

Alibaba Cloud E-MapReduce

User Guide

Issue: 20181214

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.	 Note: 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 a optional value, and only one item can be selected.	<code>ipconfig [-all -t]</code>
{ } or {a b}	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 Role authorization	1
2 Configure clusters	4
2.1 Instance types.....	4
2.2 Gateway clusters.....	5
2.3 ECS instances.....	5
2.4 Storage guide.....	6
2.5 D1 series.....	8
2.6 Enable access between classic networks and VPCs.....	9
3 Clusters	10
3.1 Create a cluster.....	10
3.2 Expand a cluster.....	15
3.3 Cluster list.....	17
3.4 Release a cluster.....	18
3.5 Cluster details.....	19
3.6 User management.....	22
3.7 Service list.....	23
3.8 Cluster scripts.....	24
3.9 Cluster renewal.....	26
3.10 Security groups.....	27
3.11 Node upgrade.....	28
3.12 Disk expansion.....	29
3.13 Convert your billing method.....	31
3.14 Remove abnormal nodes.....	31
4 Jobs	33
4.1 Configure a Hadoop MapReduce job.....	33
4.2 Configure a Hive job.....	34
4.3 Configure a Pig job.....	36
4.4 Configure a Spark job.....	38
4.5 Configure a Spark SQL.....	40
4.6 Configure a Shell job.....	41
4.7 Configure a Sqoop job.....	42
4.8 Job operations.....	42
4.9 Time and date variables.....	43
5 Execution plans	45
5.1 Create an execution plan.....	45
5.2 Manage an execution plan.....	47
5.3 Execution plan list.....	48

5.4 View job results and logs.....	49
5.5 Parallel execution of multiple execution plans.....	51
6 Alarm management.....	52
6.1 Cluster alarm management.....	52
7 Software configuration.....	55
8 Bootstrap actions.....	60
9 VPC.....	63
10 Python instructions.....	65
11 Open source components.....	66
11.1 Hue.....	66
11.2 Oozie.....	67
11.3 Presto.....	70
11.4 Zeppelin.....	71
11.5 ZooKeeper.....	71
11.6 Kafka.....	72
11.6.1 Quick start.....	72
11.6.2 Cross-cluster Kafka access.....	74
11.6.3 Kafka Ranger.....	77
11.6.4 Kafka SSL.....	80
11.6.5 Kafka Manager.....	82
11.6.6 Common Kafka issues.....	83
11.7 Druid.....	84
11.7.1 Introduction to Druid.....	84
11.7.2 Quick start.....	87
11.7.3 Ingestion Spec.....	97
11.7.4 Tranquility.....	101
11.7.5 Kafka indexing service.....	104
11.7.6 Superset.....	107
11.7.7 Druid common problems.....	117
11.8 Presto.....	120
11.8.1 Connector.....	120
11.8.2 What is Presto.....	128
11.8.3 Quick start with Presto.....	130
11.8.4 Data type.....	137
11.8.5 Common functions and operators.....	141
11.8.6 SQL statement.....	166
11.8.7 Technical support.....	201
11.9 TensorFlow instructions.....	201
11.10 Knox guide.....	204
11.11 Instructions for using Flume.....	207
12 OSS ACL.....	216

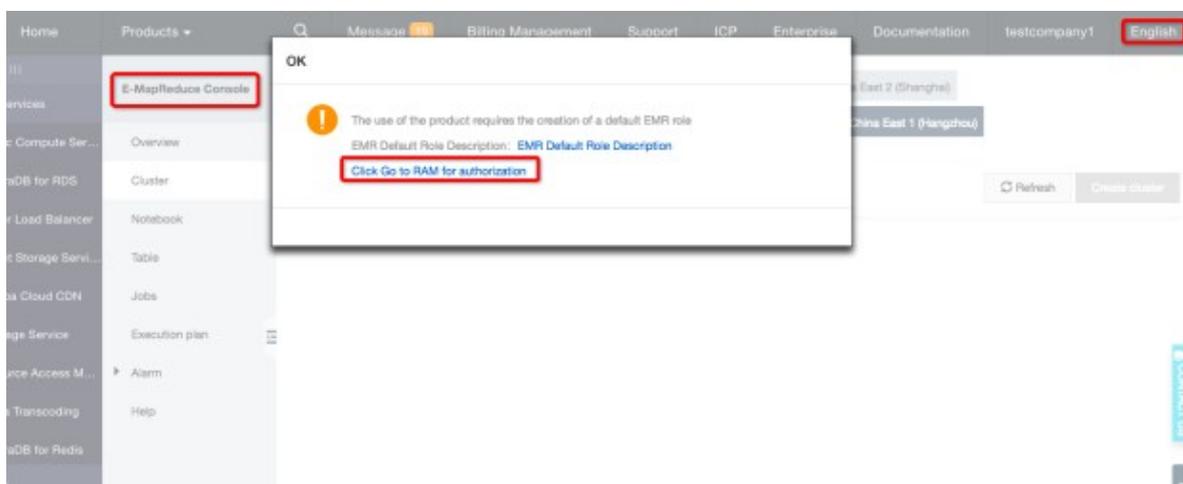
13 Connect to clusters using SSH.....	219
14 Create your gateway.....	224
15 MetaService.....	230
16 Notebook.....	233
16.1 Introduction.....	233
16.2 Manual.....	239
16.3 Examples.....	247
16.3.1 Example of bank employee information inquiry.....	247
16.3.2 Video playback data example.....	248
17 Table mangement.....	251
18 Kerberos authentication.....	256
18.1 Introduction to Kerberos.....	256
18.2 Authentication method compatible with MIT Kerberos.....	261
18.3 Ram certification.....	265
18.4 LDAP authentication.....	267
18.5 Execution plan authentication.....	269
18.6 Cross-region access.....	270
19 Component authorization.....	273
19.1 HDFS authorization.....	273
19.2 YARN authorization.....	275
19.3 Hive authorization.....	280
19.4 HBase authorization.....	285
19.5 Kafka authorization.....	288
19.6 RANGER.....	293
19.6.1 Ranger introduction.....	293
19.6.2 HDFS configuration.....	295
19.6.3 Hive configuration.....	300
20 Disaster recovery.....	306
20.1 EMR cluster disaster tolerance.....	306
21 Resource pool.....	307
22 Auto-scaling.....	310
22.1 Auto-scaling introduction.....	310
22.2 Configure auto-scaling according to time.....	310
22.3 Auto-scaling preemptible instances.....	313
22.4 Auto-scaling records.....	314
23 Workflow development.....	315
23.1 Workflow project management.....	315
23.2 Job editing.....	317
23.3 Ad hoc queries.....	319
23.4 Workflow management.....	321

1 Role authorization

When a user activates the E-MapReduce service, a default system role named AliyunEMRDefaultRole must be granted to the E-MapReduce service account. Once assigned, E-MapReduce can call the relevant services (such as ECS and OSS), create clusters, save logs, and perform other related tasks.

Role authorization process

1. When you create a cluster or an on-demand execution plan, if a default role is not authorized to the E-MapReduce service account, the following prompt is displayed. Click **Go to RAM for authorization** to authorize the role.



2. On the RAM authorization page, click **Confirm Authorization Policy** to authorize the default role AliyunE-MapReduceDefaultRole to the E-MapReduce service account.

Cloud Resource Access Authorization

Note: If you need to modify role permissions, please go to the RAM Console. [Role Management](#). If you do not configure it correctly, the following role: E-MapReduce will not be able to obtain the required permissions. ✕

E-MapReduce needs your permission to access your cloud resources.

The system has created roles for the following user. These roles can be found below. E-MapReduce. After authorization, E-MapReduce will have access to your cloud resources.

AliyunEMRDefaultRole ✓

Description: The EMR service will use this role to access ECS resources.

Permission Description: The policy for AliyunEMRDefaultRole, including the permission for ECS, VPC and OSS.

Confirm Authorization Policy
Cancel

- Refresh the E-MapReduce console and perform the relevant operations. If you want to view detailed policy information about AliyunE-MapReduceDefaultRole, log on to the RAM console, or click [View Link](#).

Default role permissions

The default role AliyunEMRDefaultRole has the following permissions:

- ECS-related permissions:

Permission Name (Action)	Description
ecs: CreateInstance	Create ECS instances.
ecs: RenewInstance	Renew ECS instances.
ecs: DescribeRegions	Query ECS region information.
ecs: DescribeZones	Query Zone information.
ecs: DescribeImages	Query image information.
ecs: CreateSecurityGroup	Create security groups.
ecs: AllocatePublicIpAddress	Allocate a public network IP address.
ecs: DeleteInstance	Delete machine instances.
ecs: StartInstance	Start machine instances.

Permission Name (Action)	Description
ecs: StopInstance	Stop machine instances.
ecs: DescribeInstances	Query machine instances.
ecs: DescribeDisks	Query the machine's relevant disk information.
ecs: AuthorizeSecurityGroup	Set security group input rules.
ecs: AuthorizeSecurityGroupEgress	Set security group output rules.
ecs: DescribeSecurityGroupAttribute	Query security group details.
ecs: DescribeSecurityGroups	Query security group list information.

- OSS-related permissions

Permission Name (Action)	Description
oss: PutObject	Upload file or folder objects.
oss: GetObject	Get file or folder objects.
oss: ListObjects	Query file list information.

2 Configure clusters

2.1 Instance types

There are three types of node instances in an E-MapReduce cluster: master, core, and task. Different service processes are deployed on each instance type. For example, with Hadoop, the HDFS NameNode and YARN ResourceManager services are deployed on master instances, while the HDFS DataNode and YARN NodeManager services are deployed on core instances. For task instances, because they are only used in computing tasks, only YARN NodeManager is deployed, and not HDFS-related services.

When you create a cluster, you must determine the ECS specifications for each instance type. ECS instances of the same type must be in the same instance group. If you increase the number of hosts in a core or task instance group, you can scale the cluster up at a later date. This does not apply to master instance groups.

**Note:**

Task instances are supported in version 3.2.0 or later.

Master instance

The master instance is where the management and control components of the cluster service are deployed. You can connect to the master instance using SSH and check service statuses in the cluster through the software's Web UI.

If you want to perform a test or run a job, log on to the master instance and submit jobs directly at the command line. By default, only one master instance is used. However, if the cluster's high availability feature is enabled, two are used.

Core instance

Core instances, which are managed by master instances, store all of the data in the cluster using a variety of different storage media. They also deploy computing services to perform computing tasks. If you need more data storage or are experiencing heavier workloads, you can scale core instances up at any time without impacting the operations of the cluster.

Task instance

Task instances are responsible for computing and can quickly add computing power to a cluster. They can also scale up and down at any time without impacting the operations of the cluster.

However, this instance type is optional, and if the core instance has enough computing power, task instances are not necessary. Depending on the fault tolerance (or retries) of the computing service, a reduction in the number of task instance nodes may cause MapReduce and Spark jobs to fail.

2.2 Gateway clusters

A gateway cluster is an independent cluster that consists of multiple nodes of the same configuration.

When you create a gateway cluster, you can associate it with an existing Hadoop cluster. To facilitate cluster operations, it is recommended that you associate it with a cluster on which Hadoop (HDFS and YARN), Hive, Spark, Sqoop, Pig, or other clients have been deployed. It is an independent submission point and does not use up cluster resources, especially when you submit jobs on the Master node. This improves the stability of the Master node. If there are too many jobs to submit, you can add nodes as and when you need them.

You can also create multiple gateway clusters for different users, allowing them to use their own environment to meet different service requirements.

2.3 ECS instances

This section contains information on the different ECS instance types.

ECS instance types supported by E-MapReduce

- General-purpose

This type uses cloud disks as storage. The ratio between vCPUs and memory is 1:4. For example, 32 cores and 128 GB memory.

- Compute

This type uses cloud disks as storage and provides more computing resources. The ratio between vCPUs and memory is 1:2. For example, 32 cores and 64 GB memory.

- Memory

This type uses cloud disks as storage and provides more memory resources. The ratio between vCPUs and memory is 1:8. For example, 32 cores and 256 GB memory.

- Big data

This type uses local SATA disks as storage, which is highly cost-effective. If you want to store massive amounts of data (TB-level), it is recommended that you use this type.

- Ephemeral SSD

This type uses ephemeral SSDs as storage, which provides high local IOPS and throughput.

- Shared (entry level)

This type shares CPUs and is not stable enough for most scenarios. It is applicable for entry-level users, not enterprise customers.

- GPU

This type is a heterogeneous GPU-based model applicable in machine learning scenarios.

ECS instance types applicable in different scenarios

- Master instances

General-purpose and memory types are applicable in master instances, where data is directly stored on Alibaba Cloud's cloud disks. There are also three backups to guarantee high data reliability.

- Core instances

General-purpose, compute, and memory types are applicable for small data volumes (not TB-level) or when OSS is used as the primary data storage. When the amount of data is large (10 TB or more), it is recommended that you use the big data type, which is more cost-effective. Using an ephemeral disk makes it harder to ensure data reliability, but this can be maintained and guaranteed by the E-MapReduce platform.

- Task instances

All types except the big data type are suitable for task instances to give additional computing power to the cluster. Currently, the ephemeral SSD type is not supported, but will be added soon.

2.4 Storage guide

There are two types of disks on a node: the system disk, which is used to install operating systems, and the data disk, which is used to store data. A node typically has one system disk by default, which must be a cloud disk. However, you can have more than one data disk (currently, up to sixteen on a single node). Each data disk can have different configurations, including having a different type or capacity. In E-MapReduce, a cluster's system disks are SSD cloud disks by default, and four are used by default. Considering current intranet bandwidth, this default configuration of four cloud disks is sufficient.

Cloud and ephemeral disks

Two types of disk are available for data storage.

- Cloud disks

Includes SSD, ultra, and basic cloud disks.

Cloud disks are not attached directly to the local computing node. Instead, they access a remote storage node through the network. Each piece of data has two real-time backups at the backend, meaning that there are three identical copies in total. When one is corrupted (due to disk damage), a backup is used automatically for recovery.

- Ephemeral disks

Includes ephemeral SATA disks in the big data type and ephemeral SSD disks used in the ephemeral SSD type.

Ephemeral disks are attached directly to the computing node and have a better performance than cloud disks. You cannot change the number of ephemeral disks. As with offline physical hosts, there is no data backup at the backend, meaning that upper-layer software is required to guarantee data reliability.

Usage scenarios

In E-MapReduce, when the hosting node is released, all of the data in the cloud and ephemeral disks is cleared. The disks can also not be kept independently and used again. Hadoop HDFS uses all data disks for data storage. Hadoop YARN uses all data disks as on-demand data storage for computing.

If you do not have massive amounts of data (below TB-level), you can use cloud disks, as the IOPS and throughput are smaller than local disks. In the event that you have large amounts of data, it is recommend that you use local disks whose data reliability is guaranteed by E-MapReduce. If you find the throughput to be insufficient, switch to ephemeral disks.

OSS

OSS can be used as HDFS in E-MapReduce, and you can have easy read and write access to OSS. All code that uses HDFS can also be easily modified to access data on OSS. Below you can find a number of examples:

Reading data from Spark

```
sc.Textfile("hdfs://user/path")
```

Changing the storage type from HDFS to OSS

```
sc.Textfile("oss://user/path")
```

This is the same for Map Reduce and Hive jobs.

HDFS commands process OSS data directly:

```
hadoop fs -ls oss://bucket/path  
hadoop fs -cp hdfs://user/path oss://bucket/path
```

In this process, you do not need to enter the AK or endpoint. E-MapReduce automatically completes your information using the current cluster owner.

However, as OSS does not have high IOPS, it is not suitable for usage scenarios that require high IOPS, such as Spark Streaming or HBase.

2.5 D1 series

To meet the demand for storage in big data scenarios, Alibaba Cloud has launched the D1 series on the cloud. Instead of using cloud disks in its data storage, the D1 series uses ephemeral disks, which solves the problem of high costs caused by keeping multiple copies of redundant data in cloud disks. Data also no longer needs to be transferred over the network, which improves disk throughput. Furthermore, with the D1 series, you can also take advantage of Hadoop's proximity computing.

Compared with cloud disks, the series greatly enhances storage performance while reducing prices. Indeed, the cost is almost the same as offline physical hosts.

Despite their advantages, however, ephemeral disks still cannot ensure data reliability, and upper-layer software is required to guarantee it. If a disk or node fails, operations and maintenance must be performed manually. Cloud disks, meanwhile, guarantee data reliability automatically, meaning that you do not need to worry about disk damage. Alibaba Cloud's default multi-disk backup policy is also helpful in this regard.

E-MapReduce + D1 solution

A complete set of automated O&M solutions, such as the D1 series, is now available for ephemeral disks in E-MapReduce. This allows Alibaba Cloud users to use ephemeral disks

conveniently and reliably, without having to worry about the entire O&M process. Data reliability and service availability are guaranteed.

The main advantages are as follows:

- Highly reliable distribution of required nodes
- Ephemeral disk and node fault monitoring
- Automatic determination of data migration opportunities
- Automatic failed node migration and data balancing
- Automatic HDFS data detection
- Network topology optimization

With the automated O&M of the entire back-end management and control system, E-MapReduce helps you make better use of ephemeral disks and develop a cost-effective big data system.

**Note:**

If you want to set up a Hadoop cluster using the D1 series, submit a ticket. We will then be able to assist you in your operations.

2.6 Enable access between classic networks and VPCs

Alibaba Cloud currently provides two types of cloud network: classic and VPC. While some users still use classic networks, E-MapReduce clusters use VPCs. This section describes how to enable inter-access between ECS on classic networks and E-MapReduce clusters on VPC networks.

ClassicLink

To grant access between ECS on a classic network and an E-MapReduce cluster on a VPC network, Alibaba Cloud launches [ClassicLink](#). Follow these steps:

1. Create a vSwitch according to the CIDR block specified in ClassicLink.
2. To deploy a cluster that you have created, use the vSwitch of the CIDR block.
3. Connect the corresponding classic network node to the VPC in the ECS console.
4. Set the security group rules.

3 Clusters

3.1 Create a cluster

In this tutorial, you will learn how to create an Alibaba Cloud E-MapReduce (EMR) cluster.

Go to the EMR cluster creation page

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. Complete RAM authorization. For details, see [Role authorization](#).
3. Select a region for the cluster. The region cannot be changed once the cluster is created.
4. Click **Create Cluster** to go to the cluster creation page.

Create a cluster

**Note:**

After you create an EMR cluster, the only thing that can be changed is its name.

To create a cluster, follow these three steps:

1. Configure the software.
 - **EMR version:** The main version of E-MapReduce represents a complete open source software environment and can be upgraded regularly based on upgrades made to the internal component software. If the software related to Hadoop is upgraded, the main version of E-MapReduce is also upgraded. Clusters from an earlier version cannot be upgraded to a later version.
 - **Cluster type:** Currently, E-MapReduce provides four cluster types.
 - Hadoop clusters, which provide the following semi-managed ecosystem components:
 - Hadoop, Hive, and Spark for large-scale offline distributed data storage and computing.
 - Spark Streaming, Flink, and Storm for stream processing.
 - Presto and Impala for running interactive analytics.
 - Oozie and Pig.
 - Druid clusters, which provide semi-managed, real-time interactive analysis services, query large amounts of data at millisecond latency, and support multiple data intake

methods. When used with services such as EMR Hadoop, EMR Spark, OSS, and RDS, Druid clusters offer real-time query solutions.

- Data Science clusters, which are mainly applicable in big data and AI scenarios, providing Hive and Spark offline big data, and TensorFlow model training.
- Kafka clusters, which are semi-managed distributed message systems that feature high throughput and high scalability, providing a complete service monitoring system that can keep a stable running environment.
- **Required Services:** Displays a list of all software components under the selected cluster type, including their name and version number.
- **Optional Services:** You can select different components as required. The selected components start relevant service processes by default.

**Note:**

The more components you select, the higher the requirements are for configuration, as there may be insufficient resources to run these services.

- **High security mode:** In this mode, you can set the cluster's Kerberos authentication. This feature is unnecessary for clusters used by individual users and is turned off by default.
- **Enable custom setting:** Before you start a cluster, you can specify a JSON file to change the software configuration.

2. Configure the hardware.

- Billing method
 - Like with ECS, both Subscription and Pay-As-You-Go modes are supported. If you select Subscription mode, you must also select the duration. You can select 1, 2, 3, 6, or 9 months, or 1, 2, or 3 years. This mode is applicable to short-term testing or flexible dynamic tasks, but is relatively expensive.
- Cluster network configuration
 - **Zone:** Select the zone where the cluster is to be located. If better network connectivity is required, we recommend selecting the same availability zone. However, this increases the risk of failure when creating a cluster, as the availability zone's storage may be insufficient. If you need a large number of nodes, please submit a ticket.
 - **Network type:** The Virtual Private Cloud (VPC) network is selected by default, which requires you to enter a VPC and a VSwitch. If you have not created a network, go to the [VPC console](#) to create one. For more information about E-MapReduce VPC, see [VPC](#).

- **VPC**: Select the region of the VPC network.
- **VSwitch**: Select a zone for VSwitch under the corresponding VPC. If no VSwitch is available in this zone, you must create a new one.
- **Security group name**: A security group does not typically exist when you first create a cluster. To create a new security group, enter a name. If you already have a security group, you can select it here.
- Cluster configuration
 - **High availability**: When enabled, two master instances in the Hadoop cluster are used to ensure the availability of the Resource Manager and Name Node. HBase clusters support high availability by default.
 - **Node type**: The three types of node supported are as follows:
 - Master, which is mainly responsible for the deployment of control processes such as Resource Manager and Name Node.
 - Core, which is mainly responsible for the storage of all data in the cluster, and can be scaled up as required.
 - Task, which is the node used for computing. It does not store data and is used to adjust the computing capacity of the cluster.
 - **Node configuration**: Select different node types. Different types of nodes have different application scenarios. You can select a type based on your requirements.
 - **Data disk type**: The data disks used by a cluster node are either standard cloud disks, high-efficiency cloud disks, or SSD cloud disks. This varies between machine type and region. When the user selects different regions, disks that are supported by those regions are displayed in the drop-down list. By default, data disks are released when the cluster is released. The ephemeral disk type is set by default and cannot be changed.
 - **Data disk volume**: The recommended minimum cluster volume for a single machine is 40 G, and the maximum is 8000 G. The capacity of the ephemeral disk is set by default and cannot be changed.
 - **Instance quantity**: This indicates the number of instances of all required nodes. A cluster requires at least three instances. However, high availability clusters require at least four, and therefore add one master node. The maximum is 50. If more than 50 instances are required, please submit a ticket. A monthly subscribed cluster can provide 100 at most. If you need more than 50 nodes, please submit a ticket.

3. Configure the basic information.

- Basic information
 - Cluster name: The cluster name can contain Chinese characters, English letters (uppercase and lowercase), numbers, hyphens (-), and underscores (_), with a length of between 1-64 characters.
- Running logs
 - **Running logs**: The function for saving running logs is turned on by default. In the default state, you can select the OSS directory as a location to save running logs, but you must have activated OSS before using this function. The cost depends on the number of uploaded files. We recommend that you open the OSS log saving function, which helps in debugging and error screening.
 - **Log path**: OSS path for saving logs.
 - **Uniform Meta Database**: This is provided by E-MapReduce to store all Hive metadata in the external database of the cluster. We recommend that you use this function when the cluster uses OSS as the main storage.
- Permission settings
 - **EMR role**: This role authorizes E-MapReduce to use other Alibaba Cloud services, such as ECS and OSS.
 - **ECS role**: This role allows your programs running on the E-MapReduce computing nodes to access cloud services like OSS without providing the Alibaba Cloud AccessKey. E-MapReduce automatically applies for an on-demand AccessKey to authorize access. The AccessKey permission is controlled by this role.
- Logon settings
 - **Remote logon**: It is turned on by default to enable security group port 22.
 - **Logon password**: Set the logon password at the master node. The logon password must contain English letters (both uppercase and lowercase letters), numbers, and special characters (!@#%\$%^&*) with a length of between 8-30 characters.
- (Optional) Bootstrap operation: Before Hadoop is enabled in the cluster, you can run the customized script. For more information, see [Bootstrap action](#).

Purchase lists and cluster costs

The cost of the cluster is displayed in the **Configuration List** pane. The price varies with the type of payment. For Subscription clusters, the total expense is shown. For Pay-As-You-Go clusters, the hourly cost is shown.

Confirm creation

Once you have entered all the necessary information, the **Create** button is highlighted. Click **Create** to create the cluster.



Note:

- If your cluster is Pay-As-You-Go, it is created immediately, and you are taken back to the **Overview** page. Here, your cluster is displayed with the status **Initializing**. It can take several minutes to finish creating the cluster. After the cluster is created, its status is switched to **Idle**.
- Subscription clusters are not created until the order is generated and paid.

Log on to the core node

To log on to the core node, perform the following steps:

1. Switch to the Hadoop account on the Master node.

```
su hadoop
```

2. Log on to the core node through SSH without a key.

```
ssh emr-worker-1
```

3. Get root permissions through the sudo command.

```
sudo vi /etc/hosts
```

Failure during cluster creation

If a cluster fails to be created, the message **Cluster creation failed** is displayed on the cluster list page. If you hover your cursor over the red exclamation point, the reason for the failure is displayed.

You do not need to perform any additional operations because the corresponding computing resources are not created. The cluster is automatically hidden after three days.

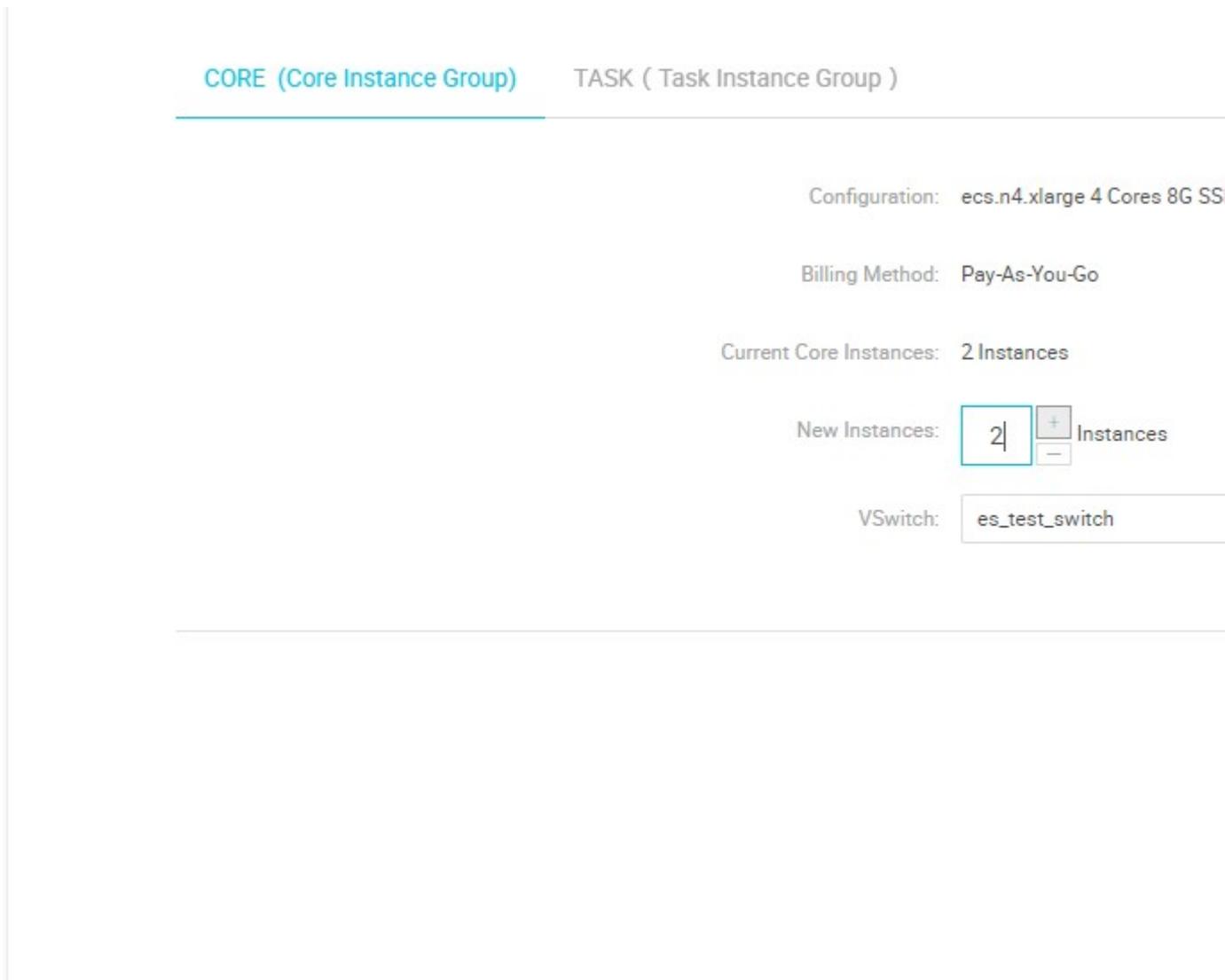
3.2 Expand a cluster

If your cluster does not have enough resources, you can expand it horizontally. Only core and task nodes can be expanded. When expanding a cluster, configurations default to be consistent with the ECS instance purchased previously.

Enter expansion interface

Select the cluster you want to expand from the list of clusters, click **More**, and select **Scale Up/Out**. You can also click **View Details** to the right of the cluster, and click **Scale Up/Out** in the upper right corner.

Expansion interface



Note:

Although expansion is supported, reduction is not.

- **Configuration:** Displays the configurations of the current instance.
- **Billing Method:** Displays the payment method of the current cluster.
- **Current Core Instances:** Displays the total number of your current core nodes.
- **New Instances:** Enter the quantity of instances that you want to add. We recommend increasing your cluster gradually by one or two instances at a time.
- **VSwitch:** Displays the VSwitch of the current cluster.

Expansion status

In the following figure, the statuses of the cluster expansions are shown.

ECS ID	Status	Public IP	Intranet IP	Created At
i-bp1htgltshh2o59kkxqz	● Normal		192.168.0.47	2018-10-25 10:36:59
i-bp1733wjv9dzvvp62n7	● Normal		192.168.0.48	2018-10-25 10:37:00
i-bp1dhrigana54rgvcfn2	● Scaling Up/Out		192.168.0.50	2018-10-25 10:53:07
i-bp1c0d1hnoaicxddd57t	● Scaling Up/Out		192.168.0.51	2018-10-25 10:53:09

To view the expansion status of a cluster, click **Core Instance Group (CORE)** in the **Cluster Overview** panel. Nodes that are being expanded are displayed as **Scaling Up/Out**. When the status of an ECS instance changes to **Normal**, ECS has been added to the cluster and can now provide services.

Change password

After you expand a cluster, you can log on to the expanded node with SSH and change your root password. To do so, follow these steps:

1. Log on the master host with SHH using the following command and obtain the public IP of the master cluster in the [Cluster Overview](#) panel.

```
ssh root@ip.of.master
```

2. Switch to the hadoop user.

```
su hadoop
```

3. Log on to the expanded cluster and obtain its intranet IP in the [Cluster Overview](#) panel.

```
ssh ip.of.worker
```

4. Change your root password using the following command.

```
sudo passwd root
```

3.3 Cluster list

The **Cluster Management** page, shown in the following figure, displays basic information about all of your clusters.

Cluster ID/Name	Cluster Type	Status	Created At	Runtime	Billing Method	Actions
C-2A5D635C8D869A92 dplust_docs	HADOOP	Idle	2018-10-29 09:41:45	4 Hours 16 Minutes 21 Seconds	Pay-As-You-Go	Monitoring Manage View Details More
C-79A6888BC83FE31F daaf a	HADOOP	Idle	2018-10-24 14:47:08	4 Days 23 Hours 10 Minutes 58 Seconds	Subscription Expiration Date 2018-11-25 00:00:00	Monitoring Manage View Details Renew More

The items in the cluster list are as follows:

- **Cluster ID/Name:** The ID and name of a cluster. Move your cursor over a cluster's name to modify it.
- **Cluster Type:** Hadoop is the only cluster type available.
- **Status:** The status of a cluster. For more information, see [Cluster statuses](#). If a cluster experiences an abnormality, such as a creation failure, prompt information appears on the right. If you hover your cursor over it, you can view detailed error information. You can also sort the statuses by clicking **Status**.
- **Created At:** Time at which a cluster was created.
- **Runtime:** The time from the point of creation to the current time. Once the cluster is released, the timing is terminated.
- **Billing Method:** The billing method of the cluster.
- **Actions:** Operations that can be performed on clusters, including the following:

- **Monitoring:** Monitors the CPU usage rate, memory capacity, and disk capacity of E-MapReduce clusters to help users monitor the running status of the cluster.
- **Manage:** Enters the **Clusters and Services** panel.
- **View details:** Enters the **Cluster Overview** panel and view detailed information after the cluster is created.
- **More:**
 - **Scale Up/Out:** Expands the cluster.
 - **Release:** Releases a cluster. For more information, see [Release a cluster](#).
 - **Restart:** Restarts a cluster.

3.4 Release a cluster

Pay-As-You-Go clusters can be released on the **Cluster Management** page. Only those with the following statuses can be released:

- Creating
- Running
- Idle

Common release

Before releasing a cluster, you are prompted to confirm the operation. Once the release is confirmed, the following occurs:

- All jobs in the cluster are forcibly terminated.
- If you have selected to save logs to OSS, all current job logs are saved to OSS. This may take several minutes.
- Clusters are released. Depending on their size, this may take seconds or minutes. ECS clusters are billed before they are released.



Warning:

To save money, make sure to release a cluster before a new billable hour starts.

Forcible release

If you no longer need logs and want to immediately terminate running clusters, perform a forcible release. Logs are not collected and clusters are released directly.

Cluster release failure

Due to system errors or other causes, clusters may fail to be released. If a failure occurs, E-MapReduce starts background protection until the cluster is finally released.

3.5 Cluster details

Cluster details display detailed information about a cluster.

Detailed information is provided in the following areas: cluster, software, network, and host.

Cluster

Cluster		
Name: dtplus_docs ID: C-47B6CF7F712EFD7E Region: cn-hangzhou Start Time: 2018-10-25 10:36:51	Software Configuration: I/O Optimization: Yes High Availability: No Security Mode: Standard	Billing Method: Pay-As-You-Go Current Status: Idle Runtime: 3 Hours27 Minutes8 Seconds

- **Name:** The name of a cluster.
- **ID:** The instance ID of a cluster.
- **Region:** The region where a cluster is located.
- **Start Time:** The time at which a cluster is created.
- **Software Configuration:** Software configurations.
- **I/O Optimization:** Whether the I/O optimization setting is enabled.
- **High Availability:** Whether high-availability clusters are enabled.
- **Security Mode:** Software in clusters is started in Kerberos secure mode. For more information about Kerberos, see [Introduction to Kerberos](#).
- **Billing Method:** Cluster billing method.
- **Current Status:** For more information, see [Cluster statuses](#).
- **Runtime:** The time from the point of creation to the current time.
- **Bootstrap:** The names, paths, and parameters of all configured bootstrap actions are listed here.
- **ECS Role:** When your program runs on an E-MapReduce compute node, you can access the related Alibaba Cloud services, such as OSS, without an AccessKey. E-MapReduce automatically requests a temporary AccessKey to authorize this access. The permission control of this temporary AccessKey is controlled by this role.

Software

Software
EMR Version: EMR-3.13.0 Cluster Type: HADOOP Software: HDFS2.7.2 / YARN2.7.2 / Hive2.3.3 / Ganglia3.7.2 / Spark2.3.1 / HUE4.1.0 / Tez0.9.1 / Sqoop1.4.7 / Pig0.14.0 / ApacheDS2.0.0 / Knox0.13.0

- **Main Version:** The main version of E-MapReduce.
- **Cluster Type:** The selected cluster type.
- **Software:** All application programs installed are listed here with their versions, such as HDFS2.7.2, Hive 2.3.3, or Spark 2.3.1.

Network

Network
Region ID: cn-hangzhou-f Network Type: vpc Security Group ID: sg-bp16guz3xdbwj3q553qw8 VPC/VSwitch: vpc-bp16guz3xdbwj3q553qw8 / vsw-bp1rb1x1lvutlj6wbgunr

- **Region ID:** The region where a cluster is located, such as cn-hangzhou-b, which is the same as ECS.
- **Network Type:** The network type of a cluster.
- **Security Group ID:** The ID of the security group that a cluster joined.
- **VPC/VSwitch:** The VPC and VSwitch IDs of a cluster.

Host

- **Master Instance Group (Master):** Configurations of all master nodes.

Master Instance Group(MASTER)	Pay-As-You-Go
Hosts: 1	CPU: 4 Cores
Memory: 8GB	
Data Disk Type: SSD Disk80GB*1 Disks	

- **Hosts:** The number of current nodes. During the creation process, the number of current nodes is less than the number of nodes you applied for until the creation is complete.

- **CPU**: The number of cores in a node's CPU.
- **Memory**: Memory capacity of a node.
- **Data Disk Type**: Data disk type and capacity of a node.
- **ECS ID**: The ID of the ECS instances purchased.

Master Instance Group 

ECS ID	Status	Public IP	Intranet IP	Created At
i-ho1-egp49h9k9k9gvvz0	● Normal	47.98.199.51	192.168.0.45	2018-10-25 10:36:58

- **Status**: Includes Creating, Normal, Expanding, and Released.
- **Public IP** : The public IP of master nodes.
- **Intranet IP**: The internal network IP of the machine that can be accessed by all nodes in the cluster.
- **Created At**: The creation time of the ECS instance purchased.
- **Core Instance Group (Core)** : Configurations of all core nodes.

Core Instance Group(CORE)	Pay-As-You-Go
Hosts: 4	CPU: 4 Cores
Memory: 8GB	
Data Disk Type: SSD Disk80GB*4 Disks	

- **Hosts**: The number of current nodes. This is the same as the number of the nodes you applied for.
- **CPU**: The number of cores in a node's CPU.
- **Memory**: Memory capacity of a node.
- **Data Disk Type**: Data disk type and capacity of a node.
- **ECS ID**: The ID of the ECS instances purchased.

Core Instance Group 

ECS ID	Status	Public IP	Intranet IP	Created At
i-ho1-egp49h9k9k9gvvz0	● Normal		192.168.0.45	2018-10-25 10:36:59
i-ho1-egp49h9k9k9gvvz1	● Normal		192.168.0.46	2018-10-25 10:37:00
i-ho1-egp49h9k9k9gvvz2	● Normal		192.168.0.47	2018-10-25 10:53:07
i-ho1-egp49h9k9k9gvvz3	● Normal		192.168.0.48	2018-10-25 10:53:09

- **Status**: Includes Creating, Normal, Expanding, and Released.

- **Intranet IP:** The internal network IP of the machine that can be accessed by all nodes in the cluster.
- **Created At:** The creation time of the ECS instance purchased.

3.6 User management

User management allows you to manage the accounts required to create services on specified clusters. E-MapReduce currently supports the creation of two types of accounts: Knox and Kerberos.

Create a RAM account

1. Log on to the [Alibaba Cloud E-MapReduce console](#) and go to the **Cluster Management** page.
2. Click **Manage** on the right side of the target cluster ID.
3. In the navigation panel on the left, click **User Management**.
4. In the upper-right corner of the page, click **Create RAM User**.

Add a Knox account

1. In the **User Management** page, select the account you want to add to a cluster, and then click **Set Knox Account Password** in the **Actions** column.
2. In the **Add Knox User** dialog box, enter a password to use for logon and click **OK**.
3. Refresh the **User Management** page. When **Synchronized** is displayed in the **Knox Account** column, you have successfully added the Knox account.

You can then sign in to Knox using the **User Name** and the password set in [Step 2](#).

Delete a Knox account

1. In the **User Management** panel, select the account you want to delete from a cluster, and then click **Delete Knox Account** in the **Actions** column.
2. Refresh the **User Management** page. When **Unsynchronized** is displayed in the **Knox Account** column, you have successfully deleted the Knox account.

FAQs

- Different clusters cannot share the same Knox account. For example, Knox account A that you added to cluster-1 cannot be used in cluster-2. If you want to use Knox account A in cluster-2, you must re-add account A to cluster-2.
- If the message **An error occurred while synchronizing the status** is displayed when you add a Knox account, click **Retry** to add it again.

- If you try to add an account multiple times but it fails each time, click **Clusters and Services** on the left side of the page to check if ApacheDS is stopped. If it is, start ApacheDS and go back to **User Management** to try again.

3.7 Service list

The **Clusters and Services** tab has been added to the tab list of the cluster management page to show the running statuses of HDFS, YARN, and other services.

The service list shows the following information.

Services 			
Normal	HDFS		Actions ▼
Normal	YARN		Actions ▼
Normal	Hive		Actions ▼
Normal	Ganglia		Actions ▼
Normal	Spark		Actions ▼
Normal	Hue		Actions ▼
Normal	Tez		Actions ▼
Normal	Sqoop		Actions ▼
Normal	Pig		Actions ▼
Normal	HAProxy		Actions ▼
Normal	ApacheDS		Actions ▼
Normal	Knox		Actions ▼
Normal	EmrFlow		Actions ▼

A service's running status is only showed for clusters that are in the Idle or Running status. Services that are not checked when creating a cluster, such as Storm, are not listed.

Click a service to view the corresponding tabs, including **Status**, **Component Topology**, **Configuration**, and **Configuration Change History**. The status can either be **Normal** or **Error**. If the status of a service on a node is **Error**, you can use the master node to log on to the node and check the service process.

3.8 Cluster scripts

To modify a cluster's running environment, you can install third-party software in the cluster. This mostly applies to Subscription clusters. After a cluster is created, the cluster script feature allows you to select nodes in batches and run your specified script to meet your own requirements.

Role of a cluster script

A cluster script is similar to a bootstrap action. After creating a cluster, you can install the following software packages to your cluster:

- YUM, which is used to install software already provided.
- Public software packages from the public network.
- Software that allows you to read your data from OSS.
- Services like Flink or Impala, which require a more complex script.

We strongly recommend that you test the cluster script on a node first. After the script has been verified, you can perform operations on the whole cluster.

Create and run a cluster script

1. A cluster script can run on an idle cluster or a running cluster. On the **Cluster Management** page, click **View Details** next to the cluster you want to run a script on.
2. On the menu on the left, click **Cluster Scripts** to enter the cluster script execution interface.
3. In the upper-right corner, click **Create and Run** to enter the creation interface.
4. Enter configurations in the script creation pane, select a node for execution, and click **OK**.

To update the running status of the cluster script on each node, click **Refresh**. To display the running status of the cluster script on each node, click **View Details**.

Cluster scripts are applicable for long-standing clusters and can only run on available clusters that are idle or running. To initialize on-demand clusters, perform a bootstrap action.

The cluster script feature downloads a script from the OSS and runs it on a specified node. If the returned value is 0, the execution has failed. If the execution fails, you can log on to the node to check the running log. The running log for each node is located at `/var/log/cluster-scripts/clusterScriptId`. If the cluster is configured with an OSS log directory, the running log is also uploaded to `osslogpath/clusterId/ip/cluster-scripts/clusterScriptId`.

By default, the root account is used to run the specified script. In the script, you can use `su hadoop` to switch to a Hadoop account.

A cluster script can successfully run on some nodes, but fail on others. For example, restarting a node can lead to a failure in script operation. After resolving the error, you can run the cluster script again. After a cluster is expanded, you can specify the expanded node for separate execution of the cluster script.

Only one cluster script can run on a cluster at a time. For each cluster, you can retain up to ten cluster script records. If you have ten records and want to create a new cluster script, you must first delete the previous records.

Script example

For a script similar to a bootstrap action, you can specify the file in the script that you want to download from OSS. In the following example, the file `oss://yourbucket/myfile.tar.gz` is downloaded and decompressed to the directory `/yourdir`:

```
#!/bin/bash
osscmd --id=<yourid> --key=<yourkey> --host=oss-cn-hangzhou-internal.
aliyuncs.com get oss://<yourbucket>/<myfile>.tar.gz ./<myfile>.tar.gz
mkdir -p /<yourdir>
tar -zxvf <myfile>.tar.gz -C /<yourdir>
```

OSSCMD is pre-installed on the node and can be called directly to download the file.



Note:

The OSS host address can be an intranet address, Internet address, or VPC network address. If a classic network is used, you must specify an intranet address. If the network is located in Hangzhou, the intranet address is `oss-cn-hangzhou-internal.aliyuncs.com`. If a VPC network is used, you must specify a domain name that can be accessed from the VPC intranet. If the network is located in Hangzhou, the domain name is `vpc100-oss-cn-hangzhou.aliyuncs.com`.

Additional system software packages can be installed on the script using YUM. For example, `ld-linux.so.2`:

```
#!/bin/bash
```

```
yum install -y ld-linux.so.2
```

3.9 Cluster renewal

When your Subscription cluster is about to expire, you have to renew it in order to continue using E-MapReduce cluster services. Cluster renewal includes the renewal of E-MapReduce services and ECS instances.

Enter the renewal page

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. At the top of the page, click **Cluster Management**.
3. In the cluster list, select the cluster that you want to renew.
4. To the right the cluster, click **Renew** to enter the cluster renewal page.

Renewal page

As shown in the following figure, the renewal page contains a number of columns, which are detailed below.

<input checked="" type="checkbox"/>	ECSExpiration Date	EMRExpiration Date	Quantity	ECSList	ECSSubscription Duration	EMRSubscription Duration	Price
<input checked="" type="checkbox"/>	2018-11-25 00:00:00	2018-11-25 00:00:00	1	i-bp15l0ep1zjnmokkz20x	1 Month ▾	1 Month ▾	0

Price: ¥ Calculating
 (E-MapReduce Service Terms)

- **ECS Expiration Date:** The expiration date of an ECS instance.
- **EMR Expiration Date:** The expiration date of E-MapReduce services.
- **Quantity:** The number of machines for instance groups.
- **ECS List:** The ECS instance ID of the machine in the cluster.
- **ECS Subscription Duration:** The duration of your ECS subscription. You can select from one to nine months, or one, two, or three years.
- **EMR Subscription Duration:** The duration of your E-MapReduce subscription. We recommend that you keep it consistent with ECS.
- **Price:** The price of the renewal of E-MapReduce services and ECS instances.

Make a payment



Note:

The fees consist of the combined costs of ECS renewal and E-MapReduce service products. If there are unpaid orders in the cluster list, you cannot expand or renew any clusters.

1. Click **OK** to view the prompt box for a successful order placement.
2. Click **Go to the payment page**. The payment page displays the total amount to be paid as well as the order details.
3. Click **Confirm payment**.
4. After you make the payment, click **Payment completed** to return to the cluster list page.

If an expired cluster is successfully renewed, its expiration date is updated to reflect the renewal.

If an expired ECS instance is successfully renewed, its expiration date is usually updated around three to five minutes later.

If you confirm the renewal, but fail to pay for it, **Cancel order** and **Make the payment** are displayed on the right side of the cluster. Click **Make the payment** to complete the payment or **Cancel** to cancel the renewal.

3.10 Security groups

Security groups created in E-MapReduce can be used during the creation of clusters.

Only port 22 is accessible in clusters created by E-MapReduce. We recommend that you divide ECS instances by function, and put them into different user security groups. For example, name the security group of E-MapReduce **E-MapReduce security group**, and name the security group that you create **User security group**. Each security group is provided with unique access control as required.

If you need to link with a cluster that has already been created, follow these steps.

Add an E-MapReduce cluster to an existing security group

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. At the top of the page, click **Cluster Management**.
3. Click **View Details**.
4. In the **Network** tab, find **Security Group ID** and click the ID link.
5. In the menu on the left, click **Instances in Security Group** to view the security group names of all ECS instances.
6. Log on to the [Alibaba Cloud ECS console](#) and click **Security Group** in the navigation panel on the left to find the security group as viewed in the preceding step.
7. Click **Manage Instances** in a security group and view ECS instance names that start with emr-xxx. These are the ECS instances in an E-MapReduce cluster.

8. Select all of these instances, click **Move to security group**, and then select a security group to move the E-MapReduce cluster to.

Add an existing cluster to an E-MapReduce security group

Find the security group where the existing cluster is located. Repeat the preceding operations, and move to the E-MapReduce security group. Select scattered machines in the ECS console directly and move the clusters to the **E-MapReduce security group** in batches.

Security group rules

When an ECS instance is in several different security groups, the security group rules are subject to the OR relationship. For example, only port 22 of the E-MapReduce security group is accessible, whereas all ports of the **User security group** are accessible. After the E-MapReduce cluster is added to the **User security group**, all ports of the machine in E-MapReduce are open.

3.11 Node upgrade

In real scenarios, the CPU or memory of a cluster node, especially master nodes, may be insufficient. Currently, E-MapReduce does not support direct upgrades. We recommend that you upgrade nodes in the following way.



Note:

Only Subscription clusters can be upgraded.

Procedure

1. In the **Cluster Management** panel, select a cluster, and click **View Details**.
2. In the upper-right corner, click **Configuration Upgrade**.
3. Configure the nodes that you want to upgrade.
4. Click **OK**.
5. Pay for your order.

Return to the **Cluster Management** page, refresh the page to make sure that the node configuration has become the target specification.

6. Log on to the [ECS console](#), find the upgraded instances, and restart them one by one.
7. Modify cluster configurations so that YARN can use new resources.
 - a. Modify the `yarn-site.xml` file.
 - b. Change the value of `yarn.nodemanager.resource.memory-mb` to machine memory * 0.8 and change the value of `yarn.scheduler.maximum-allocation-mb` to

machine memory * 0.8. The unit is MB. For example, in this new configuration, the memory is 32 GB:

```
yarn.nodemanager.resource.memory-mb=26214
yarn.scheduler.maximum-allocation-mb=26214
```

If your cluster does not support page modification, you must log on to the node, and modify the corresponding configuration values in the `/etc/emr/hadoop-conf/yarn-site.xml` file for each node.

- c. Restart YARN. Generally, you only need to restart the worker node. However, after the restart, the Node Manager port is changed. Therefore, we recommend that you restart the Resource Manager.
8. Submit a ticket to us providing information about the new node configuration. We will then synchronize the configuration.

3.12 Disk expansion

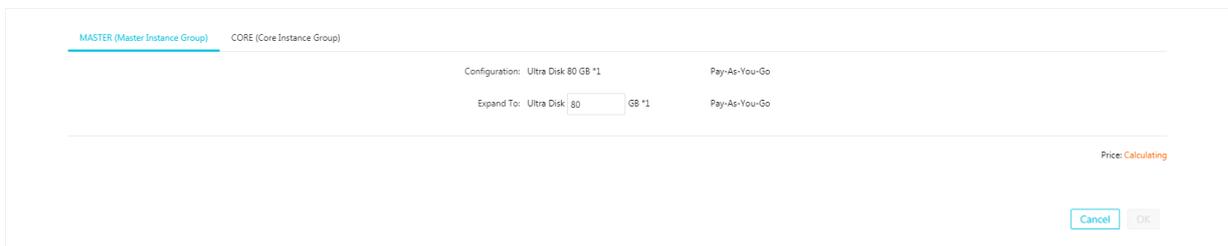
In the event that you do not have enough storage space, cluster disks of E-MapReduce 3.11.0 or later can be expanded on the cluster overview page. Cluster disks of E-MapReduce 3.11.0 or earlier can be scaled out in the ECS console.

