

Alibaba Cloud E-MapReduce

Open Source Components

Issue: 20190604

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.








1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use

or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the content of the Alibaba Cloud website, including but not limited to works, products, images, archives, information, materials, website architecture, website graphic layout, and webpage design, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of the Alibaba Cloud website, product programs, or content shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates).
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Generic conventions

Table -1: Style conventions

Style	Description	Example
	This warning information indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
	This warning information indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restore business.
	This indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: Take the necessary precautions to save exported data containing sensitive information.
	This indicates supplemental instructions, best practices, tips, and other content that is good to know for the user.	 Note: You can use Ctrl + A to select all files.
>	Multi-level menu cascade.	Settings > Network > Set network type
Bold	It is used for buttons, menus, page names, and other UI elements.	Click OK.
Courier font	It is used for commands.	Run the <code>cd / d C :/ windows</code> command to enter the Windows system folder.
<i>Italics</i>	It is used for parameters and variables.	<code>bae log list --instanceid Instance_ID</code>
[] or [a b]	It indicates that it is an optional value, and only one item can be selected.	<code>ipconfig [-all -t]</code>

Style	Description	Example
<code>{}</code> or <code>{a b}</code>	It indicates that it is a required value, and only one item can be selected.	<code>swich {stand slave}</code>

Contents

Legal disclaimer.....	I
Generic conventions.....	I
1 Hue.....	1
2 Oozie.....	3
3 Zeppelin.....	7
4 ZooKeeper.....	8
5 Kafka.....	9
5.1 Quick start.....	9
5.2 Cross-cluster access to Kafka.....	11
5.3 Kafka Ranger.....	15
5.4 Kafka SSL.....	16
5.5 Kafka Manager.....	17
5.6 Common Kafka problems.....	18
6 Druid.....	19
6.1 Introduction to Druid.....	19
6.2 Quick start.....	21
6.3 Ingestion Spec.....	34
6.4 Kafka Indexing Service.....	38
6.5 LOG Indexing Service.....	41
6.6 Tranquility.....	44
6.7 Superset.....	47
6.8 Common Druid problems.....	49
7 Presto.....	53
7.1 What is Presto?.....	53
7.2 Quick start.....	55
7.2.1 System structure.....	55
7.2.2 Basic concepts.....	55
7.2.3 Command line tool.....	56
7.2.4 Uses JDBC.....	58
7.2.5 Implement authentication with ApacheDS.....	61
7.3 Instructions.....	70
7.3.1 Overview.....	70
7.3.2 SQL manual.....	70
7.3.2.1 Data types.....	70
7.3.2.2 Common functions and operators.....	74
7.3.2.2.1 Logical operators.....	74
7.3.2.2.2 Comparison functions and operators.....	75
7.3.2.2.3 Conditional expressions.....	76

7.3.2.2.4 Conversion functions.....	78
7.3.2.2.5 Mathematical functions and operators.....	79
7.3.2.2.6 Bitwise functions.....	83
7.3.2.2.7 Decimal function.....	84
7.3.2.2.8 String functions.....	85
7.3.2.2.9 Regular expression.....	88
7.3.2.2.10 Binary functions.....	91
7.3.2.2.11 Date and time processing functions.....	93
7.3.2.2.12 Aggregate functions.....	100
7.3.2.3 SQL statements.....	107
7.3.2.3.1 SQL statement overview.....	107
7.3.2.3.2 ALTER SCHEMA.....	108
7.3.2.3.3 ALTER TABLE.....	108
7.3.2.3.4 CALL.....	109
7.3.2.3.5 COMMIT.....	109
7.3.2.3.6 CREATE SCHEMA.....	109
7.3.2.3.7 CREATE TABLE.....	110
7.3.2.3.8 CREATE TABLE AS.....	111
7.3.2.3.9 CREATE VIEW.....	112
7.3.2.3.10 DEALLOCATE PREPARE.....	113
7.3.2.3.11 DELETE.....	114
7.3.2.3.12 DESCRIBE.....	114
7.3.2.3.13 DESCRIBE INPUT.....	115
7.3.2.3.14 DESCRIBE OUTPUT.....	115
7.3.2.3.15 DROP SCHEMA.....	117
7.3.2.3.16 DROP TABLE.....	117
7.3.2.3.17 DROP VIEW.....	117
7.3.2.3.18 EXECUTE.....	118
7.3.2.3.19 EXPLAIN.....	118
7.3.2.3.20 EXPLAIN ANALYZE.....	120
7.3.2.3.21 GRANT.....	122
7.3.2.3.22 INSERT.....	122
7.3.2.3.23 PREPARE.....	123
7.3.2.3.24 RESET SESSION.....	123
7.3.2.3.25 REVOKE.....	124
7.3.2.3.26 ROLLBACK.....	125
7.3.2.3.27 SELECT clause.....	125
7.3.2.3.27.1 SELECT.....	125
7.3.2.3.27.2 WITH clause.....	126
7.3.2.3.27.3 GROUP BY clause.....	127
7.3.2.3.27.4 HAVING clause.....	134
7.3.2.3.27.5 Set operations.....	134
7.3.2.3.27.6 ORDER BY clause.....	136
7.3.2.3.27.7 LIMIT clause.....	136
7.3.2.3.27.8 TABLESAMPLE.....	137

7.3.2.3.27.9 UNNEST.....	138
7.3.2.3.27.10 Joins.....	139
7.3.2.3.27.11 Subquery.....	140
7.3.2.3.28 SET SESSION.....	141
7.3.2.3.29 SHOW CATALOGS.....	141
7.3.2.3.30 SHOW COLUMNS.....	142
7.3.2.3.31 SHOW CREATE TABLE.....	142
7.3.2.3.32 SHOW CREATE VIEW.....	143
7.3.2.3.33 SHOW FUNCTIONS.....	143
7.3.2.3.34 SHOW GRANTS.....	143
7.3.2.3.35 SHOW PARTITIONS.....	144
7.3.2.3.36 SHOW SCHEMAS.....	144
7.3.2.3.37 SHOW SESSION.....	145
7.3.2.3.38 SHOW TABLES.....	145
7.3.2.3.39 START TRANSACTION.....	145
7.3.2.3.40 USE.....	146
7.3.2.3.41 VALUES.....	146
7.3.3 Common connectors.....	147
7.3.3.1 Kafka connector.....	147
7.3.3.2 JMX connector.....	154
7.3.3.3 System connector.....	156
8 TensorFlow.....	159
9 Knox.....	162
10 Flume.....	166
10.1 Use Flume.....	166
10.2 Configure Flume.....	170
10.3 Use LogHub Source to move data from non-EMR clusters to HDFS of EMR clusters.....	178
11 Sqoop.....	182
12 Component authorization.....	191
12.1 HDFS authorization.....	191
12.2 YARN authorization.....	193
12.3 Hive authorization.....	199
12.4 HBase authorization.....	204
12.5 Kafka authorization.....	207
12.6 Ranger.....	213
12.6.1 Introduction to Ranger.....	213
12.6.2 Integrate Ranger into HDFS.....	214
12.6.3 Integrate Ranger into Hive.....	216
12.6.4 HBase configurations.....	220
12.6.5 Data masking in Hive.....	222
13 Kerberos authentication.....	224
13.1 Introduction to Kerberos.....	224

13.2 Authentication compatible with MIT Kerberos.....	228
13.3 RAM authentication.....	231
13.4 LDAP authentication.....	235
13.5 Execution plan authentication.....	237
13.6 Cross-region access.....	238

1 Hue

E-MapReduce currently supports [Hue](#), which you can access through Apache Knox. The following section provides an overview of how to use Hue.

Preparation

In the [#unique_4](#) cluster, [set the security group rules](#), and open port 8888.



Notice:

Set security group rules for limited IP ranges. IP 0.0.0.0/0 is not allowed to add into the security group.

Access Hue

To access Hue, complete the following steps:

1. In the EMR console, click Manage to the right of the cluster ID.
2. On the left side of the Configuration page, click Access Links and Ports.

View the password

If Hue does not have an administrator after the first running, the first user to log on is set automatically to administrator. For security, E-MapReduce generates an administrator account and password by default. The administrator account is admin.

To view the password, complete the following steps:

1. Click Manage to the right of the cluster ID.
2. In the Clusters and Services panel, click Hue.
3. Click the Configuration tab to go to the `admin_pwd` parameter. It is a random password.

Create a new account if you forget your password

If you forget your password for your Hue account, you can create a new account by completing the following steps:

1. In the cluster list page, click Manage next to the target cluster.
2. In the navigation panel on the left, click Cluster Overview.
3. In the Core Instance Group, obtain the public network IPs of some master nodes.
4. Log on to the master node through SSH.

5. Execute the following command:

```
/ opt / apps / hue / build / env / bin / hue  createsupe  ruser
```

6. Enter a new user name, e-mail, and password, and press Enter.

If Superuser created successfully is displayed, you have successfully created a new account. You can now log on to Hue with the new account.

Add or modify a configuration

1. In the cluster list page, click Manage next to the target cluster.
2. In the service list, click Hue, and then click the Configuration tab.
3. In the upper-right corner of the page, click Custom Configuration, and configure the Key and Value fields. The key must adhere to the following specifications:

```
$ section_pa  th  .$ real_key
```



Note:

- `$ real_key` is the actual key to be added, such as `hive_serve r_host`.
- In the `hue . ini` file, you can view the `$ section_pa th` before the `$ real_key`.

For example, if the `hive_serve r_host` belongs to the `[beeswax]` section, this means that the `$ section_pa th` is `beeswax`. If this is the case, the key to be added is `beeswax . hive_serve r_host`.

.

- If you need to modify the multilevel section `[desktop] -> [[ldap]] -> [[[ldap_serve rs]]] -> [[[[users]]]] -> user_name_ attr` value in the `hue . ini` file, the key to be configured is `desktop . ldap . ldap_serve rs . users . user_name_ attr`.

2 Oozie

The following section provides an overview of how to use Oozie in a E-MapReduce cluster.



Note:

E-MapReduce version 2.0.0 and later support Oozie. If you need to use Oozie in a cluster, make sure that the version you are using is 2.0.0 or higher.

Preparations

Before you create a cluster, you must first open an SSH tunnel. For more information, see [#unique_7](#).

In the following, which uses a MAC environment as an example, the IP address of the public network for the cluster's master node is assumed to be `xx.xx.xx.xx`:

1. Log on to the master node.

```
ssh root @ xx . xx . xx . xx
```

2. Enter your password.

3. Check the `id_rsa . pub` content of the local machine. Note that this is executed on the local machine, not the remote master node.

```
cat ~/.ssh / id_rsa . pub
```

4. Write the `id_rsa . pub` content of the local machine in `~/.ssh / authorized _keys` on the local master node, which is executed on the remote master node.

```
mkdir ~/.ssh /
vim ~/.ssh / authorized _keys
```

5. Copy and paste the content observed in [Step 2](#). You should now be able to log on to the master node without a password using `ssh root @ xx . xx . xx . xx .`
6. Execute the following command on the local machine to perform port forwarding:

```
ssh -i ~/.ssh / id_rsa -ND 8157 root @ xx . xx . xx . xx
```

7. Execute the following command to enable Chrome in the new terminal on the local machine:

```
/ Application / Google \ Chrome . app / Contents / MacOS /
Google \ Chrome -- proxy - server =" socks5 :// localhost : 8157
```

```
" -- host - resolver - rules =" MAP * 0 . 0 . 0 . 0 , EXCLUDE
localhost " -- user - data - dir =/ tmp
```

Access the Oozie UI interface

Access the following in Chrome to perform port forwarding: `xx.xx.xx.xx:11000/oozie`, `localhost:11000/oozie`, or intranet ip: `11000/oozie`.

Submit a workflow job

Before you run Oozie, you first have to install [Oozie's ShareLib](#).

In E-MapReduce clusters, ShareLib is installed by default for Oozie users. If you are using Oozie to submit a workflow job, you do not need to install ShareLib again.

Clusters with HA enabled use different methods to access NameNode and ResourceManager than clusters with HA disabled. Therefore, when you submit an Oozie workflow job, you need to specify a different NameNode and JobTracker (ResourceManager) in job.properties files. To do so, complete the following steps:

- Non-HA clusters

```
nameNode = hdfs :// emr - header - 1 : 9000
jobTracker = emr - header - 1 : 8032
```

- HA clusters

```
nameNode = hdfs :// emr - cluster
jobTracker = rm1 , rm2
```

In the following examples, configurations are made for both non-HA and HA clusters. For operations that do not require modification, the sample code can be used directly. For the specific format of a workflow file, see the relevant documentation on the [official Oozie website](#).

- Submit a workflow job on a non-HA cluster

1. Log on to the main master node of the cluster.

```
ssh root @ publicIp_o f_master
```

2. Download the sample code.

```
[ root @ emr - header - 1 ~]# su oozie
[ oozie @ emr - header - 1 root ]$ cd / tmp
[ oozie @ emr - header - 1 tmp ]$ wget http :// emr - sample
- projects . oss - cn - hangzhou . aliyuncs . com / oozie -
examples / oozie - examples . zip
```



```
[ oozie @ emr - header - 1 tmp ]$ unzip oozie - examples .
zip
```

3. Synchronize the Oozie workflow code to HDFS.

```
[ oozie @ emr - header - 1 tmp ]$ hadoop fs - copyFromLocal
examples / / user / oozie / examples
```

4. Submit a sample Oozie workflow job.

```
[ oozie @ emr - header - 1 tmp ]$ $ OOOIE_HOME / bin / oozie
job - config examples / apps / map - reduce / job . properties
- run
```

After submitting the job successfully, a jobId is returned, for example:

```
job : 00000000 - 1606271956 51086 - oozie - oozie - W
```

5. Go to the Oozie UI page to view the submitted Oozie workflow job.

• Submit a workflow job on an HA cluster

1. Log on to the main master node of the HA cluster.

```
ssh root @ main_maste r_ip
```

To determine the current main master node, check whether the Oozie UI can be accessed or not. By default, the Oozie server service is enabled on the main master node `xx . xx . xx . xx : 11000 / oozie`.

2. Download the sample code.

```
[ root @ emr - header - 1 ~ ]# su oozie
[ oozie @ emr - header - 1 root ]$ cd / tmp
[ oozie @ emr - header - 1 tmp ]$ wget http :// emr - sample
- projects . oss - cn - hangzhou . aliyuncs . com / oozie -
examples / oozie - examples - ha . zip
```

```
[ oozie @ emr - header - 1 tmp ]$ unzip oozie - examples - ha  
. zip
```

3. Synchronize the Oozie workflow code to HDFS.

```
[ oozie @ emr - header - 1 tmp ]$ hadoop fs - copyFromLo  
cal examples / / user / oozie / examples
```

4. Submit a sample Oozie workflow job.

```
[ oozie @ emr - header - 1 tmp ]$ $ OOZIE_HOME / bin / oozie  
job - config examples / apps / map - reduce / job . properties  
- run
```

After submitting the job successfully, a jobId is returned. This should be similar to:

```
job : 00000000 - 1606271956 51086 - oozie - oozi - W
```

5. Go to the Oozie UI page to view the submitted Oozie workflow job.

3 Zeppelin

E-MapReduce can access Zeppelin through Apache Knox.

Preparation

1. In the [Security groups](#) cluster, [set the security group rules](#), and open port 8080.
2. In Knox, add a user name and password. For more information on how to set Knox users, see [Knox](#). The user name and password are only used to log on to the various Knoxservices. They are not related to Alibaba Cloud RAM user names.



Notice:

Set security group rules for limited IP ranges. IP 0.0.0.0/0 is not allowed to add into the security group.

Access Zeppelin

To view the access links for Zeppelin, complete the following steps:

1. On the right of the cluster list page, click **Manage**.
2. In the pane on the left, click **Access Links and Ports**.

4 ZooKeeper

The [ZooKeeper](#) service is enabled in E-MapReduce clusters by default.



Note:

ZooKeeper only has 3 nodes, regardless of how many machines are currently in the cluster. More nodes are not currently supported.

Create a cluster

When you create a cluster, select the Zookeeper service in the software configuration page.

Node information

After you have created a cluster and its status is idle, in the Clusters and Services page, select ZooKeeper, and then click Component Topology to view ZooKeeper nodes. E-MapReduce enables 3 ZooKeeper nodes. The corresponding intranet IP address (2181 is the default port) of ZooKeeper nodes are indicated in the IP column for access to the ZooKeeper service.

5 Kafka

5.1 Quick start

E-MapReduce 3.4.0 and later support Kafka.

Create a Kafka cluster

When creating a cluster on E-MapReduce, set the cluster type to Kafka. A cluster containing only Kafka components is created by default. The components include basic components, as well as Zookeeper, Kafka, and KafkaManager components. Only one Kafka broker is deployed on each node. We recommend that you use a dedicated Kafka cluster instead of mixing with Hadoop services.

Ephemeral disk Kafka clusters

To better reduce unit costs and respond to larger storage needs, E-MapReduce 3.5.1 supports Kafka clusters on local disks (D1 cluster models). For more information, see [ECS models](#). Compared to cloud disks, local disk Kafka clusters have the following features:

- High-volume local SATA HDD disks with high I/O throughput, sequential read and write performance on a single disk of 190 MB/s, and up to 5 GB/s of storage I/O capability.
- Cost of local storage is 97% lower than that of SSD cloud disks.
- Higher network performance, with up to 17 Gbit/s instances of network bandwidth . This meets data interaction requirements for peak business instances.

Local disk models also have the following features:

Operation	Ephemeral disk data status	Description
Restart within the operating system/restart or force restart in the ECS console	Retained	The local ephemeral disk's storage volume is retained . Data is also retained.
Shut down within the operating system/Stop or force stop in the ECS console	Retained	The local ephemeral disk's storage volume is retained . Data is also retained.

Operation	Ephemeral disk data status	Description
Release (instances) on the console	Erased	The local ephemeral disk's storage volume is erased. Data is not retained.

**Notice:**

- When the host is down or the disk is corrupted, the data on the disk is lost.
- Do not store business data on a local ephemeral disk for a long period of time. Back up data in a timely manner and adopt a high-availability architecture. For long-term storage, we recommend that you store data on a cloud disk.

To be able to deploy Kafka on a local disk, E-MapReduce has the following default requirements:

1. `default.replication.factor = 3` indicates that the number of partitions and replicas in the topic is at least three. If a smaller number of replicas is set, the risk of data loss is increased.
2. `min.insync.replicas = 2` indicates that when the producer is required to set acks to all (-1), it is considered successful to write at least two replicas at a time.

When a local disk corruption occurs, E-MapReduce performs the following:

1. Removes the bad disk from the broker configuration, restarts Broker, and recovers the lost data from the bad disk on the other available local disks. The time it takes to perform data recovery varies according to the amount of data that has been written on the broken disk.
2. When the number of damaged machine disks is over 20%, E-MapReduce takes the initiative to migrate the machine and restore the abnormal disk.
3. If there is not enough disk space available on the current machine to recover lost data on the damaged disk, Broker is shut down abnormally. If this is the case, you can choose to clean some data, free up disk space, or restart the Broker service. You can also open a ticket with E-MapReduce for machine migration and to recover abnormal disks.

Parameter description

You can check Kafka software configurations on the E-MapReduce cluster configuration management interface.

Configuration item	Description
zookeeper.connect	Zookeeper connection address configured on Kafka.
kafka.heap.opts	Size of the heap memory of the Kafka broker.
num.io.threads	Number of the Kafka broker's I/O threads, which by default is twice the number of CPU cores.
num.network.threads	Number of the Kafka broker's network threads, which by default is the same as the number of CPU cores.

5.2 Cross-cluster access to Kafka

Typically, Kafka services are provided through a separate cluster. You need to cross clusters to access Kafka services.

Scenarios of cross-cluster access to Kafka

The scenarios of cross-cluster access to Kafka are described as follows.

- You access an EMR Kafka cluster from a private network.
- You access an EMR Kafka cluster from a public network.

Different solutions are provided based on the version of EMR.

V3.11.X and later versions

- Access Kafka from a private network

You can access Kafka services by connecting to the private IP addresses of the cluster nodes. The port number is 9092.

Make sure the networks are interconnected before you access Kafka.

- See [Interconnect classic networks and VPCs](#).
- See [Configure VPC peering connections](#).

- Access Kafka in the public network

By default, core nodes of a Kafka cluster cannot be accessed from the public network. You can perform the following steps to access a Kafka cluster over a public network.

1. Enable communication between the Kafka cluster and the host in the public network.
 - The following methods are based on a Kafka cluster that is deployed in a VPC.
 - Use Express Connect to enable the connection between the private network and the public network. For more information, see [Express Connect](#).
 - Mount the core nodes of a cluster with EIPs. The following steps are based on this method.
 - The following methods are based on a Kafka cluster that is deployed in a classic network.
 - For a Pay-As-You-Go cluster, you need to use ECS APIs. See [AllocatePublicIpAddress](#).
 - For a Subscription cluster, you can associate an EIP with the specified instance in the ECS console.
2. On the Cluster Management page, click View Details for a cluster to go to the Cluster Overview page.
3. Click Network Management. From the drop-down list, select Assign Public IP Address.
4. Configure security group rules for the Kafka cluster to specify the public IP addresses that can access the Kafka cluster. By doing this, the security of the Kafka cluster is improved. You can view the security group of the cluster and

configure the security group rules in the EMR console. [For more information, see *Typical applications of security group rules*.](#)

5. On the Cluster Overview page, click Instance State Management. From the drop-down list, select Sync Cluster Host Info.
6. On the Clusters and Services page, choose Kafka > Configuration. In the Service Configuration section, specify the value of the `kafka . public - access . enable` parameter to true.
7. Restart Kafka.
8. You can access Kafka services by connecting to the EIPs of the cluster nodes. The port number is 9093.

Earlier versions before V3.11.X

- Access Kafka from a private network

You need to configure the host information of the Kafka cluster nodes. Note: You need to configure the long domains of the Kafka cluster nodes on the client to avoid failed access to Kafka services. Example:

```
/ etc / hosts
# kafka cluster
10 . 0 . 1 . 23 emr - header - 1 . cluster - 48742
10 . 0 . 1 . 24 emr - worker - 1 . cluster - 48742
10 . 0 . 1 . 25 emr - worker - 2 . cluster - 48742
```

```
10 . 0 . 1 . 26   emr - worker - 3 . cluster - 48742
```

- Access Kafka from a public network

By default, core nodes of a Kafka cluster cannot be accessed from a public network. If you need to access the Kafka cluster from a public network, perform the following steps.

1. Enable communication between the Kafka cluster and the host in the public network.
 - The following methods are based on a Kafka cluster that is deployed in a VPC.
 - Use Express Connect to enable the connection between the private network and the public network. For more information, see [Express Connect](#).
 - Associate EIPs with the core nodes of the cluster. Perform the following steps.
 - The following methods are based on a Kafka cluster that is deployed in a classic network.
 - For a Pay-As-You-Go cluster, you need to use ECS APIs. See [AllocatePublicIpAddress](#).
 - For a Subscription cluster, you can associate an EIP with the specified instance in the ECS console.
2. In the [VPC console](#), purchase EIPs according to the number of the core nodes of the Kafka cluster.
3. Configure security group rules for the Kafka cluster to specify the public IP addresses that can access the Kafka cluster. By doing this, the security of the Kafka cluster is improved. You can view the security group of the cluster and configure the security group rules in the EMR console. [For more information, see Typical applications of security group rules.](#)
4. In the Service Configuration section, specify the value of the `listeners` . `address` . `principal` parameter to `HOST` . Restart the Kafka cluster.
5. Refer to Step 5 and configure the `hosts` file of the local client.

5.3 Kafka Ranger

With E-MapReduce 3.12.0 and later, Kafka allows you to configure permissions with Ranger.

Integrate Ranger into Kafka

To integrate Ranger into Kafka, complete the following steps:

- Enable Kafka Plugin

1. On the Cluster Management page, click Ranger in the service list to enter the Ranger Management page. Click Operation in the upper-right corner and select Enable Kafka PLUGIN.
2. You can check the progress by clicking View Operation History in the upper-right corner of the page.

- Restart Kafka broker

After enabling the Kafka plugin, you must restart the broker to make it take effect.

1. On the Cluster Management page, click the inverted triangle icon behind RANGER in the upper-left corner to switch to Kafka.
2. Click Actions in the upper-right corner of the page and select RESTART Broker.
3. You can check the progress by clicking View Operation History in the upper-right corner of the page.

- Add Kafka service on the Ranger WebUI

For more information about how to go to the Ranger WebUI, see [Ranger Introduction](#).

Add the Kafka service on the WebUI:

Configure the Kafka service:

Configure permissions

After integrating Ranger into Kafka, you can set the relevant permissions.



Notice:

In a standard cluster, Ranger automatically generates the all - topic rule after the Kafka service is added. This rule indicates that there are no restrictions on permissions. All users can perform all actions. In this case, Ranger cannot identify permissions through the user.

Here, `user_test` is used as an example to add the Publish permission:

After you add a policy, the permissions are granted to the `test` user. This user can then perform the write operation for `test`.



Note:

The policy takes effect one minute later after it is added.

5.4 Kafka SSL

E-MapReduce Kafka supports the SSL function in E-MapReduce 3.12.0 and later.

Create a cluster

For details about how to create a cluster, see [#unique_24](#).

Enable the SSL service

By default, the SSL function is not enabled for the Kafka cluster. You can enable it on the configuration page of the Kafka service.

As shown in the preceding figure, change `kafka . ssl . enable` to `true` and then restart the component.

Access Kafka from the client

You need to configure `security . protocol`, `truststore`, and `keystore` when you access Kafka through SSL. Take a standard mode cluster as an example. To run a job in a Kafka cluster, you can configure the cluster as follows:

```
security . protocol = SSL
ssl . truststore . location = / etc / ecm / kafka - conf / truststore
ssl . truststore . password = ${ password }
ssl . keystore . location = / etc / ecm / kafka - conf / keystore
ssl . keystore . password = ${ password }
```

If you are running a job in an environment other than a Kafka cluster, copy the truststore and keystore files (in the `/ etc / ecm / kafka - conf` directory on

any node of the cluster) in the Kafka cluster to the running environment and add configurations accordingly.

Take the producer and consumer programs in Kafka as an example.

1. Create the configuration file `ssl.properties` and add configuration items.

```
security.protocol = SSL
ssl.truststore.location = /etc/ecm/kafka-conf/truststore
ssl.truststore.password = ${password}
ssl.keystore.location = /etc/ecm/kafka-conf/keystore
ssl.keystore.password = ${password}
```

2. Create a topic.

```
kafka-topics.sh --zookeeper emr-header-1:2181/kafka-1.0.1 --replication-factor 2 --partitions 100 --topic test --create
```

3. Use an SSL configuration file to generate data.

```
kafka-producer-perf-test.sh --topic test --num-records 123456 --throughput 10000 --record-size 1024 --producer-props bootstrap.servers=emr-worker-1:9092 --producer.config ssl.properties
```

4. Use an SSL configuration file to consume data.

```
kafka-consumer-perf-test.sh --broker-list emr-worker-1:9092 --messages 100000000 --topic test --consumer.config ssl.properties
```

5.5 Kafka Manager

E-MapReduce 3.4.0 and later support Kafka Manager for use in managing Kafka clusters.

Procedure



Notice:

Kafka Manager software is installed by default and the Kafka Manager authentication function is enabled when a Kafka cluster is created. We strongly recommend that you change the default password when using Kafka Manager for the first time and access Kafka Manager through the SSH tunnel. We do not recommend that you expose Port 8085 to the public network unless an IP address whitelist is configured to avoid data leakage.

- We recommend that you access the web page through the SSH tunnel. For more information, see [#unique_7](#).
- Access `http://localhost:8085`.
- Enter your user name and password. Refer to the configuration information of Kafka Manager.
- Add an existing Kafka cluster and make sure that the Zookeeper address of the Kafka cluster is correct. For more information, see the configuration information of Kafka Manager. Select the corresponding Kafka version. We recommend that you enable the JMX function.
- Common Kafka functions are available immediately after you create a Kafka cluster.

5.6 Common Kafka problems

This section describes two common issues with Kafka.

- `Error while executing topic command : Replication factor : 1 larger than available brokers : 0 .`

Common causes:

- A fault occurs in the Kafka service and the cluster broker process exits. You need to use logs to troubleshoot the fault.
- The ZooKeeper address of the Kafka service is incorrect. View and use the `Zookeeper.connect` configuration item on the Kafka configuration management page.
- `java . net . BindException : Address already in use (Bind failed)`

You may encounter this exception when you use Kafka command line tools. This is typically caused by the unavailability of the JMX port. You can specify a JMX port manually before using the command line. For example:

```
JMX_PORT = 10101 kafka - topics -- zookeeper emr - header - 1
: 2181 / kafka - 1 . 0 . 0 -- list
```

6 Druid

6.1 Introduction to Druid

Druid is a column-oriented, open-source, distributed data store used to query and analyze issues in large datasets in real time.

Basic features

Druid has the following features:

- Sub-second OLAP queries, including multi-dimensional filtering, ad-hoc attribute grouping, and fast data aggregation.
- Real-time data consumption, collection, and querying.
- Efficient multi-tenant capability, which enables thousands of users to perform searches online at the same time.
- Strong scalability, which supports the fast processing of PB-level data, 100 billion-level events, and thousands of concurrent queries per second.
- Extremely high availability and support for rolling upgrades.

Usage scenarios

Real-time data analysis is the most typical usage scenario for Druid and covers a wide range of areas, including:

- Real-time indicator monitoring
- Model recommendations
- Advertisement platforms
- Model searches

These scenarios involve large amounts of data, and the requirement for time delay in data querying is high. In real-time indicator monitoring, problems need to be detected at the moment of occurrence so that you can be warned as soon as possible. In the recommendation model, user behavior data needs to be collected in real time and sent to the recommendation system promptly. In just a few clicks, the system is able to identify your search intent and recommend more appropriate results in future searches.

Architecture

Druid has an excellent architectural design with multiple components working together to complete a series of processes, such as data collection, indexing, storage, and querying.

The following figure shows the components contained in the Druid working-layer (for data indexing and data querying).

- The real-time component is responsible for the real-time data collection.
- In the broker phase, query tasks are distributed, and the results are collected and returned to you.
- The historical node is responsible for the storage of historical data after indexing. The data is stored in deep storage. Deep storage can be either local or a distributed file system, such as HDFS.
- The indexing service consists of two components (not shown in the figure).
 - The Overlord component is responsible for managing and distributing indexing tasks.
 - The MiddleManager component is responsible for executing indexing tasks.

The following figure shows the components involved in the management layer of Druid segments (Druid index file).

- The ZooKeeper component is responsible for storing the status of the cluster and discovering components, such as the topology information of the cluster, election of the Overlord leader, and management of the indexing task.
- The Coordinator component is responsible for managing segments, such as the downloading and deletion of the segments and balancing them with historical components.
- The Metadata storage component is responsible for storing the meta-information of segments and managing all kinds of persistent or temporary data in the cluster, such as configuration information and audit information.

Product advantages

E-MapReduce Druid has improved a lot based on open-source Druid, including integration with E-MapReduce and the peripheral Alibaba Cloud ecosystem, easy

monitoring and operation support, and easy-to-use product interfaces. You can use it immediately after purchase. It does not need 24/7 operation and maintenance.

E-MapReduce Druid supports the following features:

- Using OSS as deep storage
- Using OSS files as data sources for indexing in batches
- Supporting indexing the streaming data from Log Service and providing high reliability and exactly-once semantics
- Using RDS to store metadata
- Integrating with Superset tools
- Easy scale up and scale down (scale down is for task node)
- Diversified monitoring indicators and alarm rules
- Bad node migration
- High-security mode
- HA

6.2 Quick start

E-MapReduce V3.11.0 and later versions support Druid as a cluster type.

The use of Druid as a separate cluster type (instead of adding Druid service to the Hadoop cluster) is mainly based on the following reasons:

- Druid can be used independently of Hadoop.
- Druid has high memory requirements when there is a large amount of data, especially for Broker and Historical nodes. Druid is not controlled by Yarn and will compete for resources during multi-service operation.
- As the infrastructure, the node number of a Hadoop cluster can be relatively large, whereas a Druid cluster can be relatively small. The work is more flexible if they work together.

Create a Druid cluster

Select Druid as the cluster type when you create a cluster. You can select HDFS and Yarn when creating a Druid cluster. The HDFS and Yarn in the Druid cluster are for testing only, as described at the beginning of this guide. We recommend that you use a dedicated Hadoop cluster as the production environment.

Configure a cluster

- Configure the cluster to use HDFS as the deep storage of Druid

For a standalone Druid cluster, you may need to store your index data in the HDFS of another Hadoop cluster. Therefore, you need to complete related settings for the connectivity between the two clusters (for details, see [Interaction with Hadoop clusters](#)). Then you need to configure the following items on the configuration page of Druid and restart the service. The configuration items are in `common.runtime` of the configuration page.

- `druid.storage.type`: `hdfs`
- `druid.storage.storageDirectory`: (the `hdfs` directory must be a full one, such as `hdfs://emr-header-1.cluster-xxxxxxx:9000/druid/segments`.)



Note:

If the Hadoop cluster is an HA cluster, you must change `emr-header-1.cluster-xxxxx:9000` to `emr-cluster`, or change port 9000 to port 8020.

- Use OSS as the deep storage of Druid

E-MapReduce Druid supports the use of OSS as deep storage. Due to the AccessKey-free capability of E-MapReduce, Druid can automatically get access to OSS without the need to configure the AccessKey. Because the OSS function of HDFS

enables Druid to have access to OSS, `druid . storage . type` still needs to be configured as HDFS during the configuration process.

- `druid . storage . type : hdfs`
- `druid . storage . storageDir ectory : (for example, oss :// emr - druid - cn - hangzhou / segments)`

Because the OSS function of HDFS enables Druid to have access to OSS, you need to select one of the following two scenarios:

- Choose to install HDFS when you create a cluster. Then the system is automatically configured. (After HDFS is installed, you can choose not to use it, disable it, or use it for testing purposes only.)
- Create `hdfs - site . xml` in the configuration directory of Druid / `etc / ecm / druid - conf / druid / _common /`, the content is as follows, and then copy the file to the same directory of all nodes:

```
<? xml version =" 1 . 0 " ? >
  < configurat ion >
    < property >
      < name > fs . oss . impl </ name >
      < value > com . aliyun . fs . oss . nat . NativeOssF
ileSystem </ value >
    </ property >
    < property >
      < name > fs . oss . buffer . dirs </ name >
      < value > file :/// mnt / disk1 / data ,...</ value >
    </ property >
    < property >
      < name > fs . oss . impl . disable . cache </ name >
      < value > true </ value >
    </ property >
  </ configurat ion >
```

The `fs . oss . buffer . dirs` can be set to multiple paths.

- Use RDS to save Druid metadata

Use the MySQL database on header-1 node to save Druid metadata. You can also use the Alibaba Cloud RDS to save the metadata.

The following uses RDS MySQL as an example to demonstrate the configuration.

Before you configure it, make sure that:

- The RDS MySQL instance has been created.
- A separate account has been created for Druid to access RDS MySQL (root is not recommended). This example uses account name druid and password druidpw.
- Create a separate MySQL database for Druid metadata. Suppose the database is called druiddb.
- Make sure that account Druid has permission to access druiddb.

In the E-MapReduce console, click Manage behind the Druid cluster you want to configure. Click the Druid service, and then select the Configuration tab to find the `common.runtime` configuration file. Click Custom Configuration to add the following three configuration items:

- `druid.metadata.storage.connector.connectURI`, where the value is: `jdbc:mysql://rm-xxxxx.mysql.rds.aliyuncs.com:3306/druiddb`
- `druid.metadata.storage.connector.user`, where the value is `druid`.
- `druid.metadata.storage.connector.password`, where the value is `druidpw`.

Choose Save > > Deploy Client Configuration > Restart All Components to make the configuration take effect.

Log on to the RDS console to view the tables created by druiddb. You will find tables automatically created by druid.

- Service memory configuration

The memory of the Druid service consists of the heap memory (configured through `jvm.config`), and direct memory (configured through `jvm.config` and `runtime.properties`). E-MapReduce will automatically generate a set of configurations when

you create a cluster. However, in some cases, you may still need to configure the memory.

To adjust the service memory configuration, you can access the cluster services through the E-MapReduce console, and perform related operations on the page.



Note:

For direct memory, make sure that:

```
- XX : MaxDirectMemorySize is greater than or equal
    to druid . processing . buffer . sizeBytes * ( druid .
    processing . numMergeBuffers + druid . processing . numThreads
    + 1 ).
```

Visit the Druid web page

Druid comes with two Web pages:

- **Overlord:** `http://emr-header-1.cluster-1234:18090` is used to view the running status of tasks.
- **Coordinator:** `http://emr-header-1.cluster-1234:18081` is used to view the storage status of segments.

EMR provides three methods to access Druid Web pages:

- On the cluster management page, click Access Link and port, locate the Druid overlord or Druid coordinator link, and click the link to enter (recommended method, EMR-3.20.0, and later versions).
- Use [SSH tunneling](#) to create an SSH tunnel and enable proxy browser access.
- Access through public IP + Port, as shown in figure `Http://123.123.123.123:18090` (Not recommended. use security group settings to properly control cluster access through the public network).

Batch index

- Interaction with Hadoop clusters

If you select HDFS and Yarn (with their own Hadoop clusters) when creating Druid clusters, the system will automatically configure the interaction between HDFS and Yarn. The following example shows how to configure the interaction between a standalone Druid cluster and a standalone Hadoop cluster. It is assumed that the Druid cluster ID is 1234, and the Hadoop cluster ID is 5678. In addition, read

through and follow the instructions strictly. The clusters may not work as expected because of slightly improper operation.

For the interaction with standard-mode Hadoop clusters, perform the following operations:

1. Ensure the communication between the two clusters. (Each cluster is associated with a different security group, and access rules are configured for the two security groups.)
2. Put `core-site.xml`, `hdfs-site.xml`, `yarn-site.xml`, `mapred-site.xml` of `/ etc / ecm / hadoop - conf` of the Hadoop cluster in the `/ etc / ecm / duird - conf / druid / _common` directory on each node of the Druid cluster. (If you select the built-in Hadoop when you create the cluster, several soft links in this directory will map to the configuration of the Hadoop service of E-MapReduce. Remove these soft links first.)
3. Write the hosts of the Hadoop cluster to the hosts list on the Druid cluster. Note that the hostname of the Hadoop cluster should be in the form of a long name, such as `emr-header-1.cluster-xxxxxxx`. You are advised to put the hosts of Hadoop behind the hosts of the Druid cluster, such as:

```
...
10 . 157 . 201 . 36      emr - as . cn - hangzhou . aliyuncs .
com
10 . 157 . 64 . 5       eas . cn - hangzhou . emr . aliyuncs .
com
192 . 168 . 142 . 255   emr - worker - 1 . cluster - 1234   emr -
worker - 1   emr - header - 2 . cluster - 1234   emr - header - 2
iZbp1h9g7b oqo9x23qbi fiZ
192 . 168 . 143 . 0     emr - worker - 2 . cluster - 1234   emr -
worker - 2   emr - header - 3 . cluster - 1234   emr - header - 3
iZbp1eaa58 19tkjx55yr 9xZ
192 . 168 . 142 . 254   emr - header - 1 . cluster - 1234   emr -
header - 1   iZbp1e3zvu vnmakmsjer 2uZ
For Hadoop clusters in high - security mode , perform
the following operations :
192 . 168 . 143 . 6     emr - worker - 1 . cluster - 5678   emr -
worker - 1   emr - header - 2 . cluster - 5678   emr - header - 2
iZbp195rj7 zvx8qar4f6 b0Z
192 . 168 . 143 . 7     emr - worker - 2 . cluster - 5678   emr -
worker - 2   emr - header - 3 . cluster - 5678   emr - header - 3
iZbp15vy2r sxoegki4qh dpZ
```

```
192 . 168 . 143 . 5    emr - header - 1 . cluster - 5678    emr -
header - 1    iZbp10tx4e    gw3wfnh5oi    i1Z
```

For Hadoop clusters in high-security mode, perform the following operations:

1. Ensure the communication between the two clusters. (Each cluster is associated with a different security group, and access rules are configured for the two security groups.)
2. Put `core-site.xml`, `hdfs-site.xml`, `yarn-site.xml`, `mapred-site.xml` of `/ etc / ecm / hadoop - conf` of the Hadoop cluster in the `/ etc / ecm / duird - conf / druid / _common` directory on each node of the Druid cluster. (If you select the built-in Hadoop when creating a cluster, several soft links in this directory will point to the configuration with Hadoop. Remove these soft links first.) Modify `hadoop . security . authentica tion . use . has` in `core-site.xml` to `false` . (This configuration is completed on the client to enable AccessKey authentication for users. If Kerberos authentication is used, disable AccessKey authentication.)
3. Write the hosts of the Hadoop cluster to the hosts list of each node on the Druid cluster. Note that the hostname of the Hadoop cluster should be in the form of a long name, such as `emr-header-1.cluster-xxxxxxx`. You are advised to put the hosts of Hadoop behind the hosts of the Druid cluster.
4. Set Kerberos cross-domain mutual trust between the two clusters. For more information, see [#unique_31](#).
5. Create a local Druid account (`useradd-m-g hadoop`) on all nodes of the Hadoop cluster, or set `druid.auth.authenticator.kerberos.authtomate` to create a mapping rule for the Kerberos account to the local account. (For specific pre-release rules, see [here](#).) This method is recommended because it is easy to operate without errors.



Note:

In Hadoop cluster of the high-security mode , all Hadoop commands must be run from a local account. By default, this local account needs to have the same name as the principal. Yarn also supports mapping a principal to a local account.

6. Restart the Druid service.

- Use Hadoop to index batch data

Druid comes with an example named wikiticker, which is located in the

`${DRUID_HOME}/quickstart/tutorial` path. `${DRUID_HOME}` is set to `/usr /`

`lib / druid - current` by default. Each line of the wikiticker document

(wikiticker-2015-09-12-sampled.json.gz) is a record. Each record is a json object.

