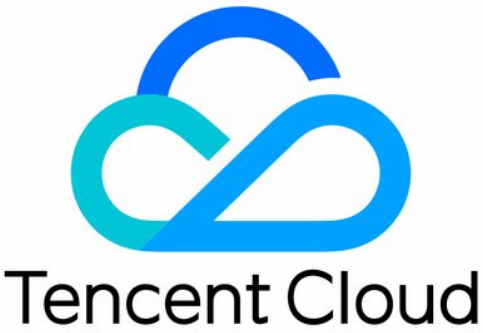
Elastic MapReduce

EMR Development Guide

Product Documentation



©2013-2024 Tencent Cloud. All rights reserved.

Copyright Notice

©2013-2025 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice

Tencent Cloud

All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

EMR Development Guide
- Hadoop Development Guide
  - HDFS Common Operations
  - HDFS Federation Management Development Guide
  - HDFS Federation Management
  - Submitting MapReduce Tasks
  - Automatically Adding Task Nodes Without Assigning ApplicationMasters
  - YARN Task Queue Management
  - Practices on YARN Label Scheduling
  - Hadoop Practical Tutorial
  - Using API to Analyze Data in HDFS and COS
  - Dumping YARN Job Logs to COS
- Spark Development Guide
  - Spark Environment Info
  - Using Spark to Analyze Data in COS
  - Using Spark Python to Analyze Data in COS
  - SparkSQL Tutorial
  - Integrating Spark Streaming with Ckafka
  - Practices on Dynamic Scheduling of Spark Resources
  - Spark Integration with Kafka
  - Spark Dependencies in Each EMR Version
- Hbase Development Guide
  - Using HBase Through API
  - Using Hbase with Thrift
  - Spark on Hbase
  - MapReduce on Hbase
- Phoenix on Hbase Development Guide
  - Phoenix Client Usage
  - Phoenix JDBC Usage
  - Phoenix Practical Tutorial
- Hive Development Guide
  - Hive Overview
  - Basic Hive Operations
    - Basic Hive Operations
    - Hive Connection Methods

©2013-2024 Tencent Cloud. All rights reserved.

- Configuring Hive Execution Engine
- Advanced Usage
  - Configuring LDAP Authentication
  - HiveServer2 CLB
  - Hive Metadata Management
  - Custom Functions UDF
- Practical Tutorial
  - Mapping Hbase Tables
  - Practices on Loading JSON Data to Hive
  - Accessing Iceberg Data with Hive
  - Accessing Hudi Data with Hive
  - Creating Databases and Tables in COS/CHDFS with Hive

Presto Development Guide
- Presto Web UI
- Connector
- Analyzing Data in COS

Sqoop Development Guide
- Import/Export of Relational Database and HDFS
- Incremental Data Import into HDFS
- Importing and Exporting Data Between Hive and TencentDB for MySQL

Hue Development Guide
- Hue Overview
- Hue Practical Tutorial

Oozie Development Guide

Flume Development Guide
- Flume Overview
- Storing Kafka Data in Hive Through Flume
- Storing Kafka Data in HDFS or COS Through Flume
- Storing Kafka Data in Hive Through Flume

Kerberos Development Guide
- Kerberos Overview

Knox Development Guide
- Knox Development Guide

Alluxio Development Guide
- Alluxio Development Documentation
- Common Alluxio Commands
- Mounting File System to Unified Alluxio File System
- Using Alluxio in Tencent Cloud

©2013-2024 Tencent Cloud. All rights reserved.

Support for COS Transparent-URI
Support for Authentication
Kylin Development Guide
Kylin Overview
Livy Development Guide
Livy Overview
Kyuubi Development Guide
Kyuubi Overview
Kyuubi Practical Tutorial
Zeppelin Development Guide
Zeppelin Overview
Zeppelin Interpreter Configuration
Hudi Development Guide
Hudi Overview
Superset Development Guide
Superset Overview
Impala Development Guide
Impala Overview
Impala OPS Manual
Analyzing Data on COS/CHDFS
Druid Development Guide
Druid Overview
Druid Usage
Ingesting Data from Hadoop in Batches
Ingesting Data from Kafka in Real Time
TensorFlow Development Guide
TensorFlow Overview
TensorFlowOnSpark Overview
Kudu Development Guide
Kudu Overview
Data Migration Guide for Kudu Node Scale-In
Ranger Development Guide
Ranger Overview
Ranger User Guide
Integrating HDFS with Ranger
Integrating YARN with Ranger
Integrating HBase with Ranger
Integrating Presto with Ranger

©2013-2024 Tencent Cloud. All rights reserved.

Ranger Audit Log Guide
Storing Ranger Audit Logs in Solr
Storing Ranger Audit Logs in Tencent Cloud ElasticSearch
Kafka Development Guide
Kafka Overview
Use Cases
Kafka Usage
Iceberg Development Guide
StarRocks Development Guide
StarRocks Overview
User Guide
Flink Development Guide
Flink Overview
Analyzing COS Data with Flink

©2013-2024 Tencent Cloud. All rights reserved.

EMR Development Guide
Hadoop Development Guide
HDFS Common Operations

Last updated：2020-12-21 17:21:45

Tencent Cloud Elastic MapReduce (EMR) Hadoop service has integrated COS. If you enable COS when purchasing an EMR cluster, you can manipulate the data stored in COS using common Hadoop commands as shown below:

```
#cat data
hadoop fs -cat /usr/hive/warehouse/hivewithhdfs.db/record/data.txt
#Modify the directory or file permission
hadoop fs -chmod -R 777 /usr
#Change the file or directory owner
hadoop fs -chown -R root:root /usr
#Create a folder
hadoop fs -mkdir <paths>
#Send a local file to HDFS
hadoop fs -put <localsrc> ... <dst>
#Copy a local file to HDFS
hadoop fs -copyFromLocal <localsrc> URI
#View the storage usage of a file or directory
hadoop fs -du URI [URI …]
#Delete a file
hadoop fs -rm URI [URI …]
#Set the number of copies of a directory or file
hadoop fs-setrep [-R] [-w] REP PATH [PATH …]
#Check for bad blocks of a cluster file
hadoop fsck <path> [-move | -delete | -openforwrite] [-files [-blocks [-locations |
```

For more HDFS commands, see the community documentation. If your cluster is a high-availability cluster (dual-namenode), you can see which namenode is active by running following commands:

```
#nn1 is the ID of the namenode; usually nn1 and nn2.
hdfs haadmin -getServiceState nn1
#View the current cluster report
hdfs dfsadmin -report
#namenode exits safe mode
hdfs dfsadmin -safemode leave
```

©2013-2024 Tencent Cloud. All rights reserved.

HDFS Federation Management Development Guide

Last updated：2022-05-16 12:52:25

Type Selection for HDFS Federation Management

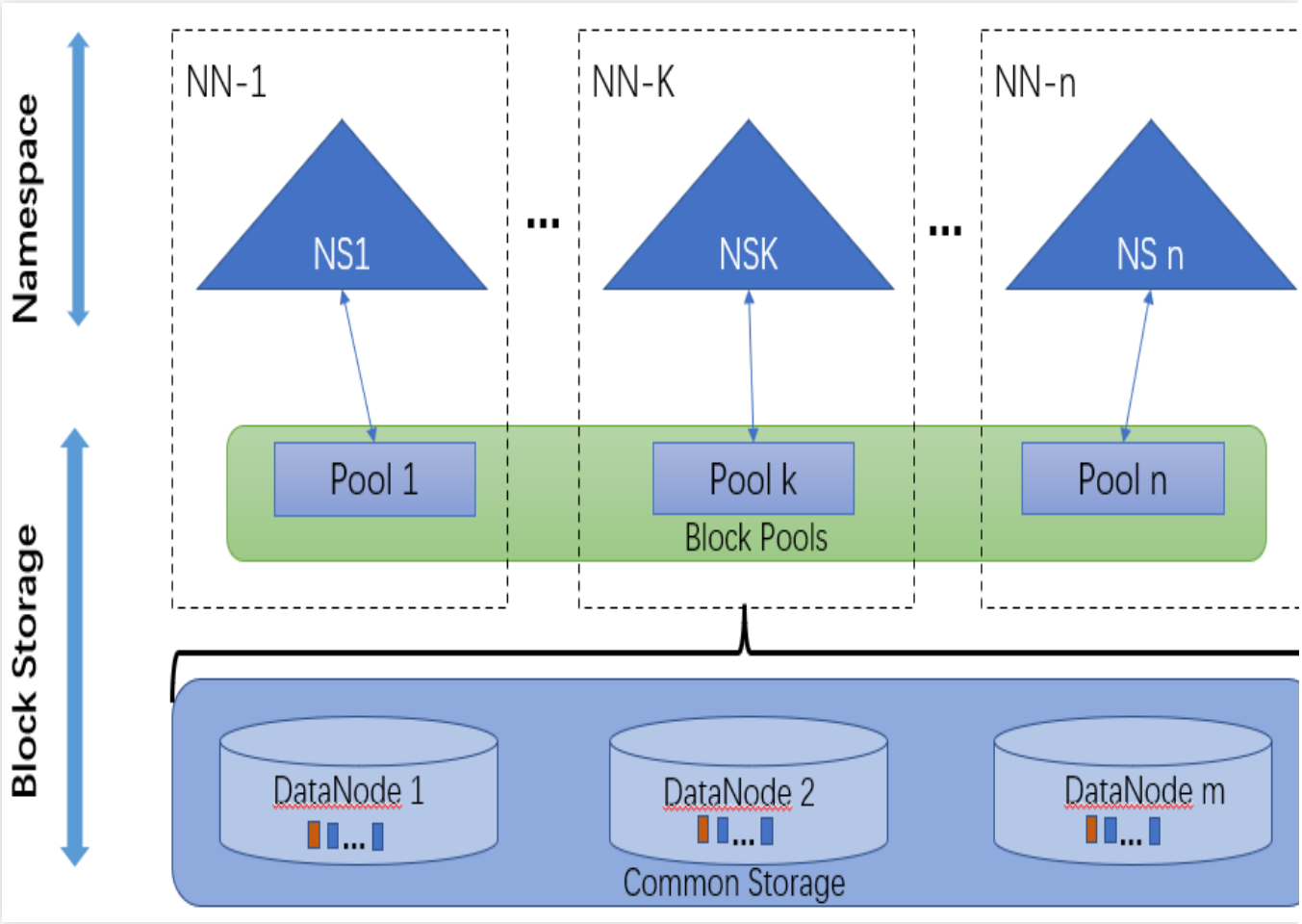
Note:

The HDFS federation management feature is currently made available through an allowlist. To use it, submit a ticket for application.

HDFS federation management architecture

In order to scale the name service horizontally, HDFS federation uses multiple independent NameNodes/nameSpaces. The DataNodes are used as common storage for blocks by all the NameNodes. Each DataNode registers with all the NameNodes in the cluster. DataNodes send periodic heartbeats and block reports. They also handle commands from the NameNodes.

©2013-2024 Tencent Cloud. All rights reserved.



For more information, see HDFS Federation.

How ViewFs federation works

To make it easier to manage multiple namespaces, HDFS ViewFs federation uses the classic client-side mount table (a feature of open-source ViewFs). Different paths of applications are mapped to specific NameServices on the client, so as to achieve storage separation or performance separation. This mechanism distributes files and loads but requires more human intervention (clear planning) to implement ideal load balancing.

How router-based federation works

Router-based federation provides a software layer to manage multiple namespaces. Compared with ViewFs which maintains the mount table information on the client, it is truly transparent to the client, because the mapping information will be additionally saved and persisted.

HDFS Federation Management Configuration

You need to consider two factors: the required number of NameServices and the mount method of the business data directory.

©2013-2024 Tencent Cloud. All rights reserved.

Planning principles for the number of NameServices

1. One NameService can store up to 100 million files securely. After this limit is exceeded, its access speed and read/write throughput will be greatly reduced. Therefore, if you expect to store more than 100 million files, you need to add an extra NameService.
2. For an application that reads/writes HDFS heavily, it is necessary to allocate a separate NameService to process its requests. This ensures that the data in the application can exclusively use all the processing capabilities of the NameService and avoids the impact on other applications.
3. For an application that demands high reliability, you can allocate an independent NameService to it, because it may be prevented from accessing HDFS and thus rendered unstable if the reads/writes of other applications are too frequent.

Planning principles for the mount method of business data directories

1. Mount the data directories of the services associated with the business data to the same NameService, because cross-NameService file reads/writes are slow and may reduce the application's file storage performance.
2. For a service that involves a large amount of data but is not associated with other services, you can directly use only one NameService.
3. For a small business, we recommend you mount its directories directly to the default NameService (HDFS${clusterid}), so as to eliminate the need of data migration and simplify the federation configuration.
4. We recommend you map only first-level directories to NameServices in order to reduce the configuration complexity.

Scheme Comparison

ViewFs federation configuration method

1. Directly use ViewFs
Strengths: Unified views are provided, and different applications can be used in the same way.
Shortcomings: Changes to ViewFs mount tables require all applications using the cluster to read the latest mount point synchronously.
2. Specify the NameService
Strengths: You don't need to modify the configurations of all applications.
Shortcomings: You need to specify directories for different components and applications, which makes the coupling of component paths more complicated.

Router-based federation configuration method

1. Directly use router-based federation
Strengths: Unified views are provided, and different applications can be used in the same way. Changes to router-based federation mount tables take effect directly, so applications using the cluster don't need to update the mount

©2013-2024 Tencent Cloud. All rights reserved.

tables synchronously.

Shortcomings: The DFSRouter forwarding layer is added, which slightly affects the performance.

2. Specify the NameService

Strengths: You don't need to modify the configurations of all applications.

