHouse
```

```
url:"jdbc:clickhouse://192.168.*.*:8123/basketballplayer"
 password:"123456"
  # The number of ClickHouse partitions.
 numPartition:"5"
  sentence: "select * from follow"
  # Specify the column names in the follow table in fields, and their corresponding values are specified as properties in the NebulaGraph.
  # The sequence of fields and nebula.fields must correspond to each other.
# If multiple column names need to be specified, separate them by commas.
  fields: [degree]
  nebula.fields: [degree]
  # In source, use a column in the follow table as the source of the edge's source vertexes.
  source: {
    field:src_player
  # In target, use a column in the follow table as the source of the edge's destination vertexes.
  target:
   field:dst_player
  # (Optional) Specify a column as the source of the rank.
  # The number of data written to NebulaGraph in a single batch.
 batch: 256
  # The number of Spark partitions.
 partition: 32
# Set the information about the Edge Type serve.
  name: serve
 type: {
    source: clickhouse
   sink: client
  url:"jdbc:clickhouse://192.168.*.*:8123/basketballplayer"
 user: "user"
  password:"123456"
  numPartition:"5"
sentence:"select * from serve"
  fields: [start_year,end_year]
  nebula.fields: [start_year,end_year]
  source: {
   field:playerid
  target: {
    field:teamid
 # (Optional) Specify a column as the source of the rank.
  #ranking: rank
 batch: 256
 partition: 32
```

STEP 3: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import ClickHouse data into NebulaGraph. For descriptions of the parameters, see Options for import.

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange <nebula-exchange-3.4.0.jar\_path> -c <clickhouse\_application.conf\_path>



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

# For example:

\${SPARK\_HONE}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.4.0.jar -c /root/nebula-exchange/target/classes/clickhouse\_application.conf

You can search for batchSuccess.<a href="tag\_name/edge\_name">tag\_name/edge\_name">tag\_name/edge\_name</a> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 4: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

LOOKUP ON player YIELD id(vertex);

Users can also run the **SHOW STATS** command to view statistics.

STEP 5: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

## 18.4.9 Import data from Neo4j

This topic provides an example of how to use Exchange to import NebulaGraph data stored in Neo4j.

#### Implementation method

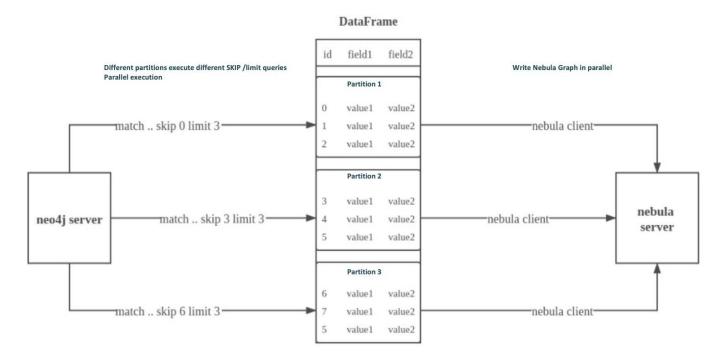
Exchange uses **Neo4j Driver 4.0.1** to read Neo4j data. Before batch export, you need to write Cypher statements that are automatically executed based on labels and relationship types and the number of Spark partitions in the configuration file to improve data export performance.

When Exchange reads Neo4j data, it needs to do the following:

- 1. The Reader in Exchange replaces the statement following the Cypher RETURN statement in the exec part of the configuration file with COUNT(\*), and executes this statement to get the total amount of data, then calculates the starting offset and size of each partition based on the number of Spark partitions.
- 2. (Optional) If the user has configured the <code>check\_point\_path</code> directory, Reader reads the files in the directory. In the transferring state, Reader calculates the offset and size that each Spark partition should have.
- 3. In each Spark partition, the Reader in Exchange adds different SKIP and LIMIT statements to the Cypher statement and calls the Neo4j Driver for parallel execution to distribute data to different Spark partitions.
- 4. The Reader finally processes the returned data into a DataFrame.

At this point, Exchange has finished exporting the Neo4j data. The data is then written in parallel to the NebulaGraph database.

The whole process is illustrated below.



### Data set

This topic takes the basketballplayer dataset as an example.

#### Environment

This example is done on MacOS. Here is the environment configuration information:

- Hardware specifications:
- CPU: Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz
- CPU cores: 14 Memory: 251 GB
- Spark: Stand-alone, 2.4.6 pre-build for Hadoop 2.7
- Neo4j: 3.5.20 Community Edition
- NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

#### **Prerequisites**

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- $\bullet$  The user name and password with NebulaGraph write permission.
- Exchange has been compiled, or download the compiled .jar file directly.
- Spark has been installed.
- Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: CONFIGURING SOURCE DATA

To speed up the export of Neo4j data, create indexes for the corresponding properties in the Neo4j database. For more information, refer to the Neo4j manual.

STEP 3: MODIFY CONFIGURATION FILES

After Exchange is compiled, copy the conf file target/classes/application.conf to set Neo4j data source configuration. In this example, the copied file is called neo4j\_application.conf. For details on each configuration item, see Parameters in the configuration file.

```
meta:["127.0.0.1:9559"]
  user: root
  pswd: nebula
space: basketballplayer
  connection: {
  timeout: 3000
  retry: 3
 retry: 3
  \hbox{execution: } \{
  error: {
    max: 32
    output: /tmp/errors
  rate: {
     limit: 1024
    timeout: 1000
# Processing vertexes
tags: [
  # Set the information about the Tag player
     name: player
    type: {
  source: neo4j
      sink: client
     server: "bolt://192.168.*.*:7687"
    user: neo4j
password:neo4j
     database:neo4j
     uacudae:news]
exec: "match (n:player) return n.id as id, n.age as age, n.name as name"
fields: [age,name]
     nebula.fields: [age,name]
     vertex: {
      field:id
     partition: 10
     batch: 1000
     check_point_path: /tmp/test
\ensuremath{\text{\#}} Set the information about the Tag Team
     name: team
     type: {
  source: neo4j
       sink: client
     server: "bolt://192.168.*.*:7687"
     password:neo4j
database:neo4j
     exec: "match (n:team) return n.id as id,n.name as name" fields: [name] nebula.fields: [name]
     vertex: {
   field:id
     partition: 10
     batch: 1000
     check_point_path: /tmp/test
# Processing edges
edges: [
# Set the information about the Edge Type follow
     name: follow
     type: {
      source: neo4j
       sink: client
     server: "bolt://192.168.*.*:7687"
     user: neo4j
password:neo4j
     database:neo4j
exec: "match (a:player)-[r:follow]->(b:player) return a.id as src, b.id as dst, r.degree as degree order by id(r)"
fields: [degree]
     nebula.fields: [degree]
     source: {
  field: src
     target: {
```

```
field: dst
  #ranking: rank
  partition: 10
batch: 1000
  check_point_path: /tmp/test
# Set the information about the Edge Type serve
  name: serve
  type: {
     source: neo4j
    sink: client
  server: "bolt://192.168.*.*:7687"
  user: neo4j
  password:neo4j
  database:neo4i
  exec: "match (a:player)-[r:serve]->(b:team) return a.id as src, b.id as dst, r.start_year as start_year, r.end_year as end_year order by id(r)"
  fields: [start_year,end_year]
  nebula.fields: [start_year,end_year]
    field: src
    field: dst
  #ranking: rank
  partition: 10
  batch: 1000
  check_point_path: /tmp/test
```

**Exec configuration** 

When configuring either the tags.exec or edges.exec parameters, you need to fill in the Cypher query. To prevent loss of data during import, it is strongly recommended to include ORDER BY clause in Cypher queries. Meanwhile, in order to improve data import efficiency, it is better to select indexed properties for ordering. If there is no index, users can also observe the default order and select the appropriate properties for ordering to improve efficiency. If the pattern of the default order cannot be found, users can order them by the ID of the vertex or relationship and set the partition to a small value to reduce the ordering pressure of Neo4j.



Using the ORDER BY clause lengthens the data import time.

Exchange needs to execute different SKIP and LIMIT Cypher statements on different Spark partitions, so SKIP and LIMIT clauses cannot be included in the Cypher statements corresponding to tags.exec and edges.exec.

tags.vertex or edges.vertex configuration

NebulaGraph uses ID as the unique primary key when creating vertexes and edges, overwriting the data in that primary key if it already exists. So, if a Neo4j property value is given as the NebulaGraph'S ID and the value is duplicated in Neo4j, duplicate IDs will be generated. One and only one of their corresponding data will be stored in the NebulaGraph, and the others will be overwritten. Because the data import process is concurrently writing data to NebulaGraph, the final saved data is not guaranteed to be the latest data in Neo4j.

check\_point\_path configuration

If breakpoint transfers are enabled, to avoid data loss, the state of the database should not change between the breakpoint and the transfer. For example, data cannot be added or deleted, and the partition quantity configuration should not be changed.

STEP 4: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import Neo4j data into NebulaGraph. For a description of the parameters, see Options for import.

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange<-3.4.0.jar\_path> -c <neo4j\_application.conf\_path>



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

## For example:

\${\$PARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.4.0.jar -c /root/nebula-exchange/nebula-exchange/target/classes/neo4j\_application.conf

You can search for batchSuccess.<tag\_name/edge\_name> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 5: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

LOOKUP ON player YIELD id(vertex);

Users can also run the SHOW STATS command to view statistics.

STEP 6: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

## 18.4.10 Import data from Hive

This topic provides an example of how to use Exchange to import NebulaGraph data stored in Hive.

#### Data set

This topic takes the basketballplayer dataset as an example.

In this example, the data set has been stored in Hive. All vertexes and edges are stored in the player, team, follow, and serve tables. The following are some of the data for each table.

```
scala> spark.sql("describe basketball.player").show
|col_name|data_type|comment|
|playerid| string|
     age
                     null
    name
           string
scala> spark.sql("describe basketball.team").show
 col_name|data_type|comment|
    teamid| string| null| name| string| null|
scala> spark.sql("describe basketball.follow").show
| col_name|data_type|comment|
|src_player| string| null|
              string
|dst_player|
    degree | bigint | null
scala> spark.sql("describe basketball.serve").show
 col_name|data_type|comment|
 playerid
             string| null|
              string
start_year|
             bigint
                      null
             bigint
                       null
 end_year
```



The Hive data type bigint corresponds to the NebulaGraph int.

# Environment

This example is done on MacOS. Here is the environment configuration information:

- Hardware specifications:
- CPU: 1.7 GHz Quad-Core Intel Core i7
- Memory: 16 GB
- Spark: 2.4.7, stand-alone
- Hadoop: 2.9.2, pseudo-distributed deployment
- Hive: 2.3.7, Hive Metastore database is MySQL 8.0.22
- NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

#### **Prerequisites**

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- Exchange has been compiled, or download the compiled .jar file directly.
- · Spark has been installed.
- · Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- Hadoop has been installed and started, and the Hive Metastore database (MySQL in this example) has been started.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: USE SPARK SQL TO CONFIRM HIVE SQL STATEMENTS

After the Spark-shell environment is started, run the following statements to ensure that Spark can read data in Hive.

```
scala> sql("select playerid, age, name from basketball.player").show
scala> sql("select teamid, name from basketball.team").show
scala> sql("select src_player, dst_player, degree from basketball.follow").show
scala> sql("select playerid, teamid, start_year, end_year from basketball.serve").show
```

The following is the result read from the table  $\mbox{\it basketball.player}$  .

```
player100| 42| Tim Duncan|
player101| 36| Tony Parker|
player102| 33 |LaMarcus Aldridge|
player103| 32| Rudy Gay|
player104| 32| Marco Belinelli|
```

#### STEP 3: MODIFY CONFIGURATION FILE

After Exchange is compiled, copy the conf file target/classes/application.conf to set Hive data source configuration. In this example, the copied file is called hive\_application.conf. For details on each configuration item, see Parameters in the configuration file.

```
# Spark configuration
spark: {
  app: {
     name: NebulaGraph Exchange 3.4.0
  driver: {
    maxResultSize: 1G
  cores: {
    max: 16
# If Spark and Hive are deployed in different clusters, you need to configure the parameters for connecting to Hive. Otherwise, skip these configurations.
#hive: {
# waredir: "hdfs://NAMENODE_IP:9000/apps/svr/hive-xxx/warehouse/"
# connectionURL: "jdbc:mysql://your_ip:3306/hive_spark?characterEncoding=UTF-8" 
# connectionDriverName: "com.mysql.jdbc.Driver"
# connectionUserName: "user"
# connectionPassword: "password"
# NebulaGraph configuration
  address:{
     # Specify the IP addresses and ports for Graph and all Meta services.
     # If there are multiple addresses, the format is "ip1:port", "ip2:port", "ip3:port".
    \mbox{\#} Addresses are separated by commas.
    graph:["127.0.0.1:9669"]
     # the address of any of the meta services
     # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
     meta:["127.0.0.1:9559"]
  # The account entered must have write permission for the NebulaGraph space.
  pswd: nebula
   .
# Fill in the name of the graph space you want to write data to in the NebulaGraph.
   space: basketballplayer
  connection: {
    timeout: 3000
     retry: 3
  execution: {
    retry: 3
  error: {
    max: 32
    output: /tmp/errors
  rate: {
     limit: 1024
     timeout: 1000
# Processing vertexes
tags: [
   # Set the information about the Tag player.
    # The Tag name in NebulaGraph.
     name: player
     type: {
       # Specify the data source file format to Hive.
      source: hive
       # Specify how to import the data into NebulaGraph: Client or SST.
     # Set the SQL statement to read the data of player table in basketball database.
     exec: "select playerid, age, name from basketball.player'
    # Specify the column names in the player table in fields, and their corresponding values are specified as properties in the NebulaGraph. # The sequence of fields and nebula.fields must correspond to each other.
     # If multiple column names need to be specified, separate them by commas.
    fields: [age,name]
nebula.fields: [age,name]
```

```
# Specify a column of data in the table as the source of vertex VID in the NebulaGraph.
     field:playerid
    # The number of data written to NebulaGraph in a single batch.
   batch: 256
    # The number of Spark partitions.
  # Set the information about the Tag Team.
    name: team
    type: {
      source: hive
      sink: client
    exec: "select teamid, name from basketball.team"
    fields: [name]
    nebula.fields: [name]
    vertex: {
      field: teamid
    batch: 256
   partition: 32
]
# Processing edges
  # Set the information about the Edge Type follow.
    # The corresponding Edge Type name in NebulaGraph.
   name: follow
    type: {
    # Specify the data source file format to Hive.
      # Specify how to import the Edge type data into NebulaGraph.
      # Specify how to import the data into NebulaGraph: Client or SST.
     sink: client
    # Set the SQL statement to read the data of follow table in the basketball database.
    exec: "select src_player, dst_player, degree from basketball.follow"
    # Specify the column names in the follow table in Fields, and their corresponding values are specified as properties in the NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
# If multiple column names need to be specified, separate them by commas.
    fields: [degree]
    nebula.fields: [degree]
    # In source, use a column in the follow table as the source of the edge's starting vertex.
    # In target, use a column in the follow table as the source of the edge's destination vertex.
   field: src_player
}
    target: {
      field: dst_player
    # (Optional) Specify a column as the source of the rank.
    # The number of data written to NebulaGraph in a single batch.
    batch: 256
    # The number of Spark partitions.
    partition: 32
  # Set the information about the Edge Type serve.
    type: {
      sink: client
    exec: "select playerid, teamid, start_year, end_year from basketball.serve"
    fields: [start_year,end_year]
nebula.fields: [start_year,end_year]
      field: playerid
    target: {
     field: teamid
    # (Optional) Specify a column as the source of the rank.
```

```
#ranking: rank

batch: 256
partition: 32
}
```

STEP 4: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import Hive data into NebulaGraph. For a description of the parameters, see Options for import.

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <-nebula-exchange<-3.4.0.jar\_path> -c <nive\_application.conf\_path> -h



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

### For example:

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.4.0.jar -c /root/nebula-exchange/nebula-exchange/target/classes/hive\_application.conf -h

You can search for batchSuccess.<tag\_name/edge\_name> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 5: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

LOOKUP ON player YIELD id(vertex);

Users can also run the SHOW STATS command to view statistics.

STEP 6: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

## 18.4.11 Import data from MaxCompute

This topic provides an example of how to use Exchange to import NebulaGraph data stored in MaxCompute.

#### Data set

This topic takes the basketballplayer dataset as an example.

### Environment

This example is done on MacOS. Here is the environment configuration information:

- Hardware specifications:
- CPU: 1.7 GHz Quad-Core Intel Core i7
- Memory: 16 GB
- Spark: 2.4.7, stand-alone
- Hadoop: 2.9.2, pseudo-distributed deployment
- MaxCompute: Alibaba Cloud official version
- NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

## Prerequisites

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- Exchange has been compiled, or download the compiled .jar file directly.
- Spark has been installed.
- Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- The Hadoop service has been installed and started.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: MODIFY CONFIGURATION FILES

After Exchange is compiled, copy the conf file target/classes/application.conf to set MaxCompute data source configuration. In this example, the copied file is called <code>maxcompute\_application.conf</code>. For details on each configuration item, see <a href="Parameters in the configuration file">Parameters in the configuration file</a>.

```
# Spark configuration
  app: {
    name: NebulaGraph Exchange 3.4.0
  driver: {
    maxResultSize: 1G
  cores: {
    max: 16
# NebulaGraph configuration
nebula: {
  address:{
    # Specify the IP addresses and ports for Graph and Meta services.
# If there are multiple addresses, the format is "ip1:port","ip2:port","ip3:port".
    # Addresses are separated by commas.
    graph:["127.0.0.1:9669"]
# the address of any of the meta services.
    # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
    meta:["127.0.0.1:9559"]
  # The account entered must have write permission for the NebulaGraph space.
  user: root
pswd: nebula
  .
# Fill in the name of the graph space you want to write data to in the NebulaGraph.
  space: basketballplayer
  connection: {
    timeout: 3000
```

```
retry: 3
  execution: {
   retry: 3
  error: {
   max: 32
   output: /tmp/errors
  rate: {
   limit: 1024
    timeout: 1000
# Processing vertexes
  # Set the information about the Tag player.
    name: player
    type: {
    # Specify the data source file format to MaxCompute.
      # Specify how to import the data into NebulaGraph: Client or SST.
      sink: client
    # Table name of MaxCompute.
    table:player
    # Project name of MaxCompute.
    project:project
    \# OdpsUrl and tunnelUrl for the MaxCompute service.
    # The address is https://help.aliyun.com/document_detail/34951.html.
odpsUrl:"http://service.cn-hangzhou.maxcompute.aliyun.com/api"
    tunne \verb|Url:"http://dt.cn-hangzhou.maxcompute.aliyun.com"|
    # AccessKeyId and accessKeySecret of the MaxCompute service.
    accessKeyId:xxx
    accessKeySecret:xxx
    # Partition description of the MaxCompute table. This configuration is optional.
    partitionSpec:"dt='partition1'"
    # Ensure that the table name in the SQL statement is the same as the value of the table above. This configuration is optional.
    sentence: "select id, name, age, playerid from player where id < 10"
    # Specify the column names in the player table in fields, and their corresponding values are specified as properties in the NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    # If multiple column names need to be specified, separate them by commas.
    fields:[name, age]
    nebula.fields:[name, age]
    # Specify a column of data in the table as the source of vertex VID in the NebulaGraph.
    vertex:{
      field: playerid
    # The number of data written to NebulaGraph in a single batch.
    batch: 256
    # The number of Spark partitions.
    partition: 32
  # Set the information about the Tag Team.
    type: {
      source: maxcompute
      sink: client
    table:team
    project:project
odpsUrl:"http://service.cn-hangzhou.maxcompute.aliyun.com/api"
    tunnelUrl:"http://dt.cn-hangzhou.maxcompute.aliyun.com
    accessKeyId:xxx
    accessKeySecret:xxx
    partitionSpec:"dt='partition1'"
sentence:"select id, name, teamid from team where id < 10"
    nebula.fields:[name]
    vertex:{
      field: teamid
    batch: 256
    partition: 32
# Processing edges
  # Set the information about the Edge Type follow.
```

```
# The corresponding Edge Type name in NebulaGraph.
 name: follow
    # Specify the data source file format to MaxCompute.
   # Specify how to import the Edge type data into NebulaGraph.
# Specify how to import the data into NebulaGraph: Client or SST.
   sink:client
  # Table name of MaxCompute.
  table:follow
  # Project name of MaxCompute.
  project:project
  # OdpsUrl and tunnelUrl for MaxCompute service.
  # The address is https://help.aliyun.com/document_detail/34951.html.odpsUrl:"http://service.cn-hangzhou.maxcompute.aliyun.com/api"
  tunnelUrl:"http://dt.cn-hangzhou.maxcompute.aliyun.com
  # AccessKeyId and accessKeySecret of the MaxCompute service.
 accessKeyId:xxx
accessKeySecret:xxx
 # Partition description of the MaxCompute table. This configuration is optional. partitionSpec:"dt='partition1'"
 # Ensure that the table name in the SQL statement is the same as the value of the table above. This configuration is optional. sentence: "select * from follow"
  # Specify the column names in the follow table in Fields, and their corresponding values are specified as properties in the NebulaGraph. # The sequence of fields and nebula.fields must correspond to each other.
  # If multiple column names need to be specified, separate them by commas.
  fields:[degree]
  nebula.fields:[degree]
  # In source, use a column in the follow table as the source of the edge's source vertex.
   field: src_player
  # In target, use a column in the follow table as the source of the edge's destination vertex.
  target:{
   field: dst_player
  # (Optional) Specify a column as the source of the rank.
  #ranking: rank
 # The number of Spark partitions.
 partition:10
  # The number of data written to NebulaGraph in a single batch.
# Set the information about the Edge Type serve.
    source:maxcompute
  table:serve
  odpsUrl: "http://service.cn-hangzhou.maxcompute.alivun.com/api"
  tunnelUrl:"http://dt.cn-hangzhou.maxcompute.aliyun.com
  accessKeyId:xxx
  accessKeySecret:xxx
 partitionSpec:"dt='partition1'"
sentence:"select * from serve"
  fields:[start_year,end_year]
  nebula.fields:[start_year,end_year]
  source:{
   field: playerid
  target:{
  # (Optional) Specify a column as the source of the rank.
  #ranking: rank
  partition:10
 batch:10
```

#### STEP 3: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import MaxCompute data into NebulaGraph. For a description of the parameters, see Options for import.

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.4.0.jar\_path> -c <maxcompute\_application.conf\_path>



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

## For example:

\${\$PARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.4.0.jar -c /root/nebula-exchange/nebula-exchange/target/classes/maxcompute\_application.conf

You can search for batchSuccess.<tag\_name/edge\_name> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 4: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

LOOKUP ON player YIELD id(vertex);

Users can also run the SHOW STATS command to view statistics.

STEP 5: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

## 18.4.12 Import data from Pulsar

This topic provides an example of how to use Exchange to import NebulaGraph data stored in Pulsar.

#### **Environment**

This example is done on MacOS. Here is the environment configuration information:

- Hardware specifications:
- CPU: 1.7 GHz Quad-Core Intel Core i7
- Memory: 16 GB
- Spark: 2.4.7, stand-alone
- NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

## Prerequisites

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- Exchange has been compiled, or download the compiled .jar file directly.
- Spark has been installed.
- Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- The Pulsar service has been installed and started.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: MODIFY CONFIGURATION FILES

After Exchange is compiled, copy the conf file target/classes/application.conf to set Pulsar data source configuration. In this example, the copied file is called pulsar\_application.conf. For details on each configuration item, see Parameters in the configuration file.

```
# Spark configuration
spark: {
  app: {
    name: NebulaGraph Exchange 3.4.0
  driver: {
   cores: 1
   maxResultSize: 1G
  cores: {
    max: 16
# NebulaGraph configuration
  address:{
    # Specify the IP addresses and ports for Graph and all Meta services.
    # If there are multiple addresses, the format is "ip1:port","ip2:port","ip3:port".
    # Addresses are separated by commas.
    graph:["127.0.0.1:9669"]
    # the address of any of the meta services.
    # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
    meta:["127.0.0.1:9559"]
  # The account entered must have write permission for the NebulaGraph space
  pswd: nebula
  # Fill in the name of the graph space you want to write data to in the NebulaGraph.
  space: basketballplayer
  {\tt connection:}\ \{
```

```
timeout: 3000
    retry: 3
  execution: {
    retry: 3
  error: {
    max: 32
    output: /tmp/errors
    limit: 1024
    timeout: 1000
# Processing vertices
  # Set the information about the Tag player.
    \mbox{\tt\#} The corresponding Tag name in NebulaGraph.
     name: player
      # Specify the data source file format to Pulsar.
      source: pulsar
      # Specify how to import the data into NebulaGraph: Client or SST.
      sink: client
     # The address of the Pulsar server.
service: "pulsar://127.0.0.1:6650"
# admin.url of pulsar.
     admin: "http://127.0.0.1:8081"
     # The Pulsar option can be configured from topic, topics or topicsPattern.
     options: {
  topics: "topic1,topic2"
}
     # Specify the column names in the player table in fields, and their corresponding values are specified as properties in the NebulaGraph.
     # The sequence of fields and nebula.fields must correspond to each other.
     \# If multiple column names need to be specified, separate them by commas.
     fields: [age,name]
     nebula.fields: [age,name]
     # Specify a column of data in the table as the source of VIDs in the NebulaGraph.
        field:playerid
     # The number of data written to NebulaGraph in a single batch.
    batch: 10
     # The number of Spark partitions.
     partition: 10
     # The interval for message reading. Unit: second.
     interval.seconds: 10
   # Set the information about the Tag Team.
     name: team
      source: pulsar
      sink: client
     service: "pulsar://127.0.0.1:6650"
     admin: "http://127.0.0.1:8081"
     options: {
  topics: "topic1,topic2"
     fields: [name]
     nebula.fields: [name]
     vertex:{
         field:teamid
     hatch: 10
    partition: 10
     interval.seconds: 10
]
# Processing edges
edges: [
# Set the information about Edge Type follow
    # The corresponding Edge Type name in NebulaGraph.
    name: follow
      # Specify the data source file format to Pulsar.
      source: pulsar
      # Specify how to import the Edge type data into NebulaGraph.
# Specify how to import the data into NebulaGraph: Client or SST.
sink: client
```

```
# The address of the Pulsar server.
 service: "pulsar://127.0.0.1:6650" # admin.url of pulsar.
 admin: "http://127.0.0.1:8081"
 # The Pulsar option can be configured from topic, topics or topicsPattern.
 options: {
   topics: "topic1,topic2"
 # Specify the column names in the follow table in fields, and their corresponding values are specified as properties in the NebulaGraph. # The sequence of fields and nebula.fields must correspond to each other.
 # If multiple column names need to be specified, separate them by commas.
 fields: [degree]
nebula.fields: [degree]
 # In source, use a column in the follow table as the source of the edge's source vertex.
  # In target, use a column in the follow table as the source of the edge's destination vertex.
 source:{
     field:src player
 target:{
     field:dst_player
 # (Optional) Specify a column as the source of the rank.
 #ranking: rank
 # The number of data written to NebulaGraph in a single batch.
 # The number of Spark partitions.
 # The interval for message reading. Unit: second.
  interval.seconds: 10
# Set the information about the Edge Type serve
 name: serve
    source: Pulsar
   sink: client
 service: "pulsar://127.0.0.1:6650"
 admin: "http://127.0.0.1:8081"
 options: {
  topics: "topic1,topic2"
 fields: [start_year,end_year]
 nebula.fields: [start_year,end_year]
 source:{
     field:playerid
 target:{
     field:teamid
 # (Optional) Specify a column as the source of the rank.
 #ranking: rank
 batch: 10
 partition: 10
 interval.seconds: 10
```

STEP 3: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import Pulsar data into NebulaGraph. For a description of the parameters, see Options for import.

\${\$PARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.4.0.jar\_path> -c -



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

For example:

\${\$PARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.4.0.jar -c /root/nebula-exchange/nebula-exchange/target/classes/pulsar\_application.conf

You can search for batchSuccess.<tag\_name/edge\_name> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 4: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

LOOKUP ON player YIELD id(vertex);

Users can also run the SHOW STATS command to view statistics.

STEP 5: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

# 18.4.13 Import data from Kafka

This topic provides a simple guide to importing Data stored on Kafka into NebulaGraph using Exchange.

# mpatibility

Please use Exchange 3.3.0/3.0.0 when importing Kafka data. In version 3.4.0, caching of imported data was added, and streaming data import is not supported.

#### **Environment**

This example is done on MacOS. Here is the environment configuration information:

- Hardware specifications:
- CPU: 1.7 GHz Quad-Core Intel Core i7
- Memory: 16 GB
- Spark: 2.4.7, stand-alone
- NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

# **Prerequisites**

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- Exchange has been compiled, or download the compiled .jar file directly.
- Spark has been installed.
- Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- The Kafka service has been installed and started.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: MODIFY CONFIGURATION FILES



If some data is stored in Kafka's value field, you need to modify the source code, get the value from Kafka, parse the value through the from JSON function, and return it as a Dataframe.

After Exchange is compiled, copy the conf file target/classes/application.conf to set Kafka data source configuration. In this example, the copied file is called kafka\_application.conf. For details on each configuration item, see Parameters in the configuration file.

```
meta:["127.0.0.1:9559"]
  # The account entered must have write permission for the NebulaGraph space.
  user: root
pswd: nebula
  # Fill in the name of the graph space you want to write data to in the NebulaGraph. space: basketballplayer
  connection: {
    timeout: 3000
    retry: 3
  execution: {
    retry: 3
  error: {
 max: 32
    output: /tmp/errors
    limit: 1024
    timeout: 1000
# Processing vertexes
  # Set the information about the Tag player.
    # The corresponding Tag name in NebulaGraph.
     name: player
     type: {
       # Specify the data source file format to Kafka.
       source: kafka
      # Specify how to import the data into NebulaGraph: Client or SST. sink: client
    # Kafka server address.
     service: "127.0.0.1:9092"
    # Message category.
topic: "topic_name1"
    # Kafka data has a fixed domain name: key, value, topic, partition, offset, timestamp, timestampType.
# If multiple fields need to be specified after Spark reads as DataFrame, separate them with commas.
     # Specify the field name in fields. For example, use key for name in NebulaGraph and value for age in Nebula, as shown in the following.
     fields: [key,value]
     nebula.fields: [name,age]
     # Specify a column of data in the table as the source of vertex VID in the NebulaGraph.
     # The key is the same as the value above, indicating that key is used as both VID and property name.
     vertex:{
        field:key
    # The number of data written to NebulaGraph in a single batch.
    batch: 10
     # The number of Spark partitions.
     partition: 10
     # The interval for message reading. Unit: second.
     interval.seconds: 10
   # Set the information about the Tag Team.
     type: {
   source: kafka
      sink: client
     service: "127.0.0.1:9092"
     topic: "topic_name2"
fields: [key]
     nebula.fields: [name]
     vertex:{
        field:key
    batch: 10
    partition: 10
     interval.seconds: 10
]
# Processing edges
edges: [
# Set the information about the Edge Type follow.
    # The corresponding Edge Type name in NebulaGraph.
    name: follow
      # Specify the data source file format to Kafka.
      source: kafka
```

```
# Specify how to import the Edge type data into NebulaGraph.
# Specify how to import the data into NebulaGraph: Client or SST.
 # Kafka server address
 service: "127.0.0.1:9092"
 # Message category.
 topic: "topic_name3"
 # Kafka data has a fixed domain name: key, value, topic, partition, offset, timestamp, timestampType.
 # If multiple fields need to be specified after Spark reads as DataFrame, separate them with commas.
# Specify the field name in fields. For example, use key for degree in Nebula, as shown in the following.
 nebula.fields: [degree]
 # In source, use a column in the topic as the source of the edge's source vertex.
 # In target, use a column in the topic as the source of the edge's destination vertex.
      field:timestamp
 target:{
      field:offset
 # (Optional) Specify a column as the source of the rank.
 #ranking: rank
 # The number of data written to NebulaGraph in a single batch.
 # The number of Spark partitions.
 # The interval for message reading. Unit: second.
  interval.seconds: 10
# Set the information about the Edge Type serve.
 name: serve
    source: kafka
   sink: client
 service: "127.0.0.1:9092"
 topic: "topic_name4"
  fields: [timestamp,offset]
 nebula.fields: [start_year,end_year]
 source:{
      field:key
 target:{
      field:value
 # (Optional) Specify a column as the source of the rank.
 #ranking: rank
 batch: 10
 partition: 10
 interval.seconds: 10
```

STEP 3: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import Kafka data into NebulaGraph. For a description of the parameters, see Options for import.

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.4.0.jar\_path> -c <kafka\_application.conf\_path>



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

#### For example:

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange/Exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.4.0.jar -c /root/nebula-exchange/nebula-exchange/target/classes/kafka\_application.conf

You can search for batchSuccess.<a href="tag\_name/edge\_name">tag\_name/edge\_name">tag\_name/edge\_name</a> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 4: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

LOOKUP ON player YIELD id(vertex);

Users can also run the **SHOW STATS** command to view statistics.

STEP 5: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

# 18.4.14 Import data from general JDBC

JDBC data refers to the data of various databases accessed through the JDBC interface. This topic provides an example of how to use Exchange to export MySQL data and import to NebulaGraph.

#### Data set

This topic takes the basketballplayer dataset as an example.

In this example, the data set has been stored in MySQL. All vertexes and edges are stored in the player, team, follow, and serve tables. The following are some of the data for each table.



#### **Environment**

This example is done on MacOS. Here is the environment configuration information:

• Hardware specifications:

• CPU: 1.7 GHz Quad-Core Intel Core i7

• Memory: 16 GB

• Spark: 2.4.7, stand-alone

• Hadoop: 2.9.2, pseudo-distributed deployment

• MySQL: 8.0.23

• NebulaGraph: 3.4.1. Deploy NebulaGraph with Docker Compose.

#### **Prerequisites**

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- Exchange has been compiled, or download the compiled .jar file directly.
- · Spark has been installed.
- · Learn about the Schema created in NebulaGraph, including names and properties of Tags and Edge types, and more.
- The Hadoop service has been installed and started.

#### **Steps**

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: MODIFY CONFIGURATION FILES

After Exchange is compiled, copy the conf file target/classes/application.conf to set JDBC data source configuration. In this case, the copied file is called jdbc\_application.conf. For details on each configuration item, see Parameters in the configuration file.