The format is as follows:

```
``` json
{
 " time ": " 2015 - 09 - 12T00 : 46 : 58 . 771Z ",
 " channel ": "# en . wikipedia ",
 " cityName ": null ,
 " comment ": " added project ",
 " countryIso Code ": null ,
 " countryName ": null ,
 " isAnonymous ": false ,
 " isMinor ": false ,
 " isNew ": false ,
 " isRobot ": false ,
 " isUnpatrolled ": false ,
 " metroCode ": null ,
 " namespace ": " Talk ",
 " page ": " Talk : Oswald Tilghman ",
 " regionIsoCode ": null ,
 " regionName ": null ,
 " user ": " GELongstreet ",
 " delta ": 36 ,
 " added ": 36 ,
 " deleted ": 0
},
},
```
```

To use Hadoop to create index for batch data, perform the following steps:

1. Decompress the compressed file and place it in a directory of HDFS (such as:

`hdfs :// emr - header - 1 . cluster - 5678 : 9000 / druid`). Run the

following command on the Hadoop Cluster.

```
### If you are operating on a standalone Hadoop
cluster , copy a druid . keytab to Hadoop cluster
after the mutual trust is established between the
two clusters , and run the kinit command .
kinit - kt / etc / ecm / druid - conf / druid . keytab
druid
###
hdfs dfs - mkdir hdfs :// emr - header - 1 . cluster - 5678
: 9000 / druid
hdfs dfs - put ${ DRUID_HOME }/ quickstart / wikiticker -
2015 - 09 - 16 - sampled . json hdfs :// emr - header - 1 .
cluster - 5678 : 9000 / druid
```



Note:

- **Modify** `hadoop . security . authentication . use . has in / etc / ecm / hadoop - conf / core - site . xml` to **false** before running HDFS command for a high-security mode cluster.
- Make sure that you have created a Linux account named **druid** on each node of the Hadoop cluster.

2. Use the following configurations to prepare a file for data indexing. The file path is set to `${DRUID_HOME}/quickstart/tutorial/wikiticker-index.json`.

```
{
  " type " : " index_hado  op ",
  " spec " : {
    " ioConfig " : {
      " type " : " hadoop ",
      " inputSpec " : {
        " type " : " static ",
        " paths " : " hdfs :// emr - header - 1 . cluster
- 5678 : 9000 / druid / wikiticker - 2015 - 09 - 16 - sampled .
json "
      }
    },
    " dataSchema " : {
      " dataSource " : " wikiticker ",
      " granularit ySpec " : {
        " type " : " uniform ",
        " segmentGra nularity " : " day ",
        " queryGranu larity " : " none ",
        " intervals " : [ " 2015 - 09 - 12 / 2015 - 09 -
13 " ]
      },
      " parser " : {
        " type " : " hadooppyStr ing ",
        " parseSpec " : {
          " format " : " json ",
          " dimensions Spec " : {
            " dimensions " : [
              " channel ",
              " cityName ",
              " comment ",
              " countryIso Code ",
              " countryNam e ",
              " isAnonymou s ",
              " isMinor ",
              " isNew ",
              " isRobot ",
              " isUnpatrol led ",
              " metroCode ",
              " namespace ",
              " page ",
              " regionIsoC ode ",
              " regionName ",
              " user "
            ]
          },
          " timestampS pec " : {
            " format " : " auto ",
            " column " : " time "
          }
        }
      }
    }
  }
}
```

```

    },
    "metricsSpec" : [
      {
        "name" : "count",
        "type" : "count"
      },
      {
        "name" : "added",
        "type" : "longSum",
        "fieldName" : "added"
      },
      {
        "name" : "deleted",
        "type" : "longSum",
        "fieldName" : "deleted"
      },
      {
        "name" : "delta",
        "type" : "longSum",
        "fieldName" : "delta"
      },
      {
        "name" : "user_unique",
        "type" : "hyperUnique",
        "fieldName" : "user"
      }
    ],
    "tuningConfig" : {
      "type" : "hadoop",
      "partitionsSpec" : {
        "type" : "hashed",
        "targetPartitionSize" : 5000000
      },
      "jobProperties" : {
        "mapreduce.job.classloader" : "true"
      }
    },
    "hadoopDependencyCoordinates" : ["org.apache.hadoop:
: hadoop-client:2.7.2"]
  }
}

```

**Note:**

- spec.ioConfig.type is set to hadoop.
- spec.ioConfig.inputSpec.paths is the path of the input file.
- tuningConfig.type is set to hadoop.
- tuningConfig.jobProperties sets the classloader of the mapreduce job.

- `hadoopDependencyCoordinator` develops the version of Hadoop client.

3. Run the batch index command on the Druid cluster.

```
cd ${ DRUID_HOME }
curl -- negotiate - u : druid - b ~/ cookies - c ~/
cookies - XPOST - H ' Content - Type : applicatio n / json
' - d @ quickstart / wikiticker - index . json http :// emr -
header - 1 . cluster - 1234 : 18090 / druid / indexer / v1 / task
```

The `-- negotiate`, `- u`, `- b`, and `- c` options are for secure Druid clusters.

The Overlord port number is 18090 by default.

4. View the running state of the jobs.

Enter `http :// emr - header - 1 . cluster - 1234 : 18090 / console . html` in your browser's address bar to view the running status of jobs.

5. Query the data based on Druid syntax.

Druid has its own query syntax. You need to prepare a json-formatted query file that describes how you want to query. A topN query to the wikiticker data is as follows `${DRUID_HOME}/quickstart/wikiticker-top-pages.json`:

```
{
  " queryType " : " topN ",
  " dataSource " : " wikiticker ",
  " intervals " : [ " 2015 - 09 - 12 / 2015 - 09 - 13 " ],
  " granularity " : " all ",
  " dimension " : " page ",
  " metric " : " edits ",
  " threshold " : 25 ,
  " aggregations " : [
    {
      " type " : " longSum ",
      " name " : " edits ",
      " fieldName " : " count "
    }
  ]
}
```

You can check the results of the query by running the following command:

```
cd ${ DRUID_HOME }
curl -- negotiate - u : druid - b ~/ cookies - c ~/
cookies - XPOST - H ' Content - Type : applicatio n / json '
- d @ quickstart / wikiticker - top - pages . json ' http ://
```

```
emr - header - 1 . cluster - 1234 : 18082 / druid / v2 /? pretty
```

Note that the items such as `- negotiate`、`- u`、`- b`、`- c` are for Druid clusters in the high-security mode. You can check the results of a specific query in normal cases.

Real-time index

For Indexing data from a Kafka cluster to a Druid cluster in real time, we recommend that you use the Kafka Indexing Service extension to ensure high reliability and support exactly-once semantics. See the Use Druid Kafka Indexing Service to consume Kafka data in real time section in [Kafka Indexing Service](#).

If your data is real-time accessed by Alibaba Cloud log Service (SLS) and you want to use Druid to index the data in real time, we provide the SLS Indexing Service extension. Using SLS Indexing Service avoids the overhead of creating and maintaining a Kafka cluster. SLS Indexing Service provides high-reliability and exactly-once semantics like Kafka Indexing Service. Here, you can use SLS as a Kafka.

For other methods, such as Flink, Storm, and Spark Streaming, we recommend that you use the Tranquility client to push data to the Druid cluster. For details, see [#unique_33](#).

Kafka Indexing Service and SLS Indexing Service are similar. They pull data from the data source to the Druid cluster in pull mode, and provide high reliability and exactly-once semantics; tranquility pushes data to Druid for indexing. Tranquility does not provide Exactly-once semantics. Therefore, if you have such requirements, you must resolve them yourself.

Analyze the indexing failure

When indexing fails, the following troubleshooting steps are typically followed:

- For batch data index

1. If curl returns an error directly, or no value returns, check the input file format. Or add a `-v` parameter to curl to observe the value returned from the REST API.
2. Observe the execution of the jobs on the Overlord page. If it fails, view the logs on the page.
3. In many cases, logs are not generated. In the case of a Hadoop job, open the Yarn page to check whether there is an index job generated, and view the job execution log.
4. If no errors are found, you need to log on to the Druid cluster, and view the execution logs of Overlord (at `/mnt/disk1/log/druid/overlord-emr-header-1.cluster-xxxx.log`). If it is an HA cluster, check the Overlord that you submitted the job to.
5. If the job has been submitted to Middlemanager, but a failure is returned, you need to view the worker that the job is submitted to in Overlord, and log on to the worker node to view the Middlemanager logs in `/mnt/disk1/log/druid/middleManager-emr-header-1.cluster-xxxx.log`.

- For Kafka Indexing Service and SLS Indexing Service

1. First, view the Overlord Web page `Http://emr-header-1:18090`, check the running status of the Supervisor, and check whether payload is valid.
2. View the log of the failed task.
3. If you cannot identify the cause of failure from the task log, you need to start with the Overlord log to troubleshoot the problem. See the last two steps in the [Batch index](#) section.

- For real-time Tranquility index

Check the Tranquility logs to see if the message is received or dropped.

The remaining troubleshooting steps are the same as Step 2 to Step 5 in the batch index section.

Most of the errors are cluster configuration issues and job problems. Cluster configuration errors are about memory parameters, cross-cluster connection, access to clusters in high-security mode, and principals. Job errors are about the format of the job description files, input data parsing, and other job-related configuration issues (such as `ioConfig`).

6.3 Ingestion Spec

This section briefly introduces Ingestion Spec, the description file of the index data.

Ingestion Spec is a unified description of the format of the data being indexed and how it is indexed by Druid. It is a JSON file, which consists of three parts:

```
{
  " dataSchema " : {...},
  " ioConfig " : {...},
  " tuningConf ig " : {...}
}
```

| Key | Format | Description | Required |
|--------------|-------------|--|----------|
| dataSchema | JSON object | Describes the schema information of the data you want to consume. dataSchema is fixed and does not change with the way in which data is consumed. | Yes |
| ioConfig | JSON object | Describes the source and destination of the data you want to consume. If the consumption method of the data is different, ioConfig is also different. | Yes |
| tuningConfig | JSON object | Configures the parameters of the data you want to consume. If the consumption method of the data is different, the adjustable parameters are also different. | No |

dataSchema

dataSchema describes the format of the data and how to parse the data. The typical structure is as follows:

```
{
  " dataSoruce ": < name_of_dataSource >,
  " parser ": {
    " type ": <>,
    " parseSpec ": {
      " format ": <>,
      " timestampSpec ": {},
      " dimensionsSpec ": {}
    }
  },
  " metricsSpec ": {},
  " granularitySpec ": {}
}
```

}

| Key | Format | Description | Required |
|-----------------|-----------------------|---|----------|
| dataSource | String | Name of the data source. | Yes |
| parser | JSON object | How the data is parsed. | Yes |
| metricsSpec | Array of JSON objects | Aggregator list. | Yes |
| granularitySpec | JSON object | Data aggregation settings, such as creating segments and aggregation granularity. | Yes |

- parser

parser determines how your data is parsed correctly. metricsSpec defines how the data is clustered for calculation. granularitySpec defines the granularity of the data fragmentation and the granularity of the query.

There are two types of parser: string and hadoopstring. The latter is used for Hadoop index jobs. ParseSpec is a specific definition of data format resolution.

| Key | Format | Description | Required |
|----------------|-------------|--|----------|
| type | String | The data format can be json, jsonLowercase, csv, or tsv. | Yes |
| timestampSpec | JSON object | Timestamp and timestamp type . | Yes |
| dimensionsSpec | JSON object | The dimension of the data (columns are included). | Yes |

For different data formats, additional parseSpec options may exist. The following table describes timestampSpec and dimensionsSpec.

| Key | Format | Description | Required |
|--------|--------|---|----------|
| column | String | Columns corresponding to the timestamp. | Yes |

| Key | Format | Description | Required |
|--------|--------|---|----------|
| format | String | The timestamp type can be ISO, millis, POSIX, auto, or whatever is supported by joda time . | Yes |

| Key | Format | Description | Required |
|---------------------|-----------------------|---|----------|
| dimensions | JSON array | Describes which dimensions the data contains. Each dimension can be just a string . You can also specify the attribute for the dimension . For example, the type of dimensions: [dimension1, dimension2, {type: long, name : dimension3}] is string by default. | Yes |
| dimensionExclusions | Array of JSON strings | Dimension to be deleted when data is consumed. | No |
| spatialDimensions | Array of JSON objects | Spatial dimension. | No |

- metricsSpec

MetricsSpec is an array of JSON objects. It defines several aggregators. Aggregators typically have the following structures:

```

{
  " type ": < type >,
  " name ": < output_name >,
  " fieldName ": < metric_name >
},

```

The following commonly used aggregators are provided:

| Type | Type optional |
|------------|--|
| count | count |
| sum | longSum, doubleSum, floatSum |
| min/max | longMin/longMax, doubleMin/doubleMax, floatMin/floatMax |
| first/last | longFirst/longLast, doubleFirst/doubleLast, floatFirst/floatLast |

| Type | Type optional |
|-------------|---------------|
| javascript | javascript |
| cardinality | cardinality |
| hyperUnique | hyperUnique |



Note:

The last three types in the table are advanced aggregators. For information about how to use them, see [Druid official documents](#).

· granularitySpec

Two aggregation modes are supported: uniform and arbitrary. The uniform mode aggregates data with a fixed interval of time. The arbitrary mode tries to make sure that each of the segments has the same size, but the time interval for aggregation is not fixed. Uniform is the current default option.

| Key | Format | Description | Required |
|--------------------|------------|--|--|
| segmentGranularity | String | Segment granularity Uniform type. The default is DAY. | No. |
| queryGranularity | String | Minimum data aggregation granularity for query. The default is true. | No |
| rollup | Bool value | Aggregate or not. | No. |
| intervals | String | Time interval of data consumption. | It is Yes for batch and No for realtime. |

ioConfig

ioConfig describes the data source. An example of Hadoop index is as follows:

```
{
  " type ": " hadoop ",
  " inputSpec ": {
    " type ": " static ",
    " paths ": " hdfs :// emr - header - 1 . cluster - 6789 : 9000
/ druid / quickstart / wikipicker - 2015 - 09 - 16 - sampled . json "
  }
}
```



Note:

This part is not required for streaming data that is processed through Tranquility.

TuningConfig

TuningConfig refers to additional settings. For example, you can specify MapReduce parameters to use Hadoop to create an index for batch data. The contents of tuningConfig may vary based on the data source. For more information, see the example file or official document of this service.

6.4 Kafka Indexing Service

This section describes how to use Druid Kafka Indexing Service in E-MapReduce to ingest Kafka data in real time.

The Kafka Indexing Service is an extension launched by Druid to ingest Kafka data in real time using Druid's indexing service. The extension enables supervisors in Overlord which start some indexing tasks in Middlemanager. These tasks connect to the Kafka cluster to ingest the topic data and complete the index creation. You need to prepare a data ingestion format file and manually start the supervisor through the RESTful API.

Interaction with the Kafka cluster

See the introduction in [Tranquility](#).

Use Druid's Kafka Indexing Service to ingest Kafka data in real time

1. Run the following command on the Kafka cluster (or gateway) to create a topic named metrics.

```
-- If the Kafka high - security mode is enabled :
export KAFKA_OPTS="-Djava.security.auth.login.config
=/etc/ecm/kafka-conf/kafka_client_jaas.conf"
--
kafka-topics.sh --create --zookeeper emr-header-1:
2181,emr-header-2,emr-header-3/kafka-1.0.0 --
partitions 1 --replication-factor 1 --topic metrics
```

You can adjust the parameters based on your needs. The `/kafka-1.0.0` section of the `--zookeeper` parameter is path, and you can see the value of the `zookeeper.connect` on the Kafka service Configuration page of the Kafka cluster. If you build your own Kafka cluster, the parameter `--zookeeper` can be changed according to your actual configuration.

2. Define the data format description file for the data source. Name it `metrics-kafka.json` and place it in the current directory (or another directory that you have specified).

```
{
  " type ": " kafka ",
  " dataSchema ": {
    " dataSource ": " metrics - kafka ",
    " parser ": {
      " type ": " string ",
      " parseSpec ": {
        " timestampSpec ": {
          " column ": " time ",
          " format ": " auto "
        },
        " dimensionsSpec ": {
          " dimensions ": [" url ", " user " ]
        },
        " format ": " json "
      }
    },
    " granularitySpec ": {
      " type ": " uniform ",
      " segmentGranularity ": " hour ",
      " queryGranularity ": " none "
    },
    " metricsSpec ": [ {
      " type ": " count ",
      " name ": " views "
    },
    {
      " name ": " latencyMs ",
      " type ": " doubleSum ",
      " fieldName ": " latencyMs "
    }
  ],
  " ioConfig ": {
    " topic ": " metrics ",
    " consumerProperties ": {
      " bootstrap.servers ": " emr - worker - 1 . cluster -
xxxxxxxx : 9092 ( the bootstrap.servers of your Kafka
clusters )",
      " group.id ": " kafka - indexing - service ",
      " security.protocol ": " SASL_PLAINTEXT ",
      " sasl.mechanism ": " GSSAPI "
    },
    " taskCount ": 1 ,
    " replicas ": 1
    " taskDuration ": " PT1H "
  },
  " tuningConfig ": {
    " type ": " Kafka ",
    " maxRowsInMemory ": " 100000 "
  }
}
```



Note:

`ioConfig.consumerProperties.security.protocol` and `ioConfig.consumerProperties.sasl.mechanism` are security-related options and are not required for standard mode Kafka clusters.

3. Run the following command to add a Kafka supervisor.

```
curl --negotiate -u :druid -b ~/cookies -c ~/cookies
-XPOST -H 'Content-Type: application/json' -d @
metrics-kafka.json http://emr-header-1.cluster-1234
:18090/druid/indexer/v1/supervisor
```

The `--negotiate`, `-u`, `-b`, and `-c` options are for high-security mode Druid clusters.

4. Enable a console producer on the Kafka cluster.

```
-- If the high-security mode of Kafka is enabled :
export KAFKA_OPTS="-Djava.security.auth.login.config
=/etc/ecm/kafka-conf/kafka_client_jaas.conf"
echo -e "security.protocol=SASL_PLAINTEXT\nsasl
mechanism=GSSAPI" > /tmp/Kafka/producer.conf
--
Kafka-console-producer.sh --producer.config /tmp/
kafka/producer.conf --broker-list emr-worker-1:9092
,emr-worker-2:9092,emr-worker-3:9092 --topic
metrics
>
```

The `--producer.config /tmp/Kafka/producer.conf` option is for high-security mode Kafka clusters.

5. Enter data at the command prompt of `kafka-console-producer`.

```
{"time": "2018-03-06T09:57:58Z", "url": "/foo/bar",
"user": "alice", "latencyMs": 32}
{"time": "2018-03-06T09:57:59Z", "url": "/", "user":
"bob", "latencyMs": 11}
{"time": "2018-03-06T09:58:00Z", "url": "/foo/bar",
"user": "bob", "latencyMs": 45}
```

The timestamp can be generated with the following Python command:

```
python -c 'import datetime; print(datetime.datetime.
utcnow().strftime("%Y-%m-%dT%H:%M:%SZ"))'
```

6. Prepare a query file named `metrics-search.json`.

```
{
  "queryType": "search",
  "dataSource": "metrics-kafka",
  "intervals": ["2018-03-06T00:00:00.000/2018-03-08T00:00:00.000"],
  "granularity": "all",
  "searchDimensions": [
    "url",
    "user"
  ]
}
```

```

    ],
    "query": {
      "type": "insensitive_contains",
      "value": "bob"
    }
  }
}

```

7. Execute the query on the master node of the Druid cluster.

```

curl --negotiate -u :Druid -b ~/cookies -c ~/cookies \
  -XPOST -H 'Content-Type: application/json' -d '@metrics-search.json' http://emr-header-1.cluster-1234:8082/druid/v2/?pretty

```

The `--negotiate`, `-u`, `-b`, and `-c` options are for high-security mode Druid clusters.

8. You will see a query result similar to the following:

```

[ {
  "timestamp": "2018-03-06T09:00:00.000Z",
  "result": {
    "dimension": "user",
    "value": "bob",
    "count": 2
  }
} ]

```

6.5 LOG Indexing Service

LOG Indexing Service is a Druid plug-in launched by EMR and is used to consume data from Log Service.

Background

LOG Indexing Service consumes data in a similar way as Kafka Indexing Service and supports exactly-once semantics. LOG Indexing Service has advantages of both Log Service and Kafka Indexing Service.

- Provides various convenient methods for collecting data to Log Service.
- Eliminates the need for Kafka clusters, which shortens the path of the data flow.
- Supports exactly-once semantics.
- Retries failed jobs to ensure reliability for data consumption, and allows for cluster restarts and service updates without service interruption.

Preparations

- Make sure that you have activated LOG and have configured projects and logstores.

- Prepare the following configuration items:
 - The endpoint of LOG. Use the intranet endpoint.
 - A pair of AccessKey ID and AccessKey Secret to access LOG.

Use LOG Indexing Service

1. Prepare the ingestion spec.

LOG Indexing Service is similar to Kafka Indexing Service. For more information, see [Kafka Indexing Service](#). The same data is indexed. The ingestion spec for the data source is as follows and is saved as metrics-sls.json.

```
{
  " type ": " sls ",
  " dataSchema ": {
    " dataSource ": " metrics - sls ",
    " parser ": {
      " type ": " string ",
      " parseSpec ": {
        " timestampSpec ": {
          " column ": " time ",
          " format ": " auto "
        },
        " dimensionsSpec ": {
          " dimensions ": [ " url ", " user " ]
        },
        " format ": " json "
      }
    },
    " granularitySpec ": {
      " type ": " uniform ",
      " segmentGranularity ": " hour ",
      " queryGranularity ": " none "
    },
    " metricsSpec ": [ {
      " type ": " count ",
      " name ": " views "
    },
    {
      " name ": " latencyMs ",
      " type ": " doubleSum ",
      " fieldName ": " latencyMs "
    }
  ],
  " ioConfig ": {
    " project ": < your_project >,
    " logstore ": < your_logstore >,
    " consumerProperties ": {
      " endpoint ": " cn - hangzhou - intranet . log .
aliyuncs . com ", ( In this example , the China ( Hangzhou
) region is used . Use the intranet endpoint . )
      " accessKeyId ": "< your - access - key - id >",
      " access - key - secret ": < your_access_key_secret
>,
      " logtail . collection - mode ": " simple "/" other "
    },
    " taskCount ": 1 ,
  }
}
```

```

    " replicas ": 1 ,
    " taskDuration ": " PT1H "
  },
  " tuningConfig ": {
    " type ": " sls ",
    " maxRowsInMemory ": " 100000 "
  }
}

```

The ingestion specs of Kafka Indexing Service and LOG Indexing Service are similar. Note the following fields:

- **type**: sls
- **dataSchema.parser.parseSpec.format**: depends on **ioConfig.consumerProperties.logtail.collection-mode** (log collection mode of Log Service). If you select Simple Mode, enter the source file format. If you do not select Simple Mode, enter json.
- **ioConfig.project**: the project of which the logs you want to collect.
- **ioConfig.logstore**: the Logstore of which the logs you want to collect.
- **ioConfig.consumerProperties.endpoint**: the intranet endpoint of LOG. For example, the endpoint for the China (Hangzhou) region is `cn - hangzhou - intranet . log . aliyuncs . com`.
- **ioConfig.consumerProperties.access-key-id**: the AccessKey ID of the account.
- **ioConfig.consumerProperties.access-key-secret**: the AccessKey Secret of the account.
- **ioConfig.consumerProperties.logtail.collection-mode**: the log collection mode of Log Service. If you select Simple Mode, enter simple. Otherwise, enter other.

2. Run the following command to add a LOG supervisor.

```

curl -- negotiate - u : druid - b ~/ cookies - c ~/ cookies
  - XPOST - H ' Content - Type : applicatio n / json ' - d @
metrics - sls . json http :// emr - header - 1 . cluster - 1234 :
18090 / druid / indexer / v1 / supervisor

```



Notice:

The `--negotiate`, `-u`, `-b`, and `-c` options are required for secure Druid clusters.

3. Import data to LOG

You can import data to LOG by using multiple methods. For more information, see [Log Service](#).

4. Perform queries by using Druid.

6.6 Tranquility

This section uses Kafka as an example and describes how to use Tranquility in E-MapReduce to capture data from the Kafka cluster and push it to the Druid cluster in real time.

Tranquility is an application that sends data to Druid in real-time in push mode. It solves many issues, such as multiple partitions, multiple copies, service discovery, and data loss. It simplifies the use of Druid and supports a wide range of data sources, including Samza, Spark, Storm, Kafka, and Fink.

Interaction with the Kafka cluster

The first interaction is between the Druid cluster and the Kafka cluster. The interaction configuration of the two clusters is similar to that of the Hadoop cluster. You have to set the connectivity and hosts. For standard mode Kafka clusters, complete the following steps:

1. Ensure the communication between clusters. (The two clusters are either in the same security group, or each cluster is associated with a different security group and access rules are configured for these security groups.)
2. Write the hosts of the Kafka cluster to the hosts list of each node on the Druid cluster. Note that the hostname of the Kafka cluster should be a long name, such as `emr-header-1.cluster-xxxxxxx`.

For high-security mode Kafka clusters, complete the following operations (the first two steps are the same as those for standard mode clusters):

1. Ensure the communication between the two clusters (The two clusters are in the same security group, or each cluster is associated with a different security group and access rules are configured for these security groups).
2. Write the hosts of the Kafka cluster to the hosts list of each node on the Druid cluster. Note that the hostname of the Kafka cluster should be a long name, such as `emr-header-1.cluster-xxxxxxx`.
3. Set Kerberos cross-domain mutual trust between the two clusters. For details, see [#unique_31](#). Bidirectional mutual trust is recommended.
4. Prepare a client security configuration file:

```
KafkaClient {
    com.sun.security.auth.module.Krb5LoginModule
    required
```



```

    useKeyTab = true
    storeKey = true
    keyTab = "/ etc / ecm / druid - conf / druid . keytab "
    principal = " druid @ EMR . 1234 . COM ";
};

```

Synchronize the configuration file to all nodes in the Druid cluster and place it in a specific directory, such as `/ tmp / kafka / kafka_clie nt_jaas . conf .`

5. In `overlord.jvm` of the Druid configuration page:

```

Add Djava . security . auth . login . config =/ tmp / kafka /
kafka_clie nt_jaas . conf

```

6. Configure the following option in `middleManager.runtime` on the Druid

configuration page: `druid . indexer . runner . javaOpts =- Djava . security . auth . login . confi =/ tmp / kafka / kafka_clie nt_jaas . conf` and other jvm startup parameters.

7. Restart the Druid service.

Use Tranquility Kafka

Because Tranquility is a service, it is a consumer for Kafka and a client for Druid. You can use a neutral machine to run Tranquility, as long as this machine is able to connect to the Kafka and the Druid clusters simultaneously.

1. Create a topic named `pageViews` on the Kafka side.

```

-- If the Kafka high - security mode is enabled :
export KAFKA_OPTS ="- Djava . security . auth . login . config
=/ etc / ecm / kafka - conf / kafka_clie nt_jaas . conf "
--
./ bin / kafka - topics . sh -- create -- zookeeper emr -
header - 1 : 2181 , emr - header - 2 : 2181 , emr - header - 3 :
2181 / kafka - 1 . 0 . 1 -- partitions 1 -- replicatio n -
factor 1 -- topic pageViews

```

2. Download the Tranquility installation package and decompress it to a path.

3. Configure the `dataSource`.

It is assumed that your topic name is `pageViews`, and each topic is a JSON file.

```

{" time ": " 2018 - 05 - 23T11 : 59 : 43Z ", " url ": "/ foo / bar
", " user ": " alice ", " latencyMs ": 32 }
{" time ": " 2018 - 05 - 23T11 : 59 : 44Z ", " url ": "/", " user
": " bob ", " latencyMs ": 11 }
{" time ": " 2018 - 05 - 23T11 : 59 : 45Z ", " url ": "/ foo / bar
", " user ": " bob ", " latencyMs ": 45 }

```

The configuration of the corresponding `dataSource` is as follows:

```

{

```

```

    "dataSourceSpec": {
      "pageViews - kafka": {
        "spec": {
          "dataSchema": {
            "dataSource": "pageViews - kafka",
            "parser": {
              "type": "string",
              "parseSpec": {
                "timestampSpec": {
                  "column": "time",
                  "format": "auto"
                },
                "dimensionsSpec": {
                  "dimensions": ["url", "user"],
                  "dimensionExclusions": [
                    "timestamp",
                    "value"
                  ]
                }
              },
              "format": "json"
            }
          },
          "granularitySpec": {
            "type": "uniform",
            "segmentGranularity": "hour",
            "queryGranularity": "none"
          },
          "metricsSpec": [
            { "name": "views", "type": "count" },
            { "name": "latencyMs", "type": "doubleSum", "fieldName": "latencyMs" }
          ],
          "ioConfig": {
            "type": "realtime"
          },
          "tuningConfig": {
            "type": "realtime",
            "maxRowsInMemory": "100000",
            "intermediatePersistPeriod": "PT10M",
            "windowPeriod": "PT10M"
          },
          "properties": {
            "task.partitions": "1",
            "task.replicants": "1",
            "topicPattern": "pageViews"
          }
        }
      },
      "properties": {
        "zookeeper.connect": "localhost",
        "druid.discovery.curator.path": "/druid/discovery",
        "druid.selectors.indexing.serviceName": "druid/overlord",
        "commit.periodMillis": "15000",
        "consumer.numThreads": "2",
        "kafka.zookeeper.connect": "emr-header-1.cluster-500148518:2181,emr-header-2.cluster-500148518:2181,emr-header-3.cluster-500148518:2181/kafka-1.0.1",
        "kafka.group.id": "tranquility-kafka",

```

```
}
```

4. Run the following command to start Tranquility.

```
./ bin / tranquilit y kafka - configFile
```

5. Start the producer and configure it to send data.

```
./ bin / kafka - console - producer . sh -- broker - list emr -  
worker - 1 : 9092 , emr - worker - 2 : 9092 , emr - worker - 3 :  
9092 -- topic pageViews
```

Enter the following codes:

```
{" time ": " 2018 - 05 - 24T09 : 26 : 12Z ", " url ": "/ foo / bar  
", " user ": " alice ", " latencyMs ": 32 }  
{" time ": " 2018 - 05 - 24T09 : 26 : 13Z ", " url ": "/", " user  
": " bob ", " latencyMs ": 11 }  
{" time ": " 2018 - 05 - 24T09 : 26 : 14Z ", " url ": "/ foo / bar  
", " user ": " bob ", " latencyMs ": 45 }
```

You can now view specific information in the Tranquility log. The corresponding real-time indexing task has also been started on the Druid side.

6.7 Superset

The Druid cluster integrates the Superset tool, which is integrated with Druid and supports a variety of relational databases. Because Druid supports SQL, you can access Druid through Superset in two ways: Druid's native query language or SQL.

Superset is installed in emr-header-1 by default and does not support high availability at present. Before you use this tool, make sure that your host can access emr-header-1. You can connect to the host by establishing the [SSH tunnel](#).

1. Log on to the Superset

Enter `http://emr-header-1:18088` in your browser to go to the Superset logon page. The default username is admin and the default password is admin. When you log on for the first time, we strongly recommend changing your password.

2. Add a Druid cluster

The English interface is displayed by default. You can select the appropriate language by clicking the flag icon in the upper-right corner. In the menu bar along the top, select Data Source > Druid Cluster to add a Druid cluster.

Configure the addresses of the coordinator and broker. The default port number in E-MapReduce is the corresponding open source port number with "1" added in front. For example, if the open-source broker port number is 8082, the port number in E-MapReduce is 18082.

3. Refresh or add a new data source

After adding the Druid cluster, you can click Data Source > Scan to add new data sources. The data sources on the Druid cluster loaded automatically.

You can also customize a new data source by clicking Sources > Druid Datasources on the interface. (This operation is equivalent to writing a JSON file for data source ingestion.)

Enter the necessary information for custom data sources, and save it.

Click the second of the three small icons on the left side to edit the data source. Enter the appropriate information, such as dimensions and metrics.

4. Query Druid

After the data source has been added successfully, click it to go to the details page.

5. (Optional) Use Druid as a database

Superset provides SQLAlchemy to support a wide variety of databases with various dialects, as shown in the following figure.

Superset also supports accessing Druid in this way. The corresponding SQLAlchemy URI of Druid is `druid://emr-header-1:18082/druid/v2/sql`. When you add Druid as a database, check the "Expose in SQL Lab" check box.

You can now use SQL to query in the SQL toolkit.

6.8 Common Druid problems

This section describes some of the common problems you may encounter with Druid.

Analyze the indexing failure

If indexing fails, complete the following steps to troubleshoot the failure:

- For batch data indexing
 1. If the curl command output displays an error or does not display any information, check the file format. Alternatively, add the `-v` parameter to the curl command to check the value returned from the RESTful API.
 2. Observe the execution of jobs on the Overlord page. If the execution fails, view the logs on that page.
 3. In many cases, logs are not generated. In the case of a Hadoop job, open the YARN page to check whether an index job has been generated.
 4. If no errors are found, you need to log on to the Druid cluster and view the execution logs of Overlord at `/mnt/disk1/log/druid/overlord-emr-header-1.cluster-xxxx.log`. In the case of an HA cluster, check the Overlord that you submitted the job to.
 5. If the job has been submitted to Middlemanager but a failure is returned from Middlemanager, you need to view the worker that the job is submitted to in Overlord, and log on to the worker node to view the Middlemanager logs (at `/mnt/disk1/log/druid/middleManager-emr-header-1.cluster-xxxx.log`).

- For real-time Tranquility indexing

Check the Tranquility log to see if the message was received or dropped.

The remaining troubleshooting steps are the same as steps 2 to 5 of batch indexing.

Most errors are about cluster configurations and jobs. Cluster configuration errors are about memory parameters, cross-cluster connection, access to clusters in high-security mode, and principals. Job errors are about the format of the job description files, input data parsing, and other job-related configuration issues (such as `ioConfig`).

Obtain the FAQ list

- Service startup fails.

Most of these problems are due to configuration problems with the running parameters of the JVM component. For example, the machine may not have a large memory, but it is configured with a larger JVM memory or a larger number of threads.

To resolve this issue, view the component logs and adjust the relevant parameters. JVM memory involves heap memory and direct memory. For more information, go to [Druid Performance FAQ](#).

- The YARN task fails during indexing and shows a JAR package conflict error like this: `Error : class com . fasterxml . jackson . datatype . guava . deser . HostAndPortDeserializer overrides final method deserialize . (Lcom / fasterxml / jackson / core / JsonParser ; Lcom / fasterxml / jackson / databind / DeserializationContext ;) Ljava / lang / Object ;`.

To resolve this issue, add the following content to the indexing job configuration file:

```
" tuningConfig " : {
  ...
  " jobProperties " : {
    " mapreduce . job . classloader " : " true "
    or
    " mapreduce . job . user . classpath . first " : " true "
  }
  ...
}
```

```
}
```

The parameter `mapreduce . job . classloader` allows MapReduce jobs to use standalone classloaders, and the parameter `mapreduce . job . user . classpath . first` gives MapReduce the priority to use your JAR packages. You can select one of these two configuration items. For more information, go to [Druid documents](#).

- The logs of the index task report that the reduce task cannot create the segments directory.

To resolve this issue, complete the following:

- Check the settings for deep storage, including type and directory. If the type is local, pay attention to the permission settings of the directory. If the type is HDFS, the directory should be written as the full HDFS path, such as `hdfs://:9000 /`. For `hdfs_master`, IP is recommended. If you want to use a domain name, use the full domain name, such as `emr-header-1.cluster-xxxxxxx` rather than `emr-header-1`.
- If you are using Hadoop for batch indexing, you must set the deep storage of segments as "hdfs". The local type may cause the MapReduce job to be in an unidentified state, because the remote YARN cluster cannot create the segments directory in the reduce task. (This is only applicable to standalone Druid clusters.)
- Failed to create directory within 10,000 attempts.

This issue occurs typically because the path set by `java.io.tmp` in the JVM configuration file does not exist. Set the path and make sure that the Druid account has permission to access it.

- `com.twitter.finagle.NoBrokersAvailableException: No hosts are available for disco! firehose:druid:overlord`

This issue is typically due to ZooKeeper connection issues. Make sure that Druid and Tranquility have the same connection string for ZooKeeper. Because the default ZooKeeper path for Druid is `/druid`, make sure that `zookeeper.connect` in the Tranquility settings includes `/druid`. (Two ZooKeeper settings exist in Tranquility Kafka. One is `zookeeper.connect` used to connect the ZooKeeper of the Druid cluster, and the other is `kafka.zookeeper.connect` used to connect the

ZooKeeper of the Kafka cluster. These two ZooKeepers may not belong to the same ZooKeeper cluster.)

- The MiddleManager reports that the `com.hadoop.compression.lzo.LzoCodec` class cannot be found during indexing.

This is because the Hadoop cluster of E-MapReduce is configured with lzo compression.

To resolve this issues, copy the JAR package and the native file under the EMR HADOOP_HOME/lib directory to Druid's `druid.extensions.hadoopDependenciesDir` (by default, `DRUID_HOME/hadoop-dependencies`).

- The following error is reported during indexing:

```
2018 - 02 - 01T09 : 00 : 32 , 647 ERROR [ task - runner - 0 -
priority - 0 ] com . hadoop . compressio n . lzo . GPLNativeC
odeLoader - could not unpack the binaries
    java . io . IOExceptio n : No such file or directory
        at java . io . UnixFileSy stem . createFile
Exclusive_l y ( Native Method ) ~[?: 1 . 8 . 0_151 ]
        at java . io . File . createTemp File ( File . java :
2024 ) ~[?: 1 . 8 . 0_151 ]
        at java . io . File . createTemp File ( File . java :
2070 ) ~[?: 1 . 8 . 0_151 ]
        at com . hadoop . compressio n . lzo . GPLNativeC
odeLoader . unpackBina ries ( GPLNativeC odeLoader . java : 115 )
[ hadoop - lzo - 0 . 4 . 21 - SNAPSHOT . jar :?]
```

This issue occurs because the `java.io.tmp` path does not exist. Set the path and make sure that the Druid account has permission to access it.

7 Presto

7.1 What is Presto?

Presto is an open-source distributed SQL-on-Hadoop query engine powered by Facebook. It is currently maintained by the open source community and Facebook engineers, and has derived multiple commercial versions.

Basic features

Presto is implemented in Java. It is easy to use and offers high performance and strong scalability. Its other features are as follows:

- Fully supports ANSI SQL.
- Supports rich data sources, accessing them as follows:
 - Interaction with Hive
 - Cassandra
 - Kafka
 - MongoDB
 - MySQL
 - PostgreSQL
 - SQL Server
 - Redis
 - Redshift
 - Local files
- Supports advanced data structures.
 - Array and map data
 - JSON data
 - GIS data
 - Color data

- Presto provides the following expansion configurations:

- Data connector expansion
- Custom data types
- Custom SQL functions

To achieve efficient service processes, you can expand the corresponding modules according to your own service features.

- Based on the Pipeline process model, data is returned to you in real time.
- Improved monitoring interfaces.
 - Friendly WebUI is provided to present the execution processes of the query tasks visually.
 - Supports the JMX protocol.

Scenarios

Presto is a distributed SQL engine that is well-suited to the following scenarios:

- ETL
- Ad-hoc queries
- Massive structured and semi-structured data analysis
- Massive multi-dimensional data aggregation/reports

In particular, Presto is a data warehouse product, which is not designed to replace traditional RDBMS databases such as MySQL and PostgreSQL. It has limited support for transactions and is not suitable for online service scenarios.

Benefits

In addition to being open source, the E-MapReduce Presto product comes with the following advantages:

- You can purchase it for immediate use to build a Presto cluster with hundreds of nodes in minutes.
- It supports elastic scalability, meaning that you can scale the cluster up and down with simple operations.
- It works perfectly in connection with the E-MapReduce software stacks, and supports the processing of data stored in OSS.
- O&M is free 24/7, providing an all-in-one service.

7.2 Quick start

7.2.1 System structure

Architecture

The following figure shows the architecture of Presto:

Presto has a typical mobile/server architecture comprising a coordinator node and multiple worker nodes. Coordinator is responsible for the following:

- Receiving and parsing your query requests, generating execution plans, and sending the execution plans to the worker nodes for execution.
- Monitoring the running status of the worker nodes. Each worker node maintains a heartbeat connection with the coordinator node, reporting the node statuses.
- Maintaining the metastore data

Worker nodes run the tasks assigned by the coordinator node, read data from external storage systems through connectors, process the data, and send the results to the coordinator node.

7.2.2 Basic concepts

This section describes the basic Presto concepts for a better understanding of the Presto work mechanism.

Data model

Data model indicates to the data organization form. Presto uses a three-level structure, namely Catalog, Schema, and Table, to manage data.

- Catalog

A catalog contains multiple schemas and is physically directed to an external data source, which can be accessed through connectors. When you run an SQL statement in Presto, you are running it against one or more catalogs.

- Schema

A schema is a database instance that contains multiple data tables.

- Table

A data table is the same as a general database table.

The relationships between catalogs, schemas, and tables are shown in the following figure.

Connector

Presto uses connectors to connect to various external data sources. To access customized data sources, Presto provides a standard [SPI](#), which allows you to develop your own connectors using this standard API.

A catalog is typically associated with a specific connector (which can be configured in the Properties file of the catalog). Presto contains multiple built-in connectors.

7.2.3 Command line tool

This section describes how to use the command line tool to operate the Presto console.

The command line tool uses [SSH to log on to an EMR cluster](#) and executes the following command to enter the Presto console:

```
$ presto --server emr-header-1:9090 --catalog hive --
  schema default --user hadoop
```

High-security clusters use the following command:

```
$ presto --server https://emr-header-1:7778 \
  --enable-authentication \
  --krb5-config-path /etc/krb5.conf \
  --krb5-keytab-path /etc/ecm/presto-conf/
presto-keytab \
  --krb5-remote-service-name presto \
  --keystore-path /etc/ecm/presto-conf/keystore
\
  --keystore-password 81ba14ce6084 \
  --catalog hive --schema default \
  --krb5-principal presto/emr-header-1.cluster-
XXXX@EMR.XXXX.COM
```

- *XXXX* is the ECM ID of the cluster, a string of numbers that can be obtained through `cat /etc/hosts`.
- *81ba14ce6084* is the default password of `/etc/ecm/presto-conf/keystore`. We recommend that you use your own keystore after deployment.