Shortcomings: You need to specify directories for different components and applications, which makes the coupling of component paths more complicated.

©2013-2024 Tencent Cloud. All rights reserved.

HDFS Federation Management

Last updated：2023-03-15 10:12:39

Overview

HDFS federation management is an HDFS federated cluster deployment and management feature, including NameService management and mount table management. Federation management is supported for Hadoop-type clusters in HA mode. There are two federation types to choose from: ViewFs federation and router-based federation, and the federation type cannot be changed once selected. A router node will be used to deploy an added NameNode. This router node does not support termination and role start/stop at the node level.

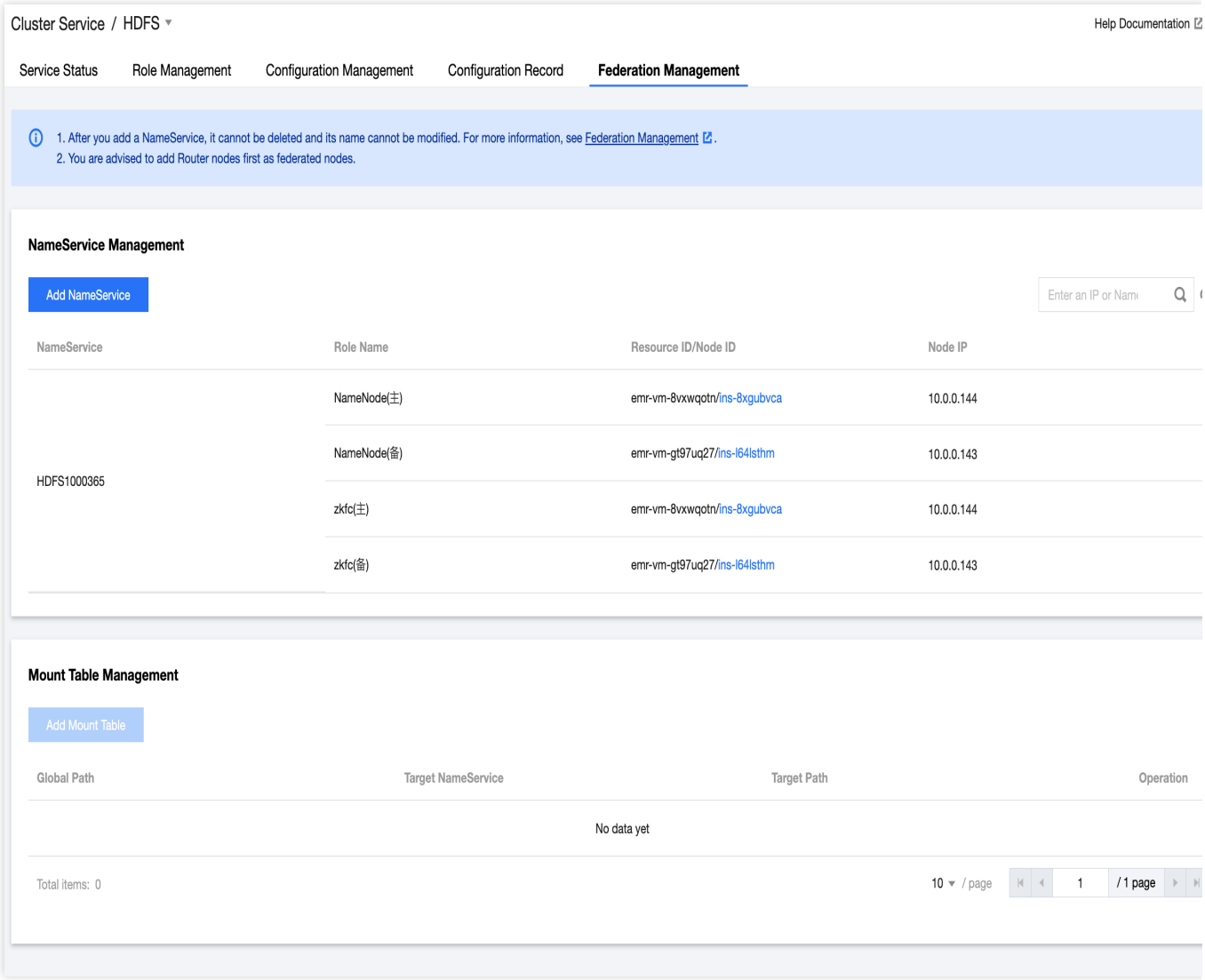
Note:

1. The HDFS federation management feature is currently made available through an allowlist. To use it, submit a ticket for application.
2. All EMR versions support ViewFs federation. As only HDFS v2.9.0 and later support router-based federation, only EMR v3.x.x and later support router-based federation.
3. Suspending the NameNode role process on a federated node on the **Role Management** page may affect the cluster scaling, so you need to resume the process first before scaling the cluster.

Directions

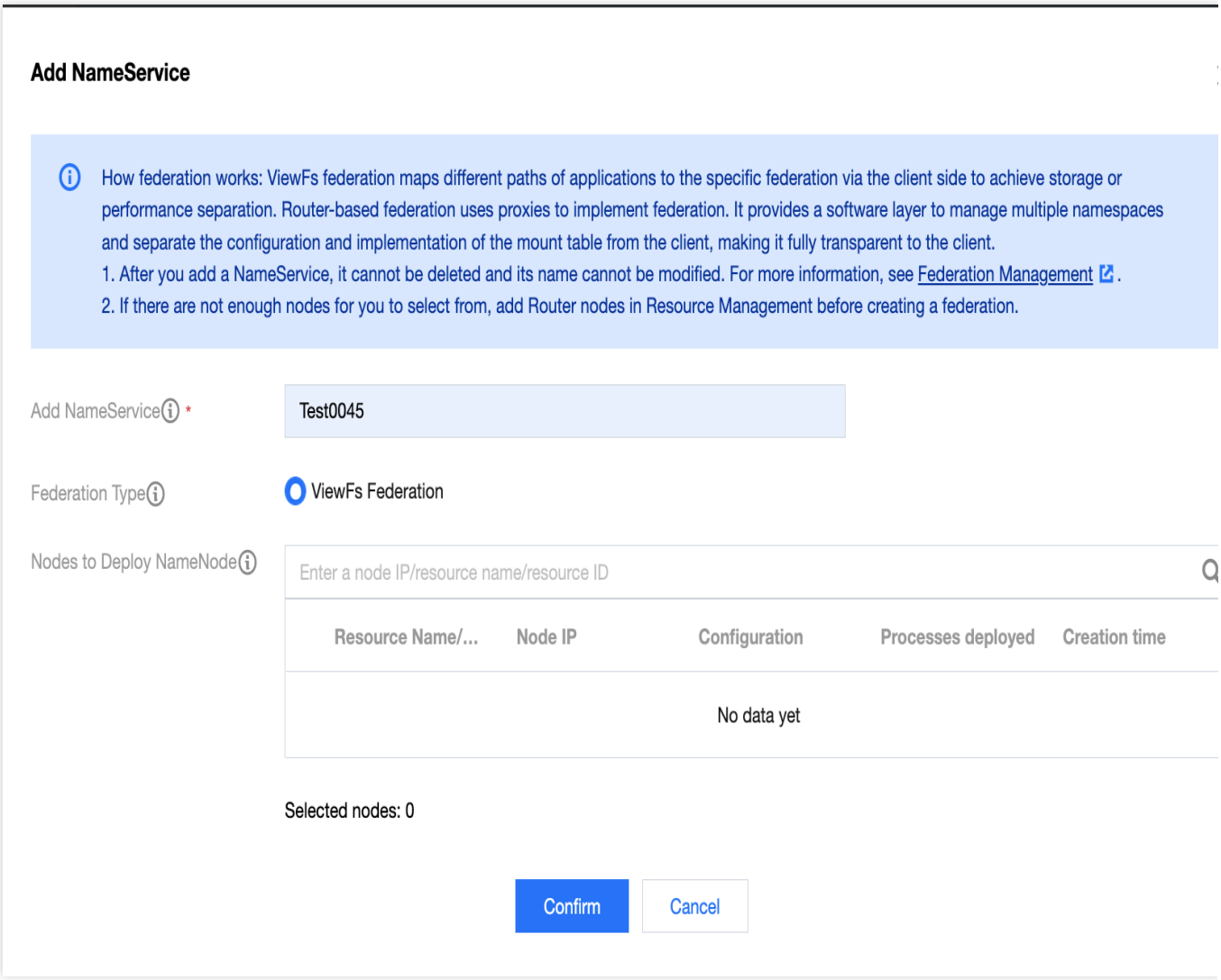
1. Log in to the EMR console and click the **ID/Name** of the target cluster in the cluster list to go to the cluster details page.
2. On the cluster details page, click **Cluster Service** and select **Operation** > **Federation Management** in the top-right corner of the HDFS component block to enter the **Federation Management** page.

©2013-2024 Tencent Cloud. All rights reserved.



3. Click **Add NameService** to create an HDFS federation. You need to enter the NameService name and select the federation type, NameNode, and DFSRouter (for router-based federation only).

©2013-2024 Tencent Cloud. All rights reserved.

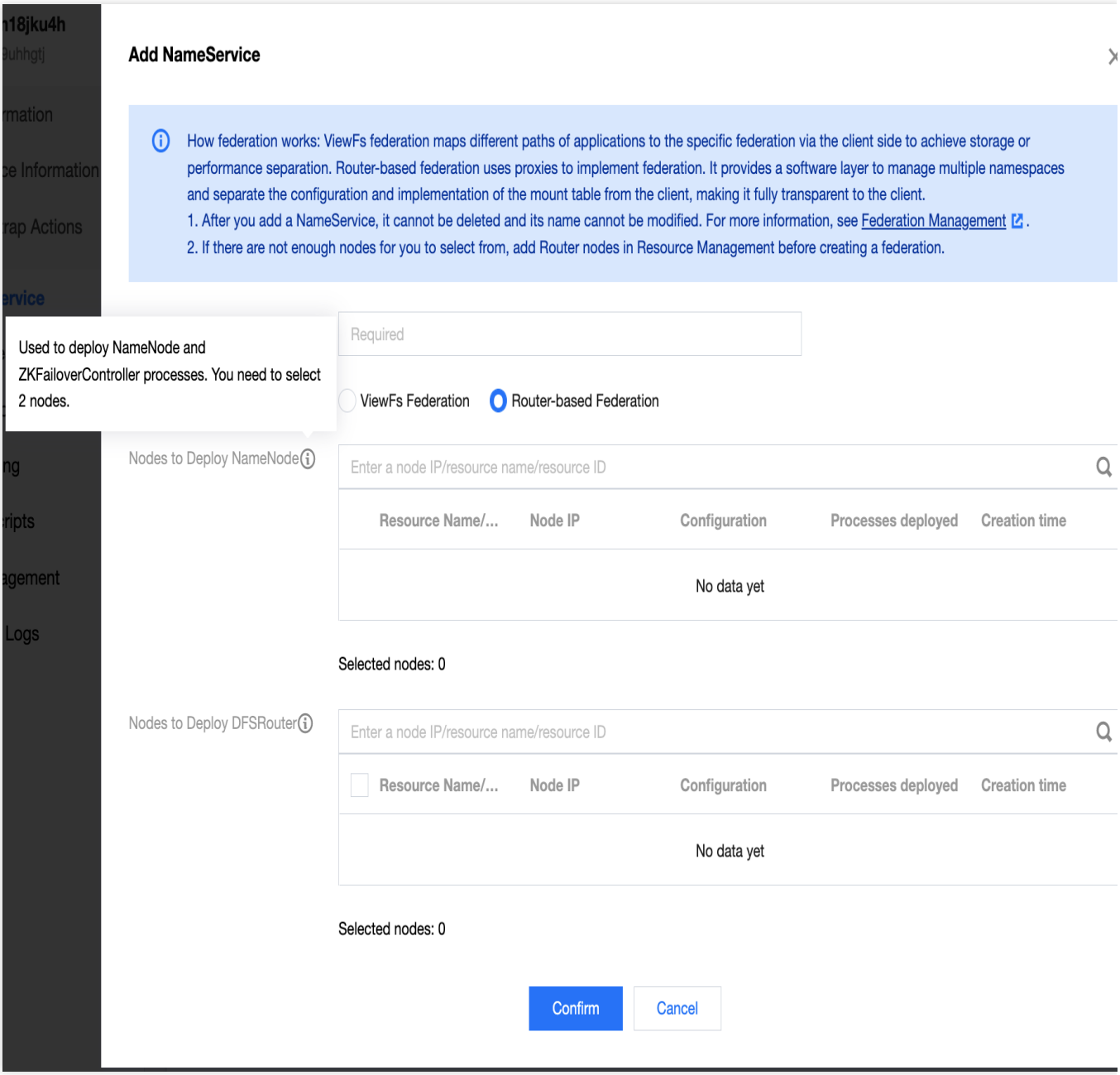


4. Select **Add Federated Node**.

Federated nodes adopt router nodes in the cluster. Therefore, you need to add a router node on the **Resource Management** page first and then set it to a federated node. The NameNode process requires two nodes, on each of which the NameNode and ZKFC processes will be deployed.

When creating a router-based federation for the first time, you need to select at least two nodes to deploy the DFSRouter process. When creating another federation, you can reuse the DFSRouter nodes, and the number of the nodes can be greater than or equal to zero.

©2013-2024 Tencent Cloud. All rights reserved.



Note:

For HDFS versions earlier than 3.3.0, when you successfully add a NameService to a router-based federation (**not for the first time**), you need to restart the old DFSRouter process on the **Role Management** page (preferably during off-peak hours). For HDFS v3.3.0 and later which support hot loading the configuration, this operation is not required.

