

Alibaba Cloud E-MapReduce

User Guide

Issue: 20181016

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 cluster.....	4
2.1 Instance types.....	4
2.2 Gateway instances.....	5
2.3 ECS instance.....	5
2.4 Storage Guide.....	6
2.5 D1 Support.....	8
2.6 Access between classic network and VPC.....	9
3 Cluster.....	11
3.1 Create a cluster.....	11
3.2 Expand a cluster.....	16
3.3 Release a cluster.....	16
3.4 Cluster list page.....	17
3.5 Cluster details page.....	18
3.6 Service list.....	20
3.7 Cluster script.....	20
3.8 Cluster renewal.....	22
3.9 Security group.....	23
3.10 Quick portal for components.....	24
3.11 Node upgrade.....	25
3.12 Disk resizing.....	26
3.13 Convert Payment Methods.....	28
4 Job.....	29
4.1 Hadoop MapReduce Job Configuration.....	29
4.2 Hive job configuration.....	30
4.3 Pig job configuration.....	31
4.4 Spark job configuration.....	33
4.5 Spark SQL job configuration.....	35
4.6 Shell job configuration.....	36
4.7 Sqoop job configuration.....	37
4.8 Job operations.....	37
4.9 Job date variables.....	38
5 Execution plan.....	40
5.1 Create an execution plan.....	40
5.2 Manage execution plans.....	42
5.3 Execution plan list.....	43
5.4 View job results and logs.....	44

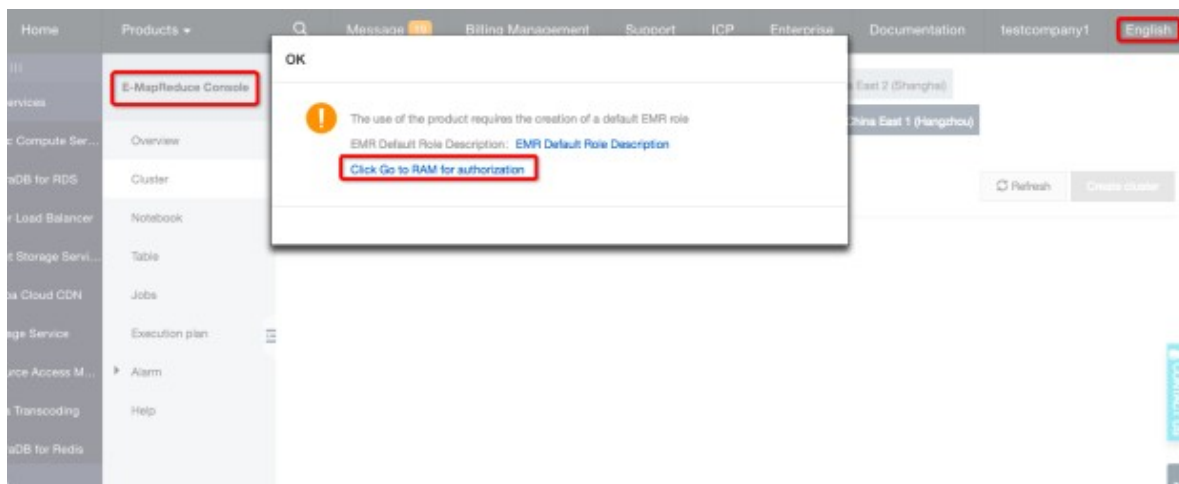
5.5 Parallel execution of multiple execution plans.....	46
6 Alert management.....	47
6.1 Cluster alarm management.....	47
6.2 Job Alarm Management.....	49
7 Software configuration.....	50
8 Bootstrap action.....	55
9 VPC.....	58
10 Python instructions.....	60
11 Open source assembly.....	61
11.1 Hue.....	61
11.2 Oozie.....	62
11.3 Presto.....	65
11.4 Zeppelin.....	66
11.5 ZooKeeper.....	66
11.6 Kafka.....	67
11.6.1 Quick start with Kafka.....	67
11.6.2 Cross-cluster access to Kafka.....	69
11.6.3 Kafka Ranger.....	72
11.6.4 Kafka SSL.....	75
11.6.5 Kafka Manager guide.....	76
11.6.6 Kafka Common Problems.....	78
11.7 Druid.....	79
11.7.1 Druid Introduction.....	79
11.7.2 Quick start.....	82
11.8 Presto.....	92
11.8.1 Connector.....	92
11.8.2 What is Presto.....	99
11.8.3 Quick start with Presto.....	101
11.8.4 Data type.....	109
11.8.5 Common functions and operators.....	113
11.8.6 SQL statement.....	138
11.8.7 Technical support.....	173
11.9 Knox guide.....	173
12 Table mangement.....	177

1 Role authorization

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

Role authorization process

1. When you create a cluster or an on-demand execution plan, if no default role is authorized correctly to the E-MapReduce service account, the prompt is displayed. Click **Go to RAM for authorization** to perform role authorization.



2. You are directed to RAM's authorization page. Click **Confirm Authorization Policy** to authorize the default role AliyunE-MapReduceDefaultRole to 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

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

Default role permissions

The permissions of default role, AliyunEMRDefaultRole, include the following:

- ECS related permissions:

Permission name (Action)	Permission 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 public network IP address.
ecs:DeleteInstance	Delete machine instances.
ecs:StartInstance	Start machine instances.

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

- OSS related permissions

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

2 Configure cluster

2.1 Instance types

The EMR cluster consists of multiple different node instance types, namely the master, core, and task instances. Completely different service processes are available for different tasks when each of these instances is deployed. For example, we deploy Hadoop HDFS's Name Node service and Hadoop YARN's Resource Manager service on master instances, and Data Node service and Hadoop YARN's Node Manager service on core instances. Task instances are only used for computing. Therefore, we deploy Hadoop YARN's Node Manager service, rather than HDFS-related services for task instances.

When creating a cluster, you must determine the ECS specifications for these three instance types. The ECS instances with the same instance type must be in the same instance group. The cluster can scale up at a later stage to accommodate the number of hosts in the appropriate instance groups (except master instance group).

**Note:**

The task instance is supported in version 3.2.0 or later.

Master instance

The master instance is the node where the cluster service's management and control components are deployed. For example, the Hadoop YARN's Resource Manager is deployed on the master instance node. You can connect to the master instance using SSH, and check the service status in the cluster by the Web UI of the software. At the same time, when you want to quickly test or run a job, you can log on to the master instance and submit jobs directly by command line. When the high availability feature is turned on for the cluster, two master instance nodes are used (by default, only one).

Core instance

The core instance is the instance node managed by the master instance. It runs the Hadoop HDFS's Data Node service and stores all the data. It also deploys computing services, such as Hadoop YARN's Node Manager service, to perform computing tasks. To meet the needs for more data storage or heavier computing workload, the core instance can scale up at any time without affecting the normal operations of the active cluster. It can use a variety of different storage media to store data. You can refer to the discussions about disks for details.

Task instance

The task instance is an optional instance type that is specifically responsible for computing. If the core instance has sufficient computing power, task instance may not be used. The task instance can quickly add computing power to the cluster, such as Hadoop's MapReduce tasks, and Spark executors. As HDFS data is not stored on the task instance, Hadoop HDFS's Data Node service is not run on it. The task instance can scale up and down at any time without affecting the normal operations of the active cluster. Depending on the fault tolerance (or retries) of the computing service, fewer task instance nodes may cause MapReduce and Spark jobs to fail.

2.2 Gateway instances

Gateway is an independent cluster consisting of multiple nodes with same configuration generally.

When creating a Gateway cluster, you can associate an existing Hadoop cluster on which Hadoop (HDFS+YARN), Hive, Spark, Sqoop, Pig, and other clients have been deployed to facilitate to operate the cluster. It is an independent submission point and does not take up the resources of the cluster, especially you submit jobs on the Master node, which can improve the stability of the Master node. If you have too many jobs to submit, you can add nodes for the cluster dynamically.

You can also create multiple Gateway clusters for different users, allowing them to use their own environment to meet different business needs.

2.3 ECS instance

ECS instance

ECS instance types available for EMR

- General-purpose type
vCPU/Memory ratio is 1:4, for example, 32 core and 128 GB. This type uses cloud disks as storage.
- Compute type
vCPU/Memory ratio is 1:2, for example, 32 core and 64 GB. This type uses cloud disks as storage and provides more computing resources.
- Memory type
vCPU/Memory ratio is 1:8, for example, 32 core and 256 GB. This type uses cloud disks as storage and provides more memory resources.
- Big data type

Utilizing local SATA disks as a highly cost-effective data storage solution, it is a recommended ECS instance type applicable for use cases involving mass data volumes (TB-level).

- Ephemeral SSD type

Utilizing ephemeral SSDs, the ECS instance type has a high local IOPS and throughput.

Overall storage

- Shared type (entry level)

The ECS instance type with shared CPUs is not stable enough for use cases involving massive computing volumes. It is applicable for entry-level learning users rather than enterprise customers.

- GPU

It is a heterogeneous GPU-based ECS instance type applicable for machine learning use cases.

Use cases applicable for the ECS instance types

- Master instances

General-purpose or memory types are applicable for master instances, where data is directly stored on Alibaba Cloud's cloud disks. It has three backups, which guarantees high data reliability.

- Core instances

General-purpose, compute, and memory types are applicable for small data volume use cases (below TB level) or when OSS is used for primary data storage. When data volume is large (10 TB or more), we recommend that you use the big data type for great cost-effectiveness. When utilizing ephemeral disks, data reliability is challenged, but it can be maintained and guaranteed by the EMR platform.

- Task instances

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

2.4 Storage Guide

Two types of disks on a node exist: one is the system disk which is used to install operating systems, the other is the data disk which is used for data storage. A node generally 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 piece of data disk can have different configurations, including different disk types and capacities. SSD cloud disks are defaulted in EMR as the cluster's system disks. Four cloud disks are used in EMR by default. Considering current intranet bandwidth, the default configuration is reasonable.

Cloud and ephemeral disks

Two types of disks are available for data storage.

- Cloud disks

Include SSD, ultra, and basic cloud disks.

Rather than being directly attached to a local computing node, cloud disks have access to a remote storage node by the network. Each piece of data has two real-time backups in the backend, thus three identical copies in total. When one of the copies is corrupted (due to disk damage, rather than damages arising from business), your backup data is automatically used for recovery.

- Ephemeral disks

Include 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 better performance than cloud disks. You cannot select the number of ephemeral disks and must keep the default configurations. Similar to offline physical hosts, no data backup is in the backend, and upper-level software to guarantee data reliability is required.

Applicable use cases

In EMR, when the hosting node is released, data in all cloud and ephemeral disks is cleared. The disks cannot be kept independently or re-used. Hadoop HDFS uses all data disks for data storage. Hadoop YARN also uses all data disks as on-demand data storage for computing.

When your business does not involve large data volume (below TB level), cloud disks can be used as the IOPS and throughput are smaller than local disks. In case of large data volumes, we recommend that you use local disks whose data reliability is guaranteed by EMR. If you encounter apparently insufficient throughput, you can switch to ephemeral disks.

OSS

OSS can be used as HDFS in EMR. You can have easy read and write access to OSS. All codes using HDFS can also be simply edited to access data on OSS.

For example:

Reading data from spark

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

Replace storage type HDFS-> OSS

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

The same is true for Mr or hive jobs.

HDFS commands directly handle OSS data

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

In this process, you do not need to enter AK and endpoint, EMR will automatically complete the user's information using the current cluster owner.

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

2.5 D1 Support

To meet the storage needs in the big data-type use case, Alibaba Cloud launched an instance series using ephemeral disks in the cloud: the D1 series. The D1 series uses ephemeral disks instead of cloud disks for data storage. This solves the high cost problem caused by keeping multiple copies of redundant data in the cloud disks. In addition, as no data must be transferred by the network, the disk throughput is improved. Furthermore, the series can also take advantage of Hadoop's proximity computing.

Compared with cloud disks, the series greatly enhances storage performance and reduces storage prices, reaching nearly the same cost as offline physical hosts.

Along with numerous advantages of using ephemeral disks, the problem of data reliability occurs . For cloud disks, because of Alibaba Cloud's default multi-disk backup policy, you do not need consider disk damages. Cloud disks automatically guarantee data reliability. However, when you use ephemeral disks, such guarantee requires upper-level software. If disk and node failures appear, manual operations and maintenance must be performed.

The EMR + D1 solution

A complete set of automated maintenance solution like D1 is available in EMR for ephemeral disks. This allows Alibaba Cloud users to utilize the instances using ephemeral disks without considering the entire maintenance process, as data reliability and service availability are guaranteed.

Highlights are as follow:

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

With automated maintenance of the entire back-end management and control system, EMR helps you make better use of ephemeral disks, and develop cost-effective big data system.



Note:

If you want to set up a Hadoop cluster using the D1 series, open a ticket so we may assist in your operations.

2.6 Access between classic network and VPC

Currently, Alibaba Cloud provides two types of cloud networks: classic network and VPC. Many users' service systems are still using classic networks, while EMR clusters are using VPCs. This section describes how to enable inter-access between ECS on classic networks and EMR clusters on VPC networks.

ClassicLink

To solve this problem, Alibaba Cloud launches the [ClassicLink Solution](#).

To solve this problem, Alibaba Cloud launches the ClassLink Solution.

1. Create a vSwitch according to the CIDR block specified in the ClassLink Solution.
2. When creating a cluster, use the vSwitch for the CIDR block to deploy the cluster.
3. Connect the corresponding classic network node to VPC in the ECS console.
4. Set security group rules.

This is how an inter-access between ECS on a classic network and EMR cluster on a VPC network is realized.

3 Cluster

3.1 Create a cluster

Create a cluster

Enter the cluster creation page

1. Log on to [Alibaba Cloud E-MapReduce Console Cluster List](#).
2. Complete RAM authorization. For procedure, see [Role authorization](#).
3. Select a region for the cluster. The region cannot be changed once the cluster is created.
4. Click **Create a cluster** in the upper right corner.

Cluster creation process

**Note:**

Except the name, clusters cannot be modified after creation. Carefully confirm the required configuration.

To create a cluster, complete three following steps:

1. Software configuration

Configuration description:

- **Product version:** The main version of E-MapReduce represents a complete open source software environment and can be upgraded regularly based on the upgrade of internal component software. If the software related to Hadoop is upgraded, the main version of E-MapReduce is also upgraded. Earlier version clusters cannot be upgraded to a later version.
- **Cluster type:** Currently E-MapReduce provides the following cluster types:
 - Hadoop, and Standard Hadoop clusters that contain most of the Hadoop-related components. Details about the components are provided in the selection list.
 - Kafka, and Independent Kafka clusters that provide messaging service.
- **Inclusion configuration:** Displays a list of all software components under the selected cluster type, including the name and version number. You can select different components as required. The selected components start relevant service processes by default.