You can execute the following command from the console:

```
Presto: Default> show schemas;
  schema .
-----
```

```

default
Hive
informatio  n_schema
tpch_100gb  _orc
tpch_10gb_  orc
tpch_10tb_  orc
tpch_1tb_o  rc
( 7      rows )

```

You can then execute the `presto -- help` command to obtain help from the console. The parameters and definitions are as follows:

```

-- server < server >                # Specifies the URI
of a Coordinator
-- user < user >                    # Sets the username
-- catalog < catalog >             # Specifies the
default Catalog
-- schema < schema >              # Specifies the
default Schema
-- execute < execute >            # Executes a
statement and then exits
-f < file >, -- file < file >      # Executes an SQL
statement and then exits
-- debug                          # Shows debugging
informatio n
-- client - request - timeout < timeout > # Specifies the
client timeout value, which is 2 minutes by default
-- enable - authentica tion        # Enables client
authentica tion
-- keystore - password < keystore password > # KeyStore
password
-- keystore - path < keystore path >      # KeyStore path
-- krb5 - config - path < krb5 config path > # Kerberos
configurat ion file path ( default : / etc / krb5 . conf )
-- krb5 - credential - cache - path < path > # Kerberos
credential cache path
-- krb5 - keytab - path < krb5 keytab path > # Kerberos Key
table path
-- krb5 - principal < krb5 principal >    # Kerberos
principal to be used
-- krb5 - remote - service - name < name > # Remote Kerberos
node name
-- log - levels - file < log levels >     # Configurat ion
file path for debugging logs
-- output - format < output - format >    # Bulk export
data format, which is CSV by default
-- session < session >              # Specifies the
session attribute, in the format key = value
-- socks - proxy < socks - proxy >        # Sets the proxy
server
-- source < source >                # Sets query source
-- version                          # Shows version info

```

```
- h , -- help # Shows help info
```

7.2.4 Uses JDBC

Java applications can access databases using the JDBC driver provided by Presto. The procedure is the same as that for general RDBMS databases.

Introduction to Maven

You can add the following configuration to the POM file to introduce the Presto JDBC driver:

```
< dependency >
  < groupId > com . facebook . presto </ groupId >
  < artifactId > presto - jdbc </ artifactId >
  < version > 0 . 187 </ version >
</ dependency >
```

Driver class name

The Presto JDBC driver class is `com . facebook . presto . jdbc . PrestoDriver`.

Connection string

The following connection string format is supported.

```
jdbc : presto ://< COORDINATOR >:< PORT >/[ CATALOG ]/[ SCHEMA ]
```

For example:

```
jdbc : presto :// emr - header - 1 : 9090 # Connects
to data base , using the default Catalog and Schema
jdbc : presto :// emr - header - 1 : 9090 / hive #
Connects to data base , using Catalog ( hive ) and the
default Schema
jdbc : presto :// emr - header - 1 : 9090 / hive / default #
Connects to data base , using Catalog ( hive ) and Schema
( default )
```

Connection parameters

The Presto JDBC driver supports various parameters that may be set as URL parameters or as `Properties` and passed to `DriverManager`.

Example of passing parameters to `DriverManager` as `Properties` :

```
String url = " jdbc : presto :// emr - header - 1 : 9090 / hive /
default ";
Properties properties = new Properties ();
properties . setProperty ( " user ", " hadoop " );
Connection connection = DriverManager . getConnection ( url
, properties );
```

```
.....
```

Example of passing parameters to DriverManager as URL parameters:

```
String url = "jdbc:presto://emr-header-1:9090/hive/default?user=hadoop";
Connection connection = DriverManager.getConnection(url);
.....
```

Parameters are described as follows:

| Parameter name | Format | Description |
|------------------------------|--------|---|
| user | STRING | User name. |
| password | STRING | Password. |
| Socksproxy | :\ | SOCKS proxy server address and port. For example, localhost:1080. |
| httpProxy | :\ | HTTP proxy server address and port. For example, localhost:8888. |
| SSL | true\ | Whether or not to use HTTPS for connections. This is false by default. |
| SSLTrustStorePath | STRING | Java TrustStore file path. |
| SSLTrustStorePassword | STRING | Java TrustStore password. |
| KerberosRemoteServiceName | STRING | Kerberos service name. |
| KerberosPrincipal | STRING | Kerberos principal. |
| KerberosUseCanonicalHostname | true\ | Whether or not to use the canonical hostname. This is false by default. |
| KerberosConfigPath | STRING | Kerberos configuration file path. |
| KerberosKeytabPath | STRING | Kerberos KeyTab file path. |
| KerberosCredentialCachePath | STRING | Kerberos credential cache path |

Java example

The following is an example of using the Presto JDBC driver with Java.

```
.....
// Loads the JDBC Driver class
try {
    Class.forName("com.facebook.presto.jdbc.PrestoDriver");
} catch (ClassNotFoundException e) {
```

```

        LOG . ERROR (" Failed to load presto jdbc driver .", e
    );
    System . exit (- 1 );
}
Connection connection = null ;
Statement stmt = null ;
try {
    String url = " jdbc : presto :// emr - header - 1 : 9090 /
hive / default ";
    Properties properties = new Properties ();
    properties . setProperty (" user ", " hadoop ");
    // Creates the connection object
    Connection = DriverManager . getConnection ( URL ,
properties );
    // Creates the Statement object
    statement = connection . createStatement ();
    Executes the query
    ResultSet rs = statement . executeQuery (" select * from
tbl ");
    Returns results
    int columnNum = rs . getMetaData (). getColumnCount ();
    int rowIndex = 0 ;
    while ( rs . next ()) {
        rowIndex ++;
        for ( int i = 1 ; i <= columnNum ; i ++) {
            System . out . println (" Row " + rowIndex + " ,
Column " + i + " : " + rs . getInt ( i ));
        }
    }
} catch ( SQLException e ) {
    LOG . ERROR (" Exception thrown .", e );
} finally {
    // Destroys Statement object
    If ( statement != null ) {
        try {
            statement . close ();
        } catch ( Throwable t ) {
            // No - ops
        }
    }
    Closes connection
    if ( connection != null ) {
        try {
            connection . close ();
        } catch ( Throwable t ) {
            // No - ops
        }
    }
}
}
}

```

Use reverse proxy

You can use the HAProxy reverse proxy Coordinator to access the Presto service through the proxy service.

- Non-Security Cluster proxy configuration

To configure a cluster proxy for a non-Security Cluster, follow these steps:

1. Install HAProxy on the proxy Node
2. Modify the HAProxy configuration (`/ Etc / haproxy . cfg`), Add the following content:

```
.....
listen    prestojdbc    : 9090
    Mode    TCP
    option    tcplog
    balance    source
    Server    presto - coordinator - 1    emr - header - 1 : 9090
```

3. Restart the HAProxy Service

Now, you can use the proxy server to access Presto. You only need to change the IP address of the Connected Server to the IP address of the proxy service.

7.2.5 Implement authentication with ApacheDS

For username and password authentication, you only need to connect the Presto coordinator to the LDAP server.

Procedure

1. Configure ApacheDS and enable LDAPS.
2. Create user information in ApacheDS.
3. Configure Presto Coordinator. Restart Presto Coordinator.
4. Verify the configurations

Enable LDAPS

1. Create a keystore used for ApacheDS server. The following example uses 123456 as the password.

```
## Create a keystore
> cd / var / lib / apacheds - 2 . 0 . 0 - M24 / default / conf /
> keytool - genkeypair - alias apacheds - keyalg RSA -
    validity 7 - keystore ads . keystore

Enter keystore password :
Re - enter new password :
What is your first and last name ?
[ Unknown ]: apacheds
What is the name of your organizational unit ?
[ Unknown ]: apacheds
What is the name of your organization ?
[ Unknown ]: apacheds
What is the name of your City or Locality ?
```

```

[ Unknown ]:  apacheds
What is the name of your State or Province ?
[ Unknown ]:  apacheds
What is the two - letter country code for this unit
?
[ Unknown ]:  CN
Is CN = apacheds , OU = apacheds , O = apacheds , L = apacheds
, ST = apacheds , C = CN correct ?
[ no ]:  yes

Enter key password for < apacheds >
( RETURN if same as keystore password ):
Re - enter new password :

Warning :
The JKS keystore uses a proprietary format . It
is recommende d to migrate to PKCS12 which is an
industry standard format using " keytool - importkeys
tore - srckeystor e ads . keystore - destkeysto re ads .
keystore - deststoret ype pkcs12 ".

## Change the owner of the keystore file to "
apacheds ".
> chown apacheds : apacheds ./ ads . keystore

## Export the certificat e .
## Enter the password . The password is set in the
previous step : 123456 .
> keytool - export - alias apacheds - keystore ads .
keystore - rfc - file apacheds . cer
Enter keystore password :
Certificat e stored in file < apacheds . cer >

Warning :
The JKS keystore uses a proprietary format . It
is recommende d to migrate to PKCS12 which is an
industry standard format using " keytool - importkeys
tore - srckeystor e ads . keystore - destkeysto re ads .
keystore - deststoret ype pkcs12 ".

## Import the certificat e to the cacerts file for
self - authentica tion .

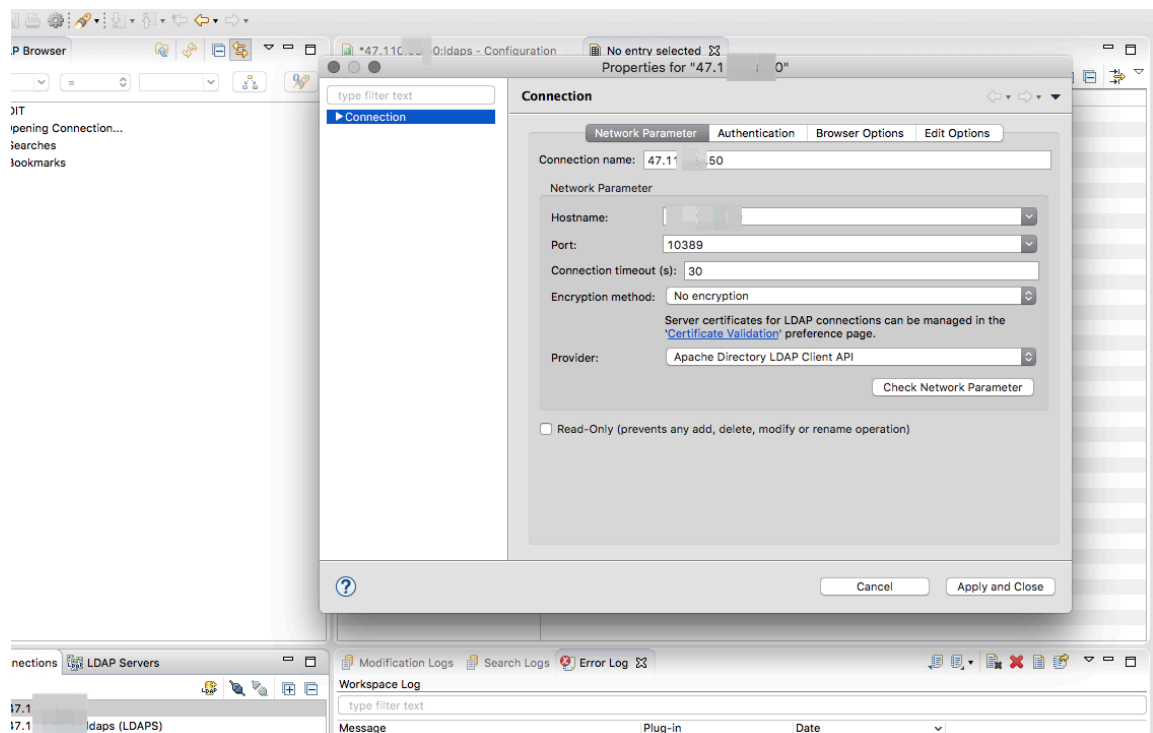
```

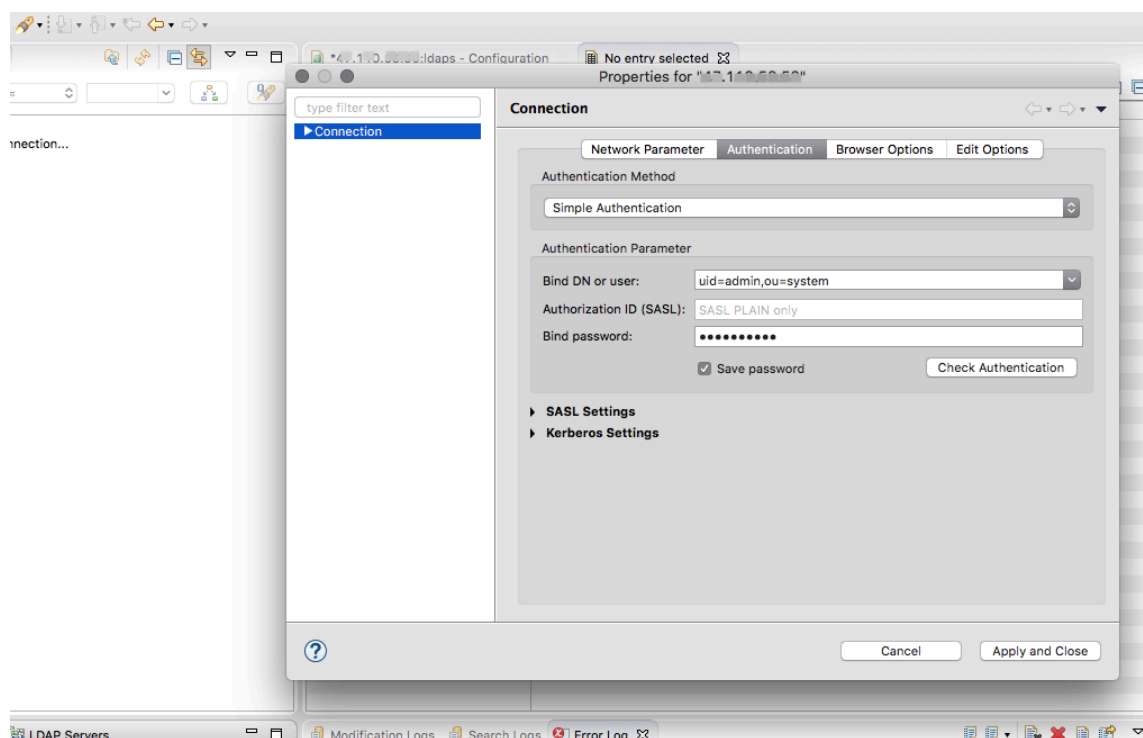
```
> keytool -import -file apacheds.cer -alias apacheds
-keystore /usr/lib/jvm/java-1.8.0/jre/lib/security/cacerts
```

2. Modify the configurations. Enable LDAPS.

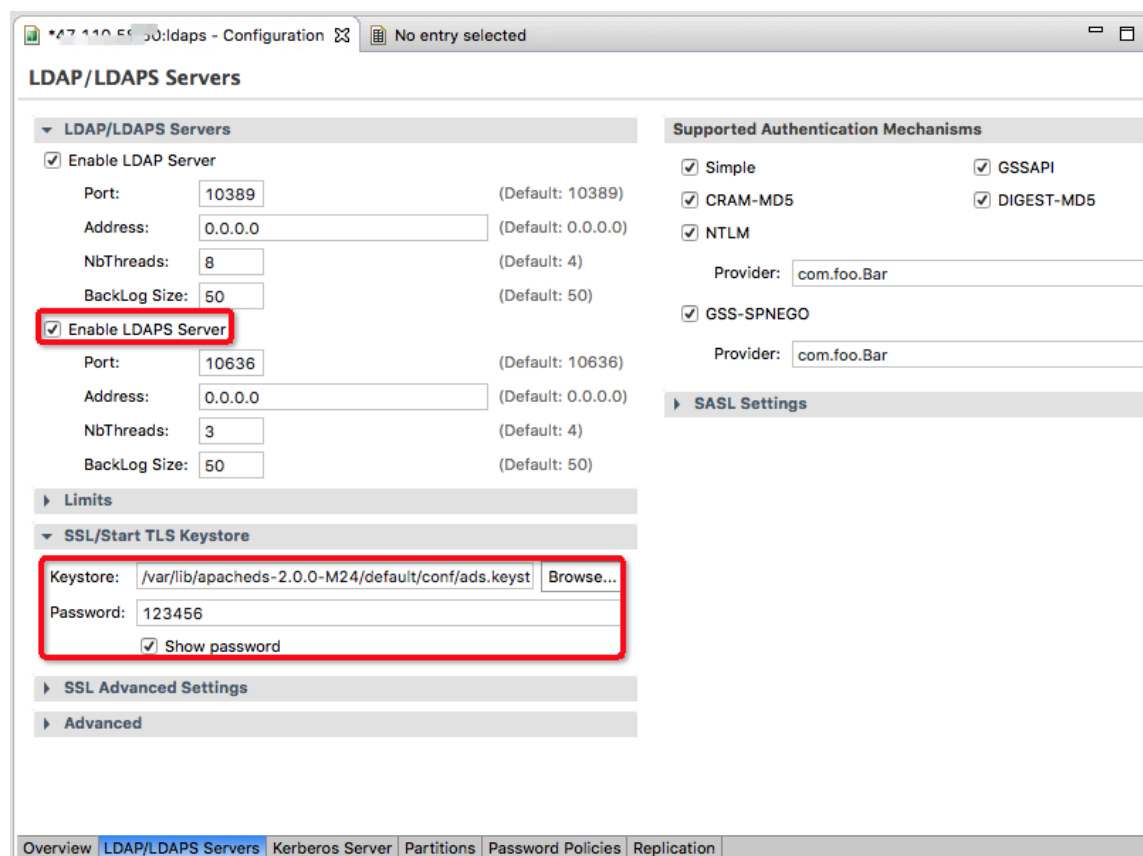
Start Apache Directory Studio. Connect to the ApacheDS service on the cluster.

- Specify the value of DN to "uid=admin,ou=system"
- You can view the password under the following path: `/var/lib/ecm-agent/cache/ecm/service/APACHEDS/2.0.0.1.1/package/files/modifypwd.ldif`





After the connection is complete, go to the Configuration page. On the page, select the Enable LDAPS Server check box. In the SSL/Start TLS Keystore section, select the keystore file you created in Step 1 and enter the password. Press Ctrl+S to save the configurations.



3. Restart ApacheDS

Log on to the cluster. Run the following command to restart ApacheDS.

```
> service apacheds - 2 . 0 . 0 - M24 - default restart
```

LDAPS has been started. The port number is 10636.



Note:

Handshake exceptions are thrown when you test LDAPS connection in Apache Directory Studio. The cause is that the default timeout value is very short. Actual use is not affected.

Create user information

In this example, users are created based on DN: dc=hadoop,dc=apache,dc=org.

1. Go to the Partitions configuration page, perform configurations as shown in the following figure, and press CTRL+S to save the configurations. By doing this, you create a partition with the suffix "dc=hadoop,dc=apache,dc=org". Restart ApacheDS.

Partitions

All Partitions

- emr (o=emr) [JDBM]
- hadoop (dc=hadoop,dc=apache,dc=org) [JDBM]**
- system (ou=system) [JDBM]

Buttons: Add, Delete

Partition General Details

Set the properties of the partition.

Partition Type: JDBM

ID: hadoop

Suffix: dc=hadoop,dc=apache,dc=org

☒ Synchronization On Write

Context Entry

Set the attribute/value pairs for the Context Entry of the partition.

☒ Auto-generate context entry from suffix DN

| Attribute | Value |
|-------------|--------|
| objectclass | domain |
| objectclass | top |
| dc | hadoop |
| o | hadoop |

Buttons: Add..., Edit..., Delete

Partition Specific Settings

Cache Size: 100

☒ Enable Optimizer

Indexed Attributes

Set the indexed attributes of the partition.

- ou [100]
- apacheAlias [100]
- uid [100]
- objectClass [100]
- apacheOneAlias [100]

Buttons: Add..., Edit..., Delete

Navigation: Overview | LDAP/LDAPS Servers | Kerberos Server | **Partitions** | Password Policies | Replication

2. Create User

Log on to the cluster. Create a file named `/tmp/users.ldif`.

```
# Entry for a sample people container
# Please replace with site specific values
dn : ou = people , dc = hadoop , dc = apache , dc = org
objectclas s : top
objectclas s : organizati onalUnit
ou : people

# Entry for a sample end user
# Please replace with site specific values
dn : uid = guest , ou = people , dc = hadoop , dc = apache , dc = org
objectclas s : top
objectclas s : person
objectclas s : organizati onalPerson
objectclas s : inetOrgPer son
cn : Guest
sn : User
uid : guest
userPasswo rd : guest - password

# entry for sample user admin
dn : uid = admin , ou = people , dc = hadoop , dc = apache , dc = org
objectclas s : top
objectclas s : person
objectclas s : organizati onalPerson
objectclas s : inetOrgPer son
cn : Admin
sn : Admin
uid : admin
userPasswo rd : admin - password

# entry for sample user sam
dn : uid = sam , ou = people , dc = hadoop , dc = apache , dc = org
objectclas s : top
objectclas s : person
objectclas s : organizati onalPerson
objectclas s : inetOrgPer son
cn : sam
sn : sam
uid : sam
userPasswo rd : sam - password

# entry for sample user tom
dn : uid = tom , ou = people , dc = hadoop , dc = apache , dc = org
objectclas s : top
objectclas s : person
objectclas s : organizati onalPerson
objectclas s : inetOrgPer son
cn : tom
sn : tom
uid : tom
userPasswo rd : tom - password

# create FIRST Level groups branch
dn : ou = groups , dc = hadoop , dc = apache , dc = org
```

```

objectclas s : top
objectclas s : organizati onalUnit
ou : groups
descriptio n : generic groups branch

# create the analyst group under groups
dn : cn = analyst , ou = groups , dc = hadoop , dc = apache , dc =
org
objectclas s : top
objectclas s : groupofnam es
cn : analyst
descriptio n : analyst group
member : uid = sam , ou = people , dc = hadoop , dc = apache , dc
= org
member : uid = tom , ou = people , dc = hadoop , dc = apache , dc
= org

# create the scientist group under groups
dn : cn = scientist , ou = groups , dc = hadoop , dc = apache , dc
= org
objectclas s : top
objectclas s : groupofnam es
cn : scientist
descriptio n : scientist group
member : uid = sam , ou = people , dc = hadoop , dc = apache , dc
= or

```

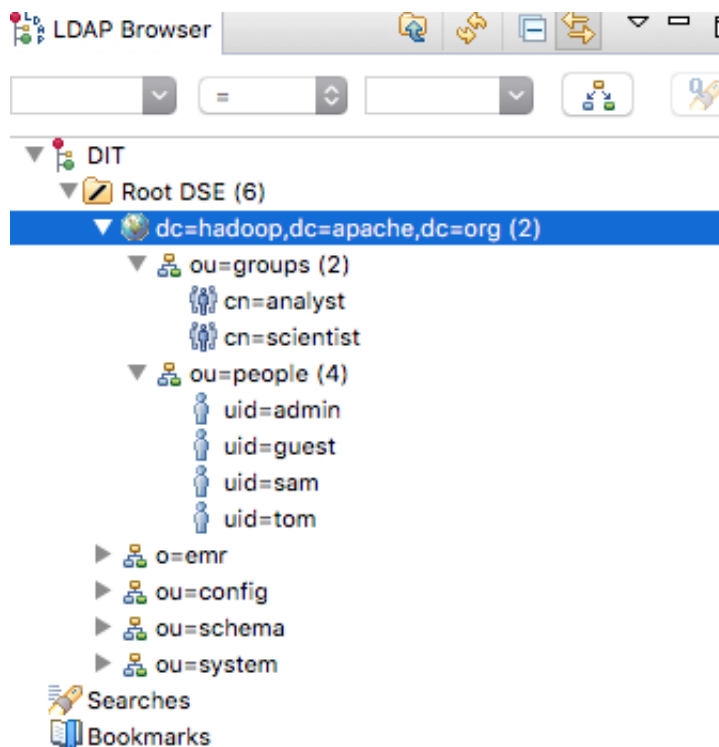
Run the following command to import the users.

```

> ldapmodify -x -h localhost -p 10389 -D "uid = admin
, ou = system" -w { password } -a -f /tmp/users.ldif

```

You can view the users in Apache Directory Studio as shown in the following figure.



Configure Presto

1. Enable Coordinator Https

a. Configure the keystore used for the Presto coordinator.

```
## Use the script that comes with EMR to generate
a keystore .
## keystore path : / etc / ecm / presto - conf / keystore
## keystore password : 81ba14ce60 84
> expect / var / lib / ecm - agent / cache / ecm / service /
PRESTO / 0 . 208 . 0 . 1 . 2 / package / files / tools / gen -
keystore . exp
```

b. Configure the Presto coordinator.

Enter the following lines in the `/ etc / ecm / presto - conf / config . properties` file.

```
http - server . https . enabled = true
http - server . https . port = 7778

http - server . https . keystore . path = / etc / ecm / presto -
conf / keystore
http - server . https . keystore . key = 81ba14ce60 84
```

2. Configure the authentication type.

a. Enter the following configurations in the `/ etc / ecm / presto - conf / config . properties` file.

```
http - server . authentication . type = PASSWORD
```

b. Enter the following configurations in the `jvm . config` file.

```
- Djavax . net . ssl . trustStore = / usr / lib / jvm / java - 1 .
8 . 0 / jre / lib / security / cacerts
- Djavax . net . ssl . trustStore Password = changeit
```

c. Create the `password - authenticator . properties` file and enter the following configurations.

```
password - authenticator . name = ldap
ldap . url = ldaps :// emr - header - 1 . cluster - 84423 :
10636
ldap . user - bind - pattern = uid = ${ USER }, ou = people , dc =
hadoop , dc = apache , dc = org
```

d. Create the `jndi . properties` file and enter the following configurations.

```
java . naming . security . principal = uid = admin , ou = system
java . naming . security . credential s = { password }
```



```
java . naming . security . authentica tion = simple
```

- e. Use the `jndi . properties` file to create a JAR file. Copy the JAR file to the Presto library.

```
jar - cvf jndi - properties . jar jndi . properties
> cp ./ jndi - properties . jar / etc / ecm / presto - current
/ lib /
```



Note:

- The following parameters are used to connect to LDAP. However, you cannot configure these parameters in Presto. Parameters added to the `jvm.config` file do not take effect. `java.naming.security.principal=uid=admin,ou=system` `java.naming.security.credentials=ZVixyOY+5k` `java.naming.security.authentication=simple`
- JNDI uses classloaders to load the `jndi.properties` file. Add these parameters to the `jndi.properties` file.
- Presto launchers only add JAR files to the classpath. You need to package `jndi.properties` as a JAR file and include it in the `lib` directory.

3. Restart Presto.

Verify the configurations

Use the Presto CLI to verify whether the configurations take effect.

```
## Correct password
> presto --server https://emr-header-1:7778 --
keystore - path /etc/ecm/presto-conf/keystore -- keystore
- password 81ba14ce6084 -- catalog hive -- schema default
-- user sam -- password
Password : < correct password >
presto : default > show schemas ;
Schema
-----
tpcds_bin_ partitione d_orc_5
tpcds_oss_ bin_partit ioned_orc_ 10
tpcds_oss_ text_10
tpcds_text _5
tst
( 5 rows )

Query 20181115_0 30713_0000 2_kp5ih , FINISHED , 3 nodes
Splits : 36 total , 36 done ( 100 . 00 %)
0 : 00 [ 20 rows , 331B ] [ 41 rows / s , 694B / s ]

## Wrong password
> presto --server https://emr-header-1:7778 --
keystore - path /etc/ecm/presto-conf/keystore -- keystore
- password 81ba14ce6084 -- catalog hive -- schema default
-- user sam -- password
Password : < wrong password >
```

```
presto : default > show schemas ;  
Error running command : Authentication failed : Access  
Denied : Invalid credentials
```

7.3 Instructions

7.3.1 Overview

This topic is intended for application developers and includes the following content:

- SQL syntax and features of the Presto database
- Performance optimization method of the Presto database
- Develop Presto database plugins and expand the database functions

7.3.2 SQL manual

7.3.2.1 Data types

Presto supports multiple common data types by default, such as boolean, integer, floating point, string, and date. You can also add custom data types by using plugins. The custom Presto connectors are not required to support all data types.

Value types

Presto has a set of built-in value types as follows:

- `BOOLEAN`

Represents an option with a value of `TRUE` or `FALSE`.

- `TINYINT`

An 8-bit signed *two's complement*

- `SMALLINT`

A 16-bit signed *two's complement*

- `INTEGER`

A 32-bit signed *two's complement*

- `BIGINT`

A 64-bit signed *two's complement*

- `REAL`

A 32-bit multi-precision implementation of the [IEEE Standard 754](#) for binary floating-point arithmetic.

- `DOUBLE`

A 64-bit multi-precision implementation of the [IEEE Standard 754](#) for binary floating-point arithmetic.

- `DECIMAL`

A fixed-precision decimal number. Precision up to 38 digits is supported, but performance is best with up to 17 digits. It takes two literal parameters to define the `DECIMAL` type:

1. Precision: the total number of digits, excluding symbols
2. Scale: the number of digits in the fractional part. It is optional and set to 0 by default.

Example: `DECIMAL '- 10 . 7 '` can be expressed with the `DECIMAL (3 , 1)` type.

The following table lists the bits and value range of the integer type.

| Value type | Bit width | Minimum value | Maximum value |
|------------|-----------|---------------|---------------|
| TINYINT | 8-bit | -2^7 | $2^7 - 1$ |
| SMALLINT | 16-bit | 2^{15} | $2^{15} - 1$ |
| INTEGER | 32-bit | -2^{31} | $-2^{31} - 1$ |
| BIGINT | 64-bit | -2^{63} | $-2^{63} - 1$ |

String types

Presto supports the following built-in string types:

- `VARCHAR`

Variable-length string with an optional maximum length.

Example: `VARCHAR` , `VARCHAR (10)`

- `CHAR`

Fixed-length string. A `CHAR` type without length specified has a default length of `1`.

Example: `CHAR`, `CHAR (10)`



Notice:

A string with the specified length always has the number of characters equal to this length. If the string length is smaller than the specified length, leading and trailing spaces are included in comparisons of the string value. As a result, two character values of different lengths can never be equal.

- `VARBINARY` indicates variable-length binary data.

Date and time

Presto supports the following built-in date and time types:

- `DATE`

Indicates a calendar date (year, month, and day) without time.

Example: `DATE ' 1988 - 01 - 30 '`

- `TIME`

Indicates time, including hours, minutes, seconds, and milliseconds. Values of this type can be rendered in the `time zone`.

Examples:

- `TIME ' 18 : 01 : 02 . 345 '` does not have a time zone definition and is parsed by using the system time zone.
- `TIME ' 18 : 01 : 02 . 345 Asia / Shanghai '` has a time zone definition and is parsed by using the defined time zone.

- `TIMESTAMP`

Indicates a point in time that includes the date and time of the day. The value range is from `' 1970 - 01 - 01 00 : 00 : 01 ' UTC` to `' 2038 - 01 - 19 03 : 14 : 07 ' UTC`, which can be rendered in the `time zone`.

Examples: `TIMESTAMP ' 1988 - 01 - 30 01 : 02 : 03 . 321 '` and `TIMESTAMP ' 1988 - 01 - 30 01 : 02 : 03 . 321 Asia / Shanghai '`

- `INTERVAL`

Mainly used in time calculation expressions to indicate a time span, in the following optional units:

- `YEAR` : year
- `QUARTER` : quarter
- `MONTH` : month
- `DAY` : day
- `HOURL` : hour
- `MINUTE` : minute
- `SECOND` : second
- `MILLISECOND` : millisecond

Example: `DATE '2012 - 08 - 08' + INTERVAL '2' DAY`

Complex types

Presto supports multiple complex built-in data types for more complex service scenarios. These data types include:

- `JSON`

JSON value type, which can be a JSON object, a JSON array, a JSON number, a JSON string, and the boolean type `true`, `false`, or `null`.

Examples:

- `JSON '[1 , null , 1988]'`
- `JSON '{" k1 ": 1 , " k2 ": " abc "'}`

- `ARRAY`

An array of the given component type. Types of elements in an array must be consistent.

Example: `ARRAY [1 , 2 , 3]`

- `MAP`

Represents a mapping relationship that consists of a key array and a value array.

Example: `MAP (ARRAY [' foo ', ' bar '], ARRAY [1 , 2])`

- **ROW**

A structure consisting of named fields. The data columns can be accessed by using the field reference operator `.` and column names.

Example: `CAST (ROW (1988 , 1 . 0 , 30) AS ROW (y BIGINT , m DOUBLE , d TINYINT))`

- **IPADDRESS**

An IP address that can represent either an IPv4 or IPv6 address. Internally, IPv4 addresses are converted into IPv6 addresses based on the [IPv4-to-IPv6 mapping table](#).

Example: `IPADDRESS ' 0 . 0 . 0 . 0 '`, `IPADDRESS ' 2001 : db8 :: 1 '`

7.3.2.2 Common functions and operators

7.3.2.2.1 Logical operators

Presto supports AND, OR, and NOT logical operators, and supports `NULL` in logical computation. For example:

```
SELECT CAST ( null as boolean ) AND true ; --- null
SELECT CAST ( null AS boolean ) AND false ; -- false
SELECT CAST ( null AS boolean ) AND CAST ( null AS
boolean ); -- null
SELECT NOT CAST ( null AS boolean ); -- null
```

A complete truth table is as follows:

| a | b | a AND b | a OR b |
|-------|-------|---------|--------|
| TRUE | TRUE | TRUE | TRUE |
| TRUE | FALSE | FALSE | TRUE |
| TRUE | NULL | NULL | TRUE |
| FALSE | TRUE | FALSE | TRUE |
| FALSE | FALSE | FALSE | FALSE |
| FALSE | NULL | FALSE | NULL |
| NULL | TRUE | NULL | TRUE |
| NULL | FALSE | FALSE | NULL |
| NULL | FALSE | NULL | NULL |

The result of `NOT NULL` is `NULL`.

7.3.2.2 Comparison functions and operators

Comparison operators

Presto supports the following comparison operators:

| Operator | Description |
|---------------------------------------|---|
| <code><</code> | Less than |
| <code>></code> | Greater than |
| <code><=</code> | Less than or equal to |
| <code>>=</code> | Greater than or equal to |
| <code>=</code> | Equal to |
| <code><>/! =</code> | Not equal to |
| <code>[NOT] BETWEEN</code> | Value X [NOT] between the minimum and maximum values |
| <code>IS [NOT] NULL</code> | Determines if a value is NULL. |
| <code>IS [NOT] DISTINCT FROM</code> | Determines if two values are identical. Generally, <code>NULL</code> indicates an unknown value, so any comparison that involves a <code>NULL</code> returns <code>NULL</code> . However, the <code>IS [NOT] DISTINCT FROM</code> operator treats <code>NULL</code> as a known value, and returns a <code>TRUE</code> or <code>FALSE</code> result. |

Comparison functions

Presto provides the following comparison functions:

- `GREATEST`

Returns the maximum value among all input values.

Example: `GREATEST (1 , 2)`

- `LEAST`

Returns the minimum value among all input values.

Example: `LEAST (1 , 2)`

Quantified comparison predicates

Presto also provides several quantified comparison predicates to improve the comparison expressions. The method is as follows:

```
< EXPRESSION >< OPERATOR >< QUANTIFIER > (< SUBQUERY >)
```

Examples:

```
SELECT 'hello' = ANY (VALUES 'hello', 'world'); -- true
SELECT 21 < ALL (VALUES 19, 20, 21); -- false
SELECT 42 >= SOME (SELECT 41 UNION ALL SELECT 42
UNION ALL SELECT 43); -- true
```

`ANY`, `ALL`, and `SOME` are quantified comparison predicates.

- `A = ALL (...) TRUE` is returned if A is equal to ALL value.
- `A <> ALL (...) TRUE` is returned if A is not equal to ALL value.
- `A < ALL (...) TRUE` is returned if A is less than ALL value.
- `A = ANY (...) TRUE` is returned if A is equal to any of the values. It is equivalent to `A IN (...)`.
- `A <> ANY (...) TRUE` is returned if A is not equal to any of the values. It is equivalent to `A IN (...)`.
- `A < ANY (...) TRUE` is returned if A is less than one of the values.

`ANY` and `SOME` have the same meaning and can be used interchangeably.

7.3.2.2.3 Conditional expressions

Conditional expressions are mainly used to express branch logic. Presto supports the following conditional expressions:

- `CASE` expression

The standard SQL `CASE` expression has two different forms:

```
CASE expression
  WHEN < value | condition > THEN result
[ WHEN ... ]
[ ELSE result ]
```


END

The `CASE` statement compares the `expression` and the value/condition in `value | condition`. It returns a result if the same value is found or the condition is met.

Examples:

```
--- Compare values
SELECT a,
       CASE a
         WHEN 1 THEN 'one'
         WHEN 2 THEN 'two'
         ELSE 'many'
       END
```

```
--- Compare conditional expressions
SELECT a, b,
       CASE
         WHEN a = 1 THEN 'aaa'
         WHEN b = 2 THEN 'bbb'
         ELSE 'ccc'
       END
```

• `IF` function

The `IF` function is a simple comparison function that is used to simplify the writing method for the comparison logic of two values. Its expression form is as follows:

```
IF ( condition , true_value , [ false_value ] )
```

Returns `true_value` if `condition` is `TRUE`, or returns `false_value` otherwise. `false_value` is optional. If it is not specified, `NULL` is returned.

- **COALESCE**

The **COALESCE** function returns the first non- **null** value in the argument list. Its expression form is as follows:

```
COALESCE ( value1 , value2 [, ...])
```

- **NULLIF**

The **NULLIF** function returns **NULL** if **value1** equals **value2** . Otherwise, it returns **value1** . The usage is as follows:

```
NULLIF ( value1 , value2 )
```

- **TRY**

The **TRY** function captures the exception that is thrown during **expression** computation and returns **NULL** . The following exceptions are handled by **TRY** :

- Division by zero, such as **x / 0**
- Incorrect type conversion
- Numeric value out of range

TRY is typically used in conjunction with **COALESCE** to return the default value in the case of errors. The usage is as follows:

```
TRY ( expression )
```

Examples:

```
--- When COALESCE and TRY are used in conjunction
, the default value 0 is returned if packages is
equal to 0 and a "division by zero" error is
thrown .
SELECT COALESCE ( TRY ( total_cost / packages ), 0 ) AS
per_packag e FROM shipping ;

per_packag e
-----
         4
        14
         0
        19
( 4 rows )
```

7.3.2.2.4 Conversion functions

Presto provides the following explicit conversion functions:

- **CAST**

Explicitly casts a value as a type, and throws an exception if the cast fails. The usage is as follows:

```
CAST ( value AS type ) -> value1 : type
```

- **TRY_CAST**

Similar to **CAST**, but returns **NULL** if the cast fails. The usage is as follows:

```
TRY_CAST ( value AS TYPE ) -> value1 : TYPE | NULL
```

- **TYPEOF**

Returns the name of the type of the provided parameter or expression value. The usage is as follows:

```
TYPEOF ( expression ) -> type : VARCHAR
```

Examples:

```
SELECT TYPEOF ( 123 ); -- integer
SELECT TYPEOF ( ' cat ' ); -- varchar ( 3 )
SELECT TYPEOF ( cos ( 2 ) + 1 . 5 ); -- double
```

7.3.2.2.5 Mathematical functions and operators

Mathematical operators

| Operator | Description |
|----------|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division (integer division performs truncation) |
| % | Modulus (remainder) |

Mathematical functions

Presto provides a variety of mathematical functions, as listed in the following table.

| Function | Syntax | Description |
|----------|------------------|-----------------------------|
| abs | abs(x) → | x |
| cbrt | cbrt(x) → double | Returns the cube root of x. |

| Function | Syntax | Description |
|---------------------------------|---|--|
| <code>ceil</code> | <code>ceil(x)</code> | Returns x rounded up to the nearest integer. This is an alias for <code>ceiling</code> . |
| <code>ceiling</code> | <code>ceiling(x)</code> | Returns x rounded up to the nearest integer. |
| <code>cosine_similarity</code> | <code>cosine_similarity(x, y) → double</code> | Returns the cosine similarity between the sparse vectors x and y. |
| <code>degrees</code> | <code>degrees(x) → double</code> | Converts angle x in radians to degrees. |
| <code>e</code> | <code>e() → double</code> | Returns the constant Euler's number. |
| <code>exp</code> | <code>exp(x) → double</code> | Returns e^x of the given number x. |
| <code>floor</code> | <code>floor(x)</code> | Returns x rounded down to the nearest integer. |
| <code>from_base</code> | <code>from_base(string, radix) → bigint</code> | Returns the value of string interpreted as a base-radix number. |
| <code>inverse_normal_cdf</code> | <code>inverse_normal_cdf(mean, sd, p) → double</code> | Computes the inverse of the Normal CDF with given mean and standard deviation (SD) for the cumulative probability. |
| <code>ln</code> | <code>ln(x) → double</code> | Returns the natural logarithm of x. |
| <code>log2</code> | <code>log2(x) → double</code> | Returns the base 2 logarithm of x. |
| <code>log10</code> | <code>log10(x) → double</code> | Returns the base 10 logarithm of x. |
| <code>log</code> | <code>log(x, b) → double</code> | Returns the base b logarithm of x. |
| <code>mod</code> | <code>mod(n, m)</code> | Returns the modulus (remainder) of n divided by m. |
| <code>pi</code> | <code>pi() → double</code> | Returns the constant Pi. |

| Function | Syntax | Description |
|----------------|--------------------------------------|--|
| pow | pow(x,p)->double | Returns x raised to the power of p. This is an alias for <code>power</code> . |
| power | power(x,p)->double | Returns x raised to the power of p. |
| radians | radians(x)->double | Converts angle x in degrees to radians. |
| rand | rand()->double | Returns a pseudo-random number in the range [0.0, 1.0). This is an alias for <code>random</code> . |
| random | random()->double | Returns a pseudo-random number in the range [0.0, 1.0). |
| random | random(n) | Returns a pseudo-random number in the range [0.0, n). |
| round | round(x) | Returns x rounded to the nearest integer. |
| round | round(x, d) | Returns x rounded to d decimal places. |
| sign | sign(x) | Sign function. If x is an integer, 0 is returned when x is equal to 0; 1 is returned when x is greater than 0; and -1 is returned when x is smaller than 0. If x is a floating point number, NAN is returned when x is NAN; 1 is returned when x is $+\infty$; and -1 is returned when x is $-\infty$. |
| sqrt | sqrt(x)->double | Returns the square root of x. |
| to_base | to_base(x, radix)->varchar | Returns the base-radix representation of x. |

| Function | Syntax | Description |
|---------------------|--|---|
| truncate | truncate(x) → double | Returns x rounded to an integer by dropping digits after the decimal point. |
| width_bucket | width_bucket(x, bound1, bound2, n) → bigint | Returns the bin number of x in an equi-width histogram with the specified bound1 and bound2 bounds and n number of buckets. |
| width_bucket | width_bucket(x, bins) | Returns the bin number of x in a histogram according to the bins specified by the array bins. |
| acos | acos(x)->double | Returns the arc cosine of x, which is a radian. |
| asin | asin(x)->double | Returns the arc sine of x, which is a radian. |
| atan | atan(x)->double | Returns the arc tangent of x, which is a radian. |
| atan2 | atan2(y,x)->double | Returns the arc tangent of y/x, which is a radian. |
| cos | cos(x)->double | Returns the cosine of x, which is a radian. |
| cosh | cosh(x)->double | Returns the hyperbolic cosine of x, which is a radian. |
| sin | sin(x)->double | Returns the sine of x, which is a radian. |
| tan | tan(x)->double | Returns the tangent of x, which is a radian. |
| tanh | tanh(x)->double | Returns the hyperbolic tangent of x, which is a radian. |
| infinity | infinity() → double | Returns the constant representing positive infinity. |
| is_finite | is_finite(x) → boolean | Determines if x is finite. |

| Function | Syntax | Description |
|--------------------------|---------------------------------------|---|
| <code>is_infinite</code> | <code>is_infinite(x) → boolean</code> | Determines if x is infinite. |
| <code>is_nan</code> | <code>is_nan(x) → boolean</code> | Determines if x is not a number. |
| <code>nan</code> | <code>nan()</code> | Returns the constant representing not-a-number (NaN). |

7.3.2.2.6 Bitwise functions

Presto provides the following bitwise functions:

| Function | Syntax | Description |
|------------------------------|--|---|
| <code>bit_count</code> | <code>bit_count(x, bits) → bigint</code> | Returns the number of bits set in x at position 1 in two's complement representation. |
| <code>bitwise_and</code> | <code>bitwise_and(x, y) → bigint</code> | Bitwise AND function |
| <code>bitwise_not</code> | <code>bitwise_not(x) → bigint</code> | Bitwise NOT function |
| <code>bitwise_or</code> | <code>bitwise_or(x, y) → bigint</code> | Bitwise OR function |
| <code>bitwise_xor</code> | <code>bitwise_xor(x, y) → bigint</code> | Bitwise XOR function |
| <code>bitwise_and_agg</code> | <code>bitwise_and_agg(x) → bigint</code> | Returns the bitwise AND of all input values in x, which is an array. |
| <code>bitwise_or_agg</code> | <code>bitwise_or_agg(x) → bigint</code> | Returns the bitwise OR of all input values in x, which is an array. |

Examples:

```
SELECT bit_count ( 9 , 64 ); -- 2
SELECT bit_count ( 9 , 8 ); -- 2
SELECT bit_count (- 7 , 64 ); -- 62
```

```
SELECT bit_count (- 7 , 8 ); -- 6
```

7.3.2.2.7 Decimal function

Literal value

Use the following syntax to define the literal value of the `DECIMAL` type :

```
DECIMAL ' xxxx . yyyyy '
```

The `precision` of the `DECIMAL` type for the literal value is equal to the number of digits in the literal value (including leading 0s). The `scale` is equal to the number of digits in the fractional part (including trailing 0s). For example:

| Literal value | Data type |
|---------------------------------|-----------------|
| DECIMAL '0' | DECIMAL(1) |
| DECIMAL '12345' | DECIMAL(5) |
| DECIMAL '0000012345.1234500000' | DECIMAL(20, 10) |

Operators

- Arithmetic operators

Assume that variables `x` and `y` are of the `DECIMAL` type.