```
{
  # Spark configuration
  spark: {
    app: {
        name: NebulaGraph Exchange 3.4.0
    }
    driver: {
        cores: 1
        maxResultSize: 16
```

```
cores: {
# NebulaGraph configuration
nebula: {
    # Specify the IP addresses and ports for Graph and Meta services.
    # If there are multiple addresses, the format is "ip1:port", "ip2:port", "ip3:port".
    # Addresses are separated by commas. graph:["127.0.0.1:9669"]
    # the address of any of the meta services.
    # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
    meta:["127.0.0.1:9559"]
  # The account entered must have write permission for the NebulaGraph space.
  user: root
  pswd: nebula
  # Fill in the name of the graph space you want to write data to in the NebulaGraph.
  space: basketballplayer
  connection: {
    timeout: 3000
    retry: 3
  execution: {
    retry: 3
  error: {
    max· 32
    output: /tmp/errors
  rate: {
    limit: 1024
    timeout: 1000
# Processing vertexes
tags: [
  # Set the information about the Tag player.
    # The Tag name in NebulaGraph.
    name: player
    type: {
     # Specify the data source file format to JDBC.
      source: jdbc
# Specify how to import the data into NebulaGraph: Client or SST.
    \# URL of the JDBC data source. The example is MySql database. 
  url:"jdbc:mysql://127.0.0.1:3306/basketball?useUnicode=true\&characterEncoding=utf-8" 
    # JDBC driver
    driver: "com.mvsql.ci.idbc.Driver"
    # Database user name and password
    user:root
    password:"12345"
    table:player
    sentence:"select playerid, age, name from player order by playerid"
     # (optional)Multiple connections read parameters. See https://spark.apache.org/docs/latest/sql-data-sources-jdbc.html
    partitionColumn:playerid  # optional. Must be a numeric, date, or timestamp column from the table in question.
lowerBound:1  # optional
     upperBound:5
                                   # optional
    numPartitions:5
                                  # optional
    fetchSize:2
                           # The JDBC fetch size, which determines how many rows to fetch per round trip.
    # Specify the column names in the player table in fields, and their corresponding values are specified as properties in the NebulaGraph. # The sequence of fields and nebula.fields must correspond to each other.
     # If multiple column names need to be specified, separate them by commas.
    fields: [age,name]
nebula.fields: [age,name]
    # Specify a column of data in the table as the source of VIDs in the NebulaGraph.
      field:playerid
    }
    # The number of data written to NebulaGraph in a single batch.
    batch: 256
    # The number of Spark partitions.
    partition: 32
  # Set the information about the Tag Team
    name: team
    type: {
```

```
source: jdbc
      sink: client
    url:"jdbc:mysql://127.0.0.1:3306/basketball?useUnicode=true&characterEncoding=utf-8"
    driver:"com.mysql.cj.jdbc.Driver"
    user:root
    password:"12345"
    table:team sentence:"select teamid, name from team order by teamid"
    partitionColumn:teamid
    lowerBound:1
upperBound:5
    numPartitions:5
    fetchSize:2
    fields: [name]
nebula.fields: [name]
      field: teamid
    batch: 256
    partition: 32
]
# Processing edges
edges: [
  # Set the information about the Edge Type follow.
    # The corresponding Edge Type name in NebulaGraph.
    name: follow
    type: {
      # Specify the data source file format to JDBC.
      source: jdbc
      # Specify how to import the Edge type data into NebulaGraph.
# Specify how to import the data into NebulaGraph: Client or SST.
    url:"jdbc:mysql://127.0.0.1:3306/basketball?useUnicode=true\&characterEncoding=utf-8"
    driver:"com.mysql.cj.jdbc.Driver"
    password:"12345"
    table:follow
    sentence: "select \ src\_player, dst\_player, degree \ from \ follow \ order \ by \ src\_player"
    partitionColumn:src_player
     lowerBound:1
    upperBound:5
    numPartitions:5
    fetchSize:2
    # Specify the column names in the follow table in fields, and their corresponding values are specified as properties in the NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    # If multiple column names need to be specified, separate them by commas.
    fields: [degree]
    nebula.fields: [degree]
    # In source, use a column in the follow table as the source of the edge's source vertex.
    # In target, use a column in the follow table as the source of the edge's destination vertex.
    source: {
    target: {
      field: dst_player
    # (Optional) Specify a column as the source of the rank.
    #ranking: rank
    # The number of data written to NebulaGraph in a single batch.
    # The number of Spark partitions.
    partition: 32
  # Set the information about the Edge Type serve.
    type: {
      source: jdbc
    url:"jdbc:mysql://127.0.0.1:3306/basketball?useUnicode=true\&characterEncoding=utf-8" \\ driver:"com.mysql.cj.jdbc.Driver"
    password:"12345"
    table:serve
```

```
sentence:"select playerid,teamid,start_year,end_year from serve order by playerid"
partitionColumn:playerid
lowerBound:1
upperBound:5
numPartitions:5
fetchSize:2

fields: [start_year,end_year]
nebula.fields: [start_year,end_year]
source: {
    field: playerid
}
target: {
    field: teamid
}
batch: 256
partition: 32
}
```

STEP 3: IMPORT DATA INTO NEBULAGRAPH

Run the following command to import general JDBC data into NebulaGraph. For a description of the parameters, see Options for import.

\${\$PARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.4.0.jar\_path> -c <jdbc\_application.conf\_path>



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

#### For example:

\${\$PARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange-3.4.0.jar -c /root/nebula-exchange/target/classes/jdbc\_application.conf

You can search for batchSuccess.<tag\_name/edge\_name> in the command output to check the number of successes. For example, batchSuccess.follow: 300.

STEP 4: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

```
LOOKUP ON player YIELD id(vertex);
```

Users can also run the SHOW STATS command to view statistics.

STEP 5: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

# 18.4.15 Import data from SST files

This topic provides an example of how to generate the data from the data source into an SST (Sorted String Table) file and save it on HDFS, and then import it into NebulaGraph. The sample data source is a CSV file.

#### **Precautions**

- The SST file can be imported only in Linux.
- The default value of the property is not supported.

#### **Background information**

Exchange supports two data import modes:

- Import the data from the data source directly into NebulaGraph as **nGQL** statements.
- Generate the SST file from the data source, and use Console to import the SST file into NebulaGraph.

The following describes the scenarios, implementation methods, prerequisites, and steps for generating an SST file and importing data.

#### Scenarios

• Suitable for online services, because the generation almost does not affect services (just reads the Schema), and the import speed is fast.



Although the import speed is fast, write operations in the corresponding space are blocked during the import period (about 10 seconds). Therefore, you are advised to import data in off-peak hours.

• Suitable for scenarios with a large amount of data from data sources for its fast import speed.

### Implementation methods

The underlying code in NebulaGraph uses RocksDB as the key-value storage engine. RocksDB is a storage engine based on the hard disk, providing a series of APIs for creating and importing SST files to help quickly import massive data.

- 911/1069 - 2023 Vesoft Inc.

The SST file is an internal file containing an arbitrarily long set of ordered key-value pairs for efficient storage of large amounts of key-value data. The entire process of generating SST files is mainly done by Exchange Reader, sstProcessor, and sstWriter. The whole data processing steps are as follows:

- 1. Reader reads data from the data source.
- 2. sstProcessor generates the SST file from the NebulaGraph's Schema information and uploads it to the HDFS. For details about the format of the SST file, see Data Storage Format.
- 3. sstWriter opens a file and inserts data. When generating SST files, keys must be written in sequence.
- 4. After the SST file is generated, RocksDB imports the SST file into NebulaGraph using the IngestExternalFile() method. For example:

When the IngestExternalFile() method is called, RocksDB copies the file to the data directory by default and blocks the RocksDB write operation. If the key range in the SST file overwrites the Memtable key range, flush the Memtable to the hard disk. After placing the SST file in an optimal location in the LSM tree, assign a global serial number to the file and turn on the write operation.

#### Data set

This topic takes the basketballplayer dataset as an example.

#### **Environment**

This example is done on MacOS. Here is the environment configuration information:

- Hardware specifications:
- CPU: 1.7 GHz Quad-Core Intel Core i7
- Memory: 16 GB
- Spark: 2.4.7, stand-alone
- Hadoop: 2.9.2, pseudo-distributed deployment
- NebulaGraph: 3.4.1.

#### **Prerequisites**

Before importing data, you need to confirm the following information:

- NebulaGraph has been installed and deployed with the following information:
- IP addresses and ports of Graph and Meta services.
- The user name and password with write permission to NebulaGraph.
- --ws\_storage\_http\_port in the Meta service configuration file is the same as --ws\_http\_port in the Storage service configuration file. For example, 19779.
- --ws\_meta\_http\_port in the Graph service configuration file is the same as --ws\_http\_port in the Meta service configuration file. For example, 19559.
- The information about the Schema, including names and properties of Tags and Edge types, and more.
- Exchange has been compiled, or download the compiled .jar file directly.
- Spark has been installed.
- JDK 1.8 or the later version has been installed and the environment variable JAVA\_HOME has been configured.
- The Hadoop service has been installed and started.



- To generate SST files of other data sources, see documents of the corresponding data source and check the prerequisites.
- To generate SST files only, users do not need to install the Hadoop service on the machine where the Storage service is deployed.
- To delete the SST file after the ingest (data import) operation, add the configuration -- move\_Files =true to the Storage Service configuration file.

#### Steps

STEP 1: CREATE THE SCHEMA IN NEBULAGRAPH

Analyze the data to create a Schema in NebulaGraph by following these steps:

1. Identify the Schema elements. The Schema elements in the NebulaGraph are shown in the following table.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge Type	follow	degree int
Edge Type	serve	start_year int, end_year int

2. Create a graph space basketballplayer in the NebulaGraph and create a Schema as shown below.

For more information, see Quick start workflow.

STEP 2: PROCESS CSV FILES

Confirm the following information:

1. Process CSV files to meet Schema requirements.



Exchange supports uploading CSV files with or without headers.

2. Obtain the CSV file storage path.

```
STEP 3: MODIFY CONFIGURATION FILES
```

After Exchange is compiled, copy the conf file target/classes/application.conf to set SST data source configuration. In this example, the copied file is called sst\_application.conf. For details on each configuration item, see Parameters in the configuration file.

```
{
  # Spark configuration
  spark: {
    app: {
        name: NebulaGraph Exchange 3.4.0
    }

  master:local

  driver: {
        cores: 1
        maxResultSize: 16
  }

  executor: {
```

```
memory:1G
  cores:{
   max: 16
# NebulaGraph configuration
nebula: {
    graph:["127.0.0.1:9669"]
    # the address of any of the meta services.
    # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta.
    meta:["127.0.0.1:9559"]
  user: root
  pswd: nebula
  space: basketballplayer
  # SST file configuration
      # The local directory that temporarily stores generated SST files
      local:"/tmp"
      # The path for storing the SST file in the HDFS
      remote:"/sst"
      # The NameNode address of HDFS
      hdfs.namenode: "hdfs://*.*.*:9000"
  # The connection parameters of clients
  connection: {
    # The timeout duration of socket connection and execution. Unit: milliseconds.
   timeout: 30000
  error: {
   # The maximum number of failures that will exit the application.
    # Failed import jobs are logged in the output path.
    output: /tmp/errors
  # Use Google's RateLimiter to limit requests to NebulaGraph.
  rate: {
    # Steady throughput of RateLimiter.
    # Get the allowed timeout duration from RateLimiter. Unit: milliseconds.
    timeout: 1000
# Processing vertices
tags: [
# Set the information about the Tag player.
    # Specify the Tag name defined in NebulaGraph.
    name: player
    type: {
    # Specify the data source file format to CSV.
      # Specify how to import the data into NebulaGraph: Client or SST.
    # Specify the path to the CSV file.
   # If the file is stored in HDFs, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx.csv". path: "hdfs://*.*.*.*9000/dataset/vertex_player.csv"
    # If the CSV file does not have a header, use [_c0, _c1, _c2, ..., _cn] to represent its header and indicate the columns as the source of the property values.
    # If the CSV file has a header, use the actual column name.
    fields: [_c1, _c2]
    # Specify the property name defined in NebulaGraph.
    # The sequence of fields and nebula fields must correspond to each other.
    nebula.fields: [age, name]
    # Specify a column of data in the table as the source of VIDs in NebulaGraph.
    # The value of vertex must be consistent with the column name in the above fields or csv.fields.
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    vertex: {
      field:_c0
    # The delimiter specified. The default value is comma.
    separator: ",
    # If the CSV file has a header, set the header to true.
# If the CSV file does not have a header, set the header to false. The default value is false.
```

```
header: false
    # The number of data written to NebulaGraph in a single batch.
   batch: 256
    # The number of Spark partitions.
   partition: 32
    # Whether to repartition data based on the number of partitions of graph spaces in NebulaGraph when generating the SST file.
    repartitionWithNebula: false
  # Set the information about the Tag Team.
    # Specify the Tag name defined in NebulaGraph.
    name: team
    type: {
     # Specify the data source file format to CSV.
     source: csv
     # Specify how to import the data into NebulaGraph: Client or SST.
    # Specify the path to the CSV file.
   # If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx.csv".path: "hdfs://*.*.*.9000/dataset/vertex_team.csv"
    # If the CSV file does not have a header, use [_c0, _c1, _c2, ..., _cn] to represent its header and indicate the columns as the source of the property values.
    # If the CSV file has a header, use the actual column name.
    fields: [_c1]
    # Specify the property name defined in NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [name]
    # Specify a column of data in the table as the source of VIDs in NebulaGraph.
    # The value of vertex must be consistent with the column name in the above fields or csv.fields.
    \mbox{\#} Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    vertex: {
     field:_c0
    }
    # The delimiter specified. The default value is comma
    separator: ",
    # If the CSV file has a header, set the header to true.
# If the CSV file does not have a header, set the header to false. The default value is false.
    # The number of data written to NebulaGraph in a single batch.
    batch: 256
    # The number of Spark partitions.
    partition: 32
    # Whether to repartition data based on the number of partitions of graph spaces in NebulaGraph when generating the SST file.
    repartitionWithNebula: false
 # If more vertices need to be added, refer to the previous configuration to add them.
# Processing edges
edges: [
# Set the information about the Edge Type follow.
    \mbox{\tt\#} The Edge Type name defined in NebulaGraph.
    name: follow
      # Specify the data source file format to CSV.
     # Specify how to import the data into NebulaGraph: Client or SST.
    # Specify the path to the CSV file.
   # If the file is stored in HDFS, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx.csv". path: "hdfs://*.*.*.*:9000/dataset/edge_follow.csv"
    # If the CSV file does not have a header, use [_c0, _c1, _c2, ..., _cn] to represent its header and indicate the columns as the source of the property values. # If the CSV file has a header, use the actual column name.
    fields: [_c2]
    # Specify the property name defined in NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
    nebula.fields: [degree]
    # Specify a column as the source for the source and destination vertices.
    # The value of vertex must be consistent with the column name in the above fields or csv.fields.
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    source: {
```

```
field: _c0
    target: {
     field: _c1
    # The delimiter specified. The default value is comma.
   separator: ",
   # (Optional) Specify a column as the source of the rank.
    #ranking: rank
    # If the CSV file has a header, set the header to true.
    # If the CSV file does not have a header, set the header to false. The default value is false.
    # The number of data written to NebulaGraph in a single batch.
    # The number of Spark partitions.
   partition: 32
    # Whether to repartition data based on the number of partitions of graph spaces in NebulaGraph when generating the SST file.
    repartitionWithNebula: false
  # Set the information about the Edge Type serve.
    # Specify the Edge type name defined in NebulaGraph.
    name: serve
    type: {
     # Specify the data source file format to CSV.
     source: csv
     \mbox{\# Specify how to import the data into NebulaGraph: Client or SST.}
     sink: sst
    # Specify the path to the CSV file.
   # If the file is stored in HDFs, use double quotation marks to enclose the file path, starting with hdfs://. For example, "hdfs://ip:port/xx/xx.csv". path: "hdfs://*.*.*.9000/dataset/edge_serve.csv"
    # If the CSV file does not have a header, use [_c0, _c1, _c2, ..., _cn] to represent its header and indicate the columns as the source of the property values.
    # If the CSV file has a header, use the actual column name.
    fields: [_c2,_c3]
    # Specify the property name defined in NebulaGraph.
    # The sequence of fields and nebula.fields must correspond to each other.
   nebula.fields: [start_year, end_year]
    # Specify a column as the source for the source and destination vertices.
    # The value of vertex must be consistent with the column name in the above fields or csv.fields.
    # Currently, NebulaGraph 3.4.1 supports only strings or integers of VID.
    source: {
     field: c0
    target: {
     field: c1
    # The delimiter specified. The default value is comma.
    separator: ",
    # (Optional) Specify a column as the source of the rank.
    #ranking: _c5
    # If the CSV file has a header, set the header to true.
    # If the CSV file does not have a header, set the header to false. The default value is false.
    # The number of data written to NebulaGraph in a single batch.
    # The number of Spark partitions.
    # Whether to repartition data based on the number of partitions of graph spaces in NebulaGraph when generating the SST file.
    repartitionWithNebula: false
# If more edges need to be added, refer to the previous configuration to add them.
```

#### STEP 4: GENERATE THE SST FILE

Run the following command to generate the SST file from the CSV source file. For a description of the parameters, see Options for import.

\${SPARK\_HOME}/bin/spark-submit --master "local" --conf spark.sql.shuffle.partition=<shuffle\_concurrency> --class com.vesoft.nebula.exchange.Exchange <nebula-exchange-3.4.0.jar\_path> -c



When generating SST files, the shuffle operation of Spark will be involved. Note that the configuration of spark.sql.shuffle.partition should be added when you submit the command.



JAR packages are available in two ways: compiled them yourself, or download the compiled .jar file directly.

#### For example:

\${\$PARK\_HOME}/bin/spark-submit --master "local" --conf spark.sql.shuffle.partition=200 --class com.vesoft.nebula.exchange.Exchange /root/nebula-exchange/nebula-exchange/target/nebula-exchange/3.4.0.jar -c /root/nebula-exchange/nebula-exchange/target/classes/sst\_application.conf

After the task is complete, you can view the generated SST file in the /sst directory (specified by the nebula.path.remote parameter) on HDFS.



If you modify the Schema, such as rebuilding the graph space, modifying the Tag, or modifying the Edge type, you need to regenerate the SST file because the SST file verifies the space ID, Tag ID, and Edge ID.

STEP 5: IMPORT THE SST FILE



Confirm the following information before importing:

- Confirm that the Hadoop service has been deployed on all the machines where the Storage service is deployed, and configure HADOOP\_HOME and JAVA\_HOME.
- The --ws\_storage\_http\_port in the Meta service configuration file (add it manually if it does not exist) is the same as the --ws\_http\_port in the Storage service configuration file. For example, both are 19779.
- The --ws\_meta\_http\_port in the Graph service configuration file (add it manually if it does not exist) is the same as the --ws\_http\_port in the Meta service configuration file. For example, both are 19559.

Connect to the NebulaGraph database using the client tool and import the SST file as follows:

 $1. \ \mbox{Run}$  the following command to select the graph space you created earlier.

nebula> USE basketballplayer;

2. Run the following command to download the SST file:

nebula> SUBMIT JOB DOWNLOAD HDFS "hdfs://<hadoop\_address>:<hadoop\_port>/<sst\_file\_path>";

For example:

nebula> SUBMIT JOB DOWNLOAD HDFS "hdfs://\*.\*.\*:9000/sst";

3. Run the following command to import the SST file:

nebula> SUBMIT JOB INGEST;

- 918/1069 - 2023 Vesoft Inc.

#### Q Note

- To download the SST file again, delete the download folder in the space ID in the data/storage/nebula directory in the NebulaGraph installation path, and then download the SST file again. If the space has multiple copies, the download folder needs to be deleted on all machines where the copies are saved.
- If there is a problem with the import and re-importing is required, re-execute SUBMIT JOB INGEST; .

STEP 6: (OPTIONAL) VALIDATE DATA

Users can verify that data has been imported by executing a query in the NebulaGraph client (for example, NebulaGraph Studio). For example:

LOOKUP ON player YIELD id(vertex);

Users can also run the SHOW STATS command to view statistics.

STEP 7: (OPTIONAL) REBUILD INDEXES IN NEBULAGRAPH

With the data imported, users can recreate and rebuild indexes in NebulaGraph. For details, see Index overview.

# 18.4.16 Export data from NebulaGraph

The Exchange allows you to export data from NebulaGraph to a CSV file or another NebulaGraph space (supporting different NebulaGraph clusters). This topic describes the specific procedure.



Only Exchange Enterprise Edition supports exporting data from NebulaGraph.

#### Preparation

This example is completed on a virtual machine equipped with Linux. The hardware and software you need to prepare before exporting data are as follows.

#### HARDWARE

Туре	Information
CPU	4 Intel(R) Xeon(R) Platinum 8260 CPU @ 2.30GHz
Memory	16G
Hard disk	50G

SYSTEM

CentOS 7.9.2009

### SOFTWARE

Name	Version
JDK	1.8.0
Hadoop	2.10.1
Scala	2.12.11
Spark	2.4.7
NebulaGraph	3.4.1

#### DATASET

As the data source, NebulaGraph stores the basketballplayer dataset in this example, the Schema elements of which are shown as follows.

Element	Name	Property
Tag	player	name string, age int
Tag	team	name string
Edge type	follow	degree int
Edge type	serve	start_year int, end_year int

# Steps

- 1. Get the JAR file of Exchange Enterprise Edition from the NebulaGraph Enterprise Edition Package.
- 2. Modify the configuration file.

Exchange Enterprise Edition provides the configuration template export\_to\_csv.conf and export\_to\_nebula.conf for exporting NebulaGraph data. For details, see Exchange parameters. The core content of the configuration file used in this example is as follows:

• Export to a CSV file:

```
# Use the command to submit the exchange job:
# spark-submit \
# --master "spark://master_ip:7077" \
# --driver-memory=2G --executor-memory=30G \
# --total-executor-cores=60 --executor-cores=20 \
# --class com.vesoft.nebula.exchange.Exchange \
# nebula-exchange-3.0-SNAPSHOT.jar -c export_to_csv.conf
  # Spark config
  spark: {
       name: NebulaGraph Exchange
  # Nebula Graph config
  # if you export nebula data to csv, please ignore these nebula config
  nebula: {
      graph:["127.0.0.1:9669"]
      # the address of any of the meta services.
      # if your NebulaGraph server is in virtual network like k8s, please config the leader address of meta. meta: ["127.0.0.1:9559"]
    user: root
    pswd: nebula
    space: test
    # nebula client connection parameters
    connection {
       # socket connect & execute timeout, unit: millisecond
      timeout: 30000
    error: {
      # max number of failures, if the number of failures is bigger than max, then exit the application.
       # failed data will be recorded in output path, format with ngql
      output: /tmp/errors
    # use google's RateLimiter to limit the requests send to NebulaGraph
    rate: {
    # the stable throughput of RateLimiter
      # Acquires a permit from RateLimiter, unit: MILLISECONDS
# if it can't be obtained within the specified timeout, then give up the request.
       timeout: 1000
  # Processing tags
  tags: [
      # you can ignore the tag name when export nebula data to csv
       name: tag-name-1
      type: {
        source: nebula
         sink: csv
      metaAddress:"127.0.0.1:9559"
      space:"test"
label:"person"
       # config the fields you want to export from nebula
      fields: [nebula-field-0, nebula-field-1, nebula-field-2] noFields:false # default false, if true, just export id
       # config the path to save your csv file. if your file in not in hdfs, config "file:///path/ test.csv"
       path: "hdfs://ip:port/path/person'
       separator: "
       header: true
  # process edges
  edges: [
      # you can ignore the edge name when export nebula data to csv
       name: edge-name-1
       type: {
         source: nebula
        sink: csv
```

```
metaAddress:"127.0.0.1:9559"
    space:"test"
    label:"friend"
    # config the fields you want to export from nebula
    fields: [nebula-field-0, nebula-field-1, nebula-field-2]
    noFields:false # default false, if true, just export id
    partition: 60
    # config the path to save your csv file. if your file in not in hdfs, config "file:///path/ test.csv"
    path: "hdfs://ip:port/path/friend"
    separator: ","
    header: true
    }
}
```

#### • Export to another graph space:

```
# Use the command to submit the exchange job:
# spark-submit \
# --master "spark://master_ip:7077" \
# --driver-memory=2G --executor-memory=30G \
# --total-executor-cores=60 --executor-cores=20 \
# --class com.vesoft.nebula.exchange.Exchange \
# nebula-exchange-3.0-SNAPSHOT.jar -c export_to_nebula.conf
 # Spark config
 spark: {
      name: NebulaGraph Exchange
 # Nebula Graph config, just config the sink nebula information
 nebula: {
    address:{
     graph:["127.0.0.1:9669"]
      # the address of any of the meta services
      meta:["127.0.0.1:9559"]
    pswd: nebula
    space: test
    # nebula client connection parameters
    connection {
      \mbox{\tt\#} socket connect & execute timeout, unit: millisecond
      timeout: 30000
    error: {
      # max number of failures, if the number of failures is bigger than max, then exit the application.
      max: 32
      # failed data will be recorded in output path, format with ngql
      output: /tmp/errors
    \mbox{\tt\#} use google's RateLimiter to limit the requests send to NebulaGraph
    rate: {
     # the stable throughput of RateLimiter
      limit: 1024
      # Acquires a permit from RateLimiter, unit: MILLISECONDS
      # if it can't be obtained within the specified timeout, then give up the request.
      timeout: 1000
  # Processing tags
  tags: [
      name: tag-name-1
      type: {
       source: nebula
        sink: client
      # data source nebula config
      metaAddress:"127.0.0.1:9559"
space:"test"
      label:"person"
      # mapping the fields of the original NebulaGraph to the fields of the target NebulaGraph. fields: [source_nebula-field-0, source_nebula-field-1, source_nebula-field-2]
      nebula.fields: [target_nebula-field-0, target_nebula-field-1, target_nebula-field-2]
      limit:10000
      vertex: _vertexId # must be `_vertexId`
      batch: 2000
      partition: 60
  # process edges
 edges: [
```

```
{
  name: edge-name-1
  type: {
    source: csv
    sink: client
  }
  # data source nebula config
  metaAddress: "127.0.0.1:9559"
  space: "test"
  label: "friend"
  fields: [source_nebula-field-0, source_nebula-field-1, source_nebula-field-2]
  nebula.fields: [target_nebula-field-0, target_nebula-field-1, target_nebula-field-2]
  limit:1000
  source: _srcId # must be `_srcId`
  target: _dstId # must be `_dstId`
  ranking: source_nebula-field-2
  batch: 2000
  partition: 60
}
```

3. Export data from NebulaGraph with the following command.

#### Q Note

The parameters of the Driver and Executor process can be modified based on your own machine configuration.

```
<spark_install_path>/bin/spark-submit --master "spark://<master_ip>:7077" \
--driver-memory=26 --executor-memory=306 \
--total-executor-cores=60 --executor-cores=20 \
--class com.vesoft.nebula.exchange.Exchange nebula-exchange-x.y.z.jar_path> \
-c <conf_file_path>
```

The following is an example command to export the data to a CSV file.

```
$ ./spark-submit --master "spark://192.168.10.100:7077" \
--driver-memory=26 --executor-memory=306 \
--total-executor-cores=60 --executor-cores=20 \
--class com.vesoft.nebula.exchange Exchange ~/exchange-ent/nebula-exchange-ent-3.4.0.jar \
-c ~/exchange-ent/export_to_csv.conf
```

- 4. Check the exported data.
- Export to a CSV file:

Check whether the CSV file is successfully generated under the target path, and check the contents of the CSV file to ensure that the data export is successful.

```
$ hadoop fs -ls /vertex/player
Found 11 items
-rw-r--r-- 3 nebula supergroup
                                             0 2021-11-05 07:36 /vertex/player/_SUCCESS
-rw-r--r-- 3 nebula supergroup
                                           160 2021-11-05 07:36 /vertex/player/
163 2021-11-05 07:36 /vertex/player/
                                                                                       part-00000-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
                                                                                       part-00001-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--
             3 nebula supergroup
             3 nebula supergroup
                                            172 2021-11-05 07:36 /vertex/player/
                                                                                       part-00002-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
                                                                                       part-00003-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--
             3 nebula supergroup
                                           172 2021-11-05 07:36 /vertex/player/
144 2021-11-05 07:36 /vertex/player/
                                                                                       part-00004-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r-- 3 nebula supergroup
             3 nebula supergroup
                                            173 2021-11-05 07:36 /vertex/player/
                                                                                       part-00005-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--
             3 nebula supergroup
                                           160 2021-11-05 07:36 /vertex/player/
148 2021-11-05 07:36 /vertex/player/
                                                                                       part-00006-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
             3 nebula supergroup
                                                                                       part-00007-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
             3 nebula supergroup
                                           125 2021-11-05 07:36 /vertex/player/
                                                                                       part-00008-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
-rw-r--r--
             3 nebula supergroup
                                           119 2021-11-05 07:36 /vertex/player/
                                                                                       part-00009-17293020-ba2e-4243-b834-34495c0536b3-c000.csv
```

• Export to another graph space:

Log in to the new graph space and check the statistics through SUBMIT JOB STATS and SHOW STATS commands to ensure the data export is successful.

# 18.5 Exchange FAQ

#### 18.5.1 Compilation

#### Q: Some packages not in central repository failed to download, error: Could not resolve dependencies for project xxx

Please check the mirror part of Maven installation directory libexec/conf/settings.xml:

```
<mirror>
     <id>alimaven</id>
     <mirror0f>central</mirror0f>
     <name>aliyun maven</name>
     <url>http://maven.aliyun.com/nexus/content/repositories/central/</url>
</mirror>
```

 $Check \ whether \ the \ value \ of \ \textit{mirrorOf} \ \ is \ configured \ to \ ^\star. \ If \ it \ is, \ change \ it \ to \ \ central \ \ or \ ^\star, !SparkPackagesRepo, !bintray-streamnative-maven \ .$ 

Reason: There are two dependency packages in Exchange's pom.xml that are not in Maven's central repository. pom.xml configures the repository address for these two dependencies. If the mirror0f value for the mirror address configured in Maven is \*, all dependencies will be downloaded from the Central repository, causing the download to fail.

#### Q: Unable to download SNAPSHOT packages when compiling Exchange

Problem description: The system reports Could not find artifact com.vesoft:client:jar:xxx-SNAPSHOT when compiling.

Cause: There is no local Maven repository for storing or downloading SNAPSHOT packages. The default central repository in Maven only stores official releases, not development versions (SNAPSHOT).

Solution: Add the following configuration in the profiles scope of Maven's setting.xml file:

### 18.5.2 Execution

#### Q: Error: java.lang.ClassNotFoundException: com.vesoft.nebula.exchange.Exchange

To submit a task in Yarn-Cluster mode, run the following command, especially the two '--conf' commands in the example.

```
$SPARK_HOME/bin/spark-submit --class com.vesoft.nebula.exchange.Exchange \
--master yarn-cluster \
--files application.conf \
--conf spark.driver.extraClassPath=./ \
--conf spark.executor.extraClassPath=./ \
nebula-exchange-3.0.jar \
-c application.conf
```

# Q: Error: method name xxx not found

Generally, the port configuration is incorrect. Check the port configuration of the Meta service, Graph service, and Storage service.

#### Q: Error: NoSuchMethod, MethodNotFound (Exception in thread "main" java.lang.NoSuchMethodError, etc)

Most errors are caused by JAR package conflicts or version conflicts. Check whether the version of the error reporting service is the same as that used in Exchange, especially Spark, Scala, and Hive.

#### Q: When Exchange imports Hive data, error: Exception in thread "main" org.apache.spark.sql.AnalysisException: Table or view not found

Check whether the h parameter is omitted in the command for submitting the Exchange task and whether the table and database are correct, and run the user-configured exec statement in spark-SOL to verify the correctness of the exec statement.

#### Q: Run error: com.facebook.thrift.protocol.TProtocolException: Expected protocol id xxx

Check that the NebulaGraph service port is configured correctly.

- For source, RPM, or DEB installations, configure the port number corresponding to --port in the configuration file for each service.
- For docker installation, configure the docker mapped port number as follows:

Execute docker-compose ps in the nebula-docker-compose directory, for example:

```
$ docker-compose ps
                Name
                                                      Command
                                                                                   State
                                                                                                                                                           Ports
                                                                               Up (healthy)
nebula-docker-compose\_graphd\_1
                                         /usr/local/nebula/bin/nebu ...
                                                                                                 {\color{red}0.0.0.0:33205}{\scriptsize ->}19669/\text{tcp}, {\color{red}0.0.0.0:33204}{\scriptsize ->}19670/\text{tcp}, {\color{red}0.0.0.0:9669}{\scriptsize ->}9669/\text{tcp}
                                                                                                 0.0.0.0:33165->19559/tcp, 0.0.0.0:33162->19560/tcp, 0.0.0.0:33167->9559/tcp, 9560/tcp
                                          ./bin/nebula-metad --flagf ...
                                                                                Up (healthy)
nebula-docker-compose metad0 1
                                          ./bin/nebula-metad --flagf ...
nebula-docker-compose\_metad1\_1
                                                                                Up (healthy)
                                                                                                 0.0.0.0:33166->19559/tcp, 0.0.0.0:33163->19560/tcp, 0.0.0.0:33168->9559/tcp, 9560/tcp
nebula-docker-compose_metad2_1
                                           /bin/nebula-metad --flagf
                                                                                Up (healthy)
                                                                                                 0.0.0.0:33161->19559/tcp, 0.0.0.0:33160->19560/tcp, 0.0.0.0:33164->9559/tcp, 9560/tcp
0.0.0.0:33180->19779/tcp, 0.0.0.0:33178->19780/tcp, 9777/tcp, 9778/tcp, 0.0.0.0:33183->9779/tcp, 9780/
                                         ./bin/nebula-storaged --fl ...
nebula-docker-compose_storaged0_1
                                                                               Up (healthy)
nebula-docker-compose_storaged1_1
                                         ./bin/nebula-storaged --fl ... Up (healthy) 0.0.0.0:33175->19779/tcp, 0.0.0.0:33172->19780/tcp, 9777/tcp, 9778/tcp, 0.0.0.0:33177->9779/tcp, 9780/
nebula-docker-compose_storaged2_1
                                         ./bin/nebula-storaged --fl ... Up (healthy) 0.0.0.0:33184->19779/tcp, 0.0.0.0:33181->19780/tcp, 9777/tcp, 9778/tcp, 0.0.0.0:33185->9779/tcp, 9780/
```

Check the Ports column to find the docker mapped port number, for example:

- The port number available for Graph service is 9669.
- The port number for Meta service are 33167, 33168, 33164.
- The port number for Storage service are 33183, 33177, 33185.

#### Q: Error: Exception in thread "main" com.facebook.thrift.protocol.TProtocolException: The field 'code' has been assigned the invalid value -4

Check whether the version of Exchange is the same as that of NebulaGraph. For more information, see Limitations.

#### Q: How to correct the messy code when importing Hive data into NebulaGraph?

It may happen if the property value of the data in Hive contains Chinese characters. The solution is to add the following options before the JAR package path in the import command:

```
--conf spark.driver.extraJavaOptions=-Dfile.encoding=utf-8
--conf spark.executor.extraJavaOptions=-Dfile.encoding=utf-8
```

#### Namely:

```
<spark_install_path>/bin/spark-submit --master "local" \
--conf spark.driver.extraJavaOptions=-Dfile.encoding=utf-8 \
--conf spark.executor.extraJavaOptions=-Dfile.encoding=utf-8 \
--class com.vesoft.nebula.exchange.Exchange \
<nebula-exchange-3.x.y.jar_path> -c <application.conf_path>
```

#### In YARN, use the following command:

```
<spark_install_path>/bin/spark-submit \
--class com.vesoft.nebula.exchange.Exchange \
--master yarn-cluster \
--files <application.conf_path> \
--conf spark.driver.extraClassPath=./ \
--conf spark.executor.extraClassPath=./ \
```