**Note:**

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

- **Security mode:** Whether to enable the Kerberos authentication function of the cluster.
- **Software configuration (optional):** Hadoop, Spark, Hive, and other basic software in the cluster can be configured. For more information, see [Software configuration](#).

2. Hardware configuration

Configuration description:

- **Billing configuration:**
 - **Billing method:** The billing method is consistent with ECS. Both Subscription and Pay-As-You-Go modes are supported. If Subscription mode is selected, you must select the duration. It is applicable to short-term testing or flexible dynamic tasks. The payment is relatively high.
 - **Purchase duration:** You can select 1, 2, 3, 6, or 9 months, or 1, 2, or 3 years.
- **Cluster network configuration**
 - **Availability zone of the cluster:** Select the zone where the cluster is to be located. Different zones can have different machine types and disks. There are several availability zones in each region. The availability zone belongs to different physical areas. If better network connectivity is required, we recommend that you select the same availability zone. However, the risk of cluster creation failure increases as the storage of a single availability zone may be insufficient. If you need a large number of nodes, open a ticket to consult with us.
 - **Network type:** Classic network and Virtual Private Cloud (VPC) can be selected. VPC requires an additional subordinate VPC and subnet (VSwitch). Click Create VPC/subnet (VSwitch) on the page to enter the [VPC console](#). Then refresh the list to see the created VPC/subnet (VSwitch). For more information about E-MapReduce VPC, see [VPC](#).

Classic network and VPC are not interoperable. The network type cannot be changed after purchase.
 - **ECS instance series:** In terms of ECS instance series, different zones have different instance series (such as I, II, III). We recommend you use the latest generation.
 - **VPC:** Select the region of VPC network.
 - **VSwitch:** Select a zone for VSwitch under the corresponding VPC. If no VSwitch is available in this Zone, then you must create a new one.

- New security group: No security group exists when you create the cluster for the first time . Click New security group, and input the new security group name.
- Main security group: The security group that the cluster belongs to. Only the security group created by the user in E-MapReduce product is displayed. The security group cannot be created outside the E-MapReduce. To create a security group, select New security group and input the security group name. The name can contain Chinese characters, English letters, numbers, hyphens (-), and underscores (_), with a length limit between 2-64 characters.
- **Cluster node configuration**
 - High availability cluster: After opening, Hadoop cluster has two masters to support the high availability of Resource Manager and Name Node. Originally, HBase cluster supports the high availability, but another node serves as a core node. If the high availability is activated, an independent master node is used to support it, which is more secure and reliable. The default mode is non-high availability, and only one master node exists.
 - Node type:
 - Master, the master instance node is mainly responsible for the deployment of control processes such as Resource Manager and Name Node.
 - Core, the core instance node is mainly responsible for the storage of all data in the cluster, and can be scaled up as needed.
 - Task, the computing node, does not store data, and is used to adjust the computing capacity of the cluster.
 - Node configuration: Select different types of nodes. Different types of nodes have different application scenarios. You can select one type based on requirements.
 - Data disk type: The data disks used by a cluster node are ordinary cloud disks, high-efficiency cloud disks, and SSD cloud disks which may vary with machine type and region. When the user selects different regions, disks that are supported by the regions are displayed in the drop-down list. The data disk is set to release with the cluster release by default. The ephemeral disk type is set by default and cannot be changed.
 - Data disk volume: The recommended minimum cluster volume of 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: The quantity of instances of all required nodes. A cluster requires at least three instances (the high availability cluster requires at least four instances, adding one master node). The maximum is 50. If more than 50 instances are required, contact us by opening a ticket. While a monthly subscribed cluster can provide 100 at most. If you need more than 50 nodes, open a ticket to consult with us.

3. Basic configuration

Configuration description:

- **Basic information**

Cluster name: The cluster name can contain Chinese characters, English letters (uppercase and lowercase), numbers, hyphens (-), and underscores (_), with a length limit between 1-64 characters.

- **Operation log**

- Operation log: The function for saving the operation log is turned on by default. In the default state, you can select the OSS directory location to save the operation log. You must activate OSS before using this function. 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 the log.
- Central metadatabase: 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 setting**

- Service role: You can authorize E-MapReduce with this role to use other Alibaba Cloud services like ECS and OSS.
- ECS application 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 the access. The AccessKey permission is controlled by this role.

- **Logon setting**

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 limit between 8-30 characters.

- **Bootstrap action (optional):** You can run the customized script before Hadoop is enabled in the cluster. For more information, see [Bootstrap action](#).

Purchase list and cluster cost

Your configuration list and cluster cost is shown on the right side of the page. The presented price information varies with the type of payment. For Subscription cluster, the total expense is shown. For Pay-As-You-Go cluster, hourly expense is shown.

Confirm creation

After all valid information is inputted, the **Create button** is highlighted. Verify the information, and click **Create** to create clusters.



Note:

- If it is a Pay-As-You-Go cluster, the cluster is created immediately, and you are taken back to the Cluster List page where you can see a cluster in **Cluster Creation** status. It can take several minutes to create the cluster. After creation, the cluster is switched to Idle **Idle** status.
- For Subscription cluster, the cluster is not created until the order is generated and paid.

Log on to the Core Node

To log on to the Core node, take the following steps:

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

```
su hadoop
```

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

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

3. Get root permissions through the sudo command.

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

Creation failed

If the cluster creation fails, the cluster list page displays a message: **Cluster creation failed**. To see the reason for failure, place the cursor over the red exclamation point.

No handling is required because the corresponding computing resources are not created. The cluster is automatically hidden after three days.

3.2 Expand a cluster

If your cluster resources (computing and storage resources) are insufficient, the cluster can be expanded horizontally. Only the core node can be expanded. The hardware configuration defaults to be consistent with the ECS purchased previously.

Expansion entry

Select the clusters to be expanded on the cluster list page. Click **Adjust scale** to enter the cluster expansion page. You can also click **View details** to enter the cluster details page, and click **Adjust cluster scale** at the core node information.

Expansion interface

**Note:**

Only expansion is supported. Reduction is not supported.

- Quantity of current master nodes: By default, the quantity is 1 and cannot be adjusted.
- Quantity of current cores: Displays the quantity of all your current core nodes.
- Increase the quantity of cores: Enter the quantity that you want to add. The total cost of the expanded cluster is displayed on the right side. Click Confirm to expand. For safety reasons, we strongly recommend that you expand small batches consisting of one or two units, rather than expanding a large quantity each time.

Expansion status

To view the cluster expansion status, go to the core node information on the details page. The node being expanded is displayed as **Expanding**. When the status of this ECS changes to **Normal**, the ECS has been added into the cluster and can provide services normally.

3.3 Release a cluster

On the cluster list page, click Release on the right side of the cluster entry to release the cluster.

Only Pay-As-You-Go clusters in the following statuses can be released:

- Creating
- Running
- Idle

Common release

You are prompted to confirm before the release. Once confirmed, procedure is as follows:

1. All jobs in the cluster are forcibly terminated.
2. If you have selected to save the log to OSS, all current job logs are saved to OSS. The required time depends on the log size. The log uploading and the job running are in parallel. The log can be uploaded once generated. Therefore, when the final job is terminated, a few logs must be uploaded. The upload can take several minutes.
3. Release the cluster. This process depends on the size of the cluster, the smaller the cluster will be faster. Normally clusters are released in seconds, up to five minutes. ECS clusters aren't charged until they are released.

**Warning:**

If you want to save money and control the release before the whole point, set aside a certain amount of release time to ensure that it is released before the whole point.

4. End of cluster release.

Forcible release

If you no longer need the logs, and want to immediately terminate the cluster operation, activate the forcible release function. The process of log collection is skipped and release start directly.

Cluster release failure

Due to system error or other causes, the cluster release may fail after confirmation. If the cluster release fails, E-MapReduce can start background protection to automatically release the cluster again until it is released successfully.

3.4 Cluster list page

Cluster list page displays the basic information about all of your clusters.

All items from the list are described as follows:

- **Instance ID/cluster name:** ID and name of the cluster. Place the cursor over the cluster name to modify it.
- **Cluster type:** Hadoop is the only cluster type available.
- **Elapsed time:** Operation time from creation until now. Once the cluster is released, and the timing is terminated.
- **Status (default):** The cluster status, for more information, see [Cluster status](#). In case of cluster abnormalities, such as creation failure, the prompt information appears on the right side. The

detailed error information can be viewed by hovering the cursor over it. You can also sort the status by clicking Status (default).

- **Billing method:** The billing method of the cluster.
- **Operation:** Operations that can be applied to clusters, including the following:
 - **View details:** Enter the cluster details page and view the detailed information after the cluster is created.
 - **Adjust scale:** The entry for cluster expansion function.
 - **Release:** Release a cluster, see [Release a cluster](#).

3.5 Cluster details page

Cluster details page displays the cluster details,
including the following four parts.

Cluster information

- **ID/name:** ID and name of the cluster instances.
- **Billing method:** The billing method of the cluster.
- **Region:** The region where the cluster is located.
- **Current status:** The current status of the cluster, for more information, see [Cluster status](#).
- **Start time:** The creation time of the cluster.
- **Running time:** The running time of the cluster.
- **Log activation:** Determines whether the log saving function is activated.
- **Log position:** If the log saving function is activated, the position where the log is saved is displayed here.
- **Software configuration:** Information about software configuration.
- **Bootstrap/software configuration:** Whether abnormalities occur.
- **High availability cluster:** Whether the high availability cluster is activated.

Bootstrap action

The names, paths, and parameters of all configured bootstrap actions are listed here.

Software information

- **Main version:** Use the main version of E-MapReduce.
- **Cluster type:** The selected cluster type.

- **Software information:** All application programs installed by the user and their versions are listed here, such as Hadoop 2.6.0 and Hive 0.14.

Hardware information

- **Network type:** Network type used by the cluster.
- **Main security group:** The security group the cluster belongs to.
- **Main availability zone:** The availability zone where the cluster is located, for example, cn-hangzhou-b, consistent with ECS.
- **VPC/subnet:** The VPC where the user cluster is located and the ID of subnet VSwitch.
- **Master node information:** Configurations corresponding to all master nodes, including the following content.
 - **Quantity of nodes:** Quantity of current nodes and the actual number of applied nodes.
Theoretically, the two values are the same. However, during creation, the quantity of current nodes is less than applied nodes until the creation is complete.
 - **CPU:** The quantity of CPU cores at a single node.
 - **Memory:** The memory volume of a single node.
 - **Data disk type:** The type of data disk.
 - **Data disk:** The data disk volume of a single node.
 - **Instance status:** Including creating, normal, expanding, and released.
 - **Public network IP:** The public network IP address of master node.
 - **Intranet IP:** The intranet IP address of a machine which can be accessed by all nodes in the cluster.
 - **ECS expiration time:** The expiration date for use of a purchased ECS.
 - **E-MapReduce expiration time:** The expiration date for use of a purchased E-MapReduce.
- **Core node information:** Configurations corresponding to all core nodes, including the following content.
 - **Quantity of nodes:** Quantity of current nodes and the actual number of applied nodes.
 - **CPU:** The quantity of CPU cores of a single node.
 - **Memory:** The memory volume of a single node.
 - **Data disk type:** The type of data disk.
 - **Data disk:** The hard disk volume of a single node.
 - **Instance status:** Including creating, normal, and expanding.

- **Intranet IP**: The intranet IP address of a machine which can be accessed by all nodes in the cluster.
- **ECS expiration time**: The expiration date for use of a purchased ECS.
- **E-MapReduce expiration time**: The expiration date for use of a purchased E-MapReduce.

3.6 Service list

The Service list page has been added to the tab list of the cluster information page to show the running status of HDFS, YARN, and other services.

The running status of services are only showed for clusters that are in Idle or Running status.

When creating a cluster, if the service you click is unchecked (such as Storm), you are prompted that no record is found.

The list includes status and the intranet IP. The status is divided into Running and Idle. If the status of a service on some node is Idle, you can use the master node to jump to the corresponding node, and check the service process.

3.7 Cluster script

A cluster, especially a subscription cluster, can have new third-party software installed to modify the operating environment of the cluster. After a cluster is created, the cluster script allows you to select nodes in batches and run your specified script to fulfill individual requirements.

The role of cluster scripts

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

- Use Yum to install the software that has been provided.
- Directly download public software packages from the public network.
- Read your data from OSS.
- Install and run a service like Flink or Impala, but the script to be compiled is more complex.

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

How to use a cluster script

1. A cluster script can run on an idle or running cluster. On the cluster list page, click **Display details** of the corresponding cluster.

2. Click **Cluster script** on the left-side menu to enter the cluster script execution interface. A list of cluster scripts that have been run is on the right side.
3. Click **Create and run** at the upper right corner to enter the creation interface.
4. Input configuration items in the creation interface. Select the node for execution and click Run to confirm and run the operations.
5. You can see the newly created cluster script in the cluster script list. Click Refresh to update the cluster script status.
6. You can click **Display details** to display the running status of a script on each node or click **Refresh** to update the running status of a cluster on each node.

The cluster script can only run on available clusters that are idle or running. The cluster script is applicable for long-standing clusters. For on-demand clusters, perform a bootstrap action to initialize the clusters.

The cluster script downloads a script from the OSS and run it on 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 Hadoop account.

A cluster script can successfully run on some nodes, but fails on others. For example, the restart of 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 at a time can run on a cluster. If a cluster script is running, you cannot submit a new cluster script for execution. For each cluster, you can retain up to ten cluster script records. Therefore, if you already have ten records, to create a new cluster script, you must delete the previous records first.