- `x`: `DECIMAL (xp , xs)`
- `y`: `DECIMAL (yp , ys)`

They observe the following rules when used in arithmetic operation:

- `x + y` or `x - y`
 - `precision = min(38, 1 + min(xs, ys) + min(xp-xs, yp-ys))`
 - `scale = max(xs, ys)`
- `x * y`
 - `precision = min(38, xp + yp)`
 - `scale = xs + ys`
- `x / y`
 - `precision = min(38, xp + ys + max(0, ys-xs))`
 - `scale = max(xs, ys)`
- `x % y`
 - `precision = min(xp - xs, yp - ys) + max(xs, bs)`
 - `scale = max(xs, ys)`

- Comparison operators

All standard comparison operators and `BETWEEN` operators work for the `DECIMAL` type.

- Unary decimal operators

The `-` operator performs negation for the `DECIMAL` type.

7.3.2.2.8 String functions

Concatenation operator

The `||` operator performs string concatenation.

String functions

The following table lists the string functions supported by Presto.

| Function | Syntax | Description |
|-----------------------------------|--|---|
| <code>chr</code> | <code>chr(n) → varchar</code> | Returns the <i>Unicode code point</i> (UCP) <code>n</code> as a single character. |
| <code>codepoint</code> | <code>codepoint(string) → integer</code> | Returns the UCP value of <code>string</code> . |
| <code>concat</code> | <code>concat(string1, ..., stringN) → varchar</code> | Returns the concatenation of <code>string1</code> , <code>string2</code> , ..., <code>stringN</code> . This function provides the same functionality as the <code> </code> operator. |
| <code>hamming_distance</code> | <code>hamming_distance(string1, string2) → bigint</code> | Returns the <i>Hamming distance</i> of <code>string1</code> and <code>string2</code> . Note that the two strings must have the same length. |
| <code>length</code> | <code>length(string) → bigint</code> | Returns the length of a string. |
| <code>levenshtein_distance</code> | <code>levenshtein_distance(string1, string2) → bigint</code> | Returns the <i>Levenshtein distance</i> of <code>string1</code> and <code>string2</code> . |
| <code>lower</code> | <code>lower(string) → varchar</code> | Converts a string into lowercase. |
| <code>upper</code> | <code>upper(string) → varchar</code> | Converts a string into uppercase. |
| <code>replace</code> | <code>replace(string, search) → varchar</code> | Removes all instances of <code>search</code> from <code>string</code> and fill in with empty characters. |
| <code>replace</code> | <code>replace(string, search, replace) → varchar</code> | Replaces all instances of <code>search</code> with <code>replace</code> in <code>string</code> . |
| <code>reverse</code> | <code>reverse(string) → varchar</code> | Returns a string with the characters in reverse order. |

| Function | Syntax | Description |
|---------------------|--|---|
| lpad | lpad(string, size, padstring) → varchar | Left pads <code>string</code> to <code>size</code> characters with <code>padstring</code> . <code>padstring</code> must not be empty, and <code>size</code> must not be 0. |
| rpadd | rpadd(string, size, padstring) → varchar | Right pads <code>string</code> to <code>size</code> characters with <code>padstring</code> . <code>padstring</code> must not be empty, and <code>size</code> must not be 0. |
| ltrim | ltrim(string) → varchar | Removes leading whitespace from string. |
| rtrim | rtrim(string) → varchar | Removes trailing whitespace from string. |
| split | split(string, delimiter) → array | Splits string on delimiter and returns an array. |
| split | split(string, delimiter, limit) → array | Splits string on delimiter and returns an array of a limited size. |
| split_part | split_part(string, delimiter, index) → varchar | Splits string on delimiter and returns the field <code>index</code> . Field indexes start from 1. |
| split_to_map | split_to_map(string, entryDelimiter, keyValueDelimiter) → map<varchar, varchar> | Splits string by entryDelimiter and keyValueDelimiter and returns a map. |
| strpos | strpos(string, substring) → bigint | Returns the starting position of the first instance of substring in string. Positions start from 1. If substring is not found, 0 is returned. |
| position | position(substring IN string) → bigint | Returns the starting position of the first instance of substring in string. |

| Function | Syntax | Description |
|---------------------|--|---|
| <code>substr</code> | <code>substr(string, start, [length]) → varchar</code> | Returns a substring from string of length from the starting position <code>start</code> . Positions start from 1. The length parameter is optional. |

Unicode functions

- `normalize (string) → varchar`

Transforms string with the [NFC Normalization Form](#).

- `normalize (string , form) → varchar`

Transforms string with the specified normalization form. `form` must be one of the following keywords:

- `NFD` Canonical Decomposition
- `NFC` Canonical Decomposition, followed by Canonical Composition
- `NFKD` Compatibility Decomposition
- `NFKC` Compatibility Decomposition, followed by Canonical Composition

- `to_utf8 (string) → varbinary`

Encodes string into a UTF-8 varbinary representation.

- `from_utf8 (binary , [replace]) → varchar`

Decodes binary data into a UTF-8 varbinary representation. Invalid sequences are replaced with `replace`, which is Unicode replacement character `U + FFFD` by default. This parameter is optional. Note that the replacement string `replace` must either be a single character or empty.

7.3.2.2.9 Regular expression

Presto supports all the regular expression functions that use the [Java Pattern](#) syntax, but there are a few exceptions:

- Multi-line mode
 - `? m` is used to enable the multi-line mode.
 - The line terminator is `\ n`.
 - The `? d` flag is not supported.
- Case-sensitive mode
 - `? i` is used to enable the case-sensitive mode.
 - The `? u` flag is not supported.
 - Context-sensitive matching is not supported.
 - Local-sensitive matching is not supported.
- Surrogate pairs are not supported.
 - For example, `U + 10000` must be expressed by `\ x { 10000 }`, not `\ uD800 \ uDC00`.
- Boundaries `\ b` are incorrectly processed for a non-spacing mark without a base character.
- `\ Q` and `\ E` are not supported in character classes (such as `[A - Z123]`).
- Unicode character classes (`\ p { prop }`) are supported with the following differences:
 - All underscores in names must be removed. For example, use `OldItalic` instead of `Old_Italic`.
 - Scripts must be specified directly, without the `Is`, `script =`, or `sc =` prefix. Example: `\ p { Hiragana }` instead of `\ p { script = Hiragana }`.
 - Blocks must be specified with the `In` prefix. The `block =` and `blk =` prefixes are not supported. Example: `\ p { InMongolia }`.
 - Categories must be specified directly, without the `Is`, `general_ca tegory =`, or `gc =` prefix. Example: `\ p { L }`.
 - Binary properties must be specified directly. Example: use `\ p { Noncharact erCodePoin t }` instead of `\ p { IsNonchara cterCodePo int }`.

Presto provides the following regular expression functions:

- `regexp_ext ract_all (string , pattern , [group]) → array < varchar >`

Returns the substrings matched by the regular expression `pattern` in `string`. If the `pattern` expression uses the grouping function, then the `group` parameter can be set to specify the *capturing group* to be matched by the regular expression.

Examples

```
SELECT  regexp_ext ract_all (' 1a 2b 14m ', '\ d +'); -- [ 1
, 2 , 14 ]
SELECT  regexp_ext ract_all (' 1a 2b 14m ', '(\ d +)([ a - z
]+)'); -- [' a ', ' b ', ' m ']
```

- `regexp_ext ract (string , pattern , [group]) → varchar`

The function and usage are similar to those of `regexp_ext ract_all`. The difference is that this function only returns the first substring that is matched by the regular expression.

Examples

```
SELECT  regexp_ext ract (' 1a 2b 14m ', '\ d +'); -- 1
SELECT  regexp_ext ract (' 1a 2b 14m ', '(\ d +)([ a - z
]+)'); -- ' a '
```

- `regexp_lik e (string , pattern) → boolean`

Evaluates the regular expression `pattern` and determines whether it is contained in `string`. If yes, `TRUE` is returned. If not, `FALSE` is returned. This function is similar to the `LIKE` operator of SQL, except that `LIKE` matches the

entire string, whereas this function returns `TRUE` when the string contains the substring matched with pattern.

Examples

```
SELECT regexp_like (' 1a 2b 14m ', '\ d + b '); -- true
```

- `regexp_replace (string , pattern , [replacement]) → varchar`

Replaces every instance of the substring that is matched by the regular expression `pattern` in `string` with `replacement`. `replacement` is optional and replaced by "" (deleting the matched substrings) if it is not specified.

Capturing groups can be referenced in `replacement` by using `$ g` (g is the group SN, starting from 1) for a numbered group or `${ name }` for a named group. A dollar sign `$` can be included in the `replacement` by escaping it with a backslash `\$`.

Examples

```
SELECT regexp_replace (' 1a 2b 14m ', '\ d +[ ab ] '); -- ' 14m '
SELECT regexp_replace (' 1a 2b 14m ', '(\ d +)([ ab ]) ', ' 3c $ 2 '); -- ' 3ca 3cb 14m '
```

- `regexp_split (string , pattern) → array < varchar >`

Splits a string by using the regular expression `pattern` and returns an array. Trailing empty strings are preserved.

Examples

```
SELECT regexp_split (' 1a 2b 14m ', '\ s *[ a - z ]+\ s *'); -- [' 1 ', ' 2 ', ' 14 ', ''] four elements
-- The last one is an empty character .
```

7.3.2.2.10 Binary functions

Concatenation operator

The `||` operator performs binary concatenation.

Binary functions

| Function | Syntax | Description |
|---------------------|--------------------------------------|--|
| <code>length</code> | <code>length(binary) → bigint</code> | Returns the length of a binary block in bytes. |

| Function | Syntax | Description |
|--------------------|---|--|
| concat | concat(binary1, ..., binaryN) → varbinary | Returns the concatenation of binary1, binary2, ..., binaryN. |
| to_base64 | to_base64(binary) → varchar | Encodes binary data into a Base64 string representation. |
| from_base64 | from_base64(string) → varbinary | Base64 decoding |
| to_base64url | to_base64url(binary) → varchar | Encodes binary into a Base64 string representation by using the URL safe alphabet. |
| from_base64url | from_base64url(string) → varbinary | Decodes binary data from a Base64-encoded string by using the URL safe alphabet. |
| to_hex | to_hex(binary) → varchar | Encodes binary data into a hex string representation. |
| from_hex | from_hex(string) → varbinary | Decodes binary data from a hex encoded string. |
| to_big_endian_64 | to_big_endian_64(bigint) → varbinary | Encodes bigint values into a 64-bit two's complement big endian format. |
| from_big_endian_64 | from_big_endian_64(binary) → bigint | Decodes bigint values from a 64-bit two's complement big endian binary. |
| to_ieee754_32 | to_ieee754_32(real) → varbinary | Encodes real in a 32-bit big endian binary in the IEEE 754 single-precision floating-point format. |
| to_ieee754_64 | to_ieee754_64(double) → varbinary | Encodes double in a 64-bit big endian binary in the IEEE 754 double-precision floating-point format. |
| crc32 | crc32(binary) → bigint | Computes the CRC-32 of binary. |

| Function | Syntax | Description |
|----------|------------------------------|--|
| md5 | md5(binary) → varbinary | Computes the MD5 hash of binary. |
| sha1 | sha1(binary) → varbinary | Computes the SHA-1 hash of binary. |
| sha256 | sha256(binary) → varbinary | Computes the SHA-256 hash of binary. |
| sha512 | sha512(binary) → varbinary | Computes the SHA-512 hash of binary. |
| xxhash64 | xxhash64(binary) → varbinary | Computes the xxHash 64 hash of binary. |

7.3.2.2.11 Date and time processing functions

Date and time operators

Presto supports two date and time operators: `+` and `-`.

Examples:

```

--- +
date '2012 - 08 - 08' + interval '2' day          ---
2012 - 08 - 10
time '01 : 00' + interval '3' hour                --- 04 :
00 : 00 . 000
timestamp '2012 - 08 - 08 01 : 00' + interval '29' hour
--- 2012 - 08 - 09 06 : 00 : 00 . 000
timestamp '2012 - 10 - 31 01 : 00' + interval '1' month
--- 2012 - 11 - 30 01 : 00 : 00 . 000
interval '2' day + interval '3' hour              --- 2
03 : 00 : 00 . 000
interval '3' year + interval '5' month            --- 3
- 5
--- -
date '2012 - 08 - 08' - interval '2' day          ---
2012 - 08 - 06
time '01 : 00' - interval '3' hour                --- 22 :
00 : 00 . 000
timestamp '2012 - 08 - 08 01 : 00' - interval '29' hour
--- 2012 - 08 - 06 20 : 00 : 00 . 000
timestamp '2012 - 10 - 31 01 : 00' - interval '1' month
--- 2012 - 09 - 30 01 : 00 : 00 . 000
interval '2' day - interval '3' hour              --- 1
21 : 00 : 00 . 000
interval '3' year - interval '5' month            ---
month 2 - 7

```

Time zone conversion

The `AT TIME ZONE` operator sets the time zone of a timestamp.

Examples:

```

SELECT timestamp ' 2012 - 10 - 31 01 : 00 UTC ';
--- 2012 - 10 - 31 01 : 00 : 00 . 000 UTC
SELECT timestamp ' 2012 - 10 - 31 01 : 00 UTC ' AT TIME
ZONE ' America / Los_Angele s ';
--- 2012 - 10 - 30 18 : 00 : 00 . 000 America / Los_Angele s

```

Date and time functions· **Basic functions**

| Function | Syntax | Description |
|-------------------------------|--|---|
| current_date | current_date -> date | Returns the current date as of the start of the query. |
| current_time | current_time -> time with time zone | Returns the current time as of the start of the query. |
| current_timestamp | current_timestamp -> timestamp with time zone | Returns the current timestamp as of the start of the query. |
| current_timezone | current_timezone() → varchar | Returns the current time zone. |
| date | date(x) → date | Parses the literal value of a date into a date. |
| from_iso8601_timestamp | from_iso8601_timestamp(string) → timestamp with time zone | Parses the literal value of an ISO 8601-formatted timestamp into a timestamp variable with a time zone. |
| from_iso8601_date | from_iso8601_date(string) → date | Parses the literal value of an ISO 8601-formatted date into a variable of the date type. |
| from_unixtime | from_unixtime(unixtime, [timezone_str]) → timestamp | Parses a UNIX timestamp into a timestamp variable. A time zone can be included. |

| Function | Syntax | Description |
|------------------------------|---|---|
| <code>from_unixtime</code> | <code>from_unixtime(unixtime, hours, minutes) → timestamp with time zone</code> | Parses a UNIX timestamp into a timestamp variable with a time zone, with <code>hours</code> and <code>minutes</code> indicating the time zone offset. |
| <code>localtime</code> | <code>localtime -> time</code> | Returns the current time as of the start of the query. |
| <code>localtimestamp</code> | <code>localtimestamp -> timestamp</code> | Returns the current timestamp as of the start of the query. |
| <code>now</code> | <code>now() → timestamp with time zone</code> | Returns the current time. This is an alias for <code>current_time</code> . |
| <code>to_iso8601</code> | <code>to_iso8601(x) → varchar</code> | Formats <code>x</code> as an ISO 8601 string. <code>x</code> can be <code>DATE</code> or <code>TIMESTAMP [with time zone]</code> . |
| <code>to_milliseconds</code> | <code>to_milliseconds(interval) → bigint</code> | Returns the number of milliseconds that have elapsed since 00:00 of the current day. |
| <code>to_unixtime</code> | <code>to_unixtime(timestamp) → double</code> | Returns a timestamp in the UNIX format. |



Notice:

The following SQL standard functions do not use parenthesis:

- `current_date`
- `current_time`
- `current_timestamp`
- `localtime`
- `localtimestamp`

- **Truncation function**

The truncation function truncates date and time values by the specified unit, and returns the date and time values of this unit. The usage is as follows:

```
date_trunc ( unit , x ) -> [ same as x ]
```

`unit` is one of:

- `second` : seconds
- `minute` : minutes
- `hour` : hours
- `day` : days
- `week` : weeks
- `month` : months
- `quarter` : quarters
- `year` : years

- **Interval functions**

Presto provides two functions for interval calculation, which are:

- `date_add (unit , value , timestamp)` → [same as input]

Adds an interval value of the type `unit` to a timestamp. Subtraction can be performed by using a negative value with a unit.

- `date_diff (unit , timestamp1 , timestamp2)` → `bigint`

Returns the interval between two timestamps expressed with a unit.

`unit` is one of the following:

- `ns` : nanoseconds
- `us` : microseconds
- `ms` : milliseconds
- `s` : seconds
- `m` : minutes
- `h` : hours
- `d` : days

- Date and time extraction functions

Presto provides the `extract` function to extract the specified fields from a date and time value, which is:

```
extract ( field FROM x ) → bigint
```

`x` is the date and time value, and `field` is the field to be extracted, which can be one of the following values:

- `YEAR` : year
- `QUARTER` : quarter
- `MONTH` : month
- `WEEK` : week
- `DAY` : day
- `DAY_OF_MON TH` : day of a month
- `DAY_OF_WEE K` : day of a week
- `DOW` : an alias for `DAY_OF_WEE K`
- `DAY_OF_YEA R` : day of a year
- `DOY` : an alias for `DAY_OF_YEA R`
- `YEAR_OF_WE EK` : year of an [ISO week](#)
- `YOW` : an alias for `YEAR_OF_WE EK`
- `HOUR` : hour
- `MINUTE` : minute
- `SECOND` : second
- `TIMEZONE_H OUR` : hour with a time zone
- `TIMEZONE_M INUTE` : minute with a time zone

For convenience, Presto provides the following helper functions:

| Function | Syntax | Description |
|---------------------------|---------------------------------------|--|
| <code>day</code> | <code>day(x) → bigint</code> | Returns the day of the month from <code>x</code> . |
| <code>day_of_month</code> | <code>day_of_month(x) → bigint</code> | This is an alias for <code>day</code> . |
| <code>day_of_week</code> | <code>day_of_week(x) → bigint</code> | Returns the day of the week from <code>x</code> . |

| Function | Syntax | Description |
|------------------------------|--|---|
| <code>day_of_year</code> | <code>day_of_year(x) → bigint</code> | Returns the day of the year from x. |
| <code>dow</code> | <code>dow(x) → bigint</code> | This is an alias for <code>day_of_week</code> . |
| <code>doy</code> | <code>doy(x) → bigint</code> | This is an alias for <code>day_of_year</code> . |
| <code>hour</code> | <code>hour(x) → bigint</code> | Returns the hour of the day from x. The value ranges from 0 to 23. |
| <code>minute</code> | <code>minute(x) → bigint</code> | Returns the minute from x. The value ranges from 0 to 59. |
| <code>month</code> | <code>month(x) → bigint</code> | Returns the month of the year from x. The value ranges from 1 to 12. |
| <code>quarter</code> | <code>quarter(x) → bigint</code> | Returns the quarter of the year from x. |
| <code>second</code> | <code>second(x) → bigint</code> | Returns the number of seconds from x. The value ranges from 0 to 59. |
| <code>timezone_hour</code> | <code>timezone_hour(timestamp) → bigint</code> | Returns the number of hours of the time zone offset from the timestamp. |
| <code>timezone_minute</code> | <code>timezone_minute(timestamp) → bigint</code> | Returns the number of minutes of the time zone offset from the timestamp. |
| <code>week</code> | <code>week(x) → bigint</code> | Returns the week of the year from x. The value ranges from 1 to 53. |
| <code>week_of_year</code> | <code>week_of_year(x) → bigint</code> | This is an alias for <code>week</code> . |
| <code>year</code> | <code>year(x) → bigint</code> | Returns the year from x. |
| <code>year_of_week</code> | <code>year_of_week(x) → bigint</code> | Returns the year of the week from x (ISO week). |

| Function | Syntax | Description |
|----------|-----------------|--|
| yow | yow(x) → bigint | This is an alias for <code>year_of_week</code> . |

• MySQL date functions

Presto provides two date parsing functions that are compatible with MySQL

`date_parse` and `str_to_date`.

- `date_format (timestamp , format) → varchar`

Formats `timestamp` as a string by using `format`.

- `date_parse (string , format) → timestamp`

Parses the literal value of a date by using `format`.

The following table lists the MySQL format specifiers supported by Presto.

| Specifier | Description |
|-----------|---|
| %a | Abbreviated weekday name (Sun .. Sat) |
| %b | Abbreviated month name (Jan .. Dec) |
| %c | Month, numeric (1 .. 12), which cannot be 0 |
| %d | Day of the month, numeric (01 .. 31), which cannot be 0 |
| %e | Day of the month, numeric (1 .. 31), which cannot be 0 |
| %f | Number of seconds (6 digits for printing : 000000 .. 999000; 1 - 9 digits for parsing : 0 .. 999999999) |
| %H | Hours (00 .. 23) |
| %h | Hours (01 .. 12) |
| %I | Hours (01 .. 12) |
| %i | Minutes (00 .. 59) |
| %j | Day of the year (001 .. 366) |
| %k | Hours (0 .. 23) |
| %l | Hours (1 .. 12) |
| %M | Month (January .. December) |

| Specifier | Description |
|-----------|---|
| %m | Month, numeric (01 .. 12) [4] |
| %p | AM / PM |
| %r | Time, 12-hour (hh:mm:ss AM/PM) |
| %S | Seconds (00 .. 59) |
| %s | Seconds (00 .. 59) |
| %T | Time, 24-hour (hh:mm:ss) |
| %v | Week (01 .. 53), where Monday is the first day of the week, used with % X |
| %W | Weekday name (Sunday .. Saturday) |
| %x | Year for the week, where Monday is the first day of the week, numeric, four digits |
| %Y | Year, numeric, four digits |
| %y | Year, numeric (two digits). During parsing, the two-digit year format assumes the range [1970, 2069]. |
| %% | A literal '%' character |

**Note:**

Currently, Presto does not support the following specifiers: % D , % U , % u , % V , % w , and % X .

- Java date functions

The following functions use a format string compatible with [JodaTime Pattern](#) of Java.

- `format_dat etime (timestamp , format) → varchar` : formats a timestamp.
- `parse_date time (string , format) → timestamp with time zone` : parses a string into a timestamp.

7.3.2.2.12 Aggregate functions

Aggregate functions have the following features:

- Input a dataset.

- Output a single computation result.

A majority of aggregate functions ignore `null` values during computation and return `null` when no input is made or all values are `null`, but there are a few exceptions:

- `count`
- `count_if`
- `max_by`
- `min_by`
- `approx_distinct`

Basic aggregate functions

| Function | Syntax | Description |
|------------------------|---|--|
| <code>arbitrary</code> | <code>arbitrary(x) → [same as input]</code> | Returns an arbitrary non-null value of x. |
| <code>array_agg</code> | <code>array_agg(x) → array<[same as input]></code> | Returns an array created from the input elements. |
| <code>avg</code> | <code>avg(x) → double</code> | Returns the average (arithmetic mean) of all input values. |
| <code>avg</code> | <code>avg(time interval type) → time interval type</code> | Returns the average interval length of all input time series. |
| <code>bool_and</code> | <code>bool_and(boolean) → boolean</code> | Returns TRUE if every input value is TRUE. Otherwise, FALSE is returned. |
| <code>bool_or</code> | <code>bool_or(boolean) → boolean</code> | Returns TRUE if any of the input values is True. Otherwise, FALSE is returned. |
| <code>checksum</code> | <code>checksum(x) → varbinary</code> | Returns an order-insensitive checksum of x. |
| <code>count</code> | <code>count(*) → bigint</code> | Returns the number of rows. |
| <code>count</code> | <code>count(x) → bigint</code> | Returns the number of non-null elements. |

| Function | Syntax | Description |
|-----------------------------|---|--|
| <code>count_if</code> | <code>count_if(x) → bigint</code> | Returns the number of TRUE elements of x. This function is equivalent to <code>count (CASE WHEN x THEN 1 END)</code> . |
| <code>every</code> | <code>every(boolean) → boolean</code> | This is an alias for <code>bool_and</code> . |
| <code>geometric_mean</code> | <code>geometric_mean(x) → double</code> | Returns the geometric mean of x. |
| <code>max_by</code> | <code>max_by(x, y) → [same as x]</code> | Returns the value of x associated with the maximum value of y over all input values. |
| <code>max_by</code> | <code>max_by(x, y, n) → array<[same as x]></code> | Returns an array of x associated with the n largest of all input values of y. |
| <code>min_by</code> | <code>min_by(x, y) → [same as x]</code> | Returns the value of x associated with the minimum value of y over all input values. |
| <code>min_by</code> | <code>min_by(x, y, n) → array<[same as x]></code> | Returns an array of x associated with the n smallest of all input values of y. |
| <code>max</code> | <code>max(x) → [same as input]</code> | Returns the maximum value among all input values. |
| <code>max</code> | <code>max(x, n) → array<[same as x]></code> | Returns the n largest values of all input values of x. |
| <code>min</code> | <code>min(x) → [same as input]</code> | Returns the minimum value among all input values. |
| <code>min</code> | <code>min(x, n) → array<[same as x]></code> | Returns the n smallest values of all input values of x. |

| Function | Syntax | Description |
|----------|--------------------------|---|
| sum | sum(x) → [same as input] | Returns the sum of all input values of x. |

Bitwise aggregate functions

For bitwise aggregate functions, see `bitwise_and_agg` and `bitwise_or_agg` functions described in [Bitwise operators](#).

Map aggregate functions

| Function | Syntax | Description |
|--------------|---|--|
| histogram | histogram(x) → map<K, bigint> | Creates a statistics histogram. |
| map_agg | map_agg(key, value) → map<K,V> | Creates a variable of the <code>MAP</code> type. |
| map_union | map_union(x<K, V>) → map<K,V> | Returns the union of all the input maps. If a key is found in multiple input maps, the key value in the resulting map comes from an arbitrary input map. |
| multimap_agg | multimap_agg(key, value) → map<K,array> | Creates a variable of the <code>MAP</code> type with multiple mappings. |

Approximate aggregate functions

| Function | Syntax | Description |
|--------------------------------|---|---|
| <code>approx_distinct</code> | <code>approx_distinct(x, [e]) → bigint</code> | Returns the approximate number of rows that contain distinct input values. This function provides an approximation of <code>count (DISTINCT x)</code> . 0 is returned if all input values are null. This function produces a standard error of no more than <code>e</code> , which is the standard deviation of the (approximately normal) error distribution over all possible sets. It is optional and set to 2.3% by default. The current implementation of this function requires that <code>e</code> be in the range [0.01150, 0.26000]. It does not guarantee an upper limit on the error for any specific input set. |
| <code>approx_percentile</code> | <code>approx_percentile(x, percentage) → [same as x]</code> | Returns the approximate percentile for all input values of x at the given percentage. |
| <code>approx_percentile</code> | <code>approx_percentile(x, percentages) → array<[same as x]></code> | Similar to the preceding function, percentages is an array and returns constant values for all input rows. |
| <code>approx_percentile</code> | <code>approx_percentile(x, w, percentage) → [same as x]</code> | Similar to the preceding function, w is the weighted value of x. |

| Function | Syntax | Description |
|--------------------------------|--|--|
| <code>approx_percentile</code> | <code>approx_percentile(x, w, percentage, accuracy) → [same as x]</code> | Similar to the preceding function, <code>accuracy</code> is the upper limit of the estimation accuracy, and the value must be in the range [0, 1]. |
| <code>approx_percentile</code> | <code>approx_percentile(x, w, percentages) → array<[same as x]></code> | Similar to the preceding function, <code>percentages</code> is an array and returns constant values for all input rows. |
| <code>numeric_histogram</code> | <code>numeric_histogram(buckets, value, [weight]) → map<double, double></code> | Computes an approximate histogram with up to a given number of buckets. <code>buckets</code> must be a <code>BIGINT</code> . <code>value</code> and <code>weight</code> must be numeric. Weight is optional and set to 1 by default. |

Statistical aggregate functions

| Function | Syntax | Description |
|-------------------------|--|--|
| <code>corr</code> | <code>corr(y, x) → double</code> | Returns the correlation coefficient. |
| <code>covar_pop</code> | <code>covar_pop(y, x) → double</code> | Returns the population covariance of input values. |
| <code>covar_samp</code> | <code>covar_samp(y, x) → double</code> | Returns the sample covariance of input values. |

| Function | Syntax | Description |
|----------------|-------------------------------|---|
| kurtosis | kurtosis(x) → double | <p>Returns the excess kurtosis of all input values. Use the following expression for unbiased estimation:</p> <pre> kurtosis (x) = n (n + 1) / ((n - 1) (n - 2) (n - 3)) sum [(x_i - mean) ^ 4] / sttdev (x) ^ 4 - 3 (n - 1) ^ 2 / ((n - 2) (n - 3)) </pre> |
| regr_intercept | regr_intercept(y, x) → double | Returns the y-intercept of the linear regression line. <code>y</code> is the dependent variable. <code>x</code> is the independent variable. |
| regr_slope | regr_slope(y, x) → double | Returns the linear regression slope of input values. <code>y</code> is the dependent variable. <code>x</code> is the independent variable. |
| skewness | skewness(x) → double | Returns the skewness of all input values. |
| sttdev_pop | sttdev_pop(x) → double | Returns the population standard deviation of all input values. |
| sttdev_samp | sttdev_samp(x) → double | Returns the sample standard deviation of all input values. |
| sttdev | sttdev(x) → double | This is an alias for <code>sttdev_samp</code> . |
| var_pop | var_pop(x) → double | Returns the population variance of all input values. |
| var_samp | var_samp(x) → double | Returns the sample variance of all input values. |

| Function | Syntax | Description |
|-----------------|-----------------------------|--|
| variance | variance(x) → double | This is an alias for <code>var_samp</code> . |

7.3.2.3 SQL statements

7.3.2.3.1 SQL statement overview

DDL

- [ALTER SCHEMA](#)
- [ALTER TABLE](#)
- [CREATE SCHEMA](#)
- [CREATE TABLE](#)
- [CREATE TABLE AS](#)
- [CREATE VIEW](#)
- [DROP SCHEMA](#)
- [DROP TABLE](#)
- [DROP VIEW](#)
- [VALUES](#)
- [DESCRIBE](#)
- [SHOW CATALOGS](#)
- [SHOW COLUMNS](#)
- [SHOW CREATE TABLE](#)
- [SHOW CREATE VIEW](#)
- [SHOW FUNCTIONS](#)
- [SHOW PARTITIONS](#)
- [SHOW SCHEMAS](#)
- [SHOW TABLES](#)

DML

- [DELETE](#)
- [INSERT](#)
- [EXPLAIN](#)

- [EXPLAIN ANALYZE](#)

DQL

- Query
 - [SELECT](#)
- Precompilation
 - [PREPARE](#)
 - [EXECUTE](#)
 - [DEALLOCATE PREPARE](#)
 - [DESCRIBE INPUT](#)
 - [DESCRIBE OUTPUT](#)

7.3.2.3.2 ALTER SCHEMA

Overview

```
ALTER    SCHEMA    name    RENAME    TO    new_name
```

Description

Renames a schema.

Examples

```
ALTER    SCHEMA    web    RENAME    TO    traffic -- Changes    the
name    of    a    schema    from    ' web '    to    ' traffic '.
```

7.3.2.3.3 ALTER TABLE

Overview

```
ALTER    TABLE    name    RENAME    TO    new_name
ALTER    TABLE    name    ADD    COLUMN    column_name    data_type
ALTER    TABLE    name    DROP    COLUMN    column_name
ALTER    TABLE    name    RENAME    COLUMN    column_name    TO
new_column_name
```

Description

Changes the definition of an existing table.

Examples

```
ALTER    TABLE    users    RENAME    TO    people ; ---    Renames    a
table .
ALTER    TABLE    users    ADD    COLUMN    zip    varchar ; ---    Adds    a
column .
```



```
ALTER TABLE users DROP COLUMN zip ; --- Deletes a
column .
ALTER TABLE users RENAME COLUMN id TO user_id ; ---
Renames a column .
```

7.3.2.3.4 CALL

Overview

```
CALL procedure_name ( [ name => ] expression [, ...] )
```

Description

Calls a stored procedure. Connectors can provide stored procedures to perform data manipulation or administrative tasks. Some connectors, such as the PostgreSQL connector, are intended for underlying systems that have their own stored procedures. These systems must use the stored procedures provided by the connectors to access their own stored procedures, which are not directly callable through `CALL`.

Examples

```
CALL test ( 123 , ' apple '); --- Calls a stored procedure
by using positional arguments .
CALL test ( name => ' apple ', id => 123 ); --- Calls a
stored procedure by using named arguments .
CALL catalog . schema . test (); --- Calls a stored
procedure using a fully qualified name .
```

7.3.2.3.5 COMMIT

Overview

```
COMMIT [ WORK ]
```

Description

Commits the current transaction.

Examples

```
COMMIT ;
COMMIT WORK ;
```

7.3.2.3.6 CREATE SCHEMA

Overview

```
CREATE SCHEMA [ IF NOT EXISTS ] schema_name
```

```
[ WITH ( property_name = expression [, ...] ) ]
```

Description

Creates a schema. A schema is a container that holds tables, views, and other database objects.

- Use the `IF NOT EXISTS` clause to suppress the exception that is thrown when the schema to be created already exists.
- Use the `WITH` clause to set properties for the new schema. To list all available schema properties, run the following statement:

```
SELECT * FROM system . metadata . schema_properties ;
```

Examples

```
CREATE SCHEMA web ;
CREATE SCHEMA hive . sales ;
CREATE SCHEMA IF NOT EXISTS traffic ;
```

7.3.2.3.7 CREATE TABLE

Overview

```
CREATE TABLE [ IF NOT EXISTS ]
table_name (
    { column_name data_type [ COMMENT comment ]
    | LIKE existing_table_name [ { INCLUDING | EXCLUDING }
    PROPERTIES ] }
    [, ...]
)
[ COMMENT table_comment ]
[ WITH ( property_name = expression [, ...] ) ]
```

Description

Creates an empty table. Use `CREATE TABLE AS` to create a table from an existing dataset.

- Use the `IF NOT EXISTS` clause to suppress the exception that is thrown when the table to be created already exists.
- Use the `WITH` clause to set properties for the new table. To list all available table properties, run the following statement:

```
SELECT * FROM system . metadata . table_properties ;
```

- Use the `LIKE` clause to reference the column definitions of an existing table. You can specify multiple `LIKE` clauses.

- Use `INCLUDING PROPERTIES` to reference the properties of an existing table when creating a table. If the `WITH` clause specifies the same property name as one of the properties that is specified by `INCLUDING PROPERTIES`, the value from the `WITH` clause is used. The default behavior is `EXCLUDING PROPERTIES`.

Examples

```

--- Create a table named " orders "
CREATE TABLE orders (
  orderkey bigint ,
  orderstatus varchar ,
  totalprice double ,
  orderdate date
)
WITH ( format = ' ORC ')

--- Create a table named " orders " and add a comment
CREATE TABLE IF NOT EXISTS orders (
  orderkey bigint ,
  orderstatus varchar ,
  totalprice double COMMENT ' Price in cents .',
  orderdate date
)
COMMENT ' A table to keep track of orders .'

--- Create a table named " bigger_orders " and reference
some column definitions from the table " orders "
CREATE TABLE bigger_orders (
  another_orderkey bigint ,
  LIKE orders ,
  another_orderdate date
)

```

7.3.2.3.8 CREATE TABLE AS

Overview

```

CREATE TABLE [ IF NOT EXISTS ] table_name [ ( column_aliases , ... ) ]
[ COMMENT table_comment ]
[ WITH ( property_name = expression [, ...] ) ]
AS query
[ WITH [ NO ] DATA ]

```

Description

Creates a new table that contains data from a `SELECT` query.

- Use the `IF NOT EXISTS` clause to suppress the exception that is thrown when the table to be created already exists.

- Use the `WITH` clause to set properties for the new table. To list all available table properties, run the following statement:

```
SELECT * FROM system . metadata . table_prop erties ;
```

Examples

```
--- Select two columns from the table " orders " to
create a table
CREATE TABLE orders_col umn_aliase d ( order_date ,
total_pric e )
AS
SELECT orderdate , totalprice
FROM orders

--- Create a table by using an aggregate function
CREATE TABLE orders_by_ date
COMMENT ' Summary of orders by date '
WITH ( format = ' ORC ')
AS
SELECT orderdate , sum ( totalprice ) AS price
FROM orders
GROUP BY orderdate

--- Create a table by using the ` IF NOT EXISTS `
clause
CREATE TABLE IF NOT EXISTS orders_by_ date AS
SELECT orderdate , sum ( totalprice ) AS price
FROM orders
GROUP BY orderdate

--- Create a table that has the same schema as the
table " nation " but contains no data
CREATE TABLE empty_nati on AS
SELECT *
FROM nation
WITH NO DATA
```

7.3.2.3.9 CREATE VIEW

Overview

```
CREATE [ OR REPLACE ] VIEW view_name AS query
```

Description

Creates a view. A view is a logic table that does not contain any data. It can be referenced by future queries. The statement for defining a view is executed each time the query references the view.

Use the `OR REPLACE` clause to suppress the exception that is thrown when the view exists.

Examples

```

--- Select two columns from the table " orders " to
create a table
CREATE TABLE orders_column_alias ( order_date ,
total_price )
AS
SELECT orderdate , totalprice
FROM orders

--- Create a table by using an aggregate function
CREATE TABLE orders_by_date
COMMENT ' Summary of orders by date '
WITH ( format = ' ORC ' )
AS
SELECT orderdate , sum ( totalprice ) AS price
FROM orders
GROUP BY orderdate

--- Create a table by using the ` IF NOT EXISTS `
clause
CREATE TABLE IF NOT EXISTS orders_by_date AS
SELECT orderdate , sum ( totalprice ) AS price
FROM orders
GROUP BY orderdate

--- Create a table that has the same schema as the
table " nation " but contains no data
CREATE TABLE empty_nation AS
SELECT *
FROM nation
WITH NO DATA
)

```

7.3.2.3.10 DEALLOCATE PREPARE

Overview

```
DEALLOCATE PREPARE statement_name
```

Description

Removes a named statement from a session.

Examples

```
--- Release the query named my_query
```

```
DEALLOCATE    PREPARE    my_query ;
```

7.3.2.3.11 DELETE

Overview

```
DELETE    FROM    table_name    [    WHERE    condition    ]
```

Description

Deletes the rows that match the WHERE clause from a table, or deletes all the rows if the WHERE clause is not specified.

Examples

```
--- Delete matching rows
DELETE FROM lineitem WHERE shipmode = 'AIR';

--- Delete matching rows
DELETE FROM lineitem
WHERE orderkey IN ( SELECT orderkey FROM orders WHERE
priority = 'LOW ');

--- Clear a table
DELETE FROM orders ;
```

Limits

Some connectors may not support `DELETE`.

7.3.2.3.12 DESCRIBE

Overview

```
DESCRIBE    table_name
```

Description

Retrieves table definitions. It is equivalent to SHOW COLUMNS.

Examples

```
DESCRIBE orders ;
```

7.3.2.3.13 DESCRIBE INPUT

Overview

```
DESCRIBE INPUT statement_ name
```

Description

Lists the parameters of a precompiled query statement and the position and type of each parameter.