```
--conf spark.driver.extraJavaOptions=-Dfile.encoding=utf-8 \
--conf spark.executor.extraJavaOptions=-Dfile.encoding=utf-8 \
<nebula-exchange-3.x.y.jar_path> \
-c application.conf
```

#### Q: org.rocksdb.RocksDBException: While open a file for appending: /path/sst/1-xxx.sst: No such file or directory

Solution:

- 1. Check if /path exists. If not, or if the path is set incorrectly, create or correct it.
- 2. Check if Spark's current user on each machine has the operation permission on /path. If not, grant the permission.

# 18.5.3 Configuration

#### Q: Which configuration fields will affect import performance?

- batch: The number of data contained in each nGQL statement sent to the NebulaGraph service.
- partition: The number of Spark data partitions, indicating the number of concurrent data imports.
- nebula.rate: Get a token from the token bucket before sending a request to NebulaGraph.
  - limit: Represents the size of the token bucket.
  - timeout: Represents the timeout period for obtaining the token.

The values of these four parameters can be adjusted appropriately according to the machine performance. If the leader of the Storage service changes during the import process, you can adjust the values of these four parameters to reduce the import speed.

#### 18.5.4 Others

#### Q: Which versions of NebulaGraph are supported by Exchange?

See Limitations.

#### Q: What is the relationship between Exchange and Spark Writer?

Exchange is the Spark application developed based on Spark Writer. Both are suitable for bulk migration of cluster data to NebulaGraph in a distributed environment, but later maintenance work will be focused on Exchange. Compared with Spark Writer, Exchange has the following improvements:

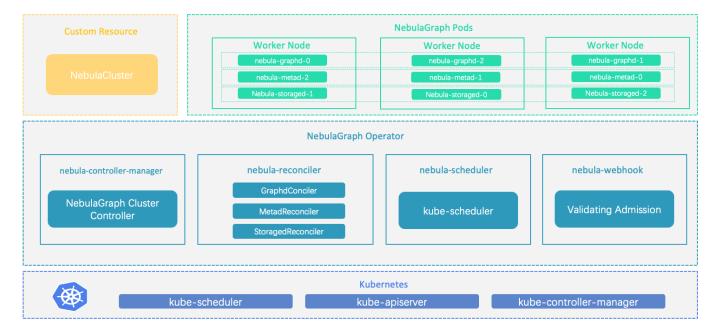
- It supports more abundant data sources, such as MySQL, Neo4j, Hive, HBase, Kafka, Pulsar, etc.
- It fixed some problems of Spark Writer. For example, when Spark reads data from HDFS, the default source data is String, which may be different from the NebulaGraph's Schema. So Exchange adds automatic data type matching and type conversion. When the data type in the NebulaGraph's Schema is non-String (e.g. double), Exchange converts the source data of String type to the corresponding type.

# 19. NebulaGraph Operator

# 19.1 What is NebulaGraph Operator

# 19.1.1 Concept

NebulaGraph Operator is a tool to automate the deployment, operation, and maintenance of NebulaGraph clusters on Kubernetes. Building upon the excellent scalability mechanism of Kubernetes, NebulaGraph introduced its operation and maintenance knowledge into the Kubernetes system, which makes NebulaGraph a real cloud-native graph database.



#### 19.1.2 How it works

For resource types that do not exist within Kubernetes, you can register them by adding custom API objects. The common way is to use the CustomResourceDefinition.

NebulaGraph Operator abstracts the deployment management of NebulaGraph clusters as a CRD. By combining multiple built-in API objects including StatefulSet, Service, and ConfigMap, the routine management and maintenance of a NebulaGraph cluster are coded as a control loop in the Kubernetes system. When a CR instance is submitted, NebulaGraph Operator drives database clusters to the final state according to the control process.

#### 19.1.3 Features

The following features are already available in NebulaGraph Operator:

- Deploy and uninstall clusters: NebulaGraph Operator simplifies the process of deploying and uninstalling clusters for users. NebulaGraph Operator allows you to quickly create, update, or delete a NebulaGraph cluster by simply providing the corresponding CR file. For more information, see Deploy NebulaGraph Clusters with Kubectl or Deploy NebulaGraph Clusters with Helm.
- Scale clusters: NebulaGraph Operator calls NebulaGraph's native scaling interfaces in a control loop to implement the scaling logic. You can simply perform scaling operations with YAML configurations and ensure the stability of data. For more information, see Scale clusters with Kubectl or Scale clusters with Helm.
- Cluster Upgrade: NebulaGraph Operator supports cluster upgrading from version 3.0.0 to version 3.4.0.
- Backup and Recovery: NebulaGraph supports data backup and recovery. Users can use NebulaGraph Operator to backup the data of the NebulaGraph cluster to storage services that are compatible with the S3 protocol, and can also restore data to the cluster from the storage service. For details, see Backup and restore using NebulaGraph Operator.
- **Self-Healing**: NebulaGraph Operator calls interfaces provided by NebulaGraph clusters to dynamically sense cluster service status. Once an exception is detected, NebulaGraph Operator performs fault tolerance. For more information, see Self-Healing.
- Balance Scheduling: Based on the scheduler extension interface, the scheduler provided by NebulaGraph Operator evenly
  distributes Pods in a NebulaGraph cluster across all nodes.

#### 19.1.4 Limitations

#### Version limitations

NebulaGraph Operator does not support the v1.x version of NebulaGraph. NebulaGraph Operator version and the corresponding NebulaGraph version are as follows:

NebulaGraph	<b>NebulaGraph Operator</b>
$3.0.0 \sim 3.4.1$	1.3.0, 1.4.0 ~ 1.4.2
$3.0.0 \sim 3.3.x$	1.0.0, 1.1.0, 1.2.0
$2.5.x \sim 2.6.x$	0.9.0
2.5.x	0.8.0

# Lacy version compatibility

- The 1.x version NebulaGraph Operator is not compatible with NebulaGraph of version below v3.x.
- Starting from NebulaGraph Operator 0.9.0, logs and data are stored separately. Using NebulaGraph Operator 0.9.0 or later versions to manage a NebulaGraph 2.5.x cluster created with Operator 0.8.0 can cause compatibility issues. You can backup the data of the NebulaGraph 2.5.x cluster and then create a 2.6.x cluster with Operator 0.9.0.

#### **Feature limitations**

The NebulaGraph Operator scaling feature is only available for the Enterprise Edition of NebulaGraph clusters and does not support scaling the Community Edition version of NebulaGraph clusters.

#### 19.1.5 Release note

Release

# 19.2 Overview of using NebulaGraph Operator

To use NebulaGraph Operator to connect to NebulaGraph databases, see steps as follows:

- 1. Install NebulaGraph Operator.
- 2. Create a NebulaGraph cluster.

For more information, see Deploy NebulaGraph clusters with Kubectl or Deploy NebulaGraph clusters with Helm.

3. Connect to a NebulaGraph database.

# 19.3 Deploy NebulaGraph Operator

You can deploy NebulaGraph Operator with Helm.

#### 19.3.1 Background

NebulaGraph Operator automates the management of NebulaGraph clusters, and eliminates the need for you to install, scale, upgrade, and uninstall NebulaGraph clusters, which lightens the burden on managing different application versions.

# 19.3.2 Prerequisites

Before installing NebulaGraph Operator, you need to install the following software and ensure the correct version of the software  $\cdot$ 

Software	Requirement
Kubernetes	>= 1.16
Helm	>= 3.2.0
CoreDNS	>= 1.6.0



- If using a role-based access control policy, you need to enable RBAC (optional).
- CoreDNS is a flexible and scalable DNS server that is installed for Pods in NebulaGraph clusters.

# 19.3.3 Steps

#### Install NebulaGraph Operator

1. Add the NebulaGraph Operator Helm repository.

helm repo add nebula-operator https://vesoft-inc.github.io/nebula-operator/charts

 $2. \ Update \ information \ of \ available \ charts \ locally \ from \ repositories.$ 

helm repo update

For more information about  $\ensuremath{\text{helm}}$  repo , see  $\ensuremath{\text{Helm}}$  Repo.

3. Install NebulaGraph Operator.

helm install nebula-operator nebula-operator/nebula-operator --namespace=<namespace\_name> --version=\${chart\_version}

For example, the command to install NebulaGraph Operator of version 1.4.2 is as follows.

helm install nebula-operator nebula-operator/nebula-operator --namespace=nebula-operator-system --version=1.4.2

- nebula-operator-system is a user-created namespace name. If you have not created this namespace, run kubectl create namespace nebula-operator-system to create one. You can also use a different name.
- 1.4.2 is the version of the NebulaGraph Operator chart. When not specifying --version, the latest version of the nebula-operator chart is used by default. Run helm search repo -l nebula-operator to see chart versions.

You can customize the configuration items of the NebulaGraph Operator chart before running the installation command. For more information, see **Customize Helm charts** below.

- 931/1069 - 2023 Vesoft Inc.

#### **Customize Helm charts**

Run helm show values [CHART] [flags] to see configurable options.

#### For example:

```
image:
nebulaOperator:
    image: vesoft/nebula-operator:v1.4.2
imagePullPolicy: Always
  kubeRBACProxy:
image: gcr.io/kubebuilder/kube-rbac-proxy:v0.8.0
imagePullPolicy: Always
  kubeScheduler:
  image: k8s.gcr.io/kube-scheduler:v1.18.8
     imagePullPolicy: Always
imagePullSecrets: []
controllerManager:
 replicas: 2
env: []
resources:
Limits:
       cpu: 200m
    memory: 200Mi
requests:
      cpu: 100m
memory: 100Mi
admissionWebhook:
create: true
scheduler:
create: true
  schedulerName: nebula-scheduler
  replicas: 2
env: []
resources:
Limits:
       cpu: 200m
    memory: 20Mi
requests:
       cpu: 100m
       memory: 100Mi
```

Part of the above parameters are described as follows:

Parameter	Default value	Description
image.nebulaOperator.image	vesoft/nebula- operator:v1.4.2	The image of NebulaGraph Operator, version of which is $1.4.2.$
image.nebulaOperator.imagePullPolicy	IfNotPresent	The image pull policy in Kubernetes.
imagePullSecrets	-	The image pull secret in Kubernetes.
kubernetesClusterDomain	cluster.local	The cluster domain.
controllerManager.create	true	Whether to enable the controller-manager component.
controllerManager.replicas	2	The numeric value of controller-manager replicas.
admissionWebhook.create	true	Whether to enable Admission Webhook.
shceduler.create	true	Whether to enable Scheduler.
shceduler.schedulerName	nebula-scheduler	The Scheduler name.
shceduler.replicas	2	The numeric value of nebula-scheduler replicas.

You can run helm install [NAME] [CHART] [flags] to specify chart configurations when installing a chart. For more information, see Customizing the Chart Before Installing.

The following example shows how to specify the NebulaGraph Operator's AdmissionWebhook mechanism to be turned off when you install NebulaGraph Operator (AdmissionWebhook is enabled by default):

helm install nebula-operator nebula-operator/nebula-operator --namespace=<nebula-operator-system> --set admissionWebhook.create=false

For more information about helm install, see Helm Install.

### **Update NebulaGraph Operator**

1. Update the information of available charts locally from chart repositories.

helm repo update

- 2. Update NebulaGraph Operator by passing configuration parameters via -set or -values flag.
- --set : Overrides values using the command line.
- --values (or -f): Overrides values using YAML files.

For configurable items, see the above-mentioned section **Customize Helm charts**.

For example, to disable the AdmissionWebhook (AdmissionWebhook is enabled by default), run the following command:

helm upgrade nebula-operator nebula-operator/nebula-operator --namespace=nebula-operator-system --version=1.4.2 --set admissionWebhook.create=false

For more information, see Helm upgrade.

- 933/1069 - 2023 Vesoft Inc.

#### Upgrade NebulaGraph Operator

#### L Jacy version compatibility

- Does not support upgrading 0.9.0 and below version NebulaGraph Operator to 1.x.
- The 1.x version NebulaGraph Operator is not compatible with NebulaGraph of version below v3.x.
- 1. Update the information of available charts locally from chart repositories.

helm repo update

2. Upgrade Operator to v1.4.2.

```
helm upgrade nebula-operator nebula-operator/nebula-operator --namespace=<namespace_name> --version=1.4.2
```

#### For example:

helm upgrade nebula-operator nebula-operator/nebula-operator --namespace=nebula-operator-system --version=1.4.2

#### Output:

```
Release "nebula-operator" has been upgraded. Happy Helming!
NAME: nebula-operator

LAST DEPLOYED: Tue Apr 16 02:21:08 2022

NAMESPACE: nebula-operator-system
STATUS: deployed
REVISION: 3

TEST SUITE: None
NOTES:
NebulaGraph Operator installed!
```

3. Pull the latest CRD configuration file.



You need to upgrade the corresponding CRD configurations after NebulaGraph Operator is upgraded. Otherwise, the creation of NebulaGraph clusters will fail. For information about the CRD configurations, see <a href="mailto:apps.nebula-graph.io\_nebulaclusters.yaml">apps.nebula-graph.io\_nebulaclusters.yaml</a>.

a. Pull the NebulaGraph Operator chart package.

```
helm pull nebula-operator/nebula-operator --version=1.4.2
```

- $\bullet \ \ \hbox{$\hbox{$--$version}:$ The Nebula Graph Operator version you want to upgrade to. If not specified, the latest version will be pulled.}$
- b. Run tar -zxvf to unpack the charts.

For example: To unpack v1.4.2 chart to the /tmp path, run the following command:

```
tar -zxvf nebula-operator-1.4.2.tgz -C /tmp
```

- -C /tmp: If not specified, the chart files will be unpacked to the current directory.
- 4. Upgrade the CRD configuration file in the nebula-operator directory.

```
kubectl apply -f crds/nebulacluster.yaml
```

#### Output:

 $custom resource definition. a piextensions. k8s. io/nebula clusters. apps. nebula-graph. io\ configured to the configuration of the c$ 

# Uninstall NebulaGraph Operator

 $1.\ Uninstall\ the\ Nebula Graph\ Operator\ chart.$ 

helm uninstall nebula-operator --namespace=<nebula-operator-system>

#### 2. Delete CRD.

kubectl delete crd nebulaclusters.apps.nebula-graph.io

# 19.3.4 What's next

Automate the deployment of NebulaGraph clusters with NebulaGraph Operator. For more information, see Deploy NebulaGraph Clusters with Kubectl or Deploy NebulaGraph Clusters with Helm.

# 19.4 Deploy clusters

# 19.4.1 Deploy NebulaGraph clusters with Kubectl

# Lacy version compatibility

The 1.x version NebulaGraph Operator is not compatible with NebulaGraph of version below v3.x.

#### Prerequisites

- Install NebulaGraph Operator
- $\bullet$  You have prepared the license file for Nebula Graph Enterprise Edition clusters.



The license file is required only when creating a NebulaGraph Enterprise Edition cluster.

# Create clusters

The following example shows how to create a NebulaGraph cluster by creating a cluster named nebula.

- 1. Create a file named <code>apps\_vlalphal\_nebulacluster.yaml</code> .
- The file contents for a NebulaGraph Community cluster are as follows:

```
apiVersion: apps.nebula-graph.io/vlalpha1 kind: NebulaCluster metadata:
spec:
  graphd:
     resources:
       requests:
         cpu: "500m"
memory: "500Mi"
       limits:
cpu: "1"
     memory: "1Gi"
replicas: 1
     image: vesoft/nebula-graphd
version: v3.4.1
     logVolumeClaim:
       resources:
         requests:
       storage: 2Gi
storageClassName: fast-disks
  metad:
     resources:
       requests:
         cpu: "500m"
          memory: "500Mi"
       limits:
cpu: "1"
     memory: "1Gi"
replicas: 1
     image: vesoft/nebula-metad
     version: v3.4.1
     logVolumeClaim:
       resources:
         requests:
            storage: 2Gi
     storageClassName: fast-disks
dataVolumeClaim:
       resources:
         requests:
           storage: 2Gi
       storageClassName: fast-disks
  storaged:
resources:
       requests:
cpu: "500m"
          memory: "500Mi"
       limits:
cpu: "1"
     memory: "1Gi"
replicas: 1
     image: vesoft/nebula-storaged
     version: v3.4.1
logVolumeClaim:
       resources:
         requests:
           storage: 2Gi
       storageClassName: fast-disks
     dataVolumeClaims: // You can mount multiple disks starting from NebulaGraph Operator 1.3.0. - resources:
          requests:
            storage: 2Gi
       storageClassName: fast-disks
     - resources: requests:
            storage: 2Gi
     storageClassName: fast-disks enableAutoBalance: true
  reference:
    name: statefulsets.apps
  schedulerName: default-scheduler
nodeSelector:
  nebula: cloud
imagePullPolicy: Always
unsatisfiableAction: ScheduleAnyway
```

 $_{\bullet}$  The file contents for a Nebula Graph Enterprise cluster are as follows:

```
# Contact our sales team to get a complete NebulaGraph Enterprise Edition cluster YAML example.
apiVersion: apps.nebula-graph.io/v1alpha1 kind: NebulaCluster metadata:
   annotations:
   nebula-graph.io/owner: test
name: nebula
   graphd:
      readinessProbe:
         failureThreshold: 3
httpGet:
         httpGet:
path: /status
port: 19669
scheme: HTTP
initialDelaySeconds: 40
periodSeconds: 10
successThreshold: 1
       timeoutSeconds: 10 image: reg.vesoft-inc.com/vesoft-ent/nebula-graphd logVolumeClaim:
         resources:
requests:
       storage: 2Gi
storageClassName: fast-disks
replicas: 1
      resources:
Limits:
cpu: "1"
memory: 1Gi
requests:
   requests:
cpu: 500m
memory: 500Mi
version: v3.4.1
imagePullPolicy: Always
imagePullSecrets:
     - name: vesoft
   metad:
     license:
          secretName: nebula-license
          licenseKey: nebula.license
```

The parameters in the file are described as follows:

Parameter	Default value	Description
metadata.name	-	The name of the created NebulaGraph cluster.
spec.graphd.replicas	I	The numeric value of replicas of the Graphd service.
spec.graphd.images	vesoft/nebula-graphd	The container image of the Graphd service.
spec.graphd.version	v3.4.1	The version of the Graphd service.
spec.graphd.service	-	The Service configurations for the Graphd service.
spec.graphd.log Volume Claim.storage Class Name	-	The log disk storage configurations for the Graphd service.
spec.metad.replicas	I	The numeric value of replicas of the Metad service.
spec.metad.images	vesoft/nebula-metad	The container image of the Metad service.
spec.metad.version	v3.4.1	The version of the Metad service.
spec.metad.data Volume Claim.storage Class Name	-	The data disk storage configurations for the Metad service.
spec.metad. log Volume Claim. storage Class Name	-	The log disk storage configurations for the Metad service.
spec.storaged.replicas	3	The numeric value of replicas of the Storaged service.
spec.storaged.images	vesoft/nebula-storaged	The container image of the Storaged service.
spec.storaged.version	v3.4.1	The version of the Storaged service.
spec.storaged.data Volume Claims.resources.requests.storage	-	Data disk storage size for the Storaged service. You can specify multiple data disks to store data. When multiple disks are specified, the storage path is /usr/local/nebula/data1, /usr/local/nebula/data2, etc.
spec.storaged.data Volume Claims.resources.storage Class Name	-	The data disk storage configurations for Storaged. If not specified, the global storage parameter is applied.
spec.storaged.log Volume Claim.storage Class Name	-	The log disk storage configurations for the Storaged service.
spec.storaged.enableAutoBalance	true	Whether to balance data automatically.
spec.reference.name	-	The name of the dependent controller.
spec.schedulerName	-	The scheduler name.
spec.imagePullPolicy	The image policy to pull the NebulaGraph image. For details, see Image pull policy.	The image pull policy in Kubernetes.
spec.metad.license	-	The configuration of the license for creating a NebulaGraph Enterprise Edition

# terpriseonly

Make sure that you have access to NebulaGraph Enterprise Edition images before pulling the image. For details, contact our sales team (inqury@vesoft.com)

2. Configure the license for the Enterprise Edition cluster.



- This step is required only for creating a NebulaGraph Grpah Enterprise Edition cluster.
- Ignore this step if you are creating a NebulaGraph Community Edition cluster.

```
kubectl create secret generic nebula-license --from-file=nebula.license
```

To check the details of the license, run the following command:

```
kubectl get secrets nebula-license -o yaml
```

3. Create a NebulaGraph cluster.

```
kubectl create -f apps_vlalpha1_nebulacluster.yaml
```

### Output:

nebulacluster.apps.nebula-graph.io/nebula created

4. Check the status of the NebulaGraph cluster.

```
kubectl get nebulaclusters.apps.nebula-graph.io nebula
```

### Output:

```
NAME GRAPHD-DESIRED GRAPHD-READY METAD-DESIRED METAD-READY STORAGED-DESIRED STORAGED-READY AGE nebula 1 1 1 1 3 3 86s
```

# Scaling clusters

- The cluster scaling feature is for NebulaGraph Enterprise Edition only.
- Scaling a NebulaGraph cluster for Enterprise Edition is supported only with NebulaGraph Operator version 1.1.0 or later.

You can modify the value of replicas in apps\_vlalphal\_nebulactuster.yaml to scale a NebulaGraph cluster.

#### SCALE OUT CLUSTERS

The following shows how to scale out a NebulaGraph cluster by changing the number of Storage services to 5:

1. Change the value of the storaged.replicas from 3 to 5 in apps\_vlalphal\_nebulacluster.yaml.

```
storaged:
  resources
    requests
      cpu: "500m"
        emory: "500Mi'
    limits:
cpu: "1"
  memory: "1Gi"
replicas: 5
  image: vesoft/nebula-storaged
  version: v3.4.1
  dataVolumeClaims:
   - resources:
      requests:
        storage: 2Gi
    storageClassName: fast-disks
    resources:
      requests:
        storage: 2Gi
    storageClassName: fast-disks
  logVolumeClaim:
     resources:
      requests:
        storage: 2Gi
    storageClassName: fast-disks
reference:
  name: statefulsets.apps
version: v1
schedulerName: default-scheduler
```

2. Run the following command to update the NebulaGraph cluster CR.

```
kubectl apply -f apps_v1alpha1_nebulacluster.yaml
```

3. Check the number of Storage services.

```
kubectl get pods -l app.kubernetes.io/cluster=nebula
```

# Output:

```
STATUS
                                     RESTARTS
NAME
                   READY
                                                AGE
nebula-graphd-0
                            Running
nebula-metad-0
                   1/1
                           Runn\,i\,ng
                                                 2m
nebula-storaged-0
                   1/1
                           Running
                                                 2m
nebula-storaged-1
                            Running
nebula-storaged-2
                   1/1
                           Running
                                                 2m
nebula-storaged-3
                   1/1
                            Running
nebula-storaged-4
                            Running
```

As you can see above, the number of Storage services is scaled up to 5.

# SCALE IN CLUSTERS

The principle of scaling in a cluster is the same as scaling out a cluster. You scale in a cluster if the numeric value of the repticas in apps\_vlatphal\_nebulactuster.yaml is changed smaller than the current number. For more information, see the **Scale out clusters** section above.



NebulaGraph Operator currently only supports scaling Graph and Storage services and does not support scale Meta services.

### **Delete clusters**

Run the following command to delete a NebulaGraph cluster with Kubectl:

```
kubectl delete -f apps_v1alpha1_nebulacluster.yaml
```

# What's next

# Connect to NebulaGraph databases

# 19.4.2 Deploy NebulaGraph clusters with Helm



The 1.x version NebulaGraph Operator is not compatible with NebulaGraph of version below v3.x.

### Prerequisite

- Install NebulaGraph Operator
- You have prepared the license file for NebulaGraph Enterprise Edition clusters.



The license file is required only when creating a NebulaGraph Enterprise Edition cluster.

#### Create clusters

1. Add the NebulaGraph Operator Helm repository.

```
helm repo add nebula-operator https://vesoft-inc.github.io/nebula-operator/charts
```

2. Update information of available charts locally from chart repositories.

helm repo update

3. Set environment variables to your desired values.

```
export NEBULA_CLUSTER_NAME=nebula  # The desired NebulaGraph cluster name.
export NEBULA_CLUSTER_NAMESPACE=nebula  # The desired namespace where your NebulaGraph cluster locates.
export STORAGE_CLASS_NAME=fast-disks  # The desired StorageClass name in your NebulaGraph cluster.
```

4. Create a namespace for your NebulaGraph cluster (If you have created one, skip this step).

```
{\tt kubectl\ create\ namespace\ "$\{NEBULA\_CLUSTER\_NAMESPACE\}"}
```

5. Configure the license for the Enterprise Edition cluster.



- $\bullet$  This step is required only for creating a Nebula Graph Grpah Enterprise Edition cluster.
- Ignore this step if you are creating a NebulaGraph Community Edition cluster.

```
kubectl create secret generic nebula-license --from-file=nebula.license
```

To check the details of the license, run the following command:

```
kubectl get secrets nebula-license -o yaml
```

6. Apply the variables to the Helm chart to create a NebulaGraph cluster.

```
helm install "${NEBULA_CLUSTER_NAME}" nebula-operator/nebula-cluster \
--namespace="${NEBULA_CLUSTER_NAME}" \
--set nameOverride=${NEBULA_CLUSTER_NAME} \
--set nebula.storageClassName="${STORAGE_CLASS_NAME}"\
# Specify the version of the NebulaGraph cluster.
--set nebula.version=v3.4.1 \
# Specify the version of the nebula-cluster chart. If not specified, the latest version of the chart is installed by default.
--version=1.4.2
```

- 944/1069 - 2023 Vesoft Inc.

# Sterpriseonly

You must add --set nebula.metad.license.secretName=nebula-license and --set nebula.metad.license.licenseKey=nebula.license when creating a NebulaGraph Enterprise Edition cluster.

7. Check the status of the NebulaGraph cluster you created.

```
kubectl -n "${NEBULA_CLUSTER_NAMESPACE}" get pod -l "app.kubernetes.io/cluster=${NEBULA_CLUSTER_NAME}"
```

# Output:

```
STATUS
                    READY
                                       RESTARTS
                                                   AGE
nebula-graphd-0
                                                   5m34s
                     1/1
                             Running
nebula-graphd-1
                             Running
                                                   5m34s
                     1/1
nebula-metad-0
                             Running
                                                   5m34s
nebula-metad-1
                     1/1
                             Running
                                                   5m34s
                    1/1
1/1
nebula-metad-2
                             Running
                                                   5m34s
nebula-storaged-0
                             Running
                                                   5m34s
nebula-storaged-1
                             Running
                                                   5m34s
nebula-storaged-2
                             Running
                                                   5m34s
```

### Scaling clusters

- The cluster scaling feature is for NebulaGraph Enterprise Edition only.
- Scaling a NebulaGraph cluster for Enterprise Edition is supported only with NebulaGraph Operator version 1.1.0 or later.

You can scale a NebulaGraph cluster by defining the value of the repticas corresponding to the different services in the cluster.

For example, run the following command to scale out a NebulaGraph cluster by changing the number of Storage services from 2 (the original value) to 5:

```
helm upgrade "${NEBULA_CLUSTER_NAME}" nebula-operator/nebula-cluster \
--namespace="${NEBULA_CLUSTER_NAMESPACE}" \
--set nameOverride=${NEBULA_CLUSTER_NAME} \
--set nebula.storageClassName="${$TORAGE_CLASS_NAME}" \
--set nebula.storaged.replicas=5
```

Similarly, you can scale in a NebulaGraph cluster by setting the value of the replicas corresponding to the different services in the cluster smaller than the original value.



NebulaGraph Operator currently only supports scaling Graph and Storage services and does not support scale Meta services.

You can click on nebula-cluster/values.yaml to see more configurable parameters of the nebula-cluster chart. For more information about the descriptions of configurable parameters, see **Configuration parameters of the nebula-cluster Helm chart** below.

### **Delete clusters**

Run the following command to delete a NebulaGraph cluster with Helm:

```
helm uninstall "${NEBULA_CLUSTER_NAME}" --namespace="${NEBULA_CLUSTER_NAMESPACE}"
```

Or use variable values to delete a NebulaGraph cluster with Helm:

```
helm uninstall nebula --namespace=nebula
```

# What's next

Connect to NebulaGraph Databases

Configuration parameters of the nebula-cluster Helm chart

Parameter	Default value	Description  Replaces the name of the chart in the Chart.yaml file.	
nameOverride	nil		
nebula.version	v3.4.1	The version of NebulaGraph.	
nebula.imagePullPolicy	IfNotPresent	The NebulaGraph image pull policy. For details, see Image pull policy.	
nebula.storageClassName	nil	The StorageClass name. StorageClass is the default persistent volume type.	
nebula.schedulerName	default-scheduler	The scheduler name of a NebulaGraph cluster.	
nebula.reference	{"name": "statefulsets.apps", "version": "v1"}	The workload referenced for a NebulaGraph cluster.	
nebula.graphd.image	vesoft/nebula-graphd  The image name for a Grap service. Uses the value of nebula.version as its version.		
nebula.graphd.replicas	2	The number of the Graphd service.	
nebula.graphd.env	0	The environment variables for the Graphd service.	
nebula.graphd.resources	{"resources":{"requests": {"cpu":"500m","memory":"500Mi"},"limits": {"cpu":"1","memory":"16i"}}}	The resource configurations for the Graphd service.	
nebula.graphd.logStorage	500Mi	The log disk storage capacity for the Graphd service.	
nebula.graphd.podLabels	0	Labels for the Graphd pod in a NebulaGraph cluster.	
nebula.graphd.podAnnotations	0	Pod annotations for the Graphd pod in a NebulaGraph cluster.	
nebula.graphd.nodeSelector	Labels for the Graphd pod to scheduled to the specified no		
nebula.graphd.tolerations	0	Tolerations for the Graphd pod.	
nebula.graphd.affinity	0	Affinity for the Graphd pod.	
nebula.graphd.readinessProbe	0	ReadinessProbe for the Graphd pod.	
nebula.graphd.sidecarContainers	Sidecar containers for the Grap pod.		
nebula.graphd.sidecarVolumes	0	Sidecar volumes for the Graphd pod.	
nebula.metad.image	vesoft/nebula-metad	The image name for a Metad service.  Uses the value of nebula.version as its version.	
nebula.metad.replicas	3	The number of the Metad service.	
nebula.metad.env	[0]	The environment variables for the Metad service.	
nebula.metad.resources	{"resources":{"requests": {"cpu":"500m","memory":"500Mi"},"Limits": {"cpu":"1","memory":"16i"}}}	The resource configurations for the Metad service.	

- 947/1069 - 2023 Vesoft Inc.

Parameter	Default value	Description
nebula.metad.logStorage	500Mi	The log disk capacity for the Metad service.
nebula.metad.dataStorage	16i	The data disk capacity for the Metad service.
nebula.metad.license	8	The license configurations for creating a NebulaGraph Enterprise Edition cluster.
nebula.metad.podLabels	8	Labels for the Metad pod in a NebulaGraph cluster.
nebula.metad.podAnnotations	0	Pod annotations for the Metad pod in a NebulaGraph cluster.
nebula.metad.nodeSelector	0	Labels for the Metad pod to be scheduled to the specified node.
nebula.metad.tolerations	0	Tolerations for the Metad pod.
nebula.metad.affinity	8	Affinity for the Metad pod.
nebula.metad.readinessProbe	0	ReadinessProbe for the Metad pod.
nebula.metad.sidecarContainers	0	Sidecar containers for the Metad pod.
nebula.metad.sidecarVolumes	0	Sidecar volumes for the Metad pod.
nebula.storaged.image	vesoft/nebula-storaged	The image name for a Storaged service. Uses the value of nebula.version as its version.
nebula.storaged.replicas	3	The number of Storaged services.
nebula.storaged.env	0	The environment variables for Storaged services.
nebula.storaged.resources	{"resources":{"requests": {"cpu":"500m","memory":"500Mi"},"limits": {"cpu":"1","memory":"1Gi"}}}	The resource configurations for Storagedss services.
nebula.storaged.logStorage	500Mi	The log disk capacity for the Metad service.
nebula.storaged.dataStorage	16i	The data disk capacity for the Metad service.
nebula.storaged.podLabels	{}	Labels for the Metad pod in a NebulaGraph cluster.
nebula.storaged.podAnnotations	{}	Pod annotations for the Metad pod in a NebulaGraph cluster.
nebula.storaged.nodeSelector	{}	Labels for the Metad pod to be scheduled to the specified node.
nebula.storaged.tolerations	0	Tolerations for the Metad pod.
nebula.storaged.affinity	0	Affinity for the Metad pod.
nebula.storaged.readinessProbe	0	ReadinessProbe for the Metad pod.

- 948/1069 - 2023 Vesoft Inc.

Parameter	Default value	Description
nebula.storaged.sidecarVolumes	0	Sidecar volumes for the Metad pod.
imagePullSecrets	0	The Secret to pull the NebulaGraph cluster image.

# 19.5 Connect to NebulaGraph databases with Nebular Operator

After creating a NebulaGraph cluster with NebulaGraph Operator on Kubernetes, you can connect to NebulaGraph databases from within the cluster and outside the cluster.

# 19.5.1 Prerequisites

Create a NebulaGraph cluster with NebulaGraph Operator on Kubernetes. For more information, see Deploy NebulaGraph clusters with Kubectl or Deploy NebulaGraph clusters with Helm.

## 19.5.2 Connect to NebulaGraph databases from outside a NebulaGraph cluster via NodePort

You can create a NodePort type Service to access internal cluster services from outside the cluster using any node IP and the exposed node port. You can also utilize load balancing services provided by cloud vendors (such as Azure, AWS, etc.) by setting the Service type to LoadBalancer. This allows external access to internal cluster services through the public IP and port of the load balancer provided by the cloud vendor.

The Service of type NodePort forwards the front-end requests via the label selector spec.selector to Graphd pods with labels app.kubernetes.io/cluster: <cluster-name> and app.kubernetes.io/component: graphd.

After creating a NebulaGraph cluster based on the example template, where <code>spec.graphd.service.type=NodePort</code>, the NebulaGraph Operator will automatically create a NodePort type Service named <code><cluster-name>-graphd-svc</code> in the same namespace. You can directly connect to the NebulaGraph database through any node IP and the exposed node port (see step 4 below). You can also create a custom Service according to your needs.

- 950/1069 - 2023 Vesoft Inc.

### Steps:

1. Create a YAML file named graphd-nodeport-service.yaml . The file contents are as follows:

```
apiVersion: v1
metadata:
 labels:
   app.kubernetes.io/cluster: nebula
   app.kubernetes.io/component: graphd app.kubernetes.io/managed-by: nebula-operator
   app.kubernetes.io/name: nebula-graph
 name: nebula-graphd-svc-nodeport
 namespace: default
 externalTrafficPolicy: Local
   name: thrift
   port: 9669
   protocol: TCP
   targetPort: 9669
   name: http
   port: 19669
   protocol: TCP
    targetPort: 19669
  selector
   app.kubernetes.io/cluster: nebula
    app.kubernetes.io/component: graphd
   app.kubernetes.io/managed-by: nebula-operator
   app.kubernetes.io/name: nebula-graph
  type: NodePort # Set the type to NodePort
```

- NebulaGraph uses port 9669 by default. 19669 is the HTTP port of the Graph service in a NebulaGraph cluster.
- The value of targetPort is the port mapped to the database Pods, which can be customized.
- 2. Run the following command to create a NodePort Service.

```
kubectl create -f graphd-nodeport-service.yaml
```

3. Check the port mapped on all of your cluster nodes.

```
kubectl get services -l app.kubernetes.io/cluster=<nebula> # <nebula> is the name of your NebulaGraph cluster.
```

# Output:

```
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE nebula-graphd-svc-nodeport NodePort 10.107.153.129 <none> 9669:32236/TCP,19669:31674/TCP,19670:31057/TCP 24h ...
```

As you see, the mapped port of NebulaGraph databases on all cluster nodes is 32236.

4. Connect to NebulaGraph databases with your node IP and the node port above.