After adding a federated NameService to a cluster with Kerberos enabled, you need to restart the YARN ResourceManager first (preferably during off-peak hours) before you can use the files on the new NameService for jobs submitted to YARN.

The NameService name cannot be modified or deleted once set and cannot be system keywords such as "nsfed", "haclusterX", and "ClusterX".

5. Add a mount table.

You can add a mount table only after successfully adding a NameService. To reduce the configuration complexity, we

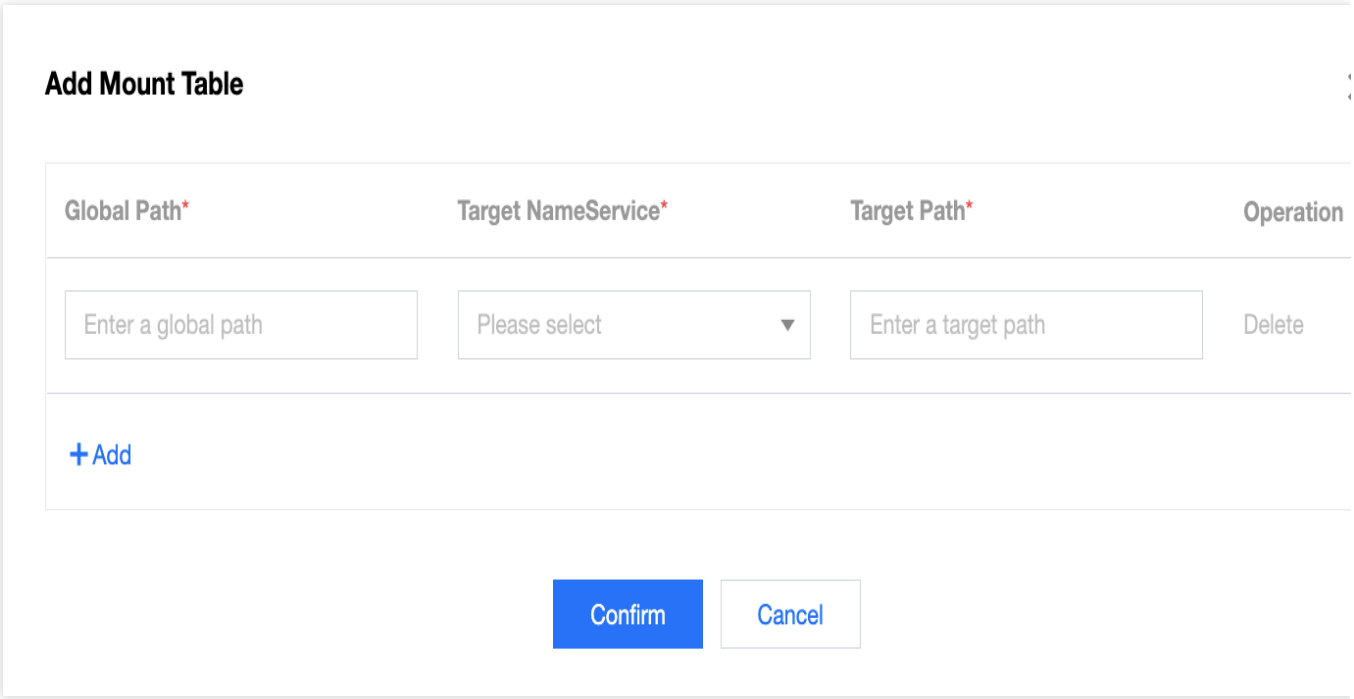
©2013-2024 Tencent Cloud. All rights reserved.

recommend you map only first-level directories such as "/tmp", "/user", and "/srv" to NameServices. You can add multiple mount paths at a time.

Path: Path name of the unified ViewFs or router-based federation namespace, also known as the mount point.

Target NameService: The NameService corresponding to the real path to which the mount point maps.

Target path: The real path on the corresponding NameService, whose name can be different from that of the global path.



Note:

Path direction:

1.1 Log in to the NameNode and run `hdfs dfs -ls /` to point to the path under the namespace managed by the NameNode. For ViewFs federation, you need to use `hdfs dfs -ls ViewFs://ClusterX/` to point to the global path; for router-based federation, use `hdfs dfs -ls hdfs://nsfed/` instead.

1.2 Log in to another node, such as the router node serving as a client. `hdfs dfs -ls /` points to the global path.

The data of all business components needs to be placed in first-level directories but not the root directory for access, as the root directory cannot be mounted.

The default NameService has the `/emr` directory, which needs to be mounted.

©2013-2024 Tencent Cloud. All rights reserved.

Submitting MapReduce Tasks

Last updated：2020-12-15 15:24:37

This operation guide describes: 1. How to perform basic MapReduce job operations in command-line interfaces. 2. How to allow MapReduce jobs to access to the data stored in COS. For more information, please see the community documentation.

The job submitted is a wordcount job. To count the words in a file, you need to upload the specified file in advance.

The path of relevant software such as Hadoop is `/usr/local/service/` .

The relevant logs are stored in `/data/emr` .

1. Preparations for Development

You need to create a bucket in COS for this job.

Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, select **Enable COS** on the basic configuration page and then enter your SecretId and SecretKey. You can find your SecretId and SecretKey at API Key Management. If you don't have a key, click **Create Key** to create one.

2. Logging in to an EMR Server

You need to log in to any server in the EMR cluster first before performing the relevant operations. A master node is recommended for this step. EMR is built on CVM instances running on Linux; therefore, using EMR in command line mode requires logging in to an CVM instance.

After creating the EMR cluster, select Elastic MapReduce in the console. In **Cluster Resource** > **Resource Management**, click **Master Node** to select the resource ID of the master node. Then, you can enter the CVM Console and find the instance of the EMR cluster.

For more information on how to log in to a CVM instance, please see Logging in to a Linux Instance. Here, you can choose to log in with WebShell. Click **Login** on the right of the desired CVM instance to enter the login page. The default username is root, and the password is the one you set when creating the EMR cluster.

Once your credentials have been validated, you can access the EMR command-line interface. All Hadoop operations are under the Hadoop user. The root user is logged in by default when you log in to the EMR server, so you need to switch to the Hadoop user. Run the following command to switch users and go to the Hadoop folder:

```
[root@172 ~]# su hadoop
[hadoop@172 root]$ cd /usr/local/service/hadoop
[hadoop@172 hadoop]$
```

©2013-2024 Tencent Cloud. All rights reserved.

3. Data Preparations

You need to prepare a text file for counting. There are two ways to do so: **storing data in an HDFS cluster** and **storing data in COS**.

The first step is to upload a local file to the CVM instance of the EMR cluster with the scp or sftp service. Run the following command on the local command line:

```
scp $localfile root@public IP address:$remotefolder
```

Here, $localfile is the path and the name of your local file; root is the CVM instance username. You can look up the public IP address in the node information in the EMR or CVM Console; and $remotefolder is the path where you want to store the file in the CVM instance.

After the upload succeeds, you can check whether the file is in the corresponding folder on the command line of the EMR cluster.

```
[hadoop@172 hadoop]$ ls -l
```

Storing data in HDFS

After uploading the data to the CVM instance, you can copy it to the HDFS cluster. The README.txt file in the `/usr/local/service/hadoop` directory is used here as an example. Copy the file to the Hadoop cluster by running the following command:

```
[hadoop@172 hadoop]$ hadoop fs -put README.txt /user/hadoop/
```

After the copy is completed, run the following command to view the copied file:

```
[hadoop@172 hadoop]$ hadoop fs -ls /user/hadoop
Output:
-rw-r--r-- 3 hadoop supergroup 1366 2018-06-28 11:39 /user/hadoop/README.txt
```

If there is no `/user/hadoop` folder in Hadoop, you can create it on your own by running the following command:

```
[hadoop@172 hadoop]$ hadoop fs -mkdir /user/hadoop
```

For more Hadoop commands, please see Common HDFS Operations

Storing data in COS

There are two ways to store data in COS: **uploading through the COS Console from the local file system** and **uploading through Hadoop command from the EMR cluster**.

When uploading through the COS Console from the local file system, if the data file is already in COS, you can view it by running the following command:

©2013-2024 Tencent Cloud. All rights reserved.

```
 [hadoop@10 hadoop]$ hadoop fs -ls cosn://$bucketname/README.txt
-rw-rw-rw- 1 hadoop hadoop 1366 2017-03-15 19:09 cosn://$bucketname /README.txt
```

Replace $bucketname with the name and path of your bucket.

To upload through Hadoop command from the EMR cluster, run the following command:

```
[hadoop@10 hadoop]$ hadoop fs -put README.txt cosn:// $bucketname /
[hadoop@10 hadoop]$ bin/hadoop fs -ls cosn:// $bucketname /README.txt
-rw-rw-rw- 1 hadoop hadoop 1366 2017-03-15 19:09 cosn://$bucketname /README.txt
```

4. Submitting a Job Through MapReduce

The job submitted this time is the wordcount routine that comes with the Hadoop cluster, which has already been compressed into a .jar package and uploaded to the created Hadoop cluster for direct call and use.

Counting a text file in HDFS

Go to the `/usr/local/service/hadoop` directory as described in data preparations, and submit the job by running the following command:

```
[hadoop@10 hadoop]$ bin/yarn jar ./share/hadoop/mapreduce/hadoop-mapreduce-
examples-2.7.3.jar wordcount
/user/hadoop/README.txt /user/hadoop/output
```

Note:

In the complete command above, `/user/hadoop/README.txt` is the input file to be processed, and `/user/hadoop/output` is the output folder. You should make sure that there is no output folder before submitting the command; otherwise, the submission will fail.

After the execution is completed, view the output file by running the following command:

```
[hadoop@10 hadoop]$ bin/hadoop fs -ls /user/hadoop/output
Found 2 items
-rw-r--r-- 3 hadoop supergroup 0 2017-03-15 19:52 /user/hadoop/output/_SUCCESS
-rw-r--r-- 3 hadoop supergroup 1306 2017-03-15 19:52 /user/hadoop/output/part-
r-00000
```

View the statistics in part-r-00000 by running the following command:

```
[hadoop@10 hadoop]$ bin/hadoop fs -cat /user/hadoop/output/part-r-00000
(BIS),   1
(ECCN)   1
(TSU)    1
(see     1
```

©2013-2024 Tencent Cloud. All rights reserved.

```
5D002.C.1,      1
740.13) 1
<http://www.wassenaar.org/>     1
……
```

Counting a text file in COS

Go to the `/usr/local/service/hadoop` directory and submit the job by running the following command:

```
[hadoop@10 hadoop]$ bin/yarn jar ./share/hadoop/mapreduce/hadoop-mapreduce-
examples-2.7.3.jar  wordcount
cosn://$bucketname/README.txt /user/hadoop/output
```

The input file for the command is changed to `cosn:// $bucketname /README.txt` , which indicates to process the file in COS, where $bucketname is your bucket name and path. The result is still outputted to the HDFS cluster, but you can also choose to output to COS. The way to view the output is the same as above.

Viewing job logs

```
# View job status
bin/mapred job -status jobid
# View job logs
yarn logs -applicationId id
```

©2013-2024 Tencent Cloud. All rights reserved.

Automatically Adding Task Nodes Without Assigning ApplicationMasters

Last updated：2022-05-16 12:52:25

Overview

In the automatic scaling scenario, when a scale-in rule is triggered, if an ApplicationMaster (AM) is running on the task node to be terminated, the running AM will also be terminated, which will cause the current job to fail.
No AMs will be assigned to task nodes automatically added through scale-out by default. This ensures that when the nodes are removed, running AMs will not be terminated, so that jobs can run properly.

Feature

No AMs will be assigned to task nodes automatically added through the transformation of YARN source code and the addition of configuration items to the automatic scaling process. Instead, all AMs will be assigned to those that are not automatically added. Automatically added task nodes are responsible for compute tasks only. Therefore, an automatic scale-in action will only terminate the nodes processing compute tasks, while the AMs remain active and will try to take over the compute tasks on other nodes to keep the current job running.

Application Scope

This only applies to task nodes automatically added.

©2013-2024 Tencent Cloud. All rights reserved.

YARN Task Queue Management

Last updated：2025-04-15 16:35:57

You can log in to the YARNwebUI, which is the web user interface of YARN, through the shortcut entry provided in EMR. For more information, please see the Software WebUI Entry. After logging in, you will see the following:

You can see some monitoring information of the entire cluster here:
Apps: 67 submitted apps, including 9 pending ones, 7 running ones, and 51 completed ones, where 15 containers are running.
Memory: 36 GB in total with 30 GB used and 10 GB reserved.
Virtual cores: 20 cores in total with 15 used ones and 5 reserved ones.
Nodes: 5 available nodes, 0 decommissioned ones, 0 lost ones, 0 unhealthy ones, and 0 rebooted ones.
Scheduler: Fair Scheduler, including maximum and minimum memory size and CPU allocation information.
The Application Queues section contains job queuing information of the cluster. Take the root.hadoop queue as an example:
Used resources: 30,720 MB (30 GB) memory, 15 virtual cores.
Active applications: 7.
Pending applications: 9.
Minimum memory occupied: 0 MB, 0 cores.
Maximum memory occupied: 36,860 MB, 20 cores.
Steady fair share.
Instantaneous fair share.
The proportion of memory occupied by this queue is 83.3%.

©2013-2024 Tencent Cloud. All rights reserved.