Examples

```
--- Create a precompiled query statement 'my_select1'
PREPARE my_select1 FROM
SELECT ? FROM nation WHERE regionkey = ? AND name < ? ;

--- Get the descriptive information about the
precompiled statement
DESCRIBE INPUT my_select1 ;
```

Query results:

| Position | Type |
|----------|---------|
| 0 | unknown |
| 1 | bigint |
| 2 | varchar |

(3 rows)

7.3.2.3.14 DESCRIBE OUTPUT

Overview

```
DESCRIBE OUTPUT statement_ name
```

Description

Lists all the column information about output results, including the column name (or alias), catalog, schema, table name, type, type size in bytes, and a boolean that indicates whether the column is aliased.

Examples

Example 1

Create a precompiled query statement:

```
PREPARE my_select1 FROM
SELECT * FROM nation ;
```

Execute `DESCRIBE OUTPUT` , which returns the following output:

```
DESCRIBE OUTPUT my_select1 ;

Column  Name | Catalog | Schema | Table | Type | Type
Size  | Aliased
-----+-----+-----+-----+-----+-----
+-----+
nationkey | tpch    | sf1     | nation | bigint |
8 | false
name      | tpch    | sf1     | nation | varchar |
0 | false
regionkey | tpch    | sf1     | nation | bigint |
8 | false
comment   | tpch    | sf1     | nation | varchar |
0 | false
( 4 rows )
```

Example 2

```
PREPARE my_select2 FROM
SELECT count (*) as my_count , 1 + 2 FROM nation
```

Execute `DESCRIBE OUTPUT` , which returns the following output:

```
DESCRIBE OUTPUT my_select2 ;

Column  Name | Catalog | Schema | Table | Type | Type
Size  | Aliased
-----+-----+-----+-----+-----+-----
+-----+
my_count |         |         |         | bigint | 8 |
true
_col1    |         |         |         | bigint | 8 |
false
( 2 rows )
```

Example 3

```
PREPARE my_create FROM
CREATE TABLE foo AS SELECT * FROM nation ;
```

Execute `DESCRIBE OUTPUT` , which returns the following output:

```
DESCRIBE OUTPUT my_create ;

Column  Name | Catalog | Schema | Table | Type | Type
Size  | Aliased
-----+-----+-----+-----+-----+-----
+-----+
rows    |         |         |         | bigint | 8 |
false
```



```
( 1 row )
```

7.3.2.3.15 DROP SCHEMA

Overview

```
DROP SCHEMA [ IF EXISTS ] schema_name
```

Description

Deletes an existing schema.

- The schema must be empty.
- Use the `IF EXISTS` clause to suppress the exception that is thrown when the schema to be deleted does not exist.

Examples

```
DROP SCHEMA web ;
DROP TABLE IF EXISTS sales ;
```

7.3.2.3.16 DROP TABLE

Overview

```
DROP TABLE [ IF EXISTS ] table_name
```

Description

Deletes a data table. Use the `IF EXISTS` clause to suppress the exception that is thrown when the table to be deleted does not exist.

Examples

```
DROP TABLE orders_by_date ;
DROP TABLE IF EXISTS orders_by_date ;
```

7.3.2.3.17 DROP VIEW

Overview

```
DROP VIEW [ IF EXISTS ] view_name
```

Description

Deletes a data table. Use the `IF EXISTS` clause to suppress the exception that is thrown when the table to be deleted does not exist.

Examples

```
DROP VIEW orders_by_date ;
DROP VIEW IF EXISTS orders_by_date ;
```

7.3.2.3.18 EXECUTE

Overview

```
EXECUTE statement_name [ USING parameter1 [ , parameter2
, ... ] ]
```

Description

Executes a precompiled query statement. Arguments are defined in the `USING` clause.

Examples

- **Example 1:**

```
PREPARE my_select1 FROM
SELECT name FROM nation ;
--- Execute a precompiled query statement
EXECUTE my_select1 ;
Example 2
```

- **Example 2:**

```
PREPARE my_select2 FROM
SELECT name FROM nation WHERE regionkey = ? and
nationkey < ? ;
--- Execute a precompiled query statement
EXECUTE my_select2 USING 1 , 3 ;
--- The preceding statement is equivalent to this
statement :
SELECT name FROM nation WHERE regionkey = 1 AND
nationkey < 3 ;
```

7.3.2.3.19 EXPLAIN

Overview

```
EXPLAIN [ ( option [, ...] ) ] statement

" option " can be one of :

    FORMAT { TEXT | GRAPHVIZ }
    TYPE   { LOGICAL | DISTRIBUTE D | VALIDATE }
```

Description

Implements one of the following functions based on the used option:

- Shows the logical plan of a query statement.
- Shows the distributed execution plan of a query statement.
- Validates a query statement.

Use the `TYPE` `DISTRIBUTE` `D` option to display plan fragments. Each plan fragment is executed by a single or multiple Presto nodes. Fragments separation represents the data exchange between Presto nodes. The fragment type specifies how the fragment is executed by Presto nodes and how data is distributed between fragments. The following fragment types are available:

- `SINGLE` : Fragments are executed on a single node.
- `HASH` : Fragments are executed on a fixed number of nodes with the input data distributed by using a hash function.
- `ROUND_ROBIN` : Fragments are executed on a fixed number of nodes with the input data distributed in a `ROUND - ROBIN` fashion.
- `BROADCAST` : Fragments are executed on a fixed number of nodes with the input data broadcast to all nodes.
- `SOURCE` : Fragments are executed on nodes where data is stored.

Examples

- Example 1:

```
PREPARE my_select1 FROM
SELECT name FROM nation ;
--- Execute a precompiled query statement
EXECUTE my_select1 ;
Example 2
```

- Example 2:

```
PREPARE my_select2 FROM
SELECT name FROM nation WHERE regionkey = ? and
nationkey < ? ;
--- Execute a precompiled query statement
EXECUTE my_select2 USING 1 , 3 ;
--- The preceding statement is equivalent to this
statement :
```

```
SELECT name FROM nation WHERE regionkey = 1 AND
nationkey < 3 ;
```

7.3.2.3.20 EXPLAIN ANALYZE

Overview

```
EXPLAIN ANALYZE [ VERBOSE ] statement
```

Description

Executes the statement and shows the distributed execution plan of the statement along with the cost of each operation. The `VERBOSE` option gives more details and underlying statistics.

Examples

In the following example, you can view the CPU time spent in each stage and the relative cost of each plan node in the stage. Note that the relative costs of the plan nodes are calculated based on the actual time and may not be correlated to the CPU time. You can also view additional statistics on each plan node, which are useful if you want to detect data errors during a query (such as skewness and hash collisions).

```
presto : sf1 > EXPLAIN ANALYZE SELECT count (*), clerk FROM
orders WHERE orderdate > date ' 1995 - 01 - 01 ' GROUP BY
clerk ;
```

| | Query | Plan |
|--|--|------------------------------------|
| Fragment 1 | [HASH] | |
| Cost : | CPU 88 . 57ms , | Input : 4000 rows (148 . 44kB) , |
| Output : | 1000 rows (28 . 32kB) | |
| Output layout : | [count , clerk] | |
| Output partitioning : | SINGLE [] | |
| - Project [] => | [count : bigint , clerk : varchar (15)] | |
| Cost : | 26 . 24 % , | Input : 1000 rows (37 . 11kB) , |
| Output : | 1000 rows (28 . 32kB) , | Filtered : 0 . 00 % |
| Input avg .: | 62 . 50 lines , | Input std . dev .: |
| 14 . 77 % | | |
| - Aggregate (FINAL) [clerk] [\$ hashvalue] => | [clerk : varchar (15) , \$ hashvalue : bigint , count : bigint] | |
| Cost : | 16 . 83 % , | Output : 1000 rows (37 . 11kB) |
| Input avg .: | 250 . 00 lines , | Input std . dev .: |
| 14 . 77 % | | |
| count := " count "(" count_8 " | | |
| - LocalExchange [HASH] [\$ hashvalue] (" clerk ") => | clerk : varchar (15) , count_8 : bigint , \$ hashvalue : bigint | |
| Cost : | 47 . 28 % , | Output : 4000 rows (148 . 44kB) |
| Input avg .: | 4000 . 00 lines , | Input std . dev .: |
| 0 . 00 % | | |
| - RemoteSource [2] => | [clerk : varchar (15) , count_8 : bigint , \$ hashvalue_ 9 : bigint] | |

```

Cost : 9 . 65 %, Output : 4000 rows (
148 . 44kB )
Input avg .: 4000 . 00 lines , Input
std . dev .: 0 . 00 %

Fragment 2 [ tpch : orders : 1500000 ]
Cost : CPU 14 . 00s , Input : 818058 rows ( 22 . 62MB ),
Output : 4000 rows ( 148 . 44kB )
Output layout : [ clerk , count_8 , $ hashvalue_ 10 ]
Output partitioning : HASH [ clerk ][$ hashvalue_ 10 ]
- Aggregate ( PARTIAL ) [ clerk ][$ hashvalue_ 10 ] => [ clerk :
varchar ( 15 ), $ hashvalue_ 10 : bigint , count_8 : bigint ]
Cost : 4 . 47 %, Output : 4000 rows ( 148 . 44kB )
Input avg .: 204514 . 50 lines , Input std . dev
.: 0 . 05 %
Collisions avg .: 5701 . 28 ( 17569 . 93 % est . ),
Collisions std . dev .: 1 . 12 %
count_8 := " count " (*)
- ScanFilter Project [ table = tpch : tpch : orders : sf1
. 0 , originalConstraint = (" orderdate " > "$ literal $ date "(
BIGINT ' 9131 ')), filterPredicate = (" orderdate " > "$ literal
$ date "( BIGINT ' 9131 '))] => [ cler
Cost : 95 . 53 %, Input : 1500000 rows ( 0B ),
Output : 818058 rows ( 22 . 62MB ), Filtered : 45 . 46 %
Input avg .: 375000 . 00 lines , Input std .
dev .: 0 . 00 %
$ hashvalue_ 10 := " combine_hash "( BIGINT ' 0
', COALESCE (" $ operator $ hash_code "(" clerk " ), 0 ))
orderdate := tpch : orderdate
clerk := tpch : clerk

```

When the `VERBOSE` option is used, some operators may report additional information.

```

EXPLAIN ANALYZE VERBOSE SELECT count ( clerk ) OVER ( ) FROM
orders WHERE orderdate > date ' 1995 - 01 - 01 ';

Query Plan
-----
...
- Window [] => [ clerk : varchar ( 15 ), count : bigint ]
Cost : { rows : ?, bytes : ?}
CPU fraction : 75 . 93 %, Output : 8130 rows
( 230 . 24kB )
Input avg .: 8130 . 00 lines , Input std .
dev .: 0 . 00 %
Active Drivers : [ 1 / 1 ]
Index size : std . dev .: 0 . 00 bytes , 0 .
00 rows
Index count per driver : std . dev .: 0 .
00
Rows per driver : std . dev .: 0 . 00
Size of partition : std . dev .: 0 . 00
count := count ( " clerk " )

```

...

7.3.2.3.21 GRANT

Overview

```
GRANT ( privilege [, ...] | ( ALL PRIVILEGES ) )
ON [ TABLE ] table_name TO ( grantee | PUBLIC )
[ WITH GRANT OPTION ]
```

Description

Grants specific permissions to the specified grantee.

- Specifying `ALL PRIVILEGES` grants the DELETE, INSERT, and SELECT permissions.
- Specifying `PUBLIC` grants permissions to all users.
- The `WITH GRANT OPTION` clause allows the grantee to grant these permissions to others.

Examples

```
GRANT INSERT , SELECT ON orders TO alice ; --- Grants
permission s to user Alice
GRANT SELECT ON nation TO alice WITH GRANT OPTION
; --- Grants the SELECT permission to user Alice and
allows Alice to grant the `SELECT` permission to
others .
GRANT SELECT ON orders TO PUBLIC ; --- Grants the `
SELECT ` permission on the table " order " to everyone .
```

Limits

Some connectors may not support `GRANT` .

7.3.2.3.22 INSERT

Overview

```
INSERT INTO table_name [ ( column [, ... ] ) ] query
```

Description

Inserts data. If a list of column names is specified, they must exactly match the list of columns returned by the `query` . Each column in the table not present in the column list is filled with a `null` value.

Examples

```
INSERT INTO orders SELECT * FROM new_orders ; --- Inserts
the SELECT results into the table " orders ".
INSERT INTO cities VALUES ( 1 , ' San Francisco '); ---
Inserts a data row .
INSERT INTO cities VALUES ( 2 , ' San Jose '), ( 3 , '
Oakland '); --- Inserts multiple rows .
INSERT INTO nation ( nationkey , name , regionkey , comment
) VALUES ( 26 , ' POLAND ', 3 , ' no comment '); --- Inserts
a single row .
INSERT INTO nation ( nationkey , name , regionkey ) VALUES
( 26 , ' POLAND ', 3 ); --- Inserts a single row ( which
only includes some columns ).
```

7.3.2.3.23 PREPARE

Overview

```
PREPARE statement_ name FROM statement
```

Description

Creates a precompiled statement for subsequent executions. A precompiled statement is a set of query statements saved to a session. The statement can include arguments that are used to enter actual values during execution. Arguments are represented by ? .

Examples

```
--- Prepare a query that does not include arguments
PREPARE my_select1 FROM
SELECT * FROM nation ;

--- Prepare a query that includes arguments
PREPARE my_select2 FROM
SELECT name FROM nation WHERE regionkey = ? AND
nationkey < ? ;

--- Prepare an INSERT statement that does not include
arguments
PREPARE my_insert FROM
INSERT INTO cities VALUES ( 1 , ' San Francisco ');
```

7.3.2.3.24 RESET SESSION

Overview

```
RESET SESSION name
RESET SESSION catalog . name
```

Description

Resets a session to use the default properties.

Examples

```
RESET SESSION optimize_h ash_genera tion ;
RESET SESSION hive . optimized_ reader_ena bled ;
```

7.3.2.3.25 REVOKE

Overview

```
REVOKE [ GRANT OPTION FOR ]
( privilege [, ...] | ALL PRIVILEGES )
ON [ TABLE ] table_name FROM ( grantee | PUBLIC )
```

Description

Revokes the specified permissions from the specified user.

- `ALL PRIVILEGE` revokes the `SELECT` , `INSERT` , and `DELETE` permissions.
- Specifying `PUBLIC` revokes permissions from the `PUBLIC` role. The permissions that are assigned by other roles are retained.
- The `GRANT OPTION FOR` clause revokes the permissions that are assigned by using `GRANT` .
- `grantee` may be a single user or a role.

Examples

```
--- Revoke the INSERT and SELECT permission s on the
table " orders " from user Alice
REVOKE INSERT , SELECT ON orders FROM alice ;

--- Revoke the SELECT permission on the table " nation
" from all users
--- In addition , revoke the SELECT permission on the
table that all users assign to other users
REVOKE GRANT OPTION FOR SELECT ON nation FROM PUBLIC
;

--- Revoke all the permission s on the table " test "
from user Alice
REVOKE ALL PRIVILEGES ON test FROM alice ;
```

Limits

Some connectors do not support `REVOKE` .

7.3.2.3.26 ROLLBACK

Overview

```
ROLLBACK [ WORK ]
```

Description

Rolls back the current transaction.

Examples

```
ROLLBACK ;
ROLLBACK WORK ;
```

7.3.2.3.27 SELECT clause

7.3.2.3.27.1 *SELECT*

Overview

```
[ WITH with_query [, ...] ]
SELECT [ ALL | DISTINCT ] select_exp r [, ...]
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ GROUP BY [ ALL | DISTINCT ] grouping_element [, ...] ]
[ HAVING condition ]
[ { UNION | INTERSECT | EXCEPT } [ ALL | DISTINCT ] select
]
[ ORDER BY expression [ ASC | DESC ] [, ...] ]
[ LIMIT [ count | ALL ] ]
```

`from_item` is in either of the following two formats:

```
table_name [ [ AS ] alias [ ( column_alias [, ...] ) ] ]
```

```
from_item join_type from_item [ ON join_condition | USING
( join_column [, ...] ) ]
```

`join_type` is in one of the following formats:

- [INNER] JOIN
- LEFT [OUTER] JOIN
- RIGHT [OUTER] JOIN
- FULL [OUTER] JOIN
- CROSS JOIN

`grouping_element` is in one of the following formats:

- ()

- **expression**
- **GROUPING SETS** ((column [, ...]) [, ...])
- **CUBE** (column [, ...])
- **ROLLUP** (column [, ...])

Description

Retrieves rows from zero or more tables to get data sets. For more information, see the following sections:

- [WITH clause](#)
- [GROUP BY clause](#)
- [HIVING clause](#)
- [UNION|INTERSECT|EXCEPT clause](#)
- [ORDER BY clause](#)
- [LIMIT clause](#)
- [TABLESAMPLE](#)
- [UNNEST](#)
- [Joins](#)
- [Subquery](#)

7.3.2.3.27.2 WITH clause

Basic functions

The **WITH** clause defines named relations for use within a query, which flattens nested queries or simplifies subqueries. For example, the following two queries are equivalent:

```
--- The WITH clause is not used .
SELECT  a , b
FROM    (
    SELECT  a , MAX ( b ) AS  b FROM  t GROUP BY  a
) AS  x ;

--- The WITH clause is used , and the query statement
    is clearer .
WITH  x AS ( SELECT  a , MAX ( b ) AS  b FROM  t GROUP
BY  a )
```

```
SELECT  a , b  FROM  x ;
```

Define multiple subqueries

Use the `WITH` clause to define multiple subqueries:

```
WITH
  t1 AS ( SELECT  a , MAX ( b ) AS b  FROM  x  GROUP  BY
    a ),
  t2 AS ( SELECT  a , AVG ( d ) AS d  FROM  y  GROUP  BY
    a )
SELECT  t1 .*, t2 . *
FROM    t1
JOIN    t2  ON  t1 . a  =  t2 . a ;
```

Form a chain structure

Additionally, the relations within a `WITH` clause can form a chain structure.

```
WITH
  x AS ( SELECT  a  FROM  t ),
  y AS ( SELECT  a  AS b  FROM  x ),
  z AS ( SELECT  b  AS c  FROM  y )
SELECT  c  FROM  z ;
```

7.3.2.3.27.3 GROUP BY clause

Basic functions

The `GROUP BY` clause divides the output of a `SELECT` statement into groups. The `GROUP BY` clause may contain any expression that consists of column names or column SNs (starting from 1).

The following queries are equivalent (the SN of the `nationkey` column is 2).

```
--- Use the column SN
SELECT count (*), nationkey FROM customer GROUP BY 2 ;

--- Use the column name
SELECT count (*), nationkey FROM customer GROUP BY
nationkey ;
```

The `GROUP BY` clause can group columns that are not specified in the output list.

Example:

```
--- The mktsegment column is not specified in the
SELECT list .
--- The result set does not contain the content of
the mktsegment column .
SELECT count (*) FROM customer GROUP BY mktsegment ;

_col0
-----
29968
30142
```

```

30189
29949
29752
( 5 rows )

```

When a `GROUP BY` clause is used in a `SELECT` statement, all the output expressions must be either aggregate functions or columns present in the `GROUP BY` clause.

Complex grouping operations

Presto supports the following three complex aggregation syntaxes, allowing you to perform analysis that requires aggregation on multiple sets of columns in a single query:

- `GROUPING SETS`
- `CUBE`
- `ROLLUP`

The complex grouping operations of Presto only support column names and column SNs, but do not support expressions.

GROUPING SETS

`GROUPING SETS` performs grouping and aggregation of multiple columns in a single query statement. The columns not present in the group list are padded with `NULL`.

The "shipping" table contains five columns as follows:

```

SELECT * FROM shipping ;

 origin_state | origin_zip | destination_state |
 destination_zip | package_weight
-----+-----+-----+-----+
+-----+
 California | 94131 | New Jersey |
8648 | 13
 California | 94131 | New Jersey |
8540 | 42
 New Jersey | 7081 | Connecticut |
6708 | 225
 California | 90210 | Connecticut |
6927 | 1337
 California | 94131 | Colorado |
80302 | 5
 New York | 10002 | New Jersey |
8540 | 3

```

```
( 6 rows )
```

To retrieve the following grouping results by using a single query statement, do as follows:

- Group by `origin_state`, and get the total `package_weight`.
- Group by `origin_state` and `origin_zip`, and get the total `package_weight`.
- Group by `destination_state`, and get the total `package_weight`.

`GROUPING SETS` retrieves the result set of the preceding three groups with a single query statement, which is shown as follows:

```
SELECT origin_state , origin_zip , destination_state , sum (
package_weight )
FROM shipping
GROUP BY GROUPING SETS (
( origin_state ),
( origin_state , origin_zip ),
( destination_state ));
```

| origin_state | origin_zip | destination_state | _col0 |
|--------------|------------|-------------------|-------|
| New Jersey | NULL | NULL | 225 |
| California | NULL | NULL | 1397 |
| New York | NULL | NULL | 3 |
| California | 90210 | NULL | 1337 |
| California | 94131 | NULL | 60 |
| New Jersey | 7081 | NULL | 225 |
| New York | 10002 | NULL | 3 |
| NULL | NULL | Colorado | 5 |
| NULL | NULL | New Jersey | 58 |
| NULL | NULL | Connecticut | 1562 |

(10 rows)

The preceding query may be considered logically equivalent to a `UNION ALL` of multiple `GROUP BY` queries:

```
SELECT origin_state , NULL , NULL , sum ( package_weight )
FROM shipping GROUP BY origin_state

UNION ALL

SELECT origin_state , origin_zip , NULL , sum ( package_weight )
FROM shipping GROUP BY origin_state , origin_zip

UNION ALL

SELECT NULL , NULL , destination_state , sum ( package_weight )
FROM shipping GROUP BY destination_state ;
```

However, the `GROUPING SETS` query performs better because it reads the underlying table data only once, whereas the `UNION ALL` query reads the

underlying table data three times. This is why queries with `UNION ALL` may return inconsistent results when the underlying table data is not deterministic during the query period.

CUBE

The `CUBE` operator generates all possible grouping sets for a given set of columns. See the following code:

```
SELECT origin_state , destination_state , sum ( package_weight )
FROM shipping
GROUP BY CUBE ( origin_state , destination_state );
```

| origin_state | destination_state | _col0 |
|--------------|-------------------|-------|
| California | New Jersey | 55 |
| California | Colorado | 5 |
| New York | New Jersey | 3 |
| New Jersey | Connecticut | 225 |
| California | Connecticut | 1337 |
| California | NULL | 1397 |
| New York | NULL | 3 |
| New Jersey | NULL | 225 |
| NULL | New Jersey | 58 |
| NULL | Connecticut | 1562 |
| NULL | Colorado | 5 |
| NULL | NULL | 1625 |

(12 rows)

This query is equivalent to:

```
SELECT origin_state , destination_state , sum ( package_weight )
FROM shipping
GROUP BY GROUPING SETS (
    ( origin_state , destination_state ),
    ( origin_state ),
    ( destination_state ),
    ());
```

ROLLUP

The `ROLLUP` operator generates all possible subtotals for a given set of columns. See the following code:

```
SELECT origin_state , origin_zip , sum ( package_weight )
FROM shipping
GROUP BY ROLLUP ( origin_state , origin_zip );
```

| origin_state | origin_zip | _col2 |
|--------------|------------|-------|
| California | 94131 | 60 |
| California | 90210 | 1337 |
| New Jersey | 7081 | 225 |
| New York | 10002 | 3 |

| | | | | |
|------------|--|------|--|------|
| California | | NULL | | 1397 |
| New York | | NULL | | 3 |
| New Jersey | | NULL | | 225 |
| NULL | | NULL | | 1625 |
| (8 rows) | | | | |

This query is equivalent to:

```
SELECT origin_state , origin_zip , sum ( package_weight )
FROM shipping
GROUP BY GROUPING SETS (( origin_state , origin_zip ), (
origin_state ), ());
```

Combine multiple grouping expressions

The following three statements are equivalent:

```
SELECT origin_state , destination_state , origin_zip , sum (
package_weight )
FROM shipping
GROUP BY
    GROUPING SETS (( origin_state , destination_state )),
    ROLLUP ( origin_zip );
```

```
SELECT origin_state , destination_state , origin_zip , sum (
package_weight )
FROM shipping
GROUP BY
    GROUPING SETS (( origin_state , destination_state )),
    GROUPING SETS (( origin_zip ), ());
```

```
SELECT origin_state , destination_state , origin_zip , sum (
package_weight )
FROM shipping
GROUP BY GROUPING SETS (
    ( origin_state , destination_state , origin_zip ),
    ( origin_state , destination_state ));
```

The output is follows:

| origin_state | destination_state | origin_zip | _col3 |
|--------------|-------------------|------------|-------|
| New York | New Jersey | 10002 | 3 |
| California | New Jersey | 94131 | 55 |
| New Jersey | Connecticut | 7081 | 225 |
| California | Connecticut | 90210 | 1337 |
| California | Colorado | 94131 | 5 |
| New York | New Jersey | NULL | 3 |
| New Jersey | Connecticut | NULL | 225 |
| California | Colorado | NULL | 5 |
| California | Connecticut | NULL | 1337 |
| California | New Jersey | NULL | 55 |

```
( 10 rows )
```

In a `GROUP BY` clause, the `ALL` and `DISTINCT` quantifiers determine whether duplicate grouping sets each produce distinct output rows. See the following code:

```
SELECT origin_state , destination_state , origin_zip , sum (
package_weight )
FROM shipping
GROUP BY ALL
      CUBE ( origin_state , destination_state ),
      ROLLUP ( origin_state , origin_zip );
```

The preceding code is equivalent to:

```
SELECT origin_state , destination_state , origin_zip , sum (
package_weight )
FROM shipping
GROUP BY GROUPING SETS (
      ( origin_state , destination_state , origin_zip ),
      ( origin_state , origin_zip ),
      ( origin_state , destination_state , origin_zip ),
      ( origin_state , origin_zip ),
      ( origin_state , destination_state ),
      ( origin_state ),
      ( origin_state , destination_state ),
      ( origin_state ),
      ( origin_state , destination_state ),
      ( origin_state ),
      ( destination_state ),
      ( ));
```

Multiple duplicate grouping sets are available. If the query uses the `DISTINCT` quantifier, only unique grouping sets are generated.

```
SELECT origin_state , destination_state , origin_zip , sum (
package_weight )
FROM shipping
GROUP BY DISTINCT
      CUBE ( origin_state , destination_state ),
      ROLLUP ( origin_state , origin_zip );
```

The preceding code is equivalent to:

```
SELECT origin_state , destination_state , origin_zip , sum (
package_weight )
FROM shipping
GROUP BY GROUPING SETS (
      ( origin_state , destination_state , origin_zip ),
      ( origin_state , origin_zip ),
      ( origin_state , destination_state ),
      ( origin_state ),
      ( destination_state ),
      ( ));
```



Note:

The default qualifier for `GROUP BY` is `ALL`.

GROUPING function

Presto provides the `GROUPING` function that returns a bit set, and each bit indicates whether the corresponding column is present in a grouping condition. Syntax:

```
grouping ( col1 , ..., colN ) -> bigint
```

`GROUPING` is typically used in conjunction with `GROUPING SETS`, `ROLLUP`, `CUBE`, or `GROUP BY`. The columns in `GROUPING` must exactly map the columns that are referenced in the corresponding `GROUPING SETS`, `ROLLUP`, `CUBE`, or `GROUP BY` clause.

```
SELECT  origin_state , origin_zip , destination_state , sum (
package_weight ),
        grouping ( origin_state , origin_zip , destination_state
)
FROM    shipping
GROUP   BY GROUPING SETS (
        ( origin_state ),
        ( origin_state , origin_zip ),
        ( destination_state ));
```

| origin_state | origin_zip | destination_state | _col3 | _col4 |
|--------------|------------|-------------------|-------|-------|
| California | NULL | NULL | 1397 | 3 |
| New Jersey | NULL | NULL | 225 | |
| New York | NULL | NULL | 3 | |
| California | 94131 | NULL | 60 | 1 |
| New Jersey | 7081 | NULL | 225 | |
| California | 90210 | NULL | 1337 | 1 |
| New York | 10002 | NULL | 3 | |
| NULL | NULL | New Jersey | 58 | |
| NULL | NULL | Connecticut | 1562 | |
| NULL | NULL | Colorado | 5 | 6 |

(10 rows)

As shown in the preceding table, bits are assigned to the parameter columns with the rightmost column being the least significant bit. For a given `GROUPING`, a bit is set to `0` if the corresponding column is included in the grouping, and set to `1` if the corresponding column is not included in the grouping.

7.3.2.3.27.4 HAVING clause

The `HAVING` clause is used in conjunction with aggregate functions and the `GROUP BY` clause to control which groups are selected during a query. The `HAVING` clause is executed after grouping and aggregation to eliminate groups that do not satisfy the given conditions.

The following example selects users with an account balance greater than 5,700,000:

```
SELECT  count (*), mktsegment , nationkey ,
        CAST ( sum ( acctbal ) AS  bigint ) AS  totalbal
FROM    customer
GROUP   BY  mktsegment , nationkey
HAVING  sum ( acctbal ) > 5700000
ORDER   BY  totalbal  DESC ;
```

The output is as follows:

| _col0 | mktsegment | nationkey | totalbal |
|-------|------------|-----------|----------|
| 1272 | AUTOMOBILE | 19 | 5856939 |
| 1253 | FURNITURE | 14 | 5794887 |
| 1248 | FURNITURE | 9 | 5784628 |
| 1243 | FURNITURE | 12 | 5757371 |
| 1231 | HOUSEHOLD | 3 | 5753216 |
| 1251 | MACHINERY | 2 | 5719140 |
| 1247 | FURNITURE | 8 | 5701952 |

(7 rows)

7.3.2.3.27.5 Set operations

Overview

Presto supports three set operations: `UNION` , `INTERSECT` , and `EXCEPT` . These clauses merge the results of more than one query statement into a single result set.

The usage is as follows:

```
query  UNION  [ ALL | DISTINCT ] query
query  INTERSECT [ DISTINCT ] query
query  EXCEPT [ DISTINCT ] query
```

The arguments `ALL` and `DISTINCT` control which rows are included in the final result set. The default value is `DISTINCT` .

- `ALL` : Duplicate rows may be returned.
- `DISTINCT` : The result set is deduplicated.

`ALL` is not supported by `INTERSECT` and `EXCEPT` .

The three preceding set operations are processed from left to right. `INTERSECT` has the highest priority. That means `A UNION B INTERSECT C EXCEPT D` is the same as `A UNION (B INTERSECT C) EXCEPT D`.

UNION

`UNION` combines two query result sets and uses the `ALL` and `DISTINCT` arguments to control whether to remove duplicates.

Example 1:

```
SELECT 13
UNION
SELECT 42 ;

 _col0
-----
    13
    42
( 2  rows )
```

Example 2:

```
SELECT 13
UNION
SELECT * FROM ( VALUES 42 , 13 );

 _col0
-----
    13
    42
( 2  rows )
```

Example 3:

```
SELECT 13
UNION ALL
SELECT * FROM ( VALUES 42 , 13 );

 _col0
-----
    13
    42
    13
( 3  rows )
```

INTERSECT

`INTERSECT` returns only the rows that are in both query result sets.

Examples:

```
SELECT * FROM ( VALUES 13 , 42 )
INTERSECT
SELECT 13 ;
```

```

_col0
-----
      13
( 1 rows )

```

EXCEPT

EXCEPT returns the complement of the result sets of two queries.

```

SELECT * FROM ( VALUES 13 , 42 )
EXCEPT
SELECT 13 ;

_col0
-----
      42
( 1 rows )

```

7.3.2.3.27.6 ORDER BY clause

The **ORDER BY** clause sorts query results. Syntax:

```

ORDER BY expression [ ASC | DESC ] [ NULLS { FIRST |
LAST } ] [, ...]

```

Specifically:

- Each **expression** consists of column names or column SNs (starting from 1).
- The **ORDER BY** clause is executed after **GROUP BY** and **HAVING**.
- **NULLS { FIRST | LAST }** controls the sorting method of the **NULL** values (regardless of **ASC** or **DESC**). The default NULL order is **LAST**.

7.3.2.3.27.7 LIMIT clause

The **LIMIT** clause limits the number of rows in a result set. **LIMIT ALL** is the same as omitting the **LIMIT** clause.

Examples:

```

In this example , rows are returned arbitrarily
because the query lacks an ORDER BY clause .
SELECT orderdate FROM orders LIMIT 5 ;

orderdate
-----
1996 - 04 - 14
1992 - 01 - 15
1995 - 02 - 01
1995 - 11 - 12
1992 - 04 - 26

```

```
( 5 rows )
```

7.3.2.3.27.8 TABLESAMPLE

Presto provides two sampling methods: `BERNOULLI` and `SYSTEM`. Neither of the two methods can determine the number of rows of the sampling result set.

BERNOULLI

This sampling method selects each row of the sampled table based on a probability of the sample percentage. When a table is sampled by using this method, all the physical blocks of the table are scanned, and certain rows are skipped based on a comparison between the sample percentage and a random value that is calculated during runtime.

The probability of a row being included in the result is independent from that of any other row. This does not reduce the time required to read the sampled table from the disk. Further processing of the sampled output may have an impact on the total query time.

SYSTEM

This sampling method divides a table into logical segments of data and samples the table at this granularity. This sampling method either selects all the rows from a particular segment of data or skips it (based on a comparison between the sample percentage and a random value calculated during runtime).

The selection of rows in `SYSTEM` sampling is dependent on which connector is used. For example, when used with Hive, it is dependent on how the data is distributed in HDFS. This method does not guarantee independent sampling probabilities.

Examples

```
--- Use BERNOULLI sampling
SELECT *
FROM users TABLESAMPLE BERNOULLI ( 50 );

--- Use SYSTEM sampling
SELECT *
FROM users TABLESAMPLE SYSTEM ( 75 );
Using sampling with joins :

--- Use sampling with JOIN
SELECT o.*, i.*
FROM orders o TABLESAMPLE SYSTEM ( 10 )
JOIN lineitem i TABLESAMPLE BERNOULLI ( 40 )
```

```
ON o . orderkey = i . orderkey ;
```

7.3.2.3.27.9 UNNEST

UNNEST expands variables of the **ARRAY** and **MAP** types into a table. Variables of the **ARRAY** type are expanded into a single-column table, and variables of the **MAP** type are expanded into a two-column (key, value) table. **UNNEST** can expand multiple variables of the **ARRAY** and **MAP** types into columns at a time, with as many rows as the maximum number of expanded rows of the input parameter list (the other columns are padded with null values). **UNNEST** may have a **WITH ORDINALITY** clause, in which case an additional ordinal column is appended to the query results. **UNNEST** is typically used with a **JOIN** and can reference columns from relations on the left side of the **JOIN**.

Example 1:

```
--- Use a single column
SELECT student, score
FROM tests
CROSS JOIN UNNEST ( scores ) AS t ( score );
```

Example 2:

```
--- Use multiple columns
SELECT numbers, animals, n, a
FROM (
  VALUES
    ( ARRAY [ 2 , 5 ], ARRAY [ ' dog ', ' cat ', ' bird ' ] ),
    ( ARRAY [ 7 , 8 , 9 ], ARRAY [ ' cow ', ' pig ' ] )
) AS x ( numbers, animals )
CROSS JOIN UNNEST ( numbers, animals ) AS t ( n, a );
```

| numbers | animals | n | a |
|---------------|----------------------|------|------|
| [2 , 5] | [dog , cat , bird] | 2 | dog |
| [2 , 5] | [dog , cat , bird] | 5 | cat |
| [2 , 5] | [dog , cat , bird] | NULL | bird |
| [7 , 8 , 9] | [cow , pig] | 7 | cow |
| [7 , 8 , 9] | [cow , pig] | 8 | pig |
| [7 , 8 , 9] | [cow , pig] | 9 | NULL |

(6 rows)

Example 3:

```
--- Use a WITH ORDINALITY clause
SELECT numbers, n, a
FROM (
  VALUES
    ( ARRAY [ 2 , 5 ] ),
    ( ARRAY [ 7 , 8 , 9 ] )
) AS x ( numbers )
CROSS JOIN UNNEST ( numbers ) WITH ORDINALITY AS t ( n, a );
```

| numbers | n | a |
|---------------|---|---|
| [2 , 5] | 2 | 1 |
| [2 , 5] | 5 | 2 |
| [7 , 8 , 9] | 7 | 1 |
| [7 , 8 , 9] | 8 | 2 |
| [7 , 8 , 9] | 9 | 3 |
| (5 rows) | | |

7.3.2.3.27.10 Joins

Joins merges data from multiple relations. `CROSS JOIN` returns the [Cartesian product](#) of two relations (all combinations). `CROSS JOIN` can be specified by using either of the following two methods:

- Use the explicit `CROSS JOIN` syntax.
- Specify multiple relations in the `FROM` clause.

The following two SQL statements are equivalent:

```
--- Use the explicit `CROSS JOIN` syntax
SELECT *
FROM nation
CROSS JOIN region ;

--- Specify multiple relations in the `FROM` clause
SELECT *
FROM nation , region ;
```

Examples:

The "nation" table contains 25 rows, and the "region" table contains 5 rows, so a `CROSS JOIN` between the two tables produces 125 rows.

```
SELECT n . name AS nation , r . name AS region
FROM nation AS n
CROSS JOIN region AS r
ORDER BY 1 , 2 ;
```

| nation | region |
|-----------|-------------|
| ALGERIA | AFRICA |
| ALGERIA | AMERICA |
| ALGERIA | ASIA |
| ALGERIA | EUROPE |
| ALGERIA | MIDDLE EAST |
| ARGENTINA | AFRICA |
| ARGENTINA | AMERICA |
| ... | |

(125 rows)

When the two tables in a join have columns with the same name, the column references must be qualified by using the table name (or alias).

Examples:

```

--- Correct
SELECT  nation . name ,  region . name
FROM    nation
CROSS   JOIN    region ;

--- Correct
SELECT  n . name ,  r . name
FROM    nation    AS  n
CROSS   JOIN    region    AS  r ;

--- Correct
SELECT  n . name ,  r . name
FROM    nation    n
CROSS   JOIN    region    r ;

--- Incorrect . " Column ' name ' is ambiguous " is thrown .
SELECT  name
FROM    nation
CROSS   JOIN    region ;

```

7.3.2.3.27.11 Subquery

A subquery is an expression that consists a query. The subquery is correlated with external queries when it references columns beyond the subquery. Presto has limited support for correlated subqueries.

EXISTS

The `EXISTS` predicate determines whether a subquery returns all rows. If the subquery returns rows, the `WHERE` expression is `TRUE`. If no rows are returned, the `WHERE` expression is `FALSE`.

Examples:

```

SELECT  name
FROM    nation
WHERE    EXISTS ( SELECT * FROM region WHERE region .
regionkey = nation . regionkey );

```

IN

The `IN` predicate determines whether any columns specified by `WHERE` are included in the result set produced by the subquery. If yes, results are returned. If not, no results are returned. A subquery returns only one column.

Examples:

```

SELECT  name
FROM    nation

```



```
WHERE    regionkey    IN    ( SELECT    regionkey    FROM    region );
```

Scalar subquery

A scalar subquery is a non-correlated subquery that returns zero or one row. The subquery cannot return more than one row. The returned value is `NULL` if the subquery returns no rows.

Examples:

```
SELECT    name
FROM      nation
WHERE     regionkey = ( SELECT    max ( regionkey ) FROM    region )
```

7.3.2.3.28 SET SESSION

Overview

```
SET    SESSION    name = expression
SET    SESSION    catalog . name = expression
```

Description

Sets a session property value.

Examples

```
SET    SESSION    optimize_h ash_genera tion = true ;
SET    SESSION    hive . optimized_ reader_ena bled = true ;
```

7.3.2.3.29 SHOW CATALOGS

Overview

```
SHOW    CATALOGS    [ LIKE    pattern ]
```

Description

Lists the available catalogs. Use the `LIKE` clause to filter catalog names.

Examples

```
SHOW CATALOGS ;
```

7.3.2.3.30 SHOW COLUMNS

Overview

```
SHOW COLUMNS FROM table
```

Description

Lists the columns of a given table and their properties.

Examples

```
SHOW COLUMNS FROM orders ;
```

7.3.2.3.31 SHOW CREATE TABLE

Overview

```
SHOW CREATE TABLE table_name
```

Description

Shows the SQL statement that creates the specified table.

Examples

```
SHOW CREATE TABLE sf1 . orders ;

-----
CREATE TABLE tpch . sf1 . orders (
  orderkey  bigint ,
  orderstatu s  varchar ,
  totalprice double ,
  orderdate  varchar
)
WITH (
  format = ' ORC ',
  partitione d_by = ARRAY [' orderdate ']
)
```

```
( 1 row )
```

7.3.2.3.32 SHOW CREATE VIEW

Overview

```
SHOW CREATE VIEW view_name
```

Description

Shows the SQL statement that creates the specified view.

Examples

```
SHOW CREATE VIEW view1 ;
```

7.3.2.3.33 SHOW FUNCTIONS

Overview

```
SHOW FUNCTIONS
```

Description

Lists all the functions available for use in queries.

Examples

```
SHOW FUNCTIONS
```

7.3.2.3.34 SHOW GRANTS

Overview

```
SHOW GRANTS [ ON [ TABLE ] table_name ]
```

Description

Lists the permissions of the current user on the specified table in the current catalog.

Examples

```
--- List the permission s of the current user on the
    table " orders "
SHOW GRANTS ON TABLE orders ;

--- List the permission s of the current user on the
    current catalog
```

```
SHOW GRANTS ;
```

Limits

Some connectors do not support `SHOW GRANTS` .

7.3.2.3.35 SHOW PARTITIONS

Overview

```
SHOW PARTITIONS FROM table [ WHERE ... ] [ ORDER BY  
... ] [ LIMIT ... ]
```

Description

Lists the partitions of a table. You can use the `WHERE` clause to filter conditions, use `ORDER BY` to sort results, and use `LIMIT` to limit the size of a result set. These clauses are used in the same way as `SELECT` .