```
kubectl run -ti --image vesoft/nebula-console:v3.4.0 --restart=Never -- <nebula_console_name> -addr <node_ip> -port <node_port> -u <username> -p  -password>
```

# For example:

```
kubectl run -ti --image vesoft/nebula-console:v3.4.0 --restart=Never -- nebula-console -addr 192.168.8.24 -port 32236 -u root -p vesoft If you don't see a command prompt, try pressing enter.

(root@nebula) [(none)]>
```

- -- image: The image for the tool NebulaGraph Console used to connect to NebulaGraph databases.
- <nebula-console> : The custom Pod name. The above example uses nebula-console .
- -addr: The IP of any node in a NebulaGraph cluster. The above example uses 192.168.8.24.
- -port: The mapped port of NebulaGraph databases on all cluster nodes. The above example uses 32236.
- -u: The username of your NebulaGraph account. Before enabling authentication, you can use any existing username. The default username is root.
- -p: The password of your NebulaGraph account. Before enabling authentication, you can use any characters as the password.

# 19.5.3 Connect to NebulaGraph databases from within a NebulaGraph cluster

You can also create a ClusterIP type Service to provide an access point to the NebulaGraph database for other Pods within the cluster. By using the Service's IP and the Graph service's port number (9669), you can connect to the NebulaGraph database. For more information, see ClusterIP.

1. Create a file named graphd-clusterip-service.yaml. The file contents are as follows:

```
apiVersion: v1
kind: Service
metadata:
  labels
    app.kubernetes.io/cluster: nebula
    app.kubernetes.io/component: graphd
app.kubernetes.io/managed-by: nebula-operator
    app.kubernetes.io/name: nebula-graph
  name: nebula-graphd-svc
  namespace: default
spec:
  externalTrafficPolicy: Local
   name: thrift
    port: 9669
    protocol: TCP
    targetPort: 9669
    port: 19669
    protocol: TCP
    targetPort: 19669
  selector
    app.kubernetes.io/cluster: nebula
    app.kubernetes.io/component: graphd
    app.kubernetes.io/managed-by: nebula-operator
  type: ClusterIP # Set the type to ClusterIP.
```

- NebulaGraph uses port 9669 by default. 19669 is the HTTP port of the Graph service in a NebulaGraph cluster.
- targetPort is the port mapped to the database Pods, which can be customized.
- 2. Create a ClusterIP Service.

```
kubectl create -f graphd-clusterip-service.yaml
```

3. Check the IP of the Service:

4. Run the following command to connect to the NebulaGraph database using the IP of the <cluster-name>-graphd-svc Service above:

```
kubectl run -ti --image vesoft/nebula-console:v3.4.0 --restart=Never -- <nebula_console_name> -addr <cluster_ip> -port <service_port> -u <username> -p <password>
```

### For example:

```
kubectl run -ti --image vesoft/nebula-console:v3.4.0 --restart=Never -- nebula-console -addr 10.98.213.34 -port 9669 -u root -p vesoft

- `--image`: The image for the tool NebulaGraph Console used to connect to NebulaGraph databases.
- `-enebula-console>`: The custom Pod name.
- `-addr': The IP of the `ClusterIP` Service, used to connect to Graphd services.
- `-port`: The port to connect to Graphd services, the default port of which is '9669'.
- `-u': The username of your NebulaGraph account. Before enabling authentication, you can use any existing username. The default username is root.
- `-p`: The password of your NebulaGraph account. Before enabling authentication, you can use any characters as the password.

A successful connection to the database is indicated if the following is returned:

```bash
If you don't see a command prompt, try pressing enter.

(root@nebula) [(none)]>
```

You can also connect to NebulaGraph databases with **Fully Qualified Domain Name (FQDN)**. The domain format is <cluster-name>-graphd.<cluster-namespace>.svc.<CLUSTER\_DOMAIN> . The default value of CLUSTER\_DOMAIN is cluster.local .

```
kubectl run -ti --image vesoft/nebula-console:v3.4.0 --restart=Never -- <nebula_console_name> -addr <cluster_name>-graphd-svc.default.svc.cluster.local -port <service_port> -u <username> - p <password>
```

service\_port is the port to connect to Graphd services, the default port of which is 9669.

# 19.5.4 Connect to NebulaGraph databases from outside a NebulaGraph cluster via Ingress

When dealing with multiple pods in a cluster, managing services for each pod separately is not a good practice. Ingress is a Kubernetes resource that provides a unified entry point for accessing multiple services. Ingress can be used to expose multiple services under a single IP address.

Nginx Ingress is an implementation of Kubernetes Ingress. Nginx Ingress watches the Ingress resource of a Kubernetes cluster and generates the Ingress rules into Nginx configurations that enable Nginx to forward 7 layers of traffic.

You can use Nginx Ingress to connect to a NebulaGraph cluster from outside the cluster using a combination of the host network and DaemonSet pattern.

As the host network is used, the Nginx Ingress pod cannot be scheduled to the same node. To avoid listening port conflicts, some nodes can be selected and labeled as edge nodes in advance, which are specially used for the Nginx Ingress deployment. Nginx Ingress is then deployed on these nodes in a DaemonSet mode.

Ingress does not support TCP or UDP services. For this reason, the nginx-ingress-controller pod uses the flags --tcp-services-configmap and --udp-services-configmap to point to an existing ConfigMap where the key refers to the external port to be used and the value refers to the format of the service to be exposed. The format of the value is <namespace/service\_name>:<service\_port>.

For example, the configurations of the ConfigMap named as tcp-services is as follows:

```
apiVersion: v1
kind: ConfigMap
metadata:
name: tcp-services
namespace: nginx-ingress
data:
# update
9769: "default/nebula-graphd-svc:9669"
```

Steps are as follows.

1. Create a file named nginx-ingress-daemonset-hostnetwork.yaml.

Click on nginx-ingress-daemonset-hostnetwork, yaml to view the complete content of the example YAML file.



The resource objects in the YAML file above use the namespace nginx-ingress. You can run kubectl create namespace nginx-ingress to create this namespace, or you can customize the namespace.

2. Label a node where the DaemonSet named nginx-ingress-controller in the above YAML file (The node used in this example is named worker2 with an IP of 192.168.8.160) runs.

```
kubectl label node worker2 nginx-ingress=true
```

3. Run the following command to enable Nginx Ingress in the cluster you created.

```
kubectl create -f nginx-ingress-daemonset-hostnetwork.yaml
```

#### Output:

configmap/nginx-ingress-controller created configmap/tcp-services created serviceaccount/nginx-ingress created serviceaccount/nginx-ingress-backend created clusterrole.rbac.authorization.k8s.io/nginx-ingress created clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress created role.rbac.authorization.k8s.io/nginx-ingress created rolebinding.rbac.authorization.k8s.io/nginx-ingress created rolebinding.rbac.authorization.k8s.io/nginx-ingress created service/nginx-ingress-controller-metrics created service/nginx-ingress-default-backend created service/nginx-ingress-proxy-tcp created daemonset.apps/nginx-ingress-controller created

Since the network type that is configured in Nginx Ingress is hostNetwork, after successfully deploying Nginx Ingress, with the IP (192.168.8.160) of the node where Nginx Ingress is deployed and with the external port (9769) you define, you can access NebulaGraph.

4. Use the IP address and the port configured in the preceding steps. You can connect to NebulaGraph with NebulaGraph Console.

```
kubectl run -ti --image vesoft/nebula-console:v3.4.0 --restart=Never -- <nebula_console_name> -addr <host_ip> -port <external_port> -u <username> -p <password>
```

# Output:

```
kubectl run -ti --image vesoft/nebula-console:v3.4.0 --restart=Never -- nebula-console -addr 192.168.8.160 -port 9769 -u root -p vesoft
```

- · --image: The image for the tool NebulaGraph Console used to connect to NebulaGraph databases.
- ullet <nebula-console> The custom Pod name. The above example uses nebula-console .
- -addr: The IP of the node where Nginx Ingress is deployed. The above example uses 192.168.8.160.
- ullet -port : The port used for external network access. The above example uses  ${\tt 9769}$  .
- • The username of your NebulaGraph account. Before enabling authentication, you can use any existing username. The default username is root.
- -p: The password of your NebulaGraph account. Before enabling authentication, you can use any characters as the password.

A successful connection to the database is indicated if the following is returned:

```
If you don't see a command prompt, try pressing enter.

(root@nebula) [(none)]>
```

Last update: February 19, 2024

- 954/1069 - 2023 Vesoft Inc.

# 19.6 Configure clusters

### 19.6.1 Customize configuration parameters for a NebulaGraph cluster

Meta, Storage, and Graph services in a NebulaGraph Cluster have their configurations, which are defined as <code>config</code> in the YAML file of the CR instance (NebulaGraph cluster) you created. The settings in <code>config</code> are mapped and loaded into the ConfigMap of the corresponding service in Kubernetes.



It is not available to customize configuration parameters for NebulaGraph Clusters deployed with Helm.

The structure of config is as follows.

```
Config map[string]string `json:"config,omitempty"`
```

### Prerequisites

You have created a NebulaGraph cluster. For how to create a cluster with Kubectl, see Create a cluster with Kubectl.

### Steps

The following example uses a cluster named nebula and the cluster's configuration file named nebula\_cluster.yaml to show how to set config for the Graph service in a NebulaGraph cluster.

1. Run the following command to access the edit page of the nebula cluster.

```
kubectl edit nebulaclusters.apps.nebula-graph.io nebula
```

2. Add enable\_authorize and auth\_type under spec.graphd.config.

```
apiVersion: apps.nebula-graph.io/vlalphal
kind: NebulaCluster
metadata:
 name: nebula
  namespace: default
spec:
  graphd:
    resources:
      requests:
        cpu: "500m
         memory: "500Mi"
      limits:
cpu: "1"
    memory: "1Gi"
replicas: 1
    image: vesoft/nebula-graphd
    version: v3.4.1
    storageClaim:
      resources:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
    config: // Custom configuration parameters for the Graph service in a cluster.
       "enable_authorize": "true"
      "auth_type": "password"
```

To add the config for the Meta and Storage services, add spec.metad.config and spec.storaged.config respectively.

3. Run kubectl apply -f nebula\_cluster.yaml to push your configuration changes to the cluster.

After customizing the parameters <code>enable\_authorize</code> and <code>auth\_type</code>, the configurations in the corresponding ConfigMap (<code>nebula-graphd</code>) of the Graph service will be overwritten.

### Modify cluster configurations online

Cluster configurations are modified online by calling the HTTP interface, without the need to restart the cluster Pod.

It should be noted that only when all configuration items in <code>config</code> are the parameters that can be dynamically modified at runtime, can the operation of online modifications be triggered. If the configuration items in <code>config</code> contain parameters that cannot be dynamically modified, then the cluster configuration will be updated by restarting the Pod.

For information about the parameters that can be dynamically modified for each service, see the parameter table column of **Whether supports runtime dynamic modifications** in Meta service configuration parameters, Storage service configuration parameters, and Graph service configuration parameters, respectively.

### Learn more

For more information about the configuration parameters of Meta, Storage, and Graph services, see Meta service configuration parameters, Storage service configuration parameters, and Graph service configuration parameters.

# 19.6.2 Reclaim PVs

NebulaGraph Operator uses PVs (Persistent Volumes) and PVCs (Persistent Volume Claims) to store persistent data. If you accidentally deletes a NebulaGraph cluster, by default, PV and PVC objects and the relevant data will be retained to ensure data security.

You can also define the automatic deletion of PVCs to release data by setting the parameter <code>spec.enablePVReclaim</code> to true in the configuration file of the cluster instance. As for whether PV will be deleted automatically after PVC is deleted, you need to customize the PV reclaim policy. See <code>reclaimPolicy</code> in <code>StorageClass</code> and <code>PV Reclaiming</code> for details.

# Prerequisites

You have created a cluster. For how to create a cluster with Kubectl, see Create a cluster with Kubectl.

# Steps

The following example uses a cluster named nebula and the cluster's configuration file named nebula\_cluster.yaml to show how to set enablePVReclaim:

 $_{\mbox{\scriptsize 1.}}$  Run the following command to access the edit page of the  $\mbox{\scriptsize nebula}$  cluster.

```
kubectl edit nebulaclusters.apps.nebula-graph.io nebula
```

2. Add enablePVReclaim and set its value to true under spec.

```
apiVersion: apps.nebula-graph.io/v1alpha1
kind: NebulaCluster
metadata:
  name: nebula
spec:
  enablePVReclaim: true //Set its value to true.
    image: vesoft/nebula-graphd
logVolumeClaim:
      resources:
        requests:
           storage: 2Gi
    storageClassName: fast-disks
replicas: 1
       limits:
cpu: "1"
         memory: 1Gi
       requests
        cpu: 500m
  memory: 500Mi
version: v3.4.1
imagePullPolicy: IfNotPresent
  metad:
    dataVolumeClaim:
      resources:
         requests:
    storage: 2Gi
storageClassName: fast-disks
image: vesoft/nebula-metad
    logVolumeClaim:
       resources:
requests:
    storage: 2Gi
storageClassName: fast-disks
replicas: 1
    resources:
      limits:
cpu: "1"
         memory: 1Gi
       requests:
         cpu: 500m
    memory: 500Mi
version: v3.4.1
  nodeSelector
    nebula: cloud
  reference:
    name: statefulsets.apps
version: v1
  schedulerName: default-scheduler
  storaged:
dataVolumeClaims:
     - resources:
        requests:
storage: 2Gi
       storageClassName: fast-disks
    - resources:
       storage: 2Gi
storageClassName: fast-disks
    image: vesoft/nebula-storaged
    logVolumeClaim:
       resources:
        requests:
       storage: 2Gi
storageClassName: fast-disks
    replicas: 3
    resources:
         cpu: "1"
memory: 1Gi
       requests
         cpu: 500m
memory: 500Mi
    version: v3.4.1
```

 $3. \ Run \ \ \text{kubectl apply -f nebula\_cluster.yaml} \ \ to \ push \ your \ configuration \ changes \ to \ the \ cluster.$ 

# 19.6.3 Balance storage data after scaling out



This feature is for NebulaGraph Enterprise Edition only.

After the Storage service is scaled out, you can decide whether to balance the data in the Storage service.

The scaling out of the NebulaGraph's Storage service is divided into two stages. In the first stage, the status of all pods is changed to Ready. In the second stage, the commands of BALANCE DATA and BALANCE LEADER are executed to balance data. These two stages decouple the scaling out process of the controller replica from the balancing data process, so that you can choose to perform the data balancing operation during low traffic period. The decoupling of the scaling out process from the balancing process can effectively reduce the impact on online services during data migration.

You can define whether to balance data automatically or not with the parameter enableAutoBalance in the configuration file of the CR instance of the cluster you created.

# Prerequisites

You have created a NebulaGraph cluster. For how to create a cluster with Kubectl, see Create a cluster with Kubectl.

- 960/1069 - 2023 Vesoft Inc.

# Steps

The following example uses a cluster named  $\$ nebula and the cluster's configuration file named  $\$ nebula\_cluster.yaml to show how to set  $\$ enableAutoBalance .

 $_{\mbox{\scriptsize 1.}}$  Run the following command to access the edit page of the  $_{\mbox{\scriptsize nebula}}$  cluster.

kubectl edit nebulaclusters.apps.nebula-graph.io nebula

2. Add enableAutoBalance and set its value to true under spec.storaged.

```
apiVersion: apps.nebula-graph.io/v1alpha1
kind: NebulaCluster
metadata:
  name: nebula
spec:
  graphd:
     image: vesoft/nebula-graphd
     logVolumeClaim:
      resources:
        requests:
          storage: 2Gi
       storageClassName: fast-disks
     replicas: 1
     resources:
      limits:
cpu: "1"
memory: 1Gi
      requests:
cpu: 500m
memory: 500Mi
  version: v3.4.1
imagePullPolicy: IfNotPresent
     dataVolumeClaim:
      resources:
        requests:
       storage: 2Gi
storageClassName: fast-disks
     image: vesoft/nebula-metad
logVolumeClaim:
      resources:
        requests:
          storage: 2Gi
       storageClassName: fast-disks
    replicas: 1 resources:
      limits:
cpu: "1"
memory: 1Gi
       requests:
cpu: 500m
memory: 500Mi
     version: v3.4.1
  nodeSelector
     nebula: cloud
  reference:
    name: statefulsets.apps
  version: v1
schedulerName: default-scheduler
    enableAutoBalance: true //Set its value to true which means storage data will be balanced after the Storage service is scaled out. dataVolumeClaims:
     - resources:
        requests:
storage: 2Gi
       storageClassName: fast-disks
     - resources:
      storage: 2Gi
storageClassName: fast-disks
     image: vesoft/nebula-storaged
     logVolumeClaim:
      resources:
        requests:
          storage: 2Gi
       storageClassName: fast-disks
     replicas: 3
     resources:
         cpu: "1"
memory: 1Gi
       requests
         cpu: 500m
memory: 500Mi
```

# version: v3.4.1

- When the value of enableAutoBalance is set to true, the Storage data will be automatically balanced after the Storage service is scaled out.
- When the value of enableAutoBalance is set to false, the Storage data will not be automatically balanced after the Storage service is scaled out.
- When the enableAutoBalance parameter is not set, the system will not automatically balance Storage data by default after the Storage service is scaled out.
- 3. Run kubectl apply -f nebula\_cluster.yaml to push your configuration changes to the cluster.

### 19.6.4 Manage cluster logs

Running logs of NebulaGraph cluster services (graphd, metad, storaged) are generated and stored in the /usr/local/nebula/logs directory of each service container by default.

# View logs

To view the running logs of a NebulaGraph cluster, you can use the kubectl logs command.

For example, to view the running logs of the Storage service:

```
// View the name of the Storage service Pod, nebula-storaged-0.
$ kubectl get pods
                                  READY STATUS
  RESTARTS
  AGE
NAME
nebula-exporter-84b6974497-cr54d 1/1
  Running
  Running
nebula-graphd-0
                                  1/1
  22h
nebula-metad-0
                                  1/1
  Running
  45h
nebula-storaged-0
  Running
                                  1/1
// Enter the container storaged of the Storage service.
$ kubectl exec -it nebula-storaged-0 -c storaged -- /bin/bash
// View the running logs of the Storage service.
$ cd /usr/local/nebula/logs
```

#### Clean logs

Running logs generated by cluster services during runtime will occupy disk space. To avoid occupying too much disk space, the Operator uses a sidecar container to periodically clean and archive logs.

To facilitate log collection and management, each NebulaGraph service deploys a sidecar container responsible for collecting logs generated by the service container and sending them to the specified log disk. The sidecar container automatically cleans and archives logs using the logrotate tool.

In the YAML configuration file of the cluster instance, you can configure log rotation to automatically clean and archive logs through the <code>spec.logRotate</code> field. By default, the log rotation feature is turned off. Here is an example of enabling log rotation:

```
spec:
graphd:
    config:
        # Whether to include a timestamp in the log file name. "true" means yes, "false" means no.
        "timestamp_in_logfile_name": "false"
metad:
    config:
        "timestamp_in_logfile_name": "false"
storaged:
    config:
        "timestamp_in_logfile_name": "false"
logRotate: # Log rotation configuration
# The number of times a log file is rotated before being deleted. The default value is 5, and 0 means the log file will not be rotated before being deleted.
rotate: 5
# The log file is rotated only if it grows larger than the specified size. The default value is 200M.
size: "200M"
```

# **Collect logs**

If you don't want to mount additional log disks to back up log files, or if you want to collect logs and send them to a log center using services like fluent-bit, you can configure logs to be output to standard error. The Operator uses the glog tool to log to standard error output.



Currently, NebulaGraph Operator only collects standard error logs.  $^{\circ}$ 

In the YAML configuration file of the cluster instance, you can configure logging to standard error output in the config and env fields of each service.

```
spec:
  graphd:
     config:

# Whether to redirect standard error to a separate output file. The default value is false, which means it is not redirected.
       # The severity level of log content: INFO, WARNING, ERROR, and FATAL. The corresponding values are 0, 1, 2, and 3. stderrthreshold: "O"
    env:
- name: GLOG_logtostderr # Write log to standard error output instead of a separate file.
value: "1" # 1 represents writing to standard error output, and 0 represents writing to a file.
image: vesoft/nebula-graphd
replicas: 1
       requests:
         cpu: 500m
          memory: 500Mi
     service:
      externalTrafficPolicy: Local
     type: NodePort
version: v3.4.1
     config:
  redirect_stdout: "false"
     stderrthreshold: "O" dataVolumeClaim:
       resources:
         requests:
           storage: 1Gi
       storageClassName: ebs-sc
     env:
- name: GLOG_logtostderr
     value: "1"
image: vesoft/nebula-metad
```

# 19.7 Upgrade NebulaGraph clusters created with NebulaGraph Operator

This topic introduces how to upgrade a NebulaGraph cluster created with NebulaGraph Operator.

# Lacy version compatibility

The 1.x version NebulaGraph Operator is not compatible with NebulaGraph of version below v3.x.

### 19.7.1 Limits

- Only for upgrading the NebulaGraph clusters created with NebulaGraph Operator.
- Only support upgrading the NebulaGraph version from 3.0.0 to 3.4.0.
- For upgrading NebulaGraph Enterprise Edition clusters, contact us.

### 19.7.2 Upgrade a NebulaGraph cluster with Kubectl

# Prerequisites

You have created a NebulaGraph cluster with Kubectl. For details, see Create a NebulaGraph cluster with Kubectl.

The version of the NebulaGraph cluster to be upgraded in this topic is 3.0.0, and its YAML file name is apps\_vlalphal\_nebulacluster.yaml.

#### Steps

1. Check the image version of the services in the cluster.

```
kubectl get pods -l app.kubernetes.io/cluster=nebula -o jsonpath="{.items[*].spec.containers[*].image}" | tr -s '[[:space:]]' '\n' | sort | uniq -c
```

### Output:

```
1 vesoft/nebula-graphd:3.0.0
1 vesoft/nebula-metad:3.0.0
3 vesoft/nebula-storaged:3.0.0
```

 $2. \ Edit \ the \ apps\_vlalphal\_nebulacluster.yaml \ file \ by \ changing \ the \ values \ of \ all \ the \ version \ parameters \ from \ 3.0.0 \ to \ v3.4.1.$ 

The modified YAML file reads as follows:

```
apiVersion: apps.nebula-graph.io/vlalpha1
kind: NebulaCluster
metadata:
 name: nebula
  graphd:
    resources:
      requests:
cpu: "500m'
           emory: "500Mi"
      limits:
cpu: "1"
           emory: "1Gi"
    replicas: 1
image: vesoft/nebula-graphd
    version: v3.4.1 //Change the value from 3.0.0 to v3.4.1.
    service:
      type: NodePort
      externalTrafficPolicy: Local
    logVolumeClaim:
        requests:
          storage: 2Gi
      storageClassName: fast-disks
  metad:
    resources:
      requests:
cpu: "500m"
```

```
memory: "500Mi"
   limits:
      cpu: "1"
      memory: "1Gi"
  replicas: 1
  image: vesoft/nebula-metad
  version: v3.4.1 //Change the value from 3.0.0 to v3.4.1.
  dataVolumeClaim:
    resources:
      requests:
       storage: 2Gi
    {\it storageClassName:}\ fast-disks
  logVolumeClaim:
    resources:
      requests:
        storage: 2Gi
   storageClassName: fast-disks
storaged:
   requests:
cpu: "500m'
        .
emory: "500Mi"
   limits:
cpu: "1"
      memory: "1Gi"
  replicas: 3
  image: vesoft/nebula-storaged
  version: v3.4.1 //Change the value from 3.0.0 to v3.4.1.
  dataVolumeClaims:
      requests:
        storage: 2Gi
    storageClassName: fast-disks
  - resources:
      requests:
        storage: 2Gi
    storageClassName: fast-disks
  logVolumeClaim:
   resources:
     requests:
    storageClassName: fast-disks
reference:
  name: statefulsets.apps
version: v1
schedulerName: default-scheduler
imagePullPolicy: Always
```

3. Run the following command to apply the version update to the cluster CR.

```
kubectl apply -f apps_v1alpha1_nebulacluster.yaml
```

4. After waiting for about 2 minutes, run the following command to see if the image versions of the services in the cluster have been changed to v3.4.1.

```
kubectl get pods -l app.kubernetes.io/cluster=nebula -o jsonpath="{.items[*].spec.containers[*].image}" | tr -s '[[:space:]]' '\n' | sort | uniq -c
```

## Output:

```
l vesoft/nebula-graphd:v3.4.1
l vesoft/nebula-metad:v3.4.1
svesoft/nebula-storaged:v3.4.1
```

# 19.7.3 Upgrade a NebulaGraph cluster with Helm

# **Prerequisites**

You have created a NebulaGraph cluster with Helm. For details, see Create a NebulaGraph cluster with Helm.

### Steps

1. Update the information of available charts locally from chart repositories.

```
helm repo update
```

2. Set environment variables to your desired values.

3. Upgrade a NebulaGraph cluster.

For example, upgrade a cluster to v3.4.1.

```
helm upgrade "${NEBULA_CLUSTER_NAME}" nebula-operator/nebula-cluster \
--namespace="${NEBULA_CLUSTER_NAMESPACE}" \
--set nameOverride=${NEBULA_CLUSTER_NAME} \
--set nebula.version=v3.4.1
```

The value of --set nebula.version specifies the version of the cluster you want to upgrade to.

4. Run the following command to check the status and version of the upgraded cluster.

Check cluster status:

```
$ kubectl -n "${NEBULA_CLUSTER_NAMESPACE}" get pod -l "app.kubernetes.io/cluster=${NEBULA_CLUSTER_NAME}"
                    READY STATUS
                                      RESTARTS AGE
nebula-graphd-0
                    1/1
                            Running
   2m
nebula-graphd-1
nebula-metad-0
                    1/1
                            Running
   2m
                            Running
nebula-metad-1
                   1/1
                            Running
   2m
nebula-metad-2
                    1/1
                            Running
   2m
nebula-storaged-0 1/1
                            Running
   2m
nebula-storaged-1
                   1/1
                            Running
   2m
nebula-storaged-2
                   1/1
                            Running
   2m
```

### Check cluster version:

```
$ kubectl get pods -l app.kubernetes.io/cluster=nebula -o jsonpath="{.items[*].spec.containers[*].image}" |tr -s '[[:space:]]' '\n' |sort |uniq -c
1 vesoft/nebula-graphd:v3.4.1
1 vesoft/nebula-metad:v3.4.1
3 vesoft/nebula-storaged:v3.4.1
```

# 19.7.4 Accelerate the upgrade process

The upgrade process of a cluster is a rolling update process and can be time-consuming due to the state transition of the leader partition replicas in the Storage service. You can configure the enableForceUpdate field in the cluster instance's YAML file to skip the leader partition replica transfer operation, thereby accelerating the upgrade process. For more information, see Specify a rolling update strategy.

# 19.8 NebulaGraph cluster rolling update strategy

NebulaGraph clusters use a distributed architecture to divide data into multiple logical partitions, which are typically evenly distributed across different nodes. In distributed systems, there are usually multiple replicas of the same data. To ensure the consistency of data across multiple replicas, NebulaGraph clusters use the Raft protocol to synchronize multiple partition replicas. In the Raft protocol, each partition elects a leader replica, which is responsible for handling write requests, while follower replicas handle read requests.

When a NebulaGraph cluster created by NebulaGraph Operator performs a rolling update, a storage node temporarily stops providing services for the update. For an overview of rolling updates, see Performing a Rolling Update. If the node hosting the leader replica stops providing services, it will result in the unavailability of read and write operations for that partition. To avoid this situation, by default, Operator migrates the leader replicas to other unaffected nodes during the rolling update process of a NebulaGraph cluster. This way, when a storage node is being updated, the leader replicas on other nodes can continue processing client requests, ensuring the read and write availability of the cluster.

The process of migrating all leader replicas from one storage node to the other nodes may take a long time. To better control the rolling update duration, Operator provides a field called <code>enableForceUpdate</code>. When it is confirmed that there is no external access traffic, you can set this field to <code>true</code>. This way, the leader replicas will not be migrated to other nodes, thereby speeding up the rolling update process.

# 19.8.1 Rolling update trigger conditions

Operator triggers a rolling update of the NebulaGraph cluster under the following circumstances:

- The version of the NebulaGraph cluster changes.
- The configuration of the NebulaGraph cluster changes.

# 19.8.2 Specify a rolling update strategy

In the YAML file for creating a cluster instance, add the spec.storaged.enableForceUpdate field and set it to true or false to control the rolling update speed.

When <code>enableForceUpdate</code> is set to <code>true</code>, it means that the partition leader replicas will not be migrated, thus speeding up the rolling update process. Conversely, when set to <code>false</code>, it means that the leader replicas will be migrated to other nodes to ensure the read and write availability of the cluster. The default value is <code>false</code>.



When setting enableForceUpdate to true, make sure there is no traffic entering the cluster for read and write operations. This is because this setting will force the cluster pods to be rebuilt, and during this process, data loss or client request failures may occur.

# Configuration example:

```
spec:
...
storaged:
enableForceUpdate: true // When set to true, it speeds up the rolling update process.
...
```

Last update: February 19, 2024

- 969/1069 - 2023 Vesoft Inc.

# 19.9 Backup and restore data using NebulaGraph Operator

This article introduces how to back up and restore data of the NebulaGraph cluster on Kubernetes.



This feature is only for the enterprise edition NebulaGraph clusters on Kubernetes.

#### 19.9.1 Overview

NebulaGraph BR (Enterprise Edition) is a command line tool for data backup and recovery of NebulaGraph enterprise edition. NebulaGraph Operator is based on the BR tool to achieve data backup and recovery for NebulaGraph clusters on Kubernetes.

When backing up data, NebulaGraph Operator creates a Job to back up the data in the NebulaGraph cluster to the specified storage service.

When restoring data, NebulaGraph Operator checks the specified backup NebulaGraph cluster for existence, and whether the access to remote storage is normally based on the information defined in the NebulaRestore resource object. It then creates a new cluster and restores the backup data to the new NebulaGraph cluster. For more information, see restore flowchart.

# 19.9.2 Prerequisites

To backup and restore data using NebulaGraph Operator, the following conditions must be met:

- Nebula Operator version >= 1.4.0.
- The enterprise edition NebulaGraph cluster deployed on Kubernetes is running.
- In the YAML file used to create the cluster, spec.enableBR is set to true.

```
// Partial content of a sample cluster YAML file.
apiVersion: apps.nebula-graph.io/v1alpha1
kind: NebulaCluster
metadata:
name: nebula
spec:
enableBR: true // Set to true to enable the backup and restore function.
...
```

- Only storage services that use the S3 protocol (such as AWS S3, Minio, etc.) can be used to back up and restore data.
- Sufficient computing resources are available in the cluster to restore data.

# 19.9.3 Backup

### Notes

- NebulaGraph Operator supports full and incremental backups.
- During data backup, DDL and DML statements in the specified graph space will be blocked. We recommend performing the operation during off-peak hours, such as from 2:00 am to 5:00 am.
- The cluster executing incremental backups and the cluster specified for the last backup must be the same, and the (storage bucket) path for the last backup must be the same.
- Ensure that the time between each incremental backup and the last backup is less than a wal\_ttl.
- Specifying the backup data of a specified graph space is not supported.

## Full backup

When backing up data to a storage service compatible with the S3 protocol, you need to create a backup Job, which will back up the full NebulaGraph data to the specified storage location.

- 970/1069 - 2023 Vesoft Inc.

Here is an example of the YAML file for a full backup Job:

```
apiVersion: batch/v1
kind: Job
metadata:
 name: nebula-full-backup
  parallelism: 1
ttlSecondsAfterFinished: 60
   template:
     spec:
       restartPolicy: OnFailure
       containers:
    image: vesoft/br-ent:v3.4.1
    imagePullPolicy: Always
            name: backup
command:
               - /bin/sh
               - -ecx
                - exec /usr/local/bin/nebula-br backup full
                - --meta $META_ADDRESS:9559
- --storage s3://$BUCKET
                - --s3.access_key $ACCESS_KEY
                - --s3.secret_key $SECRET_KEY
- --s3.region $REGION
                - --s3.endpoint https://s3.$REGION.amazonaws.com
```

# Incremental backup

Except for the name of the Job and the command specified in <code>spec.template.spec.containers[0].command</code>, the YAML file for incremental backup is the same as that for a full backup. Here is an example of the YAML file for incremental backup:

```
apiVersion: batch/v1
kind: Job
metadata:
 name: nebula-incr-backup
 parallelism: 1
ttlSecondsAfterFinished: 60
  template:
       restartPolicy: OnFailure
       containers:
    image: vesoft/br-ent:v3.4.1
            imagePullPolicy: Always
name: backup
command:
               - /bin/sh
               - -ecx
               - exec /usr/local/bin/nebula-br backup incr
               - --meta $META_ADDRESS:9559
- --base $BACKUP_NAME
               - --storage s3://$BUCKET
               - --s3.access_key $ACCESS_KEY
- --s3.secret_key $SECRET_KEY
               - --s3.region $REGION
- --s3.endpoint https://s3.$REGION.amazonaws.com
```

# Parameter description

The main parameters are described as follows:

Parameter	Default value	Description
spec.parallelism	1	The number of tasks executed in parallel.
spec.ttlSecondsAfterFinished	60	The time to keep task information after the task is completed.
spec.template.spec.containers[0].image	vesoft/br-ent: 3.4.1	The image address of the NebulaGraph BR Enterprise Edition tool.
spec.template.spec.containers[0].command	-	The command for backing up data to the storage service compatible with the S3 protocol.  For descriptions of the options in the command, see Parametr description.

For more settings of the Job, see Kubernetes Jobs.

After the YAML file for the backup Job is set, run the following command to start the backup Job:

```
kubectl apply -f <backup_file_name>.yaml
```

When the data backup succeeds, a backup file is generated in the specified storage location. For example, the backup file name is  $BACKUP_2023_02_12_10_04_16$ .

### 19.9.4 Restore

#### Notes

- After the data recovery is successful, a new cluster will be created, and the old cluster will not be deleted. Users can decide whether to delete the old cluster themselves.
- There will be a period of service unavailability during the data recovery process, so it is recommended to perform the operation during a low period of business activity.

#### **Process**

When restoring data from a compatible S3 protocol service, you need to create a Secret to store the credentials for accessing the compatible S3 protocol service. Then create a resource object (NebulaRestore) for restoring the data, which will instruct the Operator to create a new NebulaGraph cluster based on the information defined in this resource object and restore the backup data to the newly created cluster.

Here is an example YAML for restoring data based on the backup file BACKUP\_2023\_02\_12\_10\_04\_16:

```
apiVersion: v1
kind: Secret
metadata:
 name: aws-s3-secret
type: Opaque
data:
 access-key: QVNJQVE0WFLxxx
 secret-key: ZFJ60EdNcDdxenMwVGxxx
apiVersion: apps.nebula-graph.io/v1alpha1
kind: NebulaRestore
metadata:
 name: restorel
spec:
    clusterName: nebula
    backupName: "BACKUP_2023_02_12_10_04_16"
    concurrency: 5
      region: "us-west-2"
      endpoint: "https://s3.us-west-2.amazonaws.com"
secretName: "aws-s3-secret"
```

# **Parameter Description**

## • Secret

Parameter	<b>Default Value</b>	Description
metadata.name	-	The name of the Secret.
type	0paque	The type of the Secret. See Types of Secret for more information.
data.access-key	-	The AccessKey for accessing the S3 protocol-compatible storage service.
data.secret-key	-	The SecretKey for accessing the S3 protocol-compatible storage service.