Script example

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

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

```
osscli --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, an Internet address, or a 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 to the script using Yum, for example, ld-linux.so. 2 :

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

3.8 Cluster renewal

When your Subscription cluster service renewal is due to expire, you must renew the cluster to continue E-MapReduce cluster services. The cluster renewal includes the renewal of the E-MapReduce service charge and ECS fees.

Renewal entrance

1. Log on to [Alibaba Cloud E-MapReduce Cluster List Page](#).
2. Specify the cluster to be renewed.
3. Click **Renewal** under the corresponding cluster to enter the cluster renewal page.

Renewal page

- **Renewal:** Select the machine to be renewed.
- **ECS instance ID:** The machine ECS instance ID in the cluster.
- **Current ECS expiration time:** The ECS expiration date.
- **Current E-MapReduce expiration time:** The E-MapReduce product expiration date.
- **ECS renewal time:** The renewal duration for ECS (1-9 months and 1-3 years are supported).
- **E-MapReduce renewal time:** The renewal duration of the E-MapReduce service at the node. We recommend that you keep it consistent with ECS.
- **ECS renewal price:** Corresponding renewal price of the ECS nodes.

- **E-MapReduce renewal price:** The renewal price of corresponding nodes of the E-MapReduce service.

Pay for the order

**Note:**

The fees for cluster renewal is the sum of ECS renewal price and E-MapReduce service product price. If there are unpaid orders in the cluster list, you cannot expand or renew any clusters.

1. Click **Confirm** to view the prompt box for successful order placement (be patient, as the prompt information may be delayed for a while).
2. Click **Pay for the order** to go to the order payment page. The payment page displays the total amount payable and order details. One is the order of E-MapReduce product fees, and others are the ECS orders of cluster renewal.
3. Click **Confirm payment** to complete the payment.
4. After you make the payment, click **Completed** to return to the cluster list page.

The expiration time of successfully renewed clusters displayed on the cluster list page is updated to the time after renewal. For corresponding ECS, the expiration time after the renewal is usually updated after about three to five minutes.

If you confirm the renewal order, but don't pay for it, you can find the cluster entry on the cluster list page. **Pay** and **Cancel** are displayed in the operation column on the right side. You can click **Pay** to complete the corresponding order payment and cluster expansion processes, or click **Cancel** to cancel the renewal.

3.9 Security group

The security group created in E-MapReduce is used during the cluster creation.

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

If it is necessary to link with the cluster that has been created, follow these steps.

Add E-MapReduce cluster to existing security group

1. Log on to [Alibaba Cloud E-MapReduce Console Cluster List](#).

2. Specify the cluster entry to be added to the security group. Click View details to enter the cluster details page.
3. Specify the security group under Network information on the cluster details page. View the name and ID of the security group where all ECS instances are located.
4. Enter [Alibaba Cloud ECS console](#). Click **Security group** on the left side to find the security group entry in the list as viewed in previous step.
5. Click **Manage instances** in the security group, and see ECS instances names starting with emr-xxx. These are corresponding ECS instances in the E-MapReduce cluster.
6. Select all of these instances, and click Move to security group. Then select the new security group.

Add the existing cluster into 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 the scattered machines in the ECS console directly and move the clusters to E-MapReduce security group in batch.

Security group rules

The security group rules are subject to the OR relationship when an ECS instance is in several different security groups. For example, only port 22 of the E-MapReduce security group is opened, while all ports of **User security group** are opened. After the cluster of E-MapReduce is added into **User security group**, all ports of the machine in E-MapReduce are opened. Therefore, pay attention when performing this process.

3.10 Quick portal for components

Quick portal for components

Quick portal for component viewing

When a cluster is created, several domain names are bound to the cluster by default for you to access your open source components:

- Yarn
- HDFS
- Ganglia

These links can be found in **Quick portal for components** in Cluster management.

By default, there is no username and password for accessing. Therefore, the access request cannot pass the HTTP authentication. You need to click **Access setting** to set an access username and password to access your component UI interface.

Only one username and one password can be used. Therefore, the new username and password will always replace the previous ones.

**Note:**

NOTE: Currently, this function is only supported by Version 2.3 and later.

3.11 Node upgrade

In actual use, the CPU or memory of cluster nodes, especially master nodes, may be insufficient. Currently, E-MapReduce does not support direct upgrade configuration. We recommend that you upgrade nodes configuration in the following way.

**Note:**

Subscription clusters can be upgraded.

Procedure

1. Confirm the cluster to be upgraded.
2. Go to the EMR console, click **Manage** in the dashboard to enter the Cluster Management page.
3. Click **Upgrade Configuration** on the right corner.
4. Modify the upgrade configuration of nodes to be upgraded.
5. Click **OK**. Wait for a while, the order is generating.
6. Pay for the order.

Return to the Cluster Management page, refresh the page to make sure that the node configuration has become the target specification, for example, CPU is 4 core and memory is 16G.

7. Go to the ECS console, find the upgraded instances and restart them one by one.
8. Modify the cluster configuration so that Yarn can use the new resources.
 - a. Modify the `yarn-site.xml` file.
 - b. Change the value of `yarn.nodemanager.resource.memory-mb` to machine memory * 0.8 , the unit is MB. Change the value of `yarn.scheduler.maximum-allocation-`

mb to machine memory * 0.8, the unit is MB. For example, in my 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 the Yarn service. Generally, you only must restart the worker node. However, after the restart, the Node Manager port is changed. Therefore, we recommend that you restart the Resource Manager.
9. Open a ticket to the E-MapReduce team for providing information about the new node configuration. The E-MapReduce team will synchronize the configuration to guarantee normal cluster operating.

3.12 Disk resizing

Currently, direct disk resizing on the product is not supported for EMR. In case of insufficient data storage space, you can directly resize the disk in the ECS console.

The procedure is as follows.

Node data disk resize

1. Log on to the EMR console to enter the details page of the cluster to be resized.
2. View the **ECS ID** of the core node in the cluster to be resized, for example, `i-bp1bsithym5hh9h93xxx`. All the node disks in the cluster are uniformly resized 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** 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 **Management**, enter the instance details page, and click **This Instance Disk** on the left.
5. Resize the data disk. You must repeat the preceding steps individually to resize each disk, because multiple disks cannot be resized in batch.
6. Resize all the disks in the ECS console, and then restart the node.
7. See [ECS disk resize instructions](#) for disk resizing.

**Note:**

When the unmount operation fails, YARN, and HDFS services must be disabled on the cluster. Additionally, Disk1 operation may encounter unmount failure because of log writing by `ilogtail`. `ilogtail` must be temporarily killed using the command `sudo pgrep ilogtail | sudo xargs kill`. The `ilogtail` service can be recovered by restarting the node later.

8. Then you can see that all disks are resized using the command `df -h` on the node.
9. To guarantee that the subsequent EMR resize process is consistent with the disk after the resize, open a ticket to contact our team to perform cluster data updates.

Node system disk resize

System disk resize is a complex operation. if not necessary, avoid this operation.

1. In the EMR console, click to enter the details page of the cluster to be resized.
2. View the **ECS ID** of the master node in the cluster to be resized, for example, `i-bp1bsithym5hh9h93xxx`. All the node disks in the cluster are uniformly resized 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 the **Instance** 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 **Management**, enter the instance details page, and click **This Instance Disk** on the left.
5. Find the system disk (only one system disk can be available).
6. See [ECS system disk resize instructions](#) for system disk resizing.

**Note:**

- For a non-HA cluster, resizing the system disk makes the cluster unavailable while resizing.
- After resizing, the file `/etc/hosts` in the node may be changed because of some disk operations by ECS, and the file must be repaired after resizing. Also, SSH log-on-free is damaged, but the service remains unaffected and can be manually repaired.

3.13 Convert Payment Methods

There are two payment methods for EMR: Pay-As-You-Go and Subscription. We usually try to use EMR in Pay-As-You-Go method with a small cost. Once you confirm to use EMR for a long time, you can convert to Subscription payment method.

Convert Pay-As-You-Go to Subscription

You can select Pay-As-You-Go method at the beginning for EMR trial, and then at any time, in the Cluster Details page, click the Pay-As-You-Go to Subscription button to switch the cluster payment method from Pay-As-You-Go to Subscription. The payment method of the entire cluster will be switched.

Convert Subscription to Pay-As-You-Go

Currently, converting Subscription to Pay-As-You-Go is not supported.

4 Job

4.1 Hadoop MapReduce Job Configuration

Hadoop MapReduce Job Configuration

Procedure

1. Log on to [Alibaba Cloud E-MapReduce console](#) and go to the Jobs page.
2. At the upper right corner of the page, click **Create job**.
3. Enter a job name.
4. Select a Hadoop job type to create a Hadoop Mapreduce job. This type of job is Hadoop job submitted in the background via the following process.

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

5. Fill in the **Parameter** with command line parameters to be provided to submit this job. Please note that content to be filled in this option box shall be started with the first parameter after hadoop jar. That is to say, in the option box, the address for jar to be provided to run this job shall be first followed by [MainClass], and you can provide other command line parameters themselves.

For instance, you want to submit a Hadoop sleep job which doesn't write/read any data, then this job will succeed only by submitting mapper reducer tasks to the cluster and waiting for each task to sleep for a while. In Hadoop (e.g. hadoop-2.6.0), this job is packaged in hadoop-mapreduce-client-jobclient-2.6.0-tests.jar of the Hadoop release version. If this job is submitted from the command line, the command will be:

```
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, in the **Parameter** field, enter the following content:

```
/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. There is a problem that the user may put these jar packages anywhere, and as the cluster is created and

released, these jar packages become unavailable as they are released. Therefore, upload the jar package using the following methods:

1. Users send their own jar packages to the bucket of the OSS for storage. When you configure the parameters for hadoop, click **select the OSS path** to select and execute the jar package you want from the OSS directory. System will then auto-complete the OSS address for jar packages. Be sure to switch the prefix of the jar for your code to ossref (click **switch resource type**), to ensure that the jar package is downloaded correctly by MapReduce.
 2. 1. Click **OK**, the OSS path for this package will be auto completed in the **Parameters** field. When a job is submitted, the system will find the corresponding jar packages automatically as per this path.
 3. Behind the jar package path for this OSS, other command line parameters for running a job will be further filled in.
6. Select the policy for failed operations.
 7. Click **OK**. The job is configured successfully.

In above example, sleep job has no data input/output. If the job needs to read data and process input results (e.g. wordcount), the data input and output paths shall be specified. You can read/write data on HDFS of E-MapReduce cluster as well as data on OSS. To read/write data on OSS, it can be done only by writing the data path as the OSS path when filling 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 Hive job configuration

When users are applying for a cluster in E-MapReduce, they are provided with a Hive environment by default. Users can directly create and operate their tables and data by using Hive.

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 TERMIN
```

```
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 then 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 Job List](#).
4. Click **Create a job** in the upper right corner to enter the job creation page.
5. Enter the job name.
6. Select the Hive job type to create a Hive job. This type of job is submitted in the background by using the following process:

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

7. Enter the **Parameters** in the option box with parameters subsequent to Hive commands. For example, if it is necessary to use a Hive script uploaded to OSS, the following must be entered:

```
-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 Hive script on OSS. Switch the Hive script prefix to ossref (click **Switch resource type**) to guarantee this file is properly downloaded by E-MapReduce.

8. Select the policy for failed operations.
9. Click **OK** to complete the Hive job configuration.

4.3 Pig job configuration

When you are applying for clusters in E-MapReduce, a Pig environment is provided by default.

You can create and operate tables and data by using Pig.

The procedure is as follows.

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 YYMMDDHHMMSS.
-- 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 then 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 Job List](#).
4. Click **Create a job** in the upper right corner to enter the job creation page.
5. Enter the job name.
6. Select the Pig job type to create a Pig job. This type of job is submitted in the background by using the following process:

```

pig [user provided parameters]

```

7. Enter the **Parameters** in the option box with parameters subsequent to Pig commands. For example, if it is necessary to use a Pig script uploaded to OSS, the following must be entered:

```

-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 (click **Switch resource type**) to guarantee this file is properly downloaded by E-MapReduce.

8. Select the policy for failed operations.
9. Click **OK** to complete the Pig job configuration.

## 4.4 Spark job configuration

Spark job configuration

### Procedure

1. Log on to the [Alibaba Cloud E-MapReduce Console Job List](#).
2. Click **Create a job** in the upper right corner to enter the job creation page.
3. Enter the job name.

4. Select the Spark job type to create a Spark job. This type of job is submitted in the background by using the following process:

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

5. Enter the **Parameters** in the option box with command line parameters required to submit this Spark job. Only the parameters after **spark-submit** can be entered. The following example shows how to enter the parameters for creating Spark jobs and Pyspark jobs.

- Create a Spark job

Create a Spark WordCount job:

— Job name: Wordcount

— Type: Select Spark

— Parameters:

- The command is as follows:

```
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
```

- In the E-MapReduce job parameter box enter the following:

```
--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. The way to reference this Jar package is: *ossref://emr/checklist/jars/emr-checklist\_2.10-0.1.0.jar*. Click **Select OSS path** to view and select from OSS, the system will automatically complete the absolute path of Spark script on OSS. Switch the default oss protocol to ossref protocol.

- Create a pyspark job

Additionally to Scala and Java job types, E-MapReduce also supports Spark jobs of python type. Create a Spark Kmeans job for 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 resource are supported, and ossref protocol is used.

— For Pyspark, online Python installation kit is not supported.

6. Select the policy for failed operations.

7. Click **OK** to complete the Spark job configuration.

## 4.5 Spark SQL job configuration

Spark SQL job configuration



**Note:**

By default, the Spark SQL mode to submit a job is Yarn mode.

### Procedure

1. Log on to the [Alibaba Cloud E-MapReduce Console Job List](#).
2. Click **Create a job** in the upper right corner to enter the job creation page.
3. Enter the job name.
4. Select the Spark SQL job type to create a Spark SQL job. This type of job is submitted in the background by using the following process:

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

5. Enter the **Parameters** in the option box with parameters subsequent to Spark SQL commands.

- -e option

Directly write running SQL for -e options by inputting it into the **Parameters** box of the job, for example:

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

- -f option

-f options can be used to specify a Spark SQL script file. Loading well prepared Spark SQL script files on OSS can give more flexibility. We recommend that you use this operation mode, for example:

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

6. Select the policy for failed operations.
7. Click **OK** to complete Spark SQL job configuration.

## 4.6 Shell job configuration

Shell job configuration



### Note:

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

### Procedure

1. Log on to the [Alibaba Cloud E-MapReduce Console Job List](#).
2. Click **Create a job** in the upper right corner to enter the job creation page.
3. Enter the job name.
4. Select the Shell job type to create a Bash Shell job.
5. Enter the **Parameters** in the option box with parameters subsequent to Shell commands.

- -c option

-c option can be used to set Shell scripts to run by inputting it into the **Parameters** box 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 option can be used to run Shell script files. By uploading a Shell script file to OSS, Shell scripts on OSS can be directly defined in the job parameters. This is more flexible than the -c option, for example:

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

6. Select the policy for failed operations.
7. Click **OK** to complete Shell job configuration.

## 4.7 Sqoop job configuration

Sqoop job configuration



### Note:

Only E-MapReduce products version V1.3.0 and higher support the Sqoop job type. Running a Sqoop job on lower cluster versions will fail, and erlog will report “Not supported” errors. For parameter details, refer to [Data Transmission Sqoop](#).