Practices on YARN Label Scheduling

Last updated：2023-03-15 10:12:39

Overview

Spark on YARN uses YARN as the resource scheduler. Since Hadoop 2.7.2, YARN provides label-based scheduling on top of Capacity Scheduler.
Capacity Scheduler is a multi-tenant resource scheduler. It enables organizations in a Hadoop cluster to share available resources and provide computing power to the cluster based on their respective computing needs. Capacity Scheduler supports features such as hierarchical queue, capacity guarantee, security, resource elasticity, multi-tenancy, resource-based scheduling, and mapping. All resources of a cluster are allocated to queues, so that all applications submitted to a queue can use the resources allocated to the queue. Idle resources of organizations can be non-preemptively allocated to applications in queues running below the capacity. This ensures that an application runs with the minimum resource capacity while idle resources are elastically allocated to other applications as needed.
Capacity Scheduler roughly allocates cluster resources to queues, without specifying the location of applications in queues. Label-based scheduling is provided to allocate resources at a more refined granularity by assigning a node label to each node of the cluster, so that the location of applications can be specified. Node labels have the following characteristics:
1. A node has only one node label, which means that a node belongs to only one partition. A cluster is divided into multiple disjoint subclusters by node partition.
2. Node partitions are classified into exclusive and non-exclusive partitions based on their matching strategies. An exclusive node partition allocates containers only to nodes that exactly match the partition. A non-exclusive node partition shares idle resources to containers that request the default partition.
3. You can set accessible node labels for each queue to specify the node partitions in which applications can run.
4. You can set the resource percentage for different queues in each node partition.
5. If the required node label is specified in a resource request, the resource is allocated only from nodes with the specified label. If no node label is specified (the default label name of a queue can be modified by using a configuration item), the resource is allocated only from nodes that belong to the default partition.

Configuration

1. Setting ResourceManager to enable Capacity Scheduler

Label-based scheduling can be used only in conjunction with Capacity Scheduler. Capacity Scheduler is the default scheduler of YARN. If you are using another scheduler, specify the following settings in `${HADOOP_HOME}/etc/hadoop/yarn-site.xml` to enable Capacity Scheduler first:

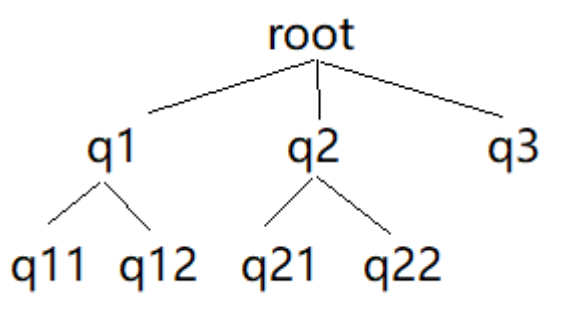
©2013-2024 Tencent Cloud. All rights reserved.

```
<property>
    <name>yarn.resourcemanager.scheduler.class</name>

<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.Capacit
yScheduler</value>
</property>
```

2. Setting the parameters of Capacity Scheduler

Set `yarn.scheduler.capacity.root` to specify the predefined root queue of Capacity Scheduler in `${HADOOP_HOME}/etc/hadoop/capacity-scheduler.xml` . All other queues are subqueues of the root queue. All queues are organized in a tree structure. Set `yarn.scheduler.capacity.<queue-path>.queues` to specify subqueues under `queue-path` and separate different subqueues by commas (,).

Example:



Settings of the example structure:

```
<property>
    <name>yarn.scheduler.capacity.root.queues</name>
    <value>q1,q2,q3</value>
</property>
<property>
    <name>yarn.scheduler.capacity.root.q1.queues</name>
    <value>q11,q12</value>
</property>
<property>
    <name>yarn.scheduler.capacity.root.q2.queues</name>
    <value>q21,q22</value>
</property>
```

For more information about other settings of Capacity Scheduler, see the documentation.

3. Configuring ResourceManager to enable node labels

©2013-2024 Tencent Cloud. All rights reserved.

Specify the following settings in `conf/yarn-site.xml` :

```
<property>
    <name>yarn.node-labels.fs-store.root-dir</name>
    <value>hdfs://namenode:port/path-to-store/node-labels/</value>
</property>
<property>
    <name>yarn.node-labels.enabled</name>
    <value>true</value>
</property>
<property>
    <name>yarn.node-labels.configuration-type</name>
    <value>centralized or delegated-centralized or distributed</value>
</property>
```

Note:

1. Make sure that `yarn.node-labels.fs-store.root-dir` is created and that ResourceManager has access to it.
2. You can store node labels in the local file system of ResourceManager in paths such as `file://home/yarn/node-label` . However, to ensure high availability of the cluster and to avoid label loss due to the failure of ResourceManager, we recommend that you store node labels in HDFS.
3. If you use Hadoop 2.8.2, you need to set `yarn.node-labels.configuration-type` .

4. Configuring node labels

You can configure node labels in `etc/hadoop/capacity-scheduler.xml` .

Configuration Item	Description
yarn.scheduler.capacity. `<queue-path>` .capacity	The percentage of nodes in the DEFAULT partition that are accessible to a queue. **The values of this configuration item of all direct subqueues under each parent queue must sum up to 100.**
yarn.scheduler.capacity. `<queue-path>` .accessible-node-labels	A list of labels that are accessible to a queue. Labels are separated by commas (,). For example, "HBASE,STORM" specifies that the queue can access the HBASE and STORM labels. All queues can access nodes without labels. If this configuration item is not specified for a queue, the queue inherits the value from its parent queue. If you want a queue to access only nodes without labels, leave this configuration item unspecified.
yarn.scheduler.capacity. `<queue-path>` .accessible-node-labels. `<label>` .capacity	The percentage of nodes in the `<label>` partition that are accessible to a queue. **The values of this configuration item of all direct subqueues under each parent queue must sum up to 100. The default value is 0.**

©2013-2024 Tencent Cloud. All rights reserved.

yarn.scheduler.capacity. `<queue-path>` .accessible-node-labels. `<label>` .maximum-capacity	Similar to yarn.scheduler.capacity. `<queue-path>` .maximum-capacity of Capacity Scheduler, this configuration item specifies the maximum percentage of nodes in the `<label>` partition that are accessible to a queue. The default value is 100.
yarn.scheduler.capacity. `<queue-path>` .default-node-label-expression	If no node label is specified in a resource request, applications are submitted to the partition specified by this configuration item. By default, the value is empty, which indicates that containers on nodes without a label are assigned to the applications.

Use Cases

Preparations

1. Prepare the cluster.

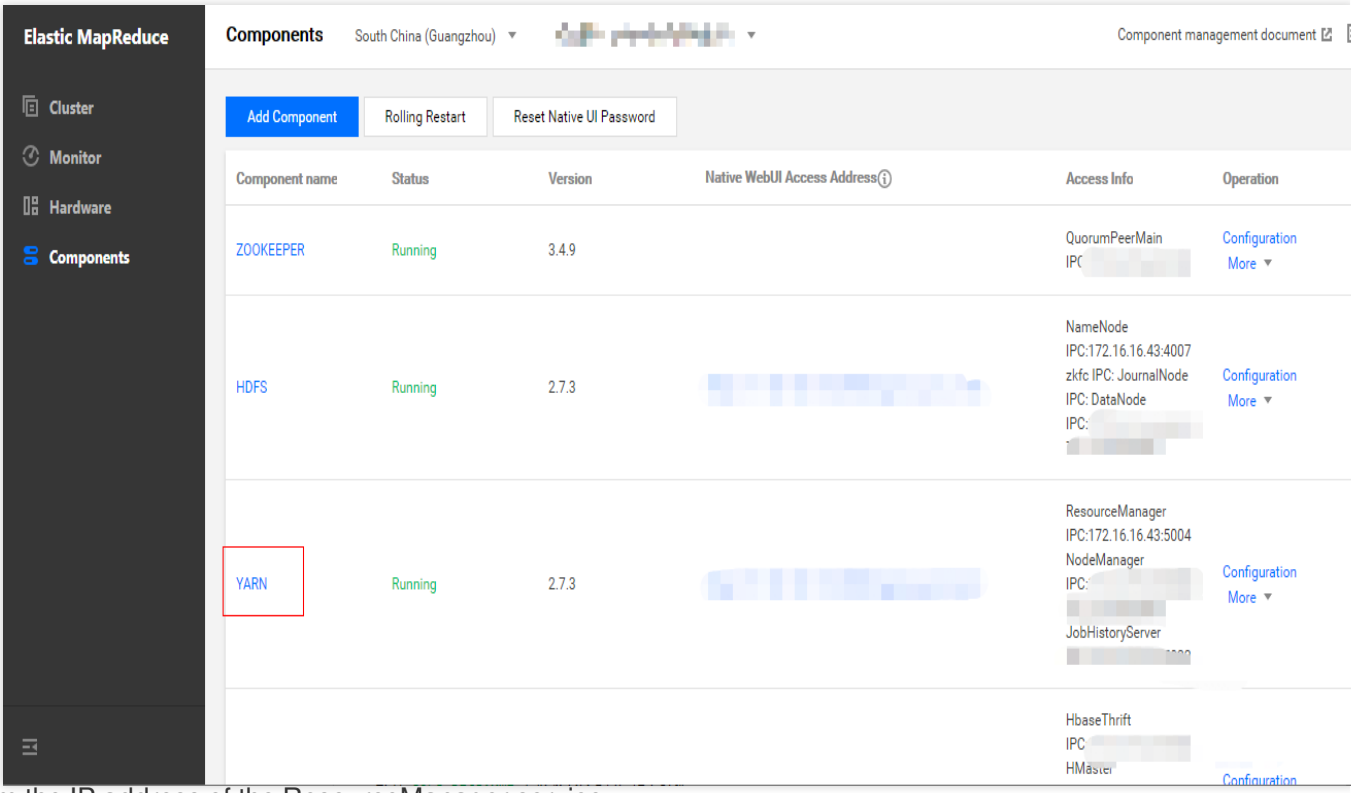
Make sure that you have activated Tencent Cloud and created an EMR cluster.

2. Check configurations of the YARN component.

On the **Cluster Service** page, click **YARN** to go to the component management page. On the **Configuration Management** tab, modify relevant parameters in `yarn-site.xml` , save the changes, and restart all YARN components. On the **Role Management** tab, confirm the IP address of the node where the ResourceManager service is located. Then, switch to the **Configuration Management** tab to modify relevant parameters in `yarn-site.xml` , save the changes, and restart all YARN components.

Find the cluster instance in the list of clusters, and click its ID to go to the cluster information page. Then, click **Cluster Service** on the left sidebar, and choose **Operations** > **Configuration Management** in the **YARN** component block.

©2013-2024 Tencent Cloud. All rights reserved.



Confirm the IP address of the ResourceManager service.

On the **Configuration Management** tab of the YARN component, select **Cluster-level** for **Dimension**, and select the IP address of the ResourceManager service as the node. Then, click **Modify Configuration** to modify the `yarn.resourcemanager.scheduler.class` parameter in `yarn-site.xml` on the selected node.

Configuring the mapping between node labels and queues and the resource percentage of queues in Capacity-Scheduler.xml

1. Create an HDFS directory to store node labels.

```
[root@172 ~]# cd /usr/local/service/hadoop/
[root@172 hadoop]# su hadoop
[hadoop@172 hadoop]$ hadoop fs -mkdir -p /yarn/node-labels
```

2. Obtain the IP address and port number of the NameNode (NN) in `core-site.xml` .

```
<property>
        <name>fs.defaultFS</name>
        <value>hdfs://172.16.32.47:4007</value>
</property>
```

3. Create new configuration items in `yarn-site.xml` for the master node and then restart ResourceManager.

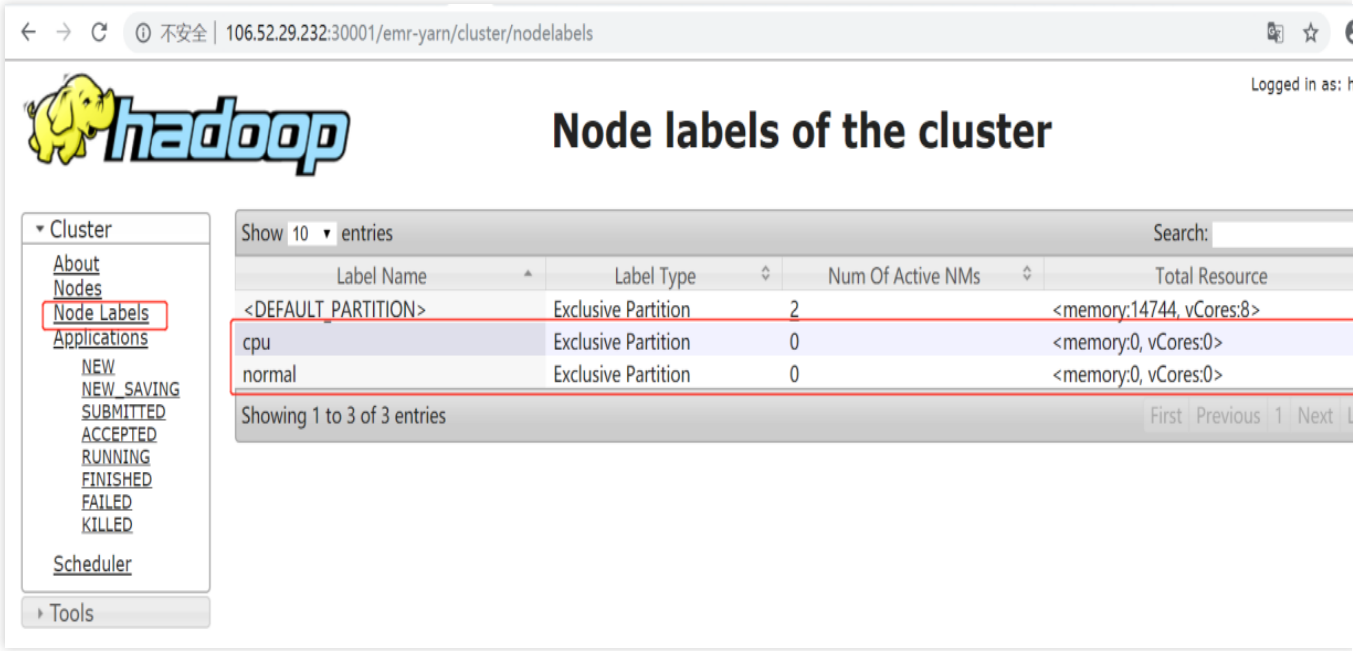
©2013-2024 Tencent Cloud. All rights reserved.

yarn.node-labels.fs-store.root-dir	hdfs://172.16.32.47:4007/sy
yarn.node-labels.enabled	true
yarn.node-labels.configuration-type	centralized

4. Run the `yarn rmadmin -addToClusterNodeLabels` command to add labels.

```
[hadoop@172 hadoop]$ yarn cluster --list-node-labels
Node Labels:
[hadoop@172 hadoop]$ yarn rmadmin -addToClusterNodeLabels "normal,cp
[hadoop@172 hadoop]$ yarn cluster --list-node-labels
Node Labels: <normal:exclusivity=true>,<cpu:exclusivity=true>
```

Open the YARN WebUI. You can see all the labels of the cluster in the **Node Labels** panel.