# • NebulaRestore

Parameter	<b>Default Value</b>	Description
metadata.name	-	The name of the resource object NebulaRestore.
spec.br.clusterName	-	The name of the backup cluster.
spec.br.backupName	-	The name of the backup file. Restore data based on this backup file.
spec.br.concurrency	5	The number of concurrent downloads when restoring data. The default value is $ 5  . $
spec.br.s3.region	-	The geographical region where the S3 storage bucket is located.
spec.br.s3.bucket	-	The path of the S3 storage bucket where backup data is stored.
spec.br.s3.endpoint	-	The access address of the S3 storage bucket.
spec.br.s3.secretName	-	The name of the Secret that is used to access the S3 storage bucket.

After setting up the YAML file for restoring the data, run the following command to start the restore job:

kubectl apply -f <restore\_file\_name>.yaml

Run the following command to check the status of the NebulaRestore object.

```bash kubectl get rt -w

Last update: February 19, 2024

- 973/1069 - 2023 Vesoft Inc.

# 19.10 Self-healing

NebulaGraph Operator calls the interface provided by NebulaGraph clusters to dynamically sense cluster service status. Once an exception is detected (for example, a component in a NebulaGraph cluster stops running), NebulaGraph Operator automatically performs fault tolerance. This topic shows how Nebular Operator performs self-healing by simulating cluster failure of deleting one Storage service Pod in a NebulaGraph cluster.

## 19.10.1 Prerequisites

Install NebulaGraph Operator

#### 19.10.2 Steps

- 1. Create a NebulaGraph cluster. For more information, see Deploy NebulaGraph clusters with Kubectl or Deploy NebulaGraph clusters with Helm.
- 2. Delete the Pod named <cluster\_name>-storaged-2 after all pods are in the Running status.

```
kubectl delete pod <cluster-name>-storaged-2 --now
```

 $\hbox{$^{<$cluster\_name>}$ is the name of your NebulaGraph cluster.}\\$ 

3. NebulaGraph Operator automates the creation of the Pod named <cluster-name>-storaged-2 to perform self-healing.

Run the kubectl get pods command to check the status of the Pod <cluster-name>-storaged-2.

```
nebula-cluster-storaged-1 1/1 Running 0 5d23h
nebula-cluster-storaged-2 0/1 ContainerCreating 0 1s
...

nebula-cluster-storaged-1 1/1 Running 0 5d23h
nebula-cluster-storaged-1 1/1 Running 0 4m2s
...
```

When the status of <cluster-name>-storaged-2 is changed from ContainerCreating to Running, the self-healing is performed successfully.

Last update: February 19, 2024

- 974/1069 - 2023 Vesoft Inc.

# 19.11 FAQ

#### 19.11.1 Does NebulaGraph Operator support the v1.x version of NebulaGraph?

No, because the v1.x version of NebulaGraph does not support DNS, and NebulaGraph Operator requires the use of DNS.

#### 19.11.2 Is cluster stability guaranteed if using local storage?

There is no guarantee. Using local storage means that the Pod is bound to a specific node, and NebulaGraph Operator does not currently support failover in the event of a failure of the bound node.

## 19.11.3 How to ensure the stability of a cluster when scaling the cluster?

It is suggested to back up data in advance so that you can roll back data in case of failure.

## 19.11.4 Is the replica in the Operator docs the same as the replica in the NebulaGraph core docs?

They are different concepts. A replica in the Operator docs indicates a pod replica in K8s, while a replica in the core docs is a replica of a NebulaGraph storage partition.

Last update: February 19, 2024

- 975/1069 - 2023 Vesoft Inc.

# 20. Graph computing

# 20.1 Algorithm overview

Graph computing can detect the graph structure, such as the communities in a graph and the division of a graph. It can also reveal the inherent characteristics of the correlation between various vertexes, such as the centrality and similarity of the vertices. This topic introduces the algorithms and parameters supported by NebulaGraph.



This topic only introduces the parameters of NebulaGraph Analytics. For details about the parameters of NebulaGraph Algorithm, see algorithm.

#### Q Note

The algorithm parameters need to be set when performing graph computing, and there are requirements for data sources. The data source needs to contain source vertexes and destination vertexes. PageRank, DegreeWithTime, SSSP, APSP, LPA, HANP, and Louvain algorithms must include weight.

- If the data source comes from HDFS, users need to specify a CSV file that contains src and dst columns. Some algorithms also need to contain a weight column.
- If the data source comes from NebulaGraph, users need to specify the edge types that provide src and dst columns. Some algorithms also need to specify the properties of the edge types as weight columns.

## 20.1.1 Node importance measurement

#### **PageRank**

The PageRank algorithm calculates the relevance and importance of vertices based on their relationships. It is commonly used in search engine page rankings. If a page is linked by many other pages, the page is more important (PageRank value is higher). If a page with a high PageRank value links to other pages, the PageRank value of the linked pages will increase.

- 976/1069 - 2023 Vesoft Inc.

- NebulaGraph Analytics
- Input parameters

| Parameter   | Predefined<br>value | Description  |
|-------------|---------------------|--|
| ITERATIONS  | 10                  | The maximum number of iterations.  |
| IS_DIRECTED | true                | Whether to consider the direction of the edges. If set to false, the system automatically adds the reverse edge.                               |
| EPS         | 0.0001              | The convergence accuracy. When the difference between the result of two iterations is less than the EPS value, the iteration is not continued. |
| DAMPING     | 0.85                | The damping coefficient. It is the jump probability after visiting a page.   |

## • Output parameters

| Parameter | Туре                   | Description                       |
|-----------|------------------------|-----------------------------------|
| VID       | Determined by vid_type | The vertex ID.                    |
| VALUE     | double                 | The PageRank value of the vertex. |

#### **KCore**

The KCore algorithm is used to calculate the subgraph composed of no vertexes less than K degree, usually used in community discovery, financial risk control and other scenarios. The calculation result is one of the most commonly used reference values to judge the importance of a vertex, which reflects the propagation ability of a vertex.

- NebulaGraph Analytics
- Input parameters

| Parameter | Predefined value | Description   |
|-----------|------------------|---|
| ТҮРЕ      | vertex           | The calculation type. Available values are <code>vertex</code> and <code>subgraph</code> . When set to <code>vertex</code> , the system calculates the number of cores for each <code>vertex</code> . |
| KMIN      | [1]              | Set the minimum value of K when performing the range calculation. Takes effect only when $\mbox{\em TYPE} = \mbox{\em subgraph}$ .  |
| КМАХ      | 1000000          | Set the maximum value of $K$ when performing the range calculation. Takes effect only when $\mbox{\em TYPE} = \mbox{\em subgraph}$ .  |

• Output parameters when TYPE=vertex

| Parameter | Туре                   | Description                            |
|-----------|------------------------|--|
| VID       | Determined by vid_type | The vertex ID.                         |
| VALUE     | int                    | Outputs the core degree of the vertex. |

• Output parameters when TYPE=subgraph

| Parameter | Туре                   | Description                          |
|-----------|------------------------|--------------------------------------|
| VID       | Determined by vid_type | The vertex ID.                       |
| VALUE     | The same with VID      | Outputs the neighbors of the vertex. |

#### DegreeCentrality (NStepDegree)

The Degree Centrality algorithm is used to find the popular vertexes in a graph. Degree centrality measures the number of incoming or outgoing (or both) relationships from a vertex, depending on the direction of the projection of the relationship. The greater the degree of a vertex is, the higher the degree centrality of the vertex is, and the more important the vertex is in the network.



NebulaGraph Analytics only estimates DegreeCentrality roughly.

- 978/1069 - 2023 Vesoft Inc.

- NebulaGraph Analytics
- Input parameters

| Parameter   | Predefined<br>value | Description  |
|-------------|---------------------|--|
| IS_DIRECTED | true                | Whether to consider the direction of the edges. If set to false, the system automatically adds the reverse edge. |
| STEP        | 3                   | The degree of calculation1 means infinity.   |
| BITS        | 6                   | The hyperloglog bit width for cardinality estimation.  |
| TYPE        | both                | The direction of the edges for calculation. Optional values are $\ \mbox{in}$ , out and both .                   |

• Output parameters when TYPE=both

| Parameter   | Туре                   | Description  |
|-------------|------------------------|--|
| VID         | Determined by vid_type | The vertex ID.   |
| BOTH_DEGREE | int                    | Outputs the bidirectional degree centrality of the vertex. |
| OUT_DEGREE  | int                    | Outputs the outbound degree centrality of the vertex.      |
| IN_DEGREE   | int                    | Outputs the inbound degree centrality of the vertex.       |
|             |                        |  |

• Output parameters when TYPE=out

| Parameter  | Туре                   | Description   |
|------------|------------------------|---|
| VID        | Determined by vid_type | The vertex ID.  |
| OUT_DEGREE | int                    | Outputs the outbound degree centrality of the vertex. |

• Output parameters when TYPE=in

| Parameter | Туре                   | Description  |
|-----------|------------------------|--|
| VID       | Determined by vid_type | The vertex ID.                                       |
| IN_DEGREE | int                    | Outputs the inbound degree centrality of the vertex. |

# DegreeWithTime

The DegreeWithTime algorithm is used to count neighbors based on the time range of edges to find out the popular vertexes in a graph.



This algorithm is supported by NebulaGraph Analytics only.

- 979/1069 - 2023 Vesoft Inc.

#### • Input parameters

| Parameter   | Predefined<br>value | Description  |
|-------------|---------------------|--|
| ITERATIONS  | 10                  | The maximum number of iterations.  |
| IS_DIRECTED | true                | Whether to consider the direction of the edges. If set to false, the system automatically adds the reverse edge. |
| BEGIN_TIME  | -                   | The begin time.  |
| END_TIME    | -                   | The end time.  |

## • Output parameters when TYPE=both

| VID Determined by vid_type The vertex ID.  BOTH DEGREE int Outputs the bidirectional popularity of the vertex. | Parameter   | Туре                   | Description   |
|--|-------------|------------------------|---|
| BOTH DEGREE int Outputs the bidirectional popularity of the vertex.  | VID         | Determined by vid_type | The vertex ID.                                      |
|  | BOTH_DEGREE | int                    | Outputs the bidirectional popularity of the vertex. |
| OUT_DEGREE int Outputs the outbound popularity of the vertex.  | OUT_DEGREE  | int                    | Outputs the outbound popularity of the vertex.      |
| IN_DEGREE int Outputs the inbound popularity of the vertex.  | IN_DEGREE   | int                    | Outputs the inbound popularity of the vertex.       |

## • Output parameters when TYPE=out

| Parameter  | Туре                   | Description                                    |
|------------|------------------------|--|
| VID        | Determined by vid_type | The vertex ID.                                 |
| OUT_DEGREE | int                    | Outputs the outbound popularity of the vertex. |

## • Output parameters when TYPE=in

| Parameter | Туре                   | Description                                   |
|-----------|------------------------|---|
| VID       | Determined by vid_type | The vertex ID.                                |
| IN_DEGREE | int                    | Outputs the inbound popularity of the vertex. |

## BetweennessCentrality

The BetweennessCentrality algorithm is used to detect the amount of influence a vertex has on the flow of information in a graph. It is used to find the vertexes that act as bridges between one part of the graph and another. Each vertex is given a score, the betweenness centrality score, based on the number of shortest paths through that vertex.

- NebulaGraph Analytics
- Input parameters

| Parameter   | Predefined<br>value | Description  |
|-------------|---------------------|--|
| ITERATIONS  | 10                  | The maximum number of iterations.  |
| IS_DIRECTED | true                | Whether to consider the direction of the edges. If set to false, the system automatically adds the reverse edge. |
| CHOSEN      | -1                  | The selected vertex ID, -1 means random selection.   |
| CONSTANT    | 2                   | The constant.  |

## • Output parameters

| Parameter | Туре                   | Description                                     |
|-----------|------------------------|---|
| VID       | Determined by vid_type | The vertex ID.                                  |
| VALUE     | double                 | The betweenness centrality score of the vertex. |

#### ClosenessCentrality

The ClosenessCentrality algorithm is used to calculate the reciprocal of the average of the shortest distance from one vertex to all other reachable vertexes. The larger the value is, the closer the vertex is to the center of the graph, and it can also be used to measure how long it takes for information to be transmitted from that vertex to other vertexes.

Parameter descriptions are as follows:

- NebulaGraph Analytics
- Input parameters

| Parameter   | Predefined value | Description  |
|-------------|------------------|--|
| IS_DIRECTED | true             | Whether to consider the direction of the edges. If set to false, the system automatically adds the reverse edge. |
| NUM_SAMPLES | 10               | The number of sample vertices.   |

#### · Output parameters

| Parameter | Туре                   | Description                                   |
|-----------|------------------------|---|
| VID       | Determined by vid_type | The vertex ID.                                |
| VALUE     | double                 | The closeness centrality score of the vertex. |

## 20.1.2 Path

## APSP

The APSP (Full Graph Shortest Path) algorithm is used to find all shortest paths between two vertexes in a graph.



This algorithm is supported by NebulaGraph Analytics only.

- 981/1069 - 2023 Vesoft Inc.

• Output parameters

| Parameter | Туре                   | Description  |
|-----------|------------------------|--|
| VID1      | Determined by vid_type | The VID of the source vertex.                            |
| VID2      | Determined by vid_type | The VID of the destination vertex.                       |
| DISTANCE  | double                 | Outputs the distance from ${\tt VID1}$ to ${\tt VID2}$ . |

#### SSSP

The SSSP (Single source shortest Path) algorithm is used to calculate the shortest path length from a given vertex (source vertex) to other vertexes. It is usually used in scenarios such as network routing and path designing.

Parameter descriptions are as follows:

- NebulaGraph Analytics
- Input parameters

| Parameter | Predefined value | Description                   |
|-----------|------------------|-------------------------------|
| ROOT      | -                | The VID of the source vertex. |

• Output parameters

| Parameter | Туре                   | Description   |
|-----------|------------------------|---|
| VID       | Determined by vid_type | The VID of the source vertex.                           |
| DISTANCE  | double                 | Outputs the distance from ${\tt ROOT}$ to ${\tt VID}$ . |

#### BFS

The BFS (Breadth First traversal) algorithm is a basic graph traversal algorithm. It gives a source vertex and accesses other vertexes with increasing hops, that is, it traverses all the adjacent vertexes of the vertex first and then extends to the adjacent vertexes of the adjacent vertexes.

Parameter descriptions are as follows:

- NebulaGraph Analytics
- Input parameters

| Parameter   | Predefined value | Description  |
|-------------|------------------|--|
| IS_DIRECTED | true             | Whether to consider the direction of the edges. If set to false, the system automatically adds the reverse edge. |
| ROOT        | -                | The VID of the source vertex.  |

• Output parameters

| Parameter | Туре                   | Description   |
|-----------|------------------------|---|
| ROOT      | Determined by vid_type | The VID of the source vertex.   |
| VISITED   | int                    | Outputs the number of the vertex accessed by $\ensuremath{^{\rm ROOT}}$ . |

#### ShortestPath

The ShortestPath algorithm is used to find the shortest path between any two vertices in the graph, which is frequently applied in scenarios such as path design and network planning.

- NebulaGraph Analytics
- Input parameters

| Parameter | Predefined value | Description  |
|-----------|------------------|--|
| src       | "100"            | Starting vertices. Multiple VIDs are separated by commas (,).    |
| dst       | "200"            | Destination vertices. Multiple VIDs are separated by commas (,). |

## • Output parameters

| Parameter | Туре | Description  |
|-----------|------|--|
| VALUE     | list | Returns the vertices in the shortest path. The format is src, vid1,vid2dst . If there are multiple shortest paths between two vertices, only one path is returned. |

#### 20.1.3 Community discovery

#### LPA

The LPA (label propagation) algorithm is a semi-supervised learning method based on graph. Its basic idea is to use label information of labeled vertexes to predict label information of unlabeled vertexes. vertexes include labeled and unlabeled data, and their edges represent the similarity of two vertexes. The labels of vertexes are transferred to other vertexes according to the similarity. Label data is like a source that can be labeled for unlabeled data. The greater the similarity of vertexes is, the easier the label is to spread.

Parameter descriptions are as follows:

- NebulaGraph Analytics
- Input parameters

| Parameter            | Predefined value | Description   |
|----------------------|------------------|---|
| ITERATIONS           | 10               | The maximum number of iterations.   |
| IS_DIRECTED          | true             | Whether to consider the direction of the edges. If set to false, the system automatically adds the reverse edge.  |
| IS_CALC_MODULARITY   | false            | Whether to calculate modularity.  |
| IS_OUTPUT_MODULARITY | false            | Whether to calculate and output module degrees. When set to true, the default output is to the third column of the file, but it can also be output to NebulaGraph with options -nebula_output_props and -nebula_output_types. Output to NebulaGraph is not yet supported when using Explorer. |
| IS_STAT_COMMUNITY    | false            | Whether to count the number of communities.   |

#### • Output parameters

| Parameter | Туре                   | Description                                      |
|-----------|------------------------|--|
| VID       | Determined by vid_type | The vertex ID.                                   |
| LABEL     | The same with VID      | Outputs the vertex IDs that have the same label. |

#### HANP

The HANP (Hop Preference & Node Preference) algorithm is an optimization algorithm of LPA algorithm, which considers other information of labels, such as degree information, distance information, etc., and introduces attenuation coefficient during propagation to prevent transition propagation.

Parameter descriptions are as follows:

- NebulaGraph Analytics
- Input parameters

| Parameter            | Predefined value | Description   |
|----------------------|------------------|---|
| ITERATIONS           | 10               | The maximum number of iterations.   |
| IS_DIRECTED          | true             | Whether to consider the direction of the edges. If set to false, the system automatically adds the reverse edge.  |
| PREFERENCE           | 1.0              | The bias of the neighbor vertex degree. $m>0$ indicates biasing the neighbor with high vertex degree, $m<0$ indicates biasing the neighbor with low vertex degree, and $m=0$ indicates ignoring the neighbor vertex degree.   |
| HOP_ATT              | 0.1              | The attenuation coefficient. The value ranges from $$ 0 to $$ 1. The larger the value, the faster it decays and the fewer times it can be passed.   |
| IS_OUTPUT_MODULARITY | false            | Whether to calculate and output module degrees. When set to true, the default output is to the third column of the file, but it can also be output to NebulaGraph with options -nebula_output_props and -nebula_output_types. Output to NebulaGraph is not yet supported when using Explorer. |
| IS_STAT_COMMUNITY    | false            | Whether to count the number of communities.   |

## • Output parameters

| Parameter | Туре                   | Description                                      |
|-----------|------------------------|--|
| VID       | Determined by vid_type | The vertex ID.                                   |
| LABEL     | The same with VID      | Outputs the vertex IDs that have the same label. |

## ConnectedComponent

The ConnectedComponent algorithm is used to calculate a subgraph of a graph in which all vertexes are connected to each other. Strongly Connected Component takes the path direction into account, while Weakly Connected Component does not.



NebulaGraph Analytics only supports Weakly Connected Component.

- 984/1069 - 2023 Vesoft Inc.

- NebulaGraph Analytics
- Input parameters

| Parameter            | Predefined value | Description   |
|----------------------|------------------|---|
| IS_DIRECTED          | true             | Whether to consider the direction of the edges. If set to false, the system automatically adds the reverse edge.  |
| IS_CALC_MODULARITY   | false            | Whether to calculate modularity.  |
| IS_OUTPUT_MODULARITY | false            | Whether to calculate and output module degrees. When set to true, the default output is to the third column of the file, but it can also be output to NebulaGraph with options -nebula_output_props and -nebula_output_types. Output to NebulaGraph is not yet supported when using Explorer. |
| IS_STAT_COMMUNITY    | false            | Whether to count the number of communities.   |

#### • Output parameters

| Parameter | Туре                   | Description                                      |
|-----------|------------------------|--|
| VID       | Determined by vid_type | The vertex ID.                                   |
| LABEL     | The same with VID      | Outputs the vertex IDs that have the same label. |

#### Louvain

The Louvain algorithm is a community discovery algorithm based on modularity. This algorithm performs well in efficiency and effect, and can be used to find hierarchical community structures. Its optimization goal is to maximize the modularity of the whole community network. Modularity is used to distinguish the differences in link density within and between communities, and to measure how well each vertex divides the community. In general, a good clustering approach will result in more modularity within communities than between communities.

- NebulaGraph Analytics
- Input parameters

| Parameter            | Predefined value | Description   |
|----------------------|------------------|---|
| IS_DIRECTED          | true             | Whether to consider the direction of the edges. If set to false, the system automatically adds the reverse edge.  |
| OUTER_ITERATION      | 20               | The maximum number of iterations in the first phase.  |
| INNER_ITERATION      | 10               | The maximum number of iterations in the second phase.   |
| IS_CALC_MODULARITY   | false            | Whether to calculate modularity.  |
| IS_OUTPUT_MODULARITY | false            | Whether to calculate and output module degrees. When set to true, the default output is to the third column of the file, but it can also be output to NebulaGraph with options -nebula_output_props and -nebula_output_types. Output to NebulaGraph is not yet supported when using Explorer. |
| IS_STAT_COMMUNITY    | false            | Whether to count the number of communities.   |

## • Output parameters

| Parameter | Туре                   | Description                                      |
|-----------|------------------------|--|
| VID       | Determined by vid_type | The vertex ID.                                   |
| LABEL     | The same with VID      | Outputs the vertex IDs that have the same label. |

## InfoMap

The InfoMap algorithm uses double encoding to classify directed graphs into communities. The encoding reuse of nodes in different communities can greatly shorten the length of description information. In terms of implementation, the algorithm includes the PageRank algorithm, which converts a random walk into a random surf.



This algorithm is supported by NebulaGraph Analytics only.

- NebulaGraph Analytics
- Input parameters

| Parameter          | Predefined value | Description   |
|--------------------|------------------|---|
| pagerank_iter      | 10               | The maximum number of iterations of the internal PageRank algorithm. \\ |
| pagerank_threshold | 0.0001           | The convergence accuracy of the internal PageRank algorithm.            |
| teleport_prob      | 0.15             | The teleportation probability.  |
| inner_iter         | 3                | The number of inner iterations.   |
| outer_iter         | 2                | The number of outer iterations.   |
| comm_info_num      | 100              | The number of communities exported.                                     |

## • Output parameters

| Paramete | r Type             | Description                                      |
|----------|--------------------|--|
| VID      | Determined by vid_ | type The vertex ID.                              |
| LABEL    | The same with VID  | Outputs the vertex IDs that have the same label. |

# 20.1.4 Graph feature

## TriangleCount

The TriangleCount algorithm is used to count the number of triangles in a graph. The more triangles, the higher the degree of vertex association in the graph, the tighter the organizational relationship.

- 987/1069 - 2023 Vesoft Inc.

- NebulaGraph Analytics
- · Input parameters

| Parameter                | Predefined<br>value | Description  |
|--------------------------|---------------------|--|
| 0PT                      | 3                   | The calculation type. Optional values are 1, 2 and 3.1 indicates counting the entire graph, 2 indicates counting through each vertex, 3 indicates listing all triangles. |
| REMOVED_DUPLICATION_EDGE | true                | Whether to exclude repeated edges.   |
| REMOVED_SELF_EDGE        | true                | Whether to exclude self-loop edge.   |

• Output parameters when OPT=1

| Parameter | Туре | Description  |
|-----------|------|--|
| COUNT     | int  | Outputs the number of the triangles in the full graph space. |

• Output parameters when OPT=2

| Parameter | Туре                   | Description  |
|-----------|------------------------|--|
| VID       | Determined by vid_type | The vertex ID.   |
| COUNT     | int                    | Outputs the number of the triangles based on the vertex. |

• Output parameters when OPT=3

| Parameter | Туре              | Description   |
|-----------|-------------------|---|
| VID1      | The same with VID | Outputs the ID of the vertex A that forms the triangle. |
| VID2      | The same with VID | Outputs the ID of the vertex B that forms the triangle. |
| VID3      | The same with VID | Outputs the ID of the vertex C that forms the triangle. |

## Node2Vec

The Node2Vec algorithm proposed a more reasonable graph feature learning method based on DeepWalk, and proposed a semi-supervised algorithm for scalable feature learning in networks. SGD was used to optimize a custom graph-based objective function, which could maximize the network domain information of nodes reserved in d-dimensional feature space. Based on the random walk, a second order random walk process is designed, which is equivalent to an extension of DeepWalk algorithm, and preserves the graph characteristics of neighbor nodes. Applicable to node function similarity comparison, node structure similarity comparison, community clustering and other scenarios.R

Parameter descriptions are as follows:

- NebulaGraph Analytics
- Input parameters |Parameter|Predefined value|Description| |:--|:--| | is\_weighted | false | Random walk with bias or not.| | p | 1.0 | The backward bias for random walk.| | q | 0.5 | The forward bias for random walk.| | epoch | 1 | The number of iterations.| | step | 10 | The number of steps per iteration.| | rate | 0.02 | The rate of the random walk.|
- Output parameters Output multiple columns where vertices in the same column are associated.

#### Tree\_stat

The Tree\_stat algorithm counts the width or depth of a subgraph with a specified root vertex.



This algorithm is supported by NebulaGraph Analytics only.

- NebulaGraph Analytics
- Input parameters

| Parameter | Predefined value | Description   |
|-----------|------------------|---|
| root      | 100              | The VID of the root vertex.   |
| stat      | width,depth      | Counts width or depth. Multiple values are separated by commas (,). |

• Output parameters

| Parameter | Туре | Description   |
|-----------|------|---|
| VALUE     | list | Returns a row of statistics in the same format as the stat parameter. |

## **HyperANF**

The HyperANF algorithm is used to evaluate the average distance between any two vertices in a graph.



This algorithm is supported by NebulaGraph Analytics only.

- NebulaGraph Analytics
- Input parameters

| Parameter | Predefined value | Description  |
|-----------|------------------|--|
| bits      | 6                | The bit length of the HyperLogLog counter. The value ranges from $6$ to $16$ . |

• Output parameters

| Parameter | Туре   | Description           |
|-----------|--------|-----------------------|
| VALUE     | double | The average distance. |

## 20.1.5 Clustering

## ClusteringCoefficient

The ClusteringCoefficient algorithm is used to calculate the clustering degree of vertexes in a graph. In all kinds of network structures reflecting the real world, especially social network structures, network groups with relatively high density tend to be formed between various vertexes. In other words, compared with the networks randomly connected between two vertexes, the aggregation coefficient of the real world network is higher.

- NebulaGraph Analytics
- Input parameters

| Parameter                | Predefined<br>value | Description  |
|--------------------------|---------------------|--|
| ТУРЕ                     | local               | The clustering type. Optional values are local and global. local indicates counting through each vertex, global indicates counting the entire graph. |
| REMOVED_DUPLICATION_EDGE | true                | Whether to exclude repeated edges.   |
| REMOVED_SELF_EDGE        | true                | Whether to exclude self-loop edge.   |

• Output parameters when TYPE=local

| Parameter | Туре                   | Description                                       |
|-----------|------------------------|---|
| VID       | Determined by vid_type | The vertex ID.                                    |
| VALUE     | double                 | Outputs the clustering coefficient of the vertex. |

• Output parameters when TYPE=global

| Parameter | Туре                   | Description   |
|-----------|------------------------|---|
| VID       | Determined by vid_type | The vertex ID.  |
| VALUE     | double                 | Outputs the clustering coefficient of the full graph space. There is only one line of data. |

# 20.1.6 Similarity

## Jaccard

The Jaccard algorithm is used to calculate the similarity of two vertexes (or sets) and predict the relationship between them. It is suitable for social network friend recommendation, relationship prediction and other scenarios.

- 990/1069 - 2023 Vesoft Inc.

- NebulaGraph Analytics
- Input parameters

| Parameter         | Predefined value | Description   |
|-------------------|------------------|---|
| IDS1              | -                | A set of VIDs. Multiple VIDs are separated by commas (,). It is not allowed to be empty.                      |
| IDS2              | -                | A set of VIDs. Multiple VIDs are separated by commas (,). It can be empty, and empty represents all vertexes. |
| REMOVED_SELF_EDGE | true             | Whether to exclude self-loop edges.   |

# • Output parameters

| Parameter | Туре                   | Description                           |
|-----------|------------------------|---------------------------------------|
| VID1      | Determined by vid_type | The ID of the first vertex.           |
| VID2      | Determined by vid_type | The ID of the second vertex.          |
| VALUE     | double                 | The similarity between VID1 and VID2. |
|           |                        |                                       |

Last update: February 19, 2024

- 991/1069 - 2023 Vesoft Inc.

# 20.2 NebulaGraph Algorithm

NebulaGraph Algorithm (Algorithm) is a Spark application based on GraphX. It uses a complete algorithm tool to perform graph computing on the data in the NebulaGraph database by submitting a Spark task. You can also programmatically use the algorithm under the lib repository to perform graph computing on DataFrame.

#### 20.2.1 Version compatibility

The correspondence between the NebulaGraph Algorithm release and the NebulaGraph core release is as follows.

| NebulaGraph        | NebulaGraph Algorithm |
|--------------------|-----------------------|
| nightly            | 3.0-SNAPSHOT          |
| $3.0.0 \sim 3.4.x$ | 3.x.0                 |
| 2.6.x              | 2.6.x                 |
| 2.5.0 \ 2.5.1      | 2.5.0                 |
| 2.0.0 \ 2.0.1      | 2.1.0                 |

## 20.2.2 Prerequisites

Before using the NebulaGraph Algorithm, users need to confirm the following information:

- The NebulaGraph services have been deployed and started. For details, see NebulaGraph Installation.
- The Spark version is 2.4.x.
- The Scala version is 2.11.
- (Optional) If users need to clone, compile, and package the latest Algorithm in Github, install Maven.

#### 20.2.3 Limitations

- For non-integer String data, it is recommended to use the algorithm interface. You can use the dense\_rank function of SparkSQL to encode the data as the Long type instead of the String type.
- Graph computing outputs vertex datasets, and the algorithm results are stored in DataFrames as the properties of vertices. You can do further operations such as statistics and filtering according to your business requirements.

!!!

Before Algorithm v3.1.0, when submitting the algorithm package directly, the data of the vertex ID must be an integer. That is, the vertex ID can be INT or String, but the data itself is an integer.

- 992/1069 - 2023 Vesoft Inc.

# 20.2.4 Supported algorithms

The graph computing algorithms supported by Nebula Graph Algorithm are as follows.

| Algorithm                    | Description                        | Scenario  | Properties name           | Properties type   |
|------------------------------|------------------------------------|---|---------------------------|-------------------|
| PageRank                     | The rank of pages                  | Web page<br>ranking, key<br>node mining                     | pagerank                  | double/<br>string |
| Louvain                      | Louvain                            | Community<br>mining,<br>hierarchical<br>clustering          | louvain                   | int/string        |
| KCore                        | K core                             | Community<br>discovery,<br>financial risk<br>control        | kcore                     | int/string        |
| LabelPropagation             | Label propagation                  | Information spreading, advertising, and community discovery | lpa                       | int/string        |
| Hanp                         | Label<br>propagation<br>advanced   | Community<br>discovery,<br>recommendation<br>system         | hanp                      | int/string        |
| ConnectedComponent           | Weakly<br>connected<br>component   | Community<br>discovery, island<br>discovery                 | СС                        | int/string        |
| Strongly Connected Component | Strongly<br>connected<br>component | Community discovery   | scc                       | int/string        |
| ShortestPath                 | The shortest path                  | Path planning,<br>network<br>planning                       | shortestpath              | string            |
| TriangleCount                | Triangle<br>counting               | Network<br>structure<br>analysis                            | trianglecount             | int/string        |
| GraphTriangleCount           | Graph<br>triangle<br>counting      | Network<br>structure and<br>tightness<br>analysis           | count                     | int               |
| BetweennessCentrality        | Intermediate centrality            | Key node mining,<br>node influence<br>computing             | betweenness               | double/<br>string |
| ClosenessCentrality          | Closeness<br>centrality            | Key node mining,<br>node influence<br>computing             | closeness                 | double/<br>string |
| DegreeStatic                 | Degree of statistical              | Graph structure analysis                                    | degree,inDegree,outDegree | int/string        |
| ClusteringCoefficient        | Aggregation coefficient            | Recommendation<br>system, telecom<br>fraud analysis         | clustercoefficient        | double/<br>string |
| Jaccard                      |                                    |   | jaccard                   | string            |

- 994/1069 - 2023 Vesoft Inc.

| Algorithm | Description              | Scenario   | Properties name | Properties type |
|-----------|--------------------------|--|-----------------|-----------------|
|           | Jaccard<br>similarity    | Similarity<br>computing,<br>recommendation<br>system |                 |                 |
| BFS       | Breadth-<br>First Search | Sequence<br>traversal,<br>shortest path<br>planning  | bfs             | string          |
| DFS       | Depth-First<br>Search    | Sequence<br>traversal,<br>shortest path<br>planning  | dfs             | string          |
| Node2Vec  | -                        | Graph<br>classification                              | node2vec        | string          |



When writing the algorithm results into the NebulaGraph, make sure that the tag in the corresponding graph space has properties names and data types corresponding to the table above.

## 20.2.5 Implementation methods

 $Nebula Graph \ Algorithm \ implements \ the \ graph \ calculating \ as \ follows:$ 

- $1. \ Read \ the \ graph \ data \ of \ Data Frame \ from \ the \ Nebula Graph \ database \ using \ the \ Nebula Graph \ Spark \ Connector.$
- 2. Transform the graph data of DataFrame to the GraphX graph.
- 3. Use graph algorithms provided by GraphX (such as PageRank) or self-implemented algorithms (such as Louvain).

For detailed implementation methods, see Scala file.

## 20.2.6 Get NebulaGraph Algorithm

#### Compile and package

1. Clone the repository nebula-algorithm.

\$ git clone -b v3.0.0 https://github.com/vesoft-inc/nebula-algorithm.git

2. Enter the directory nebula-algorithm .

\$ cd nebula-algorithm

3. Compile and package.

\$ mvn clean package -Dgpg.skip -Dmaven.javadoc.skip=true -Dmaven.test.skip=true

 $After the \ compilation, \ a \ similar \ file \ \ nebula-algorithm-3.x.x. jar \ is \ generated \ in \ the \ directory \ nebula-algorithm/target.$ 

#### Download maven from the remote repository

Download address

- 995/1069 - 2023 Vesoft Inc.

#### 20.2.7 How to use

#### Use algorithm interface (recommended)

The lib repository provides 10 common graph algorithms.

1. Add dependencies to the file pom.xml.

```
<dependency>
     <groupId>com.vesoft</groupId>
     <artifactId>nebula-algorithm</artifactId>
     <version>3.0.0</version>
</dependency>
```

2. Use the algorithm (take PageRank as an example) by filling in parameters. For more examples, see example.



By default, the DataFrame that executes the algorithm sets the first column as the starting vertex, the second column as the destination vertex, and the third column as the edge weights (not the rank in the NebulaGraph).

```
val prConfig = new PRConfig(5, 1.0)
val louvainResult = PageRankAlgo.apply(spark, data, prConfig, false)
```

If your vertex IDs are Strings, see Pagerank Example for how to encoding and decoding them.

#### Submit the algorithm package directly

1. Set the Configuration file.