Enter the expansion interface

On the **Cluster Management** page, find the target cluster. Click **View Details**, and in the upper-right corner, click **Disk Scale Up/Out**.

Expansion interface

The following figure shows the disk expansion interface displayed.



Note:

- E-MapReduce currently supports the expansion disks, but not their reduction.
- The expansion of disks is only supported for data disks.
- Configuration: Data disk configurations of the current instance group.

- Expand to: Data disk size after the disk is expanded.

Node data disk expansion

1. Log on to the E-MapReduce console, find the target cluster, and click **View Details** to go to the cluster overview page.
2. View the **ECS ID** of the core node in the cluster you want to expand, such as `i-bp1bsithym5hh9h93xxx`. All the node disks in the cluster are expanded uniformly by default to make sure the disk spaces of all nodes in the cluster are consistent.
3. Copy the **ECS ID** and go to the ECS console. Select **Instance Details** in the pane on the left, then select the instance ID and enter the ECS ID in the search box. Note that the same region must be selected.
4. When the corresponding ECS node is found, click **Manage**, and click **Disks** in the navigation pane on the left.
5. Expand the data disks. You must repeat the following steps to scale out each disk, because you cannot expand multiple disks in batches.
6. Expand all the disks in the ECS console and then restart the nodes.
7. See [Linux _ Resize a data disk](#) for more information on expanding data disks.



Note:

If the unmount operation fails, YARN and HDFS services must be disabled in the cluster. Disk1 operations may encounter unmount failures because of log writing by `ilogtail`. `ilogtail` must be temporarily killed using the command `sudo pgrep ilogtail | sudo xargs kill`. `ilogtail` can then be recovered by restarting the node.

8. You can then see that all disks are expanded using the `df -h` command on the node.
9. To guarantee that the subsequent E-MapReduce expansion process is consistent with the disk , submit a ticket to us so that we can perform cluster data updates.

Node system disk expansion

System disk expansion is a complex operation that should be avoided where necessary.

1. In the E-MapReduce console, enter the details page of the cluster you want to expand.
2. View the **ECS ID** of the master node in the cluster you want to expand, such as `i-bp1bsithym5hh9h93xxx`. All the node disks in the cluster are expanded uniformly by default to make sure the disk spaces of all nodes in the cluster are consistent.

3. Copy the `ECS ID` and go to the ECS console. Select **Instance Details** in the pane on the left, then select the instance ID and enter the ECS ID in the search box. Note that the same region must be selected.
4. When the corresponding ECS node is found, click **Manage**, and click **Disks** in the navigation pane on the left.
5. Find the system disk (there is only one).
6. See [Increase system disk size](#) for more information on expanding system disks.

**Note:**

- For non-HA clusters, while you are expanding the system disk, the cluster becomes unavailable.
- ECS disk operations may have changed the `/etc/hosts` file in the node. After the disk has been expanded, the file must be repaired. SSH log-on-free is also damaged, but the service remains unaffected and can be repaired manually.

3.13 Convert your billing method

There are two billing methods available for E-MapReduce: Pay-As-You-Go and Subscription. We recommend that you use Pay-As-You-Go, as it is more cost-effective. If you are sure that you want to use E-MapReduce for a longer period of time, you can change to the Subscription method.

Convert Pay-As-You-Go to Subscription

You can select the Pay-As-You-Go method at the beginning during your E-MapReduce trial and at any time after that. In the **Cluster Overview** panel, click the **Switch to Subscription** button to switch your payment method from Pay-As-You-Go to Subscription. The payment method for the entire cluster will be converted.

Convert Subscription to Pay-As-You-Go

You cannot currently convert from Subscription to Pay-As-You-Go.

3.14 Remove abnormal nodes

If an ECS node in an E-MapReduce cluster is abnormal, and if you do not need this node and want to remove it, you can do so.

To remove an abnormal node, follow these steps:

1. Log on to the [Alibaba Cloud E-MapReduce console](#).

2. At the top of the page, click **Cluster Management**.
3. Click the ID link of the cluster that includes the node that you want to remove.
4. In the navigation pane on the left, click **Hosts**.
5. Find the instance that you want to remove, and in the action column, click **Remove**. ECS instances can only be removed when they are in a stopped or released status.
6. Click **OK** to remove the instance.

4 Jobs

4.1 Configure a Hadoop MapReduce job

Procedure

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. At the top of the navigation bar, click **Data Platform**.
3. In the **Actions** column, click **Design Workflow** next to the specified project.
4. On the left of the Job Editing page, right-click the folder you want to operate and select **New Job**.
5. In the **New Job** dialog box, enter the job name and description.
6. Select a Hadoop job type to create a Hadoop MapReduce job. This type of job is submitted in the background using the following process.

```
hadoop jar xxx.jar [MainClass] -Dxxx ....
```

7. Click **OK**.

**Note:**

You can also create subfolders, rename folders, and delete folders by right-clicking on them.

8. Enter the parameters in the **Content** field that are required to submit this job. Enter the parameters after the Hadoop jar, followed by other command line parameters.

For instance, if you want to submit a Hadoop sleep job that does not read or write any data, this will only succeed if you submit Mapper/Reducer tasks to the cluster and wait for each task to sleep for a while. In Hadoop, this job is packaged in the Hadoop release version's `hadoop-mapreduce-client-jobclient-2.6.0-tests.jar`. If this job is submitted from the command line, the command should read as follows.

```
hadoop jar /path/to/hadoop-mapreduce-client-jobclient-2.6.0-tests.jar sleep -m 3 -r 3 -mt 100 -rt 100
```

To configure this job in E-MapReduce, enter the following content in the **Content** field.

```
/path/to/hadoop-mapreduce-client-jobclient-2.6.0-tests.jar sleep -m 3 -r 3 -mt 100 -rt 100
```

**Note:**

The jar package path used here is an absolute path on the E-MapReduce host. However, the user may put these jar packages anywhere, and as clusters are created and released, the packages become unavailable. Therefore, upload the jar package by performing the following steps:

1. Users send their own jar packages to the bucket of OSS for storage. When you configure the parameters for Hadoop, click **Select OSS path** to select and execute the jar package you want from the OSS directory. This system will then auto-complete the OSS address for jar packages. Be sure to switch the prefix of the jar to ossref by clicking **Switch resource type**. This ensures that the jar package is downloaded correctly by MapReduce.
2. Click **OK**. The OSS path for this package will be auto-completed in the **Content** field. When a job is submitted, the system will find the corresponding jar packages automatically based on this path.
3. Behind the jar package path for this OSS, other command line parameters for running jobs will be filled in further.

9. Click **Save**.

In the example above, the sleep job has no data input/output. If you want the job to read data and process input results, such as word counts, the data input and output paths need to be specified. You can read/write data on the HDFS of the E-MapReduce cluster as well as on OSS. To read/write data on OSS, write the data path as the OSS path when specifying the input and output paths. For instance:

```
jar ossref://emr/checklist/jars/chengtao/hadoop/hadoop-mapreduce-examples-2.6.0.jar randomtextwriter -D mapreduce.randomtextwriter.totalbytes=320000 oss://emr/checklist/data/chengtao/hadoop/Wordcount/
Input
```

4.2 Configure a Hive job

When you apply for clusters in E-MapReduce, you are provided with a Hive environment by default. Using Hive, you can create and operate tables and data.

Procedure

1. Prepare the Hive script in advance. For example:

```
USE DEFAULT;
DROP TABLE uservisits;
CREATE EXTERNAL TABLE IF NOT EXISTS uservisits (sourceIP STRING,
destURL STRING,visitDate STRING,adRevenue DOUBLE,user
Agent STRING,countryCode STRING,languageCode STRING,searchWord
STRING,duration INT ) ROW FORMAT DELIMITED FIELDS TERMINI
```

```
NATED BY ',' STORED AS SEQUENCEFILE LOCATION '/HiBench/Aggregation/
Input/uservisits';
DROP TABLE uservisits_aggre;
CREATE EXTERNAL TABLE IF NOT EXISTS uservisits_aggre ( sourceIP
STRING, sumAdRevenue DOUBLE) STORED AS SEQUENCEFILE LO
CATION '/HiBench/Aggregation/Output/uservisits_aggre';
INSERT OVERWRITE TABLE uservisits_aggre SELECT sourceIP, SUM(
adRevenue) FROM uservisits GROUP BY sourceIP;
```

2. Save this script into a script file, such as *uservisits_aggre_hdfs.hive*, and upload it to an OSS directory (for example, *oss://path/to/uservisits_aggre_hdfs.hive*).
3. Log on to the [Alibaba Cloud E-MapReduce console](#).
4. At the top of the navigation bar, click **Data Platform**.
5. In the **Actions** column, click **Design Workflow** next to the specified project.
6. On the left of the Job Editing page, right-click the folder you want to operate and select **New Job**.
7. In the **New Job** dialog box, enter the job name and description.
8. Select the Hive job type to create a Hive job. This type of job is submitted in the background using the following method.

```
hive [user provided parameters]
```

9. Click **OK**.



Note:

You can also create subfolders, rename folders, and delete folders by right-clicking on them.

10. Enter the parameters in the **Content** field after the Hive commands. For example, if you want to use a Hive script uploaded to OSS, enter the following.

```
-f ossref://path/to/uservisits_aggre_hdfs.hive
```

You can also click **Select OSS path** to view and select from OSS. The system will automatically complete the path of the Hive script on OSS. Switch the Hive script prefix to *ossref* by clicking **Switch resource type**. This ensures that the file is correctly downloaded by E-MapReduce.

11. Click **Save** to complete the Hive job configuration.

4.3 Configure a Pig job

When you apply for clusters in E-MapReduce, a Pig environment is provided by default. Using Pig, you can create and operate tables and data.

Procedure

1. Prepare the Pig script in advance. For example:

```

```shell
/*
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements. See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership. The ASF licenses this file
 * to you under the Apache License, Version 2.0 (the
 * "License"); you may not use this file except in compliance
 * with the License. You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing,
software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions
and
 * limitations under the License.
 */
-- Query Phrase Popularity (Hadoop cluster)
-- This script processes a search query log file from the Excite
search engine and finds search phrases that occur with particular
high frequency during certain times of the day.
-- Register the tutorial JAR file so that the included UDFs can be
called in the script.
REGISTER oss://emr/checklist/jars/chengtao/pig/tutorial.jar;
-- Use the PigStorage function to load the excite log file into
the "raw" bag as an array of records.
-- Input: (user,time,query)
raw = LOAD 'oss://emr/checklist/data/chengtao/pig/excite.log.bz2'
USING PigStorage('\t') AS (user, time, query);
-- Call the NonURLDetector UDF to remove records if the query field
is empty or a URL.
clean1 = FILTER raw BY org.apache.pig.tutorial.NonURLDetector(query
);
-- Call the ToLower UDF to change the query field to lowercase.
clean2 = FOREACH clean1 GENERATE user, time, org.apache.pig.
tutorial.ToLower(query) as query;
-- Because the log file only contains queries for a single day, we
are only interested in the hour.
-- The excite query log timestamp format is YMMDDHHMMSS.
-- Call the ExtractHour UDF to extract the hour (HH) from the time
field.
houred = FOREACH clean2 GENERATE user, org.apache.pig.tutorial.
ExtractHour(time) as hour, query;
-- Call the NGramGenerator UDF to compose the n-grams of the query.
ngramed1 = FOREACH houred GENERATE user, hour, flatten(org.apache.
pig.tutorial.NGramGenerator(query)) as ngram;

```

```

-- Use the DISTINCT command to get the unique n-grams for all
records.
ngramed2 = DISTINCT ngramed1;
-- Use the GROUP command to group records by n-gram and hour.
hour_frequency1 = GROUP ngramed2 BY (ngram, hour);
-- Use the COUNT function to get the count (occurrences) of each n
-gram.
hour_frequency2 = FOREACH hour_frequency1 GENERATE flatten($0),
COUNT($1) as count;
-- Use the GROUP command to group records by n-gram only.
-- Each group now corresponds to a distinct n-gram and has the
count for each hour.
uniq_frequency1 = GROUP hour_frequency2 BY group::ngram;
-- For each group, identify the hour in which this n-gram is used
with a particularly high frequency.
-- Call the ScoreGenerator UDF to calculate a "popularity" score
for the n-gram.
uniq_frequency2 = FOREACH uniq_frequency1 GENERATE flatten($0),
flatten(org.apache.pig.tutorial.ScoreGenerator($1));
-- Use the FOREACH-GENERATE command to assign names to the fields
.
uniq_frequency3 = FOREACH uniq_frequency2 GENERATE $1 as hour, $0
as ngram, $2 as score, $3 as count, $4 as mean;
-- Use the FILTER command to move all records with a score less
than or equal to 2.0.
filtered_uniq_frequency = FILTER uniq_frequency3 BY score > 2.0;
-- Use the ORDER command to sort the remaining records by hour and
score.
ordered_uniq_frequency = ORDER filtered_uniq_frequency BY hour,
score;
-- Use the PigStorage function to store the results.
-- Output: (hour, n-gram, score, count, average_counts_among
_all_hours)
STORE ordered_uniq_frequency INTO 'oss://emr/checklist/data/
chengtao/pig/script1-hadoop-results' USING PigStorage();
````

```

2. Save this script into a script file, such as *script1-hadoop-oss.pig*, and upload it to an OSS directory (for example, *oss://path/to/script1-hadoop-oss.pig*).
3. Log on to the [Alibaba Cloud E-MapReduce console](#).
4. At the top of the navigation bar, click **Data Platform**.
5. In the **Actions** column, click **Design Workflow** next to the specified project.
6. On the left of the Job Editing page, right-click the folder you want to operate and select **New Job**.
7. In the **New Job** dialog box, enter the job name and description.
8. Select the Pig job type to create a Pig job. This type of job is submitted in the background using the following method.

```
pig [user provided parameters]
```

9. Click **OK**.

**Note:**

You can also create subfolders, rename folders, and delete folders by right-clicking on them.

10. Enter the parameters in the **Content** field after the Pig commands. For example, if you want to use a Pig script uploaded to OSS, enter the following.

```
-x mapreduce ossref://emr/checklist/jars/chengtao/pig/script1-hadoop  
-oss.pig
```

You can click **Select OSS path** to view and select from OSS. The system will automatically complete the path of Pig script on OSS. Switch the Pig script prefix to ossref by clicking **Switch resource type**. This ensures that the file is correctly downloaded by E-MapReduce.

11. Click **Save** to complete the Pig job configuration.

4.4 Configure a Spark job

In this tutorial, you will learn how to configure a Spark job.

Procedure

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. At the top of the navigation bar, click **Data Platform**.
3. In the **Actions** column, click **Design Workflow** next to the specified project.
4. On the left of the Job Editing page, right-click the folder you want to operate and select **New Job**.
5. In the **New Job** dialog box, enter the job name and description.
6. Click **OK**.

**Note:**

You can also create subfolders, rename folders, and delete folders by right-clicking on them.

7. Select the Spark job type to create a Spark job. This type of job is submitted in the background using the following method.

```
spark-submit [options] --class [MainClass] xxx.jar args
```

8. Enter the parameters in the **Content** field that are required to submit this job. Only the parameters after **spark-submit** can be entered. The following example shows how to enter the parameters for creating a Spark job and a PySpark job.

- Create a Spark job

Create a Spark WordCount job:

— Job name: WordCount

— Type: Select Spark

— Parameters:

- Enter the following command:

```
spark-submit --master yarn-client --driver-memory 7G --
executor-memory 5G --executor-cores 1 --num-executors 32 --
class com.aliyun.emr.checklist.benchmark.SparkWordCount emr
-checklist_2.10-0.1.0.jar oss://emr/checklist/data/wc oss://
emr/checklist/data/wc-counts 32
```

- Enter the following in the E-MapReduce job **Content** field:

```
--master yarn-client --driver-memory 7G --executor-memory 5G
--executor-cores 1 --num-executors 32 --class com.aliyun.emr
.checklist.benchmark.SparkWordCount ossref://emr/checklist/
jars/emr-checklist_2.10-0.1.0.jar oss://emr/checklist/data/wc
oss://emr/checklist/data/wc-counts 32
```



Note:

Job jar packages are saved in OSS. In the example above, the way to reference the Jar package is `ossref://emr/checklist/jars/emr-checklist_2.10-0.1.0.jar`. Click **Select OSS path** to view and select one from OSS. The system will automatically complete the absolute path of the Spark script on OSS. Switch the default OSS protocol to the `ossref` protocol.

- Create a PySpark job

In addition to Scala and Java job types, E-MapReduce also supports Python job types in Spark. Create a Spark K-means job for the Python script:

— Job name: Python-Kmeans

— Type: Spark

— Parameters:

```
--master yarn-client --driver-memory 7g --num-executors 10
--executor-memory 5g --executor-cores 1 --jars ossref://emr/
checklist/jars/emr-core-0.1.0.jar ossref://emr/checklist/python
/wordcount.py oss://emr/checklist/data/kddb 5 32
```

- References of Python script resources are supported, and the `ossref` protocol is used.
- For PySpark, the online Python installation kit is not supported.

9. Click **Save** to complete the Spark job configuration.

4.5 Configure a Spark SQL

In this tutorial, you will learn how to configure a Spark SQL job.



Note:

By default, the mode of Spark SQL used for submitting a job is YARN.

Procedure

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. At the top of the navigation bar, click **Data Platform**.
3. In the **Actions** column, click **Design Workflow** next to the specified project.
4. On the left of the Job Editing page, right-click the folder you want to operate and select **New Job**.
5. In the **New Job** dialog box, enter the job name and description.
6. Click **OK**.



Note:

You can also create subfolders, rename folders, and delete folders by right-clicking on them.

7. Select the Spark SQL job type to create a Spark SQL job. This type of job is submitted in the background using the following method.

```
spark-sql [options] [cli option]
```

8. Enter the parameters in the **Content** field after the Spark SQL commands.

- -e option

-e options can be written to the running SQL by inputting them into the **Content** field of the job. For example:

```
-e "show databases;"
```

- -f option

-f options can be used to specify a Spark SQL script file. Uploading well-prepared Spark SQL script files to OSS can provide greater flexibility. We recommend that you use this operation mode. For example:

```
-f ossref://your-bucket/your-spark-sql-script.sql
```

9. Click **Save** to complete Spark SQL job configuration.

4.6 Configure a Shell job

In this tutorial, you will learn how to configure a Shell job.



Note:

By default, Shell scripts are currently run by Hadoop. If you need to use the root user, sudo can be used. Use Shell script jobs with caution.

Procedure

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. At the top of the navigation bar, click **Data Platform**.
3. In the **Actions** column, click **Design Workflow** next to the specified project.
4. On the left of the Job Editing page, right-click the folder you want to operate and select **New Job**.
5. In the **New Job** dialog box, enter the job name and description.
6. Select the Shell job type to create a Bash Shell job.
7. Click **OK**.



Note:

You can also create subfolders, rename folders, and delete folders by right-clicking on them.

8. Enter the parameters in the **Content** field after the Shell commands.

- -c option

-c options can be used to set Shell scripts to run by inputting them into the **Content** field of the job. For example:

```
-c "echo 1; sleep 2; echo 2; sleep 4; echo 3; sleep 8; echo 4;
sleep 16; echo 5; sleep 32; echo 6; sleep 64; echo 8; sleep 128;
echo finished"
```

- -f option

-f options can be used to run Shell script files. By uploading a Shell script file to OSS, Shell scripts on OSS can be defined in the job parameters, making it more flexible than the -c option. For example:

```
-f ossref://mxbucket/sample/sample-shell-job.sh
```

9. Click **Save** to complete Shell job configurations.

4.7 Configure a Sqoop job

In this tutorial, you will learn how to configure a Sqoop job.

**Note:**

Only E-MapReduce products with version V1.3.0 or later support the Sqoop job type. Running a Sqoop job on lower versions will fail and errorlog will report "Not supported" errors. For more information on parameters, see [Sqoop](#).

Procedure

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. At the top of the navigation bar, click **Data Platform**.
3. In the **Actions** column, click **Design Workflow** next to the specified project.
4. On the left of the Job Editing page, right-click the folder you want to operate and select **New Job**.
5. In the **New Job** dialog box, enter the job name and description.
6. Select the Sqoop job type to create a Sqoop job. This type of job is submitted in the background using the following method.

```
sqoop [args]
```

7. Click **OK**.

**Note:**

You can also create subfolders, rename folders, and delete folders by right-clicking on them.

8. Enter the parameters in the **Content** field after the Sqoop commands.
9. Click **Save** to complete Sqoop job configuration.

4.8 Job operations

You can create, clone, modify, and delete jobs.

Job creation

A new job can be created at any time. Currently, a job can only be used in the region where it is created.

Job cloning

Configurations that already exist for a job can be cloned. A cloned job can also only be used in the region where it is created.

Job modification

Before you can modify a job that needs to be added to an execution plan, you must first ensure that the execution plan is not running and that its periodic scheduling is not in progress.

Before you can modify a job that needs to be added to several execution plans, you must first ensure that none of the execution plans are running and that none of their periodic scheduling is in progress. Modifying a job may result in changes to all of the execution plans that use this job.

If you need to debug, we recommend that you perform cloning instead. After you debug, the original jobs in the execution plan are replaced.

Job deletion

As with modification, a job can only be deleted when the execution plan where the job is located is not running and its periodic scheduling is not in progress.

4.9 Time and date variables

When you are creating a job, variable wildcards are supported in the job parameters for both time and date.

Variable wildcard format

The format of the variable wildcards supported by E-MapReduce is either `${dateexpr-1d}` or `${dateexpr-1h}`. For example, assuming the current date and time is 20160427 12:08:01:

- If `${yyyyMMdd HH:mm:ss-1d}` is displayed, the parameter wildcard is replaced with 20160426 12:08:01 when executed, which is the current date minus one day, and time accurate to the second.
- If `${yyyyMMdd-1d}` is displayed, the parameter wildcard is replaced with 20160426 when executed, which is the current date minus one day.
- If `${yyyyMMdd}` is displayed, the parameter wildcard is replaced with 20160427, which is the current date.

dateexpr represents the standard format of expressing time. Time is therefore formatted according to this expression and is followed by the amount of time that you want to add or deduct, which can be written as N. For example, `${yyyyMMdd-5d}`, `${yyyyMMdd+5d}`, `${yyyyMMdd+5h}`, or `${yyyyMMdd-5h}`.



Note:

E-MapReduce currently supports the addition and deduction of hours and days only.

Example

When executed, the **Parameter** in the following figure will be replaced with:

```
jar ossref://emr/jar/hadoop/hadoop_wc.jar com.aliyun.emr.WordCount oss://emr/output/pt=20160426
```

Modify job ✕

*** Name :**
Length: 1 to 64 characters. Only Chinese characters, English letters, numbers '-', and '_' are allowed

*** Type :** Spark Hadoop Hive Pig
 Sqoop Spark SQL Shell

*** Parameter :**

```
jar ossref://emr/hadoop_wc.jar com.aliyun.emr.WordCount 'oss://emr/output/pt=${yyyyMMdd- 1d}
```

[+ Select OSS path](#) [oss console Upload](#)

*** Actual execution :** **spark-submit** jar ossref://emr/hadoop_wc.jar com.aliyun.emr.WordCount 'oss://emr/output/pt=\${yyyyMMdd- 1d}

Fail retry Yes No

*** Failure policy :** Pause current execution plan
 Continue execution of next job

OK Cancel

5 Execution plans

5.1 Create an execution plan

An execution plan is a set of jobs that can be executed either at one time or periodically by means of scheduling. It can be executed on an existing E-MapReduce cluster and can also create a temporary cluster to execute the jobs dynamically. Its biggest advantage is that it only uses the resources it needs during execution.

Procedure To create an execution plan, follow these steps:

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
 2. Select a region.
 3. In the upper-right corner, click **Old MER Scheduling** to go to the Jobs page.
 4. In the navigation panel on the left, click **Execution plan**.
 5. In the upper-right corner, click **Create an execution plan**.
 6. In the **Create an execution plan** page, select between **Create as needed** and **Existing clusters**.
 - a. **Create as needed:** Create a new cluster to run jobs.
 - Execution plan for one-time scheduling: Clusters with corresponding configurations are created when the execution starts and are then released upon completion of the operation. For more information about creation parameters, see [Create a cluster](#).
 - Execution plan for periodic scheduling: A new cluster is created based on the scheduling settings you define and is then released upon completion of the operation.
 - b. **Existing clusters:** Use an existing cluster that complies with the following requirement:
 - Execution plans can only be added to clusters that are **Running** or **Idle**.
- Select **Existing clusters** and then enter the **Select Cluster** page. Here, you can select a cluster to associate with the execution plan.
7. Click **Next** to enter the job configuration page. All user jobs are listed in the table on the left. You can select jobs for execution from this table. By clicking the right-facing button, the checked jobs are added to the job queue. Jobs in the queue are then submitted to the cluster for execution in order. The same job can be added and executed several times. If you have not created any jobs, see Jobs.

8. Click **Next** to enter the scheduling mode configuration page. The configuration items are as follows:

a. **Name:** Must be between 1-64 characters and may only consist of Chinese characters, English letters, numbers, hyphens (-), and underscores (_).

b. Scheduling policy

- **Manual execution:** The execution plan is not executed automatically after it is created. Instead, it must be executed manually. Once the execution is in progress, it cannot be executed again.
- **Periodic scheduling:** If you select this function, it is enabled immediately after the execution plan is created. The execution then begins from the configured scheduling time. Periodic scheduling can be disabled in the list page. If a scheduling execution starts, but its last execution is not completed, the scheduling is ignored.

c. **Set the scheduling cycle:** There are two scheduling periods: days and hours. The day cycle is one day by default and cannot be changed. However, you can set a specific time interval for hours. The range must be 1-23.

d. **First execution time:** The effective start-time of the scheduling. From this point onwards, periodic scheduling is conducted according to the intervals specified.

9. Click **OK** to complete the creation of the execution plan.

Other information

- Example of periodic scheduling

* Set the scheduling cycle :

* Set the scheduling cycle : per day(s)

* first execute time : :

First run time 2018-11-1 17:35

Subsequent intervals 1 day(s) run 1 Times

These configurations indicate that the scheduling started on 11/01/2018 at 17:35 with an interval of one day. This means that the second scheduling was conducted on 11/02/2018, at 17:35.

- Sequence of jobs

Jobs in the execution plan are executed from first to last according to the sequence that you defined in the job list.

- Sequence of multiple execution plans

When multiple execution plans are submitted to the same cluster, each one submits jobs from its own job sequence. This means that jobs run parallel with each other.

- Example of early job debugging

During the debugging of a job, it may take some time to create and start a cluster on demand. We recommend that you create a cluster manually first, select **Associate the cluster** in the execution plan to run jobs, and then set the scheduling mode to **Execute** immediately. During debugging, you can view the results by clicking **Run now** on the execution plan list page. Once the debugging is finished, modify the execution plan, modify the way you associate an existing cluster to create a new cluster on demand, and then modify the scheduling mode to periodic scheduling as required. Jobs are then executed automatically on demand.

5.2 Manage an execution plan

You can view, manage, and modify your execution plans as follows.

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. Select a region.
3. In the upper-right corner, click **Old MER Scheduling** to go to the Jobs page.
4. In the navigation panel on the left, click **Execution plan**.
5. Click **Manage** next to a plan to go to the execution plan detail page. Here, you can perform the following operations:

- View details of the execution plan

You can view the basic information of the execution plan, such as its name, associated clusters, job configurations, scheduling mode and status, and alarm information.

- Modify the execution plan



Note:

Jobs can only be modified if they are not currently running or being scheduled. For an execution plan to be executed immediately, it can only be modified when it is not currently running. If the execution plan is scheduled periodically, wait for the completion of its current operation and verify whether it is in periodical scheduling. If it is, click **Stop scheduling** before modifying it.

Each separate module can be modified independently. Click the **pen** icon to modify information.

- Configure alarm notifications

There are three types of alarm notifications:

- **Booting timeout:** If the periodical scheduling has not been conducted correctly at the specified time and is not executed within 10 minutes of timeout, an alarm is sent.
- **Failed execution:** If any job in the execution plan fails, an alarm is sent.
- **Successful execution:** If all jobs in the execution plan are executed successfully, a notification is sent.

- Run and view results

If the execution plan can be run, in **Basic Information**, there will be a **Run now** button to the right of **Scheduling status**. If you click this button, a schedule will be executed.

At the bottom of the page, there are running records displaying the execution plan instances executed each time, making it easy to view the corresponding job list and logs.

5.3 Execution plan list

An execution plan list displays basic information about all of your execution plans, as shown in the following figure.

| ID/Name | Last run cluster | Last run | Scheduling status | Operation |
|-------------------------------|------------------|--|---------------------|---------------------------|
| WF-63300C427CA72165
test1 | dtplus_docs | Start Time : 2018/10/29 10:40:15
Running time : 4second(s)
Running Status : Complete | | Manage Run now More ▾ |
| WF-C719251FFA366B3C
周期调度测试 | 高配测试 | Start Time : 2018/10/29 10:37:24
Running time : 8second(s)
Running Status : Complete | ● Scheduling paused | Manage Run now More ▾ |
| WF-A388AA222DA06359
test1 | 高配测试 | Start Time : 2018/10/31 17:32:00
Running time :
1day(s)16hour(s)36minute(s)34second(s)
Running Status : Running | ● Scheduling | Manage Run now More ▾ |

- **ID/Name:** The ID and name of the execution plan.
- **Last run cluster:** The last cluster to execute this execution plan. This can either be a cluster created on demand or an existing associated cluster. If a cluster is created automatically on

demand, (Automatically created) is displayed beneath it, indicating that the cluster was created on demand by E-MapReduce and will be released automatically after running.

- **Last run:** The running status of the last execution plan.
 - **Start time:** The time at which the last execution plan started.
 - **Running time:** The duration for which the last plan ran.
 - **Running status:** The running status of the last execution plan.
- **Scheduling status:** This indicates whether scheduling is in progress or has been stopped. Only periodic jobs have a scheduling status.
- **Operation**
 - **Manage:** View and modify execution plans.
 - **Run now:** A job can only be run manually when it is neither running nor being scheduled. Click **Run now** to run the execution plan immediately.
 - **More**
 - **Start/Stop scheduling:** If the scheduling is stopped, **Enable scheduling** is displayed, which you can click to start the scheduling. If **Stop scheduling** is displayed during scheduling, you can click it to stop the scheduling. This button is only available for periodic execution plans.
 - **Running log:** Click to enter the job log viewing page.
 - **Delete:** Deletes an execution plan. A running execution plan or one in the process of scheduling cannot be deleted.

5.4 View job results and logs

In this tutorial, you will learn how to view job results and logs.

View execution records

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. Select a region.
3. In the upper-right corner, click **Old MER Scheduling** to go to the Jobs page.
4. In the navigation panel on the left, click **Execution plan**.
5. To the right of the execution plan, click **More > > Running log**.

Running log [Back to execution plan list](#) [Refresh](#)

ID/NameWF-A388AA222DA06359/test1

| Execution order ID | Running Status | Start Time | Running time | Execute cluster | Operation |
|--------------------|----------------|---------------------|--|-----------------|---|
| 7 | Running | 2018/10/31 17:32:00 | 1day(s)17hour(s)18minute(s)17second(s) | 高配测试 | Stop all jobs View job list |
| 6 | Complete | 2018/10/30 17:32:00 | 7second(s) | 高配测试 | View job list |
| 5 | Complete | 2018/10/29 17:32:00 | 11second(s) | 高配测试 | View job list |
| 4 | Complete | 2018/10/28 17:32:00 | 5second(s) | 高配测试 | View job list |
| 3 | Complete | 2018/10/27 17:32:00 | 9second(s) | 高配测试 | View job list |
| 2 | Complete | 2018/10/26 17:31:58 | 5second(s) | 高配测试 | View job list |
| 1 | Complete | 2018/10/26 17:23:29 | 10second(s) | 高配测试 | View job list |

- **Execution order ID:** The sequence of execution for the execution record, which indicates its position in the execution queue. For example, 1 stands for the first position.
- **Running status:** The running status of each execution record.
- **Start time** The time at which the execution plan starts.
- **Running time:** The total running time until the page is viewed.
- **Execute cluster:** The cluster run by the execution plan can either be created on demand or it can be an existing associated cluster. Click to view the cluster details page.
- **Operation**

View job list: Click to enter the job list page.

View job records

On the **Job list** page, you can view the job list in the execution records of a single execution plan as well as the details of each job, as shown in the following figure.

| Job execution order ID | Name | Status | Type | Start Time | Running time | Operation |
|------------------------|------|--------|-------|---------------------|--------------|--|
| WNE-92693C5408D39E03 | test | Failed | Spark | 2018/10/26 17:23:39 | 1second(s) | Stop Job stdout stderr workers log |

- **Job execution order ID:** After a job is executed, a corresponding ID is created, which is different from the job ID. The job execution ID is the unique identifier for viewing logs on OSS.
- **Name:** The name of the job.
- **Status:** The running status of the job.
- **Type:** The type of job.
- **Start time:** The time at which the job starts. This is converted into local time.
- **Running time:** The total running time of the job, in seconds.
- **Operation**

- **Stop job:** You can stop a job if it is in the process of submission or running. If a job is in submission, stopping it will cancel execution. If the job is running, it will be killed.
- **stdout:** Records all output content from the standard output (Channel 1) of the master process. If log saving is not enabled for the cluster where jobs are run, this function cannot be executed.
- **stderr:** Records all output content from the diagnostic output (Channel 2) of the master process. If log saving is not enabled for the cluster where jobs are run, this function cannot be executed.
- **Workers log:** Views the logs of all job worker nodes. If log saving is not enabled for the cluster where jobs are run, this function cannot be executed.

View job worker logs

- **Cloud server instance IP:** The ECS instance ID of a running job and the corresponding intranet IP address.
- **Container ID:** The container ID that YARN runs.
- **Type:** Different log types. stdout and stderr come from different outputs.
- **Operation**

View the log: Click different types to view the corresponding logs.

5.5 Parallel execution of multiple execution plans

To maximize the use of a cluster's available computing resources, multiple execution plans can be associated to the same cluster and executed in parallel.

The main points are summarized as follows:

- Jobs in the same execution plan are executed in sequence. By default, preceding jobs are executed before new jobs can be submitted and executed.
- If you have enough cluster resources, you can create multiple different execution plans and associate them to the same cluster to run and execute jobs in parallel. Clusters support a maximum of 20 execution plans by default.
- The management and control system currently supports the submission to YARN of multiple execution plans associated to the same cluster. However, if the cluster itself has insufficient resources, it may take some time for jobs in the YARN queue to wait for scheduling.

For more information on how to create execution plans and associate them to a cluster, see

[Create an execution plan.](#)

6 Alarm management

6.1 Cluster alarm management

To help you monitor cluster operations, CloudMonitor offers multiple monitoring metrics for E-MapReduce clusters, including CPU idleness, memory capacity, and disk capacity. It also allows you to set alarm rules for these monitoring metrics. Once an alarm rule is triggered, CloudMonitor promptly notifies the contacts in the notification group, allowing you to deal with the problem in time.

Configure alarm rules

To set an alarm rule for an E-MapReduce cluster, perform the following steps:

1. Log on to [CloudMonitor console](#).
2. In the navigation panel on the left, click **Cloud Service Monitoring** > > **E-MapReduce** to go to the E-MapReduce monitoring list page.
3. Click the **Alarm Rules** tab.
4. In the upper-right corner of the page, click **Create Alarm Rule** to set an alarm rule for a monitoring metric.
5. In the **Related Resource** area, set **Products** and **Resource Range**.
 - **Products**: Select E-MapReduce in the drop-down list.
 - **Resource Range**: The scope of the alarm rule, which is divided into three areas: all resources, application group, and cluster. When all resources is selected, the maximum number of resources that can be monitored is 1,000. If you have more than 1,000 resources, an alarm may not occur when the threshold is reached. We recommend that you set the alarm rules first before using application group to divide resources according to your needs.
 - **All Resources**: Indicates that the rule works on all E-MapReduce instances under the account. For example, if you set an alarm for when the CPU usage becomes greater than 80%, as long as there is an instance whose CPU usage is greater than 80% in the account, the rule is triggered.
 - **Application Group**: Indicates that the rule works on all instances of a certain application group. For example, if you set an alarm for when the CPU usage becomes greater than 80%, as long as there is a host whose CPU usage is greater than 80% in this group, the rule is triggered.

- **Cluster:** Indicates that the rule only works on a specific E-MapReduce cluster. For example, if you set an alarm for when the CPU usage becomes greater than 80%, as long as there is an instance whose CPU usage is greater than 80% in this cluster, the rule is triggered.

6. Configure the **Set Alarm Rules** area.

- **Alarm Rule:** Sets the name of an alarm rule.
- **Rule Description:** The part of the alarm rule that defines whether the conditions have been met to trigger the alarm rule. If they have been met, the alarm rule is triggered. For example, if the rule is set as an average CPU usage within 1 minute of greater than or equal to 90%, the rule is triggered if the average CPU usage within 1 minute is greater than or equal to 90%. For more information about the monitoring metrics in E-MapReduce clusters, see [E-MapReduce monitoring](#).
- **Role:** By default, any role is applicable.

If you click **Add Alarm Rule**, you can set multiple alarm rules (which are billed as such). If one of the rules is triggered, the system sends notifications to the notification group.

- **Mute for:** The interval at which alarm notifications are sent again in the event that the monitoring metrics are not returned to normal after an alarm occurs.
- **Triggered when threshold is exceeded for:** Number of times the rule is triggered to send alarm notifications. For example, if the rule is set as an average CPU usage within 1 minute of greater than or equal 90%, and this happens on 3 occasions, the rule is triggered if the average CPU usage within 1 minute is greater than or equal to 90% on 3 occasions.
- **Effective Period:** The effective time of the alarm rule. CloudMonitor only checks whether the monitoring metric triggers an alarm rule during this effective period. The system also only checks whether the monitoring data requires an alarm during the effective time.

7. Configure the **Notification Method** area.

- **Notification Contact:** The contact group that receives the notification. Enter a keyword for the contact group in the search box to quickly locate the group that you want to associate. By clicking the right arrow icon, the contact group is added to the contact list on the right. If you have not yet created a contact group, click **Quickly create a contact group**. If you want to delete a contact group from the contact list on the right, click the left arrow icon.
- **Notification Methods:** Alarm information is divided into three levels: critical, warning, and info. These different levels correspond to different notification methods. Before you can configure the critical level, you must first purchase the phone alarm resource package.

- **Email Remark** (Optional): Customizes additional information for alarm emails. The remarks you enter are included in the notification message sent to contacts.
 - **HTTP CallBack** (Optional): This feature allows you to integrate the alarm notifications sent by CloudMonitor into existing maintenance systems or message notification systems. CloudMonitor pushes alarm notifications to a specified public URL through the POST request method supported by HTTP. When you receive an alarm notification, you can process it further according to its content. For more information, see [Alarm callback](#).
8. Click **Confirm** to complete the alarm rule configuration

7 Software configuration

Software such as Hadoop, Hive, and Pig contain a large number of configurations that can be modified using the software configuration function. For example, you may want to increase the number of service threads in the `dfs.namenode.handler.count` HDFS server from 10 to 50, or decrease the default size of HDFS file block `dfs.blocksize` from 128 MB to 64 MB because the system only contains small files. The software configuration function can only be performed once when a cluster is started.

Purpose

The function can only be performed once during the startup of a cluster.

Procedure

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. Select a region. The cluster associated with this region is listed.
3. Click **Create Cluster** to enter the cluster creation page.
4. All of the software and the corresponding versions can be seen in the software configuration box during cluster creation. If you want to change the configuration of the cluster, select a corresponding .json configuration file in the software configuration box. You can then override or add to the default cluster parameters. An example of a .json file is as follows:

```
{
  "configurations": [
    {
      "classification": "core-site",
      "properties": {
        "Fs. Trash. Interval": "61"
      }
    },
    {
      "classification": "hadoop-log4j",
      "properties": {
        "hadoop.log.file": "hadoop1.log",
        "hadoop.root.logger": "INFO",
        "a.b.c": "ABC"
      }
    },
    {
      "classification": "hdfs-site",
      "properties": {
        "dfs.namenode.handler.count": "12"
      }
    },
    {
      "classification": "mapred-site",
      "properties": {
        "mapreduce.task.io.sort.mb": "201"
      }
    }
  ]
}
```

```

    }
  },
  {
    "classification": "yarn-site",
    "properties": {
      "Hadoop.Security.Groups.cache.secs": 251,
      "yarn.nodemanager.remote-app-log-dir": "/tmp/logs1"
    }
  },
  {
    "classification": "httpsfs-site",
    "properties": {
      "a.b.c.d": "200"
    }
  },
  {
    "classification": "capacity-scheduler",
    "properties": {
      "yarn.scheduler.capacity.maximum-am-resource-percent":
"0.2"
    }
  },
  {
    "classification": "hadoop-env",
    "properties": {
      "BC": "CD"
    },
    "configurations": [
      {
        "classification": "export",
        "properties": {
          "AB": "${BC}",
          "HADOOP_CLIENT_OPTS": "-Xmx512m -Xms512m $
HADOOP_CLIENT_OPTS"
        }
      }
    ]
  },
  {
    "classification": "httpfs-env",
    "properties": {
    },
    "configurations": [
      {
        "classification": "export",
        "properties": {
          "HTTPFS_SSL_KEYSTORE_PASS": "passwd"
        }
      }
    ]
  },
  {
    "classification": "mapred-env",
    "properties": {
    },
    "configurations": [
      {
        "classification": "export",
        "properties": {
          "HADOOP_JOB_HISTORYSERVER_HEAPSIZE": "1001"
        }
      }
    ]
  }
}

```

```

    ]
  },
  {
    "classification": "yarn-env",
    "properties": {
    },
    "configurations": [
      {
        "classification": "export",
        "properties": {
          "HADOOP_YARN_USER": "${HADOOP_YARN_USER:-yarn1}"
        }
      }
    ]
  },
  {
    "classification": "pig",
    "properties": {
      "pig.tez.auto.parallelism": "false"
    }
  },
  {
    "classification": "pig-log4j",
    "properties": {
      "log4j.logger.org.apache.pig": "error, A"
    }
  },
  {
    "classification": "hive-env",
    "properties": {
      "BC": "CD"
    },
    "configurations": [
      {
        "classification": "export",
        "properties": {
          "AB": "${BC}",
          "HADOOP_CLIENT_OPTS1": "\-Xmx512m -Xms512m $
HADOOP_CLIENT_OPTS1\"
        }
      }
    ]
  },
  {
    "classification": "hive-site",
    "properties": {
      "hive.tez.java.opts": "-Xmx3900m"
    }
  },
  {
    "classification": "hive-exec-log4j",
    "properties": {
      "log4j.logger.org.apache.zookeeper.ClientCnxnSocketNIO
": "INFO,FA"
    }
  },
  {
    "classification": "hive-log4j",
    "properties": {
      "log4j.logger.org.apache.zookeeper.server.NIOServerCnxn
": "INFO,DRFA"
    }
  }
}

```

```

    }
  ]
}

```

The **Classification** parameter specifies the configuration file to modify. The **Properties** parameter stores the key-value pair that needs changing. If the default configuration file has a corresponding key, override the value or add the corresponding key-value pair.

The correlations between configuration files and classifications are shown in the following tables.

- Hadoop

| File Name | Classification |
|------------------------|--------------------|
| core-site.xml | core-site |
| log4j.properties | Hadoop-log4j |
| Hdfs-site.xml | hdfs-site |
| mapred-site.xml | mapred-site |
| yarn-site.xml | yarn-site |
| httpsfs-site.xml | httpsfs-site |
| capacity-scheduler.xml | capacity-scheduler |
| hadoop-env.sh | hadoop-env |
| httpfs-env.sh | httpfs-env |
| mapred-env.sh | mapred-env |
| yarn-env.sh | yarn-env |

- Pig

| File Name | Classification |
|------------------|----------------|
| pig.properties | pig |
| log4j.properties | pig-log4j |

- Hive

| File Name | Classification |
|----------------------------|-----------------|
| hive-env.sh | hive-env |
| hive-site.xml | hive-site |
| hive-exec-log4j.properties | hive-exec-log4j |
| hive-log4j.properties | hive-log4j |

The core-site and other flat .xml files only have one layer. All configurations are put in **Properties**. The hadoop-env and other .sh files may have two layers and can be set in the embedded configurations mode. See hadoop-env in the example where the `-Xmx512m -Xms512m` setting is added to the **HADOOP_CLIENT_OPTS** property of the export.

After setting, confirm and click **Next**.

8 Bootstrap actions

Bootstrap actions are used to run a customized script before the cluster starts Hadoop. The customized script is used to install your required third-party software or change the cluster operating environment.

Function

With bootstrap actions, you can install and perform a variety of things to your cluster that are not currently supported by clusters, such as:

- Install software with Yum.
- Directly download open software from a public network.
- Read your data from OSS.
- Install and operate a service, such as Flink or Impala. (The script for this is more complex.)

We strongly recommend that you test the bootstrap action with a Pay-As-You-Go cluster and create a Subscription cluster only after the test is successful.

Procedure

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. Select a region. The cluster associated with this region is listed.
3. Click **Create Cluster** to enter the cluster creation page.
4. At the bottom of the basic configuration page, click **Add** to enter the operation page.
5. Enter the configuration items.

You can add up to 16 bootstrap actions to be performed during cluster initialization in a specified sequence. By default, your customized script is run with the root account. Use `su hadoop` in the script to switch to a Hadoop account.

It is possible for a bootstrap action to fail, but this does not affect the creation of a cluster.

After a cluster is created successfully, you can view all abnormalities in the Bootstrap/software configuration column in the cluster details page. In the event of an abnormality, you can log on to all nodes to view the operation logs in the `/var/log/bootstrap-actions` directory.

Bootstrap action type

Bootstrap actions are categorized into customized bootstrap actions and operating-condition bootstrap action. The main difference is that the operating-condition bootstrap action can only operate your operation in nodes that meet the requirements.

Customized bootstrap action

For customized bootstrap actions, the position of the bootstrap action name and the execution script in OSS must be specified and, where necessary, optional parameters filled in. During cluster initialization, all nodes download the specified OSS scripts and runs them directly or after adding the optional parameters.

You can specify the files that need to be downloaded from OSS in the script. In the following example, the `oss://yourbucket/myfile.tar.gz` file is downloaded locally and extracted to the `/yourdir` directory:

```
#!/bin/bash
osscmd --id=<yourid> --key=<yourkey> --host=oss-cn-hangzhou-internal.
aliyuncs.com get oss://<yourbucket>/<myfile>.tar.gz ./<myfile>.tar.gz
mkdir -p /<yourdir>
tar -zxvf <myfile>.tar.gz -C /<yourdir>
```

The `osscmd` is preinstalled on the node and can be invoked directly to download the file.



Note:

OSS address host has an intranet address, an Internet address, and a VPC network address. If you use a classic network, you need to specify the intranet address. For Hangzhou, this is `oss-cn-hangzhou-internal.aliyuncs.com`. If you use a VPC network, you need to specify the domain name that the VPC intranet can access. For Hangzhou, this is `vpc100-oss-cn-hangzhou.aliyuncs.com`.

Bootstrap actions can install additional system software packages through Yum. The following example shows the installation of `ld-linux.so.2`:

```
#!/bin/bash
yum install -y ld-linux.so.2
```

Operating-condition bootstrap action

The execution script of an operating-condition bootstrap action is predefined. This means that you only need to specify its name and optional parameters. Operating-condition bootstrap actions must provide optional parameters, including operating conditions and commands (separated by spaces). The operating conditions support `instance.isMaster=true/false`, which specifies that it only operates on master or non-master nodes. The following example shows that the optional

parameters of an operating-condition bootstrap action are specified to only create a directory on a master node:

```
instance.isMaster=true mkdir -p /tmp/abc
```

If you need to specify multiple operating commands, you can divide them by using a semicolon (;).

For example: `instance.isMaster=true mkdir -p /tmp/abc;mkdir -p /tmp/def.`

9 VPC

Virtual Private Cloud (VPC) helps you build an isolated network environment, including customizing the IP address range, network segment, routing table, and gateway.

For more information, see [What is VPC?](#). VPC can be interconnected with physical IDC equipment rooms using [Express Connect](#).

Create a VPC cluster

When you create a cluster in E-MapReduce, you can select from two types of network: classic and VPC. If you select VPC, complete the following steps:

- Subordinate VPC: Select a VPC where the current E-MapReduce cluster is located. If you have not yet created a VPC, log on to the [VPC console](#) and create one.
- VSwitch: An ECS instance in the E-MapReduce cluster communicates through a VSwitch. If you have not yet created a VSwitch, log on to the [VPC console](#) and create one. Because a VSwitch has the properties of an availability zone, when you create a cluster in E-MapReduce, the VSwitch you create must also belong to the availability zone selected.
-
- Owner security group: The security group the cluster belongs to. Currently, only the security group of a VPC can be used, not the security group of a classic network. To ensure security, a security group created outside of E-MapReduce cannot be selected. To create a security group , select New security group and enter the name of security group.

Example

The following example shows how to enable Hive to access HBase clusters in E-MapReduce in different VPCs.

1. Create clusters.

Create two clusters in E-MapReduce. Hive cluster C1 is located in VPC1, whereas HBase cluster C2 is located in VPC2. Both clusters are located in the cn-hangzhou region.

2. Configure the high-speed channel.

For more information, see [Establish an intranet connection between VPCs under the same account](#). Select the same region.

3. Log on to the HBase cluster through SSH. A table is created through HBase Shell.

```
hbase(main):001:0> create 'testfromHbase','cf'
```

4. Log on to Hive through SSH.

- a. Modify the hosts and add the following line:

```
$zk_ip emr-cluster //$zk_ip is the zk node IP of Hbase cluster.
```

- b. Access HBase through Hive Shell.

```
hive> set hbase.zookeeper.quorum=172.16.126.111,172.16.126.112,172.16.126.113;
hive> CREATE EXTERNAL TABLE IF NOT EXISTS testfromHive (rowkey STRING, pageviews Int, bytes STRING) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES ('hbase.columns.mapping' = ':key,cf:c1,cf:c2') TBLPROPERTIES ('hbase.table.name' = 'testfromHbase');
```

At this point, the `java.net.SocketTimeoutException` exception is reported. This is because the security group where the HBase cluster's ECS is located limits access to E-MapReduce at the related port. By default, security groups created by E-MapReduce only open port 22. Therefore, a security group rule must be added to the HBase cluster's security group so as to open a port for the Hive cluster, as shown in the following figure.

| Authorization policy | Protocol type | Port range | Authorization type | Authorization object |
|----------------------|---------------|-------------|----------------------|----------------------|
| Allow | TCP | 2181/2181 | Address field access | 192.168.1.0/16 |
| Allow | TCP | 22/22 | Address field access | 0.0.0.0/0 |
| Allow | TCP | 16000/16000 | Address field access | 192.168.1.0/16 |
| Allow | TCP | 16020/16020 | Address field access | 192.168.1.0/16 |

10 Python instructions

This section provides an overview of different Python instructions.

Python 2.7

Python 2.7 is supported in E-MapReduce 2.0.0 and later. The files are located at `usr/local/Python-2.7.11/`. Numpy is included.