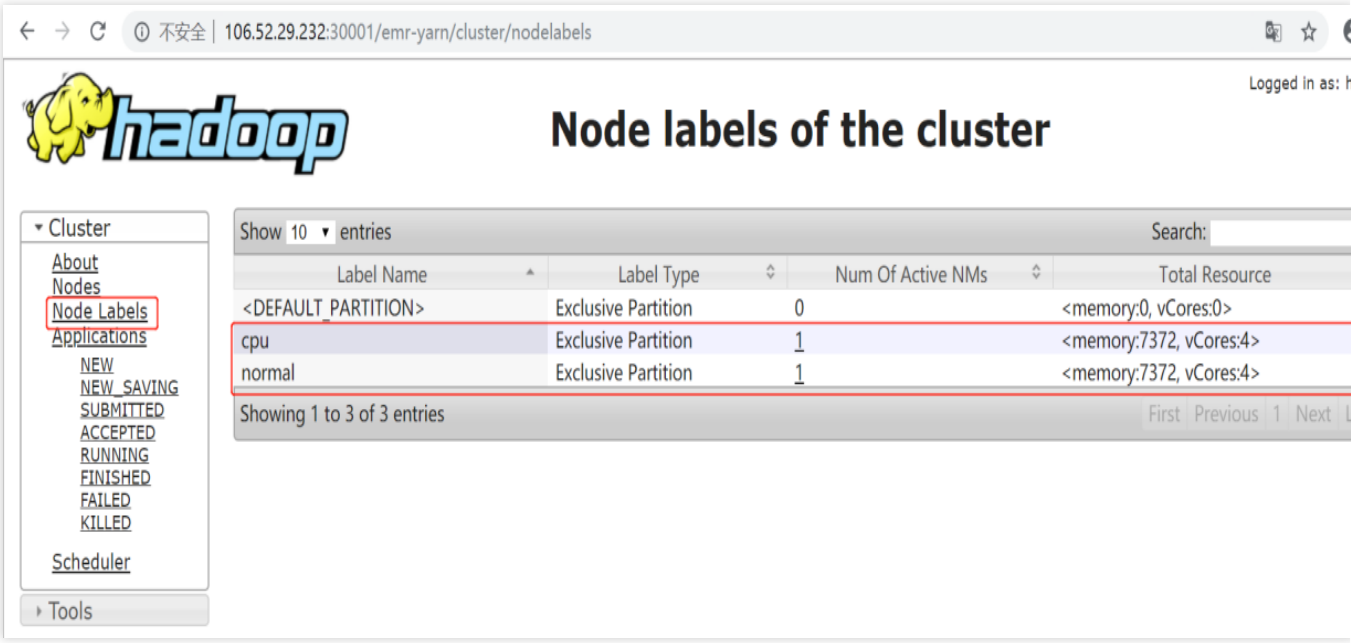


5. Run the `yarn rmadmin -replaceLabelsOnNode` command to label nodes.

©2013-2024 Tencent Cloud. All rights reserved.

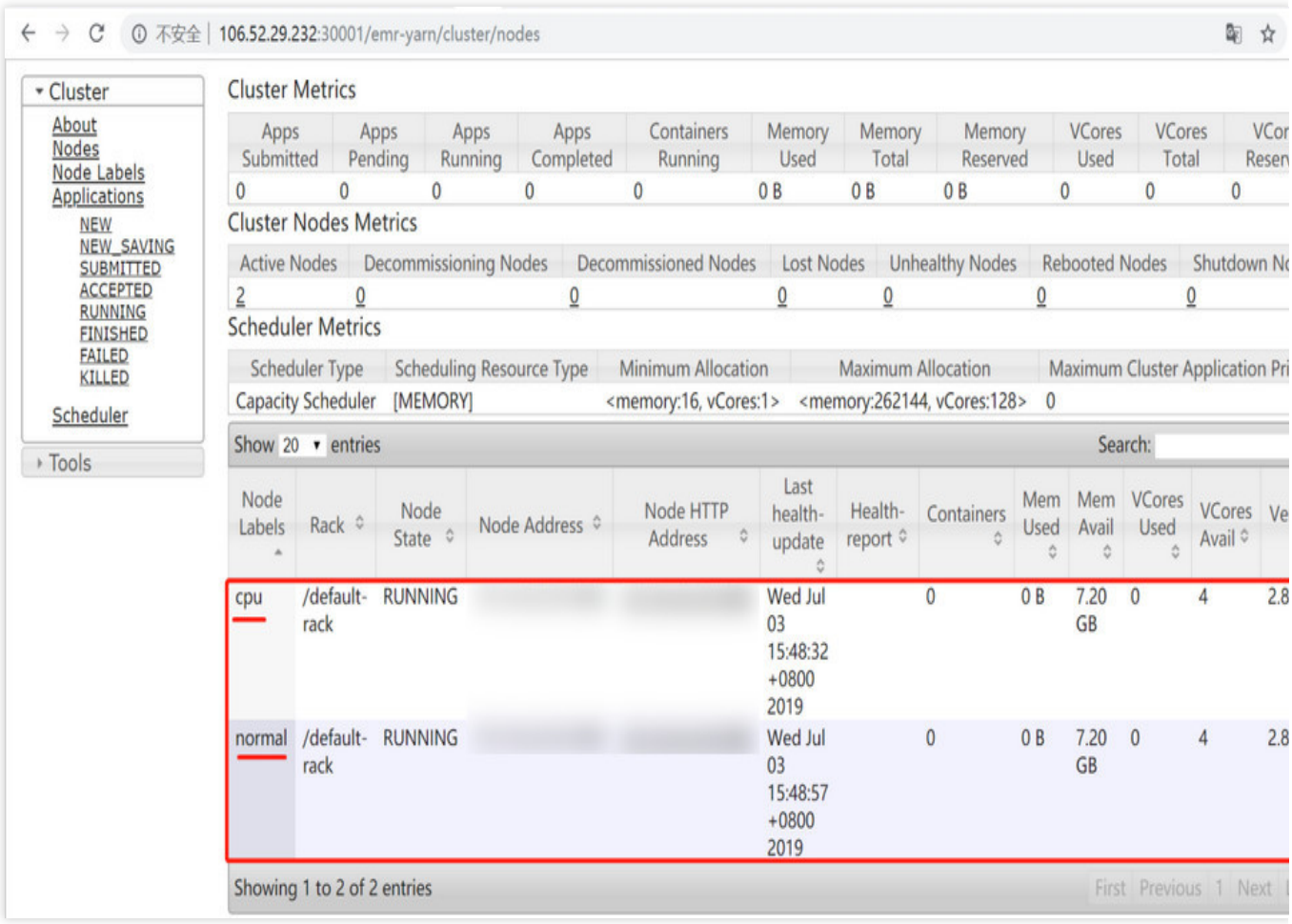
```
[hadoop@172 hadoop]$ yarn rmadmin -replaceLabelsOnNode "172.16.32.43=norma
[hadoop@172 hadoop]$ yarn rmadmin -replaceLabelsOnNode "172.16.32.25=cpu"
```

In the **Node Labels** panel, the number of nodes in the normal partition and cpu partition changes from 0 to 1.



In the **Scheduler** panel, the two nodes in the test system are respectively labeled with **normal** and **cpu**.

©2013-2024 Tencent Cloud. All rights reserved.



6. Edit the configuration items in `Capacity-Scheduler.xml` to configure the cluster queues, resource percentages of queues, and labels accessible to queues. Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration><property>
 <name>yarn.scheduler.capacity.maximum-am-resource-percent</name>
 <value>0.8</value>
</property>
<property>
 <name>yarn.scheduler.capacity.maximum-applications</name>
 <value>1000</value>
</property>
<property>
 <name>yarn.scheduler.capacity.root.queues</name>
 <value>default,dev,product</value>
</property>
<property>
 <name>yarn.scheduler.capacity.root.default.capacity</name>
 <value>20</value>
</property>
```

©2013-2024 Tencent Cloud. All rights reserved.

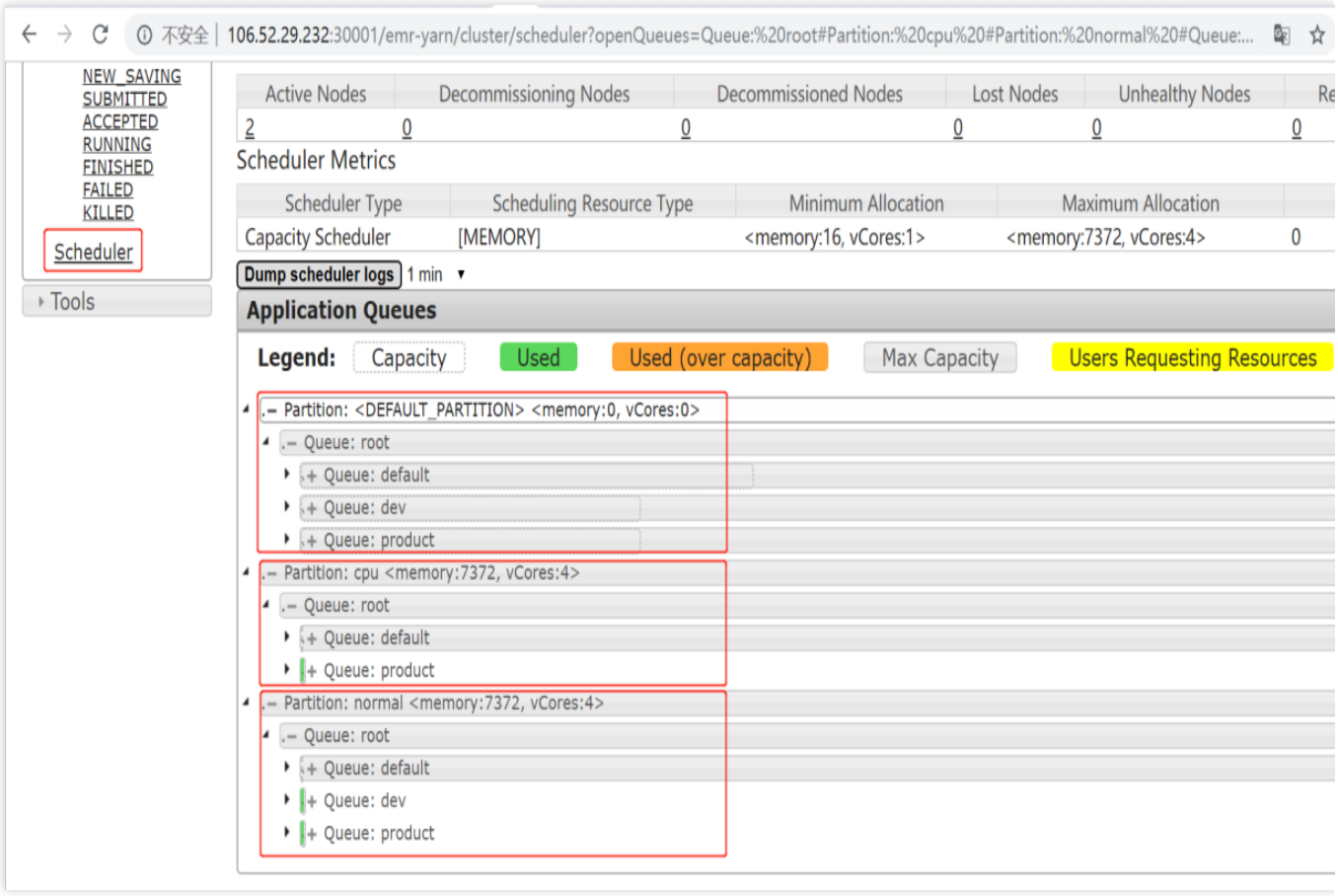
```
<property>
 <name>yarn.scheduler.capacity.root.dev.capacity</name>
 <value>40</value>
</property>
<property>
 <name>yarn.scheduler.capacity.root.product.capacity</name>
 <value>40</value>
</property>
<property>
 <name>yarn.scheduler.capacity.root.accessible-node-labels.cpu.capacity</name>
 <value>100</value>
</property>
<property>
 <name>yarn.scheduler.capacity.root.accessible-node-
labels.normal.capacity</name>
 <value>100</value>
</property>
<property>
 <name>yarn.scheduler.capacity.root.accessible-node-labels</name>
 <value>*</value>
</property>
<property>
 <name>yarn.scheduler.capacity.root.dev.accessible-node-
labels.normal.capacity</name>
 <value>100</value>
</property>
<property>
 <name>yarn.scheduler.capacity.root.product.accessible-node-
labels.cpu.capacity</name>
 <value>100</value>
</property>
<property>
 <name>yarn.scheduler.capacity.root.dev.accessible-node-labels</name>
 <value>normal</value>
</property>
<property>
 <name>yarn.scheduler.capacity.root.dev.default-node-label-expression</name>
 <value>normal</value>
</property>
<property>
 <name>yarn.scheduler.capacity.root.product.accessible-node-labels</name>
 <value>cpu</value>
</property>
<property>
 <name>yarn.scheduler.capacity.root.product.default-node-label-
expression</name>
 <value>cpu</value>
```

©2013-2024 Tencent Cloud. All rights reserved.