Examples

```
--- List all the partitions of the table " orders "  
SHOW PARTITIONS FROM orders ;  
  
--- List all the partitions of the table " orders "  
starting from year 2013 until now and sort the  
partitions by year  
SHOW PARTITIONS FROM orders WHERE ds >= ' 2013 - 01 - 01  
' ORDER BY ds DESC ;  
  
--- List the recent partitions of the table " orders "  
SHOW PARTITIONS FROM orders ORDER BY ds DESC LIMIT  
10 ;
```

7.3.2.3.36 SHOW SCHEMAS

Overview

```
SHOW SCHEMAS [ FROM catalog ] [ LIKE pattern ]
```

Description

Lists all the schemas in the specified catalog or in the current catalog if no catalog is specified. Use the `LIKE` clause to filter schema names.

Examples

```
SHOW SCHEMAS ;
```

7.3.2.3.37 SHOW SESSION

Overview

```
SHOW SESSION
```

Description

Lists the current session properties.

Examples

```
SHOW SESSION
```

7.3.2.3.38 SHOW TABLES

Overview

```
SHOW TABLES [ FROM schema ] [ LIKE pattern ]
```

Description

Lists all the tables in the specified schema or in the current schema if no schema is specified. Use the `LIKE` clause to filter table names.

Examples

```
SHOW TABLES ;
```

7.3.2.3.39 START TRANSACTION

Overview

```
START TRANSACTION [ mode [, ...] ]
```

`mode` provides the following options :

| | | | | | | | | | |
|------------|-------|------|------------|------------|---|--|------|-----------|--|
| ISOLATION | LEVEL | { | READ | UNCOMMITTE | D | | READ | COMMITTED | |
| REPEATABLE | READ | | SERIALIZAB | LE | } | | | | |
| READ | { | ONLY | | WRITE | } | | | | |

Description

Starts a new transaction for the current session.

Examples

```
START TRANSACTION ;
```

```

START TRANSACTION N ISOLATION LEVEL REPEATABLE READ ;
START TRANSACTION N READ WRITE ;
START TRANSACTION N ISOLATION LEVEL READ COMMITTED , READ
ONLY ;
START TRANSACTION N READ WRITE , ISOLATION LEVEL
SERIALIZABLE ;

```

7.3.2.3.40 USE

Overview

```

USE catalog . schema
USE schema

```

Description

Updates the current session to use the specified catalog and schema. The schema in the current catalog is used if no catalog is specified.

Examples

```

USE hive . finance ;
USE information_schema ;

```

7.3.2.3.41 VALUES

Overview

```

VALUES row [, ...]

```

`row` is a single expression or an expression list in the following format:

```

( column_expression [, ...] )

```

Description

Defines a literal inline table.

- `VALUE` can be used anywhere a query statement can be used, such as next to the `FROM` clause of a `SELECT` statement, in an `INSERT` statement, and at the top layer.
- By default, `VALUE` creates an anonymous table without column names. The table and columns can be named by using an `AS` clause.

Examples

```

--- Return a table with one column and three rows
VALUES 1 , 2 , 3

--- Return a table with two columns and three rows
VALUES

```

```
( 1 , ' a '),
( 2 , ' b '),
( 3 , ' c ')

--- Use VALUES in a query statement
SELECT * FROM (
  VALUES
    ( 1 , ' a '),
    ( 2 , ' b '),
    ( 3 , ' c ')
) AS t ( id , name )

--- Create a table
CREATE TABLE example AS
SELECT * FROM (
  VALUES
    ( 1 , ' a '),
    ( 2 , ' b '),
    ( 3 , ' c ')
) AS t ( id , name )
```

7.3.3 Common connectors

7.3.3.1 Kafka connector

Overview

The Kafka connector is used to map topics in Kafka to tables in Presto. Each record in Kafka is mapped to a message in Presto tables.



Notice:

The data that is returned by multiple queries by using Presto may be inconsistent because data in Kafka changes dynamically. Currently, Presto is incapable of processing inconsistent returned data.

Configuration

Create the file `etc / catalog / kafka . properties` , add the following content, and enable the Kafka connector.

```
connector . name = kafka
kafka . table - names = table1 , table2
kafka . nodes = host1 : port , host2 : port
```



Note:

You can connect Presto to multiple Kafka clusters by adding a new properties file in the configuration catalog. The file name is mapped to the Presto catalog. For

example, when the configuration file "orders.properties" is added, Presto creates a catalog named "orders".

```
## orders . properties
connector . name = kafka    # The connector type , which
cannot be changed .
kafka . table - names = tableA , tableB
kafka . nodes = host1 : port , host2 : port
```

The Kafka connector provides the following properties:

| Parameter | Required | Description | Remarks |
|----------------------|----------|--|--|
| kafka.table-names | Yes | A list of tables supported by the connector. | The file name can be modified by using the schema name in the format { schema_name }. { table_name }. If the file name is not modified by using the schema name, the table is mapped to the schema defined in kafka . default - schema . |
| kafka.default-schema | | The default schema name, which is default . | |
| kafka.nodes | Yes | A list of nodes in the Kafka cluster. | The format is hostname : port [, hostname : port ...]. You can configure only part of the Kafka nodes here, but Presto must be connected to all the nodes in the Kafka cluster. Otherwise, a portion of data may not be obtained. |

| Parameter | Required | Description | Remarks |
|-----------------------------|----------|--|--|
| kafka.connect-timeout | No | The timeout period of the connection between the Kafka connector and the Kafka cluster. The default value is 10 seconds. | If the Kafka cluster is under heavy load, it may take a long time to create a connection, causing a timeout when Presto runs a query. In this case, you can increase the value of this parameter. |
| kafka.buffer-size | No | The size of the read buffer. The default value is 64 KB. | This parameter is used to set the size of the internal buffer that stores the data read from Kafka. The buffer size must be greater than that of a message. A data buffer is allocated to each worker and data node, respectively. |
| kafka.table-description-dir | No | The directory that stores the topic (table) description file. The default value is <code>etc / kafka</code> . | The directory stores the data table definition files in the JSON format (the file name must be suffixed with <code>.json</code>). |

| Parameter | Required | Description | Remarks |
|-----------------------------|----------|---|--|
| kafka.hide-internal-columns | No | A list of the predefined columns that need to be hidden. The default value is <code>true</code> . | The Kafka connector maintains many extra columns for each table, in addition to the data columns defined in the table description file. This property controls whether to display the extra columns in the execution results of the <code>DESCRIBE < table - name ></code> and <code>SELECT * statements</code> . These columns are involved in the query process regardless of the setting. |

- The Kafka connector provides the following internal columns:

| Column | Type | Description |
|-------------------|--------|--|
| _partition_id | BIGINT | The ID of the Kafka partition where the current record row is located. |
| _partition_offset | BIGINT | The offset of the current record row in the Kafka partition. |
| _segment_start | BIGINT | The minimum offset of the data segment that contains the current row. This offset is applicable to each partition. |

| Column | Type | Description |
|-------------------------------|---------|---|
| <code>_segment_end</code> | BIGINT | The maximum offset of the data segment that contains the current row (which is the starting offset of the next data segment). This offset is applicable to each partition. |
| <code>_segment_count</code> | BIGINT | The serial number of the current row in the data segment. The calculation formula for an uncompressed topic is as follows: <code>_segment_start + _segment_count = _partition_offset</code> . |
| <code>_message_corrupt</code> | BOOLEAN | This field is set to <code>TRUE</code> if the records in the current row cannot be decoded by using a decoder. |
| <code>_message</code> | VARCHAR | A string encoded with UTF-8 from the message bytes. This field is useful when the topic message is of the text type. |
| <code>_message_length</code> | BIGINT | The byte length of the current message. |
| <code>_key_corrupt</code> | BOOLEAN | This field is set to <code>TRUE</code> if the records in the current row cannot be decoded by using a decoder. |
| <code>_key</code> | VARCHAR | A string encoded with UTF-8 from the key bytes. This field is useful when the topic message is of the text type. |

| Column | Type | Description |
|--------------------------|--------|-------------------------------------|
| <code>_key_length</code> | BIGINT | The byte length of the message key. |

**Note:**

For those tables without definition files, `_key_corrupt` and `_message_corrupt` are set to `FALSE` by default.

Table definition files

Kafka is a schema-less message system. The message format must be defined by the producer and consumer. Presto requires that data be mapped to tables. Therefore, you must provide corresponding table definition files based on the actual usage of messages. Messages in the JSON format can be parsed by using the JSON functions of Presto if no definition files are provided. While the method is flexible, it increases the complexity of writing SQL statements.

When JSON is used to define a table in a table definition file, the file name can be customized, with the invariable extension `.json`.

```
{
  " tableName ": ...,
  " schemaName ": ...,
  " topicName ": ...,
  " key ": {
    " dataFormat ": ...,
    " fields ": [
      ...
    ]
  },
  " message ": {
    " dataFormat ": ...,
    " fields ": [
      ...
    ]
  }
}
```

| Field | Required | Type | Description |
|-------------------------|----------|--------|--|
| <code>tableName</code> | Yes | String | The Presto table name. |
| <code>schemaName</code> | No | String | The name of the schema where the table is located. |

| Field | Required | Type | Description |
|-----------|----------|-------------|--|
| topicName | Yes | String | The Kafka topic name. |
| Key | No | JSON object | The rules for mapping message keys to columns. |
| message | No | JSON object | The rules for mapping messages to columns. |

The mapping rules for keys and messages use the following fields for description:

| Field | Required | Type | Description |
|------------|----------|------------|---|
| dataFormat | Yes | String | A decoder for setting a group of columns. |
| fields | Yes | JSON array | The column definition list. |

`fields` is a JSON array, and each element is a JSON object in the following format:

```
{
  " name ": ...,
  " type ": ...,
  " dataFormat ": ...,
  " mapping ": ...,
  " formatHint ": ...,
  " hidden ": ...,
  " comment ": ...
}
```

| Field | Required | Type | Description |
|------------|----------|--------|-------------------------------|
| name | Yes | String | The column name. |
| type | Yes | String | The data type of this column. |
| dataFormat | No | String | The column data decoder. |
| mapping | No | String | The decoder parameters. |

| Field | Required | Type | Description |
|------------|----------|---------|--|
| formatHint | No | String | The prompt set for the column, which can be used by the decoder. |
| hiddent | No | Boolean | Indicates whether a column is hidden. |
| comment | No | String | The column description. |

Decoder

A decoder maps Kafka messages (key+message) to the columns of data tables. Presto uses the `dummy` decoder when table definition files are unavailable.

The Kafka connector provides the following three decoders:

- `raw` : uses raw bytes directly without conversion.
- `csv` : processes messages as strings in the CSV format.
- `json` : processes messages in the JSON format.

7.3.3.2 JMX connector

Overview

The JMX connector is used to query the JMX information about all the nodes in the Presto cluster. The JMX connector is typically used for system monitoring and debugging. You can modify the connector configuration to perform regular dump of JMX information.

Configuration

Create the file `etc / catalog / jmx . properties` , add the following content, and enable the JMX connector.

```
connector . name = jmx
```

You can add the following content into the configuration file to implement regular dump of JMX data:

```
connector . name = jmx
jmx . dump - tables = java . lang : type = Runtime , com . facebook .
presto . execution . scheduler : name = NodeSchedu ler
jmx . dump - period = 10s
```

```
jmx . max - entries = 86400
```

In the example:

- `dump - tables` is a list of Managed Beans (MBeans) separated with commas (,). This configuration specifies which MBeans are sampled and stored in the memory during each sampling period.
- `dump - period` indicates the sampling period, which is 10s by default.
- `max - entries` indicates the maximum number of historical records, which is 86,400 by default.

If the name of a metric contains a comma (,), it must be escaped by using `\\`, as follows:

```
connector . name = jmx
jmx . dump - tables = com . facebook . presto . memory : type =
memorypool \\, name = general ,\
    com . facebook . presto . memory : type = memorypool \\, name =
system ,\
    com . facebook . presto . memory : type = memorypool \\, name =
reserved
```

Data tables

The JMX connector provides two schemas: `current` and `history`. Specifically:

`current` contains the current MBean of each node. The MBean name is the same as the table name in `current`. If the MBean name contains non-standard characters, the table name must be enclosed by quotation marks during the query. The MBean name can be obtained through the following statement:

```
SHOW TABLES FROM jmx . current ;
```

Examples:

```
--- Obtain the JVM information about each node
SELECT node , vmname , vmversion
FROM jmx . current ." java . lang : type = runtime ";

-----+-----+-----
node | vmname | vmversion
-----+-----+-----
ddc4df17 - xxx | Java HotSpot ( TM ) 64 - Bit Server VM |
24 . 60 - b09
( 1 row )
```

```
--- Obtain the metrics that indicate the maximum number
and minimum number of file descriptor s for each
node
SELECT openfilede scriptorco unt , maxfiledes criptorcou nt
FROM jmx . current ." java . lang : type = operatings ystem ";
```

```

  openfiledes  scriptorcount | maxfiledes  criptorcount
-----+-----
              329 |              10240
( 1 row )

```

`history` contains the data table corresponding to the metrics to be dumped in the configuration file. The following statement queries the data table:

```

SELECT " timestamp ", " uptime " FROM jmx . history ." java .
lang : type = runtime ";

```

```

      timestamp      | uptime
-----+-----
2016 - 01 - 28 10 : 18 : 50 . 000 | 11420
2016 - 01 - 28 10 : 19 : 00 . 000 | 21422
2016 - 01 - 28 10 : 19 : 10 . 000 | 31412
( 3 rows )

```

7.3.3.3 System connector

Overview

The system connector is used to query the basic information and measurements of the Presto cluster through SQL statements.

Configuration

All information can be obtained through a catalog called `system` without configuration.

Examples:

```

--- List all the supported data entries
SHOW SCHEMAS FROM system ;

--- List all the data entries in a project during
runtime
SHOW TABLES FROM system . runtime ;

--- Obtain the node status
SELECT * FROM system . runtime . nodes ;

```

```

      node_id | http_uri | node_version |
-----+-----+-----
3d7e8095 -...| http :// 192 . 168 . 1 . 100 : 9090 | 0 . 188
| false | active
7868d742 -...| http :// 192 . 168 . 1 . 101 : 9090 | 0 . 188
| false | active
7c51b0c1 -...| http :// 192 . 168 . 1 . 102 : 9090 | 0 . 188
| true | active

```

```

--- Cancel a query

```



```
CALL system . runtime . kill_query (' 20151207_2 15727_0014  
6_tx3nr ');
```

Data tables

The system connector provides the following data tables:

| TABLE | SCHEMA | Description |
|-------------------|----------|--|
| catalogs | metadata | This table lists all the catalogs that are supported by the system connector. |
| schema_properties | metadata | This table lists the available properties that can be set when you create a schema. |
| table_properties | metadata | This table lists the available properties that can be set when you create a table. |
| nodes | runtime | This table lists all the visible nodes of the Presto cluster and the node status . |
| queries | runtime | This table contains the information about the queries currently and recently initiated in the Presto cluster, including the raw query text (SQL), the identities of the users who initiate the queries , and information about query performance, such as the query queue and analysis time. |

| TABLE | SCHEMA | Description |
|--------------|---------|--|
| tasks | runtime | This table contains the information about the tasks that are involved in the queries in the Presto cluster, including the location of task execution and the number of lines and bytes processed by each task. |
| transactions | runtime | This table lists the currently opened transactions and their related metadata. The metadata includes the creation time, idle time, initialization parameters, and access catalogs. |

Stored procedure

The system connector supports the following stored procedure:

- `runtime . kill_query (id)`

It cancels the query with the specified ID.

8 TensorFlow

TensorFlow is supported by E-MapReduce 3.13.0 and later. You can add TensorFlow from the available services in your software configurations. If you are using TensorFlow in E-MapReduce to perform high-performance computing, you can allocate CPU and GPU resources through YARN.

Prerequisites

- On the software side, an E-MapReduce cluster installs TensorFlow and a TensorFlow on YARN (TOY) toolkit.
- On the hardware side, E-MapReduce supports computing using both CPU and GPU resources. If you need to use GPU computing, you can choose ECS instances from compute optimized families with GPU, such as gn5 and gn6, for the core and task nodes in the cluster. Compute optimized families with GPU support heterogeneous computing. After determining the instance type, choose the CUDA toolkit and cuDNN versions as required.

Submit TensorFlow jobs

You can log on to the master node in the E-MapReduce cluster to submit TensorFlow jobs using the command line. For example:

```
el_submit [- h ] [- t APP_TYPE ] [- a APP_NAME ] [- m MODE ]
          [- m_arg MODE_ARG ]

[- interact INTERACT ] [- x EXIT ]

[- enable_ten sorboard ENABLE_TEN SORBOARD ]

[- log_tensor board LOG_TENSOR BOARD ] [- conf CONF ] [- f
FILES ]

[- pn PS_NUM ] [- pc PS_CPU ] [- pm PS_MEMORY ] [- wn
WORKER_NUM ]

[- wc WORKER_CPU ] [- wg WORKER_GPU ] [- wm WORKER_MEM ORY ]

[- wnpg WNPG ] [- ppn PPN ] [- c COMMAND [ COMMAND ...]]
```

The basic parameters are described as follows:

- **-t APP_TYPE:** Specifies the type of task to be submitted. The supported types are tensorflow-ps, tensorflow-mpi, and standalone. They are used in conjunction with the following **-m MODE** parameter.
 - **tensorflow-ps:** Uses a parameter server for the communication of data, which is the PS mode of native TensorFlow.
 - **tensorflow-mpi:** Uses Horovod, an open source framework from Uber, which relies on message passing interface (MPI) primitives for the communication of data.
 - **standalone:** Users assign tasks to one instance in the YARN cluster for execution. This is similar to standalone execution.
- **-a APP_NAME:** Specifies the name of the submitted TensorFlow job. You can name jobs as required.
- **-m MODE:** Specifies the runtime environment for submitted TensorFlow jobs. E-MapReduce supports the following environments: local, virtual-env, and docker.
 - **local:** Uses Python runtime environments set up in the E-MapReduce worker nodes. If you want to use third-party Python packages, you need to install the packages on all the nodes manually.
 - **docker:** Uses the Docker containers installed on the E-MapReduce worker nodes. TensorFlow runs in Docker containers.
 - **virtual-env:** Uses isolated Python environments created by users. You can install Python libraries in Python environments. These libraries can be different from those installed in the environments that are set up in the worker nodes.
- **-m_arg MODE_ARG:** Specifies the supplemental parameter for the **-m MODE**. If the runtime environment is docker, set the value to the docker image name. If the runtime environment is virtual-env, set the value to the name of Python environment tar.gz file.
- **-x Exit:** You need to exit the parameter servers manually for certain distributed TensorFlow APIs. To exit parameter servers automatically when worker servers finish training their models, specify the **-x** option.
- **-enable_tensorboard:** Specifies whether to enable TensorBoard when TensorFlow starts training models.
- **-log_tensorboard:** Specifies the location of TensorBoard logs in HDFS. If TensorBoard is enabled when TensorFlow starts training models, this parameter is required.

- **-conf CONF:** Specifies the location of the Hadoop configuration. Setting the value is optional. The default E-MapReduce configuration is used.
- **-f FILES:** Specifies all dependent files and folders for TensorFlow to run, including executable scripts. If virtual-env files that are executed in a virtual environment are specified, you can put all dependencies in one folder. The script then automatically uploads the folders into HDFS according to the folder hierarchy.
- **-pn TensorFlow:** Specifies the number of parameter servers to start.
- **-pc:** Specifies the number of CPU cores that each parameter server requests.
- **-pm:** Specifies the memory size that each parameter server requests.
- **-wn:** Specifies the number of worker nodes started by TensorFlow.
- **-wc:** Specifies the number of CPU cores that each worker requests.
- **-wg:** Specifies the number of GPU cores that each worker requests.
- **-wm:** Specifies the memory usage that each worker requests.
- **-c COMMAND:** Specifies the command to run. For example, `pythoncensus.py`.

Advanced options. We recommend that you use advanced options with care, as they may result in job failures.

- **-wnpg:** Specifies the number of workers that use a GPU simultaneously (for `tensorflow-ps`).
- **-ppn:** Specifies the number of workers that use a GPU simultaneously (for Horovod). The preceding options refer to multitasking on a single GPU. Thresholds should be set to avoid GPU running out of memory.

9 Knox

E-MapReduce supports [Apache Knox](#). If you select a Knox-supported image to create a cluster, you can access the Web UI from the public network to use services such as YARN, HDFS, and SparkHistory.

Preparations

- Enable Knox access using a public IP address
 1. The service port of Knox on E-MapReduce is 8443. In the cluster details, find the ECS security group in which the cluster is located.
 2. Change the corresponding security group in the ECS console and add a rule in Internet inbound to enable port 8443.



Notice:

- For security reasons, the authorization object must be your limited IP address range. 0.0.0.0/0 is forbidden.
- After port 8443 of the security group is enabled, all nodes (including non-E-MapReduce ECS nodes) in the security group enable port 8443 at the ingress of the public network.

- Set a Knox user

Accessing Knox requires a username and password for authentication. The authentication is based on LDAP. You can use your own LDAP service or the LDAP service of Apache Directory Server in the cluster.

- Use the LDAP service in the cluster

Method one(recommended):

Add a Knox account in the [User Management](#) page.

Method two:

1. Log on to the cluster through SSH. For more information, see [Connect to clusters using SSH](#).
2. Prepare your user data. Here, Tom is used as the user name. In the file, replace all `emr - guest` with `Tom` and `cn : EMR GUEST` with `cn : Tom`, and set `userPassword` to your password.

```
su    Knox
cd    /usr/lib/knox-current/templates
vi    users.ldif
```



Notice:

For security reasons, before you export your user data to LDAP, change the password of `users.ldif` by changing `userPassword` to your password.

3. Export to LDAP.

```
su    Knox
cd    /usr/lib/knox-current/templates
sh    ldap-sample-users.sh
```

- Use your own LDAP service

1. Enter the cluster configuration management page. In the cluster-topo configuration, set `main.ldapRealm.userDnTemplate` to your user DN template and `main.ldapRealm.contextFactory.url` to your LDAP server domain name and port. Then, save the settings and restart Knox.
2. Your LDAP service does not typically run in the cluster. You must enable the Knox port to access the LDAP service in the public network, such as port

10389. For more information, see the preceding steps for enabling port 8443. Then, select Internet outbound.



Notice:

For security reasons, the authorization object must be the public IP address of your Knox cluster. 0.0.0.0/0** is forbidden.

Access Knox

- Access using the E-MapReduce shortcut link
 1. Log on to the [E-MapReduce console](#).
 2. Click the ID link of the target cluster.
 3. In the navigation pane on the left, click Clusters and Services.
 4. Click the relevant services on the E-MapReduce services page, such as HDFS and YARN.
 5. In the upper-right corner, click Quick Link.
- Access using the public IP address of the cluster
 1. Check the public IP address in the cluster details.
 2. Access the URLs of the relevant services in the browser.
 - HDFS UI: https://{cluster_access_ip}:8443/gateway/cluster-topo/hdfs/
 - YARN UI: https://{cluster_access_ip}:8443/gateway/cluster-topo/yarn/
 - SparkHistory UI: https://{cluster_access_ip}:8443/gateway/cluster-topo/sparkhistory/
 - Ganglia UI: https://{cluster_access_ip}:8443/gateway/cluster-topo/ganglia/
 - Storm UI: https://{cluster_access_ip}:8443/gateway/cluster-topo/storm/
 - Oozie UI: https://{cluster_access_ip}:8443/gateway/cluster-topo/oozie/
 3. website is not security is displayed in your browser because the Knox service uses a self-signed certificate. Confirm that the accessed IP address is the same as that of your cluster and the port is 8443. Click advance > continue.
 4. Enter the username and password set in LDAP in the logon dialog box.

Access control lists

Knox provides service-level permission management to limit service access to specific users, user groups, or IP addresses. See [Apache Knox Authorization](#).

- **Example**

- **Scenario:** The YARN UI only allows access by user Tom.
- **Steps:** Enter the cluster configuration management page. In the cluster-topo configuration, add access control list (ACL) code between the `< gateway >...</ gateway >` labels.

```
< provider >
  < role > authorization </ role >
  < name > AclsAuthz </ name >
  < enabled > true </ enabled >
  < param >
    < name > YARNUI . acl </ name >
    < value > Tom ;*;* </ value >
  </ param >
</ provider >
```

- **Notes**

Knox provides RESTful APIs for operating a range of services, including adding or deleting HDFS files. For security reasons, make sure that when you enable port 8443 of the security group in the ECS console, the authorization object is your limited IP address range. 0.0.0.0/0 is forbidden. Do not use the LDAP username and password in the Knox installation directory to access Knox.

10 Flume

10.1 Use Flume

This topic takes E-MapReduce-Flume moving audit log to HDFS as an example to describe how to use Flume.

Background

E-MapReduce supports cluster management for EMR-Flume since V3.19.0. By using cluster management, you can configure and manage Flume Agent on the web pages. In the example, Flume Agent is started on the master instance to collect audit log stored in the local disk and sends the log to core instances by using the Avro protocol. Failover Sink Processor is configured and started on the core instances to receive the data from the master instance. Then the data is moved to HDFS by using sinks. For more scenarios of using Flume, see [Configure Flume](#).

Preparations

Create an EMR cluster and select Flume from the optional services. For more information, see [Create a cluster](#).

Procedure

- Configure and start Flume Agent on core instances

For example, configure Flume Agent for the core instance group on the emr-worker-1 node as shown in the following figure.

1. In the Service Configuration section, set the configurations as follows.

| | |
|---|------|
| default-agent.sinks.default-sink.type | hdfs |
| default-agent.channels.default-channel.type | file |
| default-agent.sources.default-source.type | avro |

| | |
|----------------------|--------------|
| deploy_node_hostname | emr-worker-1 |
|----------------------|--------------|

2. Click Custom Configuration and add the following configurations.

| | |
|--|--|
| default-agent.sinks.default-sink.hdfs.path | For high-availability clusters, the hdfs ://emr-cluster/path format is used for the address. |
| default-agent.sinks.default-sink.hdfs.fileType | DataStream |
| default-agent.sinks.default-sink.hdfs.rollSize | 0 |
| default-agent.sinks.default-sink.hdfs.rollCount | 0 |
| default-agent.sinks.default-sink.hdfs.rollInterval | 86400 |
| default-agent.sinks.default-sink.hdfs.batchSize | 51200 |
| default-agent.sources.default-source.bind | 0.0.0.0 |
| default-agent.sources.default-source.port | Set a value as required. |
| default-agent.channels.default-channel.transactionCapacity | 51200 |
| default-agent.channels.default-channel.dataDirs | The path on which channels store the event data. |
| Default-agent.channels.default-channel.checkpointDir | The path on which the checkpoint file is stored. |
| default-agent.channels.default-channel.capacity | Set a value based on the hdfs roll. |

3. Save the configuration and start Flume Agent.
4. Click History. After Successful appears in the Status column, STARTED is displayed for Flume Agent on the emr-worker-1 node in the Status column in the Component Deployment tab page. After Flume Agent on the emr-worker-1 node is started, start Flume Agent on other worker nodes. For example, to start Flume Agent on the emr-worker-2 node, modify the following configuration items.

| | |
|----------------------|---------------------------|
| deploy_node_hostname | The hostname of the node. |
|----------------------|---------------------------|

| | |
|--|--|
| default-agent.sinks.default-sink.hdfs.path | For high-availability clusters, the hdfs ://emr-cluster/path format is used for the address. |
|--|--|

5. Save the configuration, start all components, and select emr-worker-2 as the target node.

- Configure and start Flume Agent on the master instance

For example,

Configure Agent as follows.

| | |
|---|--------------|
| additional_sinks | k1 |
| deploy_node_hostname | emr-header-1 |
| default-agent.sources.default-source.type | taildir |
| default-agent.sinks.default-sink.type | avro |
| default-agent.channels.default-channel.type | file |

The new configurations are as follows.

| Configuration item | Value |
|--|--|
| default-agent.sources.default-source.filegroups | f1 |
| default-agent.sources.default-source.filegroups.f1 | /mnt/disk1/log/hadoop-hdfs/hdfs-audit.log. * |
| default-agent.sources.default-source.positionFile | The path on which the position file is stored. |
| default-agent.channels.default-channel.checkpointDir | The path on which the checkpoint file is stored. |
| default-agent.channels.default-channel.dataDirs | The path on which channels store the event data. |
| default-agent.channels.default-channel.capacity | Set a value as required. |
| default-agent.sources.default-source.batchSize | 2000 |
| default-agent.channels.default-channel.transactionCapacity | 2000 |

| Configuration item | Value |
|--|---|
| default-agent.sources.default-source.ignoreRenameWhenMultiMatching | true |
| default-agent.sinkgroups | g1 |
| default-agent.sinkgroups.g1.sinks | default-sink k1 |
| default-agent.sinkgroups.g1.processor.type | failover |
| default-agent.sinkgroups.g1.processor.priority.default-sink | 10 |
| default-agent.sinkgroups.g1.processor.priority.k1 | 5 |
| default-agent.sinks.default-sink.hostname | The IP address of the emr-worker-1 node. |
| default-agent.sinks.default-sink.port | The port of Flume Agent on the emr-worker-1 node. |
| default-agent.sinks.k1.hostname | The IP address of the emr-worker- node . |
| default-agent.sinks.k1.port | The port of Flume Agent on the emr-worker-2 node. |
| default-agent.sinks.default-sink.batch-size | 2000 |
| default-agent.sinks.k1.batch-size | 2000 |
| default-agent.sinks.k1.type | avro |
| default-agent.sinks.k1.channel | default-channel |

View the result of synchronization

By using the HDFS command, you can see that the data is written to files named `FlumeData.${timestamp}`. "timestamp" shows when the file was created.

10.2 Configure Flume

E-MapReduce supports Apache Flume since V3.16.0. This topic describes how to use Flume to copy data from an EMR Kafka cluster to HDFS, Hive, HBase, and Alibaba Cloud OSS of an EMR Hadoop cluster.

Preparations

- Select Flume from the optional services when you create a Hadoop cluster.
- Create a Kafka cluster and create a topic named flume-test for generating data.



Note:

- If you create a high-security Hadoop cluster that consumes data of a standard Kafka cluster, see [Authentication method compatible with MIT Kerberos](#) for configuring Kerberos authentication in a Hadoop cluster.
- If you create a high-security Kafka cluster that writes data to a standard Hadoop cluster by using Flume, see [Kerberos Kafka source](#).
- If your Hadoop and Kafka clusters are both high-security clusters. See [Cross-region access](#) and the [Cross-region access using Flume](#) section.

Kafka->HDFS

- Configure Flume

Create a configuration file named `flume . properties` and add the following configurations. Set the value of the `a1 . sources . source1 . kafka . bootstrap . servers` configuration item to the hostnames and port numbers of Kafka brokers. `a1 . sources . source1 . kafka . topics` refers to the Kafka topic that Flume consumes. `a1 . sinks . k1 . hdfs . path` refers to the HDFS path to which Flume writes data.

```
a1 . sources = source1
a1 . sinks = k1
a1 . channels = c1

a1 . sources . source1 . type = org . apache . flume . source .
kafka . KafkaSource
a1 . sources . source1 . channels = c1
a1 . sources . source1 . kafka . bootstrap . servers = kafka -
host1 : port1 , kafka - host2 : port2 ...
a1 . sources . source1 . kafka . topics = flume - test
a1 . sources . source1 . kafka . consumer . group . id = flume -
test - group
```

```
# Describe the sink
a1 . sinks . k1 . type = hdfs
a1 . sinks . k1 . hdfs . path = / tmp / flume / test - data
a1 . sinks . k1 . hdfs . fileType = DataStream

# Use a channel which buffers events in memory
a1 . channels . c1 . type = memory
a1 . channels . c1 . capacity = 100
a1 . channels . c1 . transactionCapacity = 100

# Bind the source and sink to the channel
a1 . sources . source1 . channels = c1
a1 . sinks . k1 . channel = c1
```

**Note:**

Assume that you specify the `a1 . sinks . k1 . hdfs . path` configuration item with a URL. Use the `hdfs://emr-cluster` prefix for a high-availability cluster. For example,

```
a1 . sinks . k1 . hdfs . path = hdfs :// emr - cluster / tmp /
flume / test - data
```

Use the `hdfs://emr-header-1:9000` prefix for a standard cluster. For example,

```
a1 . sinks . k1 . hdfs . path = hdfs :// emr - header - 1 : 9000
/ tmp / flume / test - data
```

- **Start Flume**

The default configuration file of Flume is stored in the `/ etc / ecm / flume - conf` path. Use the configuration file to start a Flume agent.

```
flume - ng agent -- name a1 -- conf / etc / ecm / flume -
conf -- conf - file flume . properties
```

By using the `log4j . properties` file in the `/ etc / ecm / flume - conf` path, the `logs / flume . log` log file is generated after the agent is started. You can configure the `log4j . properties` file as needed.

- **Test**

Use the `kafka - console - producer . sh` file on your cluster and enter test data abc.

Flume generates a file named `FlumeData.xxxx` in HDFS with a current timestamp in milliseconds. In the file, you can view the data that you enter on Kafka.

Kafka->Hive

- Create a Hive table

Flume uses transactions to write data to Hive. You need to specify the `transactional` property when creating a Hive table. Take creating table `flume_test` as an example:

```
create table flume_test ( id int , content string )
clustered by ( id ) into 2 buckets
stored as orc TBLPROPERTIES ('transactional'='true');
```

- Configure Flume

Create a configuration file named `flume.properties` and add the following configurations. Set the value of the `a1.sources.source1.kafka.bootstrap.servers` configuration item to the hostnames and port numbers of Kafka brokers. `a1.sinks.k1.hive.metastore` refers to the URI of the Hive metastore. Set the value to the value of the `hive.metastore.uris` configuration item in the `hive-site.xml` file.

```
a1.sources = source1
a1.sinks = k1
a1.channels = c1

a1.sources.source1.type = org.apache.flume.source.kafka.KafkaSource
a1.sources.source1.channels = c1
a1.sources.source1.kafka.bootstrap.servers = kafka-host1:port1,kafka-host2:port2...
a1.sources.source1.kafka.topics = flume-test
a1.sources.source1.kafka.consumer.group.id = flume-test-group

# Describe the sink
a1.sinks.k1.type = hive
a1.sinks.k1.hive.metastore = thrift://xxxx:9083
a1.sinks.k1.hive.database = default
a1.sinks.k1.hive.table = flume_test
a1.sinks.k1.serializer = DELIMITED
a1.sinks.k1.serializer.delimiter = ","
a1.sinks.k1.serializer.serdeSeparator = ','
a1.sinks.k1.serializer.fieldnames = id,content

a1.channels.c1.type = memory
a1.channels.c1.capacity = 100
a1.channels.c1.transactionCapacity = 100

a1.sources.source1.channels = c1
```



```
a1 . sinks . k1 . channel = c1
```

- **Start Flume**

```
flume - ng agent -- name a1 -- conf / etc / ecm / flume -
conf -- conf - file flume . properties
```

- **Generate data**

Use the `kafka - console - producer . sh` file on the Kafka cluster. Enter test data 1 and a that are separated by commas (,).

- **Test data writing**

Perform the following configurations for querying Hive transaction tables.

```
hive . support . concurr enc y - true
hive . exec . dynamic . partition . mode - nonstrict
hive . txn . manager - org . apache . hadoop . hive . ql .
lockmgr . DbTxnManag er
```

Query the data in the `flume_test` table after the configurations are complete.

Kafka->HBase

- **Create an HBase table**

Create HBase table `flume_test` and column column.

- **Configure Flume**

Create a configuration file named `flume . properties` and add the following configurations. Set the value of the `a1 . sources . source1 . kafka .`

`bootstrap . servers` configuration item to the hostnames and port numbers of Kafka brokers. `a1 . sinks . k1 . table` refers to the HBase table name. `a1 . sinks . k1 . columnFami ly` refers to the column name.

```
a1 . sources = source1
a1 . sinks = k1
a1 . channels = c1

a1 . sources . source1 . type = org . apache . flume . source .
kafka . KafkaSourc e
a1 . sources . source1 . channels = c1
a1 . sources . source1 . kafka . bootstrap . servers = kafka -
host1 : port1 , kafka - host2 : port2 ...
a1 . sources . source1 . kafka . topics = flume - test
a1 . sources . source1 . kafka . consumer . group . id = flume -
test - group

a1 . sinks . k1 . type = hbase
a1 . sinks . k1 . table = flume_test
a1 . sinks . k1 . columnFami ly = column
```

```
# Use a channel which buffers events in memory
a1 . channels . c1 . type = memory
a1 . channels . c1 . capacity = 1000
a1 . channels . c1 . transactionCapacity = 100

# Bind the source and sink to the channel
a1 . sources . source1 . channels = c1
a1 . sinks . k1 . channel = c1
```

- Start Flume agent

```
flume -ng agent -- name a1 -- conf / etc / ecm / flume -
conf -- conf - file flume . properties
```

- Test

After data is generated using `kafka - console - producer . sh` in your Kafka cluster, you can query data in HBase.

Kafka->OSS

- Create an OSS path

Create an OSS bucket and the folder such as `oss :// flume - test / result .`

- Configure Flume

Flume writing data to OSS takes up much JVM memory. You can reduce the OSS cache size or increase the Xmx value for Flume agents.

- Modify the OSS cache size

Copy the `hdfs - site . xml` file in the `/ etc / ecm / hadoop - conf` path and paste it in the `/ etc / ecm / flume - conf` path. Reduce the value of `smartdata . cache . buffer . size`. For example, 1048576.

- Modify Xmx

In the Flume configuration path `/ etc / ecm / flume - conf`, copy configuration file `flume - env . sh . template`, paste it to the `/etc/ecm/flume-conf` path, rename it `flume - env . sh`, and set Xmx, for example, to 1G:

```
export JAVA_OPTS ="- Xmx1g "
```

Create a configuration file named `flume . properties` and add the following configurations. Set the value of the `a1 . sources . source1 . kafka .`

bootstrap . servers configuration item to the hostnames and port numbers of Kafka brokers. Set the value of *a1 . sinks . k1 . hdfs . path* to the OSS path.

```
a1 . sources = source1
a1 . sinks = k1
a1 . channels = c1

a1 . sources . source1 . type = org . apache . flume . source .
kafka . KafkaSource
a1 . sources . source1 . channels = c1
a1 . sources . source1 . kafka . bootstrap . servers = kafka -
host1 : port1 , kafka - host2 : port2 ...
a1 . sources . source1 . kafka . topics = flume - test
a1 . sources . source1 . kafka . consumer . group . id = flume -
test - group

a1 . sinks . k1 . type = hdfs
a1 . sinks . k1 . hdfs . path = oss :// flume - test / result
a1 . sinks . k1 . hdfs . fileType = DataStream

# Use a channel which buffers events in memory
a1 . channels . c1 . type = memory
a1 . channels . c1 . capacity = 100
a1 . channels . c1 . transactionCapacity = 100

# Bind the source and sink to the channel
a1 . sources . source1 . channels = c1
a1 . sinks . k1 . channel = c1
```

- **Start Flume**

If you modified the OSS cache size when configuring Flume, use the *classpath* parameter to pass OSS-related dependencies and configurations to Flume:

```
flume - ng agent -- name a1 -- conf / etc / ecm / flume -
conf -- conf - file flume . properties -- classpath "/ opt /
apps / extra - jars /* : / etc / ecm / flume - conf / hdfs - site .
xml "
```

If you modified the Flume agent's Xmx, you only need to pass OSS-related dependencies:

```
flume - ng agent -- name a1 -- conf / etc / ecm / flume -
conf -- conf - file flume . properties -- classpath "/ opt /
apps / extra - jars /*"
```

- **Test**

After the Kafka cluster uses *kafka - console - producer . sh* to generate data, the *FlumeData . xxxx* file is generated with the current timestamp (unit: milliseconds) as the filename suffix in the *oss :// flume - test / result* path.

Kerberos Kafka source

When you consume data of high-security Kafka clusters, you need extra configurations.

- In your Kafka cluster, configure Kerberos authentication and copy the generated keytab file `test . keytab` to the Hadoop cluster path `/ etc / ecm / flume - conf`, and copy the Kafka cluster file `/ etc / ecm / has - conf / krb5 . conf` to the Hadoop cluster path `/ etc / ecm / flume - conf`. For more information, see [Authentication method compatible with MIT Kerberos](#).
- Configure `flume . properties`

Add the following configurations in the `flume . properties` file.

```

a1 . sources . source1 . kafka . consumer . security . protocol =
SASL_PLAIN TEXT
a1 . sources . source1 . kafka . consumer . sasl . mechanism =
GSSAPI
a1 . sources . source1 . kafka . consumer . sasl . kerberos .
service . name = kafka

```

- Configure Kafka clients
 - Create the `flume \ _jaas . conf` file in the `/ etc / ecm / flume - conf` path. Enter the following configurations.

```

KafkaClien t {
    com . sun . security . auth . module . Krb5LoginM odule
    required
    useKeyTab = true
    storeKey = true
    keyTab = "/ etc / ecm / flume - conf / test . keytab "
    serviceNam e = " kafka "
    principal = " test @ EMR .${ realm }. COM ";
};

```

Replace `${ realm }` with the Kerberos realm of the Kafka cluster. Run the `hostname` command on the Kafka cluster and a hostname in the `emr -`

`header - 1 . cluster - xxx` format is returned such as `emr - header - 1 . cluster - 123456` . “123456” is the realm.

- **Modify** `/ etc / ecm / flume - conf / flume - env . sh`

Initially, the `flume - env . sh` file is not in the `/ etc / ecm / flume - conf / path`. You need to copy and paste the `flume - env . sh . template` and rename it `flume - env . sh` . Enter the following configurations.

```
export JAVA_OPTS="$ JAVA_OPTS - Djava . security . krb5 .
conf =/ etc / ecm / flume - conf / krb5 . conf "
export JAVA_OPTS="$ JAVA_OPTS - Djava . security . auth .
login . config =/ etc / ecm / flume - conf / flume_jaas . conf "
```

- **Set the domains**

Add the domain names and IP addresses of the nodes in the Kafka cluster to the `/ etc / hosts` file on the Hadoop cluster. An example of long domains is `emr - header - 1 . cluster - 123456` .

Use Flume with cross-region access

After configuring cross-region access, perform the following steps.