Python 3.6

Python 3.6.4 is supported in EMR 2.10.0, 3.10.0 and later. The files are located at `/usr/bin/python3`. Earlier versions do not support Python 3 by default. If you want to install it, complete the following steps:

- [Download the Python 3 package](#).
- Unzip the downloaded file using the following commands:

```
tar zxvf Python-3.6.4.tgz
cd Python-3.6.4 ./configure --prefix=/usr/local/Python-3.6.4
make && make install
ln -s /usr/local/Python-3.6.4/bin/python3.6 /bin/python3
ln -s /usr/local/Python-3.6.4/bin/pip3 /bin/pip3
```

- Verify the environment:

```
[root@emr-header-1 bin]# python3
```

```
Python 3.6.4 (default, Mar 12 2018, 14:03:26)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linuxType "help", "
copyright", "credits" or "license" for more information.
```

```
[root@emr-header-1 bin]# pip3 -V
```

```
pip 9.0.1 from /usr/local/Python-3.6.4/lib/python3.6/site-packages (
python 3.6)
```

If the preceding messages are displayed, you have successfully installed Python 3.

11 Open source components

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

**Note:**

Set security group rules for limited IP ranges. During configuration, you cannot open rules to 0.0.0.0/0.

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.

- Execute the following command:

```
/opt/apps/hue/build/env/bin/hue createsuperuser
```

- 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

- In the cluster list page, click **Manage** next to the target cluster.
- In the service list, click **Hue**, and then click the **Configuration** tab.
- 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:

```
$section_path.$real_key
```



Note:

- `$real_key` is the actual key to be added, such as `hive_server_host`.
- In the `hue.ini` file, you can view the `$section_path` before the `$real_key`.

For example, if the `hive_server_host` belongs to the `[beeswax]` section, this means that the `$section_path` is `beeswax`. If this is the case, the key to be added is `beeswax.hive_server_host`.

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

11.2 Oozie

The following section provides an overview of how to use Oozie.



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 [Connect to clusters using SSH](#).

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. 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. Enable Chrome to execute the following in the new terminal on the local machine:

```
/Applications/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: <https://oozie.apache.org/docs/4.2.0/WorkflowFunctionalSpec.html#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: <https://oozie.apache.org/docs/4.2.0/>.

- **Submit a workflow job on a non-HA cluster**

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

```
ssh root@publicIp_of_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]$ $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: 0000000-160627195651086-oozie-oozi-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_master_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 -copyFromLocal 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: 0000000-160627195651086-oozie-oozi-W
```

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

11.3 Presto

The following section provides an overview of how to use Presto.

E-MapReduce versions 2.0 and later support [Presto](#). Presto can be used in E-MapReduce by checking the Presto software box when you select a mirror image.

After you create a cluster, log on to the master node. The Presto software can be found in the `/usr/lib/presto-current` directory, and the PrestoServer processes can be viewed using the `jps` command.

Presto processes can be divided into coordinator and worker processes. The coordinator process is started on the master node (the HA cluster is the master node whose hostname starts with `emr-header-1`), and the worker process is started on the core node. The service process configuration can be found in the `/usr/lib/presto-current/etc` directory. Coordinator uses coordinator-

config.properties, whereas worker uses worker-config.properties. Other configuration files are shared. The web port is set as 9090.

By default, Presto services are supported by Hive. You can connect Hive's metastore on the cluster to read Hive table information and query it. The cluster is pre-installed with Presto CLI and can execute the following command to check Hive tables:

```
presto -server localhost:9090 -catalog hive -schema default -user
hadoop -execute 'show tables'
```

**Note:**

There is a delay of several seconds when Hive tables are synchronized.

11.4 Zeppelin

E-MapReduce can access Zeppelin through Apache Knox.

Preparation

1. In the [Security group](#) 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 guide](#). The user name and password are only used to log on to the various Knox services. They are not related to Alibaba Cloud RAM user names.

**Note:**

Set security group rules for limited IP ranges. During configuration, you cannot open rules to 0.0.0.0/0.

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**.

11.5 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.

11.6 Kafka

11.6.1 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. |
| Release (instances) on the console | Erased | The local ephemeral disk's storage volume is erased. Data is not retained. |



Note:

- 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. |

11.6.2 Cross-cluster Kafka access

An independent Kafka cluster is deployed to provide the Kafka service. Therefore, you may need to access this service across clusters.

Cross-cluster access to Kafka

Cross-cluster access to Kafka consists of two types:

- Accessing E-MapReduce Kafka clusters from the Alibaba Cloud intranet network.
- Accessing E-MapReduce Kafka clusters from the public network.

Different solutions are prepared for different E-MapReduce versions.

EMR-3.11.x and later

- Access Kafka from the Alibaba Cloud intranet network

You can access Kafka by using the intranet IP address of a Kafka cluster node. Use port 9092 to access Kafka from the intranet network.

Make sure that the networks are accessible before you access Kafka:

- For more information about how to access a VPC from a classic network, see [Access between classic network and VPC](#) here.
- For more information about how to access a VPC from another VPC, see [Configure a VPC-to-VPC connection](#).
- Access Kafka in the public network

The core node of the Kafka cluster is unable to access the public network by default. To access the Kafka cluster in the public network, complete the following steps:

1. Interconnect Kafka clusters with the public network.

- If Kafka clusters are deployed in a VPC environment, there are two ways to interconnect them:
 - Deploy Express Connect to interconnect the VPC with the public network. For details, see [Express Connect](#).
 - Bind EIPs to cluster core nodes. For details, see [EIP](#). Complete the following steps to bind the EIP to the ECS:
- If Kafka is deployed in a classic network, there are two ways to interconnect them:
 - To create a Pay-As-You-Go cluster, use ECS APIs. For details, see [API](#).
 - To create a Subscription cluster, you can directly assign a public IP address to the relevant host in the ECS console.

2. Create an EIP in the [VPC console](#) and purchase the relevant EIPs based on the number of core nodes in the Kafka cluster.

3. Configure security group rules that allow the Kafka cluster to control public network access to the cluster's IP addresses. This improves the security of the Kafka cluster exposed in the public network. You can view the security group to which the cluster belongs in the E-MapReduce console, and configure security group rules based on security group IDs. For more information, see [Security group rules](#).

4. On the **Cluster Management** page of the E-MapReduce console, click **Manage** next to the specified cluster, select **Cluster Overview** on the left side of the page, and then click **Sync Cluster Host Info** in the upper-right corner.

5. Restart the Kafka cluster.

6. Use the EIP of the Kafka cluster node to access Kafka in the public network. Use port 9093 to access Kafka from the public network.

Versions earlier than EMR-3.11.x

- Access Kafka from the Alibaba Cloud intranet network

You must configure the host information of the Kafka cluster node on the client host. The **Long domain** of the Kafka cluster node must also be configured. For 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 in the public network

The core node of the Kafka cluster is unable to access the public network by default. To access the Kafka cluster in the public network, complete the following steps:

1. Interconnect Kafka clusters with the public network.

- If Kafka clusters are deployed in a VPC environment, there are two ways to interconnect them:
 - Deploy Express Connect to interconnect the VPC with the public network. For details, see [Express Connect](#).
 - Bind EIPs to cluster core nodes. For details, see [EIP](#). Complete the following steps to bind the EIP to the ECS.
- If Kafka is deployed in a classic network, there are two ways to interconnect them:
 - To create a Pay-As-You-Go cluster, use ECS APIs. For details, see [API](#).
 - To create a Subscription cluster, you can directly assign a public IP address to the relevant host in the ECS console.

2. Create an EIP in the [VPC console](#) and purchase the relevant EIPs based on the number of core nodes in the Kafka cluster.

3. Configure security group rules that allow the Kafka cluster to control public network access to the cluster's IP addresses. This improves the security of the Kafka cluster exposed in the public network. You can view the security group to which the cluster belongs in the E-MapReduce console, and configure security group rules based on security group IDs. For more information, see [Security group rules](#).

4. Modify the Kafka cluster's `listeners.address.principal` software configuration to `HOST`, and restart the Kafka cluster.

5. Configure the `hosts` file on the local client host.

11.6.3 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**.

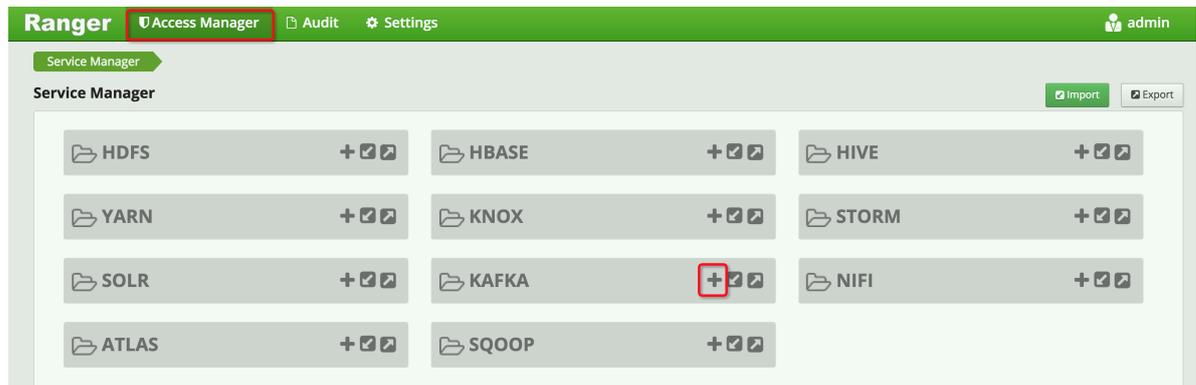
3. You can check the progress by clicking **View Operation History** in the upper-right corner of the page.



- **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:

Ranger Access Manager Audit Settings admin

Service Manager Create Service

Create Service

Service Details :

Service Name * emr-kafka

Description

Active Status Enabled Disabled

Select Tag Service Select Tag Service

Config Properties :

Username * kafka

Password * ****

Zookeeper Connect String * emr-header-1.2181/kafka-1.0.1

Ranger Plugin SSL CName

Add New Configurations

| Name | Value |
|------|-------|
| | |

+

Test Connection

Add Cancel

Configure permissions

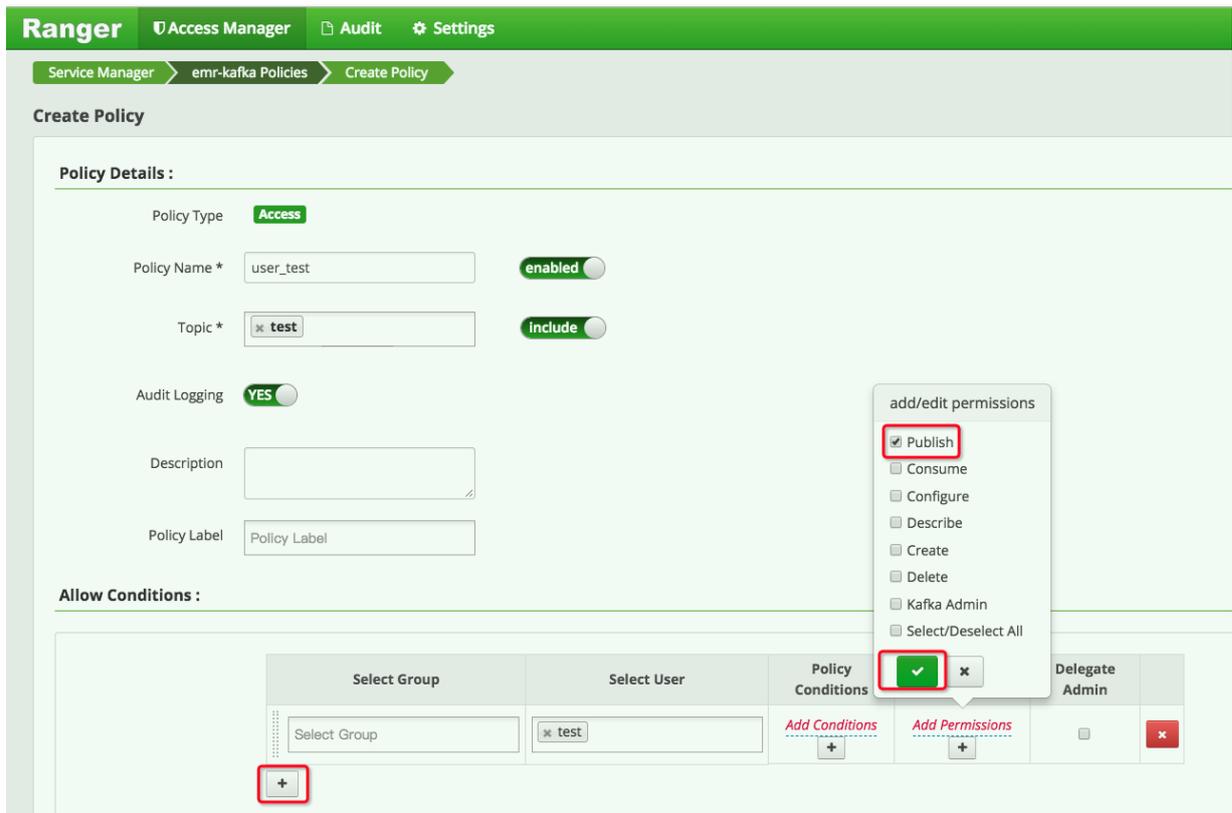
After integrating Ranger into Kafka, you can set the relevant permissions.



Note:

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.

11.6.4 Kafka SSL

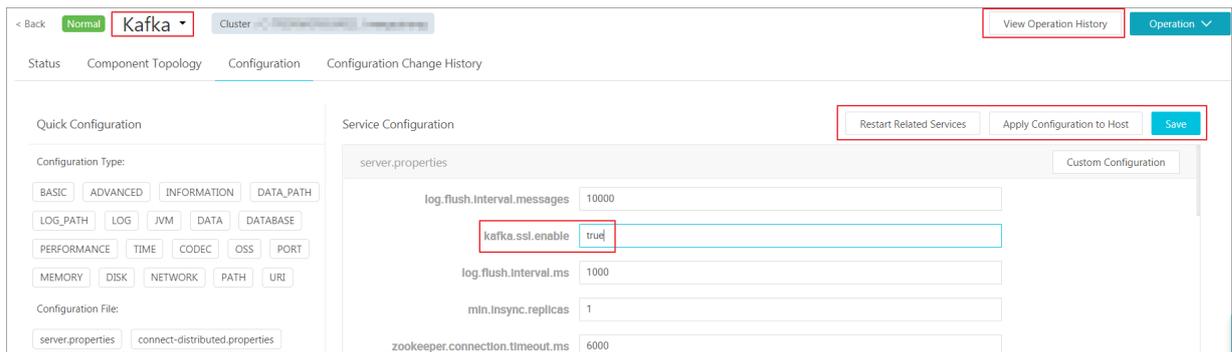
E-MapReduce Kafka supports the SSL function in E-MapReduce 3.12.0 and later.

Create a cluster

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

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
```

11.6.5 Kafka Manager

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

Procedure



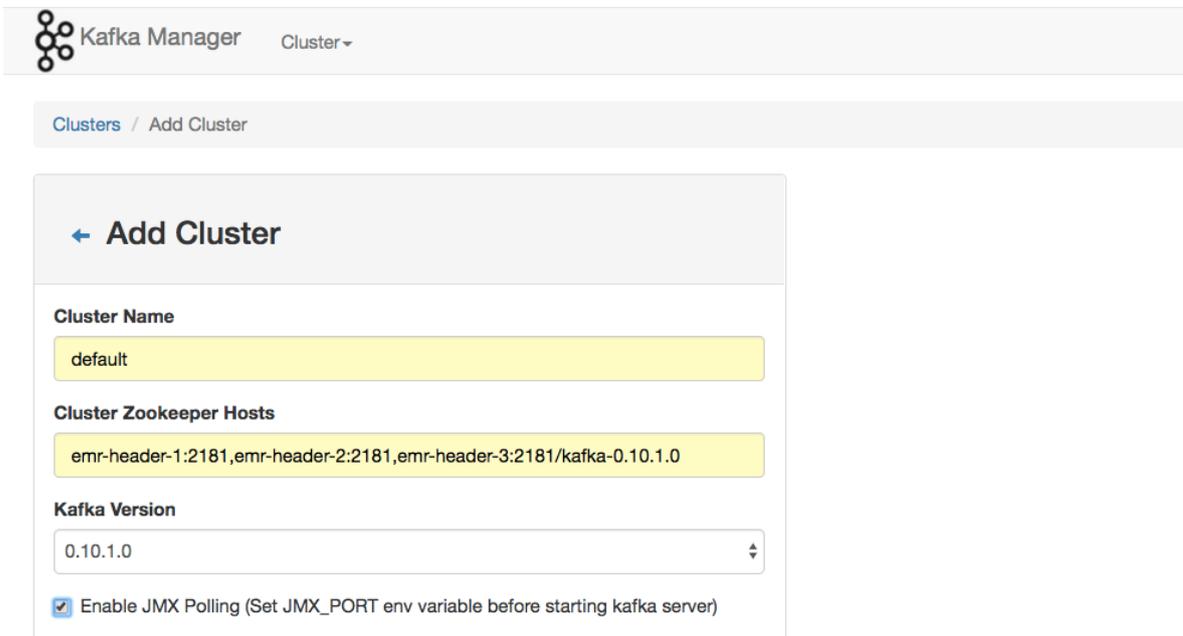
Note:

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.

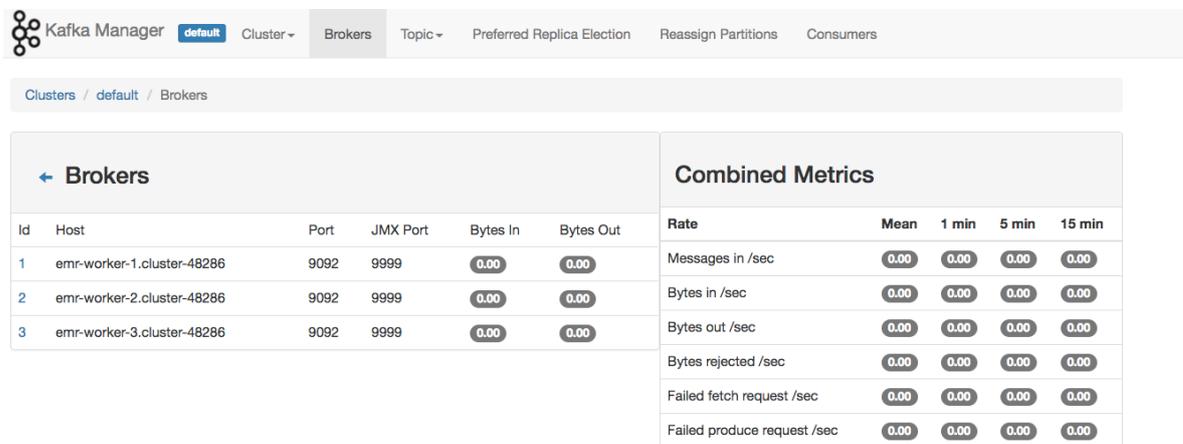
- We recommend that you access the web page through the SSH tunnel. For more information, see [Connect to clusters using SSH](#).
- Access *Cite Left*`http://localhost:8085`*Cite Right*.
- Enter your user name and password. Refer to the configuration information of Kafka Manager.

| application.conf | |
|---------------------------------------|--|
| kafka_manager_authentication_enabled | <input type="text" value="true"/> |
| kafka_manager_zookeeper_hosts | <input type="text" value="emr-header-1:2181,emr-header-2:2181,emr-header-3:2181"/> |
| kafka_manager_authentication_username | <input type="text" value="me"/> |
| kafka_manager_authentication_password | <input type="text" value=""/> |

- 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.



11.6.6 Common Kafka issues

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
```

11.7 Druid

11.7.1 Introduction to Druid

Druid is a column-oriented, open-source, distributed data store used to query and analyze issues in large data sets 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:

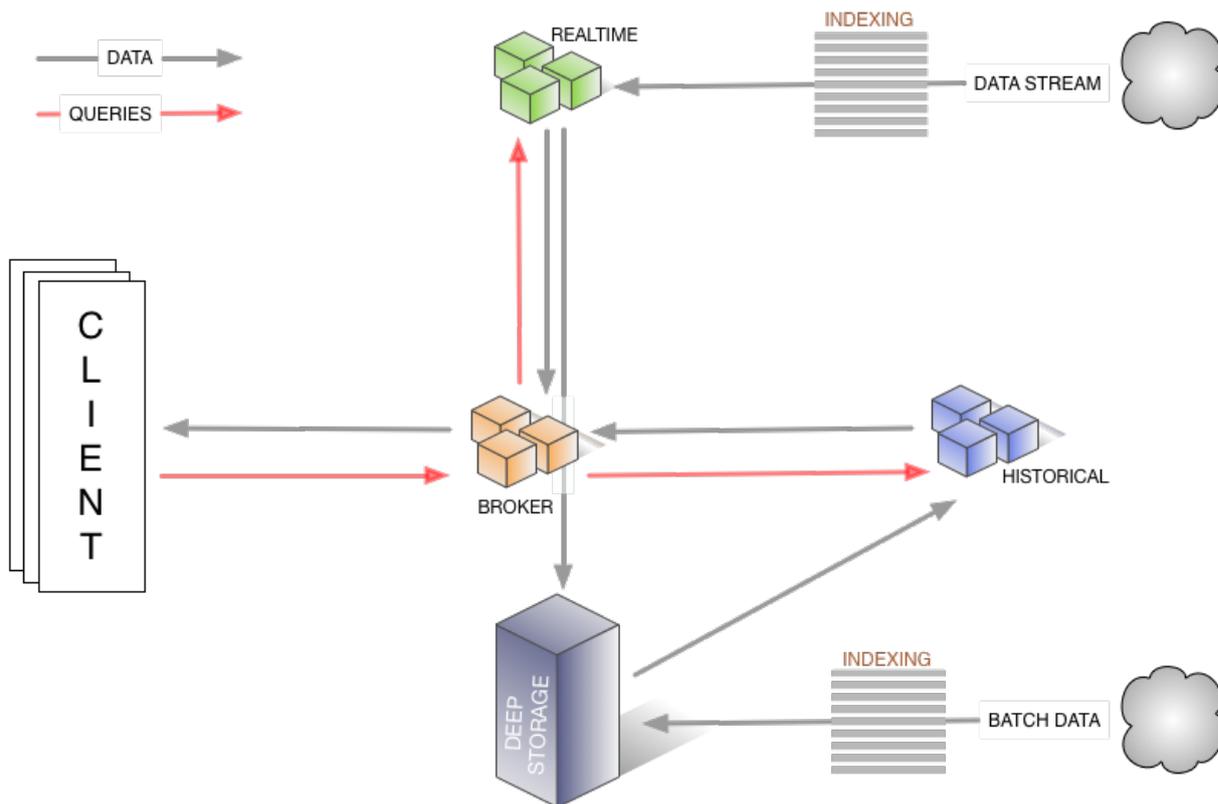
- Real-time indicator monitoring
- Model recommendations
- Advertisement platforms
- Model searches

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.

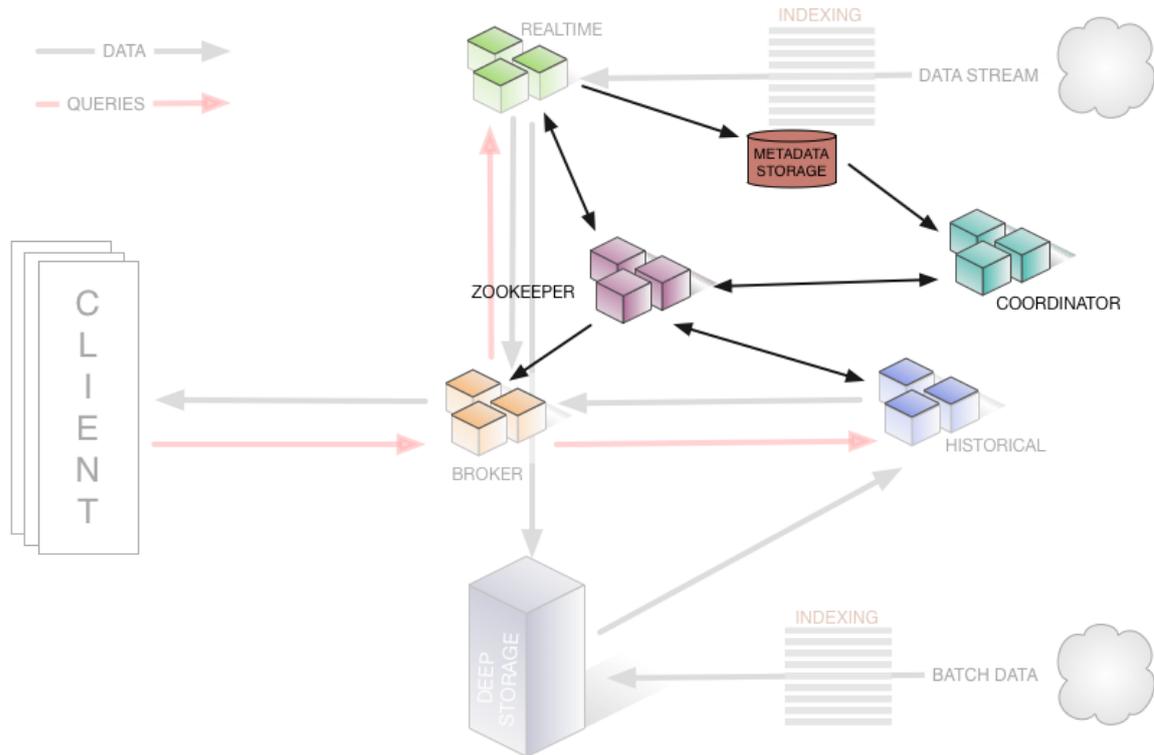
The following figure shows the components contained in the Druid working-layer (for data indexing and data querying).



- 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
- 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

11.7.2 Quick start

E-MapReduce 3.11.0 and later 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 the Druid cluster type when you create a cluster. You can check 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 strongly 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-xxxx: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.storageDirectory: (such as 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"? >
  <configuration>
    <property>
      <name>fs.oss.impl</name>
      <value>com.aliyun.fs.oss.nat.NativeOssFileSystem</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>
</configuration>

```

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`.

Click the **Save, Configuration to Host**, and **Restart Related Services** buttons in turn in the upper right corner 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.properteis`). 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).
```

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 a 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/druid-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 iZbp1h9g7boqo9x23qbifiZ
```

```

192.168.143.0    emr-worker-2.cluster-1234 emr-worker-2 emr-header-
3.cluster-1234 emr-header-3 iZbp1eaa5819tkjx55yr9xZ
192.168.142.254 emr-header-1.cluster-1234 emr-header-1 iZbp1e3zwu
vnmakmsjer2uZ
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 iZbp195rj7zvx8qar4f6b0Z
192.168.143.7    emr-worker-2.cluster-5678 emr-worker-2 emr-header-3.
cluster-5678 emr-header-3 iZbp15vy2rsxoegki4qhdpZ
192.168.143.5    emr-header-1.cluster-5678 emr-header-1 iZbp10tx4e
gw3wfnh5oiilZ

```

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/druid-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.authentication.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 details, see [Cross-region access](#) here.)
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 has an example named wikticker and located in `${DRUID_HOME}/quickstart`. (`${DRUID_HOME}` is `/usr/lib/ druid-current` by default.) Each line of the wikticker document (`wikticker-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",
 "countryIsoCode": 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/wikticker-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. Modify Druid cluster ``${DRUID_HOME}/quickstart/wikiticker-index.json`, as shown below:

```
{
  "type" : "index_hadoop",
  "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",
      "granularitySpec" : {
        "type" : "uniform",
        "segmentGranularity" : "day",
        "queryGranularity" : "none",
        "intervals" : ["2015-09-12/2015-09-13"]
      },
      "parser" : {
        "type" : "hadoopyString",
        "parseSpec" : {
          "format" : "json",
          "dimensionsSpec" : {
            "dimensions" : [
              "channel",
              "cityName",
              "comment",
              "countryIsoCode",
              "countryName",
              "isAnonymous",
              "isMinor",
              "isNew",
              "isRobot",
              "isUnpatrolled",
              "metroCode",
              "namespace",
              "page",
              "regionIsoCode",
              "regionName",
              "user"
            ]
          },
          "timestampSpec" : {
            "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" : 500000
  },
  "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 `hadoop`.
- **tuningConfig.jobProperties** sets the classloader of the mapreduce job.
- **hadoopDependencyCoordinates** 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:application/json' -d @quickstart/wikiticker-index.
json http://emr-header-1.cluster-1234:18090/druid/indexer/v1/task

```

Note that the items such as `--negotiate`, `-u`, `-b`, `-c` are for high-security mode Druid clusters. The Overlord port number is 18090 by default.

4. View the running state of the jobs.

Access `http://emr-header-1.cluster-1234:18090/console.html` in the browser to view how the jobs run. To access the page properly, you need to open an SSH tunnel in advance (see [Connect to clusters using SSH](#) View WebUI of system such as Hadoop, Spark, Ganglia section in SSH tunnel), and start an agent chrome. If the high-security mode is enabled for the Druid cluster, you have to configure your browser to support the Kerberos authentication process. For more information, see [here](#).

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 wikticker data is as follows

`${DRUID_HOME}/quickstart/wikiticker-top-pages.json`):

```
{
  "queryType" : "topN",
  "dataSource" : "wikticker",
  "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:application/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

We recommend that you use [Tranquility client](#) to send real-time data to Druid. Tranquility supports sending data to Druid in a variety of ways, such as Kafka, Flink, Storm, Spark Streaming. For the information about Kafka method, see [Tranquility](#). Druid uses Tranquility Kafka section in Tranquility. For more information about how to use Tranquility and SDK, see [Tranquility Help Document](#).

For Kafka, you can also use the kafka-indexing-service extension. For details, see [Kafka indexing service](#).

- Troubleshoot index failures

When the index fails, troubleshoot the failure as follows:

- **For the index of batch data**

1. If the curl command output displays an error or does not display any information, check the file format. Or add the `-v` parameter to the curl command to check the value returned from the REST API.
2. Observe the execution of jobs on the Overlord page. If the execution 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`). 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 to view the Middlemanager logs (at `/mnt/disk1/log/druid/middleManager-emr-header-1.cluster-xxxx.log`).

- For real-time index of Tranquility

View the Tranquility log to check whether the message was received or dropped.

The remaining troubleshooting steps are the same as two-five of batch index.

Most of the 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).

11.7.3 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 to be indexed and how that data format is indexed by Druid. It is a JSON file, which consists of three parts:

```
{
  "dataSchema" : {...},
  "ioConfig" : {...},
  "tuningConfig" : {...}
}
```

| 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 | Describe 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 | Configure 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:

```
{
  "dataSrouce": <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.

For the parser, there are two types: 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 (which 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 |
| format | String | The timestamp type can be "ISO", "millis", "POSIX", "auto", or that 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. In addition, you can 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 by the authority:

| 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 |
| javascript | javascript |
| cardinality | cardinality |
| hyperUnique | hyperUnique |



Note:

The last three are the 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 default option at present.

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

ioConfig

ioConfig describes the data source. Here is an example of Hadoop index:

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

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

Tuning Config

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

11.7.4 Tranquility

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 usage of Druid for users. It supports a wide range of data sources, including Samza, Spark, Storm, Kafka, and Fink. This section uses Kafka as an example, and describes how to use Tranquility in the EMR to capture data from the Kafka cluster and push the data to the Druid cluster in real time.

Interaction with the Kafka cluster

The first is the interaction between the Druid cluster and 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, perform the following steps:

1. Ensure the communication between 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 the two 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 in the form of a long name, such as `emr-header-1.cluster-xxxxxxx`.

For high-security mode Kafka clusters, you need to perform 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 the two 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 in the form of a long name, such as `emr-header-1.cluster-xxxxxxx`.
3. Set Kerberos cross-domain mutual trust between the two clusters. (For details, see [Cross-region access](#) here. The bidirectional mutual trust is preferred.
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 to a specific directory such as `/tmp/kafka/kafka_client_jaas.conf`.

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

```
Add Djava.security.auth.login.config=/tmp/kafka/kafka_client_jaas.conf
```

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

```
druid.indexer.runner.javaOpts=-Djava.security.auth.login.config=/tmp/kafka/kafka_client_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 cluster and the Druid cluster 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_client_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 --replication-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:

```
{
  "dataSources" : {
    "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/tranquility kafka -configFile
```

5. Start the producer and configure it to send some 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 view specific information in Tranquility log. At the same time, the corresponding real-time indexing task has been started on the Druid side.

11.7.5 Kafka indexing service

Kafka indexing service is a plug-in launched by Druid to consume Kafka data in real time using druid's indexing service. The plug-in will enable a supervisor in Overlord. The supervisor starts some indexing tasks in Middlemanager after it starts. These tasks will connect to the Kafka cluster to consume the topic data and complete the index creation. You need to prepare a data consumption format file and manually start the supervisor through the REST API.

Interaction with the Kafka cluster

See the introduction in [Tranquility](#).

Use Druid Kafka indexing service to consume 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 parmname `—zookeeper` parameter can be changed according to your actual configuration.

2. Define the data format description file for the data source. You can name it as `metrics-kafka.json` and place it in the current directory (or another directory that you 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-xxxxxxx:
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 (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 options `--negotiate`, `-u`, `-b`, and `-c` 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
>
```

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

5. Enter some 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" : ["2018-03-02T00:00:00.000/2018-03-08T00:00:00.000
"],
  "granularity" : "all",
  "searchDimensions": [
```

```

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

```

7. Execute the query on the master node of 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 options `--negotiate`, `-u`, `-b`, and `-c` are for high-security mode Druid clusters.

8. You will see a query result similar to the following in normal cases.

```

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

```

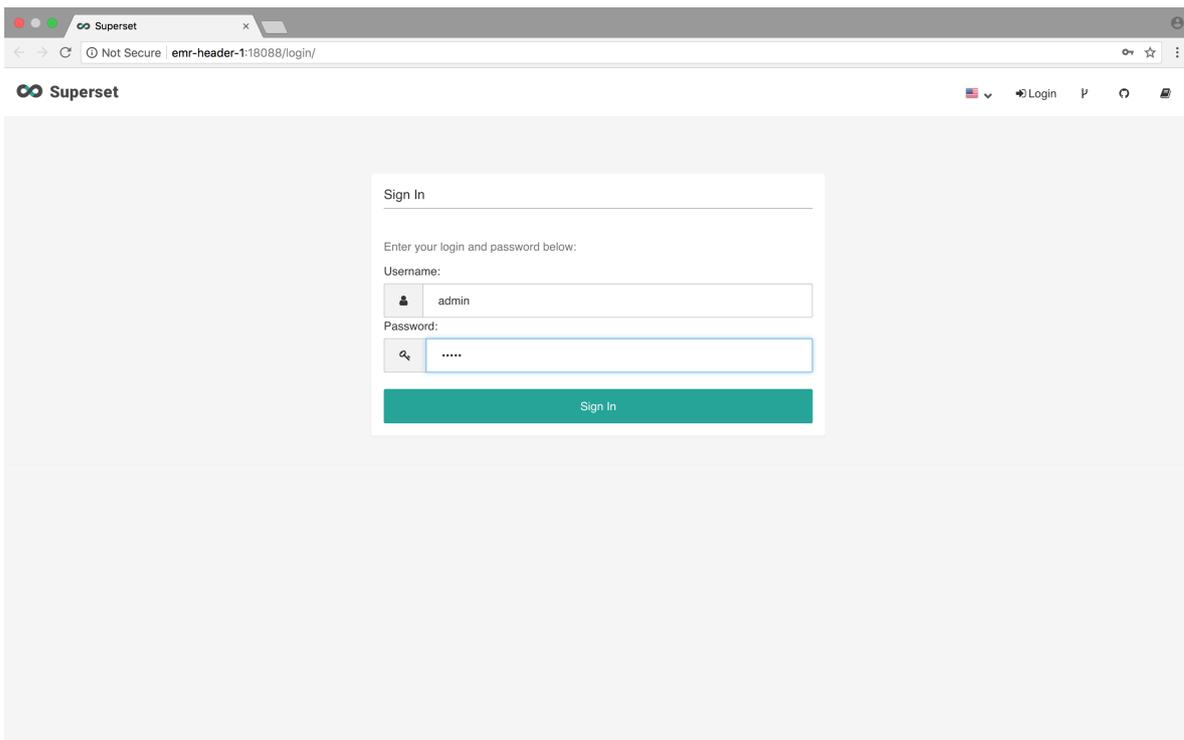
11.7.6 Superset

The Druid cluster integrates the Superset tool. Superset provides deep integration to Druid, and supports a variety of relational databases. Because Druid supports SQL, you can access Druid in two ways through Superset, that is, Druid 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](#).

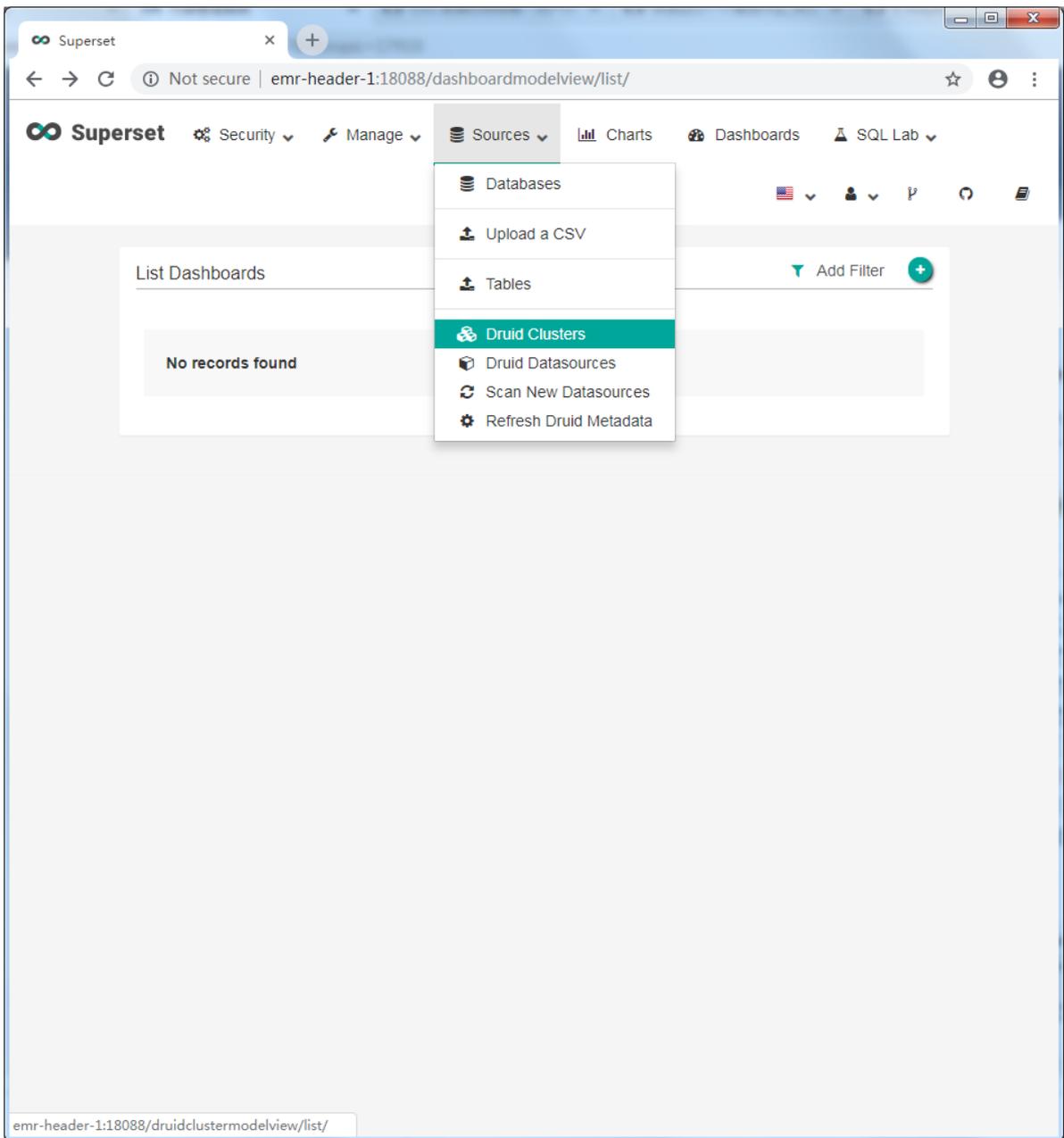
1. Log on to the Superset.

Visit `http://emr-header-1:18088` in the browser to enter the Superset logon page. The default username is `admin`, and the default password is `admin`. Change your password upon logon.



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 top menu bar, 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 , the open source broker port number is 8082, and the port number in E-MapReduce is port 18082.

Superset Security Manage Sources Charts Dashboards SQL Lab

Add Druid Cluster

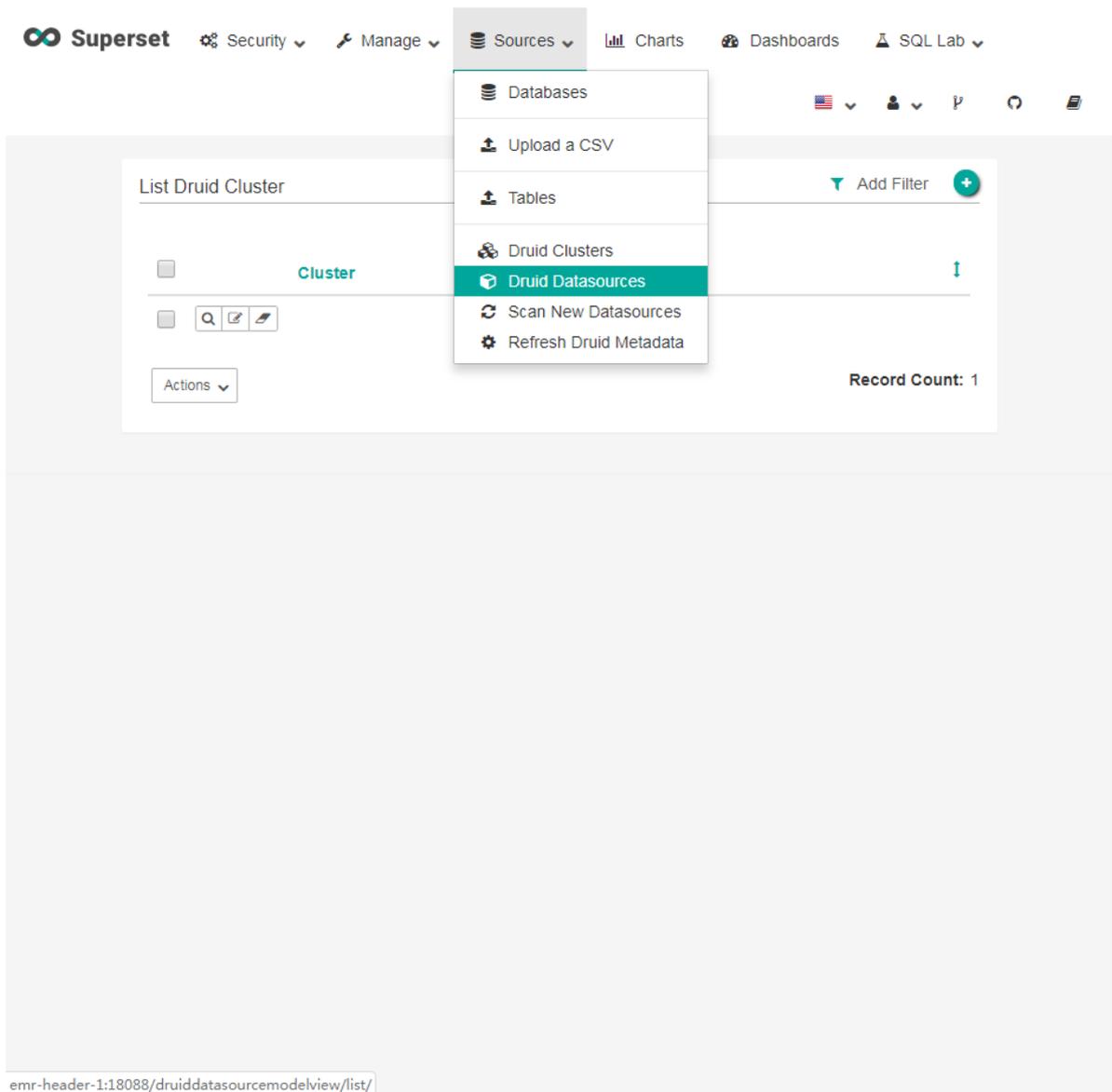
| | |
|----------------------|-------------------------------|
| Verbose Name | my druid cluster |
| Coordinator Host | emr-header-1 |
| Coordinator Port | 18081 |
| Coordinator Endpoint | druid/coordinator/v1/metadata |
| Broker Host | emr-header-1 |
| Broker Port | 18082 |
| Broker Endpoint | druid/v2 |
| Cache Timeout | Cache Timeout |
| Cluster | Cluster |

Save ←

3. Refresh or add a new data source.

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

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



You need to fill in necessary information for custom data source, and then save it.

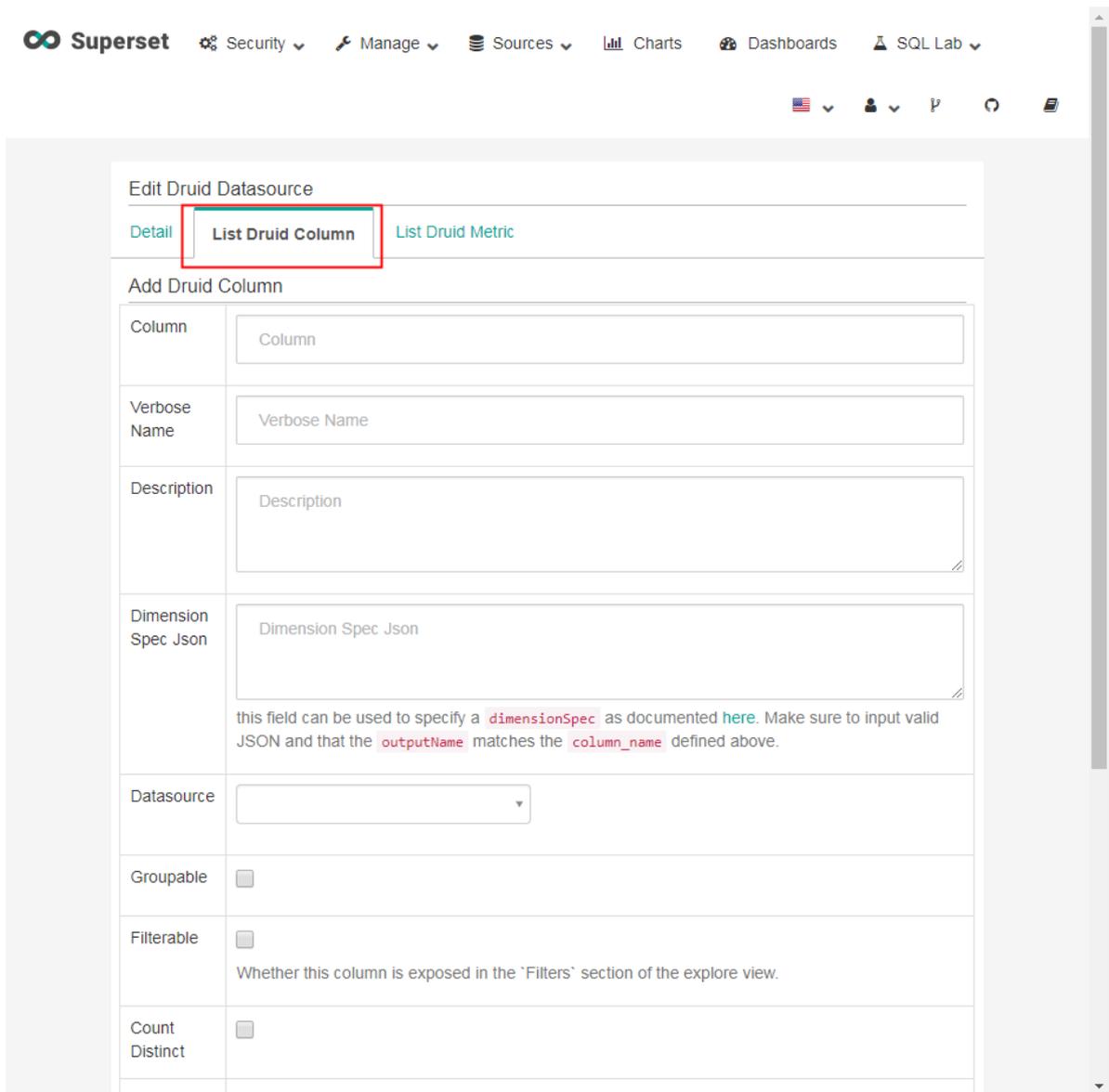
Superset Security Manage Sources Charts Dashboards SQL Lab

US User Profile Refresh Logout

Add Druid Datasource

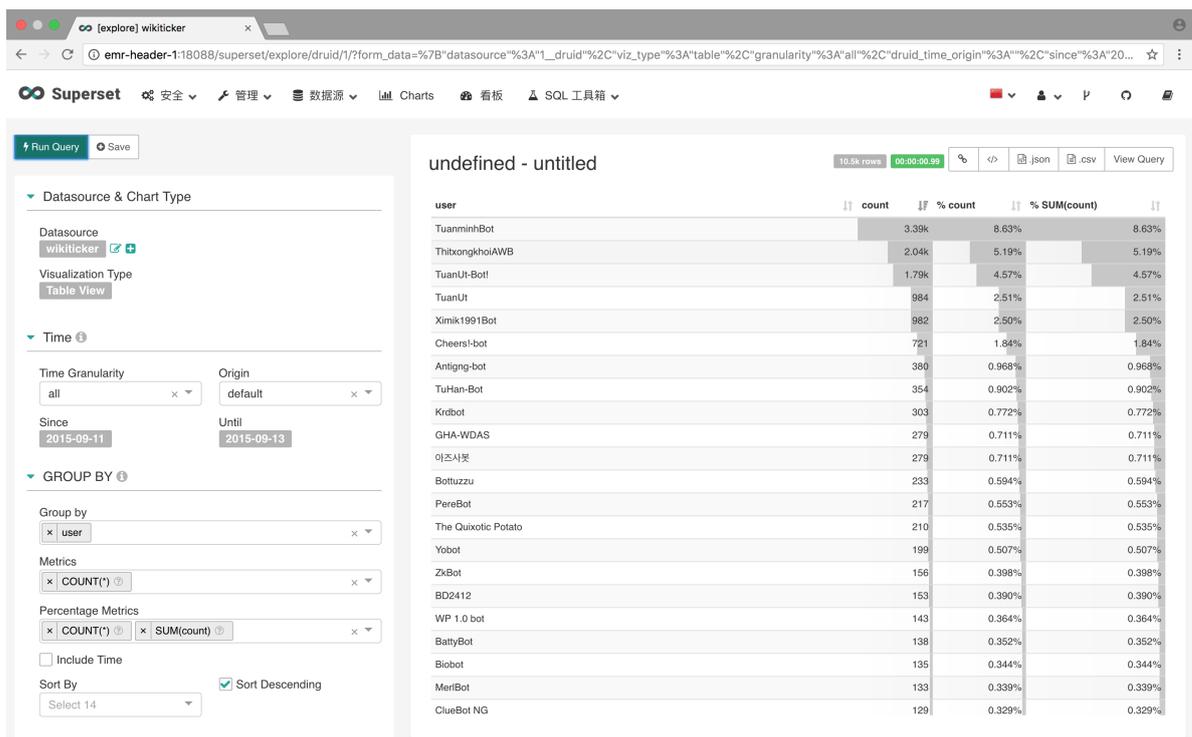
| | |
|----------------------|--|
| Datasource Name | <input type="text" value="wikiticker"/> |
| Cluster | <input type="text" value="my druid cluster"/> |
| Description | <input type="text" value="Description"/>
<small>Supports markdown</small> |
| Owner | <input type="text" value="admin admin"/> |
| Is Hidden | <input type="checkbox"/> |
| Enable Filter Select | <input checked="" type="checkbox"/>
<small>Whether to populate the filter's dropdown in the explore view's filter section with a list of distinct values fetched from the backend on the fly</small> |
| Fetch Values From | <input type="text" value="Fetch Values From"/>
<small>Time expression to use as a predicate when retrieving distinct values to populate the filter component. Only applies when 'Enable Filter Select' is on. If you enter '7 days ago', the distinct list of values in the filter will be populated based on the distinct value over the past week</small> |
| Default Endpoint | <input type="text" value="Default Endpoint"/>
<small>Redirects to this endpoint when clicking on the datasource from the datasource list</small> |

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



4. Query Druid.

After the data source has been added successfully, click the data source 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.

| database | pypi package | SQLAlchemy URI prefix |
|------------|--|--|
| MySQL | <code>pip install mysqlclient</code> | <code>mysql://</code> |
| Postgres | <code>pip install psycopg2</code> | <code>postgresql+psycopg2://</code> |
| Presto | <code>pip install pyhive</code> | <code>presto://</code> |
| Oracle | <code>pip install cx_Oracle</code> | <code>oracle://</code> |
| sqlite | | <code>sqlite://</code> |
| Redshift | <code>pip install sqlalchemy-redshift</code> | <code>redshift+psycopg2://</code> |
| MSSQL | <code>pip install pymssql</code> | <code>mssql://</code> |
| Impala | <code>pip install impyla</code> | <code>impala://</code> |
| SparkSQL | <code>pip install pyhive</code> | <code>jdbc+hive://</code> |
| Greenplum | <code>pip install psycopg2</code> | <code>postgresql+psycopg2://</code> |
| Athena | <code>pip install "PyAthenaJDBC>1.0.9"</code> | <code>awsathena+jdbc://</code> |
| Vertica | <code>pip install sqlalchemy-vertica-python</code> | <code>vertica+vertica_python://</code> |
| ClickHouse | <code>pip install sqlalchemy-clickhouse</code> | <code>clickhouse://</code> |
| Kylin | <code>pip install kylinpy</code> | <code>kylin://</code> |

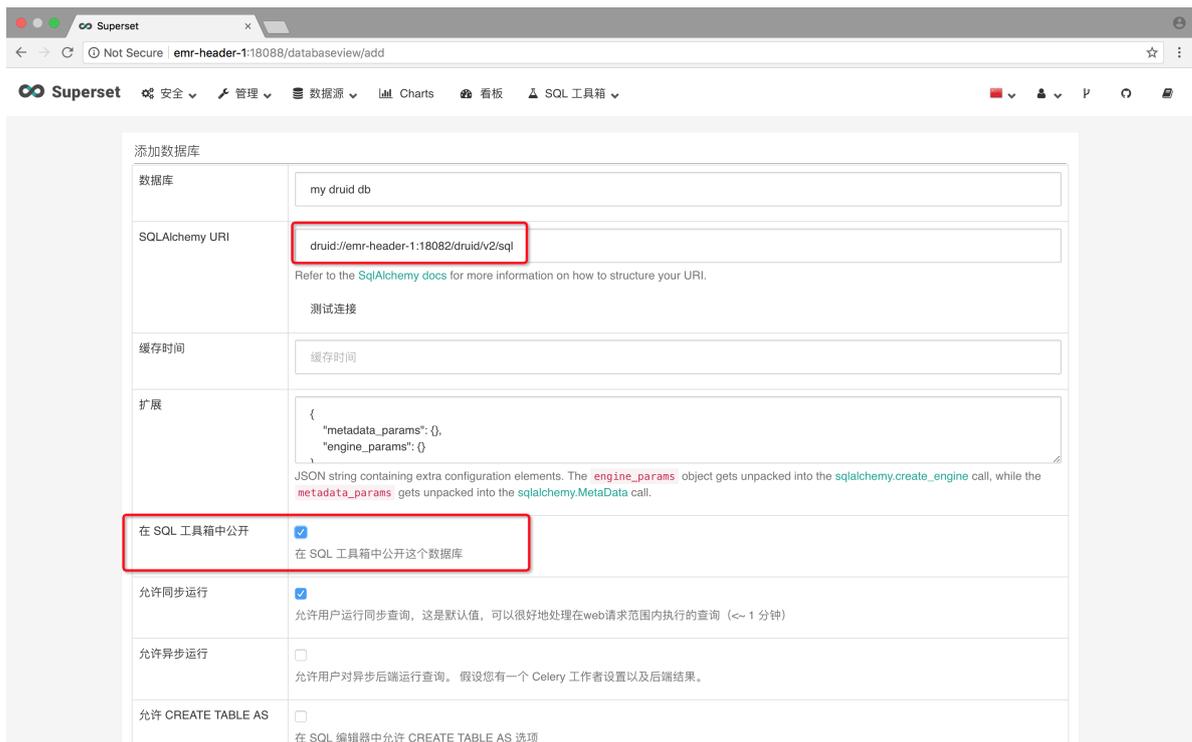
Superset also supports access to Druid in this way. The corresponding SQLAlchemy URI of Druid is “druid://emr-header-1:18082/druid/v2/sql”. When you add Druid as a database, check the Expose the database in the SQL toolkit check box.

The screenshot shows the 'Add Database' configuration page in Superset. The form contains the following fields and options:

- Database:** my druid db
- SQLAlchemy URI:** druid://emr-header1:18082/druid/v2/sql (highlighted with a red box)
- Cache Timeout:** Cache Timeout
- Extra:** A text area containing a JSON string:

```
{ "metadata_params": {}, "engine_params": {} }
```
- Expose in SQL Lab:** (highlighted with a red box)
- Allow Run Sync:**
- Allow Run Async:**
- Allow CREATE:**

Then you can use SQL to query in the SQL toolkit:



11.7.7 Druid common problems

This section describes Druid common problems you may meet

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. If it is a Hadoop job, enter the Yarn page to see if there is an indexing 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 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 2-5 of batch data index.

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).

Obtain the FAQ list

- Service startup fails.

Most of these problems are due to the configuration problems with the component JVM running parameters. 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.

Solution: View the component logs and adjust the relevant parameters to resolve these issues. JVM memory involves heap memory and direct memory. For more information, see <http://druid.io/docs/latest/operations/performance-faq.html>.

- The Yarn task fails during the indexing process, and shows Jar package conflict error like this:

```
Error: class com.fasterxml.jackson.datatype.guava.deser.HostAndPortDeserializ
er overrides final method deserialize.(Lcom/fasterxml/jackson/core/JsonPar
ser;Lcom/fasterxml/jackson/databind/DeserializationContext;)Ljava/lang/Obj
ect;.
```

Solution: Add the following content to the job configuration file of indexing:

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

The parameter `mapreduce.job.classloader` allows MapReduce job to use a standalone classloader, 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 details about this, see <http://druid.io/docs/0.9.2-rc1/operations/other-hadoop.html>.

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

Solution:

- Check the settings for deep storage, including type and directory. When the type is local, pay attention to the permission settings of directory. When the type is HDFS, directory should be written as the full HDFS path, such as `hdfs://:9000/`. For `hdfs_master`, IP is preferred. If you want to use a domain name, then use the full domain name, such as `emr-header-1.cluster-xxxxxxx` rather than `emr-header-1`.
- When you use Hadoop for batch index, you must set the deep storage of segments as “`hdfs`”. The “`local`” type may cause the MR job to be in an unidentified state, because the remote Yarn cluster cannot create the segments directory in the reduce task. (This is for standalone Druid clusters.)

- Failed to create directory within 10000 attempts.

This issue occurs typically because the path set by `java.io.tmp` in the JVM configuration file doesn't exist. Set the path and make sure that the Druid account has the 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 Tranquility settings includes `/druid`. (Two ZooKeeper settings exist in the Tranquility Kafka settings. 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 the indexing process.

This is because the Hadoop cluster of EMR is configured with lzo compression.

Solution: Copy the jar package and the native file under the directory of EMR `HADOOP_HOME/lib` to `druid.extensions.hadoopDependenciesDir` (`DRUID_HOME/hadoop-dependencies` by default) of Druid.

- The following error is reported during the indexing process:

```
2018-02-01T09:00:32,647 ERROR [task-runner-0-priority-0] com.hadoop.compression.lzo.GPLNativeCodeLoader - could not unpack the binaries
```

```

java.io.IOException: No such file or directory
    at java.io.UnixFileSystem.createFileExclusively(Native
Method) ~[?:1.8.0_151]
    at java.io.File.createTempFile(File.java:2024) ~[?:1.8.
0_151]
    at java.io.File.createTempFile(File.java:2070) ~[?:1.8.
0_151]
    at com.hadoop.compression.lzo.GPLNativeCodeLoader.
unpackBinaries(GPLNativeCodeLoader.java:115) [hadoop-lzo-0.4.21-
SNAPSHOT.jar:?]

```

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

11.8 Presto

11.8.1 Connector

System connector

- Overview

SQL can be used to query the basic information and measurements of the Presto cluster through the connector.

- Configuration

All information can be obtained through a catalog known as `system` without configuration.

Examples

```

--- List all supported data entries
SHOW SCHEMAS FROM system;
--- List all data entries in the project during runtime
SHOW TABLES FROM system.runtime;
--- Obtain node status
SELECT * FROM system.runtime.nodes;
   node_id | http_uri | node_version | coordinator
   | state
-----+-----+-----
+-----+-----+-----
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
--- Force cancel a query
CALL system.runtime.kill_query('20151207_215727_00146_tx3nr');

```

- Data tables

The connector provides the following data tables:

| TABLE | SCHEMA | DESCRIPTION |
|-------------------|----------|--|
| catalogs | metadata | This table contains the list of all catalogs supported in the connector |
| schema_properties | metadata | This table contains the list of available properties that can be set when creating a Schema. |
| table_properties | metadata | This table contains the list of available properties that can be set when creating a table. |
| nodes | runtime | This table contains the list of all visible nodes and statuses thereof in the Presto cluster. |
| queries | runtime | This table contains information queries currently and recently initiated in the Presto cluster, including the original query texts (SQL), identities of the users who initiate the queries and information on query performances, for example, query queue and analysis time, etc. |
| tasks | runtime | This table contains information on the task involved in the queries in the Presto, including the locations and numbers of lines and bytes processed in each task. |
| transactions | runtime | This table contains the list of currently opened transactions and related metadata. The data includes information like creation time, idle time, initiation parameters, and access catalogs. |

- Stored procedures

The connector supports the following stored procedures:

`runtime.kill_query(id)` Cancel query from specified ID.

JMX connector

- Overview

JMX information for all nodes in the Presto cluster can be queried through the JMX connector. The connector is generally used for system monitoring and debugging. Regular dump of JMX information can be implemented through modifying the configuration of the connector.

- Configuration

Create a file `etc/catalog/jmx.properties`, add the following content, and enable JMX connector.

```
connector.name=jmx
```

If regular dump of JMX data is expected, the following content can be added in the configuration file:

```
connector.name=jmx
jmx.dump-tables=java.lang:type=Runtime,com.facebook.presto.execution.scheduler:name=NodeScheduler
jmx.dump-period=10s
jmx.max-entries=86400
```

Where:

- **dump-tables** is a list of MBeans (Managed Beans) separated with commas. This configuration specifies which MBeans is sampled and stored in the memory for each sample period.
- **dump-period** is used for setting the sample period, which is 10s by default.
- **max-entries** is used for setting the max length of the history, which is 86400 by default.

If the name of a metric contains a comma, it must be escaped 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

JMX connector provides 2 schemas, **current** and **history**. Where:

current contains the current MBean in each node, the name of which is the table name in **current** (if the bean name contains non-standard characters, the table name must be in quotation marks for the query), which can be obtained through the following statement:

```
SHOW TABLES FROM jmx.current;
```

Examples

```
--- Obtain jvm information for 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 rows)

--- Obtain the metrics of max and min file descriptor counts for
each node
SELECT openfiledescriptorcount, maxfiledescriptorcount
FROM jmx.current."java.lang:type=operatingsystem";
  openfiledescriptorcount | maxfiledescriptorcount
-----+-----
                                329 |                                10240
(1 rows)
```

history contains the data table corresponding to the metrics to be dumped in the configuration file. The following statement can be used to query:

```
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)
```

Kafka connector

- **Overview**

The connector maps topic in Kafka to tables in Presto. Each record in Kafka is mapped to a message in Presto tables.



Note:

Since data in Kafka is dynamic, when Presto is used for multiple queries, something strange may sometimes occur. Currently, Presto is incapable of handling such cases.

- **Configuration**

Create a file *etc/catalog/kafka.properties*, add the following content, and enable Kafka connector.

```
connector.name=kafka
kafka.table-names=table1,table2
kafka.nodes=host1:port,host2:port
```



Note:

Presto can connect to multiple Kafka cluster through adding a new properties file in the configuration catalog, and the file name is mapped to the Presto catalog. For example, when a configuration file *orders.properties* is added, Presto creates a catalog named orders.

```
## orders.properties
```

```
connector.name=kafka # It denotes the connector type, which cannot
  be changed
kafka.table-names=tableA,tableB
kafka.nodes=host1:port,host2:port
```

Kafka connector provides the following properties:

- `kafka.table-names`

Description: Required, it defines the list of tables supported in the connector.

Details: The file name here can be modified using Schema name, with forms like

`{schema_name}.{table_name}`. The file name also can be not modified using Schema name, and the table is mapped to the Schema defined in `kafka.default-schema`.

- `kafka.default-schema`

Description: Optional, default Schema name, with the default value `default`.

- `kafka.nodes`

Description: Required, the node list in the Kafka cluster.

Details: the configuration form is like `hostname:port[,hostname:port...]`. You can configure only part of the Kafka nodes here, but Presto must be capable of connecting to all nodes in the Kafka cluster. Otherwise, a part of data may not be obtained.

- `kafka.connect-timeout`

Description: Optional, timeout for the connector and the Kafka cluster, which is 10s by default.

Details: If the pressure on the Kafka cluster is large, a long time may be taken for creating a connection, causing timeout when executing the query by Presto. At this time, adding the configured value is a better choice.

- `kafka.buffer-size`

Description: Optional, read buffer size, which is 64 kb by default.

Details: It is used to set the size of the buffer for internal data reads from Kafka. The data buffer must have a size larger than that of a message. A data buffer is distributed for each worker and data node.

- `kafka.table-description-dir`

Description: [Optional], the catalog of topic (table) description file, which is `etc/kafka` by default.

Details: Data table definition files in JSON format is stored in this directory (.json has to be used as a suffix).

- kafka.hide-internal-columns

Description: Optional, the list of preset columns to be hidden, the value of which is `true` by default.

Details: In addition to data columns defined in the table description file, the connector also maintains many extra columns for each table. The property is used to control whether these columns are shown in the execution result of the statement `DESCRIBE` and `SELECT *`.

Regardless of the setting in the configuration, these columns are involved in the query process.

The Kafka connector provides internal columns as shown in the following table:

| Column name | Type | Description |
|--------------------------------|---------|--|
| <code>_partition_id</code> | BIGINT | The ID of the Kafka partition where the record is located. |
| <code>_partition_offset</code> | BIGINT | The offset of the Kafka partition where the record is located. |
| <code>_segment_start</code> | BIGINT | The lowest offset containing this data segment. This offset is for each partition. |
| <code>_segment_end</code> | BIGINT | The largest offset containing this data segment (which is the start offset of the next segment). This offset is for each partition. |
| <code>_segment_count</code> | BIGINT | The serial number of the column in this segment. For an uncompressed topic, <code>_segment_start + _segment_count = _partition_offset</code> . |
| <code>_message_corrupt</code> | BOOLEAN | This field will be set to <code>TRUE</code> if a decoder cannot decode the record. |
| <code>_message</code> | VARCHAR | A string coded with UTF-8 from the message bytes. When the type of the topic message is text, the field will be useful. |
| <code>_message_length</code> | BIGINT | The byte length of the message. |
| <code>_key_corrupt</code> | BOOLEAN | This field will be set to <code>TRUE</code> if a key decoder cannot decode the record. |

| Column name | Type | Description |
|-------------|---------|---|
| _key | VARCHAR | A string coded with UTF-8 from the key bytes. When the type of the topic message is text, the field will be useful. |
| _key_length | BIGINT | The byte length of the key. |



Note:

For those tables without definition files, `_key_corrupt` and `_message_corrupt` remain FALSE.

- Table definition files

Kafka is a Schema-Less message system, and the formats of the messages must be defined by the producers and consumers. While Presto requires that data must be capable of being mapped into tables. Therefore, the users must provide corresponding table definition files according to the actual uses of the messages. For messages in JSON format, if a definition file is not provided, JSON functions in Presto can be used for resolution in the queries. While the method is flexible, it increases the difficulty of writing SQL statements.

When JSON is used to define a table in a table definition file, the file name can be customized, while the extension must be *.json.

```
{
  "tableName": ...,
  "schemaName": ...,
  "topicName": ...,
  "key": {
    "dataFormat": ...,
    "fields": [
      ...
    ]
  },
  "message": {
    "dataFormat": ...,
    "fields": [
      ...
    ]
  }
}
```

| Field | Optionality | Type | Description |
|------------|-------------|--------|---|
| tableName | required | string | Presto table name |
| schemaName | optional | string | the name of the Schema where the table is located |

| Field | Optionality | Type | Description |
|-----------|-------------|-------------|--|
| topicName | required | string | Kafka topic name |
| key | optional | JSON object | rules for mapping from message keys to columns |
| message | optional | JSON object | rules for mapping from messages to columns |

In which, the mapping rules for keys and messages use the following fields for description.

| Field | Optionality | Type | Description |
|------------|-------------|------------|--|
| dataFormat | required | string | A decoder for setting a group of columns |
| fields | required | JSON array | Column definition list |

fields here is a JSON array, and each element is a JSON object as follows:

```
{
  "name": ...,
  "type": ...,
  "dataFormat": ...,
  "mapping": ...,
  "formatHint": ...,
  "hidden": ...,
  "comment": ...
}
```

| Field | Optionality | Type | Description |
|------------|-------------|---------|---|
| name | required | string | column name |
| type | required | string | column data type |
| dataFormat | optional | string | column data decoder |
| mapping | optional | string | decoder parameters |
| formatHint | optional | string | hint set for the column, which can be used by the decoder |
| hidden | optional | boolean | whether hidden or not |
| comment | optional | string | column description |

- Decoder

The function of the decoder is to map Kafka messages (key + message) to the columns in the data tables. Presto uses **dummy** decoder in the absence of table definition files.

Kafka connector provides the following decoders:

- raw: Original bytes are directly used without conversion.
- csv: Messages are processed as strings in CSV format.
- json: Messages are processed as strings in JSON format.

11.8.2 What is Presto

Presto is a distributed SQL-on-Hadoop analytics engine powered by Facebook. Presto 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, offers high-performance, strong expandability and other features include:

- Fully supports ANSI SQL.
- Supports rich data sources. Presto can access rich data sources as follows:
 - interaction with Hive data warehouse
 - 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
- Strong expandability. Presto provides multiple expansion configurations.
 - Data connector expansion

- Custom data types
- Custom SQL functions

Users can expand the corresponding modules according to their own service features to achieve efficient service processes.

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

Scenarios

Presto is a distributed SQL engine located in data warehouse and data analytics services and is well suited to the following scenarios:

- ETL
- Ad-Hoc query
- 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 the advantages of open source, the EMR Presto product comes with the following advantages:

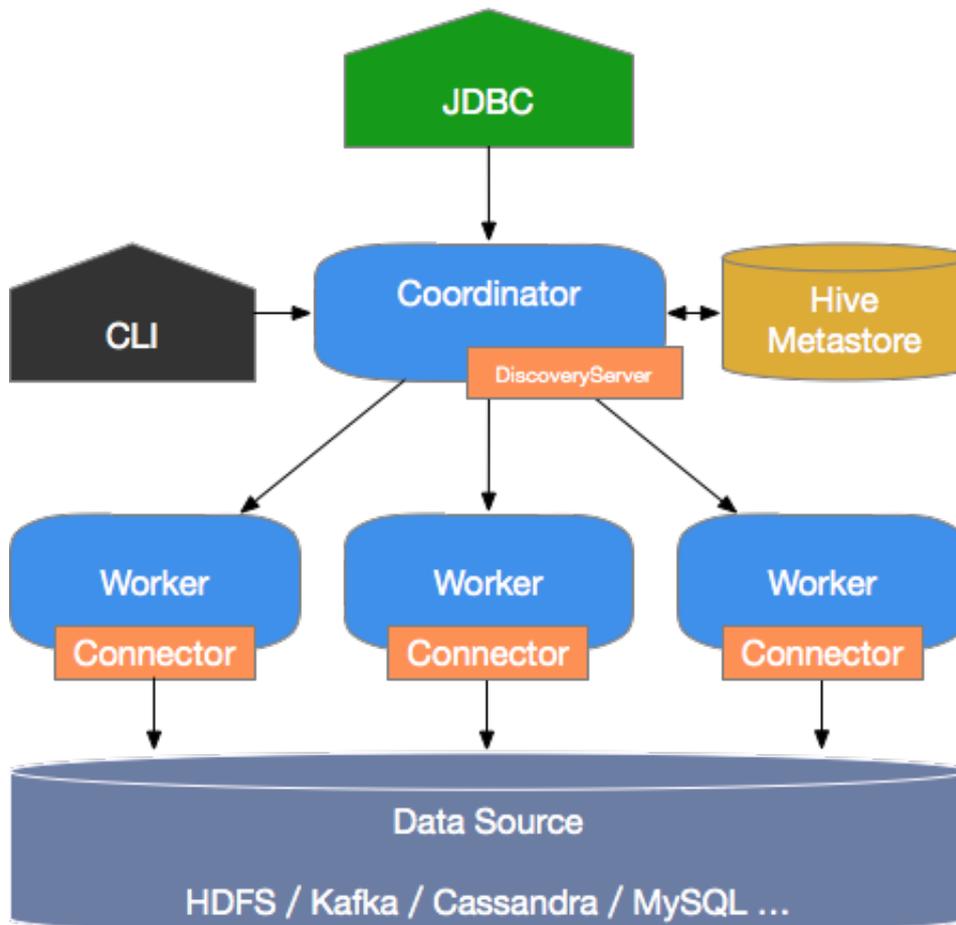
- You can purchase it for immediate use to build a Presto cluster with hundreds of nodes in minutes.
- It supports elastic resizing, you can complete up and down resizing of the cluster with simple operations.
- It works perfect in connection with the EMR software stacks, and supports processing of data stored in OSS.
- O&M free 24x7 all-in-one service.

11.8.3 Quick start with Presto

This article describes the basic usage and application development methods of the Presto database for developers to quickly start application development using Presto database.

System structure

Presto's system structure is shown in the following figure:



Presto is a typical M/S architecture system, comprising a Coordinator node and multiple Worker nodes. Coordinator is responsible for the following:

- Receiving and parsing users' 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 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.

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 a SQL statement in Presto, you are running it against one or more Catalogs.

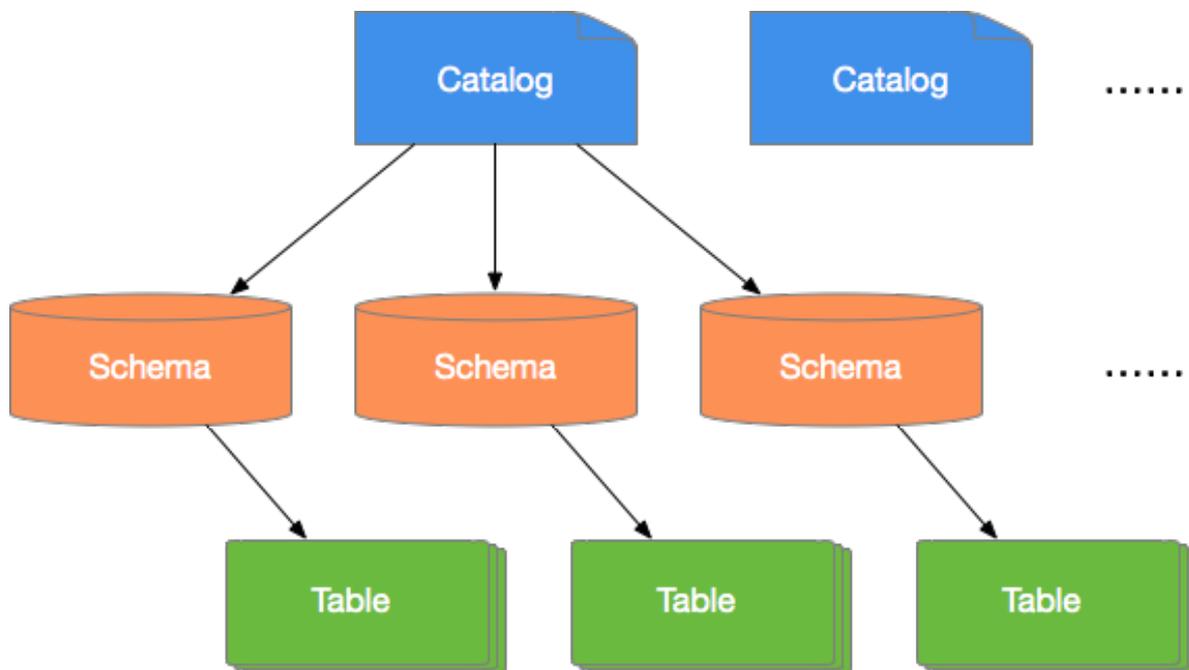
— Schema

You can take a Schema as a database instance, which contains multiple data tables.

— Table

Data table, which is the same as general database tables.

Relations among Catalog, Schema, and Table are shown in the following figure.



- Connector

Presto uses Connector to connect to various external data sources. Presto provides a standard *SPI*, which allows users to develop their own Connectors using this standard API, to access customized data sources.

Generally, a Catalog is associated with a specific Connector (which can be configured in the Properties file of the Catalog). Presto contains multiple built-in Connectors. For more information, see Connectors.

- Query-related concepts

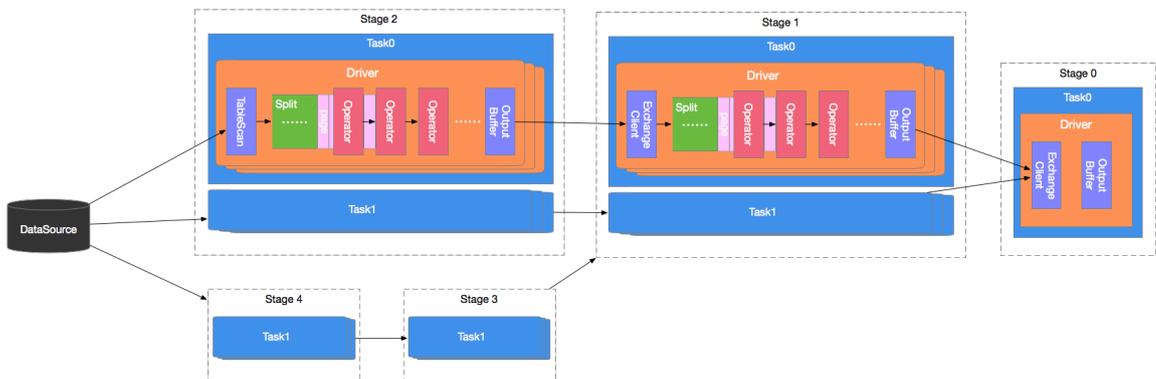
This section mainly describes related concepts in the Presto query process, for users to understand better, the execution process of Presto statements and the performance optimization methods.

- Statement

Statement refers to an SQL statement entered by a user via JDBC or CLI.

- Query

Query refers to the execution process of a query. When Presto receives an SQL Statement, the Coordinator parses this statement, generates an execution plan, and sends this plan to a Worker for execution. A Query is logically made up by several components, namely Stage, Task, Driver, Split, Operator, and DataSource, which are shown in the following figure:



- Stage

A Presto query contains multiple Stages. Stage is a logical concept, which indicates a stage of the query process, comprising one or more execution tasks. Presto uses a tree structure to organize Stages, the root node of which is Single Stage. This Stage aggregates data output from the upstream Stages, and directly sends the results to Coordinator. The leaf node of this tree is Source Stage. The Source Stage receives data from Connector for processing.

— Task

Task refers to a specific task to be executed, and it is the smallest Presto task scheduling unit. During the execution process, Presto task scheduler distributes these tasks to individual Workers for execution. Tasks in one Stage can be executed in parallel. Tasks in two different Stages transmits data via the Exchange module. Task is also a logical concept that contains the parameters and contents of the task, the actual task execution is done by the driver.

— Driver

Driver is responsible for executing the specific tasks. A Task may contain multiple Driver instances, to achieve parallel processing within the same Task. Each Driver processes a Split. A Driver is made up by a set of Operators, and is responsible for specific data operations, such as conversion and filtering.

— Operator

The Operator is the smallest execution unit, and is responsible for processing each Page of a Split, such as weighting and conversion. It is similar to logical operators in concept. Page is a column-based data structure, and is the smallest data unit that an Operator can process. A Page object constitutes of multiple Blocks, with each Block representing multiple data rows of a field. A Page can be of a maximum of 1 MB, and can contain data of up to 16 x 1024 rows.

— Exchange

Two Stages exchange data through the Exchange module. The data transmission process is actually is completed between two Tasks. Generally, a downstream Task fetches data from the Output Buffer of an upstream Task using an Exchange Client. The fetched data is then transmitted to Driver in Splits for processing.

The command line tool

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 form:

```
$ 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` are numbers as the ecm id of clusters that can be obtained through `cat /etc/hosts`.
- `81ba14ce6084` is the default password of `/etc/ecm/presto-conf/keystore`. It is recommended that you use your own keystore after the deployment.

You can execute the following command from the console:

```

Presto: Default> show schemas;
  schema.
-----
default
Hive
information_schema
tpch_100gb_orc
tpch_10gb_orc
tpch_10tb_orc
tpch_1tb_orc
(7 rows)

```

We can 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 information
--client-request-timeout <timeout> # Specifies the client timeout
value, which is 2 minutes by default
--enable-authentication # Enables client authentica
tion
--keystore-password <keystore password> # KeyStore password
--keystore-path <keystore path> # KeyStore path
--krb5-config-path <krb5 config path> # Kerberos configuration 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> # Configuration 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

```

Uses JDBC

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

- Introduction into Maven

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

```

<dependency>
  <groupId>com.facebook.presto</groupId>
  <artifactId>presto-jdbc</artifactId>
  <version>0.187</version>
</dependency>

```

- Driver class name

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

Presto JDBC driver supports various parameters that may be set as URL parameters or as **properties** passed to DriverManager. Both of the following examples are equivalent:

Example for passing 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 for passing to DriverManager as URL parameters:

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

The parameters are described as follows:

| Parameter Name | Format | Description |
|------------------------------|--------|---|
| user | STRING | User Name |
| password | STRING | Password |
| Socksproxy | \\ | SOCKS proxy server address and port.
Example: localhost:1080 |
| httpProxy | \\ | HTTP proxy server address and port.
Example: localhost:8888 |
| SSL | true\\ | Whether or not to use HTTPS for connections
. Defaults to false. |
| 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. Defaults to false. |
| 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 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 t1");
    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
        }
    }
}
}
}

```

11.8.4 Data type

Presto supports multiple common data types by default, such as Boolean, Integer, Floating-Point, String, and Date and Time. You can also add customized data types using plugins. Additionally, the customized Presto connectors are not required to support all data types.

Data types

Presto has a set of built-in data types that are as follows:

- **BOOLEAN**

Represents two option with a value of true or false.

- TINYINT

An 8-bit signed *two's complement integer*.

- SMALLINT

A 16-bit signed *two's complement integer*.

- INTEGER

A 32-bit signed *two's complement integer*.

- BIGINT

A 64-bit signed *two's complement integer*.

- REAL

A real is a 32-bit inexact, variable-precision implementing the *IEEE Standard 754* for Binary Floating-Point Arithmetic.

- DOUBLE

A 64-bit multi-precision *[IEEE-754]* binary floating-point numeric implementation.

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

- precision: total number of digits, excluding symbols
- scale: number of digits in fractional part. Scale is optional and defaults to 0.

Example: `DECIMAL '-10.7'` can be expressed with `DECIMAL(3,1)` type.

The following table describes the bits and value range of the integer type

| Value Type | Bits | 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 type

Presto supports the following built-in string types:

- VARCHAR

Variable length character data with an optional maximum length.

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

- CHAR

Fixed length character data. A CHAR type without length specified has a default length of 1.

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



Note:

A string with the specified length always has the number of characters equal to this length. Where 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

Refers to a calendar date (year, month, day) without time.

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

- TIME

Refers to a time, including hour, minute, second, and millisecond. Values of this type can be rendered in the time zone.

Example:

— `TIME '18:01:02.345'`, does not have a time zone definition, and is thus parsed using the system time zone.

— `TIME '18:01:02.345 Asia/Shanghai'`, has time zone definition, and is thus parsed using the defined time zone.

- TIMESTAMP

Refers to an instant in time that includes the date and time of 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.

Example: `TIMESTAMP '1988-01-30 01:02:03.321'` , `TIMESTAMP '1988-01-30 01:02:03.321 Asia/Shanghai'`

- INTERVAL

Mainly used in time calculated expressions to refer to a time span, the unit of which can be:

- YEAR - Year
- QUARTER - Quarter of a year
- MONTH - Month
- DAY - Day
- HOUR - 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, to support more complex business scenarios , and these data types include:

- JSON

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

Example:

- `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 consisting of a key array and a value array.

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

- ROW

A structure made up of named fields. The fields may be accessed with field reference operator `.` and the field names. Operator `+` the method of column names to access data columns.

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. An IP address that can represent either an IPv4 or IPv6 address. Internally, the type is a pure IPv6 address. Support for IPv4 is handled using the [IPv4-mapped IPv6 address range](#).

Example: `IPADDRESS '0.0.0.0'`, `IPADDRESS '2001:db8::1'`

11.8.5 Common functions and operators

This article describes common Presto functions and operators.

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 shown 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 |

Additionally, the result of NOT FALSE is TRUE, the result of NOT TRUE is FALSE, and the result of NOT NULL is NULL. For more information about the NOT operator, see [NOT operator](#).

Comparison functions and operators

- Comparison operators:

Comparison operations supported by Presto are as follows:

| Operator | Description |
|------------------------|--|
| < | Less than |
| > | Greater than |
| <= | Less than or equal to |
| >= | Greater than or equal to |
| = | Equal to |
| <> / != | Not equal to |
| [NOT] BETWEEN | Value X is [not] between the min and the max values |
| IS [NOT] NULL | Tests whether a value is NULL |
| IS [NOT] DISTINCT FROM | Determines if two values are identical. Generally, NULL signifies an unknown value, so any comparison involving a NULL will produce NULL. However, the IS [NOT] DISTINCT FROM operator treats NULL as a known value, and returns a TRUE or FALSE result. |

- Comparison functions

Presto provides the following comparison related functions:

- GREATEST

Returns the largest of the provided values.

Example: `GREATEST(1, 2)`

- LEAST

Returns the smallest of the provided values.

Example: `LEAST(1, 2)`

- Quantified comparison predicates

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

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

For example:

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

ALL, ANY and SOME are quantified comparison predicates.

- `A = ALL (...)`: Evaluates to true when A is equal to all values. Example: `SELECT 21 = ALL (VALUES 20, 20, 20);`, return TRUE.
- `A <> ALL (...)`: Evaluates to true when A doesn't match any value. Example: `SELECT 21 <> ALL (VALUES 19, 20, 22);`, return TRUE.
- `A < ALL (...)`: Evaluates to true when A is smaller than the smallest value. Example: `SELECT 18 < ALL (VALUES 19, 20, 22);`, return TRUE.
- `A = ANY (...)`: Evaluates to true when A is equal to any of the values. This form is equivalent to `A IN (...)`. Example: `SELECT 'hello' = ANY (VALUES 'hello', 'world');`, return TRUE.
- `A <> ANY (...)`: Evaluates to true when A doesn't match one or more values. This form is equivalent to `A IN (...)`. Example: `SELECT 21 <> ALL (VALUES 19, 20, 21);`, return TRUE.
- `A < ANY (...)`: Evaluates to true when A is smaller than the biggest value. Example: `SELECT 21 < ALL (VALUES 19, 20, 22);`, return TRUE.

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

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 *expression* 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.

Example:

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

```
--- Compare conditional expression
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 used to simplify the writing method for comparison logic of two values. Its expression forms are as follows:

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

Evaluates and returns *true_value* if *condition* is true, otherwise *false_value* is returned. *false_value* is optional. If it is not specified, NULL will be returned if condition is not true.

- COALESCE

The COALESCE function returns the first non-null value in the argument list. Its expression forms are as follows:

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

- NULLIF

The NULLIF function returns null if value1 equals value2, otherwise returns value1. Usage of the function is as follows:

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

- TRY

The TRY function evaluates an expression and handle certain types of errors by returning NULL. The following errors are handled by TRY:

- Division by zero, e.g. $x/0$
- Invalid cast or function argument
- Numeric value out of range

Generally used in conjunction with COALESCE to return the default value in case of errors.

The usage is as follows:

```
TRY(expression)
```

Example:

```
--- When COALESCE and TRY are used in conjunction, if packages=0,
and a "division by zero" error is thrown, the default value (0) will
be returned.
SELECT COALESCE(TRY(total_cost / packages), 0) AS per_package FROM
shipping;
per_package
-----
          4
         14
          0
         19
(4 rows)
```

Conversion functions

Presto provides the following explicit conversion functions:

- CAST

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

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

- TRY_CAST

Like 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
```

Example:

```
SELECT TYPEOF(123); -- integer
SELECT TYPEOF('cat'); -- varchar(3)
```

```
SELECT TYPEOF(cos(2) + 1.5); -- double
```

Mathematical functions and operators

- Mathematical operators

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

- Mathematical functions

Presto provides a wealth of mathematical functions, as shown in the following table:

| Function | Syntax | Description |
|--------------------|--|--|
| abs | abs(x) → | Returns the absolute value of x. |
| cbrt | cbrt(x) → double | Returns the cube root of x. |
| ceil | ceil(x) | Returns x rounded up to the nearest integer. This is an alias for ceiling . |
| ceiling | ceiling(x) | Returns x rounded up to the nearest integer. |
| cosine_similarity | cosine_similarity(x, y) → double | Returns the cosine similarity between the sparse vectors x and y. |
| degrees | degrees(x) → double | Converts angle x in radians to degrees. |
| e | e() → double | Returns the constant Euler's number. |
| exp | exp(x) → double | Returns Euler's number raised to the power of x. |
| floor | floor(x) | Returns x rounded down to the nearest integer. |
| from_base | from_base(string, radix) → bigint | Returns the value of string interpreted as a base-radix number. |
| inverse_normal_cdf | inverse_normal_cdf(mean, sd, p) → double | Computes the inverse of the Normal cdf with given mean and standard deviation (sd) for the cumulative probability. |
| ln | ln(x) → double | Returns the natural logarithm of x. |

| Function | Syntax | Description |
|--------------|---|--|
| log2 | log2(x)->double | Returns the base 2 logarithm of x. |
| log10 | log10(x)->double | Returns the base 10 logarithm of x. |
| log | log(x,b) -> double | Returns the base b logarithm of x. |
| mod | mod(n,m) | Returns the modulus (remainder) of n divided by m. |
| pi | pi()->double | Returns the constant Pi. |
| pow | pow(x,p)->double | Returns x raised to the power of p. This is an alias for power . |
| 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 value in the range $0.0 \leq x < 1.0$. This is an alias for random . |
| random | random()->double | Returns a pseudo-random value in the range $0.0 \leq x < 1.0$. |
| random | random(n) | Returns a pseudo-random number between 0 and n (exclusive). |
| round | round(x) | Returns x rounded to the nearest integer. |
| round | round(x, d) | Returns x rounded to d decimal places. |
| sign | sign(x) | Returns the signum function of x, that is: 0 if the argument is 0; if the argument is greater than 0; -1 if the argument is less than 0. For double arguments, the function additionally returns: NaN if the argument is NaN; 1 if the argument is +Infinity; -1 if the argument is -Infinity. |
| sqrt | sqrt(x)->double | Returns the square root of x. |
| to_base | to_base(x, radix)->varchar | Returns the base-radix representation of x. |
| truncate | truncate(x) → double | Returns x rounded to integer by dropping digits after 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 according to the bins specified by the array bins. |

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

Bitwise functions

Presto provides following bitwise functions:

| Function | Syntax | Description |
|-------------|-----------------------------|---|
| bit_count | bit_count(x, bits) → bigint | Returns the number of bits set in x at position 1 in 2's complement representation. |
| bitwise_and | bitwise_and(x, y) → bigint | The bitwise AND function |
| bitwise_not | bitwise_not(x) → bigint | The bitwise NOT function |
| bitwise_or | bitwise_or(x, y) → bigint | The bitwise OR function |
| bitwise_xor | bitwise_xor(x, y) → bigint | The bitwise XOR function |

| Function | Syntax | Description |
|-----------------|-----------------------------|--|
| bitwise_and_agg | bitwise_and_agg(x) → bigint | Returns the bitwise AND of all input values in 2's complement representation, and x is an array. |
| bitwise_or_agg | bitwise_or_agg(x) → bigint | Returns the bitwise OR of all input values in 2's complement representation, and x 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
```

Decimal functions and operators

- Decimal literals

Use the following syntax to define literal of DECIMAL type:

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

The *precision* of DECIMAL type for literal will be equal to number of digits in literal (including trailing and leading zeros). The *scale* will be equal to number of digits in fractional part (including trailing zeros). For example:

| Example literal | Data type |
|--------------------------------|-----------------|
| DECIMAL '0' | DECIMAL(1) |
| DECIMAL '12345' | DECIMAL(5) |
| DECIMAL '000012345.1234500000' | DECIMAL(20, 10) |

- Operators

Arithmetic operators

Assuming x is of type DECIMAL(xp, xs) and y is of type DECIMAL(y_p, y_s),

- x: DECIMAL(x_p,x_s)
- y: DECIMAL(y_p,y_s)

and 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 operator work for DECIMAL type.

- Unary decimal operators

The - operator performs negation for DECIMAL type.

String functions and operators

- Concatenation operator

The || operator performs concatenation.

- String functions

String functions supported by Presto are listed in the following table:

| Function Name | Syntax | Description |
|------------------|---|--|
| chr | chr(n) → varchar | Returns the Unicode code point n as a single character string. |
| codepoint | codepoint(string) → integer | Returns the Unicode code point of the only character of string . |
| concat | concat(string1, ..., stringN)
→ varchar | Returns the concatenation of string1, string2, ..., stringN. This function provides the same functionality as the SQL-standard concatenation operator. |
| hamming_distance | hamming_distance(string1, string2) → bigint | Returns the <i>Hamming distance</i> of string1 and string2, i.e. the number of positions at which |

| Function Name | Syntax | Description |
|----------------------|---|---|
| | | the corresponding characters are different. Note that the two strings must have the same length. |
| length | length(string) → bigint | Returns the length of string in characters. |
| levenshtein_distance | levenshtein_distance(string1, string2) → bigint | Returns the <i>Levenshtein edit distance</i> of string1 and string2. |
| lower | lower(string) → varchar | Converts string to lowercase. |
| upper | upper(string) → varchar | Converts string to uppercase. |
| replace | replace(string, search) → varchar | Removes all instances of search from string . |
| replace | replace(string, search, replace) → varchar | Replaces all instances of search with replace in string . |
| reverse | reverse(string) → varchar | Returns string with the characters in reverse order. |
| lpad | lpad(string, size, padstring) → varchar | Left pads string to size characters with padstring . If size is less than the length of string , the result is truncated to size characters. size must not be negative and padstring must be non-empty. |
| rpadd | rpadd(string, size, padstring) → varchar | Right pads string to size characters with padstring . If size is less than the length of string , the result is truncated to size characters. size must not be negative and padstring must be non-empty. |
| 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 size at the maximum of limit. |
| split_part | split_part(string, delimiter, index) → varchar | Splits string on delimiter and returns the field index . Field indexes start with 1. |
| split_to_map | split_to_map(string, entryDelimiter, keyValueDelimiter) → map | Splits string by entryDelimiter and keyValueDelimiter and returns a map. |

| Function Name | Syntax | Description |
|---------------|---|--|
| strpos | strpos(string, substring) → bigint | Returns the starting position of the first instance of substring in string. Positions start with 1. If not found, 0 is returned. |
| position | position(substring IN string) → bigint | Returns the starting position of the first instance of substring in string. |
| substr | substr(string, start, [length]) → varchar | Returns a substring from string of [length] length from the starting position start . Positions start with 1. The length parameter is optional. |

- Unicode functions

- normalize(string) → varchar

- Transforms string with *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 a UTF-8 encoded string from binary. Invalid UTF-8 sequences are replaced with **replace**, which is Unicode replacement character **U+FFFD** by default. Note that the replacement string **replace** must either be a single character or empty.

Regular expression functions

Presto supports all of the regular expression functions use the *Java Pattern* syntax, with a few notable exceptions:

- When using multi-line mode

- enabled via the `?m` flag.
- `\n` is recognized as a line terminator
- the `?d` flag is not supported
- Case-sensitive matching
 - enabled via the `?i` flag
 - 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, `\uD800\uDC00` is not treated as `U+10000` and must be specified as `\x{10000}`.

- Boundaries `\b` are incorrectly handled for a non-spacing mark without a base character.
- `\Q` and `\E` are not supported in character classes (such as `[A-Z123]`) and are instead treated as literals.
- 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=` prefixes. 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_category=` or `gc=` prefixes. Example: `\p{L}`.
 - Binary properties must be specified directly, without the `Is`. Example: use `\p{NoncharacterCodePoint}` instead of `\p{IsNoncharacterCodePoint}`.

Regular expression functions provided by Presto are as follows:

- `regexp_extract_all(string, pattern, [group])` → array

Returns the substring(s) 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*.

Examples

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

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

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

Examples

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

- `regexp_extract_all(string, pattern, [group])` → array

Returns the substring(s) 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_extract('1a 2b 14m', '\d+'); -- 1
SELECT regexp_extract('1a 2b 14m', '(\d+)([a-z]+)', 2); -- 'a'
```

- `regexp_like(string, pattern)` → boolean

Evaluates the regular expression `pattern` and determines if it is contained within `string`. It returns TRUE if yes, and False if otherwise. This function is similar to the LIKE operator, except that the pattern only needs to be contained within string, rather than needing to match all of string.

Examples

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

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

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

Capturing groups can be referenced in `replacement` using `$g` (g is the ordinal number, starting at one) for a numbered group or `${name}` for a named group. A dollar sign \$ may 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`

Splits string 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', ''] 4 elements
-- The last one is an empty string
```

Binary functions and operators

- Concatenation operator

The `||` operator performs binary concatenation.

- Binary functions

| Function | Syntax | Description |
|-------------------------------|--|--|
| <code>length</code> | <code>length(binary) → bigint</code> | Returns the length of binary in bytes. |
| <code>concat</code> | <code>concat(binary1, ..., binaryN) → varbinary</code> | Returns the concatenation of <code>binary1</code> , <code>binary2</code> , ..., <code>binaryN</code> . |
| <code>to_base64</code> | <code>to_base64(binary) → varchar</code> | Encodes binary into a base64 string representation. |
| <code>from_base64</code> | <code>from_base64(string) → varbinary</code> | Decodes binary data from the base64 encoded string. |
| <code>to_base64url</code> | <code>to_base64url(binary) → varchar</code> | Encodes binary into a base64 string representation using the URL safe alphabet. |
| <code>from_base64url</code> | <code>from_base64url(string) → varbinary</code> | Decodes binary data from the base64 encoded string using the URL safe alphabet. |
| <code>to_hex</code> | <code>to_hex(binary) → varchar</code> | Encodes binary into a hex string representation. |
| <code>from_hex</code> | <code>from_hex(string) → varbinary</code> | Decodes binary data from the hex encoded string. |
| <code>to_big_endian_64</code> | <code>to_big_endian_64(bigint) → varbinary</code> | Encodes <code>bigint</code> in a 64-bit 2's complement big endian format. |

| Function | Syntax | Description |
|--------------------|--|--|
| from_big_endian_64 | from_big_endian_64(binary)
→ bigint | Decodes bigint value from a 64-bit 2's complement big endian binary. |
| to_ieee754_32 | to_ieee754_32(real) →
varbinary | Encodes real in a 32-bit big-endian binary according to IEEE 754 single-precision floating-point format. |
| to_ieee754_64 | to_ieee754_64(double) →
varbinary | Encodes double in a 64-bit big-endian binary according to IEEE 754 double-precision floating-point format. |
| crc32 | crc32(binary) → bigint | Computes the CRC-32 of binary. |
| md5 | md5(binary) → varbinary | Computes the md5 hash of binary. |
| sha1 | sha1(binary) → varbinary | Computes the sha1 hash of binary. |
| sha256 | sha256(binary) → varbinary | Computes the sha256 hash of binary. |
| sha512 | sha512(binary) → varbinary | Computes the sha512 hash of binary. |
| xxhash64 | xxhash64(binary) → varbinary | Computes the xxhash64 hash of binary. |

Date and time functions and operators

- 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 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_Angeles';
--- 2012-10-30 18:00:00.000 America/Los_Angeles
```

- Date and time functions

— Basic functions

| Function | Syntax | Description |
|-------------------------------------|---|--|
| <code>current_date</code> | <code>current_date -> date</code> | Returns the current date as of the start of the query. |
| <code>current_time</code> | <code>current_time -> time with time zone</code> | Returns the current time as of the start of the query. |
| <code>current_timestamp</code> | <code>current_timestamp -> timestamp with time zone</code> | Returns the current timestamp as of the start of the query. |
| <code>current_timezone</code> | <code>current_timezone() -> varchar</code> | Returns the current time zone. |
| <code>date</code> | <code>date(x) -> date</code> | Parses a date literal into a date |
| <code>from_iso8601_timestamp</code> | <code>from_iso8601_timestamp(string) -> timestamp with time zone</code> | Parses the ISO 8601 formatted string into a timestamp with time zone. |
| <code>from_iso8601_date</code> | <code>from_iso8601_date(string) -> date</code> | Parses the ISO 8601 formatted string into a date. |
| <code>from_unixtime</code> | <code>from_unixtime(unixtime, [timezone_str]) -> timestamp</code> | Returns the UNIX timestamp as a timestamp. Timestamp option is allowed. |
| <code>from_unixtime</code> | <code>from_unixtime(unixtime, hours, minutes) -> timestamp with time zone</code> | Returns the UNIX timestamp as a timestamp with time zone using hours and minutes for 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. |

| Function | Syntax | Description |
|---------------------|------------------------------------|---|
| now | now() → timestamp with time zone | Returns the current time. This is an alias for current_time . |
| to_iso8601 | to_iso8601(x) → varchar | Formats x as an ISO 8601 string. x can be DATE, or TIMESTAMP [with time zone]. |
| to_millise
conds | to_milliseconds(interval) → bigint | Returns the day-to-second interval as milliseconds. |
| to_unixtime | to_unixtime(timestamp) → double | Returns timestamp as a UNIX timestamp. |

**Note:**

The following SQL-standard functions do not use parenthesis:

- current_data
- current_time
- current_timestamp
- localtime
- localtimestamp

— Truncation function

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

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

where **unit** is one of:

- second: Seconds
- minute: Minutes
- hour: Hours
- day: Days
- week: Weeks
- month: Months
- quarter: Months
- 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 type `unit` to `timestamp`. Subtraction can be performed by using a negative value with a unit.