### Procedure

1. Log on to the [Alibaba Cloud E-MapReduce Console Job List](#).
2. Click **Create a job** in the upper right corner to enter the job creation page.
3. Input the job name.
4. Select the Sqoop job type to create a Sqoop job. In E-MapReduce back-end, Sqoop job will submit through the following process:

```
sqoop [args]
```

5. Complete the **Parameters** option box with parameters subsequent to Sqoop commands.
6. Select the policy for failed operations.
7. Click **OK** to complete Sqoop job definition.

## 4.8 Job operations

Job operations

### Job creation

A new job can be created at any time. The job created can currently only be used in the region where it is created.

### Job clone

To completely clone configurations in which jobs already exist. It is also restricted to the same region.

### Job modification

If it is necessary to add a job into an execution plan, then it is required to ensure that this execution plan is not under operation and its periodic scheduling is not in progress before the job can be modified.

If it is necessary to add this job into several execution plans, the modification shall be made until the executing and periodic scheduling of all these plans. Since job modification may cause changes to all execution plans using this job and also errors to execution plans under executing and periodic scheduling.

If it is necessary to conduct debugging, clone is recommended, and after debugging, original jobs in the execution plan will be replaced.

### Job deletion

As with modification, jobs can be deleted only when the execution plan in which jobs are added in is not under operation and periodic scheduling is not in progress.

## 4.9 Job date variables

During creation, time variable wildcard settings in job parameter are supported.

### Variable wildcard format

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

- If it is written as `${yyyyMMdd HH:mm:ss-1d}` in job parameters, then this parameter wildcard will be, when executed practically, replaced with 20160426 12:08:01, that is, the current date minus one day with accuracy to the second.
- If it is written as `${yyyyMMdd-1d}`, then it will be replaced with 20160426 when being executed, representing the day before current date.
- If it is written as `${yyyyMMdd}`, then it will be replaced with 20160427, representing the current date.

The dateexpr represents the expression of standard time format, and corresponding time will be formatted as per this expression and followed by corresponding time to add or deduct. Following the expression, 1d (1 day) to add or deduct can be written as N days or hours, for example, `${yyyyMMdd-5d}`, `${yyyyMMdd+5d}`, `${yyyyMMdd+5h}`, `${yyyyMMdd-5h}` are all supported, and corresponding replacing methods are consistent with the descriptions above.



#### Note:

E-MapReduce currently supports addition and deduction only for “hour” and “day”, that is, the format of +Nd, -Nd, +Nh and -Nh after dateexpr (dateexpr refers to the expression of time format and N is an integer).

## Example

When being executed practically, the **Parameter** in the job on the figure below will be replaced with:

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

## 5 Execution plan

---

### 5.1 Create an execution plan

An execution plan is a set of jobs that can be executed once or periodically through scheduling configuration. It can be executed on an existing E-MapReduce cluster and also can create a temporary cluster to execute assignment dynamically. Its biggest advantage is to use resources actually needed during execution to maximize resource savings.

The steps to create an execution plan are as follows:

1. Log on to the [Alibaba Cloud E-MapReduce Console Plan Execution Page](#).
2. Select the region.
3. Click **Create an execution plan** in the upper right corner to enter the execution plan creation page.
4. The Select Cluster Mode page has two options: **Create on demand** and **Existing clusters**.
  - a. **Create on demand**: Create a new cluster to run jobs.
    - Execution plan for one-time scheduling: Clusters with corresponding configuration will be created when execution starts and then released upon the completion of the operation. For specific descriptions of creation parameters, see [Create a cluster](#).
    - Execution plan for periodic scheduling: A new cluster will be created as per users' settings when each scheduling period starts and then released upon the completion of the operation.
  - b. **Existing clusters**: Use an existing cluster that complies with the following requirements. Select the **Existing clusters** and then enter the Select Cluster page. You can select the cluster to associate with the execution plan.

Execution plans can only be submitted to clusters in **Running** and **Idle** status.

5. Click **Next** to enter the job configuration page. All user jobs will be listed in the left table, and you can select jobs for execution. By clicking the right-facing button, the checked jobs will be added into the job queue. Jobs in the queue will be submitted to the cluster for execution as per their order. The same job can be added and executed several times. If you have not created any jobs, see operating instructions to create jobs.
6. Click **Next** to enter the scheduling mode configuration page. The configuration items are described as follows:



a. **Execution plan name:** Must be between 1-64 characters and consist of only Chinese characters, letters, numbers, “-”, and “\_”.

b. **Scheduling strategy**

- **Manual execution:** The execution plan will not be automatically executed after creation, it must be manually executed. Once the execution is in progress, it cannot be conducted again.
- **Periodic scheduling:** This function will be enabled immediately after the execution plan is created. The execution can begin from the configured scheduling time point. Periodic scheduling can be disabled in the list page. If a scheduling execution starts, but its last scheduling execution has not completed, this scheduling will be ignored.

c. **Scheduling period setting:** There are two scheduling periods: days and hours. The ‘day’ cycle is ‘one day’ by default and remains unchanged. However, for the hour parameter, you can set the specific time interval and the range must be 1-23.

d. **First execution time:** The effective start time of scheduling. From this time on, periodic scheduling will be conducted as per scheduling periods. The first scheduling will be conducted from the latest time point when requirements are met as per actual time.

7. Click **Confirm to submit** to complete the creation of the execution plan.

## Others

- **Example for periodic scheduling**

Configure the scheduling mode
✕

---

\* Scheduling policy :

Periodic scheduling execution plan ▼

\* Set the scheduling cycle :

day(s) ▼

per

1

^  
v

day(s)

\* first execute time :

2016-11-03

:

12

^  
v

:

15

^  
v

First run time 2016-11-3 12:15

Subsequent intervals 1 day(s) run 1 Times

These configurations indicates that the scheduling is initially started on 11/03/2016, 12:15 and then conducted every other day. The second scheduling is conducted on 11/04/2016, 12:15.

- **Execution sequence of jobs**

For jobs in the execution plan, they will be executed from first to last as per the sequence of user-selected jobs in the job List.

- **Execution sequence of multiple execution plans**

Each execution plan can be deemed as an integral whole. When multiple execution plans are submitted to the same cluster, each execution plan submits jobs from its internal job sequence, which is consistent with the sequence of a single execution plan. While jobs among multiple execution plans are in parallel.

- **Practical example for early job debugging**

During job debugging, if the cluster is created on demand automatically, the speed will be slow and will take a long time to start the cluster. We recommend that you manually create a cluster first, and then select “Associate the cluster” in the execution plan to run jobs, and then to set the scheduling mode as Execute immediately. During debugging, results are viewed by clicking Run on the execution plan list page. Modify the execution plan once job debugging is completed, and then modify the mode of associating existing clusters into creating a new cluster on demand. Then modify the scheduling mode into periodic scheduling as needed. Tasks will be executed automatically on demand.

## 5.2 Manage execution plans

Manage execution plans

1. Log on to the [Alibaba Cloud E-MapReduce Console Plan Execution Page](#).
2. Find corresponding execution plan items and click the **Manage** button to enter the execution plan management page. On this page you can:

- **View details of the execution plan**

You can view the basic information of execution plans, such as names, associated clusters, job configurations, scheduling strategy and status, alarm information.

- **Modify the execution plan**



**Note:**

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

### Modify independently

Each separate module can be modified independently. Click the **Modify** button to perform modification.

- **Configure alarm notification**

There are three types of alarm notifications:

- Notification for booting timeout: If a periodical scheduling has not been conducted properly at the specified time and is not executed within 10 minutes of timeout, an alarm will be sent.
- Notification for failed execution: If any job in the execution plan fails, an alarm will be sent.
- Notification for successful execution: If all jobs in the execution plan are successfully executed, a notification will be sent.

- **Run and view results**

When the execution plan can be run, there will be a **Run** button to the right of **Scheduling status** in **Basic Information**. Once this button is clicked, a schedule will be executed.

At the bottom of the page, there are running records displaying the execution plan instances executed each time, facilitating views of the corresponding job list and logs.

## 5.3 Execution plan list

Displays basic information of all your execution plans.

- **Execution plan ID/name:** The ID and corresponding name of the execution plan.
- **Recent Execution Cluster:** The last cluster to execute this execution plan. It is a cluster created on demand or an existing associated cluster. If a cluster is created automatically on demand, (Automatically created) will be displayed below the cluster, indicating that the cluster is created automatically by E-MapReduce on demand and will be released automatically after running.
- **Last running condition:** The running status of the last execution plan.

- **Start time:** The time from which the last plan starts to run.
- **Running time:** The duration for which the last plan is running.
- **Running status:** The running status of the last execution plan.
- **Scheduling status:** Whether the scheduling is in progress or has been stopped. Only periodic jobs have the scheduling status.
- **Operation**
  - **Run now:** Manual run can be made only when the job is neither running nor being scheduled. The execution plan will be run once immediately after clicking.
  - **Enable/Stop scheduling:** When the scheduling is stopped, Enable scheduling will appear which can be clicked to start scheduling. When Stop scheduling is displayed during scheduling, clicking it will stop the scheduling. This button is only for periodic execution plans.
  - **Running records:** Click to enter the job log viewing page in the execution plan.
  - **More**
    - **Modify:** To modify the configuration of an execution plan. The execution plan in the process of scheduling and running cannot be modified.
    - **Delete:** To delete an execution plan. The execution plan in the process of scheduling or running cannot be deleted.

## 5.4 View job results and logs

View job results and logs

### View execution records

1. Log on to the [Alibaba Cloud E-MapReduce Console Plan Execution Page](#).
2. Click **Running** records in the right-side operation items to enter the execution records page.
  - **Execution sequence ID:** The sequence of execution for the corresponding record, indicating the ordinal position in the whole execution queue. For example, 1 stands for the first position of execution while n stands for the n-th position.
  - **Running status:** The running status of each execution record.
  - **Start time:** The time when the execution plan starts to run.
  - **Running time:** The total running time until the page is viewed.
  - **Execution cluster:** The cluster run by the execution plan, can be either a cluster on demand or an existing associated cluster. Click to view the cluster details page.

- **Operation**

**View the job list:** Click this button to enter the job list of a single execution plan to view the execution condition of each job.

### Job record view

Here you can view the job list in execution records of a single execution plan and details of each job.

- **Job execution ID:** A corresponding ID will be created after a job is executed, and this ID is different from the job ID. The job execution ID is the unique record identifier to view the logs on OSS.
- **Job name:** The name of the job.
- **Status:** The running status of the job.
- **Job type:** The type of job.
- **Start time:** The time when the job starts to run. It has been converted into local time.
- **Running time:** The total running time (in seconds) of this job.
- **Operation**
  - **Stop the job:** A job can be stopped 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 standard output (that is, Channel 1) of the master process. If log saving for the cluster where jobs are run is not enabled, this viewing function cannot be executed.
  - **stderr:** Records all output content from diagnostic output (that is, Channel 2) of the master process. If log saving for the cluster where jobs are run is not enabled, this viewing function cannot be executed.
  - **View the job instance:** To view the log of all job worker nodes. If log saving for the cluster where jobs are run is not enabled, this viewing function cannot be executed.

### Job worker log view

- **Elastic Compute Service (ECS) instance ID/IP:** The ECS instance ID of a running job and the corresponding intranet IP address.
- **Container ID:** The container ID of Yarn running.
- **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 available computing resources of a cluster, multiple execution plans can currently be mounted to the same cluster to utilize parallel execution.

The main points are summarized as follows:

1. Jobs in the same execution plan will be executed in series, and it is considered by default that subsequent jobs can be submitted and executed only after execution of preceding jobs is completed.
2. In case of sufficient cluster resources, it is required to create a number of different execution plans and to associate them to the same cluster for submission and running to execute multiple jobs in parallel (a cluster is considered by default to support at most 20 execution plans for parallel execution).
3. The management and control system currently supports parallel submission for execution plans associated to the same cluster to Yarn. However, if the cluster itself has insufficient resources, jobs will be congested in the Yarn queue to wait for scheduling.

For the process of creating execution plans and associating it to the cluster, refer to [Create an execution plan](#).

## 6 Alert management

---

### 6.1 Cluster alarm management

To help you monitor the operation of clusters, 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 notifies the contacts in the notification group timely so that you can deal with the problem in time.

#### Configure alarm rules

To set up alarm rules for E-MapReduce cluster, take the following steps:

1. Log on to [CloudMonitor console](#).
2. Click **Cloud Service Monitoring > E-MapReduce** in turn in the left navigation pane to enter the E-MapReduce Monitoring List page.
3. Click **Alarm Rules** tab.
4. Click **Create Alarm Rule** in the upper-right corner of the page to set an alarm rule for corresponding monitoring metrics.
5. In the **Related Resource** region, set Products and Resource Range.
  - **Products**: Select E-MapReduce in the drop-down list.
  - **Resource Range**: The scope of action of the alarm rule. It 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 1000, problem that the threshold has been reached without alarming may occur if it exceeds 1000, we recommend that you use application group to divide resources according to business before setting up alarm rules.
    - **All Resources**: indicates that the rule works on all instances of E-MapReduce for the current account. For example, set The CPU usage is greater than 80% to alarm, as long as there is instance whose CPU usage is greater than 80% in the current account, it hits this rule.
    - **Application Group**: indicates that the rule works on all instances of a certain application group. For example, set the CPU usage is greater than 80% to alarm, as long as there is a host whose CPU usage is greater than 80% in this group, it hits this rule.

- **Cluster**: indicates that the rule only works on a specific E-MapReduce cluster. For example, set the CPU usage is greater than 80% to alarm, as long as there is an instance whose CPU usage is greater than 80% in this cluster, it hits this rule.

#### 6. Configure the **Set Alarm Rules** region.

- **Rule Name**: Sets the name of alarm rules.
- **Rule Description**: The principal of the alarm rule that defines what conditions the item data meets, trigger alarm rules. For example, the rule is described as a 1 minute average of CPU usage  $\geq 90\%$ , it means that the rule is hit if the average data within 1 minute is  $\geq 90\%$ . For more information about monitoring metrics for E-MapReduce clusters, see [E-MapReduce monitoring](#).
- **Rule**: By default, any role is applicable.

Click **Add Alarm Rule**, you can set multiple alarm rules (charge as multiple alarm rules). As long as one of the rules is triggered, the system sends notifications to the notification group.

- **Mute for**: The interval for sending the alarm notification again in case of the monitoring metrics is not restored.
- **Triggered when threshold is exceeded for**: Times that the rule is hit to send alarm notifications. For example, the rule is described as a 1 minute average of System State CPU usage  $\geq 90\%$  and it appears for 3 times or more continuously, it means that the rule is hit if the average data within 1 minute is  $\geq 90\%$  and it appears for 3 times continuously.
- **Effective Period**: The effective time of the alarm rule. CloudMonitor checks whether the monitoring metric hits the alarm rule only during the effective period. The system checks whether the monitoring data requires an alarm only during the effective time.