- On your Kafka cluster, configure Kerberos authentication and copy the generated keytab file `test . keytab` to the Hadoop cluster path `/ etc / ecm / flume - conf` . For more information, see [Authentication method compatible with MIT Kerberos](#).
- Configure `flume . properties`

Add the following configurations in the `flume . properties` file.

```
a1 . sources . source1 . kafka . consumer . security . protocol =
SASL_PLAIN TEXT
a1 . sources . source1 . kafka . consumer . sasl . mechanism =
GSSAPI
a1 . sources . source1 . kafka . consumer . sasl . kerberos .
service . name = kafka
```

- **Configure Kafka clients**

- Create the `flume \ _jaas . conf` file in the `/ etc / ecm / flume - conf` path. Enter the following configurations.

```
KafkaClient {
    com . sun . security . auth . module . Krb5LoginModule
    required
    useKeyTab = true
    storeKey = true
```

```
keyTab = "/ etc / ecm / flume - conf / test . keytab "
serviceNam e = " kafka "
principal = " test @ EMR . ${ realm } . COM ";
};
```

Replace `${ realm }` with the Kerberos realm of the Kafka cluster. Run the

`hostname` command on the Kafka cluster and a hostname in the `emr - header - 1 . cluster - xxx` format is returned such as `emr - header - 1 . cluster - 123456`. “123456” is the realm.

- **Modify** `/ etc / ecm / flume - conf / flume - env . sh`

Initially, the `flume - env . sh` file is not in the `/ etc / ecm / flume -`

`conf / path`. You need to copy and paste the `flume - env . sh . template` and rename it `flume - env . sh`. Enter the following configurations.

```
export JAVA_OPTS ="$ JAVA_OPTS - Djava . security . auth .
login . config = / etc / ecm / flume - conf / flume_jaas . conf ""
```

10.3 Use LogHub Source to move data from non-EMR clusters to HDFS of EMR clusters

This topic describes how to use EMR-Flume to move data in Log Service to HDFS of an EMR cluster and store the data in partitions based on record timestamps.

Background

EMR features EMR-Flume with Log Service Source since V3.20.0. By using tools of Log Service such as Logtail, you can move data to LogHub and use EMR-Flume to move the data to HDFS of an EMR cluster. For more information, see [Collection methods](#).

Preparations

Create a Hadoop cluster and select Flume from optional services. For more information, see [Create a cluster](#).

Configure Flume

- **Configure Source**

| Configuration item | Value | Description |
|--------------------|---|-------------|
| type | org.apache.flume.source.loghub.LogHubSource | |

| Configuration item | Value | Description |
|--------------------|------------------------|---|
| endpoint | The endpoint of LogHub | If you use a VPC or classic network endpoint, make sure that the VPC or classic network is deployed in the same region as the EMR cluster . If you use a public network endpoint, make sure that the node on which the Flume agent runs is assigned with a public IP address. |
| project | The project of LogHub | |
| logstore | The logstore of LogHub | |
| accessKeyId | The AccessKey ID | |
| accessKey | The AccessKey Secret | |
| useRecordTime | true | Default value: false. If the timestamp property is not in the header, the time when events are received is encoded as timestamps , which are inserted into the header. When Flume Agent starts or stops or data synchronization is delayed, the data is placed in the wrong partitions . Set the value to true. A true value indicates using the time when LogHub collects the data as the timestamp. |

| Configuration item | Value | Description |
|--------------------|------------|--|
| consumerGroup | consumer_1 | The name of the consumer group. Default value: consumer_1. |

The other configuration items are described as follows.

- consumerPosition

The position where the consumer group consumes the LogHub data for the first time. Default value: end (indicates consuming the latest data). Valid values: begin, special, and end. "begin" indicates that the consumer group starts consuming from the earliest data. "special" indicates that the consumer group starts data consuming at a specified offset. When the value is set to special, you need to specify the offset by using the startTime configuration item. Unit: seconds. The LogHub server records the consumer position of the consumer group after first data consumption. To modify the consumerPosition value, clear the status of the consumer group that consumes LogHub data. For more information, see [Status of a consumer group](#). You can also modify the value of consumerGroup to assign another consumer group.

- heartbeatInterval and fetchInOrder

"heartbeatInterval" indicates the interval at which the consumer group sends heartbeats to the server. Unit: milliseconds. Default value: 30000. "fetchInOrder" indicates whether the consumer group consumes data with the same key in sequence. Default value: false.

- batchSize and batchDurationMillis

Common configuration items for source batch. Indicate the thresholds that trigger the events to be written to the channel.

- backoffSleepIncrement and maxBackoffSleep

Common configuration items for source sleep. Indicate the increment for time delay and the maximum time delay before retrieving LogHub data again when no data is found in LogHub.

- Configure the channel and sink

In this example, the memory channel and the HDFS sink are used.

- Configure the HDFS sink as follows.

| Configuration item | Value |
|--------------------|--|
| hdfs.path | /tmp/flume-data/loghub/datetime=%y
%m%d/hour=%H |
| hdfs.fileType | DataStream |
| hdfs.rollInterval | 3600 |
| hdfs.round | true |
| hdfs.roundValue | 60 |
| hdfs.roundUnit | minute |
| hdfs.rollSize | 0 |
| hdfs.rollCount | 0 |

- Configure the memory channel as follows.

| Configuration item | Value |
|---------------------|-------|
| capacity | 2000 |
| transactionCapacity | 2000 |

Run Flume Agent

For more information, see [Use Flume](#). After Flume Agent is started, on the configured HDFS path, you can see the logs that are stored in the partitions based on the record timestamps.

```
[root@emr-worker-3 ~]# hdfs dfs -ls /tmp/flume-data/loghub/datetime=190430/hour=11
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/apps/ecm/service/hadoop/2.7.2-1.2.8/package/hadoop-2.7.2-1.2.8/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/apps/ecm/service/tez/0.8.4/package/tez-0.8.4/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 1 items
-rw-r--r-- 3 root hadoop      1344 2019-04-30 11:18 /tmp/flume-data/loghub/datetime=190430/hour=11/FlumeData.1556594288836.tmp
[root@emr-worker-3 ~]#
```

For information about the status of consumer groups on Log Service, see [View consumer group status](#)

11 Sqoop

Sqoop is an open-source application that is used to transfer data between different data stores. It supports various data stores.

Install Sqoop



Notice:

Sqoop has been integrated with E-MapReduce since E-MapReduce 1.3. If you are using E-MapReduce 1.3 or later, you can skip this section.

If the version you are using is earlier than E-MapReduce 1.3, you can install Sqoop as follows:

1. Download Sqoop 1.4.6 from the official site ([Click to download](#)). If you cannot open the `sqoop - 1 . 4 . 6 . bin__hadoo p - 2 . 0 . 4 - alpha . tar . gz` file that you downloaded, try to download the file from the mirror site `http :// mirror . bit . edu . cn / apache / sqoop / 1 . 4 . 6 / sqoop - 1 . 4 . 6 . bin__hadoo p - 2 . 0 . 4 - alpha . tar . gz` by executing the following command.

```
wget http :// mirror . bit . edu . cn / apache / sqoop / 1 . 4 . 6 / sqoop - 1 . 4 . 6 . bin__ha doop - 2 . 0 . 4 - alpha . tar . gz
```

2. Execute the following command to extract the `sqoop - 1 . 4 . 6 . bin__hadoo p - 2 . 0 . 4 - alpha . tar . gz` file to the Master node.

```
tar zxf sqoop - 1 . 4 . 6 . bin__hadoo p - 2 . 0 . 4 - alpha . tar . gz
```

3. Install the MySQL driver to import data from MySQL. Download the latest version from the official site ([Click to download](#)). In addition, you can execute the following command to download the latest version (take version 5.1.38 as an example).

```
wget https :// dev . mysql . com / get / Downloads / Connector - J / mysql - connector - java - 5 . 1 . 38 . tar . gz
```

4. Extract the jar file to the lib folder in the Sqoop folder.

Transfer data

Scenarios:

- MySQL -> HDFS
- HDFS -> MySQL
- MySQL -> Hive
- Hive -> MySQL
- Free-form query imports

**Notice:**

You must switch your user account to `hadoop` before executing commands in later sections.

```
su    hadoop
```

- Import data from MySQL into HDFS

Execute the following command on the Master node of the cluster:

```
sqoop  import  -- connect  jdbc : mysql ://< dburi >/< dbname  
> -- username  < username > -- password  < password > -- table  <  
tablename > -- check - column  < col > -- incrementa l  < mode >  
-- last - value  < value > -- target - dir  < hdfs - dir >
```

Parameters:

- **dburi**: the connection string of a database such as `jdbc : mysql :// 192 . 168 . 1 . 124 : 3306 /`. When a connection string includes any parameter, you must enclose the connection string within single quotation marks (') such

```
as jdbc : mysql :// 192 . 168 . 1 . 124 : 3306 / mydatabase ?
useUnicode = true
```

- **dbname:** the name of a database such as user.
- **username:** the username that is used to log on to a database.
- **password:** the password that username with the username.
- **tablename:** the name of a MySQL table.
- **col:** the name of a column to be queried.
- **mode:** specifies how Sqoop determines which rows are new. Valid values: append and lastmodified.
- **value :** specifies the maximum value of a column to be checked from the previous import.
- **hdfs-dir:** the HDFS directory that you import data into such as/ *user / hive / result* .

For more information about parameters, see [Sqoop import](#).

· Import data from HDFS into MySQL

You must create MySQL tables that comply with the data structure of HDFS in advance. Then you can execute the following command on the Master node of a cluster to specify a directory that you import data into.

```
sqoop export -- connect jdbc : mysql ://< dburi >/< dbname
> -- username < username > -- password < password > -- table <
tablename > -- export - dir < hdfs - dir >
```

Parameters:

- **dburi:** the connection string of a database such as *jdbc : mysql :// 192 . 168 . 1 . 124 : 3306 /*. When any parameter is included in a connection string, enclose the connection string within single quotation marks (') such

```
as jdbc : mysql :// 192 . 168 . 1 . 124 : 3306 / mydatabase ?
useUnicode = true
```

- **dbname:** the name of a database, such as user.
- **username:** the username that is used to log on to a database.
- **password:** the password that is associated with the username.
- **tablename:** the name of a MySQL table.
- **hdfs-dir:** the directory of HDFS from which you import data into MySQL such as
/ user / hive / result .

For more information about parameters, see [Sqoop export](#)

- Import data from MySQL into Hive

When you execute the following command on the Master node of a cluster to import data from MySQL, a Hive table will be created as follows.

```
sqoop import -- connect jdbc : mysql ://< dburi >/< dbname
> -- username < username > -- password < password > -- table <
tablename > -- check - column < col > -- incrementa l < mode >
-- last - value < value > -- fields - terminated - by "\ t " --
lines - terminated - by "\ n " -- hive - import -- target - dir
< hdfs - dir > -- hive - table < hive - tablename >
```

Parameters:

- **dburi:** the connection string of a database such as `jdbc : mysql :// 192 . 168 . 1 . 124 : 3306 /`. When a connection string includes any parameter, you must enclose the connection string within single quotation marks (') such

```
as jdbc : mysql :// 192 . 168 . 1 . 124 : 3306 / mydatabase ?
useUnicode = true
```

- **dbname:** the name of a database such as user.
- **username:** the username that is used to log on to a database.
- **password:** the password that is associated with the username.
- **tablename:** the name of a MySQL table.
- **col:** the name of a column to be queried.
- **mode:** specifies how Sqoop determine which rows are new. Valid values: append and lastmodified. When you import data into Hive by Sqoop, you cannot use the append mode.
- **value:** specifies the maximum value of a column to be queried from the previous import.
- **hdfs-dir:** the directory of HDFS from which you import data into MySQL such as / user / hive / result .
- **hive-tablename:** the table name of Hive such as xxx.yyy.

For more information about how to use parameters, see [Sqoop import](#).

• Import data from Hive into MySQL

You can refer to the previous command that is used to import data from HDFS into MySQL. In addition, you must specify the HDFS directory of Hive tables from which you import data into MySQL.

• Import data from MySQL into OSS

The process is similar to importing data from MySQL into HDFS, except for the configuration of the target-dir parameter. You can execute the following command on the Master node of a cluster:

```
sqoop import -- connect jdbc : mysql ://< dburi >/< dbname
> -- username < username > -- password < password > -- table
< tablename > -- check - column < col > -- incremental < mode
> -- last - value < value > -- target - dir < oss - dir > --
temporary - rootdir < oss - tmpdir >
```



Notice:

- The endpoint of an OSS host can be: intranet endpoint, Internet endpoint, or VPC endpoint. For a classic network, you must specify the intranet endpoint. For example, the OSS intranet endpoint of the China (Hangzhou) region is oss

- `cn - hangzhou - internal . aliyuncs . com` . For a VPC, you must specify the VPC endpoint. For example, the OSS VPC endpoint of the China (Hangzhou) region is `vpc100 - oss - cn - hangzhou . aliyuncs . com` .
- Currently, when you import data into OSS, you cannot specify the `delete - target - dir` parameter. Otherwise, the error message Wrong FS occurs. When you want to overwrite a directory, you can execute the `hadoop fs - rm -r osspath` command to remove this OSS directory before using Sqoop.

```
sqoop import --connect jdbc:mysql://<dburi>/<dbname>
--username <username> --password <password> --table
<tablename> --check-column <col> --incremental <mode>
--last-value <value> --target-dir <oss-dir> --
temporary-rootdir <oss-tmpdir>
```

Parameters:

- **dburi**: the connection string of a database such as `jdbc:mysql://192.168.1.124:3306/` . When a connection string includes any parameter, enclose the connection string within single quotation marks (') such as `jdbc:mysql://192.168.1.124:3306/mydatabase?useUnicode=true`
- **dbname**: the name of a database such as `user`.
- **username**: the username that is used to log on to a database.
- **password**: the password that is associated with the username.
- **tablename**: the name of a MySQL table.
- **col**: the name of a column to be queried.
- **mode**: used by Sqoop to determine which rows are new rows. Valid values: `append` and `lastmodified`.
- **value**: specifies the maximum value of a column to be queried from the previous import.
- **oss-dir**: the OSS directory that you import data into `oss://<accessid>:<accesskey>@<bucketname>.oss-cn-hangzhou-internal.aliyuncs.com/result` .
- **oss-tmpdir**: the temporary target folder. You must specify this parameter when you specify the `append` mode. If the destination directory already exists in HDFS, Sqoop will stop to import and overwrite that directory' s contents. If you

specify the append mode, Sqoop will import data to a temporary directory and then rename the files to the normal target directory in a manner that does not conflict with the existing filenames in that directory.

For more information about available parameters, see [Sqoop import](#).

- Import data from OSS into MySQL

The process is similar to importing data from MySQL to HDFS, except for the configuration of the export-dir parameter. You must create MySQL tables that comply with the data structure of OSS in advance.

Then you can execute the following command on the Master node of a cluster to specify the directory from which you want to import data.

```
sqoop export --connect jdbc:mysql://<dburi>/<dbname>
--username <username> --password <password> --table <
tablename> --export-dir <oss-dir>
```

Parameters:

- **dburi:** the connection string of a database such as `jdbc:mysql://192.168.1.124:3306/`. When a connection string includes any parameter, you must enclose this connection string within single quotation marks (') such as `jdbc:mysql://192.168.1.124:3306/mydatabase?useUnicode=true`.
- **dbname:** the name of a database such as `user`.
- **username:** the username that is used to log on to a database.
- **password:** the password that is associated with the username.
- **tablename:** the name of a MySQL table.
- **oss-dir:** the OSS directory that you import data into such as `oss://<accessid>:<accesskey>@<bucketname>.oss-cn-hangzhou-internal.aliyuncs.com/result`.
- **oss-tmpdir:** the temporary directory that you import data into. You must specify this parameter when you specify the append mode. If the destination directory already exists in HDFS, Sqoop will stop to import and overwrite that directory's contents. If you specify the append mode, Sqoop will import data to a

temporary directory and then rename the files into the normal target directory in a manner that does not conflict with existing filenames in that directory.



Note:

The endpoint of an OSS host can be: intranet endpoint, Internet endpoint, or VPC endpoint. For a classic network, you must specify an intranet endpoint. For example, the OSS intranet endpoint of the China (Hangzhou) region is `oss - cn - hangzhou - internal . aliyuncs . com`. For a VPC, you must specify a VPC endpoint. For example, the OSS VPC endpoint of the China (Hangzhou) region is `vpc100 - oss - cn - hangzhou . aliyuncs . com`.

For more information about available parameters, see [Sqoop export](#).

- Free-form query imports

In addition to importing a set of MySQL tables, you can also import the result set of an arbitrary SQL query as follows:

```
sqoop import -- connect jdbc:mysql://<dburi>/<dbname>
-- username <username> -- password <password> -- query <
query - sql > -- split - by <sp - column> -- hive - import --
hive - table <hive - tablename> -- target - dir <hdfs - dir>
```

Parameters:

- **dburi**: the connection string of a database, such as `jdbc:mysql://192.168.1.124:3306/`. When a connection string includes any parameter, you must enclose this connection string within single quotation marks (') such

```
as jdbc : mysql :// 192 . 168 . 1 . 124 : 3306 / mydatabase ?  
useUnicode = true
```

- **dbname:** the name of a database such as user.
- **username:** the username that is used to log on to a database.
- **password:** the password that is associated with the username.
- **query-sql:** the query statement such as `SELECT * FROM profile WHERE id > 1 AND \[CONDITIONS]`. You must enclose the query statement that ends with `AND \[CONDITIONS]` within double quotation marks (").
- **sp-column:** specifies the name of a column to be split. In general, the value of this parameter is the primary key of the MySQL table.
- **hdfs-dir:** the directory of HDFS from which you import data into MySQL such as `/ user / hive / result .`
- **hive-tablename:** the name of a table that is used to import data to Hive such as `xxx.yyy`.

For more information about available parameters, see [Sqoop query import](#).

12 Component authorization

12.1 HDFS authorization

After HDFS has been enabled, you need permission to access it in order to perform operations, such as read data or create folders.

Add a configuration

The configurations related to HDFS permission are as follows:

- `dfs.permissions.enabled`

Enable permission check. Even if the value is false, `chmod/chgrp/chown/setfacl` performs a permission check.

- `dfs.datanode.data.dir.perm`

The permission of the local folder used by datanode, which is 755 by default.

- `fs.permissions.umask-mode`

- Permission mask (default permission settings when creating a new file/folder)
- File creation: $0666 \& \text{^umask}$
- Folder creation: $0777 \& \text{^umask}$
- Default umask value is 022. This means that the permission for file creation is 644 ($666 \& \text{^}022 = 644$), and permission of folder creation is 755 ($777 \& \text{^}022 = 755$).
- The default setting of the Kerberos security cluster in E-MapReduce is 027. The permission for file creation is 640. For folder creation, it is 750.

- `dfs.namenode.acls.enabled`

- Enable ACL control. This gives you permission control for owners/groups, and you can also set it for other users.
- Commands for setting ACL:

```
hadoop fs - getfacl [- R ] < path >
hadoop fs - setfacl [- R ] [- b ] [- k - m ] [- x ] < acl_spec
> < path > ] [-- set < acl_spec > < path >]
```

For example:

```
su test
# The user test creates a folder
hadoop fs - mkdir / tmp / test
# View the permission of the created folder
```

```

hadoop fs -ls / tmp
drwxr - x --- - test      hadoop      0    2017 - 11 - 26
21 : 18 / tmp / test
# Set ACL and grant rwx permission s to user
foo
hadoop fs - setfacl - m user : foo : rwx / tmp / test
# View the permission of the file (+ means that
ACL is set )
hadoop fs -ls / tmp /
drwxrwx ---+ - test      hadoop      0    2017 - 11 - 26
21 : 18 / tmp / test
# View ACL
hadoop fs - getfacl / tmp / test
# file : / tmp / test
# owner : test
# group : hadoop
user :: rwx
user : foo : rwx
group :: r - x
mask :: rwx
other :: ---

```

- `dfs.permissions.superusergroup`

Super user group. Users in this group have super user permissions.

Restart the HDFS service

For Kerberos security clusters, HDFS permissions have been set by default (umask is set to 027). Configuration and service restart are not necessary.

For non-Kerberos security clusters, a configuration must be added and the service must be restarted.

Other

- The umask value can be modified as required.
- HDFS is a basic service, and Hive/HBase are based on HDFS. Therefore, HDFS permission control must be configured in advance when configuring other upper-layer services.
- When permissions are enabled for HDFS, the services must be set up (such as / spark-history for Spark and /tmp/\$user/ for YARN).
- Sticky bit:

Sticky bit can be set for a folder to prevent anyone other than super user/file owner /directory owner from deleting files or folders in the folder (even if other users have rwx permissions for that folder). For example:

```

# That is , adding numeral 1 as the first digit
hadoop fs - chmod 1777 / tmp
hadoop fs - chmod 1777 / spark - history

```

```
hadoop fs - chmod 1777 / user / hive / warehouse
```

12.2 YARN authorization

YARN authorization can be divided into service-level and queue-level authorization.

Service-level authorization

For more information, see Hadoop's [Service Level Authorization Guide](#).

- Controls users' access to cluster services, such as job submission.
- Configures `hadoop-policy.xml`.
- Service-level permission checks are performed before other permission checks (such as for HDFS permission and YARN job submission)



Note:

Typically, if HDFS permission checks and YARN job submission controls have been set up, you may choose not to set the service-level permission control. Perform the relevant configurations as required.

Queue-level authorization

YARN supports permission control for resources by means of queues, and provides two queue scheduling methods: Capacity Scheduler and Fair Scheduler. Capacity Scheduler is used as an example here.

- Add a configuration

A queue also has two levels of authorization: for job submission (submitting a job to the queue) and for queue management.



Note:

- The ACL control object for a queue is a user or group. When you are defining the related parameters, users and groups can be set at the same time, separated by spaces. You can use a comma to separate different users or groups. Only one space indicates that no one has permission.
- ACL inheritance for a queue: If a user or group can submit an application to a queue, they can also submit applications to all of its sub-queues. The ACL that manages queues can also be inherited. If you want to prevent a user or group

from submitting jobs to a queue, you must set the ACL for this queue and all its parent queues to restrict the job submission permission for this user or group.

- `yarn.acl.enable`

Set the ACL switch to true.

- `yarn.admin.acl`

■ The YARN administrator setting, which enables or disables the execution of `yarn radmin / yarn kill` and other commands. This value must be configured. If not, the subsequent queue-based ACL administrator settings do not take effect.

■ You can set the user or group when setting the parameter values:

```
user1 , user2   group1 , group2 # users   and   groups   are
    separated by a space
    group1 , group2 # In case there are only groups
    , a leading space is required .
```

In an E-MapReduce cluster, you must configure the ACL permission for `has` as administrator.

- `yarn.scheduler.capacity.${queue-name}.acl_submit_applications`

■ Set the user or group that can submit jobs to this queue.

■ If `${queue-name}` is the queue name, multi-level queues are supported. Note that ACL is inherited in multi-level queues. For example:

```
# queue - name = root
< property >
  < name > yarn . scheduler . capacity . root . acl_submit
  _applicati ons </ name >
  < value > </ value > # Space means no one can
  submit jobs to the root queue
</ property >
# queue - name = root . testqueue
< property >
  < name > yarn . scheduler . capacity . root . testqueue .
  acl_submit _applicati ons </ name >
  < value > test testgrp </ value > # testqueue only
  allows the test user and testgrp group to
  submit jobs
```

```
</ property >
```

- yarn.scheduler.capacity.\${queue-name}.acl_administer_queue

- Set some users or groups to manage the queue, such as killing jobs in the queue.
- Multi-level queue names are supported. Note that ACL is inherited in multi-level queues.

```
# queue - name = root
< property >
  < name > yarn . scheduler . capacity . root . acl_admini
ster_queue </ name >
  < value > </ value >
</ property >
# queue - name = root . testqueue
< property >
  < name > yarn . scheduler . capacity . root . testqueue .
acl_admini ster_queue </ name >
  < value > test testgrp </ value >
</ property >
```

- Restart the YARN service
 - Kerberos secure clusters have ACL enabled by default. You can configure the relevant ACL permissions for queues as required.
 - For non-Kerberos secure clusters, enable ACL and configure the permission control for queues in accordance with the preceding instructions. Then restart the YARN service.

- Configuration example

- yarn-site.xml

```
< property >
  < name > yarn . acl . enable </ name >
  < value > true </ value >
</ property >
< property >
  < name > yarn . admin . acl </ name >
  < value > has </ value >
</ property >
```

- capacity-scheduler.xml
- Default queue: Disables the default queue and does not allow users to submit jobs or manage the queue.
- Q1 queue: Only allows the test user to submit jobs and manage the queue (such as killing jobs).
- Q2 queue: Only allows the foo user to submit jobs and manage the queue.

```
< configurat ion >
```

```

    < property >
      < name > yarn . scheduler . capacity . maximum - applicatio
ns </ name >
      < value > 10000 </ value >
      < descriptio n > Maximum number of applicatio ns
that can be pending and running .</ descriptio n >
    </ property >
    < property >
      < name > yarn . scheduler . capacity . maximum - am -
resource - percent </ name >
      < value > 0 . 25 </ value >
      < descriptio n > Maximum percent of resources in
the cluster which can be used to run applicatio n
masters i . e .
controls number of concurrent running
applicatio ns .
    </ descriptio n >
    </ property >
    < property >
      < name > yarn . scheduler . capacity . resource - calculator
</ name >
      < value > org . apache . hadoop . yarn . util . resource .
DefaultRes ourceCalcu lator </ value >
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . queues </ name
>
      < value > default , q1 , q2 </ value >
      <!-- 3 queues -->
      < descriptio n > The queues at the this level (
root is the root queue ).</ descriptio n >
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . default .
capacity </ name >
      < value > 0 </ value >
      < descriptio n > Default queue target capacity .</
descriptio n >
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . default . user
- limit - factor </ name >
      < value > 1 </ value >
      < descriptio n > Default queue user limit a
percentage from 0 . 0 to 1 . 0 .</ descriptio n >
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . default .
maximum - capacity </ name >
      < value > 100 </ value >
      < descriptio n > The maximum capacity of the
default queue .</ descriptio n >
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . default .
state </ name >
      < value > STOPPED </ value >
      <!-- Status of the default queue is set as
STOPPED -->
      < descriptio n > The state of the default
queue . State can be one of RUNNING or STOPPED .</
descriptio n >
    </ property >
  </ property >

```



```

    < name > yarn . scheduler . capacity . root . default .
acl_submit_applications </ name >
    < value > </ value >
    <!-- The default queue does not allow job
submission -->
    < description > The ACL of who can submit jobs
to the default queue .</ description >
    </ property >
    < property >
    < name > yarn . scheduler . capacity . root . default .
acl_administer_queue </ name >
    < value > </ value >
    <!-- Prevent users / groups to manage the
default queue -->
    < description > The ACL of who can administer
jobs on the default queue .</ description >
    </ property >
    < property >
    < name > yarn . scheduler . capacity . node - locality -
delay </ name >
    < value > 40 </ value >
    </ property >
    < property >
    < name > yarn . scheduler . capacity . queue - mappings </
name >
    < value > u : test : q1 , u : foo : q2 </ value >
    <!-- Queue mapping , automatica lly maps the
test user to Q1 queue -->
    < description > A list of mappings that will
be used to assign jobs to queues . The syntax for
this list is
    [ u | g ]:[ name ]:[ queue_name ][ , next mapping ]*
Typically this list will be used to map users to
queues , for
    example , u :% user :% user maps all users to
queues with the same name as the user .
    </ description >
    </ property >
    < property >
    < name > yarn . scheduler . capacity . queue - mappings -
override . enable </ name >
    < value > true </ value >
    <!-- Whether or not allow the above queue -
mapping to overwrite the queue parameters set up by
the client -->
    < description > If a queue mapping is present ,
will it override the value specified by the user ?
This can be used
    by administra tors to place jobs in queues
that are different than the one specified by the
user . The default
    is false .
    </ description >
    </ property >
    < property >
    < name > yarn . scheduler . capacity . root . acl_submit
_applications </ name >
    < value > </ value >
    <!-- ACL inheritanc e , the parent queue must
have the admin permission s -->
    < description >
    The ACL of who can submit jobs to the
root queue .
    </ description >

```

```

    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . q1 .
acl_submit _applicati ons </ name >
      < value > test </ value >
      <!-- q1 only allows the test user to submit
jobs -->
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . q2 .
acl_submit _applicati ons </ name >
      < value > foo </ value >
      <!-- q2 only allows the foo user to submit
jobs -->
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . q1 . maximum -
capacity </ name >
      < value > 100 </ value >
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . q2 . maximum -
capacity </ name >
      < value > 100 </ value >
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . q1 . capacity
</ name >
      < value > 50 </ value >
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . q2 . capacity
</ name >
      < value > 50 </ value >
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . acl_admini
ster_queue </ name >
      < value > </ value >
      <!-- ACL inheritanc e , the parent queue must
have the admin permission s -->
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . q1 .
acl_admini ster_queue </ name >
      < value > test </ value >
      <!-- q1 only allow the test user to manage
the queue , such as killing the jobs -->
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . q2 .
acl_admini ster_queue </ name >
      < value > foo </ value >
      <!-- q2 only allow the foo user to manage
the queue , such as killing the jobs -->
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . q1 . state </
name >
      < value > RUNNING </ value >
    </ property >
    < property >
      < name > yarn . scheduler . capacity . root . q2 . state </
name >

```

```
< value > RUNNING </ value >
</ property >
</ configurat ion >
```

12.3 Hive authorization

Hive has two authorization modes: one based on storage and the other based on SQL standards. For more information, see Hive's [Authorization guide](#).



Note:

Both means of authorization can be configured at the same time without conflict.

Storage based authorization (for Hive metastore)

If a user in a cluster has direct access to data in Hive through an HDFS or Hive client, a permission control must be performed on Hive data in HDFS. By doing so, operation permissions related to Hive SQL can be controlled.

For more information, see Hive's [Storage Based Authorization](#) guide.

Add configuration

In the cluster Configuration Management page, click Hive > Configuration > hive-site.xml > Add Custom Configuration.

```
< property >
< name > hive . metastore . pre . event . listeners </ name >
  < value > org . apache . hadoop . hive . ql . security .
    authorizat ion . Authorizat ionPreEven tListener </ value >
</ property >
< property >
< name > hive . security . metastore . authorizat ion . manager </
  name >
  < value > org . apache . hadoop . hive . ql . security .
    authorizat ion . StorageBas edAuthoriz ationProvi der </ value >
</ property >
< property >
< name > hive . security . metastore . authentica tor . manager </
  name >
  < value > org . apache . hadoop . hive . ql . security .
    HadoopDefa ultMetasto reAuthenti cator </ value >
</ property >
```

Restart Hive metastore

Restart the Hive metastore in the cluster's Configuration Management page.

HDFS permission control

For Kerberos security clusters in E-MapReduce, HDFS permissions for the Hive warehouse are set.

For non-Kerberos security clusters, you must complete the following steps to set the basic HDFS permission:

- Enable HDFS permissions
- Configure permissions for the Hive warehouse

```
hadoop fs - chmod 1771 / user / hive / warehouse
It can be set as follows , in which 1 denotes
stick bit ( i . e . cannot delete files / folders created
by others )
hadoop fs - chmod 1777 / user / hive / warehouse
```

With the basic permission set, users and user groups can create, read, and write tables as usual by authorizing the folder warehouse.

```
sudo su has
# Grant rwx permission of folder warehouse to user
test
hadoop fs - setfacl - m user : test : rwx / user / hive
/ warehouse
# Grant rwx permission of folder warehouse to user
hivegrp
hadoo fs - setfacl - m group : hivegrp : rwx / user /
hive / warehouse
```

With HDFS authorized, users and user groups can create, read, and write tables as usual. Data in Hive tables that is created by different users in HDFS can only be accessed by the users themselves.

Verification

- The test user creates a table testtbl.

```
hive > create table testtbl ( a string );
FAILED: Execution Error, return code 1 from org.
apache.hadoop.hive ql.exec.DDLTask.MetaExcept ion
( message: Got exception: org.apache.hadoop.security.
AccessCont rolExcepti on Permission denied: user = test ,
access = WRITE , inode ="/ user / hive / warehouse / testtbl ":
hadoop : hadoop : drwxrwx -- t
at org.apache.hadoop.hdfs.server.namenode.FSPermissi
onChecker.check ( FSPermissi onChecker . java : 320 )
at org.apache.hadoop.hdfs.server.namenode.FSPermissi
onChecker.check ( FSPermissi onChecker . java : 292 )
```

An error occurs due to the lack of permissions. Permissions should be granted to the test user.

```
# Switch from root account to has account
su has
# Add ACL and grant rwx permission s of the
directory warehouse to the account test .
```

```
hadoop fs - setfacl - m user : test : rwx / user / hive /
warehouse
```

The test account recreates the database successfully.

```
hive > create table testtbl ( a string );
OK
Time taken : 1.371 seconds
# View the directory of testtbl in HDFS . From the
permission s it can be seen that only the groups
test and hadoop can read data from the table
created by the user test , while other users have
no permission s
hadoop fs - ls / user / hive / warehouse
drwxr - x --- - test hadoop 0 2017 - 11 - 25 14
: 51 / user / hive / warehouse / testtbl
# Insert a record
hive > insert into table testtbl select " hz "
```

- User foo accesses the table testtbl.

```
# drop table
hive > drop table testtbl ;
FAILED : Execution Error , return code 1 from org .
apache . hadoop . hive . ql . exec . DDLTask . MetaExcept ion
( message : Permission denied : user = foo , access = READ ,
inode = "/ user / hive / warehouse / testtbl " : test : hadoop :
drwxr - x ---
    at org . apache . hadoop . hdfs . server . namenode .
FSPermissi onChecker . check ( FSPermissi onChecker . java : 320
)
    at org . apache . hadoop . hdfs . server . namenode .
FSPermissi onChecker . checkPermi ssion ( FSPermissi onChecker .
java : 219 )
    at org . apache . hadoop . hdfs . server . namenode .
FSPermissi onChecker . checkPermi ssion ( FSPermissi onChecker .
java : 190 )
# alter table
hive > alter table testtbl add columns ( b string );
FAILED : SemanticEx ception Unable to fetch table
testtbl . java . security . AccessCont rolExcepti on :
Permission denied : user = foo , access = READ , inode = "/
user / hive / warehouse / testtbl " : test : hadoop : drwxr - x ---
    at org . apache . hadoop . hdfs . server . namenode .
FSPermissi onChecker . check ( FSPermissi onChecker . java : 320
)
    at org . apache . hadoop . hdfs . server . namenode .
FSPermissi onChecker . checkPermi ssion ( FSPermissi onChecker .
java : 219 )
    at org . apache . hadoop . hdfs . server . namenode .
FSPermissi onChecker . checkPermi ssion ( FSPermissi onChecker .
java : 190 )
    at org . apache . hadoop . hdfs . server . namenode .
FSDirector y . checkPermi ssion ( FSDirector y . java : 1720 )
# select
hive > select * from testtbl ;
FAILED : SemanticEx ception Unable to fetch table
testtbl . java . security . AccessCont rolExcepti on :
Permission denied : user = foo , access = READ , inode = "/
user / hive / warehouse / testtbl " : test : hadoop : drwxr - x ---
```

```

    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:320)
    at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.checkPermission(FSPermissionChecker.java:219)

```

Foo cannot perform operations on the table created by the test user. HDFS authorization is needed to grant permissions to foo.

```

su has
# Only read permission is granted, and write
# permission can also be granted as needed (for
# example, alter)
# Note: - R: Set files in the folder testtbl to
# readable
hadoop fs -setfacl -R -m user:foo:r-x /user/hive/warehouse/testtbl
# The table can be selected successfully
hive> select * from testtbl;
OK
hz
Time taken: 2.134 seconds, Fetched: 1 row(s)

```



Note:

You can create a Hive user group, authorize it, and then add new users to it.

SQL Standard Based Authorization

• Scenario

If a cluster user cannot access data in Hive through an HDFS or Hive client, and the only way is to run Hive related commands through `HiveServer` (`beeline` , `jdbc` , and so on), SQL standard based authorization can be used.

If users can use the Hive shell or similar methods and as long as `hive-site.xml` in the user's client has not been configured, Hive can still be accessed as usual, even if the following settings are implemented.

For more information, see Hive's [SQL Standard Based Authorization](#) guide.

• Add configuration

- The configuration is provided to `HiveServer`.
- In the cluster Configuration Management page, click `Hive > Configuration > hive-site.xml > Add Custom Configuration`.

```

< property >
  < name > hive . security . authorization . enabled </ name >
  < value > true </ value >
</ property >
< property >

```

```
< name > hive . users . in . admin . role </ name >
< value > hive </ value >
</ property >
< property >
< name > hive . security . authorization . createtable . owner
. grants </ name >
< value > ALL </ value >
</ property >
```

- Restart HiveServer2

Restart HHiveServer2 in the cluster Configuration Management page.

- Permission operation commands

For more information on command operations, click [here](#).

- Verification

- User foo accesses the test user's table, testtbl, through beeline.

```
2 : jdbc : hive2 :// emr - header - 1 . cluster - xxx : 10 >
select * from testtbl ;
Error : Error while compiling statement : FAILED :
HiveAccess ControlException Permission denied : Principal
[ name = foo , type = USER ] does not have following
privileges for operation QUERY [[ SELECT ] on Object
[ type = TABLE_OR_V IEW , name = default . testtbl ]] ( state =
42000 , code = 40000 )
```

- Grant permissions.

```
Switch to account test to grant select permission
to user foo
hive > grant select on table testtbl to user foo ;
OK
Time taken : 1 . 205 seconds
```

- Foo can select as usual.

```
0 : jdbc : hive2 :// emr - header - 1 . cluster - xxxxx : 10 >
select * from testtbl ;
INFO : OK
+-----+
| testtbl . a |
+-----+
| hz          |
+-----+
1 row selected ( 0 . 787 seconds )
```

- Revoke permissions.

```
Switch to account test , and revoke the select
permission from user foo
hive > revoke select from user foo ;
OK
```

Time taken : 1 . 094 seconds

- Foo cannot select testtbl data.

```
User foo cannot select data from table testtbl .
Error : Error while compiling statement : FAILED :
HiveAccess ControlException Permission denied : Principal
[ name = foo , type = USER ] does not have following
privileges for operation QUERY [[ SELECT ] on Object
[ type = TABLE_OR_VIEW , name = default . testtbl ]] ( state =
42000 , code = 40000 )
```

12.4 HBase authorization

Any account can perform any operation on an HBase cluster without being authorized, including disabling and dropping tables and performing major compaction.



Note:

For clusters that do not have Kerberos authentication, users can forge identities to access the cluster service, even when HBase authorization is enabled. Therefore, we recommend that you create a high-security cluster (for example, supporting Kerberos) as detailed in the [Introduction to Kerberos](#).

Add configuration

In the Configuration Management page, choose HBase > Configuration > hbase-site > Custom Configuration in the HBase cluster.

Add the following parameters:

```
< property >
  < name > hbase . security . authorization </ name >
  < value > true </ value >
</ property >
< property >
  < name > hbase . coprocessor . master . classes </ name >
  < value > org . apache . hadoop . hbase . security . access .
AccessController </ value >
</ property >
< property >
  < name > hbase . coprocessor . region . classes </ name >
  < value > org . apache . hadoop . hbase . security . token .
TokenProvider , org . apache . hadoop . hbase . security . access .
AccessController </ value >
</ property >
< property >
  < name > hbase . coprocessor . regionserv er . classes </ name >
  < value > org . apache . hadoop . hbase . security . access .
AccessController , org . apache . hadoop . hbase . security . token .
TokenProvider </ value >
```



```
</ property >
```

Restart the HBase cluster

In the HBase cluster's Configuration Management page, click HBase > Operations > RESTART All Components.

Authorization (ACL)

- Basic concepts

In HBase, authorization consists of three elements: the granting of operational permissions for a certain scope of resources to a certain entity.

- Resources in a certain scope

- Superuser

A superuser can perform any operations. The account that runs the HBase service is the superuser by default. You can also add superusers by configuring the value of `hbase.superuser` in `hbase-site.xml`.

- Global

Global Scope has administrator permissions for all tables in the cluster.

- Namespace

This has permission control in Namespace Scope.

- Table

This has permission control in Table Scope.

- ColumnFamily

This has permission control in ColumnFamily Scope.

- Cell

This has permission control in Cell Scope.

- Operational permissions

- Read (R)

Read data from resources in a certain scope.

- Write (W)

Write data to resources in a certain scope.

- Execute (X)

Execute co-processor in a certain scope.

- Create (C)

Create or delete a table in a certain scope.

- Admin (A)

Perform cluster-related operations in a certain scope, such as balance or assign.

- Entity

- User

Authorize a user.

- Group

Authorize a user group.

- Authorization command

- grant

```
grant < user > < permission s > [<@ namespace > [< table > [<
column family > [< column qualifier >]]]
```



Note:

- The authorization methods for users and groups are the same. The prefix @ needs to be added for groups.

```
grant ' test ',' R ',' tbl1 ' # grant the read
permission of the table tbl1 to the user test
.
grant '@ test ',' R ',' tbl1 ' # grant the read
permission of the table tbl1 to the user group
test .
```

- The prefix @ needs to be added for namespace.

```
grant ' test ' C ',' @ ns_1 ' # grant the create
permission of the namespace @ ns_1 to the user
test .
```

- revoke
- user_permissions (view permissions)

12.5 Kafka authorization

If Kafka authentication (for example, Kerberos authentication or another simple authorization based on a user name and password) is disabled, users can access services with forged identities, even if Kafka authorization is enabled. Therefore, we

recommend that you create a high-security Kafka cluster. For more information, see [Introduction to Kerberos](#).

**Note:**

The permission configurations detailed in this section are for high-security E-MapReduce clusters only (Kafka is started in Kerberos).

Add configurations

1. On the Cluster Management page, click View Details next to the Kafka cluster.
2. In the navigation pane on the left, click the Clusters and Services tab, and click Kafka in the service list.
3. At the top of the page, click the Configuration tab.
4. In the upper-right corner of the Service Configuration list, click Custom Configuration and add the following parameters:

| Key | Value | Description |
|-----------------------|---|---|
| authorizer.class.name | kafka.security.auth.SimpleAclAuthorizer | N/A |
| super.users | User:kafka | User:kafka is required. Other users can be added and separated by semicolons (;). |

**Note:**

zookeeper.set.acl is used to set the permissions for Kafka to operate data in ZooKeeper. It is already set to true in the E-MapReduce cluster, so you do not need to add this configuration here. With the configuration set to true, only users named Kafka who have passed the Kerberos authentication can run the kafka-topics.sh command in the Kerberos environment. Kafka-topics.sh can read, write, and modify data in ZooKeeper.

Restart a Kafka cluster

1. On the Cluster Management page, click View Details next to the Kafka cluster you want to operate in the Operation column.
2. In the navigation pane on the left, click the Clusters and Services tab, and click Actions to the right of Kafka on the service list.