```
</property>
<property>
 <name>yarn.scheduler.capacity.normal.sharable-partitions</name>
 <value>cpu</value>
</property>
<property>
 <name>yarn.scheduler.capacity.normal.require-other-partition-resource</name>
 <value>true</value>
</property>
<property>
 <name>yarn.scheduler.capacity.cpu.sharable-partitions</name>
 <value></value>
</property>
<property>
 <name>yarn.scheduler.capacity.cpu.require-other-partition-resource</name>
 <value>true</value>
</property>
</configuration>
```

The **Scheduler** panel displays the information about the three partitions of the test cluster and the resources and queues in each partition. The **Application Queues** panel displays the **default**, **normal**, and **cpu** partitions. The **default** partition is the default one. The **normal** partition consists of nodes with the **normal** label. The **cpu** partition consists of nodes with the **cpu** label. The test cluster has two nodes, which are labeled **normal** and **cpu**. To view the queues in a partition, click the plus sign (+) on the left of the partition name.

©2013-2024 Tencent Cloud. All rights reserved.



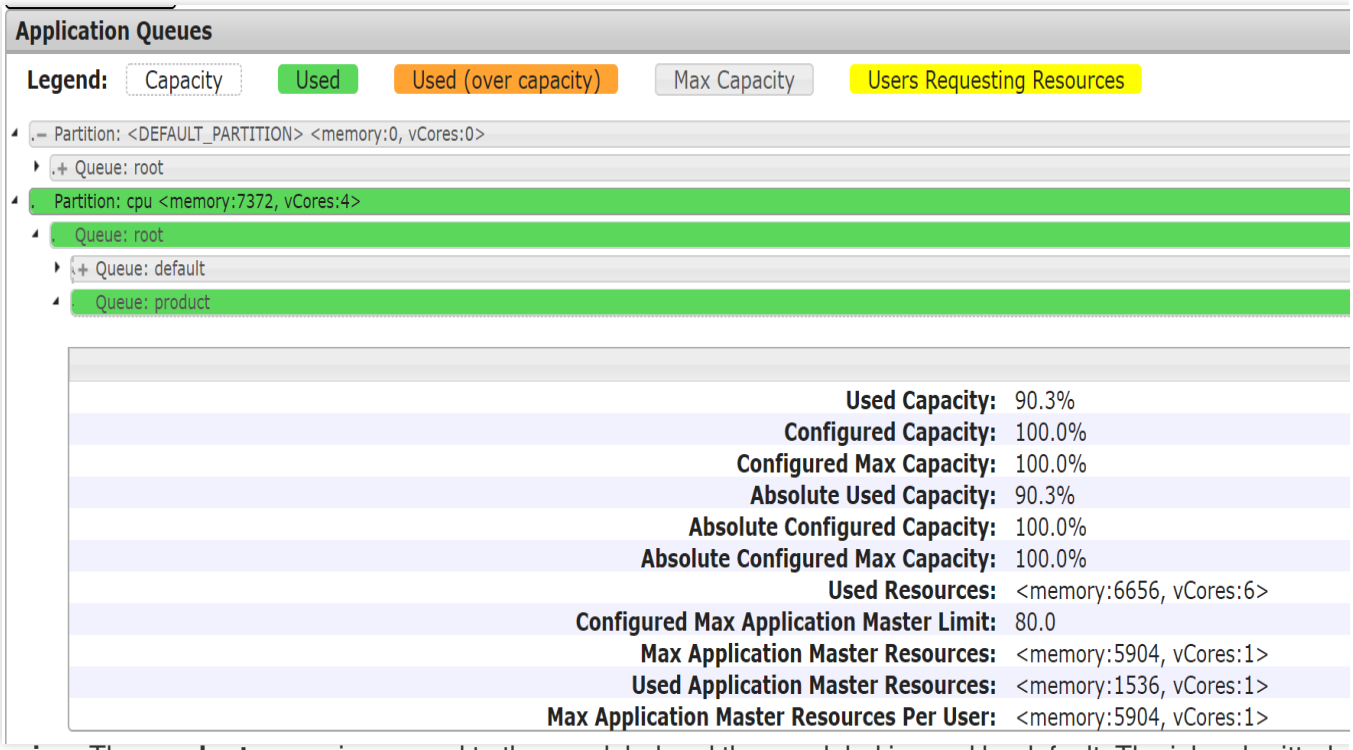
Verifying Label-Based Scheduling

Test 1: Submitting a job to the product queue

```
[hadoop@172 hadoop]$ cd /usr/local/service/hadoop
[hadoop@172 hadoop]$ yarn jar ./share/hadoop/mapreduce/hadoop-mapreduce-client-
jobclient-2.8.4-tests.jar sleep -Dmapreduce.job.queuename=product -m 32 -mt
1000
```

After the job is submitted, the usage of queue resources in each partition is shown in the following figure:

©2013-2024 Tencent Cloud. All rights reserved.



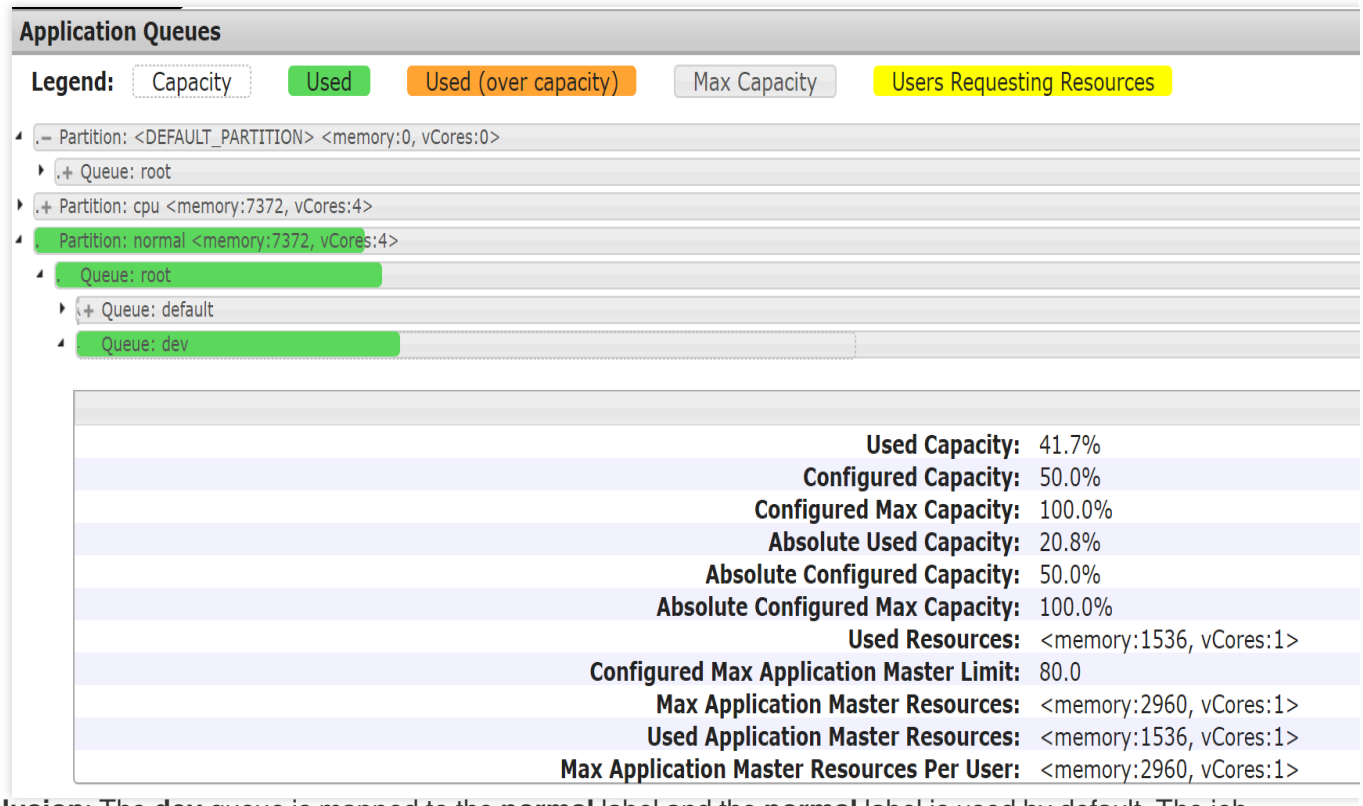
Conclusion: The **product** queue is mapped to the **cpu** label and the **cpu** label is used by default. The job submitted to the **product** queue runs on the node with the **cpu** label.

Test 2: Submitting a job to the dev queue

```
[hadoop@172 hadoop]$ cd /usr/local/service/hadoop
[hadoop@172 hadoop]$ yarn jar ./share/hadoop/mapreduce/hadoop-mapreduce-client-
jobclient-2.8.4-tests.jar sleep -Dmapreduce.job.queuename=dev -m 32 -mt 1000
```

After the job is submitted, the usage of queue resources in each partition is shown in the following figure:

©2013-2024 Tencent Cloud. All rights reserved.



Conclusion: The **dev** queue is mapped to the **normal** label and the **normal** label is used by default. The job submitted to the **dev** queue runs on the node with the **normal** label.

©2013-2024 Tencent Cloud. All rights reserved.

Hadoop Practical Tutorial

Last updated：2024-10-21 17:23:52

In Hadoop, distributed file system HDFS, resource scheduling framework YARN, and iterative computing framework MR. Tencent Cloud's Hadoop version that have integrated with COS, allowing you to access to COS by running the `hadoop fs` command line so as to separate compute and storage apart. Below are some best practices:
HDFSFor both high-availability (HA) cluster and non-HA cluster, **do not format the namenode**; otherwise, your data will be lost permanently. Tencent Cloud shall not be responsible under any circumstance for any loss of data caused by formatting the namenode.
YARNThe fair scheduler is enabled by default, and you can change the scheduler based on your actual needs.

©2013-2024 Tencent Cloud. All rights reserved.

Using API to Analyze Data in HDFS and COS

Last updated：2021-07-08 10:43:44

The WordCount application is a great example that gives you a hands-on experience in developing your first Hadoop MapReduce application. In this tutorial, you will learn how to implement WordCount example code in MapReduce to count the number of occurrences of a given word in the input file stored in HDFS or in COS. The program is the same as the one shown in the Hadoop community.

1. Development Preparations

This task requires access to COS, so you need to create a bucket in COS first.
Create an EMR cluster. When creating the EMR cluster, you need to select a cluster type that includes HDFS and enable access to COS on the basic configuration page.

2. Logging in to an EMR Server

Log in to any node (preferably a master node) in the EMR cluster before performing relevant operations. EMR is built on CVM instances running on Linux; therefore, using EMR in command line mode requires logging in to an CVM instance.
After creating the EMR cluster, select **Elastic MapReduce** in the console, click the ID/name of the cluster you just created in the cluster list, click **Cluster Resource** > **Resource Management** > **Master**, and click the resource ID of an active master node to enter the CVM console and find the CVM instance of the EMR cluster.
For information about how to log in to a CVM instance, see Logging in to Linux Instance Using Standard Login Method. Here, you can use WebShell to log in. Click **Login** on the right of the desired CVM instance to go to the login page. The default username is `root`, and the password is the one you set when creating the EMR cluster.
Once your credentials are validated, you can enter the EMR command line interface. All Hadoop operations should be performed under the `Hadoop` user. The `root` user is logged in by default when you log in to the EMR node, so you need to switch to the `Hadoop` user. Run the following command to switch users and go to the `Hadoop` folder:

```
[root@172 ~]# su hadoop
[hadoop@172 root]$ cd /usr/local/service/hadoop
[hadoop@172 hadoop]$
```

3. Data Preparations

©2013-2024 Tencent Cloud. All rights reserved.

Prepare an input text file. You can either **store data in an HDFS cluster** or **store data in COS**.

First, create a .txt file named `test.txt` locally and add the following sentences to the file:

```
Hello World.
this is a message.
this is another message.
Hello world, how are you?
```

Use scp or sftp service to upload a local file to the CVM instance in your EMR cluster. Run the following command in your local shell:

```
scp $localfile root@public IP address:$remotefolder
```

Here, $localfile is the path and the name of your local file; root is the CVM instance username. You can look up the public IP address in the node information in the EMR or CVM Console; and $remotefolder is the path where you want to store the file in the CVM instance.

After the upload is completed, you can check whether the file is in the corresponding folder on the command line of the EMR cluster. The file is uploaded to the `/usr/local/service/hadoop` path in the EMR cluster in this example.