#### 7. Configure the **Notification Method** region.

- **Notification Contact**: The contact group that receives the notification. Enter the keyword of the contact group in the search box to locate the contact group you want to associate quickly, and click the right arrow icon, then the contact group is added into the right contact list. If you haven't created the appropriate contact group, click **Quickly create a contact group**. After you select a contact group in the right contact list, click the left arrow icon to delete the contact group.
- **Notification Methods**: Alarm information is divided into three levels: critical, warning, info. Different levels correspond to different notification methods. Different alarm levels correspond to different notification methods. Before Critical level is configured, you need to buy the phone alarm resource package.



- **Email Remark** (Optional): Customizes supplemental information for alarm email. When you fill in your email remarks, your comments are included in the notification message that is being sent to the 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 of HTTP protocol. When you receive the alarm notification, you can make further process according to the notification content. For more information, see [Alarm callback](#).

8. Click **Confirm** to complete the alarm rule configuration

## 6.2 Job Alarm Management

You can add and change the contact and alarm receiving group on the alarm label page. Each alarm receiving group can be correlated with one or more contacts.

### Add a contact

1. Log in to the [Alibaba Cloud E-MapReduce Console Alarm Management Contact Page](#).
2. Click **Add Contact** to add a contact.
3. Enter the contact name and phone number. Click to send the verification code. Obtain and enter the corresponding phone verification code.
4. Click **Confirm** to complete the addition of contact.

### Associate execution plan with alarm receiving group

After entering the contact and alarm receiving group, associate the execution plan with the corresponding alarm receiving group in the execution plan management page.

1. Log on to the [Alibaba Cloud E-MapReduce Console Plan Execution Page](#).
2. Click the management button at the right of execution plan entry and enter the execution plan management page.
3. Click to open the **Alarm notice** in the **Alarm information** column on the management page and correlate with the specific alarm receiving group.

After opening of **Alarm notice**, the contacts in the correlated alarm receiving group can receive the short message notice once the plan is executed. The short message contains the execution plan name, plan execution (quantities of plans succeeded and failed), corresponding execution cluster name and specific execution time.

## 7 Software configuration

Hadoop, Hive, Pig, and other relevant software contain numerous configurations that can be changed through the software configuration function. For example, the number of service threads in HDFS server `dfs.namenode.handler.count` is 10 by default and will be increased to 50, and the size of HDFS file block `dfs.blocksize` is 128 MB by default and will be decreased to 64 MB because the system contains only small files.

### Purpose of software configuration

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

### Procedure

1. Log on to [Alibaba Cloud E-MapReduce console cluster list](#).
2. Select the region and the created cluster associated with the region is listed.
3. Click **Create cluster** to enter the cluster creation page.
4. All contained software and corresponding versions can be seen in the software configuration of cluster creation. Change the configuration of the cluster by selecting a corresponding json format configuration file in the (optional) software configuration box. Then, override or add to the defaulted cluster parameters. The sample 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 designates the configuration file to change. The **properties** parameter stores the key-value pair that requires changes. When the default configuration file has a corresponding key, override the value, otherwise, add the corresponding key-value pair.

The correspondence between configuration file and classification is shown in the following table.

- **Hadoop**

Filename	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**

Filename	classification
pig.properties	pig
log4j.properties	pig-log4j

- **Hive**

Filename	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 of structures and can be set in the embedded configurations mode. See hadoop-env in the example where `-Xmx512m -Xms512m` setting is added for **HADOOP\_CLIENT\_OPTS** property of export.

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

## 8 Bootstrap action

---

Bootstrap action is used to run your customized script before the cluster starts up Hadoop.

The customized script is used to install your required third-party software or change the cluster operating environment.

### Function of bootstrap operation

With bootstrap action, you can install many things to your cluster that are not currently supported by clusters. For example:

- Install provided 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, but the script to be compiled 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.

### How to use

1. Log on to [Alibaba Cloud E-MapReduce Console Cluster List](#).
2. Select the region where the created cluster associated with the region is listed.
3. Click **Create cluster** to enter the cluster creation page.
4. Click **Add bootstrap operation** to enter the operation page.
5. Enter the configuration item and click Complete after adding.

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

It is possible that the bootstrap action fails. For ease of use, bootstrap action failure does not affect the creation of the cluster. After the cluster is created successfully, you can view any abnormality in the Bootstrap/software configuration column of cluster information in the cluster details page. In case of any abnormality, you can log on to all nodes to view the operation logs in the directory of `/var/log/bootstrap-actions`.

## Bootstrap action type

The bootstrap action is categorized into customized bootstrap action and operating-condition bootstrap action. The main difference is that the operating-condition bootstrap action can only operate your designated operation in the node that meets the requirements.

### Customized bootstrap action

For the customized bootstrap action, the position of the bootstrap action name and the execution script in OSS must be designated and the optional parameters are set as required. During cluster initialization, all nodes download the designated OSS scripts to run them directly or after adding the optional parameters.

You can designate the files that need to be downloaded from OSS in the script. The following example downloads the file `oss://yourbucket/myfile.tar.gz` locally and extract it to the directory of `/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>
```

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



#### Note:

OSS address host contains intranet address, Internet address, and VPC network address. For the classic network, the intranet address is designated. The address in Hangzhou is `oss-cn-hangzhou-internal.aliyuncs.com`. For VPC network, the domain name that VPC intranet can visit is designated. The name in Hangzhou is `vpc100-oss-cn-hangzhou.aliyuncs.com`.

The bootstrap action 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, you do not need to make additional designations. You do need to designate the name and optional parameters. The operating-condition bootstrap action must provide the optional parameters, including the spaced operation conditions and commands. The operation conditions support `instance.isMaster=true/`



false and is designated to only operate on the master or non-master nodes. The following example shows that the optional parameters of an operating-condition bootstrap action are only designated to create the directory on the master node.

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

If multiple operation commands are designated, you can divide several statements with the semicolon “;”. For example: `instance.isMaster=true mkdir -p /tmp/abc;mkdir -p /tmp/def`.

## 9 VPC

---

Virtual Private Cloud (VPC) creates an isolated network environment for users. You can select an IP address range, divide networks, and configure the routing list and gateway.

For more information, see [What is VPC](#). The interflow of VPC intranet and between VPC and physical IDC machine rooms can be realized among regions or users through [Express Connect](#).

### Create a VPC cluster

E-MapReduce can select the type of network during cluster creation, classic network or VPC. For VPC, the following operations are required:

- Subordinate VPC: Select which VPC where the current E-MapReduce cluster is located. If no creation is made, log on to [VPC console](#) to create a VPC.
- Subnet (VSwitch): ECS instance in E-MapReduce cluster communicates through VSwitch. If no creation is made, log on to [VPC console](#) to create a VPC. The VSwitch has the property of availability zone. Therefore, the created VSwitch must also belong to the availability zone selected during cluster creation in E-MapReduce.
- Safety group creation: Once enabled, enter the name of the created security group.
- Owner security group: The security group the cluster belongs to. The security group of a classic network cannot be used in VPC. The security group of VPC can only be used in current VPC. Here only the security group created in E-MapReduce product by the user is shown. For safety reasons, a security group created outside the E-MapReduce cannot be selected. To create a security group, select New security group and enter the name of security group.

### Example

#### E-MapReduce cluster communication in different VPCs (Hive visits HBase)

##### 1. Create clusters.

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

##### 2. Configure the high-speed channel.

See VPC private network communication in the same region for configuration.

##### 3. Log on to HBase cluster through SSH and a table is created through HBase Shell.

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

##### 4. Log on to Hive through SSH.

- a. Change hosts and add a line as follows.

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

- b. Visit 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');
```

The abnormality of `java.net.SocketTimeoutException` is reported here because the security group where the ECS of HBase cluster is located limits the E-MapReduce visit at relevant ports. Port 22 is only opened by default in the security group created by E-MapReduce. Therefore, the security group rules need to be added into the security group of HBase cluster to open a port for 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

---

Python instructions

## Python 2.7

Supported Python 2.7 in E-MapReduce 2.0.0 and later versions.

Python files location: `usr/local/Python-2.7.11/` included Numpy.

## Python 3.6

EMR 2.10.0, 3.10.0 and later versions support Python 3.6.4 and the installation directory of Python is `/usr/bin/python3`. The earlier versions do not support Python 3 by default, and you have to download and install the package through the following link:

- [Download Python 3 package](#)
- Unzip the downloaded file by 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)
```

Install successfully if you see these messages.

# 11 Open source assembly

---

## 11.1 Hue

E-MapReduce currently supports Hue. You can access [Hue](#) by Apache Knox.

### Preparation

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



#### Note:

Set security group rules for limited IP ranges. It is prohibited to open rules to 0.0.0.0/0 when configuring.

### Access Hue

To access Hue, take the following steps:

1. Click **Manage** on the right side of the cluster ID in the EMR console.
2. On the left side of the Configuration page, click **Access links and ports**.

### Access password

If Hue has no administrator after the first running, the first login user is set to administrator by default. EMR will generate an administrator account and password by default for security. The administrator account is admin. You can view the password through the following way:

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

### Forgot password

If you forget the password corresponding to your Hue account, you can recreate an account by the following method:

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

5. Create a new account by executing the following command.

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

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

If **Superuser created successfully** is prompted, it means the new account is created successfully, and you can log on to Hue with the new account later.

### Add/modify configuration

1. On the right-side of the cluster list panel, click **Manage**.
2. In the service list, click **Hue**, and then click the **Configuration** tab.
3. At the right-side corner of the page, click **Custom Configuration**, configure the Key and Value fields. The Key needs to follow the following specifications:

```
$section_path.$real_key
```



#### Note:

- **\$real\_key** is the actual key to be added, such as `hive_server_host`.
- You can view **\$section\_path** before **\$real\_key**, in `hue.ini` file. For example,  
  
You can see `hive_server_host` belongs to the **[beeswax]** section in `hue.ini` file. That is, **\$section\_path** is `beeswax`.
- In conclusion, 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

### Oozie Instructions



#### Note:

Alibaba Cloud E-MapReduce version 2.0.0 and the later versions support Oozie. If it is necessary to use Oozie in a cluster, make sure that the cluster version is higher than 2.0.0.

## Preparations

Before you create a cluster, it is required to open an SSH tunnel. For detailed steps, see [Connect to the cluster using SSH](#).

Take the MAC environment as an example. Assuming the IP address of the public network for the master node of the cluster is **xx.xx.xx.xx**:

1. Log on to the master node.

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

2. Enter your password.

3. Check **id\_rsa.pub** content of the local machine (note that this will be executed on the local machine rather than the remote master node).

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

4. Write **id\_rsa.pub** content of the local machine in `~/.ssh/authorized_keys` on the local master node to execute on the far-end master node.

```
mkdir ~/.ssh/
vim ~/.ssh/authorized_keys
```

5. Paste the content observed in [Step 2](#). Now, `ssh root@xx.xx.xx.xx` can be used directly to log on to the master node without a password.

6. Execute the below commands on the local machine for port forwarding.

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

7. Enable Chrome to execute 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 Oozie UI interface

Access in Chrome browser for port forwarding: "xx.xx.xx.xx:11000/oozie", "localhost:11000/oozie", or intranet "ip: 11000/oozie".

## Submit workflow job

Before running Oozie, it is required to install the sharelib: <https://oozie.apache.org/docs/4.2.0/WorkflowFunctionalSpec.html#ShareLib>.

In E-MapReduce clusters, Oozie users are installed with sharelib by default, and there is no need to install sharelib again even though users using Oozie are to submit workflow job.

Since clusters with and without enabled HA have different modes to access NameNode and ResourceManager, when submitting an oozie workflow job, it is required to specify different NameNode and JobTracker (ResourceManager) in job.properties files. Specific steps are as follows:

- Non-HA cluster

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

- HA cluster

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

For the following operation examples, configurations are made for both non-HA and HA clusters, and the sample code can be used directly for operations without any modification. For the specific format of a workflow file, see the official documents for Oozie at <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 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 Oozie workflow code to hdfs.

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

4. Submit an Oozie workflow sample job.

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

After a successful execution, a jobId will be returned and is similar to:

```
job: 0000000-160627195651086-oozie-oozi-W
```

5. Visit the Oozie UI page to see 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
```

The current main master node can be determined by checking 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 sample codes of HA cluster.

```
[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 Oozie workflow code to hdfs.

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

4. Submit Oozie workflow sample job.

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

After a successful execution, a jobId will be returned and is similar to:

```
job: 00000000-160627195651086-oozie-oozi-W
```

5. Visit the Oozie UI page to see the submitted Oozie workflow job.

## 11.3 Presto

### Presto Instructions

E-MapReduce versions 2.0 and higher support [presto](#). Presto can be used in E-MapReduce by checking presto software when selecting the mirror image.

After cluster creation, log on to the master node. Presto software will be installed in the directory `/usr/lib/presto-current`, and PrestoServer process can be seen by command `jps`.

Presto service process can be divided into coordinator and worker. Coordinator is started on the master (the HA cluster is the master node of hostname starting with `emr-header-1`), and worker process is started on the core node. The service process configuration is under the directory `/usr/lib/presto-current/etc`, and coordinator uses `coordinator-config.properties` while worker uses `worker-config.preperities`, and other configuration files are used for public. The web port is set as 9090.

By default, presto service is set with support from Hive. Connect the metastore of hive on the cluster to read the table information of Hive and perform querying. The cluster is pre-installed with presto cli and can directly execute the following command:

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

to check the Hive list. Note that a delay of several seconds will occur when synchronizing the Hive table.

## 11.4 Zeppelin

E-MapReduce currently supports Apache Zeppelin. Zeppelin can be accessed and used in E-MapReduce just by selecting a Zeppelin-supported mirror image to create a cluster and enabling public network IP address.

### Preparation

1. In the cluster [Security group](#), [set security group rules](#), and open the 8080 port.
2. In Knox, add user name and password. For details, see [Knox guide](#) to set Knox users. The user name and password are used only to log on to Knox's various services which have no relationship with Alibaba Cloud Ram user names.



#### Note:

Set security group rules for limited IP ranges. It is prohibited to open rules to 0.0.0.0/0 when configuring.

### Access zzeppelin

Access links can be viewed as follows:

1. On the right-side of the EMR management console, click **Manage**.
2. On the left side of the page, click **Access links and ports**.

## 11.5 ZooKeeper

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

### Considerations

ZooKeeper will have only 3 nodes no matter how many machines are currently in the cluster. More nodes are not supported currently.

## Create a cluster

Zookeeper will be ticked by default in the software configuration page where the E-MapReduce creates the cluster.

## Node information

After the cluster is successfully created and the status is idle, view the cluster details page and you will find the node information of ZooKeeper. Corresponding intranet IP address (2181 for port by default) of ZooKeeper node is indicated on the process column for the access to ZooKeeper services.

# 11.6 Kafka

## 11.6.1 Quick start with Kafka

E-MapReduce 3.4.0 and later versions support the Kafka service.

### Create a Kafka cluster

Set the cluster type to Kafka when creating a cluster on E-MapReduce. Then, a cluster containing only Kafka components is created by default. The components include the basic components and the Zookeeper, Kafka, and KafkaManager components. Only one Kafka broker is deployed on each node. We recommend you use a dedicated Kafka cluster, not mixing with Hadoop services.

### Local 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, refer to the [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, 190 MB/s of sequential read and write performance on a single disk, and up to 5 GB/s of storage I/O ability.
- Price of local storage is 97% lower than that of SSD cloud disks.
- Higher network performance, up to 17 Gbit/s instances of network bandwidth which meet data interaction needs among business peak instances.

Local disk models also have the following features:

Operation	Local 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 and 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 and data is also retained.
Release (instances) on the console	Erased	The local ephemeral disk's storage volume is erased and 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 an local ephemeral disk for a long time period. Back up data in a timely manner and adopt high-availability architecture. For long-term storage, you are advised to store data on a cloud disk.

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

1. **default.replication.factor = 3** indicates that the number of the topic's partitions and replicas 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 a successful writing to write at least two replicas at a time.

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

1. Remove the bad disk from the Broker configuration, restart Broker, and recover the lost data from the bad disk on the other available local disks. Data recovery time varies according to the amount of data that have been written on the broken disk.
2. When the number of machine disks damaged (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 loss data on the damaged disk, Broker will be shut down abnormally. In this case, you can choose to clean up some data, free up disk space and restart the Broker service. You can also contact E-MapReduce for machine migration and recover abnormal disks.

## Parameter description

You can check the Kafka software configuration 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 I/O threads of the Kafka broker, which is twice the number of CPU cores by default
num.network.threads	Number of network threads of the Kafka broker, which is the same as the number of CPU cores by default

## 11.6.2 Cross-cluster access to Kafka

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

### Cross-cluster access to Kafka

Cross-cluster access to Kafka consists of two common types:

- Access E-MapReduce Kafka cluster from Alibaba Cloud internal network.
- Access E-MapReduce Kafka cluster from the public network.

Different solutions are prepared for different E-MapReduce versions.

### EMR-3.11.x and Later Versions

- **Access Kafka from the Alibaba Cloud internal network**

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

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

- 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 [here](#).
- **Access Kafka in the public network**

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

1. Interconnect Kafka clusters with the public network.
  - If Kafka clusters are deployed in a VPC environment, there are two ways:
    - Deploy Express Connect to interconnect the VPC with the public network. For details, see [Express Connect](#) document.
    - Bind EIPs to cluster Core nodes. Perform the following steps to bind the EIP to the ECS:
  - If Kafka is deployed in a classic network, there are two ways:
    - To create a Pay-As-You-Go cluster, use ECS APIs. For details, see [API document](#).
    - To create a cluster through the Subscription method, you can directly assign a public IP address to relevant host in the ECS console.
2. Create an EIP in the [ECS console](#), and purchase relevant EIPs based on the number of Core nodes in the Kafka cluster.
3. Configure security group rules for the Kafka cluster to control public network access to the IP addresses of the Kafka cluster. The objective is to improve the security of the Kafka cluster exposed in the public network. You can view the security group to which the cluster belongs in the EMR console, and configure security group rules based on security group IDs. [Document URL](#)
4. On the **Cluster List** page of E-MapReduce console, click **Manage** after the specified cluster, select **Cluster Overview** on the left side of the page, and then click the **Sync Cluster Host Info** button in the upper right corner.
5. Restart the cluster Kafka service.
6. Use the EIP of the Kafka cluster node to access Kafka service in the public network. Use port 9093 to access Kafka service from the public network.

#### Versions earlier than EMR-3.11.x

- **Access Kafka from the Alibaba Cloud internal network**

In this case, you must configure the host information of the Kafka cluster node on the client host. **Long domain** of the Kafka cluster node must be configured. 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 to the public network by default. To access the Kafka cluster in the public network, perform the following steps:

1. Interconnect Kafka clusters with the public network.
  - If Kafka clusters are deployed in a VPC environment, there are two ways:
    - Deploy Express Connect to interconnect the VPC with the public network. For details, see [Express Connect](#) document.
    - Bind EIPs to cluster Core nodes. Perform the following steps to bind the EIP to the ECS.
  - If Kafka is deployed in a classic network, there are two ways:
    - To create a Pay-As-You-Go cluster, use ECS APIs. For details, see [API document](#).
    - To create a cluster through the Subscription method, you can directly assign a public IP address to relevant host in the ECS console.
2. Create an EIP in the [ECS console](#), and purchase relevant EIPs based on the number of Core nodes in the Kafka cluster.
3. Configure security group rules for the Kafka cluster to control public network access to the IP addresses of the Kafka cluster. The objective is to improve the security of the Kafka cluster exposed in the public network. You can view the security group to which the cluster belongs in the EMR console, and configure security group rules based on security group IDs. [Document URL](#)
4. Modify the software configuration of the Kafka cluster `listeners.address.principal` to `HOST`, and restart the Kafka cluster.
5. Configure the `hosts` file on the local client host in accordance with the preceding [cross-cluster access to Kafka](#).

## 11.6.3 Kafka Ranger

Starting from EMR-3.12.0 version, E-MapReduce Kafka allows you to configure permissions with Ranger.

### Integrate Ranger into Kafka

The previous section introduced how to create a cluster with Ranger service in E-MapReduce and some preparation work. This section describes the step-by-step process for integrating Ranger into Kafka.

- **Enable Kakfa Plugin**

1. On **Cluster Management** page, click **Ranger** in the service list to enter the Ranger Management page. Click **Operation** at the upper right corner of the page and select **Enable Kafka PLUGIN**.



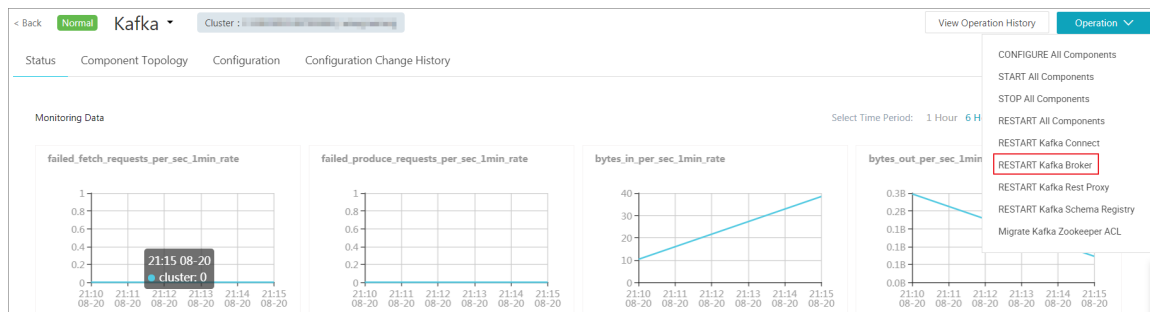
2. You can check the progress by clicking **View Operation History** at the upper right corner of the page.

- **Restart Kafka Broker**

After the preceding task is completed, it is necessary to restart the broker to make it take effect.

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

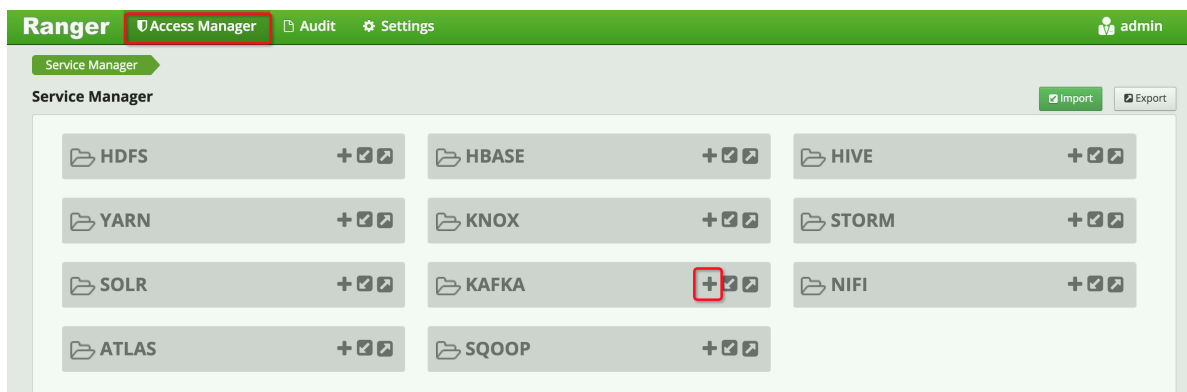




- **Add Kafka service on Ranger WebUI**

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

Add the Kafka service on **Ranger** WebUI:



Configure Kafka Service:

## Permissions configuration examples

The preceding section has integrated Ranger into Kafka, which allows you to set relevant permissions.



### Note:

In a standard cluster, Ranger generates the **all - topic** rule by default after the Kafka service is added. This rule means no restriction on permissions (that is, allow all users to perform all actions). In this case, Ranger cannot identify permissions through the user.

Use user test as an example to add the Publish permission:

**Ranger** Access Manager Audit Settings

Service Manager > emr-kafka Policies > Create Policy

### Create Policy

**Policy Details :**

Policy Type: **Access**

Policy Name \*: user\_test **enabled**

Topic \*: test **include**

Audit Logging: **YES**

Description:

Policy Label: Policy Label

**Allow Conditions :**

Select Group	Select User	Policy Conditions	Add Conditions	Add Permissions	Delegate Admin
Select Group	test		+	+	

add/edit permissions

- ☒ Publish
- ☐ Consume
- ☐ Configure
- ☐ Describe
- ☐ Create
- ☐ Delete
- ☐ Kafka Admin
- ☐ Select/Deselect All

After you add a policy by following the preceding steps, the permissions are granted to user **test**. The **test** user can perform the write operation on the topic of **test**.



### Note:

The policy takes effect 1 minute later after it is added.

## 11.6.4 Kafka SSL

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

### Create a cluster

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

### Enable the SSL service

The SSL function is not enabled for the Kafka cluster by default. You can enable SSL on the configuration page of the Kafka service.

The screenshot shows the E-MapReduce console interface for configuring the Kafka service. At the top, there's a navigation bar with '< Back', 'Normal', 'Kafka', and 'Cluster'. Below this, there are tabs for 'Status', 'Component Topology', 'Configuration', and 'Configuration Change History'. The 'Configuration' tab is active. On the left, there's a 'Quick Configuration' section with various configuration type buttons like 'BASIC', 'ADVANCED', 'INFORMATION', 'DATA\_PATH', etc. The main area is 'Service Configuration' for 'server.properties'. It lists several properties: 'log.flush.interval.messages' (10000), 'kafka.ssl.enable' (true), 'log.flush.interval.ms' (1000), 'min.insync.replicas' (1), and 'zookeeper.connection.timeout.ms' (6000). The 'kafka.ssl.enable' value is highlighted with a red box. At the top right of the configuration area, there are buttons for 'Restart Related Services', 'Apply Configuration to Host', and 'Save'. The 'Restart Related Services' button is also highlighted with a red box.

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. Use 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.

Use the producer and consumer programs in the 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 guide

E-MapReduce 3.4.0 and later versions support the Kafka Manager service for managing Kafka clusters.

### Procedure



**Note:**


By default, the Kafka Manager software is installed 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 you access the Web page through the SSH tunnel. See [Connect to the cluster using SSH](#) User Guide > Connect to Cluster through SSH for more information.
- Access *Cite Left*`http://localhost:8085`*Cite Right*.
- Enter your username and password. For more information, see 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="REDACTED"/>
kafka_manager_authentication_password	<input type="text" value="REDACTED"/>

- 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 you enable the JMX function.

 **Kafka Manager** Cluster ▾

[Clusters](#) / [Add Cluster](#)

← **Add Cluster**

**Cluster Name**

**Cluster Zookeeper Hosts**

**Kafka Version**

☒ Enable JMX Polling (Set JMX\_PORT env variable before starting kafka server)

- Common Kafka functions are available immediately after you create a Kafka cluster.

<div>Kafka Manager <span>default</span> Cluster ▾ Brokers Topic ▾ Preferred Replica Election Reassign Partitions Consumers</div>										
Clusters / default / Brokers										
← Brokers						Combined Metrics				
Id	Host	Port	JMX Port	Bytes In	Bytes Out	Rate	Mean	1 min	5 min	15 min
1	emr-worker-1.cluster-48286	9092	9999	0.00	0.00	Messagess in /sec	0.00	0.00	0.00	0.00
2	emr-worker-2.cluster-48286	9092	9999	0.00	0.00	Bytes in /sec	0.00	0.00	0.00	0.00
3	emr-worker-3.cluster-48286	9092	9999	0.00	0.00	Bytes out /sec	0.00	0.00	0.00	0.00
						Bytes rejected /sec	0.00	0.00	0.00	0.00
						Failed fetch request /sec	0.00	0.00	0.00	0.00
						Failed produce request /sec	0.00	0.00	0.00	0.00

## 11.6.6 Kafka Common Problems

### Kafka Common Problems

- 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 service configuration management page.
- java.net.BindException: Address already in use (Bind failed)

When you use the Kafka command line tools, you sometimes encounter the java.net. BindException: Address already in use (Bind failed) exception. 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 Druid Introduction

Druid is a distributed real-time memory analysis system launched by Metamarkets, a company that provides data analysis services to online media or advertising companies. Druid is used to resolve fast and interactive queries and analyse issues in large datasets.

#### 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, real-time data collection, and real-time data query.
- Efficient multi-tenant capability, which enables thousands of users to perform searches online at the same time.
- Strong scalability, which supports fast processing of PB-level data and 100 billion-level events, and thousands of concurrent queries per second.
- Extremely high availability and support for rolling upgrades.

#### Scenarios

Real-time data analysis is the most typical usage scenario for Druid. This scenario can cover a wide range of areas, such as:

- Real-time indicator monitoring
- Recommendation of model
- Advertisement platforms
- Search for models

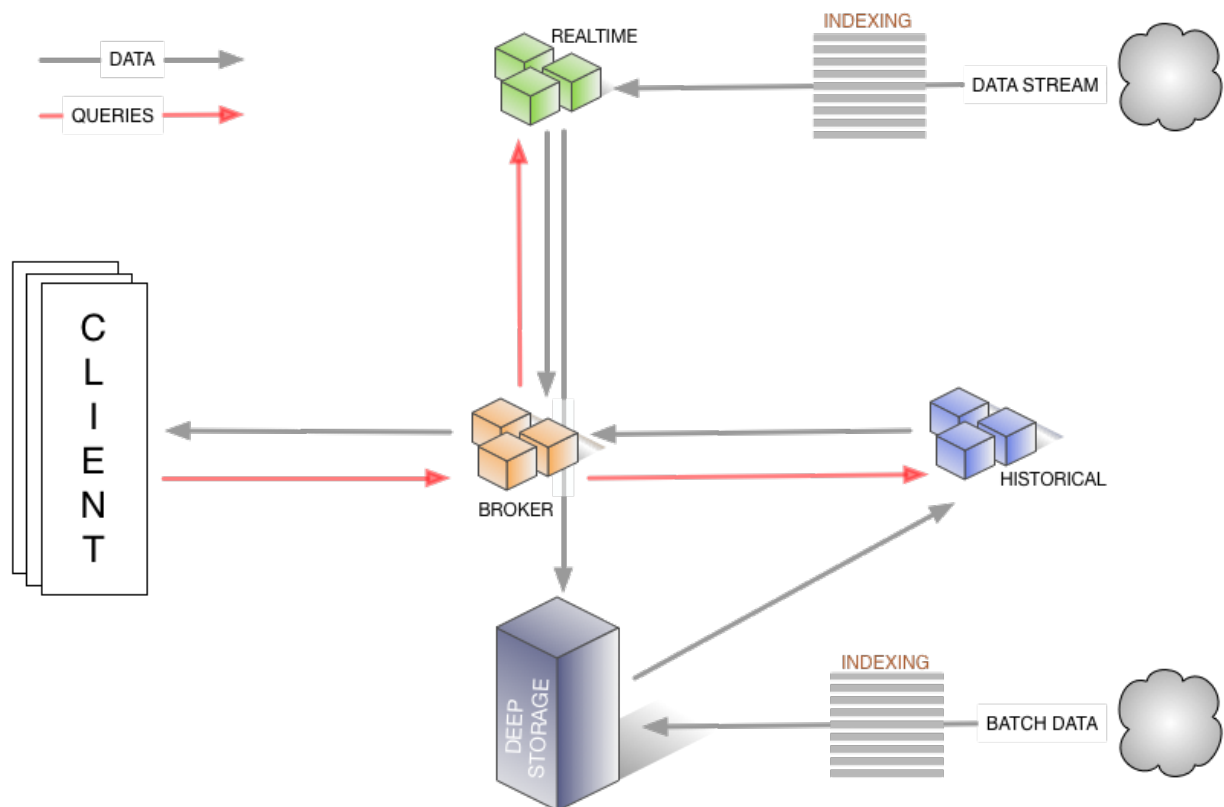
In these scenarios, there are large amounts of data, and the requirement for time delay of data query is high. In real-time indicator monitoring, system problems need to be detected at the moment of occurrence so the user can be warned. In the recommendation model, the user behavior data needs to be collected in real time, and be timely sent to the recommendation

system. After a few clicks, the system will be able to identify your search intent and recommend more reasonable results in subsequent searches.

## Druid architecture

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

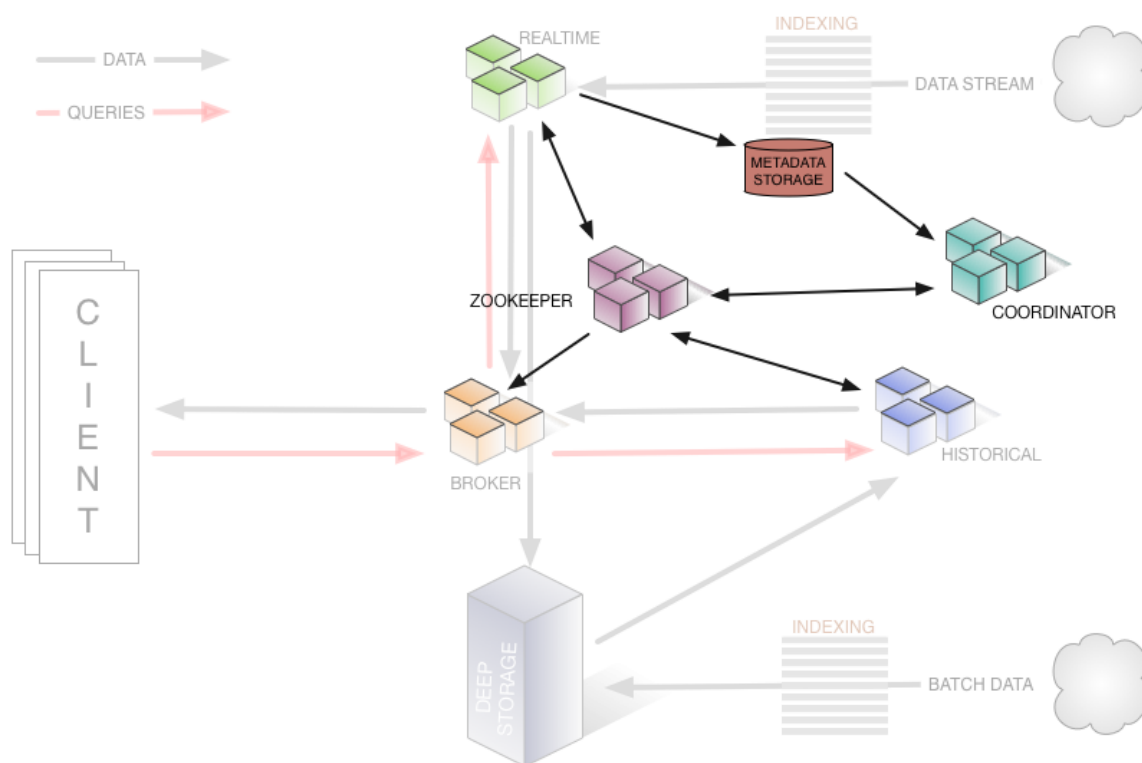
The following figure shows the components contained in the Druid working-layer (data indexing as well as data query).



- The real-time component is responsible for the real-time data collection.
- In the broker phase, query tasks are distributed, and query results are collected and returned to users.
- The historical node is responsible for the storage of the historical data after the 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 the segments, such as the download and deletion of the segments and balancing 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 is improved a lot based on the open source Druid, including integration with E-MapReduce and the peripheral ecology of Alibaba Cloud, easy monitoring and operation support, and easy-to-use product interfaces. After buying it, you can use immediately. It does not need operation and maintenance for 7x24 hours.

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 versions support Druid as a cluster type.

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

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

### Create a Druid cluster

Select 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>
```

```
</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**, **Apply Configuration to Host**, and **Restart Related Services** button 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 iZbp1e3zww
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
gw3wfnh5oii1Z
```