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

Returns interval between two timestamps expressed in terms of unit.

Where `unit` is one of:

- `ns`: Nanoseconds
- `us`: Microseconds
- `ms`: Milliseconds
- `s`: Seconds
- `m`: Minutes
- `h`: Hours
- `d`: Days

— Date and time extraction functions

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

`extract(field FROM x) → bigint`

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

- `YEAR`: Year
- `QUARTER`: Quarter of a year
- `MONTH`: Month
- `WEEK`: Week
- `DAY`: Day
- `DAY_OF_MONTH`: Day of a month
- `DAY_OF_WEEK`: Day of a week
- `DOW`: This is an alias for `DAY_OF_WEEK`
- `DAY_OF_YEAR`: Day of a year
- `DOY`: This is an alias for `DAY_OF_YEAR`

- YEAR_OF_WEEK: Year of an *ISO Week*
- YOW: This is an alias for YEAR_OF_WEEK
- HOUR: Hour
- MINUTE: Minute
- SECOND: Second
- TIMEZONE_HOUR: Hour with timezone
- TIMEZONE_MINUTE: Minute with timezone

For the sake of convenience, Presto provides the following helper functions:

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

| Function | Syntax | Description |
|--------------|--------------------------|--|
| year | year(x) → bigint | Returns the year from x. |
| year_of_week | year_of_week(x) → bigint | Returns the year of a week from x (<i>ISO Week</i>). |
| yow | yow(x) → bigint | This is an alias for year_of_week. |

— MySQL date functions

Presto uses a format string that is compatible with MySQL `date_parse` and `str_to_date` functions, which are:

- `date_format(timestamp, format) → varchar`
Formats `timestamp` as a string using `format`.
- `date_parse(string, format) → timestamp`
Parses string into a timestamp using `format`.

MySQL format specifiers supported by Presto are shown in the following table:

| Specifier | Description |
|-----------|---|
| %a | Abbreviated weekday name (Sun .. Sat). |
| %b | Abbreviated month name (Jan .. Dec). |
| %c | Month, numeric (1 .. 12), cannot be zero |
| %d | Day of the month, numeric (01 .. 31), cannot be zero |
| %e | Day of the month, numeric (1 .. 31), cannot be zero |
| %f | Fraction of second (6 digits for printing: 000000 .. 999000; 1 - 9 digits for parsing: 0 .. 999999999). |
| %H | Hour (00 .. 23). |
| %h | Hour (01 .. 12). |
| %l | Hour (01 .. 12). |
| %i | Minutes, numeric (00 .. 59). |
| %j | Day of year (001 .. 366). |
| %k | Hour (0 .. 23). |
| %l | Hour (1 .. 12). |
| %M | Month name (January .. December). |
| %m | Month, numeric (01 .. 12) [4]. |

| Specifier | Description |
|-----------|--|
| %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). When parsing, two-digit year format assumes range [1970 .. 2069] |
| %% | A literal '%' character |



Note:

The following specifiers are not currently supported: %D %U %u %V %w %X

— Java date functions

The functions in this section use a format string that is compatible with [JodaTime's DateTimeFormat pattern](#) format.

- `format_datetime(timestamp, format) → varchar`: Formats timestamp
- `parse_datetime(string, format) → timestamp with time zone`: Parses string into a timestamp

Aggregate functions

Aggregate functions have the following features:

- Input a data set
- Output a single computation result.

Almost all of these aggregate functions ignore `null` values and return `null` for no input rows or when all values are `null`, with a few notable exceptions:

- `count`

- count_if
- max_by
- min_by
- approx_distinct
- Basic aggregate functions

| Function | Syntax | Description |
|----------------|--|--|
| arbitrary | arbitrary(x) → [same as input] | Returns an arbitrary non-null value of x. |
| array_agg | array_agg(x) → array<[same as input]> | Returns an array created from the input x elements. |
| avg | avg(x) → double | Returns the average (arithmetic mean) of all input values. |
| avg | avg(time interval type) → time interval type | Returns the average interval length of all input values. |
| bool_and | bool_and(boolean) → boolean | Returns TRUE if every input value is TRUE, otherwise FALSE. |
| bool_or | bool_or(boolean) → boolean | Returns TRUE if any input value is TRUE, otherwise FALSE. |
| checksum | checksum(x) → varbinary | Returns an order-insensitive checksum of the given values. |
| count | count(*) → bigint | Returns the number of input rows. |
| count | count(x) → bigint | Returns the number of non-null input values. |
| count_if | count_if(x) → bigint | Returns the number of TRUE input values. This function is equivalent to <code>count(CASE WHEN x THEN 1 END)</code> . |
| every | every(boolean) → boolean | This is an alias for bool_and . |
| geometric_mean | geometric_mean(x) → double | Returns the geometric mean of all input values. |
| max_by | max_by(x, y) → [same as x] | Returns the value of x associated with the maximum value of y over all input values. |
| max_by | max_by(x, y, n) → array<[same as x]> | Returns n values of x associated with the n largest of all input values of y in descending order of y. |
| min_by | min_by(x, y) → [same as x] | Returns the value of x associated with the minimum value of y over all input values. |

| Function | Syntax | Description |
|----------|--------------------------------------|--|
| min_by | min_by(x, y, n) → array<[same as x]> | Returns n values of x associated with the n smallest of all input values of y in ascending order of y. |
| max | max(x) → [same as input] | Returns the maximum value of all input values. |
| max | max(x, n) → array<[same as x]> | Returns n largest values of all input values of x. |
| min | min(x) → [same as input] | Returns the minimum value of all input values. |
| min | min(x, n) → array<[same as x]> | Returns n smallest values of all input values of x. |
| sum | sum(x) → [same as input] | Returns the sum of all input values. |

- Bitwise aggregate functions

For bitwise aggregate functions, refer to `bitwise_and_agg` and `bitwise_or_agg` functions as described in [General aggregate functions](#).

- Map aggregate functions

| Function | Syntax | Description |
|--------------|--------------------------------|---|
| histogram | histogram(x) → map | Returns a map containing the count of the number of times each input value occurs. |
| map_agg | map_agg(key, value) → map | Returns a MAP created from the input key/value pairs. |
| map_union | map_union(x) → map | Returns the union of all the input maps. If a key is found in multiple input maps, that key's value in the resulting map comes from an arbitrary input map. |
| multimap_agg | multimap_agg(key, value) → map | Returns a multimap created from the input key/value pairs. |

- Close aggregate function

| Function | Syntax | Description |
|-----------------|----------------------------------|---|
| approx_distinct | approx_distinct(x, [e]) → bigint | Returns the approximate number of distinct input values. This function provides an approximation of <code>count(DISTINCT x)</code> . Zero is returned if all input values are null. |

| Function | Syntax | Description |
|-------------------|---|---|
| | | This function should produce a standard error of no more than ϵ , which is the standard deviation of the (approximately normal) error distribution over all possible sets. It is optional, and is 2.3% by default. The current implementation of this function requires that ϵ be in the range of [0.01150, 0.26000]. It does not guarantee an upper bound on the error for any specific input set. |
| approx_percentile | approx_percentile(x, percentage) → [same as x] | Returns the approximate percentile for all input values of x at the given percentage. |
| approx_percentile | approx_percentile(x, percentages) → array<[same as x]> | Similar to the preceding function, percentages is an array, and returns constant values for all input rows. |
| approx_percentile | approx_percentile(x, w, percentage) → [same as x] | Similar to the preceding function, w is the weighted value of x. |
| approx_percentile | approx_percentile(x, w, percentage, accuracy) → [same as x] | Similar to the preceding function, accuracy is the upper bound of the estimation accuracy, and the value must be in the range of [0, 1]. |
| approx_percentile | approx_percentile(x, w, percentages) → array<[same as x]> | Similar to the preceding function, percentages is an array, and returns constant values for all input rows. |
| numeric_histogram | numeric_histogram(buckets, value, [weight]) → map | Computes an approximate histogram with up to a given number of buckets. buckets must be a BIGINT. value and weight must be numeric. weight is optional, and is 1 by default. |

- Statistical aggregate functions

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

| Function | Syntax | Description |
|----------------|-------------------------------|---|
| kurtosis | kurtosis(x) → double | Returns the excess kurtosis of all input values . Unbiased estimate using the following expression: $kurtosis(x) = \frac{n(n+1)}{((n-1)(n-2)(n-3))} \frac{\sum[(x_i - \text{mean})^4]}{\text{sttdev}(x)^4} - \frac{3(n-1)^2}{((n-2)(n-3))}$ |
| regr_intercept | regr_intercept(y, x) → double | Returns linear regression intercept of input values. <i>y</i> is the dependent value. <i>x</i> is the independent value. |
| regr_slope | regr_slope(y, x) → double | Returns linear regression slope of input values. <i>y</i> is the dependent value. <i>x</i> is the independent value. |
| 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. |
| variance | variance(x) → double | This is an alias for <code>var_samp</code> . |

11.8.6 SQL statement

SQL statement

ALTER SCHEMA

- Synopsis

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

- Description

Renames SCHEMA.

- Examples

```
ALTER SCHEMA web RENAME TO traffic -- Renames Schema 'web' as '
traffic'
```

ALTER TABLE

- Synopsis

```
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; --- Rename
ALTER TABLE users ADD COLUMN zip varchar; --- Add column
ALTER TABLE users DROP COLUMN zip; --- Drop column
ALTER TABLE users RENAME COLUMN id TO user_id; --- Rename column
```

CALL

- Synopsis

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

- Description

Calls a stored procedure. Stored procedures can be provided by connectors to perform data manipulation or administrative tasks. Some connectors such as the PostgreSQL Connector, are for 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 via **CALL**.

- Examples

```
CALL test(123, 'apple'); --- Call a stored procedure using
positional arguments
CALL test(name => 'apple', id => 123); --- Call a stored procedure
using named arguments
```

```
CALL catalog.schema.test(); --- Call a stored procedure using a
fully qualified name
```

COMMIT

- Synopsis

```
COMMIT [WORK]
```

- Description

Commits the current transaction.

- Examples

```
COMMIT;
COMMIT WORK;
```

CREATE SCHEMA

- Synopsis

```
CREATE SCHEMA [ IF NOT EXISTS ] schema_name
[ WITH ( property_name = expression [, ...] ) ]
```

- Description

Creates a new SCHEMA. Schema is a container that holds tables, views, and other database objects.

- The optional `IF NOT EXISTS` clause causes the error to be suppressed if the schema already exists;
- The optional `WITH` clause can be used to set properties on the newly created schema. To list all available schema properties, run the following query:

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

- Examples

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

CREATE TABLE

- Synopsis

```
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 the `CREATE TABLE AS` to create a table from an existing data set.

- The optional `IF NOT EXISTS` clause causes the error to be suppressed if the table already exists.
- The optional `WITH` clause can be used to set properties on the newly created table. To list all available table properties, run the following query:

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

- The `LIKE` clause can be used to include all the column definitions from an existing table in the new table. Multiple `LIKE` clauses may be specified.
- If `INCLUDING PROPERTIES` is specified, all of the table properties are copied to a new table. If the `WITH` clause specifies the same property name as one of the copied properties using `INCLUDING PROPERTIES`, the value from the `WITH` clause is used. The default behavior is `EXCLUDING PROPERTIES`.

- Examples

```
--- Create a new table orders:
CREATE TABLE orders (
  orderkey bigint,
  orderstatus varchar,
  totalprice double,
  orderdate date
)
WITH (format = 'ORC')
--- Create the table orders if it does not already exist, adding a
table comment and a column 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 the table bigger_orders, using some column definitions from
orders:
CREATE TABLE bigger_orders (
  another_orderkey bigint,
  LIKE orders,
  another_orderdate date
```

```
)
```

CREATE TABLE AS

- Synopsis

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

- Description

Creates a new table containing the result of a `SELECT` query.

- The optional `IF NOT EXISTS` clause causes the error to be suppressed if the table already exists.
- The optional `WITH` clause can be used to set properties on the newly created table. To list all available table properties, run the following query:

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

- Examples

```
--- Select two columns from orders to create a new table
CREATE TABLE orders_column_aliased (order_date, total_price)
AS
SELECT orderdate, totalprice
FROM orders
--- Create a new table using the 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 new table, 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 new table with the same schema as nation and no data
Create Table maid
SELECT *
FROM nation
```

```
WITH NO DATA
```

CREATE VIEW

- Synopsis

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

- Description

Creates a view. The view is a logic table that does not contain any data. It can be referenced by future queries. The query stored by the view is run every time the view is referenced by another query.

The optional `OR REPLACE` clause causes the view to be replaced if it already exists rather than raising an error.

- Examples

```
--- Create a simple view
CREATE VIEW test AS
SELECT orderkey, orderstatus, totalprice / 2 AS half
FROM orders
--- Create view using the aggregate function
CREATE VIEW orders_by_date AS
SELECT orderdate, sum(totalprice) AS price
FROM orders
GROUP BY orderdate
--- Create a view that replaces an existing view
CREATE OR REPLACE VIEW test AS
SELECT orderkey, orderstatus, totalprice / 4 AS quarter
FROM orders
```

DEALLOCATE PREPARE

- Synopsis

```
DEALLOCATE PREPARE statement_name
```

- Synopsis

Removes a statement with the name `statement_name` from the list of prepared statements in a session.

- Examples

```
--- Deallocate a statement named my_query
```

```
DEALLOCATE PREPARE my_query;
```

DELETE

- Synopsis

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

- Description

If the `WHERE` clause is specified, delete the matching rows from the table. If the `WHERE` is not specified, all rows from the table are deleted.

- Examples

```
--- Delete the matching row
DELETE FROM lineitem WHERE shipmode = 'AIR';
--- Delete the matching row
DELETE FROM lineitem
WHERE orderkey IN (SELECT orderkey FROM orders WHERE priority = 'LOW
');
--- Clear the table
DELETE FROM orders;
```

- Limitations

Some connectors have limits or no support for `DELETE`.

DESCRIBE

- Synopsis

```
DESCRIBE table_name
```

- Description

Retrieves the table definitions, and is an alias for [SHOW COLUMNS](#).

- Examples

```
DESCRIBE orders;
```

DESCRIBE INPUT

- Synopsis

```
DESCRIBE INPUT statement_name
```

- Description

Lists the input parameters of a prepared statement along with the position and type of each parameter.

- Examples

```

--- Create a pre-compiled query 'my_select1'
PREPARE my_select1 FROM
SELECT ? From nation where regionkey =? AND name < ? ;
--- Get the descriptive information of this prepared statement
DESCRIBE INPUT my_select1;

```

```
DESCRIBE INPUT my_select1;
```

```

Position | Type
-----
          0 | unknown
          1 | bigint
          2 | varchar
(3 rows)

```

DESCRIBE OUTPUT

- Synopsis

```
DESCRIBE OUTPUT statement_name
```

- Description

Lists the output columns of a prepared statement, including the column name (or alias), catalog, schema, table name, type, type size in bytes, and a boolean indicating if the column is aliased.

- Examples

- Example one

Prepare a prepared statement:

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

Execute `DESCRIBE OUTPUT`, which outputs:

```

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 two

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

Execute `DESCRIBE OUTPUT`, which outputs:

```
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 three:

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

Execute `DESCRIBE OUTPUT`, which outputs:

```
DESCRIBE OUTPUT my_create;
 Column Name | Catalog | Schema | Table | Type | Type Size |
 Aliased
-----+-----+-----+-----+-----+-----
+-----
 rows        |         |         |         | bigint |          8 |
 false
(1 row)
```

DROP SCHEMA

- Synopsis

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

- Description

Drops an existing Schema.

- The schema must be empty.
- The optional `IF EXISTS` clause causes the error to be suppressed if the schema does not exist.
- Examples

```
DROP SCHEMA web;
```

```
DROP TABLE IF EXISTS sales;
```

DROP TABLE

- Synopsis

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

- Description

Drops an existing table. The optional **IF EXISTS** clause causes the error to be suppressed if the table does not exist.

- Examples

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

DROP VIEW

- Synopsis

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

- Description

Drops an existing view. The optional **IF EXISTS** clause causes the error to be suppressed if the view does not exist.

- Examples

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

EXECUTE

- Synopsis

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

- Description

Executes a prepared statement. Parameter values are defined in the **USING** clause.

- Examples

— Example one

```
PREPARE my_select1 FROM  
SELECT name FROM nation;  
--- Execute a prepared statement
```

```
EXECUTE my_select1;
```

— Example two

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

EXPLAIN

- Synopsis

```
EXPLAIN [ ( option [, ...] ) ] statement
where option can be one of:
    FORMAT { TEXT | GRAPHVIZ }
    TYPE { LOGICAL | DISTRIBUTED | VALIDATE }
```

- Description

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

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

Use `TYPE DISTRIBUTED` option to display fragmented plan. Each plan fragment is executed by a single or multiple Presto nodes. Fragments separation represent the data exchange between Presto nodes. Fragment type specifies how the fragment is executed by Presto nodes and how the data is distributed between fragments. Fragment types are as follows:

- `SINGLE`: Fragment is executed on a single node.
 - `HASH`: Fragment is executed on a fixed number of nodes with the input data distributed using a hash function.
 - `ROUND_ROBIN`: Fragment is executed on a fixed number of nodes with the input data distributed in a `ROUND-ROBIN` fashion.
 - `BROADCAST`: Fragment is executed on a fixed number of nodes with the input data broadcasted to all nodes.
 - `SOURCE`: Fragment is executed on nodes where input splits are accessed.
- Examples

— Example one

Logical plan:

```

presto:tiny> EXPLAIN SELECT regionkey, count(*) FROM nation GROUP
BY 1;

```

Query Plan

```

-----
- Output[regionkey, _coll] => [regionkey:bigint, count:bigint]
  _ Coll: = count?
  - RemoteExchange[GATHER] => regionkey:bigint, count:bigint
    - Aggregate(FINAL)[regionkey] => [regionkey:bigint, count
:bigint]
      count := "count"("count_8")
      - LocalExchange[HASH][$hashvalue] ("regionkey") =>
regionkey:bigint, count_8:bigint, $hashvalue:bigint
        - RemoteExchange[REPARTITION][$hashvalue_9] =>
regionkey:bigint, count_8:bigint, $hashvalue_9:bigint
          - Project[] => [regionkey:bigint, count_8:
bigint, $hashvalue_10:bigint]
            $hashvalue_10 := "combine_hash"(
BIGINT '0', COALESCE("$operator$hash_code"("regionkey"), 0))
              - Aggregate(PARTIAL)[regionkey] => [
regionkey:bigint, count_8:bigint]
                count_8 := "count"(*)
                - TableScan[tpch:tpch:nation:sf0.1,
originalConstraint = true] => [regionkey:bigint]
                  regionkey := tpch:regionkey

```

— Example two

Distributed plan:

```

presto:tiny> EXPLAIN (TYPE DISTRIBUTED) SELECT regionkey, count
(*) FROM nation GROUP BY 1;

```

Query Plan

```

-----
Fragment 0 [SINGLE]
  Output layout: [regionkey, count]
  Output partitioning: SINGLE []
  - Output[regionkey, _coll] => [regionkey:bigint, count:bigint
]
    _coll := count
    - RemoteSource[1] => [regionkey:bigint, count:bigint]
Fragment 1 [HASH]
  Output layout: [regionkey, count]
  Output partitioning: SINGLE []
  - Aggregate(FINAL)[regionkey] => [regionkey:bigint, count:
bigint]
    count := "count"("count_8")
    - LocalExchange[HASH][$hashvalue] ("regionkey") =>
regionkey:bigint, count_8:bigint, $hashvalue:bigint
      - RemoteSource[2] => [regionkey:bigint, count_8:
bigint, $hashvalue_9:bigint]
Fragment 2 [SOURCE]
  Output layout: [regionkey, count_8, $hashvalue_10]
  Output partitioning: HASH [regionkey][$hashvalue_10]
  - Project[] => [regionkey:bigint, count_8:bigint, $hashvalue_
10:bigint]
    $hashvalue_10 := "combine_hash"(BIGINT '0', COALESCE
("$operator$hash_code"("regionkey"), 0))

```

```

- Aggregate(PARTIAL)[regionkey] => [regionkey:bigint,
count_8:bigint]
    count_8 := "count"(*)
- TableScan[tpch:tpch:nation:sf0.1, originalCo
nstraint = true] => [regionkey:bigint]
    regionkey := tpch:regionkey

```

— Example three:

Validation:

```

presto:tiny> EXPLAIN (TYPE VALIDATE) SELECT regionkey, count(*)
FROM nation GROUP BY 1;
Valid
-----
true

```

EXPLAIN ANALYZE

- Synopsis

```
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 detailed information and low-level statistics.

- Examples

In the following example, you can see the CPU time spent in each stage, as well as the relative cost of each plan node in the stage. Note that the relative cost of the plan nodes is based on wall time, which may or may not be correlated to CPU time. For each plan node you can see some additional statistics, which are useful if you want to detect data anomalies for a query (skewness, abnormal 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
), $hashvalue_10:bigint, count_8:bigint]
        Cost: 4.47%, Output: 4000 rows (148.44kB)
        Input avg.: 204514.50 lines, Input std.dev.: 0.05%
        Collisions avg.: 5701.28 (17569.93% est.), Collisions
std.dev.: 1.12%
        count_8 := "count"(*)
        - ScanFilterProject[table = tpch:tpch:orders:sf1.0,
originalConstraint = ("orderdate" > "$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")
...

```

GRANT

- **Synopsis**

```

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

```

```
[ WITH GRANT OPTION ]
```

- Description

Grants the specified privileges to the specified grantee.

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

- Examples

```
GRANT INSERT, SELECT ON orders TO alice; --- Grant privileges to
user alice
GRANT SELECT ON nation TO alice WITH GRANT OPTION; --- Grant SELECT
privilege to user alice, additionally allowing alice to grant **
SELECT** privilege to others
GRANT SELECT ON orders TO PUBLIC; --- Grant **SELECT** privilege on
the table order to everyone
```

- Limitations

Some connectors have no support for `GRANT`.

INSERT

- Synopsis

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

- Description

Inserts new rows into a table. If the list of column names is specified, they must exactly match the list of columns produced 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; --- Insert the SELECT
results into the orders table.
INSERT INTO cities VALUES (1, 'San Francisco'); --- Insert a single
row
INSERT INTO cities VALUES (2, 'San Jose'), (3, 'Oakland'); ---
Insert multiple rows
INSERT INTO nation (nationkey, name, regionkey, comment) VALUES (26,
'POLAND', 3, 'no comment'); --- Insert a single row
```

```
INSERT INTO nation (nationkey, name, regionkey) VALUES (26, 'POLAND', 3); --- Inserts a single row (only includes some columns)
```

PREPARE

- Synopsis

```
PREPARE statement_name FROM statement
```

- Description

Prepares a statement for execution at a later time. Prepared statements are queries saved in a session with a given name. The statement can include parameters in place of literals to be replaced at execution time. Parameters are represented by ?.

- Examples

```
--- Prepare a query that does not include parameters
PREPARE my_select1 FROM
SELECT * FROM nation;
--- Prepare a query that includes parameters
PREPARE my_select2 FROM
SELECT name FROM nation WHERE regionkey = ? AND nationkey < ? ;
--- Prepare an insert statement that does not include parameters
PREPARE my_insert FROM
INSERT INTO cities VALUES (1, 'San Francisco');
```

RESET SESSION

- Synopsis

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

- Description

Reset a session property value to the default value.

- Examples

```
RESET SESSION optimize_hash_generation;
RESET SESSION hive.optimized_reader_enabled;
```

REVOKE

- Synopsis

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

- Description

Revokes the specified privileges from the specified grantee.

- Specifying `ALL PRIVILEGE` revokes `SELECT`, `INSERT` and `DELETE` privileges.
- Specifying `PUBLIC` revokes privileges from the `PUBLIC` role. Users will retain privileges assigned to them directly or via other roles.
- The optional `GRANT OPTION FOR` clause also revokes the privileges to `GRANT` the specified privileges.
- Usage of the term `grantee` denotes both users and roles.
- Examples

```

--- Revoke INSERT and SELECT privileges on the table orders from
user alice
REVOKE INSERT, SELECT ON orders FROM alice;
--- Revoke SELECT privilege on the table nation from everyone,
--- additionally revoking the privilege to grant SELECT privilege to
others
REVOKE GRANT OPTION FOR SELECT ON nation FROM PUBLIC;
--- Revoke all privileges on the table test from user alice
REVOKE ALL PRIVILEGES ON test FROM alice;

```

- Limitations

Some connectors have no support for `REVOKE`.

ROLLBACK

- Synopsis

```
ROLLBACK [ WORK ]
```

- Description

Rollback the current transaction.

- Examples

```

ROLLBACK;
ROLLBACK WORK;

```

SELECT

- Synopsis

```

[ WITH with_query [, ...] ]
SELECT [ ALL | DISTINCT ] select_expr [, ...]
[ 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 ] ]
```

where `from_item` is one of:

```
Table_name [[as] alias [(column_alias [,...]) ] ]
```

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

and `join_type` is one of:

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

and `grouping_element` is one of:

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

Retrieve rows from zero or more tables to get data sets.

- **WITH clause**

— Basic functions

The WITH clause defines named relations for use within a query. It allows flattening nested queries or simplifying subqueries. For example, the following 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 looks to be
much clearer
WITH x AS (SELECT a, MAX(b) AS b FROM t GROUP BY a)
SELECT a, b FROM x;
```

— Define multiple subqueries

The WITH clause can be used 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 chain:

```
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;
```

• GROUP BY clause

— Basic functions

The `GROUP BY` clause divides the output of a `SELECT` statement into groups of rows containing matching values. A simple `GROUP BY` clause may contain any expression composed of input columns or it may be an ordinal number selecting an output column by position (starting at one).

The following queries are equivalent (position for the `nationkey` column is two).

```
--- Using the ordinal number
SELECT count(*), nationkey FROM customer GROUP BY 2;
--- Using the input column name
SELECT count(*), nationkey FROM customer GROUP BY nationkey;
```

`GROUP BY` clauses can group output by input column names not appearing in the output of a select statement, for example:

```
--- The mktsegment column has not been specified in the SELECT
list.
--- The result set does not contain content of the mktsegment
column.
SELECT count(*) FROM customer GROUP BY mktsegment;
_col0
-----
29968
30142
30189
29949
29752
```

(5 rows)



Note:

When a `GROUP BY` clause is used in a `SELECT` statement, all 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, which allows users to perform analysis that requires aggregation on multiple sets of columns in a single query:

- **GROUPING SETS**

CUBE ROLLUP

The shipping table is a data table with five columns, which are shown as follows:

```
SELECT * FROM shipping;
  origin_state | origin_zip | destination_state | destination
n_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)
```

Now we want to retrieve the following grouping results using a single query statement:

- 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` allows users to retrieve the result set of the above three groups with a single query statement, as shown below:

```
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 query with the complex grouping syntax (such as `GROUPING SETS`) only reads from the underlying data source once, while the query with the `UNION ALL` reads the underlying data three times. This is why queries with a `UNION ALL` may produce inconsistent results when the data source is not deterministic.

- **CUBE**

The **CUBE** operator generates all possible grouping sets for a given set of columns. For example, the query:

```
SELECT origin_state, destination_state, sum(package_weight)
FROM shipping
Group by cube (glas_state, destiny _ 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)
```

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. For example, the query:

```
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)

is equivalent to:

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

- **Combining 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));
```

Output results are as 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. For example, the query:

```
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);
```

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. However, 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);
```

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 set quantifier for `GROUP BY BY` is `ALL`.

— GROUPING operation

Presto provides a `grouping` operation that returns a bit set converted to decimal, indicating which columns are present in a grouping. The semantics is demonstrated as follows:

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

`grouping` is used in conjunction with `GROUPING SETS`, `ROLLUP`, `CUBE`, or `GROUP BY`. `grouping` columns must match exactly the columns 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 |
| --- 011 | | | | |
| New Jersey | NULL | NULL | 225 | 3 |
| --- 011 | | | | |

| | | | | |
|------------|-------|-------------|------|---|
| New York | NULL | NULL | 3 | 3 |
| --- 011 | | | | |
| California | 94131 | NULL | 60 | 1 |
| --- 001 | | | | |
| New Jersey | 7081 | NULL | 225 | 1 |
| --- 001 | | | | |
| California | 90210 | NULL | 1337 | 1 |
| --- 001 | | | | |
| New York | 10002 | NULL | 3 | 1 |
| --- 001 | | | | |
| NULL | NULL | New Jersey | 58 | 6 |
| --- 100 | | | | |
| NULL | NULL | Connecticut | 1562 | 6 |
| --- 100 | | | | |
| NULL | NULL | Colorado | 5 | 6 |
| --- 100 | | | | |
| (10 rows) | | | | |

As shown in the preceding table, bits are assigned to the argument 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 to 1 otherwise.

- **HAVING clause**

The `HAVING` clause is used in conjunction with aggregate functions and the `GROUP BY` clause to control which groups are selected. A `HAVING` clause will be executed after completion of grouping and aggregation, to eliminate groups that do not satisfy the given conditions.

The following example selects user groups with an account balance greater than 5700000:

```
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) | | | |

- **Set operations**

Presto supports three set operations, namely `UNION`, `INTERSECT`, and `EXCEPT`. These clauses are used to combine 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 argument `ALL` or `DISTINCT` controls which rows are included in the final result set, and the default is `DISTINCT`.

- `ALL`: may return duplicated rows;
- `parmnamepar DISTINCTparmname` : eliminates duplicated rows.

The `ALL` argument is not supported for `INTERSECT` or `EXCEPT`.

The above three set operations are processed left to right, and `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 or not to remove duplicates.

- **Example one**

```
SELECT 13
UNION
Select 42;
 _col0
-----
      13
      42
(2 rows)
```

- **Example two**

```
SELECT 13
UNION
SELECT * FROM (VALUES 42, 13);
 _col0
-----
      13
      42
(2 rows)
```

- **Example three:**

```
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 row)

```

- **EXCEPT**

EXCEPT returns the rows that are in the result set of the first query, but not the second.

```

SELECT * FROM (VALUES 13, 42)
EXCEPT
SELECT 13;
  _col0
-----
      42
(1 row)

```

- **ORDER BY clause**

The **ORDER BY** clause is used to sort a result set. The semantics is demonstrated as follows:

```

ORDER BY expression [ ASC | DESC ] [ NULLS { FIRST | LAST } ]
[ , ... ]

```

Where:

- Each **expression** may be composed of output columns or it may be an ordinal number selecting an output column by position (starting at one).
- The **ORDER BY** clause is the last step of a query after any **GROUP BY** or **HAVING** clause;
- **NULLS { FIRST | LAST }** is used to control the sorting method of the **NULL** value (regardless of **ASC** or **DESC**), and the default null ordering is **LAST**.
- **LIMIT** clause

The **LIMIT** clause restricts the number of rows in the result set. **LIMIT ALL** is the same as omitting the **LIMIT** clause.

Examples

In this example, because the query lacks an ORDER BY, exactly which rows are returned is arbitrary.

```
SELECT orderdate FROM orders LIMIT 5;
orderdate
-----
1996-04-14
1992-01-15
1995-02-01
1995-11-12
1992-04-26
(5 rows)
```

- TABLESAMPLE

Presto provides two sampling methods, namely `BERNOULLI` and `SYSTEM`. However, neither of the two methods allow deterministic bounds on the number of rows returned.

- `BERNOULLI`:

Each row is selected to be in the table sample with a probability of the sample percentage. When a table is sampled using the Bernoulli method, all physical blocks of the table are scanned and certain rows are skipped based on a comparison between the sample percentage and a random value calculated at runtime.

The probability of a row being included in the result is independent from any other row.

This does not reduce the time required to read the sampled table from disk. It may have an impact on the total query time if the sampled output is processed further.

- `SYSTEM`

This sampling method divides the 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 at runtime).

The rows selected in a system sampling is dependent on which connector is used. For example, when used with Hive, it is dependent on how the data is laid out on HDFS. This method does not guarantee independent sampling probabilities.

Examples

```
--- Using BERNOULLI sampling
SELECT *
FROM users TABLESAMPLE BERNOULLI (50);
--- Using system sampling
SELECT *
FROM users TABLESAMPLE SYSTEM (75);
```

```
Using sampling with joins:
--- Using sampling with JOIN
SELECT o.*, i. *
FROM orders o TABLESAMPLE SYSTEM (10)
JOIN lineitem i TABLESAMPLE BERNOULLI (40)
  ON o.orderkey = i.orderkey;
```

- **UNNEST**

UNNEST can be used to expand an ARRAY or MAP into a relation. Arrays are expanded into a single column, and maps are expanded into two columns (key, value). **UNNEST** can also be used with multiple arrays and maps, in which case they are expanded into multiple columns, with as many rows as the highest cardinality argument (the other columns are padded with nulls). **UNNEST** can optionally have a **WITH ORDINALITY** clause, in which case an additional ordinal column is added to the end. **UNNEST** is normally used with a **JOIN** and can reference columns from relations on the left side of the join.

- Example one

```
--- Using a single column
SELECT student, score
FROM tests
CROSS JOIN UNNEST(scores) AS t (score);
```

- Example two

```
--- Using 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 three:

```
--- Using 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 | 5 |
| [7, 8, 9] | 7 | 8 |
| [7, 8, 9] | 8 | 9 |

```

-----+-----+-----
 [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)

```

— Joins

Joins allow you to combine data from multiple relations. A `CROSS JOIN` returns the *Cartesian product* of two relations (all combinations). `CROSS JOIN` can either be specified using

- the explicit `CROSS JOIN` syntax, or
- by specifying multiple relations in the `FROM` clause.

Both of the following queries are equivalent:

```

--- using the explicit **CROSS JOIN** syntax
SELECT *
FROM nation
CROSS JOIN region;
--- specifying multiple relations in the **FROM** clause
VALUES
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 two relations in a join have columns with the same name, the column references must be qualified using the relation name (or alias).

```

--- 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;
--- Wrong, it will raise the "Column 'name' is ambiguous" error
SELECT name
FROM nation
CROSS JOIN region;

```

— Subquery

A subquery is an expression which is composed of a query. The subquery is correlated when it refers to columns outside of the subquery. Presto has limited support for correlated subqueries.

■ EXISTS

The `EXISTS` predicate determines if a subquery returns any rows. If subquery returns any rows, the `WHERE` expression is `TRUE`, and `FALSE` if otherwise.

Examples

```

SELECT name
FROM nation
WHERE EXISTS (SELECT * FROM region WHERE region.regionkey =
nation.regionkey);

```

■ IN

The `IN` predicate determines if any columns specified by `WHERE` are included in the result set produced by the subquery. If yes, it returns results, and does not return results if otherwise. The subquery must produce exactly 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 produce more than one row. The returned value is `NULL` if the subquery produces no rows.

Examples

```

SELECT name
FROM nation

```

```
WH ERregionkey = (SELECT max(regionkey) FROM regio);
```

SET SESSION

- Synopsis

```
SET SESSION name = expression  
SET SESSION catalog.name = expression
```

- Description

Sets a session property value.

- Examples

```
SET SESSION optimize_hash_generation = true;  
SET SESSION hive.optimized_reader_enabled = true;
```

SHOW CATALOGS

- Synopsis

```
SHOW CATALOGS [ LIKE pattern ]
```

- Description

Lists the available catalogs. The `LIKE` clause can be used to filter the catalog names.

- Examples

```
SHOW CATALOGS;
```

SHOW COLUMNS

- Synopsis

```
SHOW COLUMNS FROM table
```

- Description

Lists the columns in a given table along with their data type and other attributes.

- Examples

```
SHOW COLUMNS FROM orders;
```

SHOW CREATE TABLE

- Synopsis

```
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,
  orderstatus varchar,
  totalprice double,
  orderdate varchar
)
WITH (
  format = 'ORC',
  partitioned_by = ARRAY['orderdate']
)
(1 row)
```

SHOW CREATE VIEW

- Synopsis

```
SHOW CREATE VIEW view_name
```

- Description

Shows the SQL statement that creates the specified view.

- Examples

```
SHOW CREATE VIEW view1;
```

SHOW FUNCTIONS

- Synopsis

```
SHOW FUNCTIONS
```

- Description

List all the functions available for use in queries.

- Examples

```
SHOW FUNCTIONS
```

SHOW GRANTS

- Synopsis

```
SHOW GRANTS [ ON [ TABLE ] table_name ]
```

- Description

Lists the grants for the current user on the specified table in the current catalog.

- Examples

```
--- List the grants for the current user on table orders
SHOW GRANTS ON TABLE orders;
--- List the grants for the current user on all the tables in all
schemas of the current catalog
SHOW GRANTS;
```

- Limitations

Some connectors have no support for `SHOW GRANTS`.

SHOW SCHEMAS

- Synopsis

```
SHOW SCHEMAS [ FROM catalog ] [ LIKE pattern ]
```

- Description

Lists all schemas in the specified catalog, or in the current catalog if no catalog has been specified. The `LIKE` clause can be used to filter the schema names.

- Examples

```
SHOW SCHEMAS;
```

SHOW SESSION

- Synopsis

```
SHOW SESSION
```

- Description

Lists the current session properties.

- Examples

```
SHOW SESSION
```

SHOW TABLES;

- Synopsis

```
SHOW TABLES [ FROM schema ] [ LIKE pattern ]
```

- Description

Lists all tables in the specified schema, or in the current schema if no schema has been specified. The `LIKE` clause can be used to filter the table name.

- Examples

```
SHOW TABLES;
```

START TRANSACTION

- Synopsis

```
START TRANSACTION [ mode [, ...] ]  
where **mode** is one of:  
ISOLATION LEVEL { READ UNCOMMITTED | READ COMMITTED | REPEATABLE  
READ | SERIALIZABLE }  
READ { ONLY | WRITE }
```

- Description

Starts a new transaction for the current session.

- Examples

```
START TRANSACTION;  
START TRANSACTION ISOLATION LEVEL REPEATABLE READ;  
START TRANSACTION READ WRITE;  
START TRANSACTION ISOLATION LEVEL READ COMMITTED, READ ONLY;  
START TRANSACTION READ WRITE, ISOLATION LEVEL SERIALIZABLE;
```

USE

- Synopsis

```
USE catalog.schema  
USE schema
```

- Description

Updates the session to use the specified catalog and schema. If a catalog is not specified, the schema is resolved relative to the current catalog.

- Examples

```
USE hive.finance;  
USE information_schema;
```

VALUES

- Synopsis

```
VALUES row [, ...]  
where **row** is a single expression or
```

```
( column_expression [, ...] )
```

- Description

Defines a literal inline table.

- `VALUE` can be used anywhere a query can be used. For example, behind the `FROM` clause of a `SELECT`, in an `INSERT`, or even at the top level.
- `VALUE` creates an anonymous table without column names by default. The table and columns can be named 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')
--- Using 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)
```

11.8.7 Technical support

Technical support

For any questions, contact technical support:

- [Open a ticket](#)

11.9 TensorFlow instructions

[TensorFlow](#) is supported by E-MapReduce 3.13.0 and later versions. Users can add the TensorFlow component from available services in software configurations. When using

TensorFlow in E-MapReduce to perform high-performance computing, you can allocate CPU and GPU resources through YARN.

Preparations

- 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 type families with GPU, such as gn5 and gn6, for the core nodes and task nodes in the cluster. Compute optimized type families with GPU support heterogeneous computing. After determining the instance type, choose the CUDA toolkit and cuDNN versions as required.

Submit TensorFlow jobs

Users 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_tensorboard ENABLE_TENSORBOARD]
[-log_tensorboard LOG_TENSORBOARD] [-conf CONF] [-f FILES]
[-pn PS_NUM] [-pc PS_CPU] [-pm PS_MEMORY] [-wn WORKER_NUM]
[-wc WORKER_CPU] [-wg WORKER_GPU] [-wm WORKER_MEMORY]
[-wnpg WNPG] [-ppn PPN] [-c COMMAND [COMMAND ...]]
```

Description of basic parameters:

- The `-t APP_TYPE` parameter 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 the tasks to one instance in the YARN cluster for execution. It is similar to a standalone execution.

- The `-a APP_NAME` parameter specifies the name of the submitted TensorFlow job. Users can name jobs as needed.
- The `-m MODE` parameter 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 EMR 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 EMR 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`: Users need to exit the parameter servers manually for some APIs of distributed TensorFlow. To exit parameter servers automatically when worker servers finish training their models, specify the `-x` option.
- The `-enable_tensorboard` parameter specifies whether to enable TensorBoard when TensorFlow starts training models.
- The `-log_tensorboard` parameter specifies the location of TensorBoard logs on HDFS. If TensorBoard is enabled when TensorFlow starts training models, then this parameter is required.
- The `-conf CONF` parameter specifies the location of the Hadoop configuration. It is optional to set the value. The default EMR configuration is used.
- The `-f FILES` parameter 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, users can put all dependencies in one folder, the script automatically uploads the folders into HDFS according to the folder hierarchy.
- The `-pn TensorFlow` parameter specifies the number of parameter servers to start.
- The `-pc` parameter specifies the number of CPU cores that each parameter server requests.
- The `-pm` parameter specifies the memory size that each parameter server requests.
- The `-wn` parameter specifies the number of worker nodes started by TensorFlow.

- The `-wc` parameter specifies the number of CPU cores that each worker requests.
- The `-wg` parameter specifies the number of GPU cores that each worker requests.
- The `-wm` parameter specifies the memory usage that each worker requests.
- The `-c COMMAND` parameter specifies the command to run. For example, `pythoncensus.py`.

Advanced options. Users need to carefully use advanced options, which may cause failure of running jobs.

- The `-wnpg` parameter specifies the number of workers that use a GPU simultaneously (for `tensorflow-ps`).
- The `-ppn` parameter 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.

11.10 Knox guide

Currently E-MapReduce supports [Apache Knox](#). If you select the Knox-supported image to create a cluster, you can directly access the Web UI from the public network to use services such as YARN, HDFS, and SparkHistory after completing the following preparations.

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.



Note:

- For security, 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 the 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):

In the [User Management](#) page, add Knox account directly.

Method Two

1. Log on to the cluster over SSH. See [SSH Logon to Clusters](#) for detailed steps.
2. Prepare your user data, for example, user Tom. 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
```



Note:

For security, before you export your user data to LDAP, change the password of `users.ldif` by setting `userPassword` to your password.

3. Export to LDAP.

```
su Knox
cd /usr/lib/knox-current/templates
sh ldap-sample-users.sh
```

— Use your LDAP service

1. In the cluster configuration management, find the Knox configuration management. 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.

```
cluster-topo
xml-direct-to-file-content
</param>
<param>
<name>main.ldapRealm.userDnTemplate</name>
<value>uid=(0),ou=people,dc=emr,dc=com</value>
</param>
<param>
<name>main.ldapRealm.contextFactory.url</name>
<value>ldap://{{hostname_master_main}}:10389</value>
```

2. Generally, your LDAP service is not running in the cluster. You must enable the Knox port for accessing the LDAP service in the public network, for example, Port 10389. For more information, see the preceding steps for enabling Port 8443. Select **Internet outbound**.

**Note:**

For security, 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 shortcut link of E-MapReduce
 1. Log on to the [E-MapReduce console](#).
 2. Click the ID link of the target cluster.
 3. In the left-side navigation pane, click **Clusters and Services**.
 4. Click the relevant services on the EMR Service console 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 cluster details.
 2. Access the URLs of 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. The browser shows **website is not security** because the Knox service uses the 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 (ACLs)

Knox provides service-level permission management to limit service access to specific users, user groups, or IP addresses. See [Apache Knox Authorization](#).

- Example:
 - Scenario: YARN UI only allows the access by user Tom.

- Steps: In the cluster configuration management, find the Knox configuration management and the cluster-topo configuration, and add ACL code between `<gateway>...</gateway>` labels in the cluster-topo configuration.

```
<provider>
  <role>authorization</role>
  <name>AclsAuthz</name>
  <enabled>true</enabled>
  <param>
    <name>YARNUI.acl</name>
    <value>Tom; *; *</value>
  </param>
</provider>
```

- Notes:

Knox provides REST APIs for operating a range of services, for example, adding or deleting HDFS files. For security, make sure the authorization object is your limited IP address range when you enable Knox Port 8443 of the security group in the ECS console. 0.0.0.0/0 is forbidden. Do not use the LDAP username and password in the Knox installation directory to access Knox.

11.11 Instructions for using Flume

E-MapReduce version 3.16.0 and later support Apache Flume. This topic describes how to use Flume to synchronize E-MapReduce Kafka cluster data to HDFS, Hive, and HBase running on E-MapReduce Hadoop clusters, and to Alibaba Cloud OSS.

Prerequisites

- You must have selected Flume in the **Optional Service** menu when you created a Hadoop cluster.
- You must have created a Kafka cluster and created a topic named flume-test to generate data.



Note:

- If you have created a high security mode Hadoop cluster to consume standard Kafka cluster data, and you need to configure Kerberos authentication on the Hadoop cluster, see [Authentication method compatible with MIT Kerberos](#).
- If you have created a high security mode Kafka cluster and you need to write data to a standard Hadoop cluster using Flume, see [Kerberos Kafka Source in this topic](#).

- If you have created a high security mode Hadoop cluster and a high security mode Kafka cluster, and you need to configure Kerberos, see [Cross-region access](#) and [Cross-region access using Flume](#).

Kafka->HDFS

- Configure Flume

Create a configuration file named `flume.properties`, and add the following configurations for the file, where `a1.sources.source1.kafka.bootstrap.servers` indicates the host and port for a Kafka broker, `a1.sources.source1.kafka.topics` indicates the Kafka topic where Flume is used to consume data, and `a1.sinks.k1.hdfs.path` indicates the path where Flume writes data to HDFS.

```
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
```

- Start Flume

Flume's default configuration file is stored in `/etc/ecm/flume-conf`. Use the following configuration file to start a Flume agent.

```
flume-ng agent --name a1 --conf /etc/ecm/flume-conf --conf-file
flume.properties
```

After the agent is started, the log `logs/flume.log` will be generated in the current path due to `log4j.properties` being in `/etc/ecm/flume-conf`. You can configure `log4j.properties` according to your requirements.

- Test

Use the `kafka-console-producer.sh` command and input the test data `abc` in your Kafka cluster.

```
[root@emr-header-1 ~]# kafka-console-producer.sh --topic flume-test --broker-list emr-header-1:9092
>abc
>
```

Flume generates a file `FlumeData.xxx` with a timestamp (in milliseconds) suffix based on the current time. When you view the file content, you can see the data that you input in Kafka.

```
[root@emr-header-1 ~]# hdfs dfs -cat /tmp/flume/test-data/FlumeData.1543386053173
abc
[root@emr-header-1 ~]#
```

Kafka->Hive

- Create a Hive table

Before Flume writes data into Hive using transactions, you need to set the `transactional` property when creating a Hive table. The following example shows how to create a table named `flume_test` table.

```
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 `flume.properties` and add the following configurations for the file, where `a1.sources.source1.kafka.bootstrap.servers` indicates the host and port for a Kafka broker and `a1.sinks.k1.hive.metastore` indicates a Hive metastore URI. Then, configure the value of `hive.metastore.uris` 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` command and input the comma-separated test data 1,a in your Kafka cluster.

- Verify the input data

Note that querying Hive transaction tables require configuration on the Hive client:

```

hive.support.concurrency - true
hive.exec.dynamic.partition.mode - nonstrict
hive.txn.manager - org.apache.hadoop.hive ql.lockmgr.DbTxnManager

```

After the preceding configurations are set, you can query data in the `flume_test` table.

```

hive> set hive.support.concurr
hive> set hive.exec.dynamic.po
hive> set hive.txn.manager=org
hive> select * from flume_test
OK
1      a
Time taken: 0.149 seconds, Fet

```

Kafka->HBase

- Create a HBase table

Create a HBase table `flume_test` and a column family `column`.

```
hbase(main):001:0> create 'flume_test', 'column'
0 row(s) in 1.3940 seconds

=> Hbase::Table - flume_test
hbase(main):002:0>
```

- **Configure Flume**

Create a configuration file `flume.properties` and add the following configurations, where `a1.sources.source1.kafka.bootstrap.servers` indicates the host and port for a Kafka broker, `a1.sinks.k1.table` indicates the name of the HBase table, and `a1.sinks.k1.columnFamily` indicates the name of the column family:

```
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 = hbase
a1.sinks.k1.table = flume_test
a1.sinks.k1.columnFamily = 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**

```
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.