```
[hadoop@172 hadoop]$ ls -l
```

Storing Data in HDFS

After uploading the data to the CVM instance, you can copy the data file to the Hadoop cluster by running the following command:

```
[hadoop@172 hadoop]$ hadoop fs -put /usr/local/service/hadoop/test.txt
/user/hadoop/
```

After the copy is completed, run the following command to view the copied file:

```
[hadoop@172 hadoop]$ hadoop fs -ls /user/hadoop
Output:
-rw-r--r-- 3 hadoop supergroup 85 2018-07-06 11:18 /user/hadoop/test.txt
```

If there is no `/user/hadoop` folder in Hadoop, you can create it on your own by running the following command:

```
[hadoop@172 hadoop]$ hadoop fs -mkdir /user/hadoop
```

For more Hadoop commands, see HDFS Common Operations.

Storing Data in COS

There are two ways to store data in COS: **uploading via the COS console from the local storage** and **uploading via a Hadoop command**.

©2013-2024 Tencent Cloud. All rights reserved.

When uploading via the COS console from the local storage, you can view the uploaded data file by running the following command:

```
[hadoop@10 hadoop]$ hadoop fs -ls cosn://$bucketname/ test.txt
```

```
rw-rw-rw- 1 hadoop hadoop 1366 2017-03-15 19:09 cosn://$bucketname/test.txt
```

Replace `$bucketname` with the name and path of your bucket.

To upload via Hadoop command, run the following command:

```
[hadoop@10 hadoop]$ hadoop fs -put test.txt cosn://$bucketname /
[hadoop@10 hadoop]$ hadoop fs -ls cosn:// $bucketname / test.txt
```

```
rw-rw-rw- 1 hadoop hadoop 1366 2017-03-15 19:09 cosn://$bucketname / test.txt
```

4. Creating a Project with Maven

Maven is recommended for project management. It can help you manage project dependencies with ease. Specifically, it can get .jar packages through the configuration of the `pom.xml` file, eliminating the need to add them manually.

Download and install Maven first and then configure its environment variables. If you are using the IDE, please set the Maven-related configuration items in the IDE.

Creating a Maven Project

Enter the directory of the Maven project, such as `D://mavenWorkplace`, and create the project using the following commands:

```
mvn     archetype:generate     -DgroupId=$yourgroupID     -
DartifactId=$yourartifactID
-DarchetypeArtifactId=maven-archetype-quickstart
```

Here, `$yourgroupID` is your package name, `$yourartifactID` is your project name, and `maven-archetype-quickstart` indicates to create a Maven Java project. Some files need to be downloaded during the project creation, so please keep the network connected.

After successfully creating the project, you will see a folder named `$yourartifactID` in the `D://mavenWorkplace` directory. Files in the folder have the following structure:

```
simple
     ---pom.xml       Core configuration, under the project root directory
     ---src
        ---main
           ---java         Java source code directory
           ---resources    Java configuration file directory
```

©2013-2024 Tencent Cloud. All rights reserved.

```
        ---test
           ---java         Test source code directory
           ---resources    Test configuration directory
```

We need to pay attention to the pom.xml file and the Java folder under the main directory, as the former is primarily used for dependency and packaging configuration, and the latter for source code storage.

First, add the Maven dependencies to pom.xml:

```
<dependencies>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>2.7.3</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-mapreduce-client-core</artifactId>
            <version>2.7.3</version>
        </dependency>
</dependencies>
```

Then, add the packaging and compiling plugins to pom.xml:

```
<build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                    <encoding>utf-8</encoding>
                </configuration>
            </plugin>
            <plugin>
                <artifactId>maven-assembly-plugin</artifactId>
                <configuration>
                    <descriptorRefs>
                        <descriptorRef>jar-with-dependencies</descriptorRef>
                    </descriptorRefs>
                </configuration>
                <executions>
                    <execution>
                        <id>make-assembly</id>
                        <phase>package</phase>
                        <goals>
                            <goal>single</goal>
```

©2013-2024 Tencent Cloud. All rights reserved.

```
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
</build>
```

Right-click in src>main>java and create a Java Class. Enter the Class name (e.g., WordCount here) and add the sample code to the Class:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

import java.io.IOException;
import java.util.StringTokenizer;

/**
 * Created by tencent on 2018/7/6.
 */
public class WordCount {
    public static class TokenizerMapper
            extends Mapper<Object, Text, Text, IntWritable>
    {
        private static final IntWritable one = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Mapper<Object, Text, Text,
IntWritable>.Context context)
                throws IOException, InterruptedException
        {
            StringTokenizer itr = new StringTokenizer(value.toString());
            while (itr.hasMoreTokens())
            {
                this.word.set(itr.nextToken());
                context.write(this.word, one);
            }
        }
    }
```

©2013-2024 Tencent Cloud. All rights reserved.

```
    public static class IntSumReducer
            extends Reducer<Text, IntWritable, Text, IntWritable>
    {
        private IntWritable result = new IntWritable();

        public void reduce(Text key, Iterable<IntWritable> values,
Reducer<Text, IntWritable, Text, IntWritable>.Context context)
                throws IOException, InterruptedException
        {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            this.result.set(sum);
            context.write(key, this.result);
        }
    }

    public static void main(String[] args)
            throws Exception
    {
        Configuration conf = new Configuration();
        String[] otherArgs = new GenericOptionsParser(conf,
args).getRemainingArgs();
        if (otherArgs.length < 2)
        {
            System.err.println("Usage: wordcount <in> [<in>...] <out>");
            System.exit(2);
        }
        Job job = Job.getInstance(conf, "word count");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(TokenizerMapper.class);
        job.setCombinerClass(IntSumReducer.class);
        job.setReducerClass(IntSumReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        for (int i = 0; i < otherArgs.length - 1; i++) {
            FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
        }
        FileOutputFormat.setOutputPath(job, new
Path(otherArgs[(otherArgs.length - 1)]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

As you can see, there is a Map function and a Reduce function.

©2013-2024 Tencent Cloud. All rights reserved.

If your Maven is configured correctly and its dependencies are successfully imported, the project can be compiled directly. Enter the project directory in the local shell, and run the following command to package the entire project:

```
mvn package
```

Some files may need to be downloaded during the running process. "Build success" indicates that package is successfully created. You can see the generated .jar package in the target folder under the project directory.
Upload the packaged project file to the CVM instance of the EMR cluster using the scp or sftp service. Run the following command in the local shell:

```
scp $jarpackage root@public IP address: /usr/local/service/hadoop
```

Here, `$jarpackage` is the path plus name of your local .jar package; `root` is the CVM instance username; and the public IP address can be viewed in the node information in the EMR console or the CVM console. The file is uploaded to the `/usr/local/service/hadoop` folder of the EMR cluster.

Counting a Text File in HDFS

Go to the `/usr/local/service/hadoop` directory as described in data preparations, and submit the task by running the following command:

```
[hadoop@10 hadoop]$ bin/hadoop jar
/usr/local/service/hadoop/WordCount-1.0-SNAPSHOT-jar-with-dependencies.jar
WordCount /user/hadoop/test.txt /user/hadoop/WordCount_output
```

Note:

Above is a complete command, where `/user/hadoop/ test.txt` is the input file and `/user/hadoop/ WordCount_output` is the output folder. You should not create the `WordCount_output` folder before the command is submitted; otherwise, the submission will fail.
After the execution is completed, view the output file by running the following command:

```
[hadoop@172 hadoop]$ hadoop fs -ls /user/hadoop/WordCount_output
Found 2 items
-rw-r--r-- 3 hadoop supergroup 0 2018-07-06 11:35
/user/hadoop/MEWordCount_output/_SUCCESS
-rw-r--r-- 3 hadoop supergroup 82 2018-07-06 11:35
/user/hadoop/MEWordCount_output/part-r-00000
```

View the statistics in part-r-00000 by running the following command:

```
[hadoop@172 hadoop]$ hadoop fs -cat /user/hadoop/MEWordCount_output/part-r-
00000
Hello   2
World.  1
a       1
```

©2013-2024 Tencent Cloud. All rights reserved.

```
another 1
are     1
how     1
is      2
message.        2
this    2
world,  1
you?    1……
```

Counting a Text File in COS

Go to the `/usr/local/service/hadoop` directory and submit the task by running the following command:

```
[hadoop@10 hadoop]$ hadoop jar
/usr/local/service/hadoop/WordCount-1.0-SNAPSHOT-jar-with-dependencies.jar
WordCount cosn://$bucketname/test.txt cosn://$bucketname /WordCount_output
```

The input file for the command is changed to `cosn:// $bucketname/ test.txt` , where $bucketname is your bucket name and path. The result will go to COS as well. Run the following command to view the output file:

```
[hadoop@10 hadoop]$ hadoop fs -ls cosn:// $bucketname /WordCount_output
Found 2 items
-rw-rw-rw- 1 hadoop Hadoop 0 2018-07-06 10:34 cosn://$bucketname
/WordCount_output/_SUCCESS
-rw-rw-rw- 1 hadoop Hadoop 1306 2018-07-06 10:34 cosn://$bucketname
/WordCount_output/part-r-00000
```

View the final output result:

```
[hadoop@10 hadoop]$ hadoop fs -cat cosn:// $bucketname /WordCount_output1/part-
r-00000
Hello   2
World.  1
a       1
another 1
are     1
how     1
is      2
message.        2
this    2
world,  1
you?    1
```

©2013-2024 Tencent Cloud. All rights reserved.

Dumping YARN Job Logs to COS

Last updated：2021-07-01 15:34:12

By default, Hadoop stores YARN job logs in HDFS. Tencent Cloud EMR also provides the ability to store YARN job logs in external storage (COS).

Prerequisites

The EMR cluster needs to support COS. For more information, please see Analyzing Data in HDFS/COS with API.

Directions

1. Modify the configuration in `yarn-site.xml` and deliver the configuration to all nodes.

```
yarn.nodemanager.remote-app-log-dir=cosn://[bucket_name]/[logs_dirs]
```

2. Add a new configuration item to `core-site.xml` and deliver it to all nodes.

```
fs.AbstractFileSystem.cosn.impl=org.apache.hadoop.fs.cosnative.COS
```

3. Restart all the `nodemanager/datanode` services in the cluster.
4. Run the `hive/spark` job to view the job logs stored in COS.

```
hdfs dfs -ls cosn://[bucket_name]/[logs_dirs]
```

©2013-2024 Tencent Cloud. All rights reserved.

Spark Development Guide
Spark Environment Info

Last updated：2025-01-03 14:50:16

EMR supports Spark 3.x and 2.x. The software environment information is as follows:

By default, Spark is installed on a master node.

After logging in, run the `su hadoop` command to switch to the `hadoop` user.

The Spark software path is `/usr/local/service/spark` .

The relevant logs are stored in `/data/emr` .

The above is mainly about how to access COS by using Spark. For more information, see the community documentation.

©2013-2024 Tencent Cloud. All rights reserved.

Using Spark to Analyze Data in COS

Last updated：2025-01-03 14:50:17

Apache Spark is an open-source project for fast, general-purpose, large-scale data processing. It is similar to Hadoop's MapReduce but faster and more efficient for batch processing. It utilizes in-memory caching and optimized execution for fast performance, and it supports reading/writing Hadoop data in any format. Now Spark has become a unified big data processing platform with a lightning-fast analysis engine for real-time streaming processing, machine learning, and interactive queries.
Spark is an in-memory parallel computing framework for big data processing. Its in-memory computing feature improves the real-time performance of data processing in a big data environment, while ensuring high fault tolerance and scalability. Spark can be deployed on a large number of inexpensive hardware devices to create clusters.
The task submitted in this tutorial is a wordcount task, i.e., counting the number of words. You need to upload the file for counting to the cluster in advance.

1. Development Preparations

This task requires access to COS, so you need to create a bucket in COS first.
Create an EMR cluster. When creating the EMR cluster, you need to select the Spark component on the software configuration page and enable access to COS on the basic configuration page.

2. Creating a Project with Maven

Here, the demo that comes with the system is not used; instead, you need to create a project and compile, compress, and upload it to the EMR cluster on your own for execution. Maven is recommended for project management, as it can help you manage project dependencies with ease. Specifically, it can get .jar packages through the configuration of the `pom.xml` file, eliminating the need to add them manually.
Download and install Maven, then configure its environment variables. If you are using the IDE, please set the Maven-related configuration in the IDE.

Creating a Maven Project

In the local shell environment, enter the directory where you want to create the Maven project, such as `D://mavenWorkplace`, and enter the following command to create it:

```
mvn archetype:generate -DgroupId=$yourgroupID -DartifactId=$yourartifactID -
DarchetypeArtifactId=maven-archetype-quickstart
```

©2013-2024 Tencent Cloud. All rights reserved.

Here, $yourgroupID is your package name, $yourartifactID is your project name, and maven-archetype-quickstart indicates to create a Maven Java project. Some files need to be downloaded during the project creation, so please keep the network connected.

After successfully creating the project, you will see a folder named `$yourartifactID` in the `D://mavenWorkplace` directory. Files in the folder have the following structure:

```
simple
    ---pom.xml       Core configuration, under the project root directory
    ---src
        ---main
            ---java         Java source code directory
            ---resources    Java configuration file directory
        ---test
             ---java         Test source code directory
             ---resources    Test configuration directory
```

Among the files above, pay extra attention to the pom.xml file and the Java folder under the main directory. The pom.xml file is primarily used to create dependencies and package configurations; the Java folder is used to store your source codes.

First, add the Maven dependencies to pom.xml:

```
<dependencies>
    <dependency>
        <groupId>org.apache.spark</groupId>
        <artifactId>spark-core_2.11</artifactId>
        <version>2.0.2</version>
    </dependency>
</dependencies>
```

Then, add the packaging and compiling plugins to pom.xml:

```
<build>
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
      <source>1.8</source>
      <target>1.8</target>
      <encoding>utf-8</encoding>
    </configuration>
  </plugin>
  <plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <configuration>
```

©2013-2024 Tencent Cloud. All rights reserved.