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/druird-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 `wikiticker` and located in `${DRUID_HOME}/quickstart`.

(`${DRUID_HOME}` is `/usr/lib/druid-current` by default.) Each line of the `wikiticke` document

(wikiticker-2015-09-12-sampled.json.gz) is a record. Each record is a json object. The format is as follows:

```
```json
{
  "Time": "2015-09-12T00: 46: 58.771Z ",
  "channel": "#en.wikipedia",
  "cityName": null,
  "comment": "added project",
  "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/wikiticker-2015-09-16-
sampled.json hdfs://emr-header-1.cluster-5678:9000/druid
```



#### Note:

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

2. 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" : 5000000
 },
 "jobProperties" : {
 "mapreduce.job.classloader": "true"
 }
},
"hadoopDependencyCoordinates": ["org.apache.hadoop:hadoop-client:2.7.2"]
}

```

**Note:**

- **spec.ioConfig.type** is set to `hadoop`.
- **spec.ioConfig.inputSpec.paths** is the path of the input file.
- **tuningConfig.type** is `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 the cluster 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" : "wikiticker",
 "intervals" : ["2015-09-12/2015-09-13"],
 "granularity" : "all",
 "dimension" : "page",
 "metric" : "edits",
 "threshold" : 25,
 "aggregations" : [
 {
 "type" : "longSum",
 "name" : "edits",
 "fieldName" : "count"
 }
]
}
```

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

```
cd ${DRUID_HOME}
curl --negotiate -u:druid -b ~/cookies -c ~/cookies -XPOST -H '
Content-Type: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 the [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 [2-5](#) 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).

## 11.8 Presto

### 11.8.1 Connector

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. |

| TABLE        | SCHEMA  | DESCRIPTION                                                                                                                                                                                                                                                                        |
|--------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 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**, respectively. 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.                                                         |
| <code>_key</code>              | 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.                            |
| <code>_key_length</code>       | 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 |
| 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 |
| hiddent    | 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:

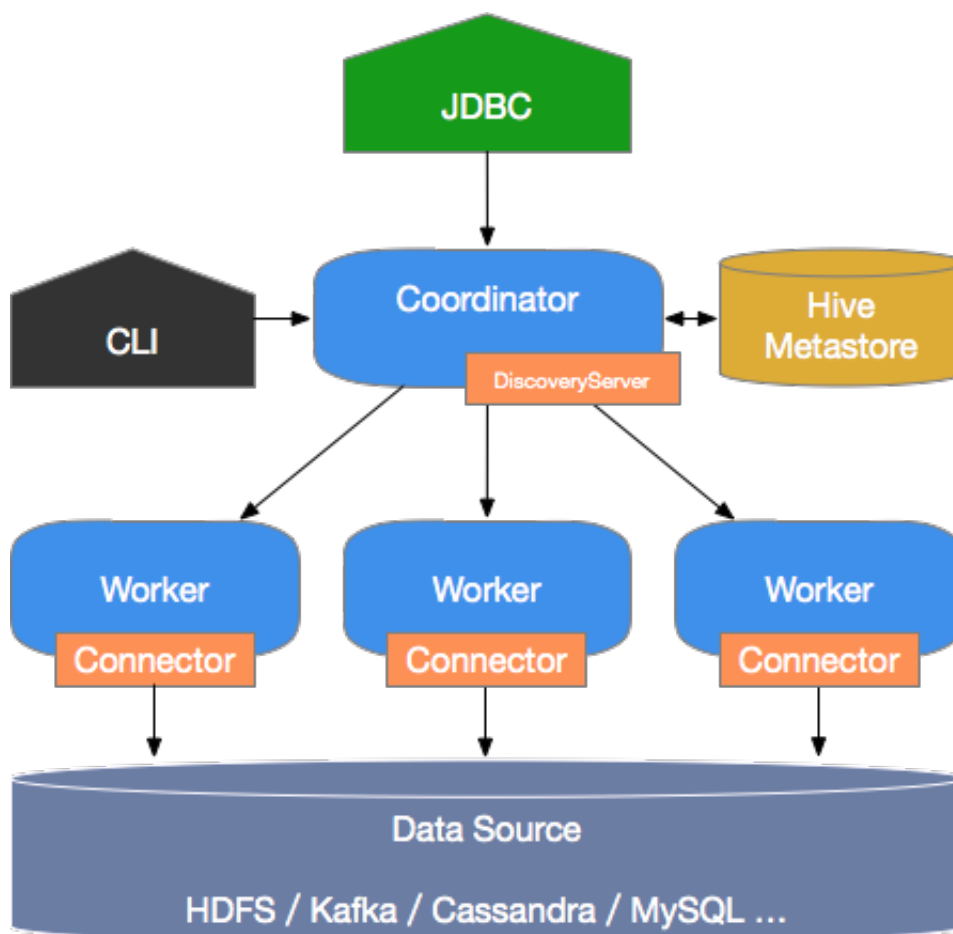
- You can purchase it for immediate use to build a Presto cluster with hundreds of nodes in minutes.
- It supports elastic resizing, you can complete up and down resizing of the cluster with simple operations.
- It works perfect in connection with the EMR software stacks, and supports processing of data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

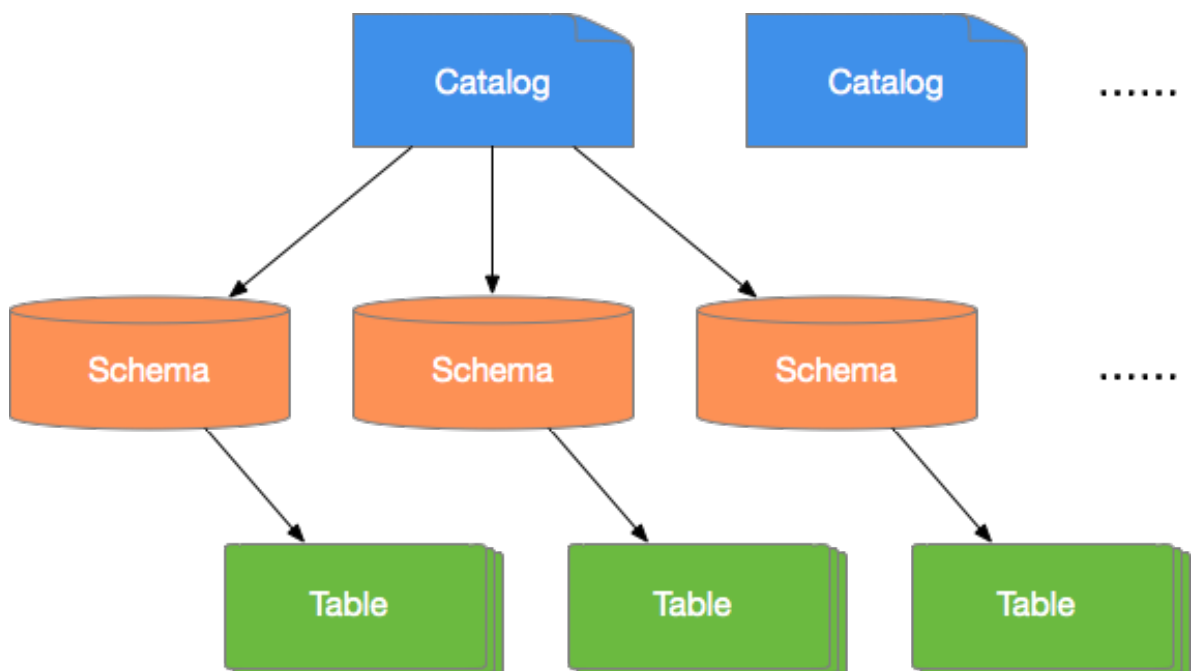
#### — Schema

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**

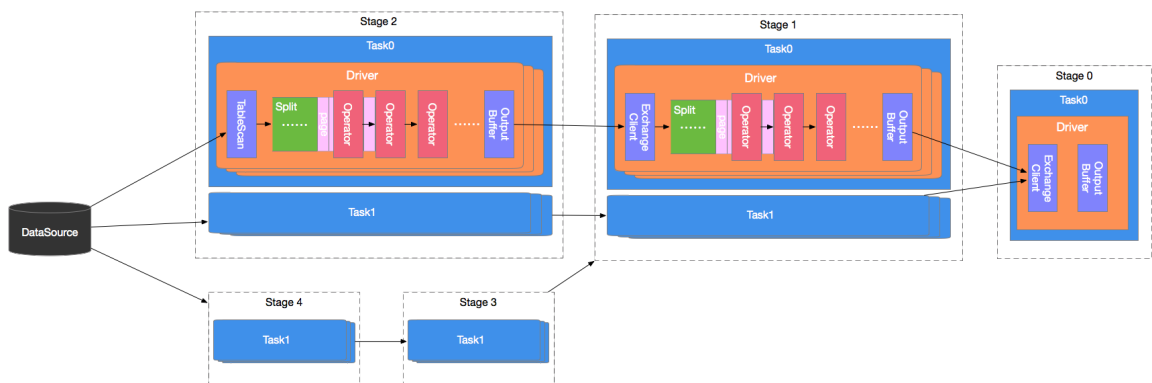
This section mainly describes related concepts in the Presto query process, for users to understand better, the execution process of Presto statements and the performance optimization methods.