```

=> ["flume_test"]
hbase(main):003:0> scan 'flume_test'
ROW COLUMN+CELL
defaultf2add0ee-5040-4 column=column:pCol, timestamp=1543493834351, value=data
7dc-b002-f269b679977b
incRow column=column:iCol, timestamp=1543493834373, value=\x00\x00\x00\
x00\x00\x00\x00\x01
2 row(s) in 0.0310 seconds
hbase(main):004:0>

```

Kafka->OSS

- Create an OSS path

Create an OSS bucket and directory, such as `oss://flume-test/result`.

- Configure Flume

Flume requires a large amount of JVM memory when writing data to OSS. To resolve this issue, you can:

- Reduce the OSS cache size

Copy the file `hdfs-site.xml` from `/etc/ecm/hadoop-conf` to `/etc/ecm/flume-conf`, and reduce the value of the configuration term `smartdata.cache.buffer.size`, for example, to 1048576.

- Increase the Flume agent's heap size (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 `flume.properties` and add the following configurations, where `a1.sources.source1.kafka.bootstrap.servers` indicates the host and port for a Kafka broker, and `a1.sinks.k1.hdfs.path` indicates an 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 data is generated using `kafka-console-producer.sh` in your Kafka cluster, in the OSS path `oss://flume-test/result`, a file `FlumeData.xxxx` is generated with a timestamp (in milliseconds) suffix based on the current time.

Kerberos Kafka source

If high security Kafka cluster data is consumed, you must configure the following variables:

- 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`

In the `flume.properties` file, add the following configurations:

```
a1.sources.source1.kafka.consumer.security.protocol = SASL_PLAINTEXT
a1.sources.source1.kafka.consumer.sasl.mechanism = GSSAPI
a1.sources.source1.kafka.consumer.sasl.kerberos.service.name = kafka
```

- Configure the Kafka client

— In `/etc/ecm/flume-conf`, create the file `flume_jaas.conf` with the following content:

```
KafkaClient {
```


In the `flume.properties` file, add the following configurations:

```
a1.sources.source1.kafka.consumer.security.protocol = SASL_PLAINTEXT
a1.sources.source1.kafka.consumer.sasl.mechanism = GSSAPI
a1.sources.source1.kafka.consumer.sasl.kerberos.service.name = kafka
```

- Configure the Kafka client

- In `/etc/ecm/flume-conf`, create the file `flume_jaas.conf` with the following content:

```
KafkaClient {
  com.sun.security.auth.module.Krb5LoginModule required
  useKeyTab=true
  storeKey=true
  keyTab="/etc/ecm/flume-conf/test.keytab"
  serviceName="kafka"
  principal="test@EMR.${realm}.COM";
};
```

where, `${realm}` is replaced with a Kerberos realm of your Kafka cluster. To obtain a Kerberos realm, in your Kafka cluster, run the command `hostname` to obtain a hostname in the form of `emr-header-1.cluster-xxx`, such as `emr-header-1.cluster-123456`.

The last numeric string 123456 is a Kerberos realm.

- Change `/etc/ecm/flume-conf/flume-env.sh`

By default, in `/etc/ecm/flume-conf/`, the file `flume-env.sh` does not exist. You need to copy `flume-env.sh.template`, paste it to `/etc/ecm/flume-conf/`, rename it `flume-env.sh`, and add the following content:

```
export JAVA_OPTS="$JAVA_OPTS -Djava.security.auth.login.config=/
etc/ecm/flume-conf/flume_jaas.conf"
```

12 OSS ACL

Procedure

E-MapReduce supports using RAM to isolate the data of different sub-accounts. The procedures are shown as follows:

1. Log on to the [Alibaba Cloud RAM console](#).
2. Create the sub-account in RAM with the process in [How to create a sub-account in RAM](#).
3. In the left-side navigation pane, click **policies** to enter the page of policy management.
4. Click **Custom Policy**.
5. In the upper-right corner, click **Create Authorization Policy** enter the authorization policy creation page. Create the policy according to the prompted steps. You can create as many policies as the sets of authorization control you need.

It is assumed that you need the following two sets of data control policies:

- Testing environment, bucketname: test-bucket. The corresponding complete policy is as follows.

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "oss:ListBuckets"
      ],
      "Resource": [
        "acs:oss:*:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "oss:Listobjects",
        "oss:GetObject",
        "oss:PutObject",
        "oss>DeleteObject"
      ],
      "Resource": [
        "acs:oss:*:*:test-bucket",
        "acs:oss:*:*:test-bucket/*"
      ]
    }
  ]
}
```

```
}

```

- Production environment, bucketname: prod-bucket. The corresponding complete policy is as follows:

```
{
  "Version": "1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "oss:ListBuckets"
      ],
      "Resource": [
        "acs:oss:*:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "oss:Listobjects",
        "oss:GetObject",
        "oss:PutObject"
      ],
      "Resource": [
        "acs:oss:*:*:prod-bucket",
        "acs:oss:*:*:prod-bucket/*"
      ]
    }
  ]
}
```

6. Click **Users**.
7. Find out the sub-account item which the policy is given to and click the **Manage** button.
8. In the left-side navigation pane, click **User Authorization Policy**.
9. In the upper-right corner, click **Edit Authorization Policy**.
10. Select and add authorization policies.
11. Click **OK** to complete the policy authorization of the sub-account.
12. In the upper-right corner, click **User Details** to enter the user details page of the sub-account.
13. Click **Start Console Logon** in the Web console logon management bar to start up the authorization of the sub-account logon console.

Complete and use

After completing all preceding steps, use the corresponding sub-account to log on to E-MapReduce with following limits:

- All buckets can be seen in the OSS selection interface for cluster, operation, and plan execution creations, but the authorized bucket can only be entered.
- The content under authorized bucket can only be seen, rather than those under other buckets.

- The authorized bucket can only be read and written. Otherwise, an error is reported.

13 Connect to clusters using SSH

If the existing jobs and execution plans cannot meet your more complex application requirements, log on to the master node of the E-MapReduce cluster. Here, navigate to the cluster details page where the public network IP address of the cluster master exists. You can log on to the master node through SSH to view various settings and states.

The function of logging clusters

Relevant environment variables have been set on the machine, including the following common ones:

- JAVA_HOME
- HADOOP_HOME
- HADOOP_CONF_DIR
- HADOOP_LOG_DIR
- YARN_LOG_DIR
- HIVE_HOME
- HIVE_CONF_DIR
- PIG_HOME
- PIG_CONF_DIR

You can quote these variables in the script, however, we recommend that you do not change them to avoid unexpected E-MapReduce errors.

Connect to the Master

1. SSH logs on to the master with the following commands. Obtain the public network IP of the cluster master in the host information column on the [Cluster details](#).

```
ssh root@ip.of.master
```

2. Enter the password set during creation.

Connect to a cluster using SSH without a password

You must connect to the cluster for management and operation. To connect to the cluster master, you can break through the SSH password-less logon from the master machine (by default, the cluster master opens up the public network IP). The procedure is as follows:

1. Connect to the master with the root and password mode as mentioned previously.
2. Change to Hadoop or hdfs user.

SSH mode of Linux

1. Copy the private key to the local machine.

```
sz ~/.ssh/id_rsa
```

2. Return to your local machine and attempt to connect to the master again.

```
ssh -i private_key_path/id_rsa hadoop@server_ip_address
```

If only one private key exists, you can put it in your `~/.ssh/` and use it by default without designation of `-i`.

Connect to the Master Node using SSH on Windows

You can connect to the master through SSH without input password with multiple methods under Windows.

- Method I: PuTTY

1. Click [Download PuTTY](#).
2. Download PuTTYgen from the same location.
3. Open PuTTYgen and load your private key.



Note:

Keep the private key safe. In case of accidental disclosure, generate a new private key immediately for replacement.

4. Use default configuration and save the private key. Obtain a secret PuTTY key file with a suffix of `ppk`.
 5. Operate PuTTY and select Session on the configuration page.
 6. Enter the public network IP address of the target machine you will connect to and add the user name (for example, `hadoop@MasterNodeIP`).
 7. Select **Connetion > SSH > Auth** on the configuration page.
 8. Select the generated `ppk` file.
 9. Click **Open** to log on to the master node automatically.
- Method II: Cygwin | MinGW

It is a convenient tool to simulate Linux env in Windows.

For this method, see the preceding SSH method of Linux.

MinGW method is recommended for use because it is the most compact. If the official website cannot be opened, download a Git client. The default Git Bash can be used.

View webui of Hadoop, Spark, Ganglia, and other systems



Note:

Confirm you have finished the preceding [SSH logon without password](#) process before this step.

For safety, the webui monitoring system ports of Hadoop, Spark, Ganglia, and other systems in the E-MapReduce cluster are not opened to the outside world. If you want to visit these webUIs, a SSH tunnel needs to be built to forward through a port. The following two methods are available:



Note:

The following operations are completed in your local machine, instead of the machine in the cluster.

- Method I: Port dynamic forwarding

Create a SSH tunnel that can connect certain dynamic port connections between the local machine and the master machine in E-MapReduce cluster.

```
ssh -i /path/id_XXX -ND 8157 hadoop@masterNodeIP
```

8157 is any port not used in the local machine and can be customized by you.

After dynamic forwarding, you can view the following:

- Recommended methods

We recommend that you use the Chrome browser. Visit Web UI in the following methods:

```
chrome --proxy-server="socks5://localhost:8157" --host-resolver-rules="MAP * 0.0.0.0 , EXCLUDE localhost" --user-data-dir=/tmp/
```

For Windows system, the *tmppath* can be written into similar *d:/tmppath*. For Linux or OSX, */tmp/* can be written directly.

The Chrome location varies with operating systems. See the following table.

| Operating System | Chrome Location |
|------------------|--|
| Mac OS X | /Applications/Google Chrome.app/Contents/MacOS/Google Chrome |
| Linux | /usr/bin/google-chrome |

| Operating System | Chrome Location |
|------------------|---|
| Windows | C:\Program Files (x86)\Google\Chrome\Application\chrome.exe |

- Plug-in method

SSH tunnel between local machine and master machine in the E-MapReduce cluster has been broken through. You need to configure a local agent in order to view webUI of Hadoop, Spark and Ganglia in the browser. The procedure is as follows:

1. For Chrome or Firefox browser, please click [\[Download FoxyProxy Standard agent software\]](#).
2. After installation and restart of browser, open a text editor and edit the following content:

```
<?<!--?--> xml version="1.0" encoding="UTF-8"? >
<foxyproxy>
<proxies>
<proxy name="aliyun-emr-socks-proxy" id="2322596116" notes
=" " fromSubscription="false" enabled="true" mode="manual"
selectedTabIndex="2" lastresort="false" animatedIcons="true
" includeInCycle="true" color="#0055E5" proxyDNS="true"
noInternalIPs="false" autoconfMode="pac" clearCacheBeforeUse
="false" disableCache="false" clearCookiesBeforeUse="false"
rejectCookies="false">
<matches>
<match enabled="true" name="120.*" pattern="http://120.*"
isRegEx="false" isBlackList="false" isMultiLine="false"
caseSensitive="false" fromSubscription="false" ></match>
</matches>
<manualconf host="localhost" port="8157" socksVersion="5"
isSocks="true" username="" password="" domain="" ></manualconf>
</proxy>
</proxies>
</foxyproxy>
```

Specifically:

- Port 8157 is a port which your local machine uses to build SSH connection with cluster master machine. It will match with the port used in SSH command executed in the terminal.
 - 120.* is used to match with IP address in the master machine. Please confirm it based on master IP address.
3. In a browser, click the **Foxyproxy** button and then select **Options**.
 4. Select **Import/Export**.
 5. Select the xml file you edited just now and click **Open**.
 6. In the **Import FoxyProxy Setting** dialogue box, click **Add**.

7. In a browser, click the **Foxyproxy** button and select **Use Proxy aliyun-emr-socks-proxy for all URLs**.
 8. nter localhost:8088 in the browser to open the Hadoop interface at the far end.
- Method II: Local port forwarding

**Note:**

A local port forwarding disadvantage is that only the interface in the outermost layer can be seen. The viewing of detailed job information results in an error.

```
ssh -i /path/id_rsa -N -L 8157:masterNodeIP:8088 hadoop@masterNodeIP
```

Parameter description:

- **path**: Private key storage path.
- **masterNodeIP**: IP address of the master node to be connected.
- **8088**: Access port number of ResourceManager on the master node.

14 Create your gateway

Gateway is an ECS server in the same intranet as the E-MapReduce cluster. You can use Gateway to achieve load balancing and security isolation, or submit jobs to the E-MapReduce cluster.

You can create Gateway in the following two ways:

- (Recommended) Create in [E-MapReduce](#).
- Set up a Gateway manually.

Create a gateway on the E-MapReduce Console

E-MapReduce gateways only support Hadoop clusters. You must create a Hadoop cluster before creating an E-MapReduce gateway. To create an E-MapReduce gateway, follow these steps.

1. Log on to the [E-MapReduce console](#) .

2. Click **Create Gateway**.

3. Configure in the **Create Gateway** page.

- **Billing Method:**

- **Subscription:** Subscription billing method pays for a period one time, the price is relatively cheap compared to Pay-As-You-Go method, especially when you pay for three years one time, the discount is larger.

- **Pay-As-You-Go:** Pay-As-You-Go method is based on the actual number of hours that you used the product to calculate the order, and it charges by hour.

- **Cluster:** Create a gateway for the cluster, that is, the created gateway can submit jobs to which cluster. Gateway will automatically configure the Hadoop environment that is consistent with the cluster.

- **Configuration:** The available ECS instance specifications in the zone.

- **System Disk Type:** The system disk type of the gateway node. There are two types of system disk: SSD cloud disk and efficient cloud disk. The displayed type of system disk varies according to different server models and different regions. The system disk is released with the release of the cluster by default.

- **System Disk Size:** The minimum is 40GB and the maximum is 500GB. The default is 300 GB.

- **Data Disk Type:** The data disk type of the gateway node. There are two types of data disk: SSD Disk and efficient cloud disk. The displayed type of data disk varies according to

different server models and different regions. The data disk is released with the release of the cluster by default.

- **Data Disk Size:** The minimum is 200GB and the maximum is 4000GB. The default value is 300GB.
- **Quantity:** The number of data disks. The minimum is 1 and the maximum is 10.
- **Cluster Name:** The name of a gateway. The length can be 1~64 characters. It only allows Chinese character, letter, number, hyphen (-), and underscore (_).
- **Password/Key Pair:**
 - **Password Mode:** Enter the password for gateway login in the text box.
 - **Key Pair Mode:** Select the key pair name for Gateway login in the drop-down menu. If no key pair has been created yet, click **Create Key Pair** on the right to go to the ECS console to create. Do not disclose the private key file with .pem format corresponding to the key pair. After Gateway is created successfully, the public key of the key pair will be bound to the ECS in which Gateway is located automatically. When you log on to Gateway via SSH, enter the private key in the private key file.

4. Click **Create** to save the configurations.

The newly created gateway will be displayed in the cluster list, and the status in the **Status** column becomes **Idle** when the creation is successful.

Set up a gateway manually

- **Network environment**

Make sure that the Gateway machine is in the security group of the corresponding EMR cluster, so that the Gateway nodes can smoothly access the EMR cluster. For setting the security group of the machine, see the [Create a security](#).

- **Software environment**

- System environment: CentOS 7.2+ is recommended.
- Java environment: JDK 1.7 or later version must be installed and OpenJDK 1.8.0 is recommended.

- **Procedure**

- **EMR 2.7 or later version, 3.2 or later version**

To create Gateway in these versions, we recommend that you use the EMR console.

If you want to set up a Gateway manually, copy the following script to the gateway host and run it. The command is `sh deploy.sh <master_ip> master_password_file`.

- `deploy.sh` is the script name, and the content is as follows.
- `master_ip` is the IP address of the master node in the cluster, which needs to be accessible.
- `master_password_file` is the file for storing password of the master node, which is directly written in the file.

```
#!/usr/bin/bash
If [$ #! = 2]
then
    echo "Usage: $0 master_ip master_password_file"
    exit 1;
fi
masterip=$1
masterpwdfile=$2
if ! type sshpass >/dev/null 2>&1; then
    yum install -y sshpass
fi
if ! type java >/dev/null 2>&1; then
    yum install -y java-1.8.0-openjdk
fi
mkdir -p /opt/apps
mkdir -p /etc/ecm
echo "Start to copy package from $masterip to local gateway(/opt/
apps)"
echo " -copying hadoop-2.7.2"
sshpass -f $masterpwdfile scp -r -o 'StrictHostKeyChecking no'
root@$masterip:/usr/lib/hadoop-current /opt/apps/
echo " -copying hive-2.0.1"
sshpass -f $masterpwdfile scp -r root@$masterip:/usr/lib/hive-
current /opt/apps/
echo " -copying spark-2.1.1"
sshpass -f $masterpwdfile scp -r root@$masterip:/usr/lib/spark-
current /opt/apps/
echo "Start to link /usr/lib/\${app}-current to /opt/apps/\${app}"
if [ -L /usr/lib/hadoop-current ]
then
    unlink /usr/lib/hadoop-current
fi
ln -s /opt/apps/hadoop-current /usr/lib/hadoop-current
if [ -L /usr/lib/hive-current ]
then
    unlink /usr/lib/hive-current
fi
ln -s /opt/apps/hive-current /usr/lib/hive-current
if [ -L /usr/lib/spark-current ]
then
    unlink /usr/lib/spark-current
fi
ln -s /opt/apps/spark-current /usr/lib/spark-current
echo "Start to copy conf from $masterip to local gateway(/etc/ecm
)"
sshpass -f $masterpwdfile scp -r root@$masterip:/etc/ecm/hadoop-
conf /etc/ecm/hadoop-conf
```

```

sshpass -f $masterpwdfile scp -r root@$masterip:/etc/ecm/hive-conf
/etc/ecm/hive-conf
sshpass -f $masterpwdfile scp -r root@$masterip:/etc/ecm/spark-
conf /etc/ecm/spark-conf
echo "Start to copy environment from $masterip to local gateway(/
etc/profile.d)"
sshpass -f $masterpwdfile scp root@$masterip:/etc/profile.d/hdfs.
sh /etc/profile.d/
sshpass -f $masterpwdfile scp root@$masterip:/etc/profile.d/yarn.
sh /etc/profile.d/
sshpass -f $masterpwdfile scp root@$masterip:/etc/profile.d/hive.
sh /etc/profile.d/
sshpass -f $masterpwdfile scp root@$masterip:/etc/profile.d/spark.
sh /etc/profile.d/
if [ -L /usr/lib/jvm/java ]
then
    unlink /usr/lib/jvm/java
fi
echo "" >>/etc/profile.d/hdfs.sh
echo export JAVA_HOME=/usr/lib/jvm/jre-1.8.0 >>/etc/profile.d/hdfs
.sh
echo "Start to copy host info from $masterip to local gateway(/etc
/hosts)"
sshpass -f $masterpwdfile scp root@$masterip:/etc/hosts /etc/
hosts_bak
cat /etc/hosts_bak | grep emr | grep cluster >>/etc/hosts
if ! id hadoop >& /dev/null
then
    useradd hadoop
fi

```

— EMR 2.7 earlier version, 3.2 earlier version

Copy the following script to the Gateway host and run it. The command is **sh deploy.sh**

<master_ip> master_password_file.

- *deploy.sh* is the script name, and the content is as follows.
- *master_ip* is the IP address of the master node in the cluster, which needs to be accessible.
- *master_password_file* is the file for storing password of the master node, which is directly written in the file.

```

! /usr/bin/bash
if [ $# != 2 ]
then
    echo "Usage: $0 master_ip master_password_file"
    exit 1;
fi
masterip=$1
masterpwdfile=$2
if ! type sshpass >/dev/null 2>&1; then
    yum install -y sshpass
fi
if ! type java >/dev/null 2>&1; then
    yum install -y java-1.8.0-openjdk
fi
mkdir -p /opt/apps

```

```

mkdir -p /etc/emr
echo "Start to copy package from $masterip to local gateway(/opt/
apps)"
echo " -copying hadoop-2.7.2"
sshpass -f $masterpwdfile scp -r -o 'StrictHostKeyChecking no'
root@$masterip:/usr/lib/hadoop-current /opt/apps/
echo " -copying hive-2.0.1"
sshpass -f $masterpwdfile scp -r root@$masterip:/usr/lib/hive-
current /opt/apps/
echo " -copying spark-2.1.1"
sshpass -f $masterpwdfile scp -r root@$masterip:/usr/lib/spark-
current /opt/apps/
echo "Start to link /usr/lib/\${app}-current to /opt/apps/\${app}"
if [ -L /usr/lib/hadoop-current ]
then
    Unlink/usr/lib/hadoop-Current
fi
ln -s /opt/apps/hadoop-current /usr/lib/hadoop-current
if [ -L /usr/lib/hive-current ]
then
    unlink /usr/lib/hive-current
fi
ln -s /opt/apps/hive-current /usr/lib/hive-current
if [ -L /usr/lib/spark-current ]
then
    unlink /usr/lib/spark-current
fi
Ln-S/opt/apps/spark-current/usr/lib/spark-Current
echo "Start to copy conf from $masterip to local gateway(/etc/emr
)"
sshpass -f $masterpwdfile scp -r root@$masterip:/etc/emr/hadoop-
conf /etc/emr/hadoop-conf
sshpass -f $masterpwdfile scp -r root@$masterip:/etc/emr/hive-conf
/etc/emr/hive-conf
sshpass -f $masterpwdfile scp -r root@$masterip:/etc/emr/spark-
conf /etc/emr/spark-conf
Echo "start to copy environment from $ masterip to local Gateway
(/etc/profile. d )"
sshpass -f $masterpwdfile scp root@$masterip:/etc/profile.d/hadoop
.sh /etc/profile.d/
if [ -L /usr/lib/jvm/java ]
then
    unlink /usr/lib/jvm/java
fi
ln -s /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.131-3.b12.el7_3.x86_64
/jre /usr/lib/jvm/java
echo "Start to copy host info from $masterip to local gateway(/etc
/hosts)"
sshpass -f $masterpwdfile scp root@$masterip:/etc/hosts /etc/
hosts_bak
cat /etc/hosts_bak | grep emr | grep cluster >>/etc/hosts
if ! id hadoop >& /dev/null
then
    useradd hadoop
fi

```

- **Test**

- **Hive**

```
[hadoop@iz23bc05hrvZ ~]$ hive
```

```
hive> show databases;
OK
default
Time taken: 1.124 seconds, Fetched: 1 row(s)
hive> create database school;
OK
Time taken: 0.362 seconds
hive>
```

— Run the Hadoop job

```
[hadoop@iz23bc05hrvZ ~]$ hadoop jar /usr/lib/hadoop-current/share
/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar pi 10 10
Number of Maps = 10
Samples per Map = 10
Wrote input for Map #0
Wrote input for Map #1
Wrote input for Map #2
Wrote input for Map #3
Wrote input for Map #4
Wrote input for Map #5
Wrote input for Map #6
Wrote input for Map #7
Wrote input for Map #8
Wrote input for Map #9
  File Input Format Counters
    Bytes Read=1180
  File Output Format Counters
    Bytes Written=97
Job Finished in 29.798 seconds
Estimated value of Pi is 3.20000000000000000000
```

15 MetaService

MetaService is provided under the E-MapReduce environment. MetaService allows you to access Alibaba Cloud resources in the E-MapReduce cluster without using AK.

Default roles

By default, you must authorize an application role (AliyunEmrEcsDefaultRole) to E-MapReduce when creating a cluster. After authorization, you can perform operations on E-MapReduce to access Alibaba Cloud resources without the need to explicitly input AK. By default, the following permission policies are granted to AliyunEmrEcsDefaultRole:

```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "oss:GetObject",
        "oss:ListObjects",
        "oss:PutObject",
        "oss:DeleteObject",
        "oss:ListBuckets",
        "oss:AbortMultipartUpload"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

By default, your operations based on MetaService can Access OSS Data Only. If you want to access other Alibaba Cloud resources, such as LogService by using MetaService, you must grant permissions to AliyunEmrEcsDefaultRole. You can perform the preceding operations by using the [RAM console](#).



Note:

MetaService only supports AK-free operations on the OSS, LogService, and MNS data. You must edit and delete the default role with caution. Otherwise, your cluster creation or operations may fail.

Custom application roles

In most cases, you only need to use a default role or modify the default role. E-MapReduce also allows you to create your own application role. When creating a cluster, you can use a default role or create your own application role. For more information about how to create and authorize a role to E-MapReduce, see [RAM documentations](#).

Access to MetaService

MetaService is an HTTP service which can be accessed directly to obtain metadata information.

For example, you can obtain the region where the current cluster is located using the `curl http://localhost:10011/cluster-region` command.

MetaService supports the following types of information:

- Region: /cluster-region
- Role name: /cluster-role-name
- AccessKeyId: /role-access-key-id
- AccessKeySecret: /role-access-key-secret
- SecurityToken: /role-security-token
- Network type: /cluster-network-type

Use MetaService

You can use MetaService to access Alibaba Cloud resources without the need to use AK, which can:

- Reduce the risk of an AK leak. The RAM-based usage can minimize security risk. The permissions are minimized by only granting the required permissions to the role.
- Improve user experience. This is intended especially when you interactively access the OSS resources by using MetaService. You do not need to write a long string of OSS path.

Several usage methods are introduced as follows:

```
I. Using the Hadoop command line to display OSS data
   Previously, we used: hadoop fs -ls oss://ZaH*****As1s:Ba23N
   *****sdaBj2@bucket.oss-cn-hangzhou-internal.aliyuncs.com/a/b/
   c
   Now, we use: hadoop fs -ls oss://bucket/a/b/c
II. Using Hive to create a table
   Previously, we used:
   CREATE EXTERNAL TABLE test_table(id INT, name string)
   ROW FORMAT DELIMITED
   FIELDS TERMINATED BY '/t'
   LOCATION 'oss://ZaH*****As1s:Ba23N*****sdaBj2@bucket
   .oss-cn-hangzhou-internal.aliyuncs.com/a/b/c';
   Now, we use:
   CREATE EXTERNAL TABLE test_table(id INT, name string)
   ROW FORMAT DELIMITED
   FIELDS TERMINATED BY '/t'
   LOCATION 'oss://bucket/a/b/c';
III. Spark
   Previously, we used: val data = sc.textFile("oss://ZaH*****As1s:
   Ba23N*****sdaBj2@bucket.oss-cn-hangzhou-internal.aliyuncs.com
   /a/b/c")
   Now, we use: val data = sc.textFile("oss://bucket/a/b/c")
```

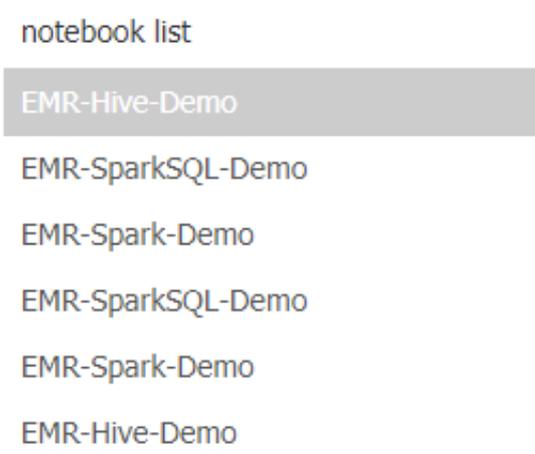

16 Notebook

16.1 Introduction

Notebook allows you to compile and run Spark, Spark SQL, and Hive SQL tasks directly on the E-MapReduce console. You can view the running results directly on Notebook. Notebook is ideal for processing debugging tasks that require shorter runtime and whose data results need to be viewed directly. For tasks with longer runtime and requiring regular execution, the job and execution plan function must be used. This section describes how to create and run a demo task. For other examples and operation instructions, refer to the subsequent chapters.

Create a demo task

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. At the top of the navigation bar, click **Old EMR Scheduling**.
3. In the left-side navigation bar, click **Notebook**.
4. Click **New notebook demo**.



5. A confirmation box is displayed, indicating the required cluster environment. Click **OK** to create demo tasks. Three examples of interactive tasks will be created.



Run a Spark demo task

1. Click **EMR-Spark-Demo** to display the example of a Spark notebook. Before running the notebook, you need to associate the task to a created cluster. Select a created cluster in the list of available clusters. Note that the associated cluster must be EMR-2.3 or later and has no less than three nodes, each with at least 4 cores and 8 GB of memory.



The screenshot shows a notebook interface with a toolbar at the top containing 'File', 'View', 'Run All', 'Type: SPARK', and 'Attached cluster: Not Attached'. Below the toolbar is a code cell with the following Scala code:

```
> %spark
import scala.math.random
import org.apache.spark._

val slices = 20
val n = math.min(100000L * slices, Int.MaxValue).toInt
val count = sc.parallelize(1 until n, slices).map { i =>
  val x = random * 2 - 1
  val y = random * 2 - 1
  if (x*x + y*y < 1) 1 else 0}.reduce(_ + _)
println("Pi is roughly ~ + 4.0 * count / n)
```

Below the code cell is a 'run' button. Underneath the button, the text 'Run results :' is followed by 'status : READY ,'.

2. After a cluster is associated, click **Run**. When the associated cluster executes the Spark or Spark SQL notebook for the first time, it takes about one minute to build the Spark context and running environment. It does not need to be built in subsequent executions. The running result is displayed below the Run button.

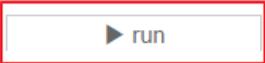
Save Paragraph

Hide results

delete

```
> %spark
import scala.math.random
import org.apache.spark._

val slices = 20
val n = math.min(100000L * slices, Int.MaxValue).toInt
val count = sc.parallelize(1 until n, slices).map { i =>
  val x = random * 2 - 1
  val y = random * 2 - 1
  if (x*x + y*y < 1) 1 else 0}.reduce(_ + _)
println("Pi is roughly " + 4.0 * count / n)
```

run

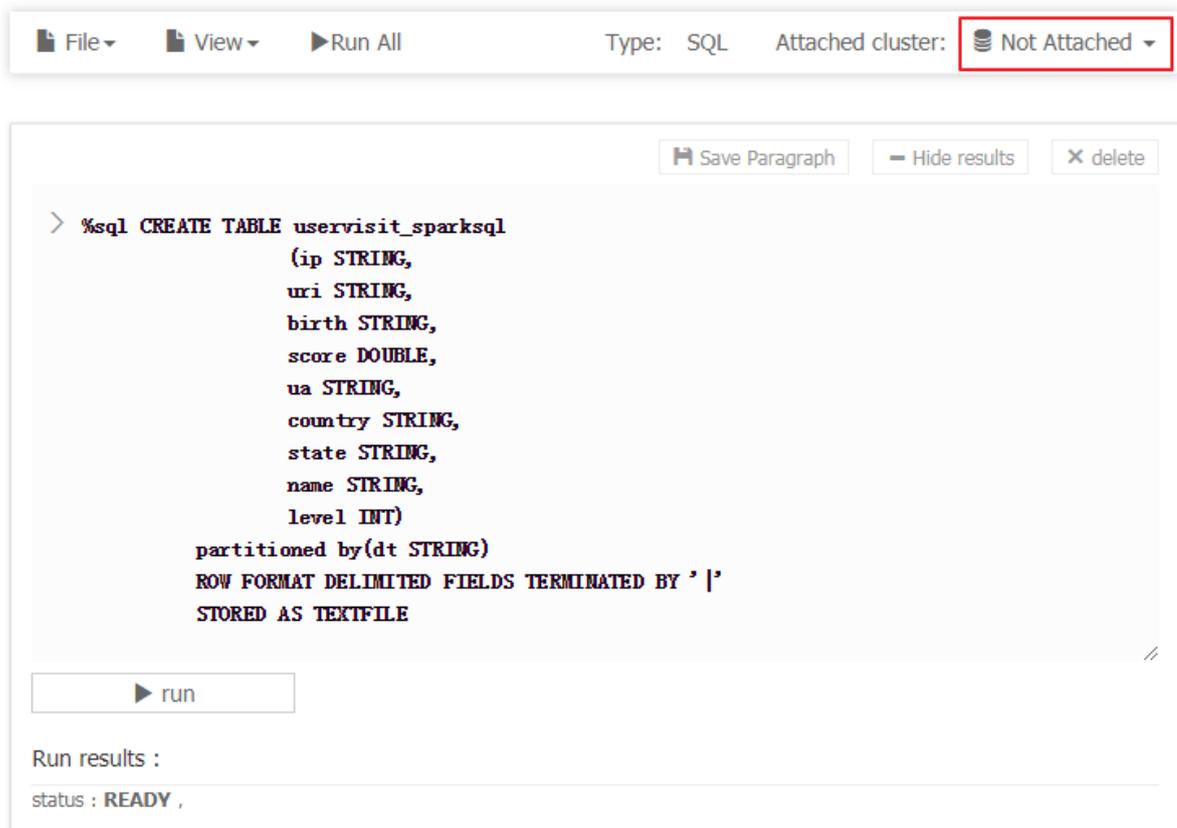
Run results :

```
import scala.math.random
import org.apache.spark._
slices: Int = 20
n: Int = 2000000
count: Int = 1570075
Pi is roughly 3.14015
```

status : FINISHED , run 0second(s) , finish Time : Nov 9, 2018 11:25:52 AM

Run a SparkSQL demo task

1. Click **EMR-Spark-Demo** to display the SparkSQL notebook example. Before running the notebook, you need to associate the notebook to a created cluster. Click the upper right corner and select a created cluster in the list of available clusters.



The screenshot shows a notebook interface with a menu bar at the top containing 'File', 'View', and 'Run All'. To the right, it displays 'Type: SQL' and 'Attached cluster: Not Attached'. The main area contains a code editor with the following SparkSQL query:

```
> %sql CREATE TABLE uservisit_sparksql
      (ip STRING,
       uri STRING,
       birth STRING,
       score DOUBLE,
       ua STRING,
       country STRING,
       state STRING,
       name STRING,
       level INT)
      partitioned by(dt STRING)
      ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'
      STORED AS TEXTFILE
```

Below the code editor is a 'run' button. Underneath, the 'Run results' section shows 'status : READY ,'. At the top right of the code editor, there are buttons for 'Save Paragraph', 'Hide results', and 'delete'.

2. The SparkSQL demo contains several demo sections which can be run individually or all together by clicking **Run All**. After the running, you can see all returned data results of each section.

**Note:**

If the section for creating a table is run multiple times, an error will be reported indicating that the table already exists.

Save Paragraph Hide results delete

```
> %sql
  get data
  select * from uservisit_sparksql limit 10
```

run

Run results :

| ip | uri | birth | score | ua | country | state | name | level | dt |
|---------------|--|-----------|---------------|--------------------------|---------|--------|---------------------|-------|------------|
| 170.131.22.2 | 13rdgckzlcblruc.html | 1984-8-7 | 336.869186722 | NuSearch Spider | HUN | HUN-NL | remnants | 3 | 2016-01-01 |
| 162.114.4.2 | 6xpirezjeytxdjsmwtnyeugkesratmpvamliekrijlgmvyysrlqwgw.html | 1978-1-9 | 331.791153595 | Superdown loads Spiderma | AUT | AUT-ZR | MHD | 8 | 2016-01-01 |
| 177.110.45.18 | 11zvmoamsyaameokoeylkbkivgquksibqbalnpmailbiyfxitbhfdroyxesixbjndkyqz.html | 1986-9-25 | 411.968497603 | Mozilla/4.0 | FLK | FLK-GB | apj@as.arizona.edu. | 7 | 2016-01-01 |
| 157.111.12.37 | 44mvdnls.html | 2002-7-3 | 486.660926201 | PHP/4.0. | FIN | FIN-CZ | diffuse | 3 | 2016-01-01 |

status : FINISHED , run 0second(s) , finish Time : Nov 9, 2018 11:18:38 AM

Run a Hive demo task

1. Click **EMR-Hive-Demo** to display the Hive notebook example. Before running the notebook, you need to associate the notebook to a created cluster. Click the upper right corner and select a created cluster in the list of available clusters.
2. The Hive demo task contains several demo sections, which can be run individually, or all together by clicking **Run All**. After running, you can see all returned data results of each section.



Note:

- When the associated cluster executes the Hive notebook for the first time, it takes a few seconds to build the Hive client running environment. It will no longer need to be built in subsequent execution.

- If the section for creating a table is run multiple times, an error will be reported indicating that the table already exists.

Save Paragraph Hide results delete

```
> %hive
  -- get data
  select * from uservisit_hive limit 10
```

run

Run results :

| uservisit_hive.ip | uservisit_hive.uri | uservisit_hive.birth | uservisit_hive.score | uservisit_hive.ua | uservisit_hive.country | uservisit_hive.state | uservisit_hive.name | uservisit_hive.level | uservisit_hive.dt |
|-------------------|---|----------------------|----------------------|-------------------------|------------------------|----------------------|---------------------|----------------------|-------------------|
| 170.131.22.2 | 13rdgckzlcblruc.html | 1984-8-7 | 336.869186722 | NuSearch Spider | HUN | HUN-NL | remnants | 3 | 2016-01-01 |
| 162.114.4.2 | 6xpizrzejytxdjsmwtmyeugkesratmpvamlietrijlgmvyyrsiqwqw.html | 1978-1-9 | 331.791153595 | Superdownloads Spiderma | AUT | AUT-ZR | MHD | 8 | 2016-01-01 |
| 177.110.45.18 | 11zvmoamsyaameokoeylkbivgquksibqbalnpmailbiyfxitbhfdroyxesixbjndkyqz.html | 1986-9-25 | 411.968497603 | Mozilla/4.0 | FLK | FLK-GB | apj@as.arizona.edu. | 7 | 2016-01-01 |

status : FINISHED , run 0second(s) , finish Time : Nov 9, 2018 11:21:08 AM

Cancel the association with clusters

After a notebook is run in a cluster, the cluster creates a process for catching some context running environments in order to ensure quick response upon re-execution. If you do not need to execute other notebooks, and you want to release the cluster resources occupied by caching, you can disassociate all interactive tasks that have been run from the associated clusters. In this way, you can release the memory resources occupied on the original associated clusters.

File View Run All Type: HIVE Attached cluster: test1

Cancel Associate

Save Paragraph Hide results delete

```
> %hive CREATE TABLE uservisit_hive
      (ip STRING,
       uri STRING,
       birth STRING,
       score DOUBLE,
       ua STRING,
       country STRING,
       state STRING,
       name STRING,
       level INT)
      partitioned by(dt STRING)
      ROW FORMAT DELIMITED FIELDS TERMINATED BY '|'
      STORED AS TEXTFILE
```

run

Run results :
status : READY ,

16.2 Manual

This article shows you how to create a new notebook task on the EMR console and guide you through the task creation and operation.

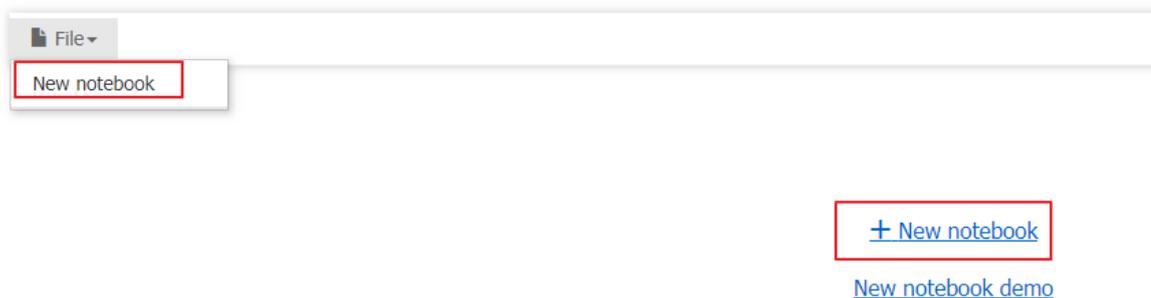
Create a new notebook task



Note:

The cluster on which an interactive task is run must be EMR-2.3 or later, and has no less than three nodes, each with at least 4 cores and 8 GB of memory.

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. At the top of the navigation bar, click **Old EMR Scheduling**.
3. In the left-side navigation bar, click **Notebook**.
4. Click **New notebook** or **File > -New notebook**.



5. Input the name and select the default type. The associated cluster is optional. Click **OK** to create a notebook.

Notebook ✕

* Name :

Length: 1 to 64 characters. Only Chinese characters, English letters, numbers '-', and '_' are allowed

* defaultType : Spark Spark SQL Hive

when you run notebook without specify any type , notebook will use this as default type

Attached cluster :

Three types are supported for a notebook task. Spark can be used to write scala spark codes. Spark SQL can be used to write SQL statements supported by Spark. Hive can be used to write SQL statements supported by Hive.

6. An associated cluster must be a created cluster of EMR-2.3 or later, and has no less than three nodes, each with at least 4 cores and 8 GB of memory. You can also associate the cluster before running the task.

Up to 20 interactive tasks can be created in one account.

Enter and save a section

A paragraph is the smallest unit for running a notebook. For a notebook, you can fill in multiple paragraphs. Each paragraph can start with `%spark`, `%sql`, and `%hive` indicating that this paragraph is a scala spark code paragraph, spark SQL paragraph, or Hive SQL paragraph. The type prefix is segregated by a blank space or by line feed and actual content. If the type prefix is not specified, the default type of the interactive task will be used as the run type of this paragraph.

The following is an example showing how to create a temporary Spark table:

Paste the following code in the section and a red * symbol is displayed, indicating that this notebook has been changed. You can click the **Save Paragraph** button or **run** button to save the modifications to the paragraph. Click **+** below the paragraph to create a new paragraph. Up to 30 paragraphs can be created in one notebook.

```
%spark
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
// load bank data
val bankText = sc.parallelize(
  IOUtils.toString(
    new URL("http://emr-sample-projects.oss-cn-hangzhou.aliyuncs.
com/bank.csv"),
    Charset.forName("utf8")).split("\n"))
case class Bank(age: Integer, job: String, marital: String, education
: String, balance: Integer)
val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "\"age
\").map(
  s => Bank(s(0).toInt,
    s(1).replaceAll("\"", ""),
    s(2).replaceAll("\"", ""),
    s(3).replaceAll("\"", ""),
    s(5).replaceAll("\"", "").toInt
  )
).toDF()
```

```
bank.registerTempTable("bank")
```



The screenshot shows a notebook interface with a code cell. The code is as follows:

```
> %spark import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
// load bank data
val bankText = sc.parallelize(
  IOUtils.toString(
    new URL("http://emr-sample-projects.oss-cn-hangzhou.aliyuncs.com/bank.csv"),
    Charset.forName("utf8")).split("\n")
)
case class Bank(age: Integer, job: String, marital: String, education: String, balance:
Integer) val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "\age\").map(
s => Bank(s(0).toInt,
s(1).replaceAll("\\", ""),
s(2).replaceAll("\\", ""),
s(3).replaceAll("\\", ""),
s(5).replaceAll("\\", "").toInt ) ).toDF()
bank.registerTempTable("bank")
```

Below the code is a "run" button. The "Run results" section shows "status : READY ,". At the bottom of the cell, there is a plus sign icon.

Run a paragraph

Before running a notebook, you must associate it to a created cluster. If a created notebook is not associated with a cluster, **Not Attached** is displayed in the upper right corner of the page. You can click to select a cluster in the list of available clusters. Note that the associated cluster must be EMR-2.3 or later, and has no less than three nodes, each with at least 4 cores and 8 GB of memory.

File View Run AllType: SPARK Attached cluster: Not Attached

* Save ParagraphHide resultsdelete

```
> %spark import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
// load bank data
val bankText = sc.parallelize(
  IOUtils.toString(
    new URL("http://emr-sample-projects.oss-cn-hangzhou.aliyuncs.com/bank.csv"),
    Charset.forName("utf8")).split("\n"))
case class Bank(age: Integer, job: String, marital: String, education: String, balance:
Integer) val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "\age").map(
s => Bank(s(0).toInt,
s(1).replaceAll("\\", ""),
s(2).replaceAll("\\", ""),
s(3).replaceAll("\\", ""),
s(5).replaceAll("\\", "").toInt ) ).toDF()
bank.registerTempTable("bank")
```

run

Run results :
status : READY ,

Click the **Run** button to automatically save the current paragraph and run the content. If this paragraph is the last one, a new paragraph is automatically created.

PENDING means the paragraph has not run yet and **RUNNING** means the paragraph is running. **FINISHED** means the running process has finished. **ERROR** means an error has occurred. The running result is displayed below the run button of the paragraph. **FINISHED** means the running process has been finished. **ERROR** means an error occurs. The running result is displayed below the run button of the section. During running, you can click "Cancel" below the "Run" button to cancel running. **ABORT** is displayed after running is canceled.

Save Paragraph

Hide results

delete

```

> %spark
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset // load bank data
val bankText = sc.parallelize(
  IOUtils.toString(
    new URL("http://emr-sample-projects.oss-cn-hangzhou.aliyuncs.com/bank.csv"),
    Charset.forName("utf8")).split("\n")
)
case class Bank(age: Integer, job: String, marital: String, education: String, balance: Integer)
val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "\ age\").map(
  s => Bank(s(0).toInt,
    s(1).replaceAll("\\", ""),
    s(2).replaceAll("\\", ""),
    s(3).replaceAll("\\", ""),
    s(5).replaceAll("\\", "").toInt
  )
).toDF()
bank.registerTempTable("bank")

```

run

Run results :

```

import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
bankText: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[29] at parallelize at <console>:36
>:36
defined class Bank
bank: org.apache.spark.sql.DataFrame = [age: int, job: string, marital: string, education: string, balance: int]

```

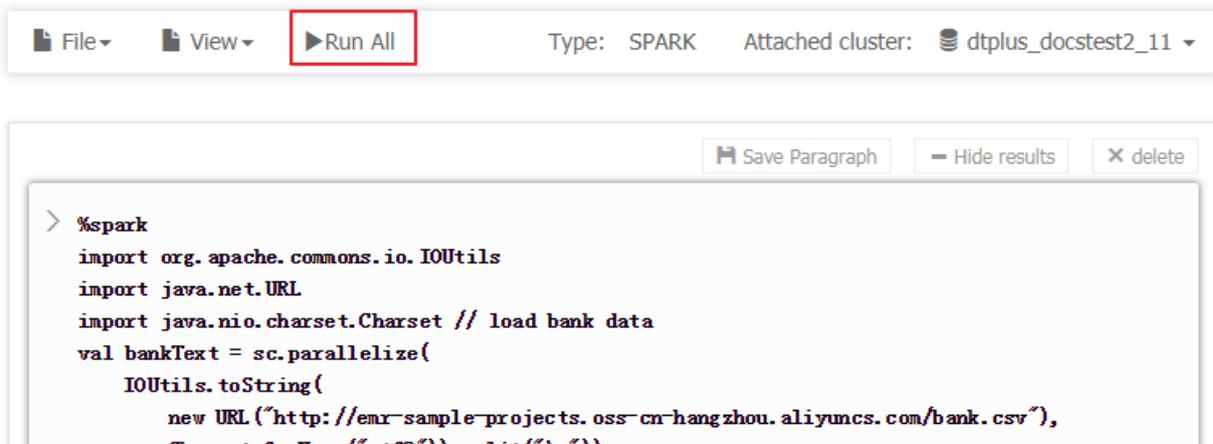
status : FINISHED , run 0second(s) , finish Time : Nov 9, 2018 11:43:32 AM

The paragraph can be run multiple times and only the result of the last running is retained. You cannot modify the entered content of the paragraph during running. The content can be modified only after running of the paragraph is finished.

Run all

For a notebook, you can click **Run All** on the menu bar to run all paragraphs. The paragraphs are submitted sequentially for running. Different types have independent execution queues. If a notebook contains multiple paragraph types, the order for executing the paragraphs on the cluster is based on type after these paragraphs are submitted sequentially. Spark and Spark SQL support one-by-one execution. Hive supports concurrent execution and the maximum number of concurrently executed interactive paragraphs on the same cluster is 10. Note that all concurrently

executed paragraphs are restricted by cluster resources. If the cluster size is small and many paragraphs need to be executed concurrently, the paragraphs still need to queue on the Yarn.



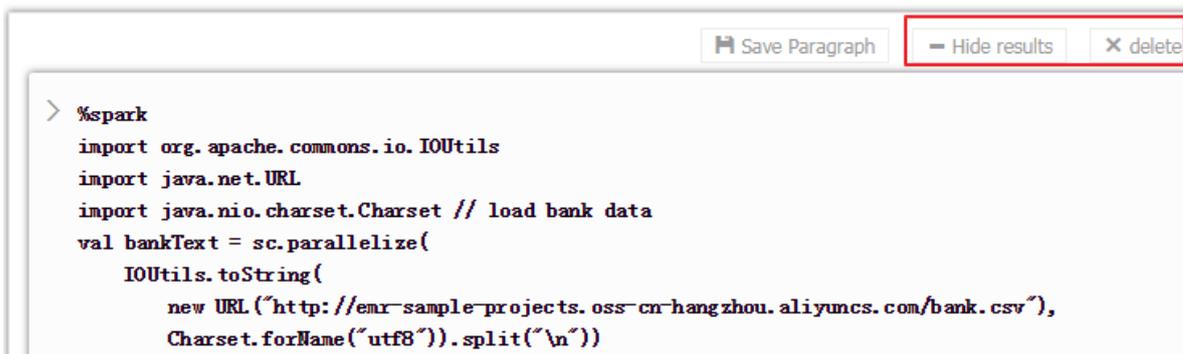
Cancel the association with clusters

After a notebook is run in a cluster, the cluster creates a process for catching of some context running environments to ensure quick response upon re-run. If you do not need to run other notebooks and you want to release the cluster resources occupied by caching, you can disassociate all notebooks that have been run from the associated clusters. In this way, you can release the memory resources occupied on the original associated clusters.



Other operations

- Paragraph operations



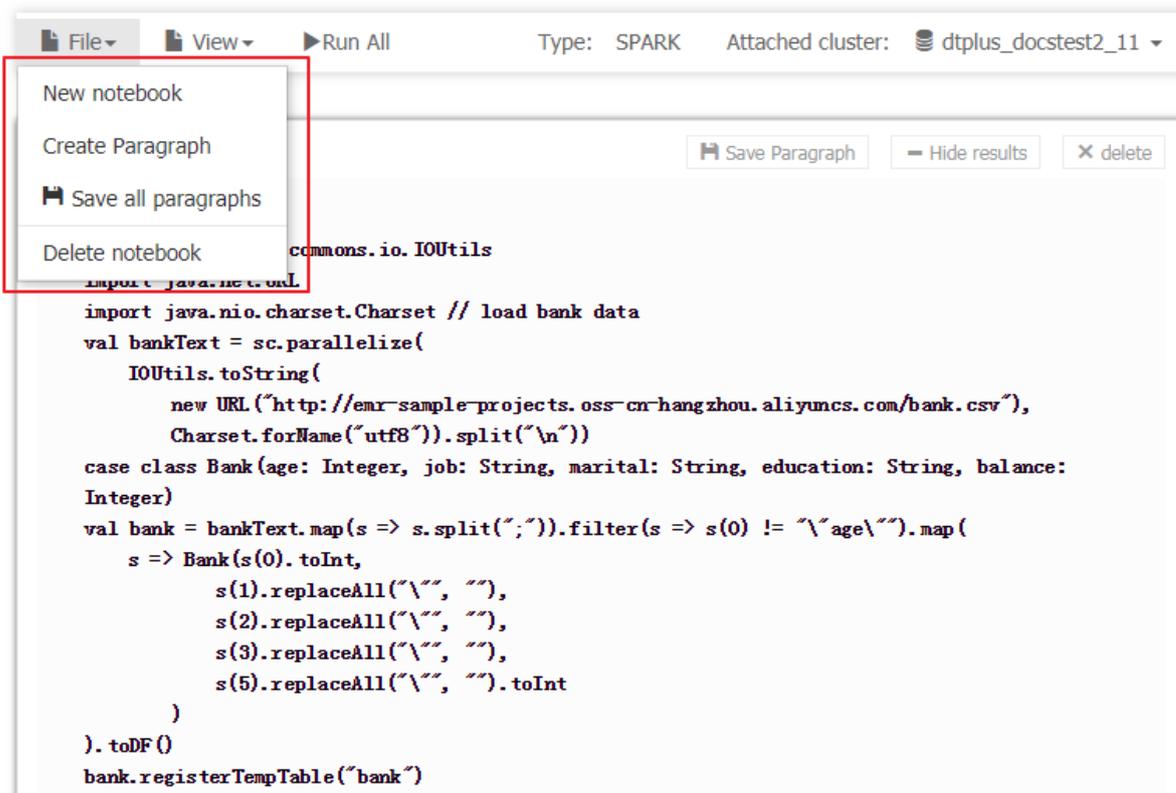
— Hide and display the results

You can hide the paragraph results and display the entered content of the paragraph only.

— Delete a paragraph

Delete the current paragraph. The paragraph that is running can also be deleted.

- File menu



— New notebook

Create a notebook and switch to the created notebook interface.

— Create Paragraph

Add a new paragraph to the end of the notebook. A notebook can only have up to 30 paragraphs.

— Save all paragraphs

Save all modified paragraphs.

— Delete notebook

Delete the current notebook. If the cluster has been associated, it will be disassociated.

- **View**

Display codes only or display codes and results

Only the entered codes for all paragraphs are displayed or both the codes and results are displayed.

16.3 Examples

16.3.1 Example of bank employee information inquiry

Section one: Create a temporary table