```
# Configurations related to Spark
spark:
  app: {
      name: LPA
      # The number of partitions of Spark
      partitionNum:100
  master:local
data: {
  # Data source. Optional values are nebula, csv, and json.
  source: csv
   # Data sink. The algorithm result will be written into this sink. Optional values are nebula, csv, and text.
  sink: nebula
# Whether the algorithm has a weight.
  hasWeight: false
# Configurations related to NebulaGraph
nebula:
  # Data source. When NebulaGraph is the data source of the graph computing, the configuration of `nebula.read` is valid.
      # The IP addresses and ports of all Meta services. Multiple addresses are separated by commas (,). Example: "ip1:port1,ip2:port2".
       # To deploy NebulaGraph by using Docker Compose, fill in the port with which Docker Compose maps to the outside
      # Check the status with `docker-compose ps`
metaAddress: "192.168.*.10:9559"
      # The name of the graph space in NebulaGraph
      space: basketballplaver
        Edge types in NebulaGraph. When there are multiple labels, the data of multiple edges will be merged
      labels: ["serve"]
       # The property name of each edge type in NebulaGraph. This property will be used as the weight column of the algorithm. Make sure that it corresponds to the edge type.
      weightCols: ["start_year"]
  # Data sink. When the graph computing result sinks into NebulaGraph, the configuration of `nebula.write` is valid.
  write:{
      # The IP addresses and ports of all Graph services. Multiple addresses are separated by commas (,). Example: "ip1:port1,ip2:port2".
      # To deploy by using Docker Compose, fill in the port with which Docker Compose maps to the outside # Check the status with `docker-compose ps`.
      graphAddress: "192.168.*.11:9669"
      # The IP addresses and ports of all Meta services. Multiple addresses are separated by commas (,). Example: "ip1:port1,ip2:port2". # To deploy NebulaGraph by using Docker Compose, fill in the port with which Docker Compose maps to the outside.
       # Check the staus with `docker-compose ps`
      metaAddress: "192.168.*.12:9559"
       pswd:nebula
      # Before submitting the graph computing task, create the graph space and tag.
```

```
# The name of the graph space in NebulaGraph.
       space:nb
         The name of the tag in NebulaGraph. The graph computing result will be written into this tag. The property name of this tag is as follows
       # PageRank: pagerank
# Louvain: Louvain
       # ConnectedComponent: cc
       # StronglyConnectedComponent: scc
# LabelPropagation: lpa
        # ShortestPath: shortestpath
       # DegreeStatic: degree,inDegree,outDegree
       # TriangleCount: tranglecpunt
# BetweennessCentrality: betweennedss
       tag:pagerank
local: {
   # Data source. When the data source is csv or json, the configuration of `local.read` is valid.
  read:{
       filePath: "hdfs://127.0.0.1:9000/edge/work_for.csv"
       # If the CSV file has a header or it is a json file, use the header. If not, use [_c0, _c1, _c2, ..., _cn] instead.
       # The header of the source VID column.
       srcId:"_c0"
       # The header of the destination VID column
dstId:"_c1"
       # The header of the weight column.
       weight: "_c2"
# Whether the csv file has a header.
       header: false
       # The delimiter in the csv file.
delimiter:","
  # Data sink. When the graph computing result sinks to the csv or text file, the configuration of `local.write` is valid.
  write:{
       resultPath:/tmp/
algorithm: {
  # The algorithm to execute. Optional values are as follow:
  # pagerank, louvain, connectedcomponent, labelpropagation, shortestpaths,
# degreestatic, kcore, stronglyconnectedcomponent, trianglecount ,
# betweenness, graphtriangleCount.
  executeAlgo: pagerank
  # PageRank
  pagerank: {
   maxIter: 10
       resetProb: 0.15
  # Louvain
  louvain: {
       maxIter: 20
       internalIter: 10
       tol: 0.5
```

#### Q Note

When sink: nebula is configured, it means that the algorithm results will be written back to the NebulaGraph cluster. The property names of the tag have implicit conventions. For details, see **Supported algorithms** section of this topic.

## 2. Submit the graph computing task.

```
${SPARK_HOME}/bin/spark-submit --master <mode> --class com.vesoft.nebula.algorithm.Main <nebula-algorithm-3.0.0.jar_path> -p <application.conf_path>
```

#### Example:

\${SPARK\_HOME}/bin/spark-submit --master "local" --class com.vesoft.nebula.algorithm.Main /root/nebula-algorithm/target/nebula-algorithm-3.0-SNAPSHOT.jar -p /root/nebula-algorithm/src/main/resources/application.conf

Last update: February 19, 2024

# 20.3 NebulaGraph Analytics

NebulaGraph Analytics is a high-performance graph computing framework tool that performs graph analysis of data in the NebulaGraph database.

## 20.3.1 Prerequisites

- The NebulaGraph Analytics installation package has been obtained. Contact us to apply.
- The license is ready.
- The HDFS 2.2.x or later has been deployed.
- The JDK 1.8 has been deployed.

#### 20.3.2 Scenarios

You can import data from data sources as NebulaGraph clusters, CSV files on HDFS, or local CSV files into NebulaGraph Analytics and export the graph computation results to NebulaGraph clusters, CSV files on HDFS, or local CSV files from NebulaGraph Analytics.

#### 20.3.3 Limitations

When you import NebulaGraph cluster data into NebulaGraph Analytics and export the graph computation results from NebulaGraph Analytics to a NebulaGraph cluster, the graph computation results can only be exported to the graph space where the data source is located.

## 20.3.4 Version compatibility

The version correspondence between NebulaGraph Analytics and NebulaGraph is as follows.

NebulaGraph	<b>NebulaGraph Analytics</b>
3.4.0 ~ 3.4.1	3.4.0
3.3.0	3.3.0
3.1.0 ~ 3.2.x	3.2.0
3.0.x	1.0.x
2.6.x	0.9.0

- 998/1069 - 2023 Vesoft Inc.

## 20.3.5 Graph algorithms

NebulaGraph Analytics supports the following graph algorithms.

Algorithm	Description	Category
APSP	All Pair Shortest Path	Path
SSSP	Single Source Shortest Path	Path
BFS	Breadth-first search	Path
ShortestPath	The shortest path	Path
PageRank	It is used to rank web pages.	Node importance measurement
KCore	k-Cores	Node importance measurement
DegreeCentrality	It is a simple count of the total number of connections linked to a vertex.	Node importance measurement
DegreeWithTime	Neighbor statistics based on the time range of edge ranks	Node importance measurement
BetweennessCentrality	Intermediate centrality	Node importance measurement
ClosenessCentrality	Closeness centrality	Node importance measurement
TriangleCount	It counts the number of triangles.	Graph feature
Node2Vec	Graph neural network	Graph feature
Tree_stat	Tree structure statistics	Graph feature
HyperANF	Estimate the average distance of the graph	Graph feature
LPA	Label Propagation Algorithm	Community discovery
WCC	Weakly connected component	Community discovery
LOUVAIN	It detects communities in large networks.	Community discovery
InfoMap	Community classification	Community discovery
HANP	Hop attenuation & Node Preference	Community discovery
Clustering Coefficient	It is a measure of the degree to which nodes in a graph tend to cluster together.	Clustering
Jaccard	Jaccard similarity	Similarity

## 20.3.6 Install NebulaGraph Analytics

1. Install the NebulaGraph Analytics. When installing a cluster of multiple NebulaGraph Analytics on multiple nodes, you need to install NebulaGraph Analytics to the same path and set up SSH-free login between nodes.

```
sudo rpm -ivh <analytics_package_name> --prefix <install_path>
sudo chown <user>:<user> -R <install path>
```

## For example:

 $sudo\ rpm\ -ivh\ nebula-analytics -3.4.0-centos.x86\_64.rpm\ --prefix=/home/vesoft/nebula-analytics sudo\ chown\ vesoft:vesoft\ -R\ /home/vesoft/nebula-analytics$ 

- 999/1069 - 2023 Vesoft Inc.

2. Configure the correct Hadoop path and JDK path in the file set\_env.sh, the file path is nebula-analytics/scripts/set\_env.sh. If there are multiple machines, ensure that the paths are the same.



The default TCP port range used by the MPICH process manager and MPICH library is 10000 to 10100. To adjust this, modify the value of the environment variable MPIR\_CVAR\_CH3\_PORT\_RANGE in the set\_env.sh file.

export HADOOP\_HOME=<hadoop\_path>
export JAVA\_HOME=<java\_path>

3. Copy the license into the directory scripts of the NebulaGraph Analytics installation path on all machines.

# 20.3.7 How to use NebulaGraph Analytics

After installation, you can set parameters of different algorithms and then execute a script to obtain the results of the algorithms and export them to the specified format.

1. Select one node from the NebulaGraph Analytics cluster and then access the scripts directory.

\$ cd scripts

- 2. Confirm the data source and export path. Configuration steps are as follows.
- · NebulaGraph clusters as the data source
- a. Modify the configuration file nebula.conf to configure the NebulaGraph cluster.

```
# The number of retries connecting to NebulaGraph.
--retry=3
# The name of the graph space where you read or write data.
--space=baskeyballplayer
# Read data from NebulaGraph.
--edges=LIKES
# The name of the property to be read as the weight of the edge. Can be either the attribute name or _rank.
#--edge_data_fields
# The number of rows read per scan.
--read_batch_size=10000
# Write data to NebulaGraph
# The graphd process addres
--graph_server_addrs=192.168.8.100:9669
# The account to log into NebulaGraph.
 -user=root
# The password to log into NebulaGraph.
--password=nebula
# The pattern used to write data back to NebulaGraph: insert or update.
--mode=insert
# The tag name written back to NebulaGraph.
--tag=pagerank
# The property name corresponding to the tag.
# The property type corresponding the the tag.
--type=double
# The number of rows per write
--write batch size=1000
# The file path where the data failed to be written back to NebulaGraph is stored.
--err_file=/home/xxx/analytics/err.txt
# The access timeout period of the service.
--graphd_timeout=60000
--metad_timeout=60000
--storaged_timeout=60000
```

b. Modify the related parameters in the script to be used, such as  $\mbox{run\_pagerank.sh}$ .

```
# The sum of the number of processes running on all machines in the cluster. It is recommended to be the number of machines or the number of nodes in the NUMA architecture.
WNUM=3
# The number of threads per process. It is recommended to set the maximum value to be the number of hardware threads of the machine.
WCORFS=4
# The path to the data source.
# Set to read data from NebulaGraph via the nebula.conf file.
INPUT:=\Input:=\nebula:\PROJECT/scripts/nebula.conf\}
# Set to read data from the CSV files on HDFS or on local directories.
# #INPUT=${INPUT:="$PROJECT/data/graph/v100_e2150_ua_c3.csv"}
# The export path to the graph computation results
# Data can be exported to a NebulaGraph. If the data source is also a NebulaGraph, the results will be exported to the graph space specified in nebula.conf.
OUTPUT=${OUTPUT:="nebula:$PROJECT/scripts/nebula.conf"}}
# Data can also be exported to the CSV files on HDFS or on local directories.
# OUTPUT=${OUTPUT:='hdfs://192.168.8.100:9000/_test/output'}
# If the value is true, it is a directed graph, if false, it is an undirected graph.
IS_DIRECTED=${IS_DIRECTED:=true}
# Set whether to encode ID or not.
NEED_ENCODE=${NEED_ENCODE:=true}
# The ID type of the data source vertices. For example string, int32, and int64.
VTYPE=${VTYPE:=int32}
# Encoding type. The value distributed specifies the distributed vertex ID encoding. The value single specifies the single-machine vertex ID encoding. ENCODER:="distributed"}
# The parameter for the PageRank algorithm. Algorithms differ in parameters. EPS=${EPS:=0.0001}
DAMPING=${DAMPING:=0.85}
# The number of iterations.
ITERATIONS=${ITERATIONS:=100}
```

. Local or HDFS CSV files as the data source

Modify parameters in the script to be used, such as  $\mbox{run\_pagerank.sh}$ .

```
# The sum of the number of processes running on all machines in the cluster. It is recommended to be the number of machines or the number of nodes in the NUMA architecture.
# The number of threads per process. It is recommended to set the maximum value to be the number of hardware threads of the machine.
WCORES=4
# The path to the data source
# Set to read data from NebulaGraph via the nebula.conf file.
# INPUT=${INPUT:="nebula:$PROJECT/scripts/nebula.conf"}
INPUT=${INPUT:="$PROJECT/data/graph/v100_e2150_ua_c3.csv"}
# The export path to the graph computation results
# Data can be exported to a NebulaGraph. If the data source is also a NebulaGraph, the results will be exported to the graph space specified in nebula.conf. # OUTPUT=${OUTPUT:="nebula:$PROJECT/scripts/nebula.conf"}
# Data can also be exported to the CSV files on HDFS or on local directories.
OUTPUT=${OUTPUT:='hdfs://192.168.8.100:9000/ test/output'}
# If the value is true, it is a directed graph, if false, it is an undirected graph.
IS_DIRECTED=${IS_DIRECTED:=true}
# Set whether to encode ID or not
NEED ENCODE=${NEED ENCODE:=true}
# The ID type of the data source vertices. For example string, int32, and int64.
VTYPE=${VTYPE:=int32}
# The value distributed specifies the distributed vertex ID encoding. The value single specifies the single-machine vertex ID encoding.
ENCODER=${ENCODER:="distributed"}
\# The parameter for the PageRank algorithm. Algorithms differ in parameters. 
 EPS=${EPS:=0.0001}
DAMPING=${DAMPING:=0.85}
# The number of iterations
ITERATIONS=${ITERATIONS:=100}
```

3. Modify the configuration file cluster to set the NebulaGraph Analytics cluster nodes and task assignment weights for executing the algorithm.

```
# NebulaGraph Analytics Cluster Node IP Addresses: Task Assignment Weights
192.168.8.200:1
192.168.8.201:1
192.168.8.202:1
```

4. Run the algorithm script. For example:

```
./run_pagerank.sh
```

- 5. View the graph computation results in the export path.
- $\bullet \ \ For \ exporting \ to \ a \ Nebula Graph \ cluster, \ check \ the \ results \ according \ to \ the \ settings \ in \ \ nebula.conf.$
- $\bullet$  For exporting the results to the CSV files on HDFS or on local directories, check the results according to the settings in output, which is a compressed file in the <code>.gz</code> format.

Last update: February 19, 2024

# 20.4 NebulaGraph Analytics license

A license is a software authorization certificate used to authorize the use of a software product. When deploying NebulaGraph Analytics, you need to add a license to start it. This document describes the license information on NebulaGraph Analytics.

## 20.4.1 Precautions

- If the license file is not deployed, NebulaGraph Analytics cannot be started.
- Do not modify the license file, otherwise the license will become invalid.
- If the license is about to expire, contact us to apply for renewal.
- The transition period after the license expires is 14 days:
- If you start NebulaGraph Analytics within 30 days before the license expires or on the day the license expires, a log will be printed as a reminder.
- The license can still be used for 14 days after it expires.
- If the license has expired for 14 days, you will not be able to start the NebulaGraph Analytics, and a log will be printed as a reminder.

## 20.4.2 Obtain a NebulaGraph Analytics license

Contact us to apply for a NebulaGraph Analytics license.



You can apply online for a 30-day free trial of NebulaGraph Analytics.

## 20.4.3 License description

NebulaGraph Analytics license is a file named nebula. License that contains the following information:

The license file contains information such as issuedDate and expirationDate. The description is as follows.

Parameter	Description
vendor	The supplier.
organization	The username.
issuedDate	The date that the license is issued.
expirationDate	The date that the license expires.
product	The product type. The product type of NebulaGraph Analytics is $\mbox{nebula\_graph\_analytics}$ .
version	The version information.
LicenseType	The license type (a reserved parameter), including enterprise, samll_bussiness, pro, and individual.
gracePeriod	The buffer time (in days) for the service to continue to be used after the license expires, and the service will be stopped after the buffer period. The trial version of license has no buffer period after expiration and the default value of this parameter is 0.
nodes	The max number of Analytics services in the cluster.
vcpu	The max number of threads for the Analytics services in the cluster.
clusterCode	The user's hardware information, which is also the unique identifier of the cluster. This parameter is not available in the trial version of the license.

## 20.4.4 Use a NebulaGraph Analytics license

For how to use a NebulaGraph Analytics license, see NebulaGraph Analytics.

# 20.4.5 Renew a NebulaGraph Analytics license

Follow the steps below to renew your NebulaGraph Analytics license.

- 1. Contact us to apply for a new NebulaGraph Analytics license file nebula. License.
- 2. In the NebulaGraph Analytics installation directory, such as /usr/local/nebula-analytics/scripts/, replace the old license file with the new one.



You cannot use NebulaGraph Analytics once the license expires. To avoid business interruptions, please renew your license in time.

Last update: February 19, 2024

# 20.5 NebulaGraph Explorer Workflow

 $Nebula Graph\ Explorer\ provides\ workflows\ for\ visual\ calculations.$ 

For more details, see Workflows.



To apply for the NebulaGraph Explorer installation package, contact us.

Last update: February 19, 2024

# 21. NebulaGraph Spark Connector

NebulaGraph Spark Connector is a Spark connector application for reading and writing NebulaGraph data in Spark standard format. NebulaGraph Spark Connector consists of two parts: Reader and Writer.

#### • Reader

Provides a Spark SQL interface. This interface can be used to read NebulaGraph data. It reads one vertex or edge type data at a time and assemble the result into a Spark DataFrame.

#### • Writer

Provides a Spark SQL interface. This interface can be used to write DataFrames into NebulaGraph in a row-by-row or batch-import way.

For more information, see NebulaGraph Spark Connector.

# 21.1 Version compatibility

The correspondence between the NebulaGraph Spark Connector version, the NebulaGraph core version and the Spark version is as follows.

Spark Connector version	NebulaGraph version	Spark version
$nebula-spark-connector\_3.0-3.0-SNAPSHOT.jar$	nightly	3.x
$nebula-spark-connector\_2.2-3.0-SNAPSHOT.jar$	nightly	2.2.x
nebula-spark-connector-3.0-SNAPSHOT.jar	nightly	2.4.x
nebula-spark-connector_2.2-3.4.0.jar	3.x	2.2.x
nebula-spark-connector-3.4.0.jar	3.x	2.4.x
nebula-spark-connector_2.2-3.3.0.jar	3.x	2.2.x
nebula-spark-connector-3.3.0.jar	3.x	2.4.x
nebula-spark-connector-3.0.0.jar	3.x	2.4.x
nebula-spark-connector-2.6.1.jar	2.6.0, 2.6.1	2.4.x
nebula-spark-connector-2.6.0.jar	2.6.0, 2.6.1	2.4.x
nebula-spark-connector-2.5.1.jar	2.5.0, 2.5.1	2.4.x
nebula-spark-connector-2.5.0.jar	2.5.0, 2.5.1	2.4.x
nebula-spark-connector-2.1.0.jar	2.0.0, 2.0.1	2.4.x
nebula-spark-connector-2.0.1.jar	2.0.0, 2.0.1	2.4.x
nebula-spark-connector-2.0.0.jar	2.0.0, 2.0.1	2.4.x

- 1008/1069 - 2023 Vesoft Inc.

## 21.2 Use cases

NebulaGraph Spark Connector applies to the following scenarios:

- Migrate data between different NebulaGraph clusters.
- Migrate data between different graph spaces in the same NebulaGraph cluster.
- Migrate data between NebulaGraph and other data sources.
- Graph computing with NebulaGraph Algorithm.

## 21.3 Benefits

The features of NebulaGraph Spark Connector 3.4.0 are as follows:

- Supports multiple connection settings, such as timeout period, number of connection retries, number of execution retries, etc.
- Supports multiple settings for data writing, such as setting the corresponding column as vertex ID, starting vertex ID, destination vertex ID or attributes.
- Supports non-attribute reading and full attribute reading.
- Supports reading NebulaGraph data into VertexRDD and EdgeRDD, and supports non-Long vertex IDs.
- Unifies the extended data source of SparkSQL, and uses DataSourceV2 to extend NebulaGraph data.
- Three write modes, insert, update and delete, are supported. insert mode will insert (overwrite) data, update mode will only update existing data, and delete mode will only delete data.

#### 21.4 Release note

Release

## 21.5 Get NebulaGraph Spark Connector

## 21.5.1 Compile package

1. Clone repository nebula-spark-connector.

\$ git clone -b release-3.4 https://github.com/vesoft-inc/nebula-spark-connector.git

- $2. \ Enter \ the \ \textit{nebula-spark-connector} \ directory.$
- $2.\ \mbox{Compile}$  package. The procedure varies with Spark versions.



Spark of the corresponding version has been installed.

- Spark 2.4

```
```bash
$ mvn clean package -Dmaven.test.skip=true -Dgpg.skip -Dmaven.javadoc.skip=true -pl nebula-spark-connector -am -Pscala-2.11 -Pspark-2.4
...
```

- Spark 2.2

```
```bash
$ mvn clean package -Dmaven.test.skip=true -Dgpg.skip -Dmaven.javadoc.skip=true -pl nebula-spark-connector_2.2 -am -Pscala-2.11 -Pspark-2.2
```

#### - Spark 3.x

```
```bash
$ mvn clean package -Dmaven.test.skip=true -Dgpg.skip -Dmaven.javadoc.skip=true -pl nebula-spark-connector_3.0 -am -Pscala-2.12 -Pspark-3.0
```

After compilation, a similar file nebula-spark-connector-3.4.0-SHANPSHOT.jar is generated in the directory target of the folder.

## 21.5.2 Download maven remote repository

Download

## 21.6 How to use

When using NebulaGraph Spark Connector to reading and writing NebulaGraph data, You can refer to the following code.

```
# Read vertex and edge data from NebulaGraph.
spark.read.nebula().loadVerticesToDF()
spark.read.nebula().loadEdgesToDF()

# Write dataframe data into NebulaGraph as vertex and edges.
dataframe.write.nebula().writeVertices()
dataframe.write.nebula().writeEdges()
```

nebula() receives two configuration parameters, including connection configuration and read-write configuration.

#### 21.6.1 Reading data from NebulaGraph

```
val config = NebulaConnectionConfig
  .builder()
.withMetaAddress("127.0.0.1:9559")
   . \verb|withConenctionRetry(2)| \\
  .withExecuteRetry(2)
.withTimeout(6000)
   .build()
val nebulaReadVertexConfig: ReadNebulaConfig = ReadNebulaConfig
   .builder()
  .withSpace("test")
  .withLabel("person")
.withNoColumn(false)
.withReturnCols(List("birthday"))
   .withLimit(10)
   .withPartitionNum(10)
val vertex = spark.read.nebula(config, nebulaReadVertexConfig).loadVerticesToDF()
val nebulaReadEdgeConfig: ReadNebulaConfig = ReadNebulaConfig
  .builder()
.withSpace("test")
   .withLabel("knows
  .withNoColumn(false)
.withReturnCols(List("degree"))
   .withLimit(10)
  . with \texttt{PartitionNum}(\textcolor{red}{\textbf{10}})
```

```
.build()
val edge = spark.read.nebula(config, nebulaReadEdgeConfig).loadEdgesToDF()
```

• NebulaConnectionConfig is the configuration for connecting to the nebula graph, as described below.

Parameter	Required	Description	
withMetaAddress	Yes Specifies the IP addresses and ports of all Meta Services. Separate multiple addresses with commas. The format is $ip1:port1, ip2:port2, \ldots$ . Read data is no ne to configure with Graph Address.		
withConnectionRetry	No	The number of retries that the NebulaGraph Java Client connected to the NebulaGraph. The default value is $^{1}$ .	
withExecuteRetry	No	The number of retries that the NebulaGraph Java Client executed query statements. The default value is 1.	
withTimeout	No	The timeout for the NebulaGraph Java Client request response. The default value is 6000 , Unit: ms.	

 $\bullet \ \ {\tt ReadNebulaConfig} \ \ is the \ configuration \ to \ read \ Nebula Graph \ data, \ as \ described \ below.$ 

Parameter	Required	Description	
withSpace	Yes	NebulaGraph space name.	
withLabel	Yes	The Tag or Edge type name within the NebulaGraph space.	
withNoColumn	No	Whether the property is not read. The default value is false, read property. If the value is true, the property is not read, the withReturnCols configuration is invalid.	
withReturnCols	No	Configures the set of properties for vertex or edges to read. the format is $ \mbox{List(property1,property2,)} \ , \mbox{The default value is } \mbox{List()} \ , \mbox{indicating that all properties are read.} $	
withLimit	No	Configure the number of rows of data read from the server by the NebulaGraph Java Storage Client at a time. The default value is 1000.	
withPartitionNum	No	Configures the number of Spark partitions to read the NebulaGraph data. The default value is 100. This value should not exceed the number of slices in the graph space (partition_num).	

## 21.6.2 Write data into NebulaGraph



The values of columns in a dataframe are automatically written to the NebulaGraph as property values.

```
val config = NebulaConnectionConfig
.builder()
.withMetaAddress("127.0.0.1:9559")
.withConenctionRetry(2)
.build()

val nebulaWriteVertexConfig: WriteNebulaVertexConfig = WriteNebulaVertexConfig
.builder()
.withTag("person")
.withTag("person")
.withTag("person")
.withVidPolicy("hash")
.withVidPolicy("hash")
.withVidAsProp(true)
.withVidAsProp(true)
.withWasSProp(true)
.withWasSProp(true)
.withBatch(1000)
.build()
df.write.nebula(config, nebulaWriteVertexConfig).writeVertices()
val nebulaWriteEdgeConfig: WriteNebulaEdgeConfig = WriteNebulaEdgeConfig
.builder()
```

```
.withSpace("test")
.withEdge("friend")
.withSrcTdField("src")
.withSrcTdField("dst")
.withDstTdField("dst")
.withDstTdField("dst")
.withDstPolicy(null)
.withRankField("degree")
.withSrcAsProperty(true)
.withSrcAsProperty(true)
.withDstAsProperty(true)
.withDstAsProperty(true)
.withUser("root")
.withBrack("loot")
.withBrack("loot")
.withBrack("loot")
.withBrack(1000)
.build()
df.write.nebula(config, nebulaWriteEdgeConfig).writeEdges()
```

The default write mode is insert, which can be changed to update or delete via withWriteMode configuration:

```
val config = NebulaConnectionConfig
.builder()
.withMetaAddress("127.0.0.1:9559")
.withGraphAddress("127.0.0.1:9669")
.build()
val nebulaWriteVertexConfig = WriteNebulaVertexConfig
.builder()
.withSpace("test")
.withTag("person")
.withVidField("id")
.withVidFop(true)
.wittNBatch(1000)
.wittNBatch(1000)
.wittNWriteMode(WriteMode.UPDATE)
```

# .build() df.write.nebula(config, nebulaWriteVertexConfig).writeVertices()

 $\bullet \ \ {\tt NebulaConnectionConfig} \ \ is \ the \ configuration \ for \ connecting \ to \ the \ nebula \ graph, \ as \ described \ below.$ 

Parameter	Required	Description
withMetaAddress	Yes	Specifies the IP addresses and ports of all Meta Services. Separate multiple addresses with commas. The format is $ip1:port1,ip2:port2,$
withGraphAddress	Yes	Specifies the IP addresses and ports of Graph Services. Separate multiple addresses with commas. The format is <code>ip1:port1,ip2:port2,</code> .
withConnectionRetry	No	Number of retries that the NebulaGraph Java Client connected to the NebulaGraph. The default value is 1.

 $\bullet \ \ {\tt WriteNebulaVertexConfig} \ is the \ configuration \ of the \ write \ vertex, \ as \ described \ below.$ 

Parameter	Required	Description	
withSpace	Yes	NebulaGraph space name.	
withTag	Yes	The Tag name that needs to be associated when a vertex is written.	
withVidField	Yes	The column in the DataFrame as the vertex ID.	
withVidPolicy	No	When writing the vertex ID, NebulaGraph use mapping function, supports HASH only. No mapping is performed by default.	
withVidAsProp	No	Whether the column in the DataFrame that is the vertex ID is also written as an property. The default value is $$ false . If set to $$ true , make sure the Tag has the same property name as $$ VidField .	
withUser	No	NebulaGraph user name. If authentication is disabled, you do not need to configure the user name and password.	
withPasswd	No	The password for the NebulaGraph user name.	
withBatch	Yes	The number of rows of data written at a time. The default value is $\ 1000 \ .$	
withWriteMode	No	Write mode. The optional values are $% \left( 1\right) =\left( 1\right) +\left( 1\right) =\left( 1\right) +\left( 1\right) +\left( 1\right) =\left( 1\right) +\left( 1\right)$	
withDeleteEdge	No	Whether to delete the related edges synchronously when deleting a vertex. The default value is false. It takes effect when withWriteMode is delete.	

 $_{\bullet}$  WriteNebulaEdgeConfig is the configuration of the write edge, as described below.

Parameter	Required	Description	
withSpace	Yes	NebulaGraph space name.	
withEdge	Yes	The Edge type name that needs to be associated when a edge is written.	
withSrcIdField	Yes	The column in the DataFrame as the vertex ID.	
withSrcPolicy	No	When writing the starting vertex ID, NebulaGraph use mapping function, supports HASH only. No mapping is performed by default.	
withDstIdField	Yes	The column in the DataFrame that serves as the destination vertex.	
withDstPolicy	No	When writing the destination vertex ID, NebulaGraph use mapping function, supports HASH only. No mapping is performed by default.	
withRankField	No	The column in the DataFrame as the rank. Rank is not written by default.	
withSrcAsProperty	No	Whether the column in the DataFrame that is the starting vertex is also written as an property. The default value is false. If set to true, make sure Edge type has the same property name as SrcIdField.	
withDstAsProperty	No	Whether column that are destination vertex in the DataFrame are also written as property. The default value is false. If set to true, make sure Edge type has the same property name as <code>DstIdField</code> .	
withRankAsProperty	No	Whether column in the DataFrame that is the rank is also written as property. The default value is false. If set to true, make sure Edge type has the same property name as $RankField$ .	
withUser	No	NebulaGraph user name. If authentication is disabled, you do not need to configure the user name and password.	
withPasswd	No	The password for the NebulaGraph user name.	
withBatch	Yes	The number of rows of data written at a time. The default value is $1000$ .	
withWriteMode	No	Write mode. The optional values are $% \left( 1\right) =\left( 1\right) +\left( 1\right) +\left( 1\right) =\left( 1\right) +\left( 1\right) +\left( 1\right) +\left( 1\right) =\left( 1\right) +\left( 1\right)$	

## 22. NebulaGraph Flink Connector

NebulaGraph Flink Connector is a connector that helps Flink users quickly access NebulaGraph. NebulaGraph Flink Connector supports reading data from the NebulaGraph database or writing other external data to the NebulaGraph database.

For more information, see NebulaGraph Flink Connector.

## 22.1 Use cases

NebulaGraph Flink Connector applies to the following scenarios:

- Migrate data between different NebulaGraph clusters.
- Migrate data between different graph spaces in the same NebulaGraph cluster.
- Migrate data between NebulaGraph and other data sources.

## 22.2 Release note

#### Release

Last update: February 19, 2024

- 1015/1069 - 2023 Vesoft Inc.

# 23. NebulaGraph Bench

 $Nebula Graph \ Bench \ is \ a \ performance \ test \ tool \ for \ Nebula Graph \ using \ the \ LDBC \ data \ set.$ 

## 23.1 Scenario

- Generate test data and import NebulaGraph.
- Performance testing in the NebulaGraph cluster.

## 23.2 Release note

Release

## 23.3 Test process

For detailed usage instructions, see NebulaGraph Bench.

## 24. Appendix

## 24.1 Release Note

## 24.1.1 NebulaGraph 3.4.1 release notes

## Bugfix

- $\bullet$  Fix the crash caused by encoding parameter expressions to the storage layer for execution. #5336
- Fix some crashes for the list function. #5383

## Legacy versions

Release notes of legacy versions

## 24.1.2 NebulaGraph 3.4.1 release notes

## Bugfix

- $\bullet$  Fix the crash caused by encoding the parameter expression and evaluating it in the storage.
- Fix some crashes for the list function.

## Legacy versions

Release notes of legacy versions

## 24.1.3 NebulaGraph Studio release notes

#### v3.6.0

- Feature
- Support viewing the creation statements of the schema.
- Add a product feedback page.
- Enhancement
- Remove the timeout limit for slow queries.
- Display browser compatibility hints.
- Optimize the login page.
- Support adding comments with # on the console page.
- Optimize the console page.
- Bugfix
- Fix the bug that the list has not been refreshed after uploading files.
- Fix the invalid error message of the schema drafting.
- Fix the bug that the **view schema** data has not been cleared after switching the login user.
- Fix the presentation problem of the thumbnail in the schema drafting.

## 24.1.4 NebulaGraph Dashboard Community Edition 3.4.1 release notes

## Community Edition 3.4.0

- Feature
- Support the built-in dashboard.service script to manage the Dashboard services with one-click and view the Dashboard version.
- Support viewing the configuration of Meta services.
- Enhancement
- Adjust the directory structure and simplify the deployment steps.
- Display the names of the monitoring metrics on the overview page of machine.
- $\bullet \ \, \text{Optimize the calculation of monitoring metrics such as } \ \, \text{num\_queries} \ , \ \text{and adjust the display to time series aggregation}. \\$

## 24.1.5 NebulaGraph Dashboard Enterprise Edition release notes

## **Enterprise Edition 3.4.2**

- Enhancement
- ullet Support viewing the data backup and restoration progress on the  ${f Backup\&Restore}$  page.
- The installation package for NebulaGraph Enterprise v3.4.1 is built in.

## **Enterprise Edition 3.4.1**

- Bugfix
- $\bullet$  Fix the bug that the RPM package cannot execute  $\ensuremath{\mathsf{nebula-agent}}$  due to permission issues.
- Fix the bug that the cluster import information can not be viewed due to the <code>goconfig</code> folder permission.
- ullet Fix the page error when the license expiration time is less than 30 days and <code>gracePeriod</code> is greater than 0.

#### **Enterprise Edition 3.4.0**

- Feature
- Support viewing the runtime log of the NebulaGraph clusters.
- Support viewing the audit log of the NebulaGraph clusters.
- Support jog management.
- Support incremental backup for Backup & Restore (BR) tool.
- Support the built-in dashboard.service script to manage the Dashboard services with one-click and view the Dashboard version.
- Add a product feedback page.
- Enhancement
- Automatically detects whether the installation package is compatible with the operating system when creating a cluster.
- Support specifying the NebulaGraph installation directory when importing nodes in batches.
- Support deleting the installation directory when deleting a cluster.
- Dependent services are displayed in the importing cluster and service monitoring.
- Support canceling the alert rule silence midway.
- Support killing the Graph service processes forcibly.
- $\bullet$  Support viewing and modifying configuration information of multiple services.
- Support modifying the configuration of the Meta service.
- Support logging update configuration and delete backup operations on operation record page.
- Support auto-registration after LDAP is enabled.
- Detail Log information of the task center.
- Display browser compatibility hint.
- NebulaGraph license expiration reminder.
- Support for Red Flag OS Asianux Linux 7 (Core).
- Optimize multiple interactions such as connecting to the database, creating a cluster, scaling and batch node importing.
- $\bullet$  Optimize the interface error message.
- $\bullet$  Display the names of the monitoring metrics on the overview page of  $\ensuremath{\mathsf{node}}$  .
- $\bullet \ \ \text{Optimize the calculation of monitoring metrics such as } \ \ \text{num\_queries} \ , \ \text{and adjust the display to time series aggregation}.$
- Bugfix
- Fix the bug that the selection of monitoring time range does not take effect in the overview page of service monitoring.
- Fix the bug that the corresponding NebulaGraph file is not deleted when deleting empty nodes during scale-in reduction.
- Fix the bug that the global language is switched at the same time when switching the language of the diagnosis report.
- Fix the bug that an import cluster task blocks and causes other import tasks to be in waiting state.