#### — 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 transmit 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 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`, `VARCHAR(10)`

- CHAR

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

Example: `CHAR`, `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 <> ANY (VALUES 19, 20, 21);`, return TRUE.
- **A < ANY (...)**: Evaluates to true when A is smaller than the biggest value. Example: `SELECT 21 < ANY (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 COALESCE 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 <b>ceiling</b> .
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.
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.

Function	Syntax	Description
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 <b>power</b> .
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 <b>random</b> .
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.
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.

Function	Syntax	Description
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
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.



Function	Syntax	Description
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 '0000012345.1234500000'	DECIMAL(20, 10)

### • Operators

Arithmetic operators

Assuming x is of type DECIMAL(xp, xs) and y is of type DECIMAL(yp, ys),

- x: DECIMAL(xp,xs)
- y: DECIMAL(yp,ps)

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 =  $\max(xs, ys)$
- $x * y$

- $\text{precision} = \min(38, x_p + y_p)$
- $\text{scale} = x_s + y_s$
- $x / y$ 
  - $\text{precision} = \min(38, x_p + y_s + \max(0, y_s - x_s))$
  - $\text{scale} = \max(x_s, y_s)$
- $x \% y$ 
  - $\text{precision} = \min(x_p - x_s, y_p - y_s) + \max(x_s, y_s)$
  - $\text{scale} = \max(x_s, y_s)$
- 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	$\text{chr}(n) \rightarrow \text{varchar}$	Returns the Unicode code point <code>n</code> as a single character string.
codepoint	$\text{codepoint}(\text{string}) \rightarrow \text{integer}$	Returns the Unicode code point of the only character of <code>string</code> .
concat	$\text{concat}(\text{string1}, \dots, \text{stringN}) \rightarrow \text{varchar}$	Returns the concatenation of <code>string1</code> , <code>string2</code> , ..., <code>stringN</code> . This function provides the same functionality as the SQL-standard concatenation operator.
hamming_distance	$\text{hamming\_distance}(\text{string1}, \text{string2}) \rightarrow \text{bigint}$	Returns the <i>Hamming distance</i> of <code>string1</code> and <code>string2</code> , i.e. the number of positions at which the corresponding characters are different. Note that the two strings must have the same length.

Function Name	Syntax	Description
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 <b>search</b> from <b>string</b> .
replace	replace(string, search, replace) → varchar	Replaces all instances of <b>search</b> with <b>replace</b> in <b>string</b> .
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 <b>padstring</b> . If size is less than the length of <b>string</b> , the result is truncated to <b>size</b> characters. <b>size</b> must not be negative and <b>padstring</b> must be non-empty.
rpadd	rpadd(string, size, padstring) → varchar	Right pads string to size characters with <b>padstring</b> . If size is less than the length of <b>string</b> , the result is truncated to <b>size</b> characters. <b>size</b> must not be negative and <b>padstring</b> 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 <b>index</b> . 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.
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.

Function Name	Syntax	Description
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 <b>start</b> . 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-insensitive 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
length	length(binary) → bigint	Returns the length of binary in bytes.
concat	concat(binary1, ..., binaryN) → varbinary	Returns the concatenation of binary1, binary2, ..., binaryN.
to_base64	to_base64(binary) → varchar	Encodes binary into a base64 string representation.
from_base64	from_base64(string) → varbinary	Decodes binary data from the base64 encoded string.
to_base64url	to_base64url(binary) → varchar	Encodes binary into a base64 string representation using the URL safe alphabet.
from_base64url	from_base64url(string) → varbinary	Decodes binary data from the base64 encoded string using the URL safe alphabet.
to_hex	to_hex(binary) → varchar	Encodes binary into a hex string representation.
from_hex	from_hex(string) → varbinary	Decodes binary data from the hex encoded string.
to_big_endian_64	to_big_endian_64(bigint) → varbinary	Encodes bigint in a 64-bit 2's complement big endian format.
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 <a href="#">IEEE 754</a> single-precision floating-point format.

Function	Syntax	Description
to_ieee754_64	to_ieee754_64(double) → varbinary	Encodes double in a 64-bit big-endian binary according to <a href="#">IEEE 754</a> 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
current_date	current_date -> date	Returns the current date as of the start of the query.
current_time	current_time -> time with time zone	Returns the current time as of the start of the query.
current_timestamp	current_timestamp -> timestamp with time zone	Returns the current timestamp as of the start of the query.
current_timezone	current_timezone() -> varchar	Returns the current time zone.
date	date(x) -> date	Parses a date literal into a date
from_iso8601_timestamp	from_iso8601_timestamp(string) -> timestamp with time zone	Parses the ISO 8601 formatted string into a timestamp with time zone.
from_iso8601_date	from_iso8601_date(string) -> date	Parses the ISO 8601 formatted string into a date.
from_unixtime	from_unixtime(unixtime, [timezone_str]) -> timestamp	Returns the UNIX timestamp as a timestamp. Timestamp option is allowed.
from_unixtime	from_unixtime(unixtime, hours, minutes) -> timestamp with time zone	Returns the UNIX timestamp as a timestamp with time zone using <b>hours</b> and <b>minutes</b> for the time zone offset.
localtime	localtime -> time	Returns the current time as of the start of the query.
localtimestamp	localtimestamp -> timestamp	Returns the current timestamp as of the start of the query.
now	now() -> timestamp with time zone	Returns the current time. This is an alias for <b>current_time</b> .
to_iso8601	to_iso8601(x) -> varchar	Formats <b>x</b> as an ISO 8601 string. <b>x</b> can be <b>DATE</b> , or <b>TIMESTAMP</b> [with time zone].
to_milliseconds	to_milliseconds(interval) -> bigint	Returns the day-to-second interval as milliseconds.

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

## — 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].
%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)

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

Function	Syntax	Description
max	$\text{max}(x, n) \rightarrow \text{array}<[\text{same as } x]>$	Returns n largest values of all input values of x.
min	$\text{min}(x) \rightarrow [\text{same as input}]$	Returns the minimum value of all input values.
min	$\text{min}(x, n) \rightarrow \text{array}<[\text{same as } x]>$	Returns n smallest values of all input values of x.
sum	$\text{sum}(x) \rightarrow [\text{same as input}]$	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	$\text{histogram}(x) \rightarrow \text{map}$	Returns a map containing the count of the number of times each input value occurs.
map_agg	$\text{map\_agg}(\text{key}, \text{value}) \rightarrow \text{map}$	Returns a MAP created from the input key/value pairs.
map_union	$\text{map\_union}(x) \rightarrow \text{map}$	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	$\text{multimap\_agg}(\text{key}, \text{value}) \rightarrow \text{map}>$	Returns a multimap created from the input key/value pairs.

- **Close aggregate function**

Function	Syntax	Description
approx_distinct	$\text{approx\_distinct}(x, [e]) \rightarrow \text{bigint}$	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. This function should produce a standard error of no more than <i>e</i> , 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 <i>e</i>



Function	Syntax	Description
		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, percentage s 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, <b>accuracy</b> 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, percentage s 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. <b>buckets</b> must be a BIGINT. <b>value</b> and <b>weight</b> must be numeric. <b>weight</b> 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.
kurtosis	kurtosis(x) → double	Returns the excess kurtosis of all input values . Unbiased estimate using the following expression: $kurtosis(x) = \frac{n(n+1)}{((n-1)(n-2)(n-3))} \frac{\sum[(x_i - \text{mean})^4]}{\text{stddev}(x)^4} - \frac{3(n-1)^2}{((n-2)(n-3))}$

Function	Syntax	Description
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_aliasd (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
_coll1 | | | | 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, _col1] => [regionkey:bigint, count:bigint]
 _ Col1: = 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_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	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
WHERE regionkey = (SELECT max(regionkey) FROM region);
```

## 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:

- [Consult an expert](#)
- [Open a ticket](#)
- [Intelligent diagnosis](#)

## 11.9 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.IdapRealm.userDnTemplate** to your user DN template and **main.IdapRealm.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>
 </param>

```

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 relevant services on the EMR Service console page, such as HDFS and YARN.
  3. Click **quick link** in the upper-right corner.
- **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/](https://{cluster_access_ip}:8443/gateway/cluster-topo/hdfs/)
    - Yarn UI: [https://{cluster\\_access\\_ip}:8443/gateway/cluster-topo/yarn/](https://{cluster_access_ip}:8443/gateway/cluster-topo/yarn/)
    - SparkHistory UI : [https://{cluster\\_access\\_ip}:8443/gateway/cluster-topo/sparkhistory/](https://{cluster_access_ip}:8443/gateway/cluster-topo/sparkhistory/)
    - Ganglia UI: [https://{cluster\\_access\\_ip}:8443/gateway/cluster-topo/ganglia/](https://{cluster_access_ip}:8443/gateway/cluster-topo/ganglia/)
    - Storm UI: [https://{cluster\\_access\\_ip}:8443/gateway/cluster-topo/storm/](https://{cluster_access_ip}:8443/gateway/cluster-topo/storm/)
    - Oozie UI: [https://{cluster\\_access\\_ip}:8443/gateway/cluster-topo/oozie/](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:

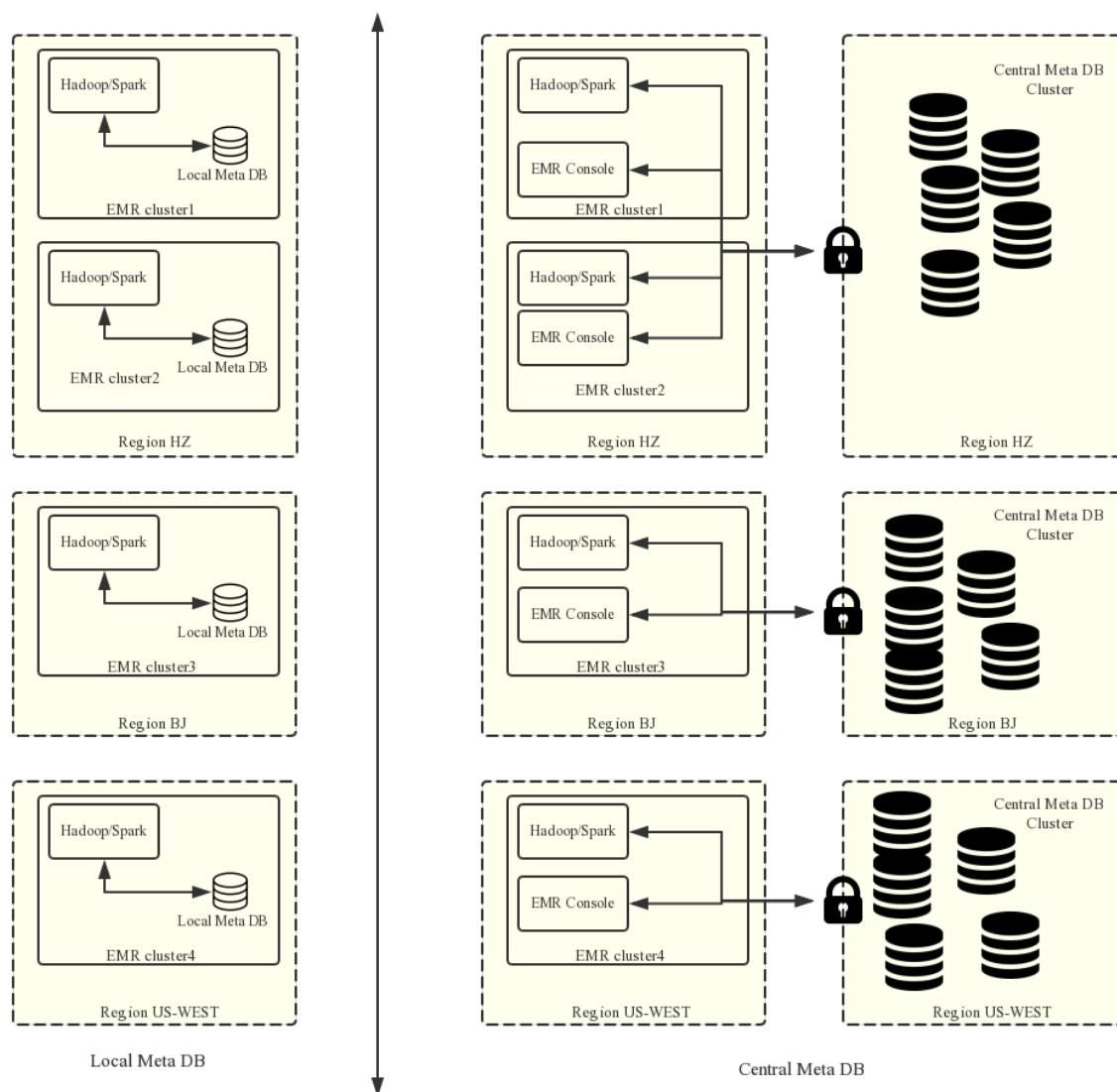
Knox provides REST APIs for operating a range of services, for example, adding or deleting HDFS files. For security, make sure the authorization object is your limited IP address range when you enable Knox Port 8443 of the security group in the ECS console. 0.0.0.0/0 is forbidden. Do not use the LDAP username and password in the Knox installation directory to access Knox.



# 12 Table mangement

## Introduction

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.



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&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;
```