```
%spark
import org.apache.commons.io.IOUtils
import java.net.URL
import java.nio.charset.Charset
// Zeppelin creates and injects sc (SparkContext) and sqlContext (
HiveContext or SqlContext)
// So you don't need create them manually
// load bank data
val bankText = sc.parallelize(
  IOUtils.toString(
    new URL("http://emr-sample-projects.oss-cn-hangzhou.aliyuncs.
com/bank.csv"),
    Charset.forName("utf8")).split("\n"))
case class Bank(age: Integer, job: String, marital: String, education
: String, balance: Integer)
val bank = bankText.map(s => s.split(";")).filter(s => s(0) != "\"age
\").map(
  s => Bank(s(0).toInt,
    s(1).replaceAll("\"", ""),
    s(2).replaceAll("\"", ""),
    s(3).replaceAll("\"", ""),
    s(5).replaceAll("\"", ").toInt
  )
).toDF()
bank.registerTempTable("bank")
```

Section two: Query the table structure

```
%sql
```

```
desc bank
```

Section three: Query the number of employees of each age group below 30

```
%sql select age, count(1) value from bank where age < 30 group by age
order by age
```

Section 4: Query the information of employees at the age less than or equal to 20

```
%sql select * from bank where age <= 20
```

16.3.2 Video playback data example

Data preparations

In this example, you need to download data from OSS and upload it to your OSS bucket. The data includes:

- [User Table Sample Data](#)
- [Video Table Sample Data](#)
- [Playvideo Table Sample Data](#)

Upload the sample data of User Table, Video Table, and Playvideo Table to the specified UserInfo, Videoinfo, and Playvideo, respectively, on your OSS bucket. For example, upload the data to Demo or UserInfo directory under Bucket Example.

In the following created table, replace the SQL [bucketname] with your bucket name, replace [region] with your OSS region name, and replace [bucketpath] with your specified path prefix of the OSS, for example, "Demo".

Section one: Create a user table

```
%hive
CREATE EXTERNAL TABLE user_info(id int,sex int,age int, marital_status int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 'oss://[bucketname].oss-cn-[region]-internal.aliyuncs.com/[bucketpath]/userinfo'
```

Section two: Create a video table

```
%hive
CREATE EXTERNAL TABLE video_info(id int,title string,type string) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 'oss://[bucketname].oss-cn-[region]-internal.aliyuncs.com/[bucketpath]/videoinfo'
```

Section three: Create a video play table

```
%hive
```

```
CREATE EXTERNAL TABLE play_video(user_id int,video_id int, play_time
  bigint) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' LOCATION 'oss
  ://[bucketname].oss-cn-[region]-internal.aliyuncs.com/[bucketpath]/
  playvideo'
```

Section four: User table count

```
%sql select count(*) from user_info
```

Section five: Video table count

```
%sql select count(*) from video_info
```

Section six: Video play table count

```
%sql select count(*) from play_video
```

Section seven: Video play count of each video type

```
%sql select video.type, count(video.type) as count from play_video
  play join video_info video on (play.video_id = video.id) group by
  video.type order by count desc
```

Section eight: Video information of video play top 10

```
%sql select video.id, video.title, video.type, video_count.count from
  (select video_id, count(video_id) as count from play_video group by
  video_id order by count desc limit 10) video_count join video_info
  video on (video_count.video_id = video.id) order by count desc
```

Section nine: Age groups of audience watching the video with largest video play count

```
%sql select age , count(*) as count from (select distinct(user_id)
  from play_video where video_id =49 ) play join user_info userinfo on
  (play.user_id = userinfo.id) group by userinfo.age
```

Section ten: Gender, age group, and marital status of audience watching the video with largest video play count

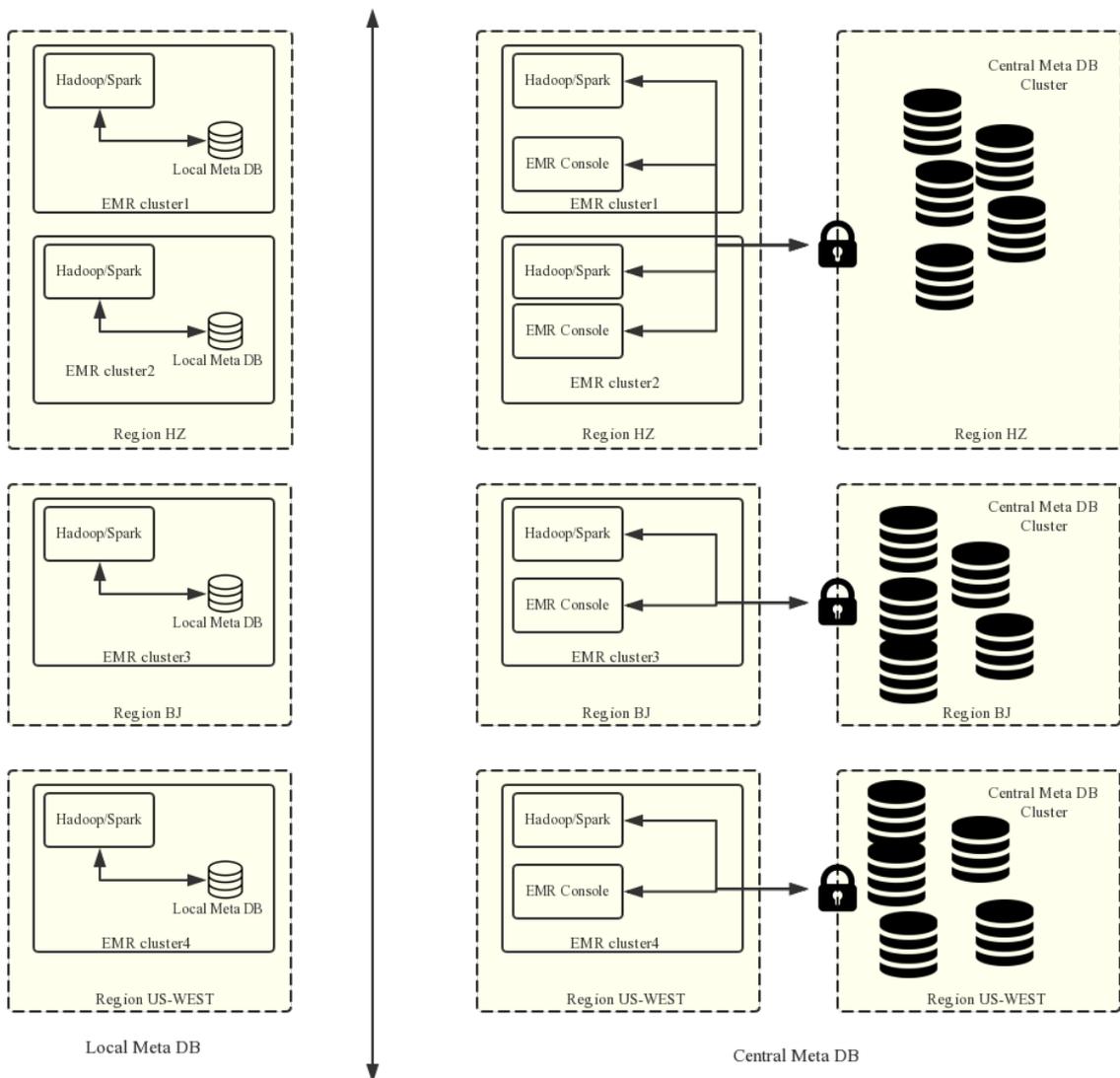
```
%sql select if(sex=0,'Female','Male') as title, count(*) as count,
  'Gender' as type from (select distinct(user_id) from play_video
  where video_id =49 ) play join user_info userinfo on (play.user_id =
  userinfo.id) group by userinfo.sex
  union all
  select case when userinfo.age<15 then 'Less than 15' when age<25 then
  '15-25' when age<35 then '25-35' else 'More than 35' end , count
  (*) as count, 'Age Group' as type from (select distinct(user_id) from
  play_video where video_id =49) play join user_info userinfo on (play
  .user_id = userinfo.id) group by case when userinfo.age<15 then 'Less
  than 15' when age<25 then '15-25' when age<35 then '25-35' else 'More
  than 35' end
  union all
  select if(marital_status=0,'Unmarried','Married') as title, count(*)
  as count, 'Marital Status' as type from (select distinct(user_id) from
```

```
play_video where video_id =49 ) play join user_info userinfo on (  
play.user_id = userinfo.id) group by marital_status
```

17 Table mangement

E-MapReduce 2.4.0 and later versions support central metadata management. In the versions earlier than E-MapReduce 2.4.0, all clusters use the local MySQL database as the Hive metadatabase. In E-MapReduce 2.4.0 and later versions, E-MapReduce supports the central highly-reliable Hive metadatabase.

Introduction



You can enable the central metadatabase function when creating a cluster to use the external metadatabase.



Note:

- The current metadatabase needs to be connected using the public IP address. Therefore, the cluster must have a public IP address. Do not change the public IP address. Otherwise, the corresponding database whitelist is invalid.
- The table management function can be used only when the central metadatabase function is enabled when a cluster is created. A local metadatabase does not support table management currently. In that case, you may use the Hue tool in the cluster for table management.

The central metadata management function can:

1. Provide long-term metadata storage.

When metadata is stored in the local MySQL database of the cluster, metadata is lost when the cluster is released. Especially when E-MapReduce supports the flexible creation mode, clusters can be created and released anytime upon requirements. To retain the metadata, you must log on to the cluster and manually export the metadata. This issue can be resolved with the central metadata management function.

2. Separate computing and storage.

E-MapReduce supports storing data in Alibaba Cloud OSS, which reduces the usage cost especially when the data volume is large. Meanwhile, E-MapReduce clusters are mainly used as computing resources and can be released anytime after use. Since data is stored in OSS, the metadata migration issue does not exist.

3. Implement data sharing.

With the central metadatabase, if all data is stored in OSS, all clusters can directly access data without migrating or restructuring metadata. This enables E-MapReduce clusters to provide different services while still ensuring direct data sharing.



Note:

Before central metadata management is supported, metadata is stored in the local MySQL database of each cluster and is lost when the cluster is released. With central metadata management, releasing clusters does not clean up metadata. Before you delete the data in OSS or in the HDFS of a cluster or you release a cluster, make sure that the corresponding metadata is already deleted. That means the tables and database that store the data have been dropped. This prevents dirty metadata in the database.

Table management operations

Before E-MapReduce clusters support metadata management, you have to log on to the internal environment of a cluster to check, add, or delete tables in the cluster. If more than one clusters exist, you have to log on to the clusters one by one, which is inconvenient. With the central metadata management function, E-MapReduce enables table management on the console. This includes checking the list of databases and tables, checking table details, creating and deleting databases and tables, and previewing data.

- Database and table list
- Table details
- Data preview
- Create a database
- Create a table

Two methods to create a table: Manual creation and Creation from a file

- Manual creation, When no service data exists, you can manually input the table structure to create an empty table;
- Creation from a file, When service data already exists, you can use the service data as a table directly by parsing the table interface from the file. Make sure that the separators used for creating a table must correspond to those used in the data file to have a proper table structure.

The separators can be common characters such as commas and spaces, or special characters TAB, ^A, ^B, and ^C.



Note:

1. Databases and tables can be created and deleted only in E-MapReduce clusters.
2. The HDFS is the internal file system of each cluster and does not support cross-cluster communication without special network settings. Therefore, the E-MapReduce table management function only supports creating databases and tables based on the OSS file system.
3. The location of a database or table must be in a directory under the OSS bucket, rather than the OSS bucket.

FAQs

1. Wrong FS: `oss://yourbucket/xxx/xxx/xxx`

This error occurs when the table data on OSS is deleted while the metadata of the table is not. The table schema persists, while the actual data does not exist or is moved to another location. In this case, you can change the table location to an existing path and delete the table again.

```
alter table test set location 'oss://your_bucket/your_folder'
```

This can be completed on the E-MapReduce interactive console.



Note:

`oss://your_bucket/your_folder` must be an existing OSS path.

2. Wrong FS: hdfs://yourhost:9000/xxx/xxx/xxx

This error occurs when the table data in the HDFS is deleted while the table schema persists.

The error can be removed by using the preceding solution.

3. The message “java.lang.IllegalArgumentException: java.net.UnknownHostException: xxxxxxxx” is displayed when the Hive database is deleted.

This error occurs because the Hive database is created in the HDFS of a cluster and it is not cleaned up when the cluster is released. As a result, its data in the HDFS of the released cluster cannot be accessed after a new cluster is created. Therefore, when releasing a cluster, remember to clean up the databases and tables that are manually created in the HDFS of the cluster.

Solution

Log on to the master node of the cluster using the command line, and find the address, username, and password for accessing the Hive metadatabase in `$HIVE_CONF_DIR/hive-site.xml`.

```
javax.jdo.option.ConnectionUserName //Username for accessing the
database;
javax.jdo.option.ConnectionPassword //Password for accessing the
database;
javax.jdo.option.ConnectionURL //Address and name of the database ;
```

```
<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>${ConnectionUserName}</value>
  <description>Username to use against metastore database</description>
</property>
<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>${ConnectionPassword}</value>
  <description>password to use against metastore database</description>
</property>
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://${DBConnectionURL}/${DBName}?createDatabaseIfNotExist=true&amp;characterEncoding=UTF-8</value>
  <description>JDBC connect string for a JDBC metastore</description>
</property>
```

Log on to the Hive metadatabase on the master node of the cluster:

```
mysql -h ${DBConnectionURL} -u ${ConnectionUserName} -p [Press Enter
]
[Enter the password]${ConnectionPassword}
```

After logging on to the Hive metadatabase, change its location to an existing OSS path in the region:

```
mysql> show databases;
+-----+
| Database
+-----+
| information_schema
| xxxxxxxxx77ac43c3bd0efae77e0bf1947d45fb4c896fb99
+-----+

mysql> use xxxxxxxxx77ac43c3bd0efae77e0bf1947d45fb4c896fb99;

mysql> select * from dbs;
+-----+-----+-----+-----+-----+-----+
| DB_ID | DESC          | DB_LOCATION_URI          | NAME   | OWNER_NAME | OWNER_TYPE |
+-----+-----+-----+-----+-----+-----+
| 1     | Default Hive database | oss://mybucket/hive/warehouse | default | public     | ROLE
| 6     | NULL          | hdfs://dirty-hostname/warehouse | dirty_db | NULL      | USER
+-----+-----+-----+-----+-----+-----+

mysql> update dbs set DB_LOCATION_URI = 'oss://your-bucket/your-db-folder' where DB_ID = 6;
```

18 Kerberos authentication

18.1 Introduction to Kerberos

E-MapReduce supports creating secure clusters from EMR-2.7.x and EMR-3.5.x where open source components in the cluster are started in the Kerberos security mode. In this mode, only authenticated clients can access the cluster service such as HDFS.

Prerequisites

Kerberos components supported by the current 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:**

Remarks: Currently Kafka, Presto, and Storm do not support Kerberos.

Create a save cluster

You can turn on **High Security Mode** switch on the software configuration tab of the cluster creation page as shown in the following figure:

Create Cluster

Software Configuration Hardware Configuration Basic

Version Configuration

EMR Version:

Cluster Type: Hadoop Druid Data Science

Required Services:

- Knox (0.13.0)
- ApacheDS (2.0.0)
- Zeppelin (0.8.0)
- Tez (0.9.1)
- Sqoop (1.4.7)
- Pig (0.14.0)
- Sp... (0.14.0)
- YARN (2.7.2)
- HDFS (2.7.2)
- Ganglia (3.7.2)

Optional Services:

- Superset (0.27.0)
- Ranger (1.0.0)
- Flink (1.4.0)
- Phoenix (4.10.0)
- HBase (1.1.1)
- ZooKeeper (3.4.9)
- Presto (0.208)
- Impala (2.10.0)

Click to Choose

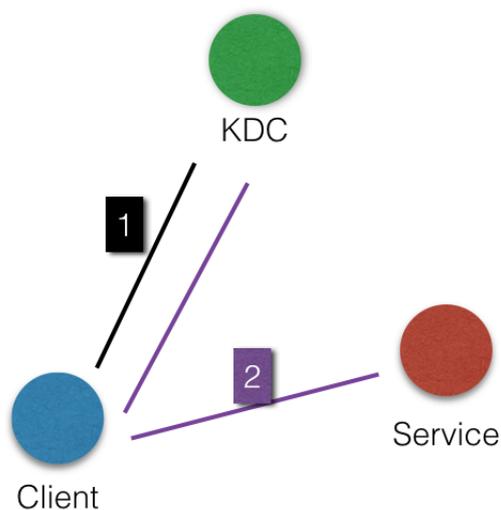
High Security Mode: 

Enable Custom Setting: 

Kerberos identity authentication principle

Kerberos is an identity authentication protocol based on the symmetric key technology. As an independent third-party identity authentication service, Kerberos can provide its ID authentication function for other services, and it supports SSO (the client can access multiple services, such as HBase and HDFS, after ID authentication).

The Kerberos protocol process is mainly divided into two stages where KDC authenticates the Client identity in the first stage, and the Service authenticates the Client identity in the second stage.



- KDC

Kerberos server

- Client

If a user (principal) needs to access the service, KDC and Service authenticates the principal's identity.

- Service

Services that have integrated with Kerberos include HDFS, YARN, and HBase.

- KDC ID authentication

Before a client user (principal) can access a service integrated with Kerberos, it must first pass the KDC ID authentication.

After passing the KDC ID authentication, the client receives a TGT (Ticket Granting Ticket), which can be used to access a service that has integrated Kerberos.

- Service ID authentication

When a principal receives the TGT in step 2.1, it can access the Service. It uses the TGT and the name of the service that it must access (such as HDFS) to obtain an SGT (Service Granting Ticket) from KDC, and use the SGT to access Service, which uses the relevant information to conduct ID authentication on the client. After passing the ID authentication, the client can normally access the Service.

EMR practice

Services in the EMR Kerberos security cluster starts in the Kerberos security mode when creating a cluster.

- The Kerberos server is HasServer
 - Log on to the [EMR Console](#), choose **Cluster** > > **Configuration Management** > **HAS** , and conduct operations including View, Modify configuration, and Restart.
 - Non-HA clusters are deployed on emr-header-1, while 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 to be used by HasServer through configuring the related 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 Service, such as 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/has/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 and 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/has/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 and HBase). For more information, click [here](#).

— Execution plan authentication

If you have jobs submitted through the execution plan of the EMR 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 cluster administrator can also log on to the master node to access the cluster service. The administrator can use the has account (the default logon method is the MIT-Kerberos-compatible method) to log on to the master node and access the cluster service, which is convenient to conduct some troubleshooting or O&M tasks.

```
>sudo su has
>hadoop fs -ls /
```



Note:

Other accounts can also be used to log on the master node, provided that such accounts have already passed Kerberos authentication. In addition, if you must 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/has/hadoop-conf/
```

18.2 Authentication method compatible with MIT Kerberos

This article will introduce the MIT Kerberos authentication process through the HDFS service.

Authentication method compatible with MIT Kerberos identity

The Kerberos server in the EMR cluster is started at the master node, and some management operations must be performed with the root account of the master node (emr-header-1).

The user test's access to the HDFS service is used as an example to introduce the relevant procedure as follows:

- Executed `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 cluster 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/ecm/has-conf
-k /etc/ecm/has-conf/admin.keytab
HadminLocalTool.local: #Press Enter to view the use of the
commands
HadminLocalTool.local: addprinc #Input the command and press
Enter to view the use of the specific command
```

```
HadminLocalTool.local: addprinc -pw 123456 test #Add
principal for the user test, and set the password to 123456
```

#### — Export the keytab file

Admin tool with Kerberos can be used to export the keytab file corresponding to the principal

```
HadminLocalTool.local: ktadd -k /root/test.keytab test #Export the
keytab file, which can be used subsequently
```

#### — Use kinit to obtain the Ticket

On the client machine where HDFS commands are executed, such as the gateway

- Add Linux account `test`

```
useradd test
```
- Install MITKerberos client tools

MITKerberos tools can be used for relevant operations (such as kinit and klist). For detailed usage, see MITKerberos document

```
yum install krb5-libs krb5-workstation -y
```

- Switch to account `test` 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:

Practices of MITKerberos tools

```

[test@iZbp13nu0s9j404h9h15b9Z ~]$ kinit
Password for test@EMR.500141285.COM:
[test@iZbp13nu0s9j404h9h15b9Z ~]$ klist
Ticket cache: FILE:/tmp/krb5cc_1002
Default principal: test@EMR.500141285.COM

Valid starting Expires Service principal
11/16/2017 17:47:14 11/17/2017 17:47:14 krbtgt/EMR.500141285.COM@EMR.500141285.COM
renew until 11/17/2017 17:47:14
[test@iZbp13nu0s9j404h9h15b9Z ~]$ kinit -l 5d
Password for test@EMR.500141285.COM:
[test@iZbp13nu0s9j404h9h15b9Z ~]$ klist
Ticket cache: FILE:/tmp/krb5cc_1002
Default principal: test@EMR.500141285.COM

Valid starting Expires Service principal
11/16/2017 17:47:22 11/21/2017 17:47:22 krbtgt/EMR.500141285.COM@EMR.500141285.COM
renew until 11/18/2017 17:47:22
[test@iZbp13nu0s9j404h9h15b9Z ~]$ kdestroy
[test@iZbp13nu0s9j404h9h15b9Z ~]$ klist
klist: No credentials cache found (filename: /tmp/krb5cc_1002)

```

#### — Execute HDFS commands

When a ticket is obtained, HDFS commands can be normally executed.

```

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:

Corresponding Linux accounts must be added to all the nodes in the cluster in advance for running yarn job (for more information, see the following [Add test account to the EMR cluster])

- Use Java code to access HDFS

#### — Use local ticket cache



#### Note:

You need to execute kinit in advance to obtain the ticket, and the application will not be normally accessed when the ticket expires.

```

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());
}
}

```

### — Use keytab file (recommended)



#### Note:

keytab has long-term validity, and is independent 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>

```

```
</dependencies>
```

## 18.3 Ram certification

The Kerberos server in the EMR cluster supports not only the authentication method compatible with MIT Kerberos, but also the identity authentication by using RAM as the identity information.

### RAM ID authentication

**RAM** product supports creating/managing subaccounts and using subaccounts to implement access control for various resources on the cloud.

Administrator of the master account can create a subaccount on the RAM user management page (subaccount name must comply with Linux username specifications) and download the subaccount AccessKey for the corresponding developer. The developer can then configure the AccessKey to pass Kerberos authentication and access the cluster service.

Unlike using the first type MIT Kerberos authentication, RAM identity authentication does not require adding principle to the Kerberos server in advance.

The following example uses subaccount test that has already been created to access a gateway:

- Add the test subaccount into the EMR cluster

The EMR 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 EMR cluster administrator executes the following code on the EMR cluster's master node:

```
sudo su hadoop
sh adduser.sh test 1 2
```

Attachment: adduser.sh code

```
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 StrictHostKeyChecking=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 /pathto/emrheader1
_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>
```

#### Attachment: config\_gateway\_kerberos.sh script

```
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
```

- Test user logs on to the gateway and configures AccessKey

```
Log on the test account of Gateway
Run the script
sh add_accesskey.sh test
```

#### Attachment: add\_accesskey.sh script (modify the AccessKey)

```
user=$1
if [[`cat /home/$user/.bashrc | grep 'export AccessKey'` == ""]];
then
 echo "
 # Change to the test user's AccessKeyId/AccessKeySecret
 export AccessKeyId=YOUR_AccessKeyId
 export AccessKeySecret=YOUR_AccessKeySecret
 " >> ~/.bashrc
```

```
else
 echo $user AccessKey has been added to .bashrc
fi
```

- Test user executes the command

After taking the preceding steps, 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
```

## 18.4 LDAP authentication

EMR cluster also supports authentication based on LDAP, which manages the account system through LDAP. Kerberos client uses LDAP account information as identity for authentication.

### LDAP Identity Authentication

LDAP account can be shared with other services, such as Hue. Users must only configure it on the Kerberos server. Users can use the LDAP service (ApacheDS) that has been configured in the EMR cluster or use the existing LDAP service. Users must only configure it on the Kerberos server.

Here's an example of an LDAP service (ApacheDS) that has been started by default in a cluster:

- Configure the basic environment in Gateway management (the same as that in the second part of the RAM, which can be skipped if it has been configured).

The only difference is that `auth_type` in `/etc/has/has-client.conf` needs to be modified to LDAP

Or the user can also not modify `/etc/has/has-client.conf`. The user test can copy the file and modify `auth_type` with their account and specify the path through environment variables, for example:

```
export HAS_CONF_DIR=/home/test/has-conf
```

- Configure LDAP manager user/password to Kerberos server end (Has) in the EMR console.

Enter the EMR console Cluster Management - HAS software, configure the LDAP manager user name and password to the corresponding `bind_dn` and `bind_password` fields and restart the HAS service.

In this example, the LDAP service is the ApacheDS in the EMR cluster, and related fields can be obtained from ApacheDS.

- EMR cluster manager adds user information to LDAP
  - Obtain ApacheDS LDAP service and manager user and password `manager_dn` and `manager_password` can be seen in EMR Console Cluster Configuration Management/ ApacheDS Configuration
  - Add user test and password in 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
objectclass: inetOrgPerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: test
sn: test
mail: test@example.com
userpassword: test1234
#Add to LDAP, in which -w denotes that password is changed to
manager_password
ldapmodify -x -h localhost -p 10389 -D "uid=admin,ou=system" -w "
NslaSe" -a -f test.ldif
#Delete test.ldif
rm test.ldif
```

Provide added user name/passowrd to user test.

- User test configures LDAP information

```
Log on the test account of Gateway
Run the script
```

```
sh add_ldap.sh test
```

Attachment: Script add\_ldap.sh (modifying 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
```

- User test access to the cluster services

Execute HDFS commands

```
[test@izbplcyio18s5ymggr7yhrZ ~]$ 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 Hadoop/Spark job.

## 18.5 Execution plan authentication

E-MapReduce clusters support execution plan authentication. You can authorize subaccounts to access execution plans using the master account.

### Master account access

After logging on to E-MapReduce console with the master account, you can run the corresponding execution plan on the Execution plan page. Submit the jobs to the security cluster for execution and access the related open source component services involved in the jobs using the hadoop username.

### Subaccount access

After logging on to E-MapReduce console with the RAM subaccount, you can run the corresponding execution plan on the Execution plan page. Submit the jobs to the security cluster for execution and access the related open source component services involved in the jobs using the corresponding username of the RAM subaccount.

## Examples

- The master account administrator can create multiple subaccounts (such as A, B, and C) as needed and grant the [Aliyun EMR Full Access](#) permissions to these subaccounts from the RAM console. Then, the subaccounts can log on to the E-MapReduce console and use the related functions.
- The master account administrator may provide the subaccounts to developers.
- After job creation and plan execution, developers may start executing the execution plans to submit jobs to the cluster, and then access the relevant component services in the cluster using usernames (such as A, B, and C) corresponding to their subaccounts.

**Note:**

Currently, the periodic execution plans are uniformly executed using the hadoop account.

- Relevant permission control for component services, for example, whether Account A has the permission to access a file in hdfs or not, is performed by using the username of a subaccount.

## 18.6 Cross-region access

Kerberos in E-MapReduce supports the cross-region feature, that is, different Kerberos clusters can inter-access each other.

This article describe cross-region access by 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 through executing the command `hostname` on emr-header-1.
- Region can be obtained in `/etc/krb5.conf` on emr-header-1.

### Add principal

emr-header-1 nodes in Cluster-A and Cluster-B run the same command exactly as follows:

```
root account
sh /usr/lib/has-current/bin/hadmin-local.sh /etc/ecm/has-conf -
k /etc/ecm/has-conf/admin.keytab
```

```
HadadminLocalTool.local: addprinc -pw 123456 krbtgt/EMR.6789.COM@
EMR.1234.COM 6789. COM@EMR. 1234. Com
```

**Note:**

- 123456 is the password, which can be changed.
- EMR.6789.COM is the region of Cluster-B, namely, the region of the cluster to be accessed.
- EMR.1234.COM is the region of Cluster-A, namely, the region of the cluster that initiates the 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_preference_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 needed) in the file /etc/hosts in Cluster-B 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 a job is running on Cluster-A to access Cluster-B, yarn must be restarted.

- Configure host binding information for all Cluster-A nodes.

### Access services in Cluster-B

The keytab file /ticket in Kerberos of Cluster-A can be used on Cluster-A as a cache to access services in Cluster-B. Matters:

For example, access hdfs service in Cluster-B:

```
su has;
hadoop fs -ls hdfs://emr-header-1.cluster-6789:9000/
Found 4 items
-rw-r----- 2 has 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
```

# 19 Component authorization

## 19.1 HDFS authorization

When HDFS is enabled, user access to HDFS requires legal permissions in order to operate HDFS properly, such as read data, create folders and others.

### Add a configuration

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 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, i.e. the permission of file creation is 644 ( $666 \& \text{^}022 = 644$ ), and permission of folder creation is 755 ( $777 \& \text{^}022 = 755$ ).

— The default setting of Kerberos security cluster in the EMR is 027, the corresponding permission of file creation is 640, and permission of folder creation is 750.

- `dfs.namenode.acls.enabled`

— Enable ACL control. This gives you permission control on owner/group, 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 permissions 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 the 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), without configuration and service restart.

For non-Kerberos security clusters, a configuration must be added and the service must be restarted.

### Other

- Umask value can be modified as needed.
- 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 users other than superuser/file owner/dir owner from deleting files/folders in the folder (even if other users have rwx permissions on the 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
```

## 19.2 YARN authorization

YARN authorization can be divided to service-level authorization and queue-level authorization based on the authorization entity.

### Service-level authorization

For more information, see [Hadoop official documentation](#).

- Control cluster service access by specific users, such as submitting jobs
- Configures `hadoop-policy.xml`
- Service level permission validation has a higher priority than other permission validation procedures (such as HDFS permission verification and YARN job submission control)

**Note:**

Generally, if HDFS permission verification and YARN job submission control have been set up, you may choose not to set the service level permission control. You can perform the relevant configurations as needed.

### Queue-level authorization

YARN supports permission control over resources through queues, and it provides two queue scheduling methods, namely Capacity Scheduler and Fair Scheduler. We will take the Capacity Scheduler as an example here.

- Add a configuration

A queue also has a two-level authorization: the authorization for job submission (submitting a job to the queue) and the authorization for queue management.

**Note:**

- The ACL control object for a queue is `user/group`. The users and groups can be set at the same time (separated by spaces) when you set up the relevant parameters. You can use a comma to separate different users and groups. Using only one space means that no one has the permission.
- ACL inheritance for a queue: If a `user/group` can submit an application to a queue, then this `user/group` can submit applications to all sub-queues of this queue. Likewise, the ACL that manages queues can also be inherited. Therefore, if you want to prevent a `user/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 of this user/group.

— `yarn.acl.enable`

ACL switch, set to true

— `yarn.admin.acl`

- YARN administrator setting, which enables/disables executing `yarn radmin/yarn kill`, and other commands. This value must be configured, otherwise, the subsequent queue-based ACL administrator settings cannot take effect.
- As mentioned in the preceding note, you can set up user/group when setting up the 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 EMR cluster, you must configure the ACL permission for has as admin.

— `yarn.scheduler.capacity.${queue-name}.acl_submit_applications`

- Set user/group that can submit jobs to this queue
- where `${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_applications</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
_applications</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 user/group to manage the queue, such as killing a job 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_administer_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
  - The Kerberos secure cluster has enabled ACL by default. You can configure the relevant ACL permissions for queues as needed.
  - For a non-Kerberos secure cluster, enable ACL and configure the permission control for queues in accordance with the preceding instructions, and 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 do not allow any user to submit jobs or manage the queue.
- Q1 queue: only allows the test user to submit jobs and manage the queue (such as killing the jobs).
- Q2 queue: only allows the foo user to submit jobs and manage the queue. Q2 Queues: Only Foo users are allowed to submit jobs and manage queues.

```

<configuration>
 <property>
 <name>yarn.scheduler.capacity.maximum-applications</name>
 <value>10000</value>
 <description>Maximum number of applications that can be
pending and running.</description>
 </property>
 <property>
 <name>yarn.scheduler.capacity.maximum-am-resource-percent</
name>
 <value>0.25</value>
 <description>Maximum percent of resources in the cluster
which can be used to run application masters i.e.
controls number of concurrent running applications.

```

```

 </description>
 </property>
 <property>
 <name>yarn.scheduler.capacity.resource-calculator</name>
 <value>org.apache.hadoop.yarn.util.resource.DefaultResourceCalculator</value>
 </property>
 <property>
 <name>yarn.scheduler.capacity.root.queues</name>
 <value>default,q1,q2</value>
 <!--3 queues-->
 <description>The queues at the this level (root is the root queue).</description>
 </property>
 <property>
 <name>yarn.scheduler.capacity.root.default.capacity</name>
 <value>0</value>
 <description>Default queue target capacity.</description>
 </property>
 <property>
 <name>yarn.scheduler.capacity.root.default.user-limit-factor</name>
 <value>1</value>
 <description>Default queue user limit a percentage from 0.0 to 1.0.</description>
 </property>
 <property>
 <name>yarn.scheduler.capacity.root.default.maximum-capacity</name>
 <value>100</value>
 <description>The maximum capacity of the default queue.</description>
 </property>
 <property>
 <name>yarn.scheduler.capacity.root.default.state</name>
 <value>STOPPED</value>
 <!-- Status of the default queue is set as STOPPED-->
 <description>The state of the default queue. State can be one of RUNNING or STOPPED.</description>
 </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, automatically 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 administrators 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 inheritance, the parent queue must have the admin
permissions-->
 <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_administer_queue</
name>
 <value> </value>
 <!-- ACL inheritance, the parent queue must have the admin
permissions-->
</property>
</property>
 <name>yarn.scheduler.capacity.root.q1.acl_administer_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_administer_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>
</configuration>

```

## 19.3 Hive authorization

Hive has two authorization modes, storage based authorization and SQL standard based authorization. For more information, see [official Hive documentation](#).



### Note:

The two authorization features can be configured at the same time without conflict.

Storage based authorization is for HiveMetaStore.

Scenario:

If a user in the cluster has a direct access to data in Hive through HDFS/Hive Client, a permission control must be performed on Hive data in HDFS. Through HDFS permission control, operation permissions related to Hive SQL can be controlled.

For more information, see [Hive documents](#).

## Add configuration

In the cluster Configuration Management page, **Hive > Configuration > hive-site.xml > Add Custom Configuration**.

```
<property>
<name>hive.metastore.pre.event.listeners</name>
 <value>org.apache.hadoop.hive.ql.security.authorization.Authorizat
ionPreEventListener</value>
</property>
<property>
<name>hive.security.metastore.authorization.manager</name>
 <value>org.apache.hadoop.hive.ql.security.authorization.StorageBas
edAuthorizationProvider</value>
</property>
<property>
<name>hive.security.metastore.authenticator.manager</name>
 <value>org.apache.hadoop.hive.ql.security.HadoopDefaultMetasto
reAuthenticator</value>
</property>
```

## Restart HiveMetaStore

Restart HiveMetaStore in the cluster Configuration Management page.

## HDFS permission control

HDFS related permissions of warehouse in Hive has been set for Kerberos security cluster in the EMR.

For non-Kerberos security cluster, users must set basic HDFS permission through the following steps:

- EnableHDFS permissions
- Configure permissions of warehouse in Hive

```
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 (mentioned earlier), related users/user groups can normally create/read/write tables through 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
```

```
hadoop fs -setfacl -m group:hivegrp:rwX /user/hive/warehouse
```

With the HDFS authorization, related users/user groups can normally create/read/write tables, and data in Hive tables created by different users in HDFS can only be accessed by the users themselves.

## Verification

- User test creates a table testtbl.

```
hive> create table testtbl(a string);
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive
.ql.exec.DDLTask. MetaException(message:Got exception: org.apache.
hadoop.security.AccessControlException Permission denied: user=test
, access=WRITE, inode="/user/hive/warehouse/testtbl":hadoop:hadoop:
drwxrwx--t
at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(
FSPermissionChecker.java:320)
at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(
FSPermissionChecker.java:292)
```

An error occurs due to no permissions. Permissions should be granted to the user test.

```
#Switch from root account to has account
su has
#Add ACL and grant rwX permissions of the directory warehouse to the
account test.
hadoop fs -setfacl -m user:test:rwX /user/hive/warehouse
```

The account test 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 permissions 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 to table testtbl.

```
#drop table
hive> drop table testtbl;
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.
ql.exec.DDLTask. MetaException(message: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)
```

```

 at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
 checkPermission(FSPermissionChecker.java:190)
#alter table
hive> alter table testtbl add columns(b string);
FAILED: SemanticException Unable to fetch table testtbl. java.
security.AccessControlException: 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)
 at org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.
 checkPermission(FSPermissionChecker.java:190)
 at org.apache.hadoop.hdfs.server.namenode.FSDirectory.checkPermi
 ssion(FSDirectory.java:1720)
#select
hive> select * from testtbl;
FAILED: SemanticException Unable to fetch table testtbl. java.
security.AccessControlException: 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)

```

It can be seen that the user foo cannot perform any operations on the table created by the user test. 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:

In general, a Hive user group can be created and authorized, then add new users the group.

## SQL Standards Based Authorization

- Scenario

If a cluster user can't access through a HDFS/Hive client, and the only way is to run Hive related commands through `HiveServer` (beeline, jdbc, and so on). SQL Standards Based Authorization can be used.

If you are able to use methods such as Hive shell, as long as no related configuration has been made to the `hive-site.xml` in the user's client, Hive can still be normally accessed even if the following settings are implemented.

For more information, see [Hive documentation](#).

- 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 detailed command operations, click [here](#).

- Verification

- User foo access to user test'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_VIEW, 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
```

- User foo can normally select.

```
0: jdbc:hive2://emr-header-1.cluster-xxxxx:10> select * from
testtbl;
INFO : OK
+-----+
| testtbl.a |
+-----+
```

```
| hz |
+-----+
1 row selected (0.787 seconds)
```

#### — Revoke permission.

```
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 could not select data for testtbl.

```
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)
```

## 19.4 HBase authorization

Without authorization, any account can perform any operations on the HBase cluster that includes disable table, drop table, major compact, and others.



#### Note:

For clusters without Kerberos authentication, users can forge identities to access to the cluster service even when HBase authorization is enabled. Therefore, we recommend that you create a cluster with high security mode (for example, supporting Kerberos) as detailed in [Kerberos Security Document](#).

### Add configuration

In Configuration Management, 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.regionserver.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 Configuration Management page, click **HBase > Operations > RESTART All Components**.

## Authorization (ACL)

- Basic concepts

Authorization is for grant [operation permissions] of [resources in a certain scope] to [a certain entity].

In HBase, the preceding three concepts are:

### — Resources in a certain scope

- Superuser

A Superuser can perform any operations, and the account running HBase service is the Superuser by default. You can also add Superusers through configuring the value of `hbase.superuser` in `hbase-site.xml`.

- Global

Global Scope has Admin permissions of all tables in the cluster.

- Namespace

It has permission control in Namespace Scope.

- Table

It has permission control in Table Scope.

- ColumnFamily

It has permission control in ColumnFamily Scope.

- Cell

It has permission control in Cell Scope.

### — Operation permission

- 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/delete a table in a certain Scope.

- Admin (A)

Perform cluster related operations in a certain Scope, such as balance/assign.

- A certain entity

- User

Authorize a user

- Group

Authorize a user group

- Authorization command

- grant

```
grant <user> <permissions> [<@namespace> [<table> [<column family>
[<column qualifier>]]]]
```



**Note:**

- The authorization methods for users or groups are the same. A prefix @ needs to be added for group.

```
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.
```

- A 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)

## 19.5 Kafka authorization

If Kafka authentication (for example, Kerberos authentication or simple authorization based on username 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-mode Kafka cluster. For more information, see [Kerberos](#).



### Note:

The permission configurations described in this document are only for high-security mode clusters of E-MapReduce. That is, Kafka is started in Kerberos mode.

### Add configurations

1. On the **Cluster Management** page, click **View Details** after the Kafka cluster.
2. In the left-side navigation pane, 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                                   | Note                                                                              |
|-----------------------|-----------------------------------------|-----------------------------------------------------------------------------------|
| authorizer.class.name | kafka.security.auth.SimpleAclAuthorizer | None                                                                              |
| 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 in this step. With the configuration set to true, only the users who are named Kafka and have passed the Kerberos authentication can run the kafka-topics.sh command in the Kerberos environment. Kafka-topics.sh will directly read, write, and modify data in ZooKeeper.

### Restart a Kafka cluster

1. On the **Cluster Management** page, click **View Details** after the Kafka cluster you want to operate in the Operation column.

2. In the left-side navigation pane, click the **Clusters and Services** tab , and click **Actions** at the right side of Kafka on the service list.
3. In the the drop-down menu, select **RESTART All Components**. Enter a record information and click **OK**.

## Authorization (ACL)

- Basic concepts

Definition in the official Kafka documents:

```
Kafka ACLs are defined in the general format of "Principal P is [
Allowed/Denied] Operation O From Host H On Resource R"
```

That is, the ACL process relates to Principal, Allowed/Denied, Operation Host, and Resource.

— Principal: username

| Security protocol | Value                                                                                                                                                               |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PLAINTEXT         | ANONYMOUS                                                                                                                                                           |
| SSL               | ANONYMOUS                                                                                                                                                           |
| SASL_PLAINTEXT    | When the mechanism is PLAIN, the user name is specified by client_jaas.conf. When the mechanism is GSSAPI, the username 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, for example, the supporting relationships between resources and the authorization of operations, see [KIP-11 - Authorization Interface](#).

- Authorization command

Perform authorization by using the script `kafka-acls.sh` (`/usr/lib/kafka-current/bin/kafka-acls.sh`).For details about how to use this script to authorize Kafka, run the `kafka-acls.sh --help` command and view how to use the command.

## Operation example

Perform related operations on the master node of the created E-MapReduce high-security mode Kafka cluster.

1. Create a user named test.

```
useradd test
```

2. Create a topic.

As mentioned in section 1 “Add configurations”, `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_jaas.conf.
export KAFKA_HEAP_OPTS="-Djava.security.auth.login.config=/etc/ecm/
kafka-conf/kafka_client_jaas.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:

```
[2018-02-28 22:25:36,178] INFO Kafka commitId : aaa7af6d4a11b29d (
org.apache.kafka.common.utils.AppInfoParser)
>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_PLAINTEXT
sasl.mechanism=GSSAPI
```

**b. Run kafka-console-consumer.sh.**

```
su test
Kafka_client_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_client_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.GroupAuthorizationException: 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_client_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_client_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_client_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:

```
alibaba
```

E-MapReduce

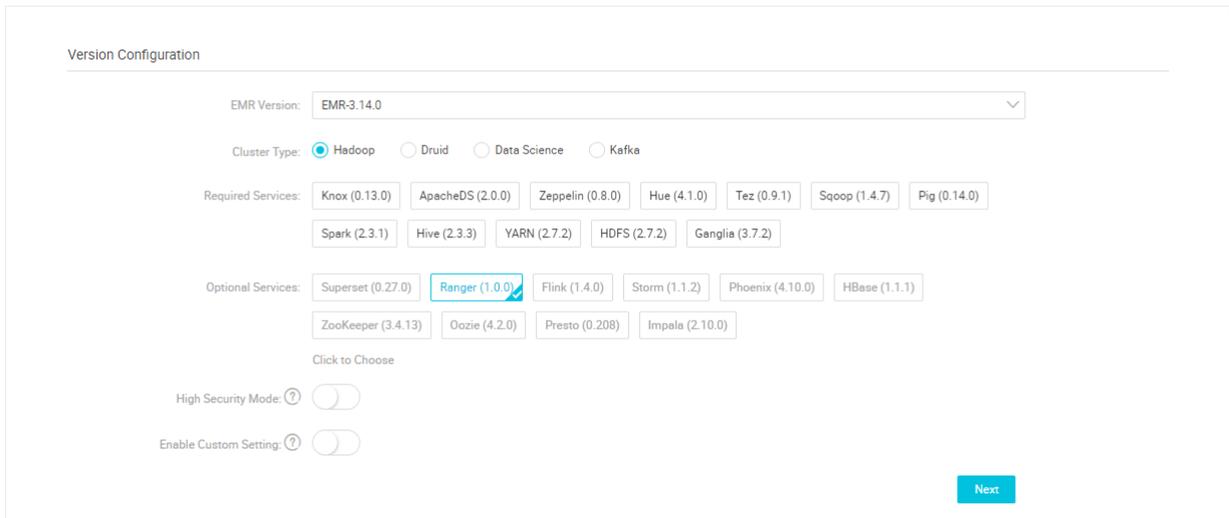
## 19.6 RANGER

### 19.6.1 Ranger introduction

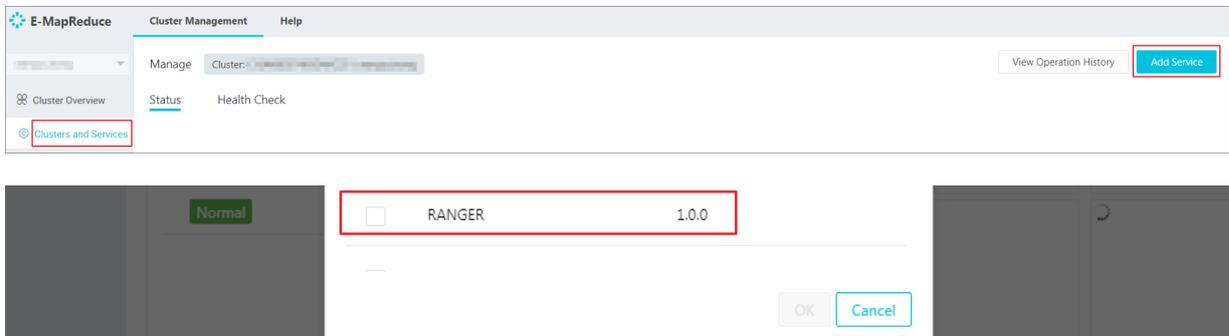
Apache Ranger provides a centralized framework for permission management. It can implement fine-grained access control for components in the Hadoop ecosystem, such as HDFS, Hive, Yarn, Kafka, Storm, and Solr. It provides WebUI that allows administrators to perform operations more conveniently.

#### Create a cluster

Select Ranger service when you create a cluster of EMR-2.9.2/EMR-3.9.0 or a later version on the E-MapReduce console.

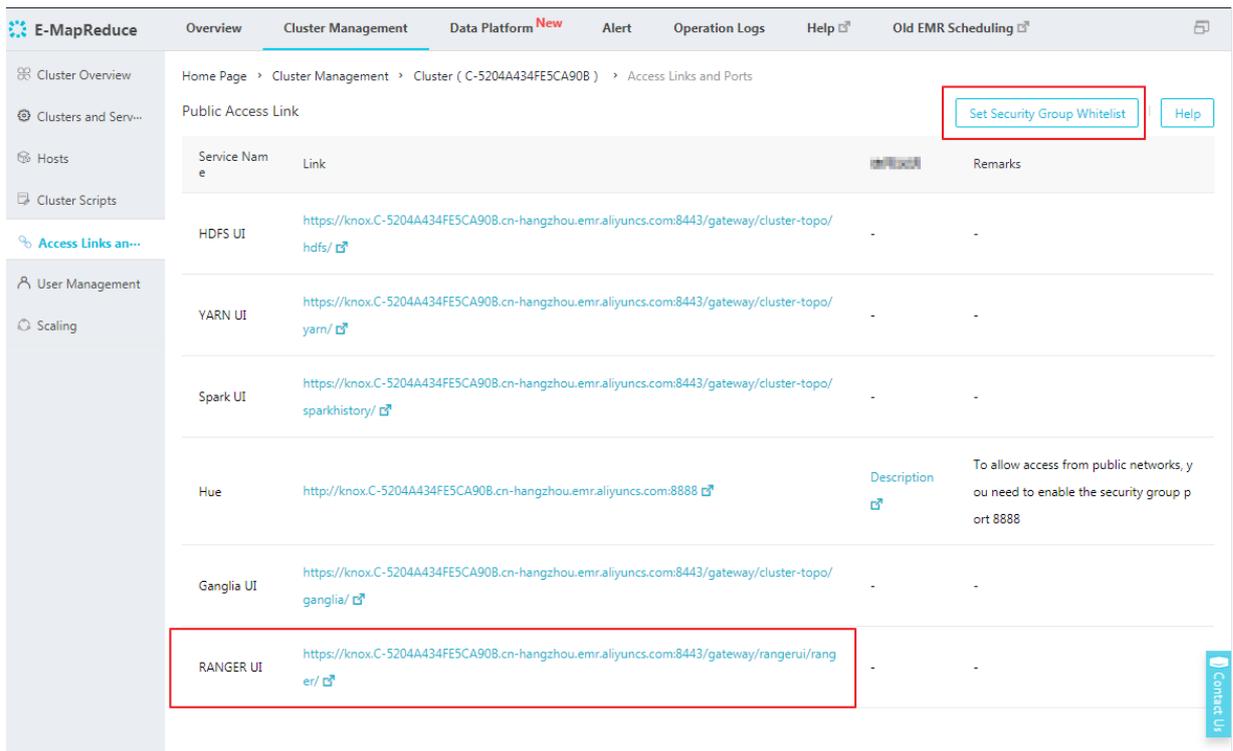


If an E-MapReduce cluster of EMR-2.9.2 or EMR-3.9.0 or a later version has been created without Ranger, you can go to the Clusters and Services page to add the Ranger service.

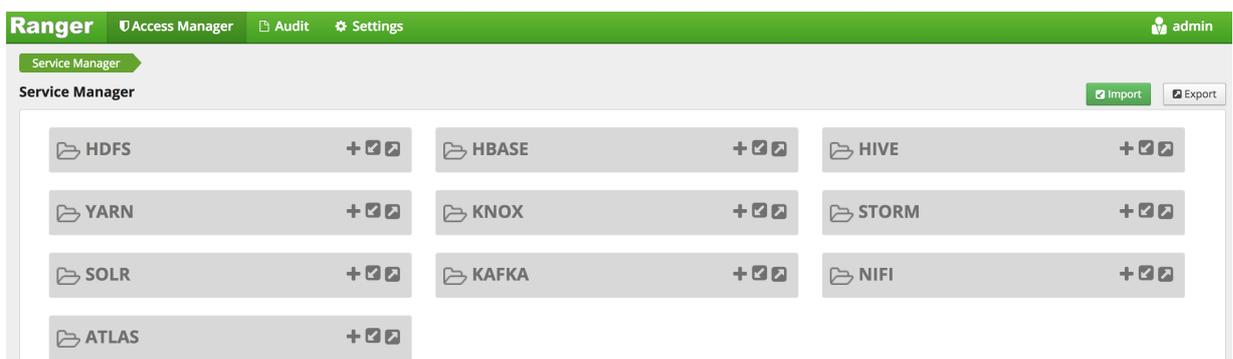
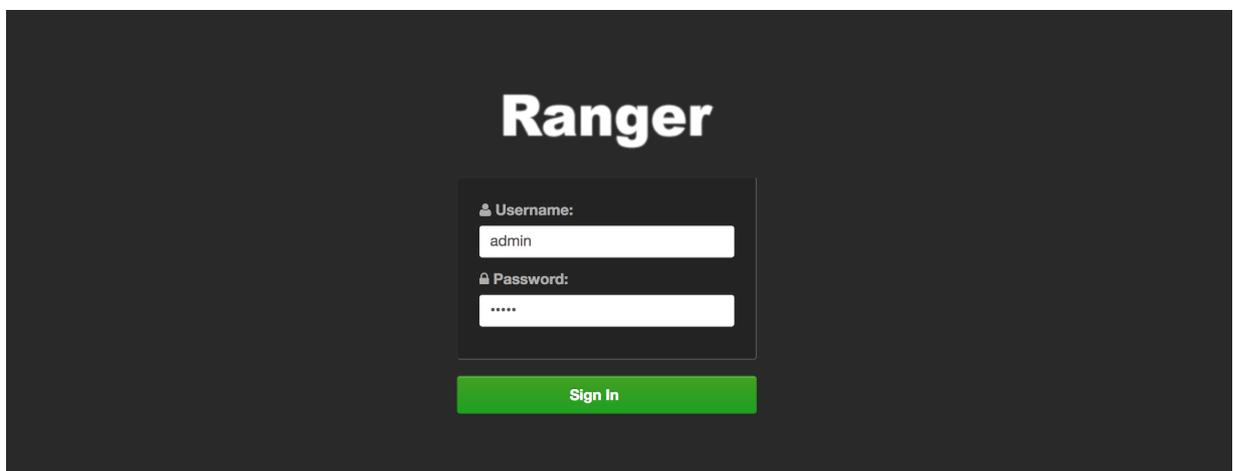


#### Ranger UI

Behind the cluster installed with Ranger, click **Manage** in the **Actions** column. Click **Access Links and Ports** in the left navigation pane to access Ranger WebUI through quick links.



Go to the Ranger WebUI. The default Username/Password is admin/admin, as shown in the following figure.



## Modify the password

Upon the first logon, the administrator needs to modify the password of the admin account, as shown in the following figure.

The figure consists of two screenshots from the Ranger web interface. The top screenshot shows the 'Settings' menu highlighted in the top navigation bar. Below it, the 'User List' table is displayed with the 'admin' user selected. The bottom screenshot shows the 'User Edit' page for the 'admin' user, with the 'Change Password' tab selected and the 'New Password' and 'Password Confirm' fields highlighted.

|                                     | User Name      | Email Address | Role  | User Source | Groups | Visibility |
|-------------------------------------|----------------|---------------|-------|-------------|--------|------------|
| <input checked="" type="checkbox"/> | admin          |               | Admin | Internal    | hadoop | Visible    |
| <input type="checkbox"/>            | rangerusersync |               | Admin | Internal    | --     | Visible    |
| <input type="checkbox"/>            | rangertagsync  |               | Admin | Internal    | --     | Visible    |
| <input type="checkbox"/>            | hive           |               | User  | External    | hadoop | Visible    |

After you change the admin password, click **Log Out** in the drop-down list of **admin** in the upper right corner. Log on with the new password.

## Integrate services into Ranger

After the preceding settings, you can integrate the relevant services in the cluster into Ranger to control relevant permissions. For details, see the following sections.

- [Integrate HDFS into Ranger](#)
- [Integrate Hive into Ranger](#)
- [Integrate HBase into Ranger](#)

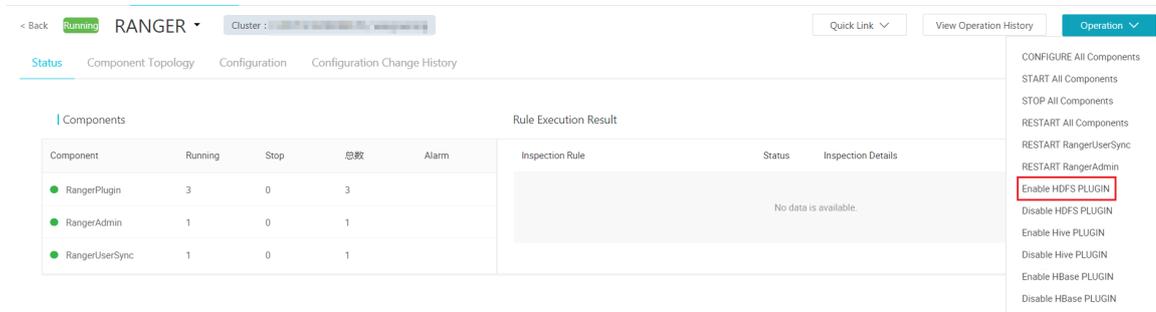
## 19.6.2 HDFS configuration

### Integrate Ranger into HDFS

The previous section introduced how to create and start the cluster of the Ranger service in E-MapReduce. This section describes the step-by-step process for integrating Ranger into HDFS.

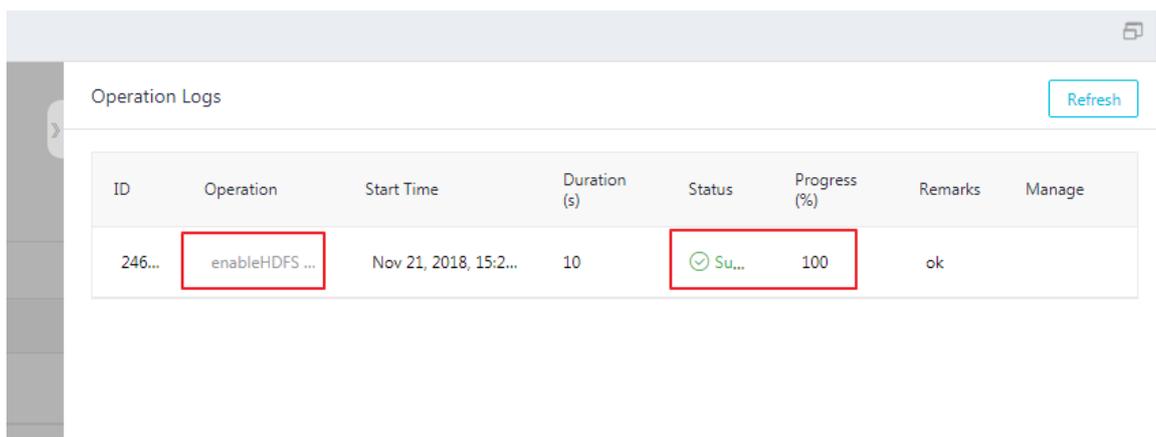
- Enable HDFS Plugin
  1. On Cluster Management page, click **Manage** after the cluster you want to operate in the **Actions** column.

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, and select **Enable HDFS PLUGIN**, then click **OK**.



4. Enter record information in the prompt box, and then click **OK**.

You can check the progress by clicking **View Operation Logs** in the upper right corner of the page.



- Restart NameNode

After the preceding task is completed, you need to restart NameNode. To restart NameNode, perform the following steps:

1. In the Ranger management page, click the inverted triangle icon behind RANGER in the upper left corner to switch to 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.

The screenshot shows the Ranger HDFS configuration page for cluster C-5204A434F5ECA90B / RANGER\_test\_hfx. The page is divided into two main sections: Components and Rule Execution Result.

| Component         | Normal | Stop | Total | Alert | Actions        |
|-------------------|--------|------|-------|-------|----------------|
| HttpFS            | 0      | 1    | 1     |       | Start          |
| KMS               | 0      | 1    | 1     |       | Start          |
| NameNode          | 1      | 0    | 1     |       | Restart   Stop |
| DataNode          | 2      | 0    | 2     |       | Restart   Stop |
| SecondaryNameNode | 1      | 0    | 1     |       | Restart   Stop |
| HDFS Client       | 3      | 0    | 3     |       | Restart   Stop |

| Inspection Rule       | Status          | Inspection Details |
|-----------------------|-----------------|--------------------|
| AgentHeartBeatCheck   | Service Running |                    |
| TotalDFSUsedCheck     | Service Running |                    |
| DataNodeDFSUsedCheck  | Service Running |                    |
| NameNodeHttpPortCheck | Service Running |                    |
| NameNodeJpcPortCheck  | Service Running |                    |
| NameNodeSafeModeCheck | Service Running |                    |

An 'Actions' dropdown menu is open, showing options such as 'RESTART NameNode', which is highlighted with a red box.

- Add HDFS service on Ranger WebUI

About how to enter the Ranger UI page, see [Ranger introduction](#).

Add HDFS service.

The screenshot shows the Ranger Service Manager interface. The 'Access Manager' tab is selected. Under the 'Service Manager' section, there are several service cards: HDFS, HBASE, HIVE, YARN, KNOX, STORM, SOLR, KAFKA, NIFI, and ATLAS. Each card has a '+' icon and a 'lock' icon. The '+' icon for the HDFS service is highlighted with a red box.

— Standard cluster

If the cluster you created is in standard mode (go to the **Cluster Overview** page of the cluster to check Security Mode to make sure whether it is or not), configure as follows:

**Ranger** Access Manager Audit Settings

Service Manager > Create Service

### Create Service

**Service Details :**

Service Name \*  fixed value:emr-hdfs

Description

Active Status  Enabled  Disabled

Select Tag Service

**Config Properties :**

Username \*

Password \*

HA cluster: hdfs://emr-header1:8020  
NameNode URL \*

Non-HA cluster: hdfs://emr-header1:9000

Authorization Enabled  default value of Standard cluster (non- high-security-mode cluster)

Authentication Type \*

— High-security-mode cluster

If the cluster you created is in High-security mode(go to the go to the **Cluster Overview** page of the cluster to check Security Mode to make sure whether it is or not), configure as follows:

**Ranger** Access Manager Audit Settings

Service Manager > Create Service

### Create Service

**Service Details :**

Service Name \*  fixed value: emr-hdfs

Description

Active Status  Enabled  Disabled

Select Tag Service

**Config Properties :**

Username \*

Password \*

HA cluster: hdfs://\${master1\_fullhost}:8020  
you can log onto master1 and run hostname cmd  
to get the value of \${master1\_fullhost}

Namenode URL \*

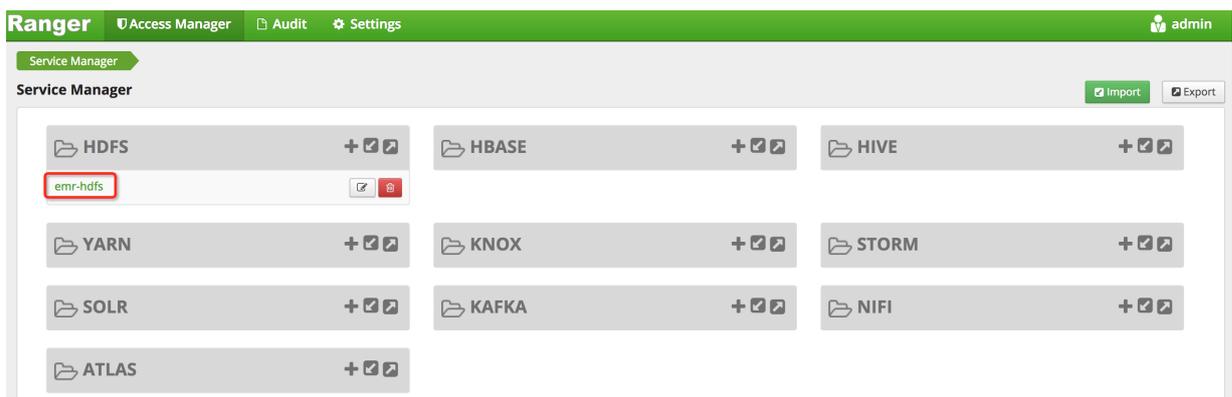
Non-HA cluster: hdfs://\${master1\_fullhost}:9000

Authorization Enabled  default value of high-security-mode cluster

Authentication Type \*

### Permission Configuration Example

The preceding section has already integrated Ranger into HDFS, which allows you to set relevant permissions. For example, authorize user test the write or execute permission of the `/user/foo`.



Click **emr-hdfs** in the preceding figure to enter the policy configuration page.

**Ranger** Access Manager Audit Settings

Service Manager emr-hdfs Policies Create Policy

**Create Policy**

**Policy Details :**

Policy Type **Access**

Policy Name \*  Any name is available. **enabled**

Resource Path \*  Press Enter after a resource path is selected. You can select multiple paths. **recursive** Whether subdirectories/files inherit permissions

Audit Logging **YES**

Description

Policy Label

**Allow Conditions :** Users/Groups are synchronized from the master.1 machine which takes about 1 minute. You can also add the relative users/groups on your cluster in advance.

You can also set permissions according to group settings.

| Select Group                              | Select User                                      | Permissions                                                    | Delegate Admin           |                          |
|-------------------------------------------|--------------------------------------------------|----------------------------------------------------------------|--------------------------|--------------------------|
| <input type="text" value="Select Group"/> | <input type="text" value="admin"/> Select a user | <input type="checkbox"/> Execute <input type="checkbox"/> Read | <input type="checkbox"/> | <input type="checkbox"/> |

**+** You can authorize multiple users/groups. **Select permissions to be authorized.**

In terms of the preceding settings, the permissions are granted to user test. User test can access the HDFS path of `/user/foo`.



#### Note:

The policy takes effect in 1 minute after it is added.

## 19.6.3 Hive configuration

### Integrate Hive into Ranger

[Ranger introduction](#) introduced how to create and start a cluster of the Ranger service in E-MapReduce and some preparation work. 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

- Scenario: You can access Hive data only through HiveServer2, not through Hive client or HDFS.
- Mode: Use the Beeline client or the JDBC code to run the related Hive script through HiveServer2.
- Permission settings:

The [SQL Standards Based Authorization](#) in Hive is used to control the permissions for the usage scenarios of HiveServer2.

The table or column level permission control for Hive in Ranger is also the usage scenario for HiveServer2. If you are still able to access Hive data through Hive client or HDFS, table or column level control is insufficient. Further permission control for the following two ways is required.

- Hive client
  - Scenario: Users can access Hive data from Hive Clients.
  - Mode: Access from Hive clients.
  - **Permission settings:**

The Hive client requests HiveMetaStore to perform some DDL operations such as alter table, add columns, and read and process data in HDFS by submitting MapReduce jobs.

The [Storage Based Authorization](#) in Hive is used to control the permissions for the usage scenarios of Hive client. It will determine whether this user can perform relevant 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)`.

You can control the permissions of the HDFS path in Hive tables in Ranger. This, in combination with the HiveMetaStore configuration (Storage Based Authorization) enables you to implement permission control over the access scenario of Hive client.

**Note:**

The DDL operation permissions in Hive client scenarios are actually controlled by the underlying HDFS. If you have HDFS permissions, you will also have the DDL operation permissions for the tables (such as drop tables and alter tables).

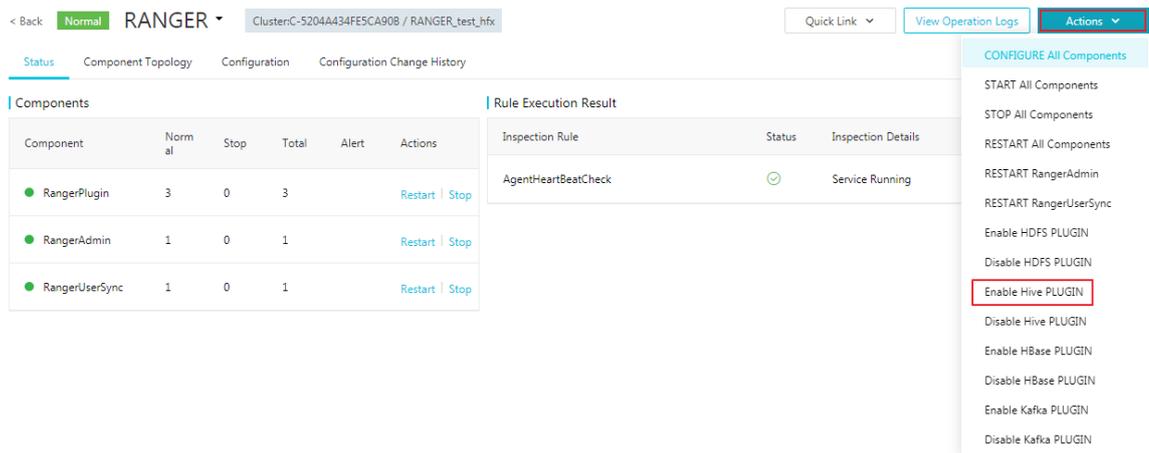
- HDFS Mode
  - Scenario: Users have direct access to HDFS.
  - Mode: HDFS client and code.
  - Permission settings:

To have direct access to HDFS, you need to add permission control for HDFS on the underlying HDFS data for Hive tables.

You can perform [Permission control](#) for the underlying HDFS path of Hive tables through Ranger.

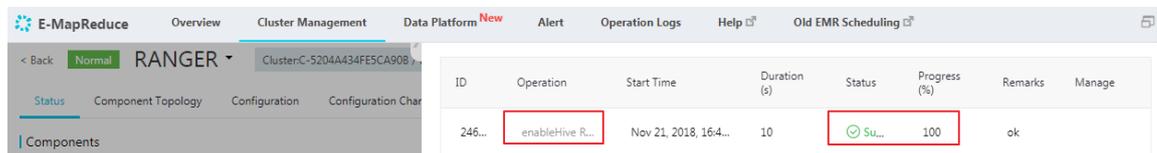
- Enable Hive Plugin

1. On Cluster Management page, click **Manage** after 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, and select **Enable Hive PLUGIN**, then click **OK**.



4. Enter record information in the prompt box, and then click **OK**.

You can check the progress by clicking **View Operation Logs** at the upper right corner of the page.



**Note:**

After you enable Hive Plugin and restart Hive, HiveServer2 scenario and HiveClient scenario (Storage Based Authorization) are configured accordingly. For information about HDFS permissions, see [HDFS configuration](#).

- Restart Hive

After the preceding task is completed, it is necessary to restart Hive to make it effect.

1. In the Ranger management page, click the inverted triangle icon behind RANGER in the upper left corner to switch to Hive.
2. Click **Actions** in the upper-right corner, Select **RESTART All Components** from the drop-down menu, and then 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 WebUI

About how to enter the Ranger UI page, see [Ranger introduction](#).

Add the Hive service to the Ranger WebUI

**Service Details :**

Service Name \*  Fixed input: emr-hive

Description

Active Status  Enabled  Disabled

Select Tag Service

**Config Properties :**

Username \*  All inputs are available.

Password \*

jdbc.driverClassName \*  Fixed input

jdbc.url \*    
For high-security clusters, input jdbc:hive2://{master1\_fullhost}:1000;principal=hive://{master1\_fullhost}@EMR.Sid.COM  
 For standard clusters, input jdbc:hive2://emr-header-1:1000  
 Log on to master and execute hostname to obtain \${master\_fullhost}.  
 The number in \${master\_fullhost} is \${id}.

Common Name for Certificate

Add New Configurations

| Name                                                    | Value                               |
|---------------------------------------------------------|-------------------------------------|
| <input type="text" value="policy.download.auth.users"/> | <input type="text" value="hadoop"/> |

Input hadoop for standard clusters.  
Input hive for security clusters.

Ignore test contention failures.

— 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-security mode cluster: jdbc:hive2://\${master1_fullhost}:10000/;principal=hive/\${master1_fullhost}@EMR.\$id.COM |
| policy.download.auth.users | standard cluster: hadoop high-security mode cluster: hive                                                                                                                |

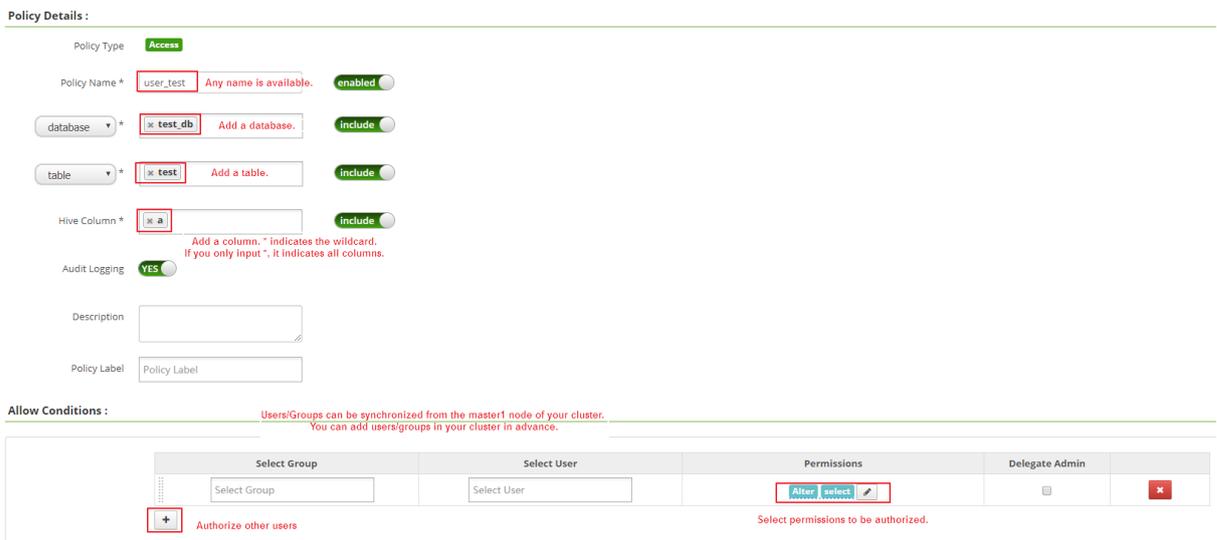
`\${master1\_fullhost}` is a long domain name of master1, you can log on to master 1 and run `hostname` command to obtain this name. The number in `\${master1\_fullhost}` is the value of `$id`.

### Permission configuration example

Ranger is integrated into Hive in the previous section, which allows you to set relevant permissions. For example, authorize the user foo for the Select permission for column a of the table testdb.test.



Click **emr-hive** in the preceding figure to configure relevant permissions.



In terms of the preceding settings, the permissions are granted to the user foo. The user foo can access the table testdb.test.

**Note:**

The policy takes effect in 1 minute after it is added.

## 20 Disaster recovery

---

### 20.1 EMR cluster disaster tolerance

#### Data disaster tolerance

The Hadoop Distributed File System (HDFS) stores the data of each file in blocks, and each block has some copies (Each block has three copies by default). This makes sure these copies of data block can be stored in the different frameworks. In most situations, the storage strategy of HDFS is to store the first copy in the local framework, the second copy is stored in the same framework with the first one, but in different nodes, the last copy is stored in the different frameworks.

HDFS will scan the data copies regularly, if a data copy was lost, HDFS will make another data copy quickly to make sure the number of data copy is stable. If a node that stores a data copy was lost, HDFS will make another node to recover to data in that node. In Alibaba Cloud, if you use cloud disk, each cloud disk has three data copies in the backend, whenever any of them has some issues, the data copies will exchange and recover data to ensure the reliability of data.

HDFS is a file storage system that has stood the test of time and has high reliability. It can store massive data with high reliability. At the same time, based on the features in Alibaba Cloud, HDFS can make extra backups for the data stored in OSS, in this way, HDFS makes the data higher reliability.

#### Service disaster tolerance

The core components of HDFS will deploy the HA, that is, there are at least two nodes be the backups for each other, such as, YARN, HDFS, Hive Server, Hive Meta. In this way, whenever there's a node with issues, the nodes will exchange and recover data to ensure that the services has no impact.

## 21 Resource pool

The Dynamic Resource Pools function is a usage strategy for the Yarn application. E-MapReduce Yarn uses the capacity scheduler by default. EMR Yarn uses the capacity scheduler by default.

### Enable Resource Pool

1. Log on to the [Alibaba Cloud E-MapReduce console](#) and enter the cluster list page.
2. Click the **Manage** link corresponding to the cluster you want to configure.
3. In the service list, click **YARN**, and go to the Yarn configuration page.
4. At the top of the page, click the **Resource Pool** tab to go to the resource pool configuration page.
5. Click **Initialize Resource Pool** button to select a scheduler policy. E-MapReduce supports [Capacity Scheduler](#) and [Fair Scheduler](#).



#### Note:

With the Resource Pool function enabled, once the Capacity Scheduler or Fair Scheduler is selected, it cannot be changed later. You can just configure relevant parameters in the resource pool.

### Configure Resource Pool

When you select the scheduler policy, click the drop-down list of **More settings** for the global configuration. It is recommended that the system administrator set these configurations.

- **Capacity Scheduler**

**Table 21-1: More configurations**

| EMR parameters                                       | Hadoop Yarn parameters                                                                                                                              |
|------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Default configuration-maximum number of applications | Configure global parameters yarn.scheduler.capacity.maximum-applications or yarn.scheduler.capacity.<queue-path>.maximum-applications               |
| Default configuration-maximum am ratio               | Configure global parameters yarn.scheduler.capacity.maximum-am-resource-percent or yarn.scheduler.capacity.<queue-path>.maximum-am-resource-percent |
| Default configuration-resources calculation class    | yarn.scheduler.capacity.resource-calculator                                                                                                         |

| EMR parameters                              | Hadoop Yarn parameters                                                                       |
|---------------------------------------------|----------------------------------------------------------------------------------------------|
| Default configuration-number of node delays | yarn.scheduler.capacity.node-locality-delay                                                  |
| Placement rules                             | Configure mappings between users/groups and queue                                            |
| Placement rules: queue mapping override     | yarn.scheduler.capacity.queue-mappings-override.enable                                       |
| ACL settings-enable ResourceManager ACL     | yarn.acl.enable on the yarn-site                                                             |
| ACL settings-manage ACL                     | yarn.admin.acl on the yarn-site                                                              |
| User limit-acl_submit_applications          | Configure global parameter yarn.scheduler.capacity.root.<queue-path>.acl_submit_applications |
| User limit-acl_administer_queue             | Configure global parameter yarn.scheduler.capacity.root.<queue-path>.acl_administer_queue    |

**Table 21-2: Create Resource Pool**

| EMR parameters                                                  | Hadoop Yarn parameters                                                                                                                                                    |
|-----------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Resource Pool name                                              | Queue name in yarn                                                                                                                                                        |
| Resource pool limit-proportion                                  | yarn.scheduler.capacity.<queue-path>.capacity                                                                                                                             |
| Resource pool limit-miniUserLimit                               | yarn.scheduler.capacity.<queue-path>.minimum-user-limit-percent                                                                                                           |
| Resource pool limit-maximumCapacity                             | yarn.scheduler.capacity.<queue-path>.maximum-capacity                                                                                                                     |
| Resource pool limit-limit on the proportion of individual users | yarn.scheduler.capacity.<queue-path>.user-limit-factor                                                                                                                    |
| Resource pool limit-maximum memory                              | yarn.scheduler.capacity.<queue-path>.maximum-allocation-mb                                                                                                                |
| Resource pool limit-maximum number of cores                     | yarn.scheduler.capacity.<queue-path>.maximum-allocation-vcores                                                                                                            |
| Resource pool limit-maximum number of applications              | The priority of the parameter is higher than the global variable yarn.scheduler.capacity.maximum-applications / yarn.scheduler.capacity.<queue-path>.maximum-applications |
| Resource pool limit-maximum AM resource percentage              | The priority of the parameter is higher than the global variable yarn.scheduler.capacity.maximum-am-resource-percent                                                      |

| EMR parameters                 | Hadoop Yarn parameters                                                                                                             |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------|
|                                | or yarn.scheduler.capacity.<queue-path>.maximum-am-resource-percent                                                                |
| Submit permission control      | The priority of the parameter is higher than the global variable yarn.scheduler.capacity.root.<queue-path>.acl_submit_applications |
| Manage access control policies | The priority of the parameter is higher than the global variable yarn.scheduler.capacity.root.<queue-path>.acl_administer_queue    |

- **Fair Scheduler**

**Table 21-3: More settings**

| EMR parameters                          | Hadoop Yarn parameters           |
|-----------------------------------------|----------------------------------|
| ACL settings-enable ResourceManager ACL | yarn.acl.enable on the yarn-site |
| ACL settings-manage ACL                 | yarn.admin.acl on the yarn-site  |

**Table 21-4: Create Resource Pool**

| EMR parameters                             | Hadoop Yarn parameters                                       |
|--------------------------------------------|--------------------------------------------------------------|
| Enable the preemption mode                 | yarn.scheduler.fair.preemption                               |
| Resource pool name                         | Queue name in yarn                                           |
| Preemption-enable the preemption mode      | yarn.scheduler.fair.preemption                               |
| Preemption-fair share preemption threshold | yarn.scheduler.fair.preemption.cluster-utilization-threshold |

After the resource pool configuration is complete, click **Synchronize Configuration to Cluster** to make the configuration take effect.

### Disable Resource Pool

If you want to configure the resource pool directly through XML, click **Disable Resource Pool** button first in the **Resource Pool** page, and then go to the **Configuration** page of YARN.

## 22 Auto-scaling

---

### 22.1 Auto-scaling introduction

This article will introduce how to enable and disable the scaling feature.

In the following scenarios, you can save cost and improve execution efficiency through E-MapReduce scaling.

- Add computing nodes according to the time period to supplement computing capability temporarily.
- Make sure that important jobs are completed on time, and expand computing nodes according to certain cluster indicators.

**Note:**

- The scaling feature can only expand or reduce the number of task nodes.
- The scaling feature is only available in Subscribed and Pay-As-You-Go clusters.

#### Enable auto-scaling

1. Log on to the [Alibaba Cloud E-MapReduce console](#) and enter the **Cluster Management** page.
2. At the right side of the target cluster ID, click the **Manage**.
3. In the left-side navigation pane, click the **Scaling** to enter the auto-scaling page.
4. In the top right corner of the page, click the **Enable Auto-scaling** button.

If it is the first time to use the auto-scaling function with your account, you need to authorize the default role of Elastic Scaling Service(ESS) to your E-MapReduce account.

5. Click **Confirm** on the ESS authorization page.

#### Disable auto scaling

After you click the **Disable Auto-scaling** button, all task nodes expanded by auto-scaling will be released. The data stored in HDFS is located in a core node and will not be affected.

### 22.2 Configure auto-scaling according to time

If there are significant peaks and troughs in a Hadoop cluster computing capability during a certain period of time, you can set up a fixed period of time to expand a certain number of task nodes to

supplement the computing capability. This will not only ensure the completion of jobs, but also save you costs.

The expansion nodes are billed in Pay-As-You-Go mode, and the price of the same computing capability for Pay-As-You-Go mode and Subscription mode is about 3:1. So it is necessary to design the ratio of computing capability for the Pay-As-You-Go mode and Subscription mode respectively according to expansion time you need. For example, the peak period of business lasts for 8 hours a day, and the price paid by Pay-As-You-Go mode are roughly the same as that of Subscription mode. When the peak period is more than 8 hours, the Subscription mode are more favorable than the Pay-As-You-Go mode.

### **Configure scaling instance number**

- **Maximum number of nodes:** The maximum number of the nodes that can be expanded. Once it is reached, even if the auto-scaling rule is matched, the expansion and contraction will not continue. Currently, you can set up to 1,000 task nodes.
- **Minimum number of nodes:** The minimum number of the nodes that can be expanded. If the expansion or contraction number of task nodes set in the auto-scaling rule is less than the minimum number of nodes here, the cluster will scale with the minimum number of nodes at the first execution.

For example, the auto-scaling rule is set to expand 1 node at 00:00:00 of every day, but the minimum number of nodes is 3. Then the system will expand 3 nodes at the 00:00:00 of the first day.

### **Configure scaling rules**

The auto-scaling rules include expansion rules and contraction rules. When the auto-scaling function is disabled, all rules are cleared. If auto-scaling function is enabled again, the scaling rules need to be reconfigured.

\* Rule Name:

Rule names should not be repeated.

Repeat  Run Once

Specified a time

\* Retry Timeout( Seconds):  Valid range: 0 to 21600 seconds

\* Increase Task Nodes:  Valid range: 1 to 100 nodes 1 Nodes

\* Cool-down Time( Seconds):  Valid range: 0 to 86400 seconds

- **Rule Name:** In a cluster, the scaling rule names (including expansion rules and contraction rules) are not allowed to be repeated.
  - Execution cycle:
    - **Run Once:** The cluster performs a scaling operation at a specified time.
    - **Repeat:** You can choose to perform a scaling operation at a specific time every day, every week or every month.
  - **Retry Timeout:** Auto-scaling may not be performed for various reasons when the specified time is reached. With the retry expiration time set, the system will detect the condition that scaling can be performed every 30 seconds in the time range. Once the condition is met, scaling is performed. The range is 0 to 21600 seconds.
- It is assumed that the expansion operation A needs to be performed in the specified time period, if another expansion operation B is performing or the cluster is in the cooling period at that time, the operation A cannot be performed. During the retry expiration time you set, the system will detect the condition that operation A can be performed every 30 seconds. Once the conditions are met, the cluster will immediately perform scaling.
- **Increase Task Nodes:** The number of task nodes to be increased or decreased each time by cluster when the rule is triggered.
  - **Cool-down Time:** The interval between scaling operation is completed and the same operation can be performed again. No scaling operation will be performed during cooling time.

## Configure scaling instance specification

You can specify the hardware specifications of the scaling nodes. It can only be configured when the auto-scaling function is enabled, and can not be modified after the configuration is saved. If it needs to be modified for a special case, you can disable the auto-scaling function and enable it again.

- When you select specifications for vCPU and memory, the system automatically matches the instances that meet the criteria based on your selection and displays them in the instance list below. You need to add an optional instance to the list on the right so that the cluster can scale according to the selected instance specification.
- To avoid scaling failures due to insufficient ECS stock, you can choose up to 3 ECS instance types.
- Whether you choose an efficient cloud disk or a SSD cloud disk, the data disk is set to a minimum of 40G.

## 22.3 Auto-scaling preemptible instances

E-MapReduce *Preemptible instances* are suitable for scenarios where there is no requirement for the successful execution of big data jobs and where the price of computing resources is important. You can purchase preemptible instances to increase the computing resources of clusters using auto scaling.

### Enable auto scaling

To enable the auto scaling feature and set scaling rules, follow these steps:

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. Click **Cluster Management**.
3. Find the cluster where you want to add preemptible instances and the click **Manage**.
4. In the left-side navigation pane, click **Scaling**.
5. Click **Enable Auto Scaling**.
6. Configure scaling rules. For more information, see [Configure auto scaling according to time](#).
7. In scaling configuration area, select **Preemptible instance**.

### Configure preemptible instances



**Note:**

The price of preemptible instances is more favorable than Pay-As-You-Go instances, but Alibaba Cloud may release your instances at any time based on changes in demand resources or market transaction prices.

To configure your preemptible instance, follow these steps:

1. Select the vCPU and memory for your instance.
2. Select instance types. You can select up to three instance types. E-MapReduce filters out all other instance types to ensure that you purchase a preemptible instance that meets your requirements.
3. After you select instance types, click the maximum price of each instance type, and then click **OK**. The instance types will appear in the selected instances list. If you want to modify the price of a selected instance type, select the instance type in the selectable instances list and change the price (by hour). Your instance will run when your bid is higher than the current market price. Your final instance type will be billed at the market price.
4. The system disk is used for deploying basic services such as the OS and EMR, which are set by default. You can set the data disk type and size according to your needs.
5. The final configuration price includes the maximum bid price, system disk price and data disk price. Click **Save**.

For more information about preemptible instances, see [FAQ about preemptible instances](#).

## 22.4 Auto-scaling records

After the auto-scaling operation is complete, you can click the **Scaling Records** tab on the top of the **Scaling** page to see the records of auto-scaling operation and the nodes number after auto-scaling operation is completed.

The execution status of auto-scaling includes the following four types:

- **Running**: Auto-scaling operation is being implemented.
- **Success**: All specified nodes involved in the scaling rule are added or removed from the cluster.
- **Partial success**: According to scaling rules, some nodes were successfully added or removed from the cluster, but some were failed due to disk quota or ECS inventory.
- **Failure**: According to scaling rules, no node is added or removed from the cluster.

## 23 Workflow development

---

### 23.1 Workflow project management

After creating an E-MapReduce cluster, you can create workflow projects so that multiple jobs can be run simultaneously or sequentially.

#### Create a project

1. At the top of the page, click the **Data Platform** tab on the top to enter the **Projects** page.

Under the master account, you can view projects of itself and its all sub-accounts. You can only view projects with development permission in the sub-account. To authorize project development permission, you need to configure it in the primary account. For more information about authorization, see [User Management](#).

2. In the upper right corner, click the **New Project** button to see the **New Project** dialog box.
3. Enter the project name and description, and click **Create**.

**Note:**

You can only create a project with the master account, that is, the **Create Project** button is only visible to the master account administrator.

#### User management

After creating a new project, you can authorize the RAM sub-account operational permission of the project.

1. In the **Project List** page, click the **View Details** link in the **Actions** column.
2. Click the **User Management** tab.
3. Click the **Add User** button to add RAM sub-accounts under the master account to the project.

The added sub-accounts will be a member of the project and can be able to view and develop the jobs and workflows under the project. If you don't want to set the sub-account to be the selected project member any more, click **Delete** in the **Actions** column.

**Note:**

You can only add project members with the primary account, that is, the **User Management** tab is only visible to the primary account administrator.

## Associate clusters

After creating a new project, you need to associate a cluster for the project so that the workflow in the project can run on it.

1. In the **Projects** page, click the **View Details** link in the **Actions** column.
2. Click the **Cluster Settings** tab.
3. Click the **Add Cluster** button, from the drop-down menu, you can select the created Subscription and Pay-As-You-Go clusters (clusters created by running temporary jobs are not listed here).
4. Click **OK**.

You can click **Delete** in the Operation column to unassociate the cluster.



### Note:

You can only associate cluster with the primary account , that is, the **Cluster Settings** tab is only visible to the primary account administrator.

Click **Modify Configuration** in the Operation column to set up the queue and user to submit jobs to the cluster. The specific configuration items are described as follows:

- **Default Submit Job User:** Sets the default Hadoop user who submits the job to the selected cluster in the project. The default value is hadoop, and there only can be one default user.
- **Default Submit Job Queue:** Sets the default queue that the jobs are submitted to in the project. If you leave this blank, the job will be submitted to the default queue.
- **Submit Job User Whitelist:** Sets Hadoop users who can submit jobs to the selected cluster in the project. If there are more than one users, they can be separated by a half-width comma (,).
- **Submit Job Queue Whitelist:** Sets the queue of the selected cluster that jobs in the project can run in. If there are more than one queues, they can be separated by a half-width comma (,).
- **Client whitelist:** Configures the client that you can submit jobs. The E-MapReduce master node or the E-MapReduce Gateway can be selected. Currently, your self-built gateways are not listed here.

## 23.2 Job editing

In the project, you can create jobs such as Shell, Hive, Spark, SparkSQL, MapReduce, Sqoop, Pig and Spark Streaming.

### Create a Job

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. Click the **Data Platform** tab on the top to enter the **Projects** page.
3. Click **Design Workflow** of the specified project in the **Actions** column.
4. On the left side of the job editing page, right-click on the folder you want to operate and select **New Job**.
5. In the **New Job** dialog box, enter the job name, job description, and select the job type.

Once the job type is selected, it cannot be modified.

6. Click **OK**.

**Note:**

You can also create subfolder, rename folder, and delete folder by right-clicking on the folder.

### Develop jobs

For the development guides for various types of jobs, see [Job](#) in the *Cite LeftE-MapReduce User GuideCite Right*.

**Note:**

When you insert an OSS path, if you select the OSSREF file prefix, the OSS file will be downloaded to the cluster and added to the classpath.

- Basic settings

Click the **Job Settings** in the upper right corner of the page to enter the **Job Running Configuration** page.

- **Number of Retries:** Sets the number of retries when this job fails during the workflow running. This option will not take effect when you run the job directly on the **Job Editing** page.
- **Failure Policy:** Sets whether to continue running the next job or suspend the current workflow when this job fails during the workflow running.

- **Resource File:** If you add resources such as Jar packages or UDFs that the job running depends on, you need to upload the resources to OSS first. After adding the resource, you can reference the resource directly in the job code.
- **Parameter Configuration:** Specify the value of the variable referenced in the job code. You can reference variables in the code with the format `${variable name}`. Click the plus icon on the right to add key and value, the key is the variable name, and the value is the value of the variable. In addition, you can customize the time variable according to the schedule time. The rules are as follows:

- yyyy indicates the year which is 4-digit.
- MM indicates the month.
- dd indicates the day.
- hh24 indicates the hour, uses \*hh\* if 12-hour system is adopted.
- mm indicates the minute.
- ss indicates the second.

The time variable can be any combination of time containing yyyy. You can also use the '+' symbol to advance time and use '-' symbol to delay time. For example, the variable `${yyyy-MM-dd}` indicates the current date, then:

- The representation of 1 year later: `${yyyy+Ny}` or `${yyyy-MM-dd hh:mm:ss+1y}`.
- The representation of 3 months later: `${yyyyMM+Nm}` or `${hh:mm:ss yyyy-MM-dd +3m}`.
- The representation of 5 days before: `${yyyyMMdd-Nd}` or `${hh:mm:ss yyyy-MM-dd-5d}`.

- **Advanced settings**

On the **Job Settings** page, click the **Advanced** tab.

- **Mode:** Job running modes, including YARN and LOCAL. In YARN mode, the job is submitted on the YARN by the Launcher. In LOCAL mode, jobs run directly on the assigned host.
- **Environment Variables:** Add environment variables for job running, or export environment variables directly in the job script.
- **Scheduling Parameters:** Sets job running configuration such as the YARN queue, CPU, memory and Hadoop users. You can leave it unset, and the default value of the Hadoop cluster is adopted when the job is running.

## Run job

Once the job has been developed and configured, you can click the **Run** button at the top right corner to run the job.

## View log

After the job runs, you can view the running log of the job in the **View Records** tab at the bottom of the page. Click **Workflow** to jump to the detailed log page of the job, you can see information such as the job's submitting log and YARN Container log.

## 23.3 Ad hoc queries

You can only select HiveSQL, SparkSQL, and Shell as the type of an ad hoc query. When you execute an ad hoc query statement, the log and query results show at the bottom of the log and query page.

### Create a job

When you execute a job on the Edit Jobs page and click Details, you will be directed to the Details page that shows the operation logs and run logs of this job. Ad hoc queries and jobs are used in different places. Ad hoc queries are usually used by data scientists and data analysts. In addition, you need to use SQL as a tool to implement an ad hoc query.

1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. Click the **Data Platform** tab to enter the **Projects** page.
3. On the right side of the associated project, click **Design Workflow** to enter the **Edit Jobs** page.
4. In the left-side navigation pane, click the **Query** tab to enter the **Query** page.
5. In the left-side navigation pane, right-click a folder as required and select **New Job**.
6. In the **New Job** dialog box, enter the job name and job description, and select a job type.

The job type cannot be modified once the job has been created.

7. Click **OK**.



#### Note:

You can right click on a folder and then select the corresponding option to perform New Subfolders, Rename Folder, and Delete Folder operations.

## Develop a job

For more information about how to develop jobs with HiveSQL, SparkSQL, and Shell types, see the *Cite LeftjobsCite Right* section of E-MapReduce user guide.



### Note:

When you insert an OSS UNI and select OSSREF as a File Prefix, E-MapReduce will download OSS files to your cluster and add these files to the classpath.

- Basic job settings

In the top-right corner, click **Configure Jobs**, and then the **Job Settings** dialog box appears.

- Resource File: If you want to add resources such as jar packages or UDF that a job execution depends on, you must upload these files to OSS. When you select a resource, you can use this resource in a job directly.
- Parameter Configuration: specifies the values of variables used in a job. You can use variables in your code. The format is:  $\${variable\ name}$ . Click the plus (+) icon on the right side to add key-value pairs. Key is the name of a variable and value is the value of a variable. In addition, you can follow these rules to customize time variables according to the start time of a schedule.
  - yyyy represents a 4-digit year.
  - MM represents a 2-digit month.
  - dd represents a 2-digit day.
  - hh24 indicates that the 24-hour clock is used. hh indicates that the 12-hour clock is used.
  - mm represent a 2-digit minute.
  - ss represents a 2-digit second.

A time variable consists of the combination of a 4-digit year and one or more other time formats. In addition, you can use plus (+) and minus (-) to add or reduce a period of time for the current time. For example, the  $\${yyyy-MM-dd}$  variable represents the current date.

- One year after the current date can be represented as  $\${yyyy+Ny}$  or  $\${yyyy-MM-dd\ hh:mm:ss+1y}$ .
- Three months after the current date can be represented as:  $\${yyyyMM+Nm}$  or  $\${hh:mm:ss\ yyyy-MM-dd+3m}$ .

- Five days before the current date can be represented as:  $\${yyyyMMdd-Nd}$  or  $\${hh:mm:ss yyyy-MM-dd-5d}$ .
- Advanced job settings

In the **Job Settings** dialog box, click the **Advanced** tab.

- Mode: includes the YARN and LOCAL modes. YARN: Jobs are submitted by allocating resources on YARN through Launcher. LOCAL: The job runs on a specified local host.
- Scheduling parameters: Includes information, such as YARN queues of a job, vCPU, memory, and Hadoop user. If you do not specify these parameters, a job uses the default values of the Hadoop cluster.

### Execute a job

After you develop and configure a job, in the top-right corner, you can click the **Run** button to execute a job.

### View logs

After you execute a job, you can view run logs on the **Log** tab at the bottom of the query page.

## 23.4 Workflow management

You can run big data jobs parallelly in a DAG manner by E-MapReduce workflows. In addition, you can suspend, stop, rerun workflows, and view the running status of workflows in WebUI.

### Creating a workflow

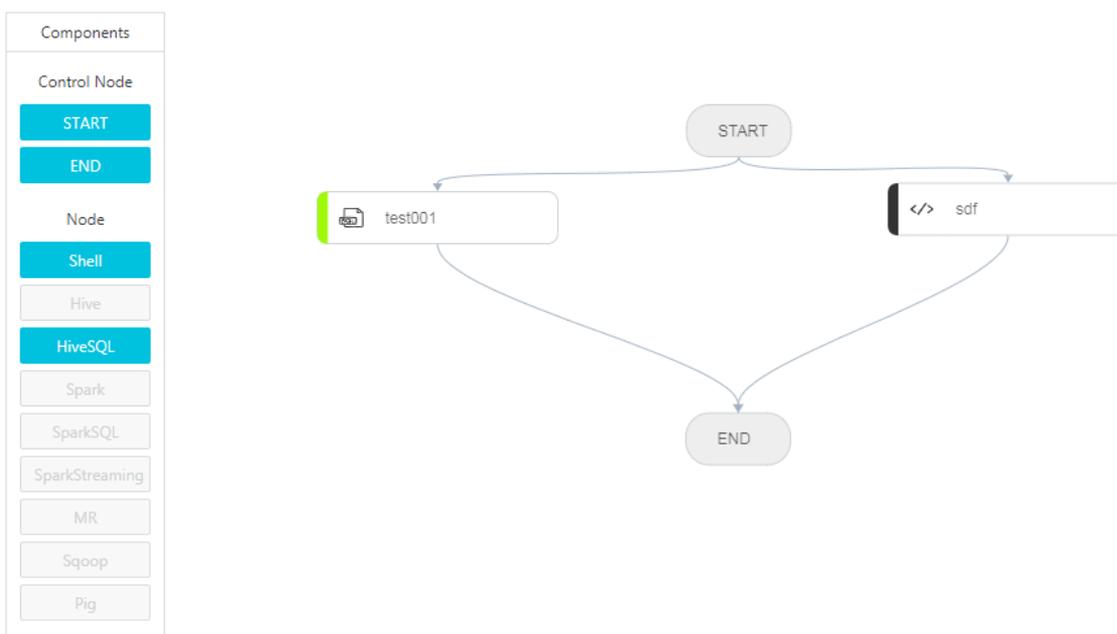
1. Log on to the [Alibaba Cloud E-MapReduce console](#).
2. At the top of the page, click the **Data Platform** tab.
3. Click **Design Workflow** of the specified project in the **Actions** column, and then select the **Design Workflow** tab.
4. On the left side, right-click on the folder you want to operate and select **New Workflow**.
5. In the **New Workflow** dialog box, enter the workflow name, workflow description, and select the E-MapReduce cluster where the workflow is to run.

You can select a Subscription and Pay-As-You-Go E-MapReduce cluster that has been created and is associated with the project for the workflow running, or a new temporary cluster can be created through the cluster template to run the workflow.

6. Click **OK**.

## Editing workflow

You can drag different types of jobs to the workflow editing canvas, and specify the order of job instances by curve. After the jobs needed to run has been dragged, drag the **END** component from the control node area to the canvas, which indicates that the entire workflow is complete.



## Configure workflow

On the right side of the **Workflow Design** page, click **Configure** button to configure the workflow scheduling.

- **Run In:** The E-MapReduce cluster where the workflow is to run can be modified.
- **Scheduling Policy:** After workflow scheduling is enabled, you can choose a scheduling policy, including time scheduling and dependency scheduling.
  - **Time Scheduler:** Sets the start time and end time of the workflow scheduling, the system will run the workflow according to the schedule you set during the time scope.
  - **Dependency:** From the selected project, select the dependency workflow of the current workflow. After the dependency workflow is completed, the current workflow will be scheduled to run. Currently, only one workflow can be selected.

## Run a workflow

Once the workflow is designed and configured, you can run the workflow by clicking the **Run** button in the upper right corner.

## View and operate workflow instances

After the workflow is running, click the **View Records** tab on the left to view the running status of the workflow instance. Click the **View Details** of the workflow instance to view the running status of the job instance. You can also suspend, resume, stop, and rerun the workflow instance.

Home Page > Data Platform > Project (FP-19DF27ECF61E5AF1) > View Records > Workflow Instance (FI-EC6641ECF29D5E2F) > View Details

Cluster Information Chart

|                                 |                               |
|---------------------------------|-------------------------------|
| ID: FI-EC6641ECF29D5E2F         | Name: test                    |
| Workflow ID: F-798F6400F5485142 | Run In: C-555D28C90826D188    |
| Status: <b>FAILED</b>           | Run Time: 11 Seconds          |
| Start Time: 2018-11-12 17:08:42 | End Time: 2018-11-12 17:08:53 |

Name

| Job Instance ID      | Name    | Run In             | Type     | Host                       | Start Time          | Completed At        | Run Time   | Execution State | Actions                      |
|----------------------|---------|--------------------|----------|----------------------------|---------------------|---------------------|------------|-----------------|------------------------------|
| FNI-7C0AF5769E6802FB | test001 | C-555D28C90826D188 | HIVE_SQL | emr-header-1.cluster-84010 | 2018-11-12 17:08:43 | 2018-11-12 17:08:53 | 10 Seconds | <b>FAILED</b>   | <a href="#">View Details</a> |
| FNI-05CD626FF3B17CC9 | sdf     | C-555D28C90826D188 | SHELL    | emr-header-1.cluster-84010 | 2018-11-12 17:08:44 | 2018-11-12 17:08:53 | 9 Seconds  | <b>OK</b>       | <a href="#">View Details</a> |

- Suspend workflow: The running job instance will continue to run, but the subsequent job instances will not. You can click **Resume Workflow** and the system will continue to run the subsequent jobs after the job instance is suspended.
- Stop workflow: All running job instances stop immediately.
- Rerun workflow Instance: The system will run the workflow from the start component.