## 24.1.6 NebulaGraph Explorer release notes

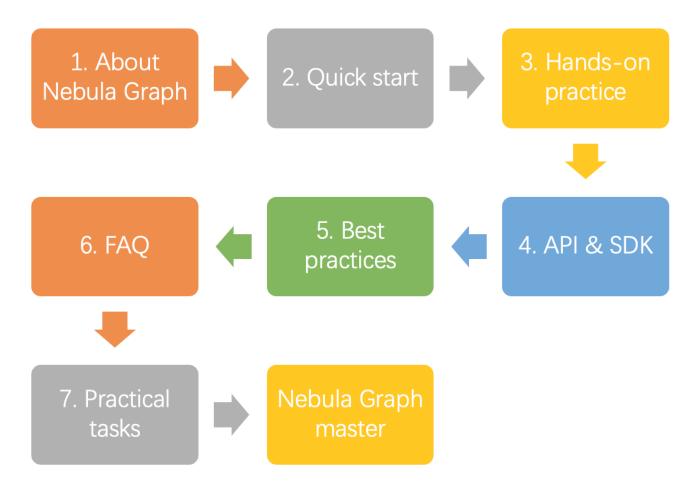
#### v3.4.0

- Feature
- Support viewing the creation statements of the schema.
- Add a **Beta functions** switch button on the global settings page.
- · Add a product feedback page.
- Enhancement
- Remove the timeout limit for slow queries.
- Keep history on the console page after switching pages.
- Support adding comments with # on the console page.
- Support adding comments with # or // when creating nGQL templates.
- Update the global settings page.
- Support the visual modification of the IP whitelist.
- Show VID on canvas by default.
- Display browser compatibility hints.
- Show the kernel version in the connection information.
- Add indexes to the built-in dataset.
- Optimize the login page.
- Optimize Workflow:
- Add algorithm descriptions.
- Optimize the parameter configurations of the graph algorithm.
- Optimize the presentation of the result.
- Optimize interactions:
- Vertex filter
- Query by tag
- Search path
- Optimize presentations:
- $\bullet$  Optimize the presentation of schema statistics.
- Optimize the layout of force.
- Optimize the layout of the visual query results after importing them to the canvas.
- $\bullet$  Optimize the presentation of vertices on dangling edges.
- Optimize the console page.
- Optimize hints:
- Optimize guidances.
- Optimize error messages.

- Bugfix
- Fix the bug that can not be able to view the import task log.
- ullet Fix the bug that some data of the edges in the  ${\tt demo\_basketballplayer}$  dataset is missing.
- Fix the crash of the page.
- Fix the bug that the results of the graph algorithm in the workflow can not show the details of vertices after importing them to canvas.

## 24.2 NebulaGraph learning path

This topic is for anyone interested in learning more about NebulaGraph. You can master NebulaGraph from zero to hero through the documentation and videos in NebulaGraph learning path.



After completing the NebulaGraph learning path, taking NebulaGraph Certification exams will earn you certifications. For more information, see the **Get NebulaGraph Certifications** section below.

## 24.2.1 1. About NebulaGraph

## 1.1 What is NebulaGraph?

Document Video

What is NebulaGraph NebulaGraph

#### 1.2 Data models

#### **Document**

Data modeling

- 1025/1069 - 2023 Vesoft Inc.

#### 1.3 Path

#### Document

Path

## 1.4 NebulaGraph architecture

## Document

Meta service

Graph service

Storage service

## 24.2.2 2. Quick start

## 2.1 Install NebulaGraph

Document	Video
Install with a RPM or DEB package	-
Install with a TAR package	-
Install with Docker	Install NebulaGraph with Docker and Docker Compose
Install from source	Install NebulaGraph with Source Code

## 2.2 Start NebulaGraph

#### Document

Start and stop NebulaGraph

## 2.3 Connect to NebulaGraph

#### Document

Connect to NebulaGraph

## 2.4 Use nGQL statements

#### Document

nGQL cheatsheet

## 24.2.3 3. Hands-on practices

## 3.1 Deploy a multi-machine cluster

## Document

Deploy a NebulaGraph cluster with RPM/DEB

## 3.2 Upgrade NebulaGraph

## Document

Upgrade NebulaGraph to release-3.4

## 3.3 Configure NebulaGraph

## Document

Configure Meta

Configure Graph

Configure Storage

Configure Linux kernel

## 3.4 Configure logs

#### Document

Log managements

## 3.5 O&M and Management

• Account authentication and authorization

#### **Document**

Local authentication

OpenLDAP

User management

Roles and privileges

• Balance the distribution of partitions

#### **Document**

Storage load balancing

• Monitoring

## **Document**

NebulaGraph metrics

RocksDB statistics

• Data snapshot

#### Document

Create snapshots

• Backup & Restore

#### **Document**

Backup&Restore

• SSL encryption

## **Document**

SSL

## 3.6 Performance tuning

## **Document**

Graph data modeling suggestions

System design suggestions

Compaction

## 3.7 Derivative software

## • Visualization

Visualization tools	Document	Video
Data visualization	NebulaGraph Studio	NebulaGraph Studio
Data monitoring and O&M	NebulaGraph Dashboard Community Edition NebulaGraph Dashboard Enterprise Edition	-
Data analysis	NebulaGraph Explorer Enterprise Edition	-

## • Data import and export

Import and export Document		Video	
Data import	NebulaGraph Importer	NebulaGraph Importer	
Data import	NebulaGraph Spark Connector	-	
Data import	NebulaGraph Flink Connector	-	
Data import	NebulaGraph Exchange Community Edition	-	
Data export	NebulaGraph Exchange Enterprise Edition	-	

## • Performance test

## Document

NebulaGraph Bench

• Cluster O&M

## Document

NebulaGraph Operator

• Graph algorithm

## Document

 ${\bf Nebula Graph\ Algorithm}$ 

• Clients

#### **Document**

NebulaGraph Console

NebulaGraph CPP

NebulaGraph Java

NebulaGraph Python

NebulaGraph Go

## 24.2.4 4. API & SDK

#### Document

API & SDK

#### 24.2.5 5. Best practices

#### Document

Handling Tens of Billions of Threat Intelligence Data with Graph Database at Kuaishou

Import data from Neo4j to NebulaGraph via NebulaGraph Exchange: Best Practices

Hands-On Experience: Import Data to NebulaGraph with Spark

How to Select a Graph Database: Best Practices at RoyalFlush

Practicing NebulaGraph Operator on Cloud

Using Ansible to Automate Deployment of NebulaGraph Cluster

#### 24.2.6 6. FAO

#### **Document**

FAQ

#### 24.2.7 7. Practical tasks

You can check if you have mastered NebulaGraph by completing the following practical tasks.

Task	Reference
Compile the source code of NebulaGraph	Install NebulaGraph by compiling the source code
Deploy Studio, Dashboard, and Explorer	Deploy Studio, Deploy Dashboard, and Deploy Explorer
Load test NebulaGraph with K6	NebulaGraph Bench
Query LDBC data (such as queries for vertices, paths, or subgraphs.)	LDBC and interactive-short-1.cypher

## 24.2.8 8. Get NebulaGraph Certifications

Now you could get NebulaGraph Certifications from NebulaGraph Academy.

- NebulaGraph Certified Insider(NGCI): The NGCI certification provides a birdview to graph databases and the NebulaGraph database. Passing NGCI shows that you have a good understanding of NebulaGraph.
- NebulaGraph Certified Professional(NGCP): The NGCP certification drives you deep into the NebulaGraph database and its ecosystem, providing a 360-degree view of the leading-edge graph database. Passing NGCP proves that you are a professional with a profound understanding of NebulaGraph.

## 24.2.9 Reference documents

- For an introduction to the principles of NebulaGraph, see Nebula Graph: An open source distributed graph database.
- For the principle description of NebulaGraph indexes, see Section 2.4 in the Nebula Graph: An open source distributed graph database paper.
- For an overview of the NebulaGraph language, see Section 2.8 in the Nebula Graph: An open source distributed graph database paper.

## 24.3 About NebulaGraph licenses

A license is an official permission or permit that defines and restricts the rights of users to use the software. NebulaGraph licenses are used to restrict the use of the Enterprise Edition software. This document describes the NebulaGraph products that require a license and information about the license.

#### 24.3.1 NebulaGraph Enterprise Edition

NebulaGraph Enterprise Edition is scalable software with high availability and high performance.

A NebulaGraph Enterprise Edition license is required to deploy a NebulaGraph Enterprise Edition cluster.

For more information, see Deploy a license for NebulaGraph Enterprise Edition.

#### 24.3.2 NebulaGraph Dashboard Enterprise Edition

NebulaGraph Dashboard Enterprise Edition is an out-of-the-box visualization tool that monitors and manages the status of cluster machines and services.

A NebulaGraph Dashboard Enterprise Edition license is required to deploy NebulaGraph Dashboard Enterprise Edition. For more information, see NebulaGraph Dashboard Enterprise Edition license.

#### 24.3.3 NebulaGraph Explorer

NebulaGraph Explorer, which has the Enterprise Edition only, is a browser-based visualization tool. It is used with the NebulaGraph core to visualize interaction with graph data. Even if there is no experience in graph databases, you can quickly become a graph exploration expert.

A NebulaGraph Explorer license is required to deploy NebulaGraph Explorer. For more information, see NebulaGraph Explorer license.

#### 24.3.4 FAQ about NebulaGraph licenses

#### Are the Dashboard/Explorer/NebulaGraph Enterprise Edition licenses the same?

No, the licenses of Dashboard, Explorer, and NebulaGraph Enterprise Editions are independent of each other and cannot be used interchangeably.

During the validity period of the NebulaGraph Enterprise Edition license, after replacing the enterprise edition Meta with the community edition Meta, can the community edition Meta be used with the enterprise edition Graph and Storage?

No, mixed deployments of the enterprise edition services and the community edition services are not supported.

After the NebulaGraph Enterprise Edition license expires, is it possible that copy the data in the data directory and paste it to the same directory of NebulaGraph Community Edition, and then use NebulaGraph services as normal?

Yes, it is possible. The data of the Enterprise Edition can be used in the Community Edition. The pasted data will only work properly in the services deployed in the Community Edition. Mixed deployments of the enterprise edition services and the community edition services are not supported. For example, the mixed deployment of the enterprise edition Meta service and the community edition Graph and Storage services is not supported.

## Is there any message before the license expires, and how to renew the license after it expires?

The system will send expiration notifications before the license expires.

The notification time before the license expires is different for the full version license and the trial version license.

- For the full version license:
- Within 30 days before the license expires or on the day the license expires, there is an expiration reminder when NebulaGraph/Dashboard/Explorer is started.
- There is a 14-day buffer period after expiration. During the buffer period, you will receive expiration notifications and can continue using NebulaGraph/Dashboard/Explorer. After the buffer period ends, the corresponding service will be down and cannot be started.
- For the trial version license:
- Within 7 days before the license expires or on the day the license expires, there is an expiration reminder when NebulaGraph/Dashboard/Explorer is started.
- There is no buffer period after expiration. Once the license expires, the corresponding service will be down and cannot be started.

After your license expires, contact us via inqury@vesoft.com to renew it.

## 24.4 FAQ

This topic lists the frequently asked questions for using NebulaGraph 3.4.1. You can use the search box in the help center or the search function of the browser to match the questions you are looking for.

If the solutions described in this topic cannot solve your problems, ask for help on the NebulaGraph forum or submit an issue on GitHub issue.

#### 24.4.1 About manual updates

#### "Why is the behavior in the manual not consistent with the system?"

NebulaGraph is still under development. Its behavior changes from time to time. Users can submit an issue to inform the team if the manual and the system are not consistent.



If you find some errors in this topic:

- 1. Click the pencil button at the top right side of this page.
- 2. Use markdown to fix this error. Then click "Commit changes" at the bottom, which will start a Github pull request.
- 3. Sign the CLA. This pull request will be merged after the acceptance of at least two reviewers.

## 24.4.2 About legacy version compatibility



Neubla Graph 3.4.1 is **not compatible** with NebulaGraph 1.x nor 2.0-RC in both data formats and RPC-protocols, and **vice versa**. The service process may **quit** if using an **lower version** client to connect to a **higher version** server.

To upgrade data formats, see Upgrade NebulaGraph to the current version. Users must upgrade all clients.

#### 24.4.3 About execution errors

#### "How to resolve the error SemanticError: Missing yield clause.?"

Starting with NebulaGraph 3.0.0, the statements LOOKUP, 60, and FETCH must output results with the YIELD clause. For more information, see YIELD.

## "How to resolve the error Host not enough! ?"

From NebulaGraph version 3.0.0, the Storage services added in the configuration files **CANNOT** be read or written directly. The configuration files only register the Storage services into the Meta services. You must run the ADD HOSTS command to read and write data on Storage servers. For more information, see <u>Manage Storage hosts</u>.

"How to resolve the error To get the property of the vertex in 'v.age', should use the format 'var.tag.prop'?"

From NebulaGraph version 3.0.0, patterns support matching multiple tags at the same time, so you need to specify a tag name when querying properties. The original statement RETURN variable\_name.property\_name is changed to RETURN variable\_name.<a href="mailto:return">return</a> variable\_name.stag\_name.property\_name .

#### "How to resolve the error Storage Error E\_RPC\_FAILURE ?"

The reason for this error is usually that the storaged process returns too many data back to the graphd process. Possible solutions are as follows:

- Modify configuration files: Modify the value of --storage\_client\_timeout\_ms in the nebula-graphd.conf file to extend the connection timeout of the Storage client. This configuration is measured in milliseconds (ms). For example, set --storage\_client\_timeout\_ms=60000 . If this parameter is not specified in the nebula-graphd.conf file, specify it manually. Tip: Add --local\_config=true at the beginning of the configuration file and restart the service.
- Optimize the query statement: Reduce queries that scan the entire database. No matter whether LIMIT is used to limit the number of returned results, use the 60 statement to rewrite the MATCH statement (the former is optimized, while the latter is not).
- Check whether the Storaged process has OOM. (dmesg |grep nebula).
- Use better SSD or memory for the Storage Server.
- · Retry.

#### "How to resolve the error The leader has changed. Try again later?"

It is a known issue. Just retry 1 to N times, where N is the partition number. The reason is that the meta client needs some heartbeats to update or errors to trigger the new leader information.

If this error occurs when logging in to NebulaGraph, you can consider using df -h to view the disk space and check whether the local disk is full.

#### Unable to download SNAPSHOT packages when compiling Exchange, Connectors, or Algorithm

Problem description: The system reports Could not find artifact com.vesoft:client:jar:xxx-SNAPSHOT when compiling.

Cause: There is no local Maven repository for storing or downloading SNAPSHOT packages. The default central repository in Maven only stores official releases, not development versions (SNAPSHOTs).

Solution: Add the following configuration in the profiles scope of Maven's setting.xml file:

#### "How to resolve [ERROR (-1004)]: SyntaxError: syntax error near?"

In most cases, a query statement requires a YIELD or a RETURN. Check your query statement to see if YIELD or RETURN is provided.

#### "How to resolve the error can't solve the start vids from the sentence?"

The graphd process requires start vids to begin a graph traversal. The start vids can be specified by the user. For example:

```
> GO FROM ${vids} ...
> MATCH (src) WHERE id(src) = ${vids}
# The "start vids" are explicitly given by ${vids}.
```

It can also be found from a property index. For example:

```
# CREATE TAG INDEX IF NOT EXISTS i_player ON player(name(20));
# REBUILD TAG INDEX i_player;
```

```
> LOOKUP ON player WHERE player.name == "abc" | ... YIELD ...
> MATCH (src) WHERE src.name == "abc" ...
# The "start vids" are found from the property index "name".
```

Otherwise, an error like can't solve the start vids from the sentence will be returned.

#### "How to resolve the error Wrong vertex id type: 1001?"

Check whether the VID is INT64 or FIXED\_STRING(N) set by create space. For more information, see create space.

#### "How to resolve the error The VID must be a 64-bit integer or a string fitting space vertex id length limit.?"

Check whether the length of the VID exceeds the limitation. For more information, see create space.

#### "How to resolve the error edge conflict or vertex conflict?"

NebulaGraph may return such errors when the Storage service receives multiple requests to insert or update the same vertex or edge within milliseconds. Try the failed requests again later.

#### "How to resolve the error RPC failure in MetaClient: Connection refused?"

The reason for this error is usually that the metad service status is unusual, or the network of the machine where the metad and graphd services are located is disconnected. Possible solutions are as follows:

- Check the metad service status on the server where the metad is located. If the service status is unusual, restart the metad service.
- Use telnet meta-ip:port to check the network status under the server that returns an error.
- Check the port information in the configuration file. If the port is different from the one used when connecting, use the port in the configuration file or modify the configuration.

# "How to resolve the error StorageClientBase.inl:214] Request to "x.x.x.x":9779 failed: N6apache6thrift9transport19TTransportExceptionE: Timed Out in nebula-graph.INFO?"

The reason for this error may be that the amount of data to be queried is too large, and the storaged process has timed out. Possible solutions are as follows:

- When importing data, set Compaction manually to make read faster.
- Extend the RPC connection timeout of the Graph service and the Storage service. Modify the value of --storage\_client\_timeout\_ms in the nebula-graphd.conf file. This configuration is measured in milliseconds (ms). The default value is 60000ms.

# "How to resolve the error MetaClient.cpp:65] Heartbeat failed, status:Wrong cluster! in nebula-storaged.INFO, or HBProcessor.cpp:54] Reject wrong cluster host "x.x.x.x":9771! in nebula-metad.INFO?"

The reason for this error may be that the user has modified the IP or the port information of the metad process, or the storage service has joined other clusters before. Possible solutions are as follows:

Delete the cluster.id file in the installation directory where the storage machine is deployed (the default installation directory is /usr/local/nebula), and restart the storaged service.

## "How to resolve the error Storage Error: More than one request trying to add/update/delete one edge/vertex at he same time.?"

The reason for this error is that the current NebulaGraph version does not support concurrent requests to the same vertex or edge at the same time. To solve this error, re-execute your commands.

#### 24.4.4 About design and functions

#### "How is the time spent value at the end of each return message calculated?"

Take the returned message of SHOW SPACES as an example:



- The first number 1235 shows the time spent by the database itself, that is, the time it takes for the query engine to receive a query from the client, fetch the data from the storage server, and perform a series of calculations.
- The second number 1934 shows the time spent from the client's perspective, that is, the time it takes for the client from sending a request, receiving a response, and displaying the result on the screen.

#### "Why does the port number of the nebula-storaged process keep showing red after connecting to NebulaGraph?"

Because the nebula-storaged process waits for nebula-metad to add the current Storage service during the startup process. The Storage works after it receives the ready signal. Starting from NebulaGraph 3.0.0, the Meta service cannot directly read or write data in the Storage service that you add in the configuration file. The configuration file only registers the Storage service to the Meta service. You must run the ADD HOSTS command to enable the Meta to read and write data in the Storage service. For more information, see Manage Storage hosts.

#### "Why is there no line separating each row in the returned result of NebulaGraph 2.6.0?"

This is caused by the release of NebulaGraph Console 2.6.0, not the change of NebulaGraph core. And it will not affect the content of the returned data itself.

## About dangling edges

A dangling edge is an edge that only connects to a single vertex and only one part of the edge connects to the vertex.

Dangling edges may appear in NebulaGraph 3.4.1 as the design. And there is no MERGE statements of openCypher. The guarantee for dangling edges depends entirely on the application level. For more information, see INSERT VERTEX, DELETE VERTEX, INSERT EDGE, DELETE EDGE.

#### "Can I set replica\_factor as an even number in CREATE SPACE statements, e.g., replica\_factor = 2?"

NO.

The Storage service guarantees its availability based on the Raft consensus protocol. The number of failed replicas must not exceed half of the total replica number.

When the number of machines is 1, replica\_factor can only be set to 1.

When there are enough machines and <code>replica\_factor=2</code>, if one replica fails, the Storage service fails. No matter <code>replica\_factor=3</code> or <code>replica\_factor=4</code>, if more than one replica fails, the Storage Service fails. To prevent unnecessary waste of resources, we recommend that you set an odd replica number.

We suggest that you set replica\_factor=3 for a production environment and replica\_factor=1 for a test environment. Do not use an even number.

#### "Is stopping or killing slow queries supported?"

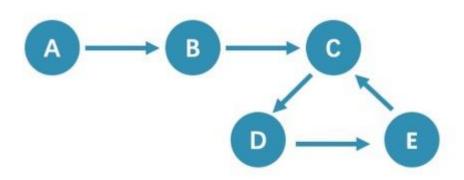
Yes. For more information, see Kill query.

#### "Why are the query results different when using 60 and MATCH to execute the same semantic query?"

The possible reasons are listed as follows.

- 60 statements find the dangling edges.
- RETURN commands do not specify the sequence.
- The dense vertex truncation limitation defined by max\_edge\_returned\_per\_vertex in the Storage service is triggered.
- Using different types of paths may cause different query results.
- 60 statements use walk. Both vertices and edges can be repeatedly visited in graph traversal.
- MATCH statements are compatible with openCypher and use trail. Only vertices can be repeatedly visited in graph traversal.

The example is as follows.



All queries that start from A with 5 hops will end at C (A->B->C->D->E->C). If it is 6 hops, the GO statement will end at D (A->B->C->D->E->C), because the edge C->D can be visited repeatedly. However, the MATCH statement returns empty, because edges cannot be visited repeatedly.

Therefore, using 60 and MATCH to execute the same semantic query may cause different query results.

For more information, see Wikipedia.

## "How to count the vertices/edges number of each tag/edge type?"

See show-stats.

#### "How to get all the vertices/edge of each tag/edge type?"

1. Create and rebuild the index.

```
> CREATE TAG INDEX IF NOT EXISTS i_player ON player();
> REBUILD TAG INDEX IF NOT EXISTS i_player;
```

2. Use LOOKUP or MATCH. For example:

```
> LOOKUP ON player;
> MATCH (n:player) RETURN n;
```

For more information, see  $\mbox{\sc INDEX}$  ,  $\mbox{\sc LOOKUP}$  , and  $\mbox{\sc MATCH}$  .

## "How to get all the vertices/edges without specifying the types?"

By nGQL, you CAN NOT directly getting all the vertices without specifying the tags, neither the edges, or you can use the LIMIT clause to limit the number of returns.

E.g., You CAN NOT run MATCH (n) RETURN (n). An error like Scan vertices or edges need to specify a limit number, or limit number can not push down. will be returned.

You can use NebulaGraph Algorithm.

Or get vertices by each tag, and then group them by yourself.

"Can non-English characters be used as identifiers, such as the names of graph spaces, tags, edge types, properties, and indexes?"

Yes, for more information, see Keywords and reserved words.

## "How to get the out-degree/the in-degree of a given vertex?"

The out-degree of a vertex refers to the number of edges starting from that vertex, while the in-degree refers to the number of edges pointing to that vertex.

```
nebula > MATCH (s)-[e]->() WHERE id(s) == "given" RETURN count(e); #Out-degree
nebula > MATCH (s)<-[e]-() WHERE id(s) == "given" RETURN count(e); #In-degree</pre>
```

This is a very slow operation to get the out/in degree since no accelaration can be applied (no indices or caches). It also could be out-of-memory when hitting a supper-node.

#### "How to quickly get the out-degree and in-degree of all vertices?"

There is no such command.

You can use NebulaGraph Algorithm.

#### 24.4.5 About operation and maintenance

#### "The runtime log files are too large. How to recycle the logs?"

By default, the runtime logs of NebulaGraph are stored in /usr/tocat/nebula/togs/. The INFO level log files are nebula-graphd.INFO, nebula-storaged.INFO, nebula-metad.INFO. If an alarm or error occurs, the suffixes are modified as .WARNING or .ERROR.

NebulaGraph uses glog to print logs. glog cannot recycle the outdated files. To rotate logs, you can:

- Use crontab to delete logs periodically. For more information, see Glog should delete old log files automatically.
- Use logrotate to manage log files. Before using logrotate, modify the configurations of corresponding services and set timestamp\_in\_logfile\_name to false.

#### "How to check the NebulaGraph version?"

If the service is running: run command SHOW HOSTS META in nebula-console . See SHOW HOSTS.

If the service is not running:

 $Different\ installation\ methods\ make\ the\ method\ of\ checking\ the\ version\ different.\ The\ instructions\ are\ as\ follows:$ 

If the service is not running, run the command ./<binary\_name> --version to get the version and the Git commit IDs of the NebulaGraph binary files. For example:

#### \$ ./nebula-graphd --version

• If you deploy NebulaGraph with Docker Compose

Check the version of NebulaGraph deployed by Docker Compose. The method is similar to the previous method, except that you have to enter the container first. The commands are as follows:

```
docker exec -it nebula-docker-compose_graphd_1 bash
cd bin/
./nebula-graphd --version
```

• If you install NebulaGraph with RPM/DEB package

Run rpm -qa |grep nebula to check the version of NebulaGraph.

#### "How to scale out or scale in? (Enterprise Edition only)"

- You can scale Graph and Storage services with Dashboard Enterprise Edition. For details, see Scale.
- You can also use NebulaGraph Operator to scale Graph and Storage services. For details, see Deploy NebulaGraph clusters with Kubectl and Deploy NebulaGraph clusters with Helm.

NebulaGraph 3.4.1 does not provide any commands or tools to support automatic scale out/in. You can refer to the following steps:

1. Scale out and scale in metad: The metad process can not be scaled out or scale in. The process cannot be moved to a new machine. You cannot add a new metad process to the service.



You can use the Meta transfer script tool to migrate Meta services. Note that the Meta-related settings in the configuration files of Storage and Graph services need to be modified correspondingly.

- 2. Scale in graphd: Remove the IP of the graphd process from the code in the client. Close this graphd process.
- 3. Scale out graphd: Prepare the binary and config files of the graphd process in the new host. Modify the config files and add all existing addresses of the metad processes. Then start the new graphd process.
- 4. Scale in storaged: See Balance remove command. After the command is finished, stop this storaged process.
- 5. Scale out storaged: Prepare the binary and config files of the storaged process in the new host, Modify the config files and add all existing addresses of the metad processes. Then register the storaged process to the metad, and then start the new storaged process. For details, see Register storaged services.

You also need to run Balance Data and Balance leader after scaling in/out storaged.

"After changing the name of the host, the old one keeps displaying OFFLINE . What should I do?"

Hosts with the status of OFFLINE will be automatically deleted after one day.

#### "How do I view the dmp file?"

The dmp file is an error report file detailing the exit of the process and can be viewed with the gdb utility. the Coredump file is saved in the directory of the startup binary (by default it is /usr/local/nebula) and is generated automatically when the NebulaGraph service crashes.

1. Check the Core file process name, pid is usually a numeric value.

```
$ file core.<pid>
```

2. Use gdb to debug.

```
$ gdb process.name core.<pid>
```

3. View the contents of the file.

```
$(gdb) bt
```

#### For example:

```
$ file core.1316027
core.1316027: ELF 64-bit LSB core file, x86-64, version 1 (SYSV), SVR4-style, from '/home/workspace/fork/nebula-debug/bin/nebula-metad --flagfile /home/k', real uid: 1008, effective uid: 1008, real gid: 1008, effective gid: 1008, execfn: '/home/workspace/fork/nebula-debug/bin/nebula-metad', platform: 'x86_64'

$ gdb /home/workspace/fork/nebula-debug/bin/nebula-metad core.1316027

$(gdb) bt
#0 0x00007f9de58fecf5 in __memcpy_ssse3_back () from /lib64/libc.so.6
#1 0x00000000002299 in void std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>::_M_construct<char*, char*, std::forward_iterator_tag) ()
#2 0x00000000000f71a7 in nebula::meta::cpp2::QueryDesc(nebula::meta::cpp2::QueryDesc const&) ()
...
```

If you are not clear about the information that dmp prints out, you can post the printout with the OS version, hardware configuration, error logs before and after the Core file was created and actions that may have caused the error on the NebulaGraph forum.

## How can I set the NebulaGraph service to start automatically on boot via systemctl?

1. Execute systematl enable to start the metad, graphd and storaged services.

```
[root]# systemctl enable nebula-metad.service
Created symlink from /etc/systemd/system/multi-user.target.wants/nebula-metad.service to /usr/lib/systemd/system/nebula-metad.service.
[root]# systemctl enable nebula-graphd.service
Created symlink from /etc/systemd/system/multi-user.target.wants/nebula-graphd.service to /usr/lib/systemd/system/nebula-graphd.service.
[root]# systemctl enable nebula-storaged.service
Created symlink from /etc/systemd/system/multi-user.target.wants/nebula-storaged.service to /usr/lib/systemd/system/nebula-storaged.service.
```

2. Configure the service files for metad, graphd and storaged to set the service to pull up automatically.



The following points need to be noted when configuring the service file. - The paths of the PIDFile, ExecStart, ExecReload and ExecStop parameters need to be the same as those on the server. - RestartSec is the length of time (in seconds) to wait before restarting, which can be modified according to the actual situation. - (Optional) StartLimitInterval is the unlimited restart, the default is 10 seconds if the restart exceeds 5 times, and set to 0 means unlimited restart. - (Optional) LimitNOFILE is the maximum number of open files for the service, the default is 1024 and can be changed according to the actual situation.

Configure the service file for the metad service.

```
$ vi /usr/lib/systemd/system/nebula-metad.service
[Unit]
Description=Nebula Graph Metad Service
After=network.target

[Service ]
Type=forking
Restart=always
Restart=always
RestartSec=15s
PIDFile=/usr/local/nebula/pids/nebula-metad.pid
```

ExecStart=/usr/local/nebula/scripts/nebula.service start metad ExecReload=/usr/local/nebula/scripts/nebula.service restart metad ExecStop=/usr/local/nebula/scripts/nebula.service stop metad PrivateImp=true StartLimitInterval=0 LimitNOFILE=1024

[Install]
WantedBy=multi-user.target

#### Configure the service file for the graphd service.

\$ vi /usr/lib/systemd/system/nebula-graphd.service
[Unit]
Description=Nebula Graph Graphd Service
After=network.target

[Service]
Type=forking
Restart=always
Restart=c=15s
PIDFile=/usr/local/nebula/pids/nebula-graphd.pid
ExecStart=/usr/local/nebula/scripts/nebula.service start graphd
ExecReload=/usr/local/nebula/scripts/nebula.service restart graphd
ExecReload=/usr/local/nebula/scripts/nebula.service stop graphd
PrivateTmp=true
StartLimitInterval=0
LimitNOFILE=1024

[Install]
WantedBy=multi-user.target

#### Configure the service file for the storaged service.

\$ vi /usr/lib/systemd/system/nebula-storaged.service
[Unit]
Description=Nebula Graph Storaged Service
After=network.target

[Service]
Type=forking
Restart=always
RestartSec=15s
PIDFile=/usr/local/nebula/pids/nebula-storaged.pid
ExecStart=/usr/local/nebula/scripts/nebula.service start storaged
ExecReload=/usr/local/nebula/scripts/nebula.service restart storaged
ExecReload=/usr/local/nebula/scripts/nebula.service stop storaged
PrivateTmp=true
StartLimitInterval=0
LimitNOFILE=1024

[Install]
WantedBy=multi-user.target

#### 3. Reload the configuration file.

[root]# sudo systemctl daemon-reload

#### 4. Restart the service.

\$ systemctl restart nebula-metad.service
\$ systemctl restart nebula-graphd.service
\$ systemctl restart nebula-storaged.service

#### 24.4.6 About connections

## "Which ports should be opened on the firewalls?"

If you have not modified the predefined ports in the Configurations, open the following ports for the NebulaGraph services:

Service	Port
Meta	9559, 9560, 19559
Graph	9669, 19669
Storage	9777 ~ 9780, 19779

If you have customized the configuration files and changed the predefined ports, find the port numbers in your configuration files and open them on the firewalls.

For those eco-tools, see the corresponding document.

The following are the default ports used by NebulaGraph core and peripheral tools.

No.	Product / Service	Туре	Default	Description
1	NebulaGraph	TCP	9669	Graph service RPC daemon listening port (commonly used for client connections to the Graph service).
2	NebulaGraph	TCP	19669	Graph service HTTP port.
3	NebulaGraph	TCP	19670	Graph service HTTP/2 port. (Deprecated after version 3.x)
4	NebulaGraph	TCP	9559	Meta service RPC daemon listening port. (Commonly used by Graph and Storage services for querying and updating metadata in the graph database).
5	NebulaGraph	TCP	9560	Raft communication port between Meta services.
6	NebulaGraph	TCP	19559	Meta service HTTP port.
7	NebulaGraph	TCP	19560	Meta service HTTP/2 port. (Deprecated after version 3.x)
8	NebulaGraph	TCP	9777	Drainer service port in Storage services (exposed only in Enterprise Edition clusters).
9	NebulaGraph	TCP	9778	Admin service port in Storage services.
10	NebulaGraph	TCP	9779	Storage service RPC daemon listening port. (Commonly used by Graph services for data storage-related operations, such as reading, writing, or deleting data).
11	NebulaGraph	TCP	9780	Raft communication port between Storage services.
12	NebulaGraph	TCP	19779	Storage service HTTP port.
13	NebulaGraph	TCP	19780	Storage service HTTP/2 port. (Deprecated after version $3.x$ )
14	NebulaGraph	TCP	8888	Backup and restore Agent service port. The Agent is a daemon running on each machine in the cluster, responsible for starting and stopping NebulaGraph services and uploading and downloading backup files.
15	NebulaGraph	ТСР	9789, 9790, and 9788	Full-text index Raft Listener port, which reads data from Storage services and writes it to the Elasticsearch cluster. Also the port for Storage Listener in inter-cluster data synchronization, used for synchronizing Storage data from the primary cluster. Ports 9790 and 9788 are generated by adding and subtracting one from port 9789.
16	NebulaGraph	TCP	9200	NebulaGraph uses this port for HTTP communication with Elasticsearch to perform full-text search queries and manage full-text indexes.
17	NebulaGraph	TCP	9569, 9570, and 9568	Meta Listener port in inter-cluster data synchronization, used for synchronizing Meta data from the primary cluster. Ports 9570 and 9568 are generated by adding and subtracting one from port 9569.
18	NebulaGraph	TCP	9889, 9890, and 9888	Drainer service port in inter-cluster data synchronization, used for synchronizing Storage and Meta data to the primary cluster. Ports 9890 and 9888 are generated by adding and subtracting one from port 9889.
19	NebulaGraph Studio	TCP	7001	Studio web service port.
20		TCP	8090	

No.	Product / Service	Туре	Default	Description
	NebulaGraph Dashboard			Nebula HTTP Gateway dependency service port. Provides an HTTP interface for cluster services to interact with the NebulaGraph database using nGQL statements.
21	NebulaGraph Dashboard	TCP	9200	Nebula Stats Exporter dependency service port. Collects cluster performance metrics, including service IP addresses, versions, and monitoring metrics (such as query count, query latency, heartbeat latency, etc.).
22	NebulaGraph Dashboard	TCP	9100	Node Exporter dependency service port. Collects resource information for machines in the cluster, including CPU, memory, load, disk, and traffic.
23	NebulaGraph Dashboard	TCP	9090	Prometheus service port. Time-series database for storing monitoring data.
24	NebulaGraph Dashboard	TCP	7003	Dashboard Community Edition web service port.
25	NebulaGraph Dashboard	TCP	7005	Dashboard Enterprise Edition web service port.
26	NebulaGraph Dashboard	TCP	9093	Alertmanager service port. Receives alerts from Prometheus and sends alert notifications to Dashboard.
27	NebulaGraph Explorer	TCP	7002	Explorer web service port.