3. In the drop-down menu, select RESTART All Components. Enter the record information and click OK.

Authorization (ACL)

- Basic concepts

Definition in official Kafka documentation:

```
Kafka ACLs are defined in the general format of "
Principal P is [ Allowed / Denied ] Operation O From
Host H On Resource R "
```

This indicates that the ACL process relates to Principal, Allowed/Denied, Operation Host, and Resource.

- Principal: username

| Security protocol | Value |
|-------------------|--|
| PLAINTEXT | ANONYMOUS |
| SSL | ANONYMOUS |
| SASL_PLAINTEXT | If the mechanism is PLAIN, the user name is specified by client_jaas.conf. If the mechanism is GSSAPI, the user name is principal specified by client_jaas.conf. |
| SASL_SSL | |

- Allowed/Denied
- Operation: Operations include Read, Write, Create, DeleteAlter, Describe, ClusterAction, AlterConfigs, DescribeConfigs, IdempotentWrite, and All.
- Host: The target machine.
- Resource: Resource objects, including Topic, Group, Cluster, and TransactionalId.

For detailed mapping relationships between operations and resources, such as the supporting relationships between resources and the authorization of operations, see [KIP-11 - Authorization Interface](#).

- Authorization command

Perform authorization using the kafka-acls.sh script (/usr/lib/kafka-current/bin/kafka-acls.sh). For more information about how to use this script to authorize Kafka, run the `kafka - acls . sh -- help` command.

Procedure

Complete the following operations on the master node of the high-security Kafka cluster you created in E-MapReduce.

1. Create a user named test.

```
useradd test
```

2. Create a topic.

`zookeeper.set.acl` is set to true, and `kafka-topics.sh` must be run under a Kafka account. The Kafka account must pass Kerberos authentication.

```
# The Kerberos authentication information related to
the kafka account is set in kafka_client_test.conf
.
export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/etc/ecm/kafka-conf/kafka_client_test.conf"
# Change the ZooKeeper address to the actual address
(run hostnamed to acquire) of your Kafka cluster.
kafka-topics.sh --create --zookeeper emr-header-1:2181/kafka-1.0.0 --replication-factor 3 --
partitions 1 --topic test
```

3. Run `kafka-console-producer.sh` with the test user.

- a. Create a keytab file for the test user to authenticate ZooKeeper and Kafka.

```
su root
sh /usr/lib/has-current/bin/hadmin-local.sh /etc/ecm/has-conf-k/etc/ecm/has-conf/admin.keytab
HadminLocalTool.local: # Press Enter to display the
usage instructions on some commands.
HadminLocalTool.local: addprinc # Enter a command
and press Enter to display the usage instructions
on the command.
HadminLocalTool.local: addprinc -pw 123456 test #
Add a principal for the test user and set the
password to 123456.
HadminLocalTool.local: ktadd -k /home/test/test.keytab test # Export the keytab file for later
use.
```

- b. Add a `kafka_client_test.conf` file.

Put the file in `/home/test/kafka_client_test.conf`. The content of the file is as follows:

```
KafkaClient {
com.sun.security.auth.module.Krb5LoginModule
required
useKeyTab = true
storeKey = true
serviceName = "kafka"
keyTab = "/home/test/test.keytab"
```

```
principal = "test";
};
// Zookeeper client authentication
Client {
  com.sun.security.auth.module.Krb5LoginModule
  required
  useKeyTab = true
  useTicketCache = false
  serviceName = "zookeeper"
  keyTab = "/home/test/test.keytab"
  principal = "test";
};
```

c. Add producer.conf.

Put the file in `/home/test/producer.conf`. The content of the file is as follows:

```
security.protocol = SASL_PLAINTEXT
sasl.mechanism = GSSAPI
```

d. Run kafka-console-producer.sh.

```
su test
export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/home/test/kafka_client_test.conf"
kafka-console-producer.sh --producer.config /home/test/producer.conf --topic test --broker-list emr-worker-1:9092
```

Because no ACL is set, an error is reported after the preceding command is run:

```
org.apache.kafka.common.errors.TopicAuthorizationException: Not authorized to access topics: [test]
```

e. Set an ACL.

Similarly, the `kafka-acls.sh` command must be run under the Kafka account.

```
su kafka
export KAFKA_OPTS="-Djava.security.auth.login.config=/etc/ecm/kafka-conf/kafka_client_jaas.conf"
kafka-acls.sh --authorizer-properties zookeeper.connect=emr-header-1:2181/kafka-1.0.0 --add --allow-principal User:test --operation Write --topic test
```

f. Run kafka-console-producer.sh again.

```
su test
export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/home/test/kafka_client_test.conf"
```

```
kafka - console - producer . sh -- producer . config / home /
test / producer . conf -- topic test -- broker - list emr -
worker - 1 : 9092
```

Normal output is as follows:

```
[ 2018 - 02 - 28 22 : 25 : 36 , 178 ] INFO Kafka commitId
: aaa7af6d4a 11b29d ( org . apache . kafka . common . utils .
AppInfoPar ser )
> alibaba
> E - MapReduce
>
```

4. Run `kafka - console - consumer . sh` with the test user.

After `kafka-console-producer.sh` is successfully run and data is written to the topic, you can run `kafka - console - consumer . sh` to perform a consumption test.

a. Add consumer.conf.

Put the file in `/ home / test / consumer . conf` . The content of the file is as follows:

```
security . protocol = SASL_PLAIN TEXT
sasl . mechanism = GSSAPI
```

b. Run `kafka-console-consumer.sh`.

```
su test
# Kafka_clie nt_test . conf is used in the same way
as kafka - console - producer . sh .
export KAFKA_HEAP_OPTS="- Djava . security . auth . login .
config =/ home / test / kafka_clie nt_test . conf "
kafka - console - consumer . sh -- consumer . config consumer
. conf -- topic test -- bootstrap - server emr - worker -
1 : 9092 -- group test - group -- from - beginning
```

Because no permissions are set, an error is reported:

```
org . apache . kafka . common . errors . GroupAutho rizationEx
ception : Not authorized to access group : test - group
```

c. Set an ACL.

```
su kafka
export KAFKA_HEAP_OPTS="- Djava . security . auth . login .
config =/ etc / ecm / kafka - conf / kafka_clie nt_jaas . conf
"
# test - group permission
kafka - acls . sh -- authorizer - properties zookeeper .
connect = emr - header - 1 : 2181 / kafka - 1 . 0 . 0 -- add --
allow - principal User : test -- operation Read -- group
test - group
# topic permission
kafka - acls . sh -- authorizer - properties zookeeper .
connect = emr - header - 1 : 2181 / kafka - 1 . 0 . 0 -- add --
```



```
allow - principal User : test -- operation Read -- topic  
test
```

d. Run `kafka - console - consumer . sh` again.

```
su test  
# Kafka_clie nt_test . conf is used in the same way  
as kafka - console - producer . sh .  
export KAFKA_HEAP_OPTS="- Djava . security . auth . login .  
config =/ home / test / kafka_clie nt_test . conf "  
kafka - console - consumer . sh -- consumer . config consumer  
. conf -- topic test -- bootstrap - server emr - worker -  
1 : 9092 -- group test - group -- from - beginning
```

Normal output is as follows:

```
alibaba  
E - MapReduce
```

12.6 Ranger

12.6.1 Introduction to Ranger

Apache Ranger provides a centralized framework for permission management, implementing fine-grained access control for components in the Hadoop ecosystem, such as HDFS, Hive, YARN, Kafka, Storm, and Solr. It also provides a UI that allows administrators to perform operations more conveniently.

Create a cluster

Select the Ranger service when you create a cluster in E-MapReduce 2.9.2/3.9.0 or later on the E-MapReduce console.

If an E-MapReduce cluster 2.9.2/3.9.0 or later has been created without Ranger, you can go to the Clusters and Services page to add it.



Note:

- When Ranger is enabled, there is no impact or limitation on the application until the security control policy is set.
- The user policy set in Ranger is the cluster Hadoop account.

Ranger UI

After installing Ranger on the cluster, click **Manage** in the **Actions** column, and then click **Access Links and Ports** in the navigation pane on the left. You can then access the Ranger UI by clicking on the link, as shown in the following figure.

Enter the Ranger UI. The default user name and password are both **admin**, as shown in the following figure.

Modify the password

After you first log on, the administrator needs to modify the password of the admin account, as shown in the following figure.

After you change the admin password, in the admin drop-down list in the upper-right corner, click **Log Out**. You can then log on again with the new password.

Integrate Ranger into other services

After completing the preceding steps, you can integrate Ranger into the services in the cluster to control the relevant permissions. For more information, see the following:

- [Integrate Ranger into HDFS](#)
- [Integrate Ranger into Hive](#)
- [Integrate Ranger into HBase](#)

12.6.2 Integrate Ranger into HDFS

Procedure

This section describes the step-by-step process for integrating Ranger into HDFS.

- **Enable the HDFS plug-in**

1. On the Cluster Management page, click Manage next to the cluster you want to operate in the Actions column.
2. Click Ranger in the service list to enter the Ranger Management page.
3. On the Ranger Configuration page, click the Actions drop-down menu in the upper-right corner, select Enable HDFS PLUGIN, and click OK.
4. Enter the record information in the prompt box and click OK.

You can check the progress by clicking View Operation Logs in the upper-right corner of the page.

- **Restart NameNode**

After enabling the HDFS plug-in, you need to restart NameNode. To do so, complete the following steps:

1. In the Ranger Management page, click the Ranger drop-down menu in the upper-left corner, and select HDFS.
2. Click Actions in the upper-right corner of the page and select RESTART NameNode.
3. You can check the progress by clicking View Operation Logs in the upper-right corner of the page.

- Add the HDFS service to Ranger UI

For more information about how to access the Ranger UI, see [Introduction to Ranger](#).

Add the HDFS service.

- Standard cluster

To check the mode of the cluster you created, go to the Cluster Overview page. If your cluster is in standard mode, configure it as follows:

- High-security-mode cluster

To check the mode of the cluster you created, go to the Cluster Overview page. If your cluster is in high-security mode, configure it as follows:

Permission configuration

After integrating Ranger into HDFS, you can set permissions, such as granting the test user the write or execute permission for `/ user / foo` .

In the preceding figure, click `emr-hdfs` to enter the policy configuration page.

Permissions are granted to the test user. They can now access the HDFS path of `/ user / foo` .



Note:

The policy takes effect one minute after it is added.

12.6.3 Integrate Ranger into Hive

Procedure

This section describes the step-by-step process for integrating Ranger into Hive.

- Hive access model

You can access Hive data in three ways: HiveServer2, Hive client, and HDFS.

- HiveServer2

- **Mode:** Use the Beeline client or the JDBC code to run the related Hive script through HiveServer2.

- **Permission settings:**

Hive's [SQL Standard Based Authorization](#) is used to control the permissions of HiveServer2.

Hive's table- and column-level permission control in Ranger is also used for HiveServer2. However, if you are still able to access Hive data through a Hive client or HDFS, table- or column-level permission control is insufficient and further control is required.

- Hive client

- **Mode:** Access from a Hive client.

- **Permission settings:**

The Hive client requests the metastore to perform DDL operations, such as altering tables, adding columns, and reading and processing data in HDFS, by submitting MapReduce jobs.

Hive's [Storage Based Authorization](#) is used to control the permissions of Hive clients. It determines whether a user can perform DDL and DML operations based on the read and write permissions of the HDFS path where the table involved in SQL is located, such as `ALTER TABLE test ADD COLUMNS (b STRING)`.

In Ranger, you can control the permissions of the HDFS path in Hive tables. This, in combination with the Hive metastore which is configured with storage-based authorization, enables you to implement permission control over the access of the Hive client.



Note:

The DDL operation permissions of a Hive client are actually controlled by the underlying HDFS permissions. If you have HDFS permissions, you also have DDL permissions for tables, such as dropping and altering tables.

- HDFS

- Mode: HDFS client and code.

- Permission settings:

- To have direct access to HDFS, you need to add permission control for HDFS on the underlying HDFS data of the Hive tables.

- You can use Ranger to perform permission control for the underlying HDFS path of Hive tables.

- Enable the Hive plug-in

1. On the Cluster Management page, click Manage next to the cluster you want to operate in the Actions column.
2. Click Ranger in the service list to enter the Ranger Management page.
3. On the Ranger Configuration page, click the Actions drop-down menu in the upper-right corner, select Enable Hive PLUGIN, and click OK.
4. Enter the record information in the prompt box and click OK.

You can check the progress by clicking View Operation Logs in the upper-right corner of the page.



Note:

After you enable the Hive plug-in and restart Hive, HiveServer2 and Hive client scenarios are configured accordingly. For more information about HDFS permissions, see [Integrate Ranger into HDFS](#).

- Restart Hive

After enabling the Hive plug-in, you need to restart Hive. To do so, complete the following steps:

1. In the Ranger Management page, click the Ranger drop-down menu in the upper-left corner, and select Hive.
2. Click Actions in the upper-right corner, select RESTART All Components from the drop-down menu, and click OK.
3. You can check the progress by clicking View Operation Logs in the upper-right corner of the page.

- Add the Hive service to the Ranger UI

For more information about how to access the Ranger UI page, see [Introduction to Ranger](#).

Add the Hive service.

- Instructions

Enter a fixed value for the following configuration items:

| Name | Value |
|----------------------|---------------------------------|
| Service Name | emr-hive |
| jdbc.driverClassName | org.apache.hive.jdbc.HiveDriver |

- Enter a variable value for the following configuration items:

| Name | Value |
|----------|--|
| jdbc.url | Standard cluster: jdbc:hive2://emr-header-1:10000/ high-High-security cluster: jdbc:hive2://{master1_fullhost}:10000/;principal=hive/{master1_fullhost}@EMR.\$id.COM |

| Name | Value |
|----------------------------|--|
| policy.download.auth.users | Standard cluster: hadoop High-security cluster: hive |

`${master1_fullhost}` is the long domain name of master1. To obtain this name, log on to master1 and run the `hostname` command. The number in `${master1_fullhost}` is the value of `$id`.

Permission configuration

After integrating Ranger into Hive, you can set permissions, such as granting user foo the Select permission for column A in the `testdb.test` table.

In the preceding figure, click `emr-hive` to enter the policy configuration page.

Permissions are granted to user foo. They can now access the `testdb.test` table.



Note:

The policy takes effect one minute after it is added.

12.6.4 HBase configurations

The Ranger introduction topic describes how to start a Ranger cluster in the EMR console and the preparations. This topic describes how to integrate HBase with Ranger.

Enable HBase Plugin

1. On the Cluster Management page, click **Clusters and Services** in the Actions column for the cluster that you want to operate.
2. In the Services list, click **RANGER** and click the **Configuration** tab to go to the Configuration tab page.
3. On the Configuration tab page, select **EnableHBase** from the Actions drop-down list.
4. Click **View Operation Logs** to view the status of operations.

Add HBase Service in Ranger UI

For access to Ranger UI, see [Ranger introduction](#).

Add the HBase service in Ranger UI.



Note:

`#{id}`: You can log on to the host and run the `host` command. The number in `hostname` is the value of `#{id}`.

Restart HBase

Restart HBase for the preceding procedures to take effect. Perform the following steps.

1. On the Ranger page, click RANGER. From the drop-down list, select HBase.
2. From the Actions drop-down list, select RESTART All Components.
3. Click View Operation Logs to view the status of operations.

Set Administrator Account

You need to set permissions of the administrator accounts (administrator permissions) for running administrative commands such as `balance` / `compaction` / `flush` / `split`.

Click the Edit icon in the Action column of the policy that you want to set users for. Add user accounts in the Users column as needed. You can also modify the permissions. For example, only retain the default admin permissions. You need to set `hbase` as the administrator account.

If you use Phoenix, you also need to add the following policies in HBase for Ranger.

| Table | SYSTEM. * |
|---------------|-----------------------------|
| Column Family | * |
| Column | * |
| Groups | public |
| Permissions | Read Write , Create , Admin |

Permission configuration examples

After HBase is integrated into Ranger, you can perform permission configurations. For example, grant user test the `Create / Write / Read` permissions on the `foo_ns:test` table.

Click `emr-hbase` as shown in this figure to go to the configurations page. Configure permissions.

It takes about one minute to synchronize the user and group information of the cluster.

Follow the steps as shown in the figure to complete the adding of the policy. Then user test can access the `foo_ns:test` table.



Note:

A policy will not take effect until it has been added over one minute.

12.6.5 Data masking in Hive

Ranger supports data masking in Hive by masking return values of SELECT statements to hide sensitive information from users.



Note:

This feature only supports scenarios involving HiveServer2, such as using Beeline, JDBC, or Hue to run SELECT statements. HiveClient-based scenarios are not supported, such as `hive -e 'select xxxx'`.

This topic describes how to use this feature in E-MapReduce.

Configure the Hive plug-in for Ranger

For more information, see [Hive configurations](#).

Configure Data Mask Policy

You can mask Hive data accessed by users on the `emr - hive` service page in the Ranger UI.

- Ranger supports a variety of masking types, such as show the first four characters, show the last four characters, and Hash masks.
- A mask policy does not support wildcards. For example, you cannot use an asterisk (*) to replace columns or tables in a mask policy.
- Each mask policy is corresponding to one column. You need to configure mask policies for each column.

Perform the following steps to configure a mask policy.

Save your mask policy.

Mask test data

- Scenario:

User test selects column a from the testdb1.testtbl table to display only the first four characters of each value.

- Procedure:

1. Configure a mask policy

The last figure in the previous section shows the mask policy for this scenario. "show first 4" is selected as the masking type.

2. Verify data masking

User test uses Beeline to connect to HiveServer2 and runs the `select a from testdb1 . testtbl` statement.

As shown above, after user test runs the SELECT statement, only the first four characters of values of column a are shown. The rest characters are replaced by `x` for data masking.

13 Kerberos authentication

13.1 Introduction to Kerberos

Kerberos is a network authentication protocol that allows nodes communicating over a non-secure network to securely prove their identity. From versions 2.7.x and 3.5.x onwards, E-MapReduce supports the creation of clusters in which open source components are started in the Kerberossecurity mode. In this mode, only authenticated clients can access the cluster service, such as HDFS.

Prerequisites

The Kerberos components supported by the latest E-MapReduce version are shown in the following table:

| Component name | Component version |
|----------------|-------------------|
| YARN | 2.7.2 |
| Spark | 2.1.1/1.6.3 |
| Hive | 2.0.1 |
| Tez | 0.8.4 |
| ZooKeeper | 3.4.6 |
| Hue | 3.12.0 |
| Zeppelin | 0.7.1 |
| Oozie | 4.2.0 |
| Sqoop | 1.4.6 |
| HBase | 1.1.1 |
| Phoenix | 4.7.0 |

**Note:**

Kafka, Presto, and Storm do not currently support Kerberos.

Create a security cluster

In the software configuration tab on the cluster creation page, you can turn on High Security Mode, as shown in the following figure:

Kerberos authentication

Kerberos is an identity authentication protocol based on symmetric key cryptography. As a third-party authentication service, Kerberos can provide its authentication function for other services. It also supports SSO, and the client can access multiple services, such as HBase and HDFS, after authentication.

The Kerberos protocol process is mainly divided into two stages: the KDC authenticates the client ID, and the service authenticates the client ID.

- KDC

Kerberos server

- Client

If a user (principal) needs to access the service, the KDC and service authenticate the principal's identity.

- Service

Services that have integrated with Kerberos include HDFS, YARN, and HBase.

- KDC ID authentication

Before a principal can access a service integrated with Kerberos, it must first pass KDC ID authentication.

After doing so, the client receives a ticket-granting ticket (TGT), which can be used to access a service that has integrated Kerberos.

- Service ID authentication

When a principal receives the TGT, it can access the service. It uses the TGT and the name of the service that it must access (such as HDFS) to obtain a service-granting ticket (SGT) from the KDC, and uses the SGT to access the service. This then uses the relevant information to conduct ID authentication on the client. After passing authentication, the client can access the service as normal.

Kerberos and E-MapReduce

When you create a cluster, services in the E-MapReduce Kerberos security cluster start in the Kerberos security mode.

- The Kerberos server is a HasServer
 - Log on to the [Alibaba Cloud E-MapReduce console](#), choose Cluster > > Configuration Management > HAS, and conduct operations such as view, modify configuration, and restart.
 - Non-HA clusters are deployed on the emr-header-1 node, whereas HA clusters are deployed on both the emr-header-1 and emr-header-2 nodes.
- Supports four ID authentication methods

HasServer supports the following four ID authentication methods. The client can specify the method that is used by HasServer by configuring the relevant parameters.

- ID authentication compatible with MIT Kerberos

Client configuration:

```
If you want to execute a client request on a cluster node, you must set hadoop.security.authentication.use.has in /etc/ecm/hadoop-conf/core-site.xml to false. In case of any jobs are running through the execution plan of the console, then values in the /etc/ecm/hadoop-conf/core-site.xml file on the master node must not be modified. Otherwise, the job in the execution plan fails because of the authentication failure. You can follow these steps:
export HADOOP_CONF_DIR=/etc/has/hadoop-conf Export a temporary environment variable. The hadoop.security.authentication.use.has value under this path has already been set to false.
```

Access method: You can use open source clients to access the service, such as an HDFS client. For more information, [click here](#).

- RAM ID authentication

Client configuration:

```
If you want to run a client request on a cluster node, you must set hadoop.security.authentication.use.has in /etc/ecm/hadoop-conf/core-site.xml to true, and auth_type in /etc/has/has-client.conf to RAM. In case of any jobs are running through the execution plan of the console, then values in the /etc/ecm/hadoop-conf/core-site.xml and /etc/has/has-client.conf files on the master node must not be modified. Otherwise, the job in the execution plan fails because of the authentication failure. You can use the following method:
```

```
export HADOOP_CONF_DIR=/etc/hadoop/conf; export
HAS_CONF_DIR=/path/to/has-client.conf Export a
temporary environment variable, and then set the
auth_type in the has-client.conf file of the
HAS_CONF_DIR folder to RAM.
```

Access method: The client must use a software package of the cluster, such as Hadoop or HBase. For more information, [click here](#).

- LDAP ID authentication

Client configuration:

```
If you want to execute a client request on a
cluster node, you must set
hadoop.security.authentication.use.has in /etc
/ecm/hadoop-conf/core-site.xml to true, and
auth_type in /etc/has/has-client.conf to LDAP.
In case of any jobs are running through the
execution plan of the console, then values in the
/etc/ecm/hadoop-conf/core-site.xml and /etc/
has/has-client.conf files on the master node
must not be modified. Otherwise, the job in the
execution plan fails because of the authentication
failure. You can follow these steps:
export HADOOP_CONF_DIR=/etc/hadoop/conf; export
HAS_CONF_DIR=/path/to/has-client.conf Export
temporary environment variables, and then set the
auth_type in the has-client.conf file of the
HAS_CONF_DIR folder to LDAP.
```

Access method: The client must use a software package of the cluster, such as Hadoop or HBase. For more information, [click here](#).

- Execution plan authentication

If you have jobs submitted through the execution plan of the E-MapReduce console, you must not modify the default configuration of the emr-header-1 node.

Client configuration:

```
Set hadoop.security.authentication.use.has in /
etc/ecm/hadoop-conf/core-site.xml to true, and
auth_type in /etc/has/has-client.conf on emr-
header-1 to EMR.
```

For more information, [click here](#).

- Others

Log on to the master node to access the cluster

The administrator can use the Has account (the default logon method is MIT-Kerberos-compatible) to log on to the master node and access the cluster service. This facilitates troubleshooting and O&M tasks.

```
& gt ; sudo su has
& gt ; hadoop fs - ls /
```



Note:

Other accounts can also be used to log on to the master node, provided that such accounts have already passed Kerberos authentication. In addition, if you have to use the MIT-Kerberos-compatible method on the master node, you must first export an environment variable under this account.

```
export HADOOP_CONF_DIR=/etc/hadoop/conf/
```

13.2 Authentication compatible with MIT Kerberos

This section details the MIT Kerberos authentication process through the HDFS service.

Authentication method

The Kerberos server in the E-MapReduce cluster is started at the master node. Some management operations must be performed with the root account of the master node (emr-header-1).

In the following example, the test user accesses the HDFS service to introduce relevant procedures.

- Execute `hadoop fs - ls /` on the gateway.
- Configure `krb5.conf`.

```
Use root account on the Gateway
```



```
scp root@emr-header-1:/etc/krb5.conf /etc/
```

- Add principal.

■ Log on to the emr-header-1 node and switch to the root account.

■ Open the admin tool in Kerberos.

```
sh /usr/lib/has-current/bin/hadmin-local .
sh /etc/ectm/has-conf-k/etc/ectm/has-conf/
admin.keytab
HadminLoca lTool . local : # Press Enter to view
the use of the commands
HadminLoca lTool . local : addprinc # Input the
command and press Enter to view the use of
the specific command
HadminLoca lTool . local : addprinc -pw 123456 test
# Add principal for the user test, and set the
password to 123456
```

- Export the keytab file.

Use the Kerberos admin tool to export the keytab file that corresponds to the principal.

```
HadminLoca lTool . local : ktadd -k /root/test.keytab
test # Export the keytab file, which can be used
subsequent ly
```

- Use kinit to obtain the ticket.

On the client machine where HDFS commands are executed, such as the gateway :

■ Add the Linux account test `useradd test` .

■ Install MIT Kerberos client tools.

MIT Kerberos tools can be used for related operations (such as kinit and klist). For more information, see [MIT Kerberos](#).

```
yum install krb5-libs krb5-workstation -y
```

■ Switch to the test account to execute kinit.

```
su test
# If the keytab file does not exist ,
execute
kinit # Press Enter
Password for test : 123456 # Done
# the keytab file exists , execute
kinit -kt test.keytab test
# View the ticket
klist
```



Note:

Application of MIT Kerberos tools

- Execute HDFS commands.

When a ticket is obtained, HDFS commands can be executed as usual.

```
hadoop fs -ls /
Found 5 items
drwxr-xr-x - hadoop hadoop 0 2017 -
11 - 12 14 : 23 / apps
drwx----- - hbase hadoop 0 2017 -
11 - 15 19 : 40 / hbase
drwxrwx--t+ - hadoop hadoop 0 2017 - 11
- 15 17 : 51 / spark-history
drwxrwxrwt - hadoop hadoop 0 2017 - 11 -
13 23 : 25 / tmp
drwxr-x--t - hadoop hadoop 0 2017
- 11 - 13 16 : 12 / user
```



Note:

To run a YARN job, you need to add the corresponding Linux accounts to all of the nodes in the cluster in advance. For more information, see [Add test account to the E-MapReduce cluster](#).

- Use Java to access HDFS.
- Use a local ticket cache.



Note:

To obtain the ticket, you need to execute kinit in advance. When the ticket expires, the application is not accessed.

```
public static void main ( String [] args ) throws
IOException {
    Configuration conf = new Configuration ();
    // Load the HDFS configuration, which is copied
    from the EMR cluster
    conf . addResource ( new Path ("/ etc / ecm / hadoop -
conf / hdfs - site . xml "));
    conf . addResource ( new Path ("/ etc / ecm / hadoop -
conf / core - site . xml "));
    // kinit needs to be executed in advance to
    obtain the ticket with the Linux account of the
    application
    UserGroupInformation . setConfiguration ( conf );
    UserGroupInformation . loginUserFromSubject ( null );
    FileSystem fs = FileSystem . get ( conf );
    FileStatus [] fsStatus = fs . listStatus ( new Path
("/"));
    for ( int i = 0 ; i < fsStatus . length ; i ++){
        System . out . println ( fsStatus [ i ]. getPath ().
toString ());
    }
}
```

}

- (Recommended) Use a keytab file.

**Note:**

The keytab is valid for a long time and has nothing to do with the local ticket.

```
public static void main ( String [] args ) throws
IOException {
    String keytab = args [ 0 ];
    String principal = args [ 1 ];
    Configuration conf = new Configuration ();
    // Load the HDFS configuration , which is copied
    from the EMR cluster
    conf . addResource ( new Path ("/ etc / ecm / hadoop - conf
/ hdfs - site . xml "));
    conf . addResource ( new Path ("/ etc / ecm / hadoop - conf
/ core - site . xml "));
    // Directly use keytab file , which is obtained
    through executing relevant commands on master - 1
    in the EMR cluster [ the commands are introduced
    earlier in this article ]
    UserGroupInformation . setConfiguration ( conf );
    UserGroupInformation . loginUserFromKeytab ( principal ,
keytab );
    FileSystem fs = FileSystem . get ( conf );
    FileStatus [] fsStatus = fs . listStatus ( new Path
("/"));
    for ( int i = 0 ; i < fsStatus . length ; i ++){
        System . out . println ( fsStatus [ i ]. getPath ().
toString ());
    }
}
```

POM dependencies are attached:

```
< dependencies >
  < dependency >
    < groupId > org . apache . hadoop </ groupId >
    < artifactId > hadoop - common </ artifactId >
    < version > 2 . 7 . 2 </ version >
  </ dependency >
  < dependency >
    < groupId > org . apache . hadoop </ groupId >
    < artifactId > hadoop - hdfs </ artifactId >
    < version > 2 . 7 . 2 </ version >
  </ dependency >
</ dependencies >
```

13.3 RAM authentication

In addition to supporting an authentication method compatible with MIT Kerberos, the Kerberos server in the E-MapReduce cluster also supports using Alibaba

Cloud Resource Access Management (RAM) as the identity information to perform authentication.

RAM ID authentication

[RAM](#) supports creating and managing RAM user accounts, as well as using these accounts to control access to various resources on the cloud.

The administrator of the master account can create RAM users on the RAM user management page (the user name must comply with Linux user name specifications) and download their AccessKey for the corresponding developer. The developer can then configure the AccessKey to pass Kerberos authentication and access the cluster service.

Unlike the MIT Kerberos authentication, RAM identity authentication does not require adding principals to the Kerberos server in advance.

The following example uses a RAM user account that has already been created to access a gateway.

- Add the RAM user to the E-MapReduce cluster.

The E-MapReduce security cluster's YARN uses LinuxContainerExecutor. Running the YARN job on a cluster requires all cluster nodes to add the user account that is going to run the job. LinuxContainerExecutor conducts the related permission validation based on the user account during the execution process.

The E-MapReduce cluster administrator executes the following code on the cluster's master node:

```
sudo su hadoop
sh adduser . sh test 1 2
```

The adduser.sh code is attached:

```
# Username
user_name =$ 1
# Master node count in the cluster . For example ,
the HA cluster has two master nodes .
master_cnt =$ 2
# Worker node count in the cluster
worker_cnt =$ 3
for (( i = 1 ; i <=$ master_cnt ; i ++))
do
    ssh -o StrictHost KeyCheckin g = no emr - header -$ i
    sudo useradd $ user_name
done
for (( i = 1 ; i <=$ worker_cnt ; i ++))
do
```

```
ssh -o StrictHostKeyChecking=no emr-worker-$i
sudo useradd $user_name
done
```

- The gateway administrator adds the test user account on the gateway machine.

```
useradd test
```

- The gateway administrator configures the basic Kerberos environment.

```
sudo su root
sh config_gateway_kerberos.sh 10.27.230.10 /
path/to/emr-header-1-pwd-file
# Ensures the value of the /etc/ecm/hadoop-conf/core-site.xml file on the Gateway is true
<property>
  <name>hadoop.security.authentication.use.has</name>
  <value>true</value>
</property>
```

The config_gateway_kerberos.sh script is attached:

```
# IP address of the emr-header-1 in the EMR cluster
masterip=$1
# Saves the corresponding root logon password file for masterip
masterpwdfile=$2
if ! type sshpass >/dev/null 2>&1; then
  yum install -y sshpass
fi
## Kerberos conf
sshpass -f $masterpwdfile scp root@$masterip:/etc/krb5.conf /etc/
mkdir /etc/has
sshpass -f $masterpwdfile scp root@$masterip:/etc/has/has-client.conf /etc/has
sshpass -f $masterpwdfile scp root@$masterip:/etc/has/truststore /etc/has/
sshpass -f $masterpwdfile scp root@$masterip:/etc/has/ssl-client.conf /etc/has/
# Modifies Kerberos client configuration, changing the default auth_type from EMR to RAM
# This file can be manually modified
sed -i 's/EMR/RAM/g' /etc/has/has-client.conf
```

- The test user logs on to the gateway and configures the AccessKey.

```
Log on the test account of Gateway
# Run the script
sh add_accesskey.sh test
```

The add_accesskey.sh script is attached to modify the AccessKey:

```
user=$1
if [[ `cat /home/$user/.bashrc | grep 'export AccessKey '` == "" ]]; then
  echo "
```

```
# Change to the test user 's AccessKeyId /
AccessKeyId secret
export AccessKeyId = YOUR_AccessKeyId
export AccessKeyIdSecret = YOUR_AccessKeyIdSecret
" >> ~/. bashrc
else
    echo $ user AccessKey has been added to . bashrc
fi
```

- The test user executes the command.

The test user is now able to execute the relevant commands to access the cluster service.

Execute HDFS commands:

```
[ test @ gateway ~]$ hadoop fs -ls /
17 / 11 / 19 12 : 32 : 15 INFO client . HasClient : The
plugin type is : RAM
Found 4 items
drwxr-x--- - has hadoop 0 2017 - 11 - 18
21 : 12 / apps
drwxrwxrwt - hadoop hadoop 0 2017 - 11 - 19
12 : 32 / spark - history
drwxrwxrwt - hadoop hadoop 0 2017 - 11 - 18
21 : 16 / tmp
drwxrwxrwt - hadoop hadoop 0 2017 - 11 - 18
21 : 16 / user
```

Run the Hadoop job:

```
[ test @ gateway ~]$ hadoop jar / usr / lib / hadoop - current
/ share / hadoop / mapreduce / hadoop - mapreduce - examples - 2 .
7 . 2 . jar pi 10 1
```

Run the Spark job:

```
[ test @ gateway ~]$ spark - submit -- conf spark . ui . view
. acls =* -- class org . apache . spark . examples . SparkPi
-- master yarn - client -- driver - memory 512m -- num
- executors 1 -- executor - memory 1g -- executor - cores
```

```
2 /usr/lib/spark-current/examples/jars/spark-examples_2.11-2.1.1.jar 10
```

13.4 LDAP authentication

E-MapReduce clusters also support authentication based on LDAP, which manages the account system through LDAP. The Kerberos client uses LDAP account information as identity information for authentication.

LDAP identity authentication

LDAP accounts can be shared with other services, such as Hue. You can use an LDAP service (in ApacheDS) configured in the E-MapReduce cluster or use an existing LDAP service, and you only need to configure it on the Kerberos server.

In the following example, an LDAP service (in ApacheDS) has been started by default in a cluster:

- Configure the basic environment in gateway management. (This is the same as the second part of RAM. If it has already been configured, this step can be skipped).

The only difference is that *auth_type* in */etc/has/has-client.conf* needs to be modified in LDAP.

You may also not modify */etc/has/has-client.conf*. The test user can copy the file, modify *auth_type* with their account, and specify the path through environment variables. For example:

```
export HAS_CONF_DIR=/home/test/has-conf
```

- Configure the LDAP administrator user name/password to Kerberos server (HAS) in the E-MapReduce console.

On the E-MapReduce console, enter Configuration Management > HAS Software, configure the LDAP administrator user name and password in the corresponding *bind_dn* and *bind_password* fields, and restart the HAS service.

In this example, the LDAP service is the ApacheDS service in the E-MapReduce cluster. Related fields can be obtained from ApacheDS.

- The E-MapReduce cluster administrator adds user information to LDAP.
 - Obtain the administrator user name and password for the ApacheDS LDAP service. *manager_dn* and *manager_password* can be seen in the E-MapReduce console's Configuration Management/ApacheDS Configuration page.
 - Add the test user and password to ApacheDS.

```
Log on to root account in the cluster emr -
header - 1 node
Create a file test.ldif with the following
content :
dn : cn = test , ou = people , o = emr
objectclas s : inetOrgPer son
objectclas s : organizati onalPerson
objectclas s : person
objectclas s : top
cn : test
sn : test
mail : test @ example . com
userpasswo rd : test1234
# Add to LDAP , in which - w denotes that password
is changed to manager_pa ssword
ldapmodify - x - h localhost - p 10389 - D " uid =
admin , ou = system " - w " Ns1aSe " - a - f test . ldif
# Delete test . ldif
rm test . ldif
```

Provide added user name/password to the test user.

- The test user configures the LDAP information.

```
Log on the test account of Gateway
# Run the script
sh add_ldap . sh test
```

The *add_ldap.sh* script is attached to modify the LDAP account information:

```
user =$ 1
if [[ ` cat / home /$ user /. bashrc | grep ' export LDAP_
' ` == "" ]]; then
echo "
# Modify to the user test ' s LDAP_USER / LDAP_PWD
export LDAP_USER = YOUR_LDAP_ USER
export LDAP_PWD = YOUR_LDAP_ USER
" >> ~/. bashrc
else
echo $ user LDAP user info has been added to .
bashrc
fi
```

- The test user accesses the cluster services.

Execute HDFS commands.

```
[ test @ iZbp1cyio1 8s5ymggr7y hrZ ~]$ hadoop fs - ls /
17 / 11 / 19 13 : 33 : 33 INFO client . HasClient : The
plugin type is : LDAP
```



```

Found 4 items
drwxr-x--- - has      hadoop      0    2017 - 11 - 18
21 : 12 / apps
drwxrwxrwt - hadoop   hadoop      0    2017 - 11 - 19
13 : 33 / spark - history
drwxrwxrwt - hadoop   hadoop      0    2017 - 11 - 19
12 : 41 / tmp
drwxrwxrwt - hadoop   hadoop      0    2017 - 11 - 19
12 : 41 / user

```

Run the Hadoop/Spark job.

13.5 Execution plan authentication

E-MapReduce clusters support execution plan authentication. You can authorize Alibaba Cloud Resource Access Management (RAM) user accounts to access execution plans using the master account.

Master account access

After logging on to the E-MapReduce console with the master account, you can run execution plans on the Execution plan page. Submit jobs to the security cluster for execution and access the related open source services involved in the jobs using the Hadoop user name.

RAM user account access

After logging on to the E-MapReduce console with the RAM user account, you can run execution plans on the Execution plan page. Submit jobs to the security cluster for execution and access the related open-source component services involved in the jobs using the user name of the RAM user account.

Examples

- The master account administrator can create multiple RAM user accounts as required and grant them permissions from the RAM console. The RAM users can then log on to the E-MapReduce console and use the related functions.
- The master account administrator provides RAM user accounts to developers.
- After creating jobs and execution plans, developers start running them to submit jobs to the cluster. They can then access the relevant component services in the cluster using the user names that correspond to the RAM user accounts.



Note:

Periodic execution plans are currently uniformly executed using the Hadoop account.

- Relevant permission control for component services, such as whether account A is permitted to access a file in HDFS, is performed using the user name of a RAM user.

13.6 Cross-region access

Kerberos in E-MapReduce supports cross-region access, meaning that different Kerberos clusters can access each other. This section describes cross-region access using cluster A and cluster B as an example.

- Hostname of emr-header-1 in cluster A → emr-header-1.cluster-1234. Region → EMR.1234.COM
- Hostname of emr-header-1 in cluster B → emr-header-1.cluster-6789. Region → EMR.6789.COM



Note:

- The hostname can be obtained by executing the hostname command on emr-header-1.
- The region can be obtained in /etc/krb5.conf on emr-header-1.

Add principal

The emr-header-1 nodes in cluster A and cluster B both run the following command:

```
# root account
sh /usr/lib/has-current/bin/hadmin-local.sh
/etc/ecm/has-conf-k/etc/ecm/has-conf/admin.
keytab
HadminLocalTool.local: addprinc -pw 123456 krbtgt
/EMR.6789.COM@EMR.1234.COM 6789.COM@EMR.1234.
Com
```



Note:

- The password is 123456. This can be modified.
- The region of cluster B is EMR.6789.COM. This is the region of the cluster being accessed.
- The region of cluster A is EMR.1234.COM. This is the region of the cluster that initiates access.

Configure /etc/krb5.conf for cluster A

Configure [regions]/[domain_region]/[capaths] on cluster A as follows:

```
[ libdefaults ]
    kdc_realm = EMR . 1234 . COM
    default_realm = EMR . 1234 . COM
    udp_prefer_ence_limit = 4096
    kdc_tcp_port = 88
    kdc_udp_port = 88
    dns_lookup_kdc = false
[ realms ]
    EMR . 1234 . COM = {
        kdc = 10 . 81 . 49 . 3 : 88
    }
    EMR . 6789 . COM = {
        kdc = 10 . 81 . 49 . 7 : 88
    }
[ domain_realm ]
    . cluster - 1234 = EMR . 1234 . COM
    . cluster - 6789 = EMR . 6789 . COM
[ capaths ]
    EMR . 1234 . COM = {
        EMR . 6789 . COM = .
    }
    EMR . 6789 . COM = {
        EMR . 1234 . COM = .
    }
```

Synchronize /etc/krb5.conf to all cluster A nodes.

Copy the binding information (only the long domain name emr-xxx-x.cluster-xxx is required) from cluster B's /etc/hosts file to /etc/hosts for all cluster A nodes.

```
10 . 81 . 45 . 89    emr - worker - 1 . cluster - xxx
10 . 81 . 46 . 222  emr - worker - 2 . cluster - xx
10 . 81 . 44 . 177  emr - header - 1 . cluster - xxx
```



Note:

- If you want to run a job on cluster A to access cluster B, you must first restart YARN.
- Configure cluster B's host binding information for all cluster A nodes.

Access services in cluster B

You can use cluster A's Kerberos keytab file /ticket as a cache on cluster A to access services in cluster B.

For example, access the HDFS service in cluster B as follows:

```
su    has ;
hadoop fs - ls hdfs :// emr - header - 1 . cluster - 6789 : 9000
/
```

```
Found 4 items
-rw-r----- 2 hadoop 34 2017-12-05
18:15 hdfs://emr-header-1.cluster-6789:9000/abc
drwxrwxrwt - hadoop hadoop 0 2017-12-05 18
:32 hdfs://emr-header-1.cluster-6789:9000/spark-
history
drwxrwxrwt - hadoop hadoop 0 2017-12-05 17:
53 hdfs://emr-header-1.cluster-6789:9000/tmp
drwxrwxrwt - hadoop hadoop 0 2017-12-05 18:
24 hdfs://emr-header-1.cluster-6789:9000/user
```