#### "How to test whether a port is open or closed?"

You can use telnet as follows to check for port status.

telnet <ip> <port>



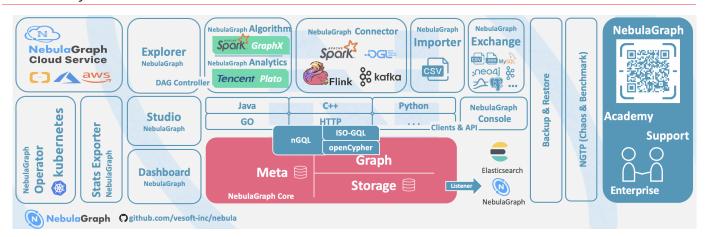
If you cannot use the telnet command, check if telnet is installed or enabled on your host.

# $For \ example:$

```
// If the port is open:
$ telnet 192.168.1.10 9669
Trying 192.168.1.10...
Connected to 192.168.1.10.
Escape character is '^]'.

// If the port is closed or blocked:
$ telnet 192.168.1.10 9777
Trying 192.168.1.10...
telnet: connect to address 192.168.1.10: Connection refused
```

# 24.5 Ecosystem tools overview



# **E**mpatibility

The core release number naming rule is X.Y.Z, which means Major version X, Medium version Y, and Minor version Z. The upgrade requirements for the client are:

- Upgrade the core from X.Y.Z1 to X.Y.Z2: It means that the core is fully forward compatible and is usually used for bugfixes. It is recommended to upgrade the minor version of the core as soon as possible. At this time, the client can stay **not upgraded**.
- Upgrade the core from X.Y1.\* to X.Y2.\*: It means that there is some incompatibility of API, syntax, and return value. It is usually used to add functions, improve performance, and optimize code. The client needs to be upgraded to X.Y2.\*.
- Upgrade the core from X1.\*.\* to X2.\*.\*: It means that there is a major incompatibility in storage formats, API, syntax, etc. You need to use tools to upgrade the core data. The client must be upgraded.
- ullet The default core and client do not support downgrade: You cannot downgrade from X.Y.Z2 to X.Y.Z1.
- The version released at the beginning of the year is usually named X.O.O, and in the middle of the year, it is named X.5.O.
- The file name contains RC to indicate an unofficial version (Release Candidate) that is only used for preview. Its maintenance period is only until the next RC or official version is released. Its client, data compatibility, etc. are not guaranteed.
- The files with nightly, SNAPSHOT, or date are the nightly versions. There is no quality assurance and maintenance period.

#### 24.5.1 NebulaGraph Studio

NebulaGraph Studio (Studio for short) is a graph database visualization tool that can be accessed through the Web. It can be used with NebulaGraph DBMS to provide one-stop services such as composition, data import, writing nGQL queries, and graph exploration. For details, see What is NebulaGraph Studio.



The release of the Studio is independent of NebulaGraph core, and its naming method is also not the same as the core naming rules.

NebulaGraph version	Studio version
v3.4.1	v3.6.0

#### 24.5.2 NebulaGraph Dashboard Community Edition

NebulaGraph Dashboard Community Edition (Dashboard for short) is a visualization tool for monitoring the status of machines and services in the NebulaGraph cluster. For details, see What is NebulaGraph Dashboard.

NebulaGraph version	Dashboard Community version
v3.4.1	v3.4.0

#### 24.5.3 NebulaGraph Dashboard Enterprise Edition

NebulaGraph Dashboard Enterprise Edition (Dashboard for short) is a visualization tool that monitors and manages the status of machines and services in NebulaGraph cluster. For details, see What is NebulaGraph Dashboard.

NebulaGraph version	Dashboard Enterprise version
v3.4.1	v3.4.2

### 24.5.4 NebulaGraph Explorer

NebulaGraph Explorer (Explorer for short) is a graph exploration visualization tool that can be accessed through the Web. It is used with the NebulaGraph core to visualize interaction with graph data. Users can quickly become map experts, even without experience in map data manipulation. For details, see What is NebulaGraph Explorer.

NebulaGraph version	<b>Explorer Enterprise version</b>
v3.4.1	v3.4.0

#### 24.5.5 NebulaGraph Stats Exporter

 ${\color{red} \textbf{Nebula-stats-exporter}}\ \textbf{exports}\ \textbf{monitor}\ \textbf{metrics}\ \textbf{to}\ \textbf{Promethus}.$ 

NebulaGraph version	Stats Exporter version
v3.4.1	v3.3.0

#### 24.5.6 NebulaGraph Exchange

NebulaGraph Exchange (Exchange for short) is an Apache Spark&trade application for batch migration of data in a cluster to NebulaGraph in a distributed environment. It can support the migration of batch data and streaming data in a variety of different formats. For details, see What is NebulaGraph Exchange.

NebulaGraph version	Exchange Community version	Exchange Enterprise version
v3.4.1	v3.4.0	v3.4.0

#### 24.5.7 NebulaGraph Operator

NebulaGraph Operator (Operator for short) is a tool to automate the deployment, operation, and maintenance of NebulaGraph clusters on Kubernetes. Building upon the excellent scalability mechanism of Kubernetes, NebulaGraph introduced its operation and maintenance knowledge into the Kubernetes system, which makes NebulaGraph a real cloud-native graph database. For more information, see What is NebulaGraph Operator.

NebulaGraph version	Operator version
v3.4.1	v1.4.2

#### 24.5.8 NebulaGraph Importer

NebulaGraph Importer (Importer for short) is a CSV file import tool for NebulaGraph. The Importer can read the local CSV file, and then import the data into the NebulaGraph database. For details, see What is NebulaGraph Importer.

NebulaGraph version	Importer version
v3.4.1	v3.4.0

#### 24.5.9 NebulaGraph Spark Connector

NebulaGraph Spark Connector is a Spark connector that provides the ability to read and write NebulaGraph data in the Spark standard format. NebulaGraph Spark Connector consists of two parts, Reader and Writer. For details, see What is NebulaGraph Spark Connector.

NebulaGraph version	<b>Spark Connector version</b>
v3.4.1	v3.4.0

#### 24.5.10 NebulaGraph Flink Connector

NebulaGraph Flink Connector is a connector that helps Flink users quickly access NebulaGraph. It supports reading data from the NebulaGraph database or writing data read from other external data sources to the NebulaGraph database. For details, see What is NebulaGraph Flink Connector.

NebulaGraph version	Flink Connector version
v3.4.1	v3.3.0

#### 24.5.11 NebulaGraph Algorithm

NebulaGraph Algorithm (Algorithm for short) is a Spark application based on GraphX, which uses a complete algorithm tool to analyze data in the NebulaGraph database by submitting a Spark task To perform graph computing, use the algorithm under the lib repository through programming to perform graph computing for DataFrame. For details, see What is NebulaGraph Algorithm.

NebulaGraph version	Algorithm version
v3.4.1	v3.0.0

#### 24.5.12 NebulaGraph Analytics

NebulaGraph Analytics is an application that integrates the open-source Plato Graph Computing Framework, with which NebulaGraph Analytics performs graph computations on NebulaGraph database data. For details, see What is NebulaGraph Analytics.

NebulaGraph version	Analytics version
v3.4.1	v3.4.0

#### 24.5.13 NebulaGraph Console

NebulaGraph Console is the native CLI client of NebulaGraph. For how to use it, see NebulaGraph Console.

NebulaGraph version	Console version
v3.4.1	v3.4.0

#### 24.5.14 NebulaGraph Docker Compose

Docker Compose can quickly deploy NebulaGraph clusters. For how to use it, please refer to Docker Compose Deployment NebulaGraph.

NebulaGraph version	<b>Docker Compose version</b>
v3.4.1	v3.4.0

#### 24.5.15 Backup & Restore

Backup&Restore (BR for short) is a command line interface (CLI) tool that can help back up the graph space data of NebulaGraph, or restore it through a backup file data.

NebulaGraph version	BR version
v3.4.1	v3.3.0

### 24.5.16 Backup & Restore Enterprise Edition

Backup Restore (BR for short) Enterprise Edition is a Command-Line Interface (CLI) tool. With BR Enterprise Edition, you can back up and restore NebulaGraph Enterprise Edition data.

NebulaGraph version	BR version
v3.4.1	v3.4.1

## 24.5.17 NebulaGraph Bench

NebulaGraph Bench is used to test the baseline performance data of NebulaGraph. It uses the standard data set of LDBC.

NebulaGraph version	Bench version
v3.4.1	v1.2.0

#### 24.5.18 API, SDK



Select the latest version of  $\text{ X.Y.}^{\star}$  which is the same as the core version.

Language
C++
Go
Python
Java
HTTP

#### 24.5.19 Not Released

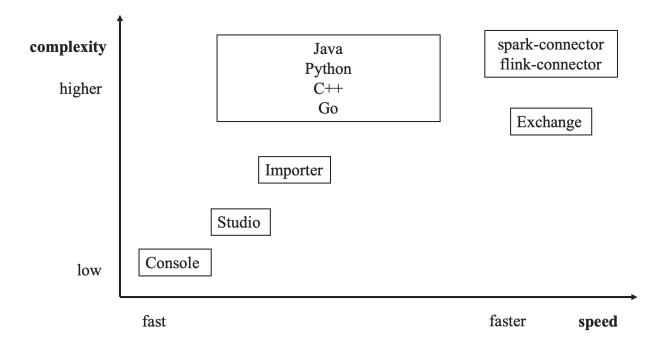
- Rust Client
- Node.js Client
- Object Graph Mapping Library (OGM, or ORM)

# 24.6 Import tools

There are many ways to write NebulaGraph 3.4.1:

- Import with the command -f: This method imports a small number of prepared nGQL files, which is suitable to prepare for a small amount of manual test data.
- Import with Studio: This method uses a browser to import multiple csv files of this machine. A single file cannot exceed 100 MB, and its format is limited.
- Import with Importer: This method imports multiple csv files on a single machine with unlimited size and flexible format.
- Import with Exchange: This method imports from various distribution sources, such as Neo4j, Hive, MySQL, etc., which requires a Spark cluster.
- Import with Spark-connector/Flink-connector: This method has corresponding components (Spark/Flink) and writes a small amount of code.
- Import with  $C++/GO/Java/Python\ SDK$ : This method imports in the way of writing programs, which requires certain programming and tuning skills.

The following figure shows the positions of these ways:



#### 24.7 How to Contribute

#### 24.7.1 Before you get started

#### Commit an issue on the github or forum

You are welcome to contribute any code or files to the project. But firstly we suggest you raise an issue on the github or the forum to start a discussion with the community. Check through the topic for Github.

#### Sign the Contributor License Agreement (CLA)

What is **CLA**?

Here is the vesoft inc. Contributor License Agreement.

Click the Sign in with GitHub to agree button to sign the CLA.

If you have any questions, send an email to  $\mbox{info@vesoft.com}$ .

#### 24.7.2 Modify a single document

This manual is written in the Markdown language. Click the pencil icon on the right of the document title to commit the modification.

This method applies to modify a single document only.

## 24.7.3 Batch modify or add files

This method applies to contribute codes, modify multiple documents in batches, or add new documents.

#### 24.7.4 Step 1: Fork in the github.com

The NebulaGraph project has many repositories. Take the nebul repository for example:

- 1. Visit https://github.com/vesoft-inc/nebula.
- 2. Click the Fork button to establish an online fork.

# 24.7.5 Step 2: Clone Fork to Local Storage

1. Define a local working directory.

```
# Define the working directory.
working_dir=$HOME/Workspace
```

2. Set user to match the Github profile name.

```
user={the Github profile name}
```

3. Create your clone.

```
mkdir -p $working_dir
cd $working_dir
git clone https://github.com/$user/nebula.git
# or: git clone git@github.com:$user/nebula.git

cd $working_dir/nebula
git remote add upstream https://github.com/vesoft-inc/nebula.git
# or: git remote add upstream git@github.com:vesoft-inc/nebula.git
# Never push to upstream master since you do not have write access.
git remote set-url --push upstream no_push
# Confirm that the remote branch is valid.
```

```
# The correct format is:
# origin git@github.com:$(user)/nebula.git (fetch)
# origin git@github.com:$(user)/nebula.git (push)
# upstream https://github.com/vesoft-inc/nebula (fetch)
# upstream no_push (push)
git remote -v
```

#### 4. (Optional) Define a pre-commit hook.

Please link the NebulaGraph pre-commit hook into the .git directory.

This hook checks the commits for formatting, building, doc generation, etc.

```
cd $working_dir/nebula/.git/hooks
ln -s $working_dir/nebula/.linters/cpp/hooks/pre-commit.sh .
```

Sometimes, the pre-commit hook cannot be executed. You have to execute it manually.

```
cd $working_dir/nebula/.git/hooks
chmod +x pre-commit
```

#### 24.7.6 Step 3: Branch

1. Get your local master up to date.

```
cd $working_dir/nebula
git fetch upstream
git checkout master
git rebase upstream/master
```

2. Checkout a new branch from master.

```
git checkout -b myfeature
```



Because the PR often consists of several commits, which might be squashed while being merged into upstream. We strongly suggest you to open a separate topic branch to make your changes on. After merged, this topic branch can be just abandoned, thus you could synchronize your master branch with upstream easily with a rebase like above. Otherwise, if you commit your changes directly into master, you need to use a hard reset on the master branch. For example:

```
git fetch upstream
git checkout master
git reset --hard upstream/master
git push --force origin master
```

#### 24.7.7 Step 4: Develop

• Code style

**NebulaGraph** adopts copplint to make sure that the project conforms to Google's coding style guides. The checker will be implemented before the code is committed.

· Unit tests requirements

Please add unit tests for the new features or bug fixes.

· Build your code with unit tests enabled

For more information, see Install NebulaGraph by compiling the source code.



Make sure you have enabled the building of unit tests by setting -DENABLE\_TESTING=ON.

· Run tests

In the root directory of nebula, run the following command:

cd nebula/build
ctest -j\$(nproc)

#### 24.7.8 Step 5: Bring Your Branch Update to Date

# While on your myfeature branch. git fetch upstream git rebase upstream/master

Users need to bring the head branch up to date after other contributors merge PR to the base branch.

### 24.7.9 Step 6: Commit

Commit your changes.

git commit -a

Users can use the command --amend to re-edit the previous code.

#### 24.7.10 Step 7: Push

When ready to review or just to establish an offsite backup, push your branch to your fork on github.com:

git push origin myfeature

# 24.7.11 Step 8: Create a Pull Request

- 1. Visit your fork at https://github.com/\$user/nebula (replace \$user here).
- 2. Click the Compare & pull request button next to your myfeature branch.

#### 24.7.12 Step 9: Get a Code Review

Once your pull request has been created, it will be assigned to at least two reviewers. Those reviewers will do a thorough code review to make sure that the changes meet the repository's contributing guidelines and other quality standards.

#### 24.7.13 Add test cases

For detailed methods, see How to add test cases.

# 24.7.14 Donation

#### Step 1: Confirm the project donation

Contact the official NebulaGraph staff via email, WeChat, Slack, etc. to confirm the donation project. The project will be donated to the NebulaGraph Contrib organization.

Email address: info@vesoft.com

WeChat: NebulaGraphbot

Slack: Join Slack

#### Step 2: Get the information of the project recipient

The NebulaGraph official staff will give the recipient ID of the NebulaGraph Contrib project.

#### Step 3: Donate a project

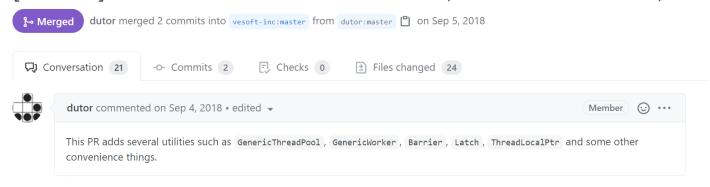
The user transfers the project to the recipient of this donation, and the recipient transfers the project to the NebulaGraph Contrib organization. After the donation, the user will continue to lead the development of community projects as a Maintainer.

For operations of transferring a repository on GitHub, see Transferring a repository owned by your user account.

# 24.8 History timeline for NebulaGraph

1. 2018.9: dutor wrote and submitted the first line of NebulaGraph database code.

# [Feature] Added some concurrent utilities, GenericThreadPool, etc.



 $2.\ 2019.5:\ Nebula Graph\ v0.1.0-alpha\ was\ released\ as\ open-source.$ 





 $Nebula Graph\ v1.0.0\text{-}beta,\ v1.0.0\text{-}rc1,\ v1.0.0\text{-}rc2,\ v1.0.0\text{-}rc3,\ and\ v1.0.0\text{-}rc4\ were\ released\ one\ after\ another\ within\ a\ year\ thereafter.$ 



# Nebula Graph v0.1.0

darionyaphet released this on May 14, 2019 · 1075 commits to master since this release

Compare ▼

This is the first release of Nebula Graph, a brand new, fast and distributed graph database.

#### **Available Features**

- Physical data isolation with Graph Space
- Strongly typed schema support
- Vertices and edges insertion
- Graph traversal(the Go statement)
- Variable definition and reference
- Piping query result between statements
- Client API in C++, Golang and Java

## **Features Coming Soon**

- Raft support
- Query based on secondary index(the LOOKUP statement)
- Sub-graph retrieval(the MATCH statement)
- User defined function call
- User management

## **Try Out**

A Docker image is available for trial purpose. You can get it by following the guide here.

→ Assets 2

Source code (zip)

Source code (tar.gz)

3. 2019.7: NebulaGraph's debut at HBaseCon<sup>1</sup>. @dangleptr



4. 2020.3: NebulaGraph v2.0 was starting developed in the final stage of v1.0 development.

5. 2020.6: The first major version of NebulaGraph v1.0.0 GA was released.

v1.0.0o- 06a5db4Verified

# V1.0.0 GA

jude-zhu released this on Jun 10, 2020 · 146 commits to master since this release

in the second seco



#### **Basic Features**

- Online DDL & DML. Support updating schemas and data without stopping or affecting your ongoing operations.
- Graph traversal. 60 statement supports forward/reverse and bidirectional graph traversal. 60 minHops TO maxHops is supported to get variable hops relationships.
- Aggregate. Support aggregation functions such as GROUP BY , ORDER BY , and LIMIT .
- Composite query. Support composite clauses: UNION , UNION DISTINCT , INTERSECT , and MINUS .
- PIPE statements. The result yielded from the previous statement could be piped to the next statement as input.
- Use defined variables. Support user-defined variables to pass the result of a query to another.
- Index. Both the single-property index and composite index are supported to make searches of related data more efficient. LOOKUP ON statement is to query on the index.

#### **Advanced Features**

- Privilege Management. Support user authentication and role-based access control. Nebula Graph can easily integrate with third-party
  authentication systems. There are five built-in roles in Nebula Graph: GOD, ADMIN, DBA, USER, and GUEST. Each role has its
  corresponding privileges.
- Support Reservoir Sampling, which will retrieve k elements randomly for the sampling of the supernode at the complexity of O(n).
- Cluster snapshot. Support creating snapshots for the cluster as an online backup strategy.
- . TTL. Support TTL to expire items after a certain amount of time automatically.
- Operation & Maintenance
  - Scale in/out. Support online scale in/out and load balance for storage
  - o HOSTS clause to manage storage hosts
  - o CONFIGS clause to manage configuration options
- Job Manager & Scheduler. A tool for job managing and scheduling. Currently, COMPACT and FLUSH jobs are supported.
- Graph Algorithms. Support finding the full path and the shortest path between vertices.
- Provide OLAP interfaces to integrate with third-party graph analytics platforms.
- Support multiple character sets and collations. The default CHARSET and COLLATE are utf8 and utf8\_bin.

#### Clients

- Java Client. Support source code building and downloading from the MVN repository, see Java Client for more details.
- Python Client. Support source code building and installation with pip, see Python Client for more details.
- Golang Client. Install the client with the command go get -u -v github.com/vesoft-inc/nebula-go , see Go Client for more details.

#### Nebula Graph Studio

A graphical user interface for working with Nebula Graph. Support querying, designing schema, data loading, and graph exploring. See Nebula Graph Studio for more details.

6. 2021.3: The second major version of NebulaGraph v2.0 GA was released.

# © v2.0.0 ◆ 916399b (Verified) Nebula Graph v2.0 GA @ jude-zhu released this on Mar 23

- New Features
- vertexID supports both Integer and String
- NULL: the property can be set to NULL. NOT NULL constraint is also supported.
- Composite types: LIST, SET, and MAP(Cannot be set as property types)
- Temporal types: DATE and DATETIME.
   FIXED\_STRING: a fixed size String
- Full-text indexes are supported to do prefix, wildcard, regex, and fuzzy search on a string property.
- . Explain & Profile outputs the execution plan of an nGQL statement and execution profile
- Subgraph to retrieve vertices and edges reachable from the start vertices.
- Support to collect statistics of the graph space.
- OpenCypher compatibility
   Partially support the MATCH clause
  - Support RETURN, WITH, UNWIND, LIMIT & SKIP clauses
- More built-in functions
  - Predicate functions
  - Scalar functions
  - List functions

  - Aggregating functions

  - String functions

#### Improvements

- Optimize the performance of inserting, updating, and deleting data with indexes.
- . LOOKUP ON filtering data supports OR and AND operators.
- FIND PATH supports finding paths with or without regard to direction, and also supports excluding cycles in paths.
- SHOW HOSTS graph/meta/storage supports to retrieve the basic information of graphd/metad/storaged hosts

- The data type of vertexID must be specified when creating a graph space.
- FETCH PROP ON returns a composite object if not specify the result set.
- Changed the default port numbers of metad , graphd , and storaged

#### Nebula-graph Console

Supports local commands mode. :set csv outputs the query results to the console and the specified CSV file. For more information,

#### Clients

Support connection pool and load balance.

- cpp client https://github.com/vesoft-inc/nebula-c
- java client https://github.com/vesoft-inc/nebula-java
- python client https://github.com/vesoft-inc/nebula-py
- go client https://github.com/vesoft-inc/nebula-go

#### Nebula Graph Studio

With Studio, you can create a graph schema, load data, execute nGQL statements, and explore graphs in one stop. For more information please refer to https://github.com/vesoft-inc/nebula-web-docke

- 7. 2021.8: NebulaGraph v2.5.0 was released.
- 8. 2021.10: NebulaGraph v2.6.0 was released.
- 9. 2022.2: NebulaGraph v3.0.0 was released.
- 10. 2022.4: NebulaGraph v3.1.0 was released.
- 11. 2022.7: NebulaGraph v3.2.0 was released.
- 12. 2022.10: NebulaGraph v3.3.0 was released. 13. 2023.2: NebulaGraph v3.4.0 was released.
  - 1. NebulaGraph v1.x supports both RocksDB and HBase as its storage engines. NebulaGraph v2.x removes HBase supports.  $\leftarrow$

# 24.9 Error code

 $Nebula Graph\ returns\ an\ error\ code\ when\ an\ error\ occurs.\ This\ topic\ describes\ the\ details\ of\ the\ error\ code\ returned.$ 

#### O Note

- If an error occurs but no error code is returned, or if the error code description is unclear, we welcome your feedback or suggestions on the forum or GitHub.
- ullet When the code returned is ullet0, it means that the operation is successful.

Error name	Error Code	Description
E_DISCONNECTED	-1	Lost connection
E_FAIL_TO_CONNECT	-2	Unable to establish connection
E_RPC_FAILURE	-3	RPC failure
E_LEADER_CHANGED	-4	Raft leader has been changed
E_SPACE_NOT_FOUND	-5	Graph space does not exist
E_TAG_NOT_FOUND	-6	Tag does not exist
E_EDGE_NOT_FOUND	-7	Edge type does not exist
E_INDEX_NOT_FOUND	-8	Index does not exist
E_EDGE_PROP_NOT_FOUND	-9	Edge type property does not exist
E_TAG_PROP_NOT_FOUND	-10	Tag property does not exist
E_ROLE_NOT_FOUND	-11	The current role does not exist
E_CONFIG_NOT_FOUND	-12	The current configuration does not exist
E_MACHINE_NOT_FOUND	-13	The current host does not exist
E_LISTENER_NOT_FOUND	-15	Listener does not exist
E_PART_NOT_FOUND	-16	The current partition does not exist
E_KEY_NOT_FOUND	-17	Key does not exist
E_USER_NOT_FOUND	-18	User does not exist
E_STATS_NOT_FOUND	-19	Statistics do not exist
E_SERVICE_NOT_FOUND	-20	No current service found
E_DRAINER_NOT_FOUND	-21	Drainer does not exist
E_DRAINER_CLIENT_NOT_FOUND	-22	Drainer client does not exist
E_PART_STOPPED	-23	The current partition has already been stopped
E_BACKUP_FAILED	-24	Backup failed
E_BACKUP_EMPTY_TABLE	-25	The backed-up table is empty
E_BACKUP_TABLE_FAILED	-26	Table backup failure
E_PARTIAL_RESULT	-27	MultiGet could not get all data
E_REBUILD_INDEX_FAILED	-28	Index rebuild failed
E_INVALID_PASSWORD	-29	Password is invalid
E_FAILED_GET_ABS_PATH	-30	Unable to get absolute path
E_BAD_USERNAME_PASSWORD	-1001	Authentication failed
E_SESSION_INVALID	-1002	Invalid session
E_SESSION_TIMEOUT	-1003	Session timeout
E_SYNTAX_ERROR	-1004	Syntax error
E_EXECUTION_ERROR	-1005	Execution error
E_STATEMENT_EMPTY	-1006	Statement is empty

Error name	Error Code	Description
E_BAD_PERMISSION	-1008	Permission denied
E_SEMANTIC_ERROR	-1009	Semantic error
E_TOO_MANY_CONNECTIONS	-1010	Maximum number of connections exceeded
E_PARTIAL_SUCCEEDED	-1011	Access to storage failed (only some requests succeeded)
E_NO_HOSTS	-2001	Host does not exist
E_EXISTED	-2002	Host already exists
E_INVALID_HOST	-2003	Invalid host
E_UNSUPPORTED	-2004	The current command, statement, or function is not supported
E_NOT_DROP	-2005	Not allowed to drop
E_CONFIG_IMMUTABLE	-2007	Configuration items cannot be changed
E_CONFLICT	-2008	Parameters conflict with meta data
E_INVALID_PARM	-2009	Invalid parameter
E_WRONGCLUSTER	-2010	Wrong cluster
E_ZONE_NOT_ENOUGH	-2011	Listener conflicts
E_ZONE_IS_EMPTY	-2012	Host not exist
E_SCHEMA_NAME_EXISTS	-2013	Schema name already exists
E_RELATED_INDEX_EXISTS	-2014	There are still indexes related to tag or edge, cannot drop it
E_RELATED_SPACE_EXISTS	-2015	There are still some space on the host, cannot drop it
E_STORE_FAILURE	-2021	Failed to store data
E_STORE_SEGMENT_ILLEGAL	-2022	Illegal storage segment
E_BAD_BALANCE_PLAN	-2023	Invalid data balancing plan
E_BALANCED	-2024	The cluster is already in the data balancing status
E_NO_RUNNING_BALANCE_PLAN	-2025	There is no running data balancing plan
E_NO_VALID_HOST	-2026	Lack of valid hosts
E_CORRUPTED_BALANCE_PLAN	-2027	A data balancing plan that has been corrupted
E_IMPROPER_ROLE	-2030	Failed to recover user role
E_INVALID_PARTITION_NUM	-2031	Number of invalid partitions
E_INVALID_REPLICA_FACTOR	-2032	Invalid replica factor
E_INVALID_CHARSET	-2033	Invalid character set
E_INVALID_COLLATE	-2034	Invalid character sorting rules
E_CHARSET_COLLATE_NOT_MATCH	-2035	Character set and character sorting rule mismatch
E_SNAPSHOT_FAILURE	-2040	Failed to generate a snapshot
E_BLOCK_WRITE_FAILURE	-2041	Failed to write block data
E_ADD_JOB_FAILURE	-2044	Failed to add new task
E_STOP_JOB_FAILURE	-2045	Failed to stop task

Error name	Error Code	Description
E_SAVE_JOB_FAILURE	-2046	Failed to save task information
E_BALANCER_FAILURE	-2047	Data balancing failed
E_JOB_NOT_FINISHED	-2048	The current task has not been completed
E_TASK_REPORT_OUT_DATE	-2049	Task report failed
E_JOB_NOT_IN_SPACE	-2050	The current task is not in the graph space
E_JOB_NEED_RECOVER	-2051	The current task needs to be resumed
E_JOB_ALREADY_FINISH	-2052	The job status has already been failed or finished
E_JOB_SUBMITTED	-2053	Job default status
E_JOB_NOT_STOPPABLE	-2054	The given job do not support stop
E_JOB_HAS_NO_TARGET_STORAGE	-2055	The leader distribution has not been reported, so can't send task to storage
E_INVALID_JOB	-2065	Invalid task
E_BACKUP_BUILDING_INDEX	-2066	Backup terminated (index being created)
E_BACKUP_SPACE_NOT_FOUND	-2067	Graph space does not exist at the time of backup
E_RESTORE_FAILURE	-2068	Backup recovery failed
E_SESSION_NOT_FOUND	-2069	Session does not exist
E_LIST_CLUSTER_FAILURE	-2070	Failed to get cluster information
E_LIST_CLUSTER_GET_ABS_PATH_FAILURE	-2071	Failed to get absolute path when getting cluster information
E_LIST_CLUSTER_NO_AGENT_FAILURE	-2072	Unable to get an agent when getting cluster information
E_QUERY_NOT_FOUND	-2073	Query not found
E_AGENT_HB_FAILUE	-2074	Failed to receive heartbeat from agent
E_HOST_CAN_NOT_BE_ADDED	-2082	The host can not be added for it's not a storage host
E_ACCESS_ES_FAILURE	-2090	Failed to access elasticsearch
E_GRAPH_MEMORY_EXCEEDED	-2600	Graph memory exceeded
E_CONSENSUS_ERROR	-3001	Consensus cannot be reached during an election
E_KEY_HAS_EXISTS	-3002	Key already exists
E_DATA_TYPE_MISMATCH	-3003	Data type mismatch
E_INVALID_FIELD_VALUE	-3004	Invalid field value
E_INVALID_OPERATION	-3005	Invalid operation
E_NOT_NULLABLE	-3006	Current value is not allowed to be empty
E_FIELD_UNSET	-3007	Field value must be set if the field value is $\ensuremath{^{\rm NOT}}$ NULL or has no default value
E_OUT_OF_RANGE	-3008	The value is out of the range of the current type
E_DATA_CONFLICT_ERROR	-3010	Data conflict
E_WRITE_STALLED	-3011	Writes are delayed

Error name	Error Code	Description
E_INVALID_SPACEVIDLEN	-3022	Invalid VID length
E_INVALID_FILTER	-3031	Invalid filter
E_INVALID_UPDATER	-3032	Invalid field update
E_INVALID_STORE	-3033	Invalid KV storage
E_INVALID_PEER	-3034	Peer invalid
E_RETRY_EXHAUSTED	-3035	Out of retries
E_TRANSFER_LEADER_FAILED	-3036	Leader change failed
E_INVALID_STAT_TYPE	-3037	Invalid stat type
E_INVALID_VID	-3038	VID is invalid
E_LOAD_META_FAILED	-3040	Failed to load meta information
E_FAILED_TO_CHECKPOINT	-3041	Failed to generate checkpoint
E_CHECKPOINT_BLOCKED	-3042	Generating checkpoint is blocked
E_FILTER_OUT	-3043	Data is filtered
E_INVALID_DATA	-3044	Invalid data
E_MUTATE_EDGE_CONFLICT	-3045	Concurrent write conflicts on the same edge
E_MUTATE_TAG_CONFLICT	-3046	Concurrent write conflict on the same vertex
E_OUTDATED_LOCK	-3047	Lock is invalid
E_INVALID_TASK_PARA	-3051	Invalid task parameter
E_USER_CANCEL	-3052	The user canceled the task
E_TASK_EXECUTION_FAILED	-3053	Task execution failed
E_PLAN_IS_KILLED	-3060	Execution plan was cleared
E_NO_TERM	-3070	The heartbeat process was not completed when the request was received
E_OUTDATED_TERM	-3071	Out-of-date heartbeat received from the old leader (the new leader has been elected)
E_WRITE_WRITE_CONFLICT	-3073	Concurrent write conflicts with later requests
E_RAFT_UNKNOWN_PART	-3500	Unknown partition
E_RAFT_LOG_GAP	-3501	Raft logs lag behind
E_RAFT_LOG_STALE	-3502	Raft logs are out of date
E_RAFT_TERM_OUT_OF_DATE	-3503	Heartbeat messages are out of date
E_RAFT_UNKNOWN_APPEND_LOG	-3504	Unknown additional logs
E_RAFT_WAITING_SNAPSHOT	-3511	Waiting for the snapshot to complete
E_RAFT_SENDING_SNAPSHOT	-3512	There was an error sending the snapshot
E_RAFT_INVALID_PEER	-3513	Invalid receiver
E_RAFT_NOT_READY	-3514	Raft did not start
E_RAFT_STOPPED	-3515	Raft has stopped

Error name	Error Code	Description
E_RAFT_BAD_ROLE	-3516	Wrong role
E_RAFT_WAL_FAIL	-3521	Write to a WAL failed
E_RAFT_HOST_STOPPED	-3522	The host has stopped
E_RAFT_TOO_MANY_REQUESTS	-3523	Too many requests
E_RAFT_PERSIST_SNAPSHOT_FAILED	-3524	Persistent snapshot failed
E_RAFT_RPC_EXCEPTION	-3525	RPC exception
E_RAFT_NO_WAL_FOUND	-3526	No WAL logs found
E_RAFT_HOST_PAUSED	-3527	Host suspended
E_RAFT_WRITE_BLOCKED	-3528	Writes are blocked
E_RAFT_BUFFER_OVERFLOW	-3529	Cache overflow
E_RAFT_ATOMIC_OP_FAILED	-3530	Atomic operation failed
E_LEADER_LEASE_FAILED	-3531	Leader lease expired
E_RAFT_CAUGHT_UP	-3532	Data has been synchronized on Raft
E_STORAGE_MEMORY_EXCEEDED	-3600	Storage memory exceeded
E_LOG_GAP	-4001	Drainer logs lag behind
E_LOG_STALE	-4002	Drainer logs are out of date
E_INVALID_DRAINER_STORE	-4003	The drainer data storage is invalid
E_SPACE_MISMATCH	-4004	Graph space mismatch
E_PART_MISMATCH	-4005	Partition mismatch
E_DATA_CONFLICT	-4006	Data conflict
E_REQ_CONFLICT	-4007	Request conflict
E_DATA_ILLEGAL	-4008	Illegal data
E_CACHE_CONFIG_ERROR	-5001	Cache configuration error
E_NOT_ENOUGH_SPACE	-5002	Insufficient space
E_CACHE_MISS	-5003	No cache hit
E_CACHE_WRITE_FAILURE	-5005	Write cache failed
E_NODE_NUMBER_EXCEED_LIMIT	-7001	Number of machines exceeded the limit
E_PARSING_LICENSE_FAILURE	-7002	Failed to resolve certificate
E_UNKNOWN	-8000	Unknown error



https://docs.nebula-graph.io/3.4.1