```
      <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
      </descriptorRefs>
    </configuration>
    <executions>
      <execution>
        <id>make-assembly</id>
        <phase>package</phase>
        <goals>
          <goal>single</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</plugins>
</build>
```

Right-click in src>main>Java and create a Java Class. Enter the Class name (e.g., WordCountOnCos here) and add the sample code to the Class:

```
import java.util.Arrays;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaRDD;
import org.apache.spark.api.java.JavaSparkContext;
import scala.Tuple2;

/**
 * Created by tencent on 2018/6/28.
 */
public class WordCountOnCos {
    public static void main(String[] args){
        SparkConf sc = new SparkConf().setAppName("spark on cos");
        JavaSparkContext context = new JavaSparkContext(sc);
        JavaRDD<String> lines = context.textFile(args[0]);

        lines.flatMap(x -> Arrays.asList(x.split(" ")).iterator())
                .mapToPair(x -> new Tuple2<String, Integer>(x, 1))
                .reduceByKey((x, y) -> x+y)
                .saveAsTextFile(args[1]);
    }
}
```

If your Maven is configured correctly and its dependencies are successfully imported, the project will be compiled directly. Enter the project directory in the local shell, and run the following command to package the entire project:

```
mvn package
```

©2013-2024 Tencent Cloud. All rights reserved.

Some files may need to be downloaded during the running process. "Build success" indicates that package is successfully created. You can see the generated .jar package in the target folder under the project directory.

Data Preparations

First, you need to upload the compressed .jar package to the EMR cluster using the scp or sftp tool by running the following command in local command line mode:

```
scp $localfile root@public IP address:$remotefolder
```

Here, `$localfile` is the path plus name of your local file; `root` is the CVM instance username. You can look up the public IP address in the node information in the EMR console or the CVM console. `$remotefolder` is the path where you want to store the file in the CVM instance. After the upload is completed, you can check whether the file is in the corresponding folder on the EMR command line.

You need to upload the to-be-processed file to COS in advance. If the file is in your local storage, you can upload it directly via the COS console; if it is in the EMR cluster, you can upload it by running the following Hadoop command:

```
[hadoop@10 hadoop]$ hadoop fs -put $testfile cosn://$bucketname/
```

Here, `$testfile` is the full path plus name of the file for counting, and `$bucketname` is your bucket name. After the upload is completed, you can check whether the file is present in COS in the COS console.

Running the Demo

First, log in to any node (preferably a master one) in the EMR cluster. For information about how to log in to EMR, see Logging in to Linux Instance Using Standard Login Method. Here, you can use WebShell to log in. Click **Login** on the right of the desired CVM instance to go to the login page. The default username is `root` , and the password is the one you set when creating the EMR cluster. Once your credentials are validated, you can enter the command line interface.

Run the following command in EMR command-line interface to switch to the Hadoop user:

```
[root@172 ~]# su hadoop
```

Then, go to the folder where the .jar package is stored and run the following command:

```
[hadoop@10spark]$ spark-submit --class $WordCountOnCOS --master
yarn-cluster $packagename.jar cosn:// $bucketname /$testfile cosn://
$bucketname
/output
```

Here, `$WordCountOnCOS` is your Java Class name, `$packagename` is the name of the .jar package generated in the new Maven project you created, `$bucketname` is your bucket name plus path, and `$testfile` is the name of the file for counting. The output file is stored in the output folder, **which cannot be created beforehand; otherwise, the execution will fail**.

©2013-2024 Tencent Cloud. All rights reserved.

After successful execution, you can see the result of the wordcount task in the specified bucket and folder.

```
[hadoop@172 /]$ hadoop fs -ls cosn:// $bucketname /output
Found 3 items
-rw-rw-rw- 1 hadoop Hadoop  0 2018-06-28 19:20 cosn:// $bucketname
/output/_SUCCESS
-rw-rw-rw- 1 hadoop Hadoop 681 2018-06-28 19:20 cosn:// $bucketname
/output/part-00000
-rw-rw-rw- 1 hadoop Hadoop 893 2018-06-28 19:20 cosn:// $bucketname
/output/part-00001

[hadoop@172 demo]$ hadoop fs -cat cosn://$bucketname/output/part-00000
18/07/05 17:35:01 INFO cosnative.NativeCosFileSystem: Opening 'cosn://
$bucketname/output/part-00000' for reading
(under,1)
(this,3)
(distribution,2)
(Technology,1)
(country,1)
(is,1)
(Jetty,1)
(currently,1)
(permitted.,1)
(Security,1)
(have,1)
(check,1)
```

©2013-2024 Tencent Cloud. All rights reserved.

Using Spark Python to Analyze Data in COS

Last updated：2025-01-03 14:50:17

This section describes running a Spark wordcount application in Python.

Development Preparations

This task requires access to COS, so you need to create a bucket in COS first.
Create an EMR cluster. When creating the EMR cluster, you need to select the Spark component on the software configuration page and enable access to COS on the basic configuration page.

Data Preparations

Upload the to-be-processed file to COS first. If the file is in your local storage, upload it directly via the COS console; if it is in the EMR cluster, upload it by running the following Hadoop command:

```
[hadoop@10 hadoop]$ hadoop fs -put $testfile cosn:// $bucketname/
```

Here, $testfile is the full path with file name and $bucketname is your bucket name. After the upload is completed, you can check whether the file is available in COS.

Running the Demo

First, log in to any node (preferably a master one) in the EMR cluster. For information about how to log in to EMR, see Logging in to Linux Instance Using Standard Login Method. Here, you can use WebShell to log in. Click **Login** on the right of the desired CVM instance to go to the login page. The default username is `root`, and the password is the one you set when creating the EMR cluster. Once your credentials are validated, you can enter the command line interface.
Run the following command on the EMR command-line interface to switch to the Hadoop user and go to the Spark installation directory `/usr/local/service/spark`:

```
[root@172 ~]# su hadoop
[hadoop@172 root]$ cd /usr/local/service/spark
```

Create a Python file named wordcount.py and add the following code:

```
from __future__ import print_function
```

©2013-2024 Tencent Cloud. All rights reserved.

```
import sys
from operator import add
from pyspark.sql import SparkSession

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: wordcount <file>", file=sys.stderr)
        exit(-1)

    spark = SparkSession\\
        .builder\\
        .appName("PythonWordCount")\\
        .getOrCreate()

    sc = spark.sparkContext

    lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0])
    counts = lines.flatMap(lambda x: x.split(' ')) \\
                  .map(lambda x: (x, 1)) \\
                  .reduceByKey(add)

    output = counts.collect()
    counts.saveAsTextFile(sys.argv[2])

    spark.stop()
```

Submit the task by running the following command:

```
[hadoop@10   spark]$   ./bin/spark-submit   --master yarn   ./wordcount.py
cosn://$bucketname/$yourtestfile cosn:// $bucketname/$output
```

Here, $bucketname is your COS bucket name, $yourtestfile is the full path with test file name in the bucket, and $output is your output folder. **If the $output folder already exists before the command is executed, the program will fail.**

After the program is running automatically, you can find the output file in the destination bucket:

```
[hadoop@172 spark]$ hadoop fs -ls cosn:// $bucketname/$output
Found 2 items
-rw-rw-rw- 1 hadoop Hadoop 0 2018-06-29 15:35 cosn:// $bucketname/$output
/_SUCCESS
-rw-rw-rw- 1 hadoop Hadoop 2102 2018-06-29 15:34 cosn:// $bucketname/$output
/part-00000
```

You can also look up the the result by running the following command:

```
[hadoop@172 spark]$ hadoop fs -cat cosn:// $bucketname/$output /part-00000
```

©2013-2024 Tencent Cloud. All rights reserved.

```
(u'', 27)
(u'code', 1)
(u'both', 1)
(u'Hadoop', 1)
(u'Bureau', 1)
(u'Department', 1)
```

You can also output the result to HDFS by changing the output location in the command as follows:

```
[hadoop@10spark]$   ./bin/spark-submit   ./wordcount.py
cosn://$bucketname/$yourtestfile /user/hadoop/$output
```

Here, `/user/hadoop/` is the path in HDFS. If this path does not exist, you can create one.

After the task is completed, you can view the Spark execution log by running the following command:

```
[hadoop@10 spark]$  /usr/local/service/hadoop/bin/yarn logs -applicationId
$yourId
```

Here, $yourId should be replaced with your task ID, which can be viewed in Yarn's WebUI.

©2013-2024 Tencent Cloud. All rights reserved.

SparkSQL Tutorial

Last updated：2025-01-03 14:50:17

Spark SQL is Apache Spark's module for structured data processing. It provides a DataFrame abstraction in a variety of languages to simplify working with structured datasets and lets you query the data with distributed SQL.

1. Preparations for Development

Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, select the Spark component on the software configuration page.

2. Using the Interactive SparkSQL Console

First, log in to a master node of the EMR cluster before using SparkSQL. For more information on how to log in to EMR, please see Logging in to a Linux Instance. Here, you can use WebShell to log in. Click *Login* button on the right of the desired CVM instance and then enter the login page. The default username is root, and the password is the one you set when creating the EMR cluster. Once your credentials have been validated, you can access to the EMR command-line interface.
Run the following command in EMR command-line interface to switch to the Hadoop user and go to the directory `/usr/local/service/spark`:

```
[root@172 ~]# su hadoop
[hadoop@172 root]$ cd /usr/local/service/spark
```

You can access the interactive SparkSQL Console by running the following command:

```
[hadoop@10spark]$ bin/spark-sql --master yarn --num-executors 64 --executor-
memory 2g
```

Here, --master indicates your master URL, --num-executors the number of executors, and --executor-memory the storage capacity of the executors. You can modify these parameters based on your actual conditions and start/stop a SparkSQLthriftserver through `sbin/start-thriftserver.sh` / `sbin/stop-thriftserver.sh`.
Below are some basic operations in SparkSQL：
Create and view a database:

```
spark-sql> create database sparksql;
Time taken: 0.907 seconds

spark-sql> show databases;
```

©2013-2024 Tencent Cloud. All rights reserved.

```
default
sparksql
test
Time taken: 0.131 seconds, Fetched 5 row(s)
```

Create a new table in the database you just created and view the table:

```
spark-sql> use sparksql;
Time taken: 0.076 seconds

spark-sql> create table sparksql_test(a int,b string);
Time taken: 0.374 seconds

spark-sql> show tables;
sparksql_test   false
Time taken: 0.12 seconds, Fetched 1 row(s)
```

Insert two rows of data into the table and view them:

```
spark-sql> insert into sparksql_test values (42,'hello'),(48,'world');
Time taken: 2.641 seconds

spark-sql> select * from sparksql_test;
42      hello
48      world
Time taken: 0.503 seconds, Fetched 2 row(s)
```

For more information on Spark command line parameters, please see the community documentation.

3. Creating a Project with Maven

Download and install Maven first and then configure its environment variables. If you are using the IDE, please set the Maven-related configuration items in the IDE.

Creating a Maven project

Enter the directory of the Maven project, such as `D://mavenWorkplace`, and create the project by running the following commands:

```
mvn    archetype:generate    -DgroupId=$yourgroupID    -
DartifactId=$yourartifactID
-DarchetypeArtifactId=maven-archetype-quickstart
```

Here, $yourgroupID is your package name; $yourartifactID is your project name; maven-archetype-quickstart indicates to create a Maven Java project. Some files need to be downloaded during the project is being created, so

©2013-2024 Tencent Cloud. All rights reserved.

please stay connected to the Internet.

After successfully creating the project, you will see a folder named $yourartifactID in the `D://mavenWorkplace` directory. The files included in the folder have the following structure:

```
simple
     ---pom.xml       Core configuration, under the project root directory
     ---src
        ---main
            ---java         Java source code directory
            ---resources    Java configuration file directory
       ---test
           ---java         Test source code directory
           ---resources    Test configuration directory
```

Among the files above, pay extra attention to the pom.xml file and the Java folder under the main directory. The pom.xml file is primarily used to create dependencies and package configurations; the Java folder is used to store your source code.

Adding Hadoop dependencies and sample code

First, add the Maven dependencies to the pom.xml file:

```
<dependencies>
         <dependency>
             <groupId>org.apache.spark</groupId>
             <artifactId>spark-core_2.11</artifactId>
             <version>2.0.2</version>
         </dependency>
         <!--spark sql-->
         <dependency>
             <groupId>org.apache.spark</groupId>
             <artifactId>spark-sql_2.11</artifactId>
             <version>2.0.2</version>
         </dependency>
</dependencies>
```

Then, add the packaging and compiling plugins to the pom.xml file:

```
<build>
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
      <source>1.8</source>
      <target>1.8</target>
      <encoding>utf-8</encoding>
```

©2013-2024 Tencent Cloud. All rights reserved.

```
    </configuration>
  </plugin>
  <plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <configuration>
      <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
      </descriptorRefs>
    </configuration>
    <executions>
      <execution>
        <id>make-assembly</id>
        <phase>package</phase>
        <goals>
          <goal>single</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</plugins>
</build>
```

Below is an example of a complete pom.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>$yourgroupID </groupId>
    <artifactId>$yourartifactID </artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-core_2.11</artifactId>
            <version>2.0.2</version>
        </dependency>
        <!--spark sql-->
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-sql_2.11</artifactId>
            <version>2.0.2</version>
        </dependency>
```

©2013-2024 Tencent Cloud. All rights reserved.

```
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                    <encoding>utf-8</encoding>
                </configuration>
            </plugin>
            <plugin>
                <artifactId>maven-assembly-plugin</artifactId>
                <configuration>
                    <descriptorRefs>
                        <descriptorRef>jar-with-dependencies</descriptorRef>
                    </descriptorRefs>
                </configuration>
                <executions>
                    <execution>
                        <id>make-assembly</id>
                        <phase>package</phase>
                        <goals>
                            <goal>single</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Note:

Replace $yourgroupID and $yourartifactID with your real information.

Then, create a Java Class named Demo.java in the main>Java folder and add the following code to it:

```
import org.apache.spark.rdd.RDD;
import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.SparkSession;

/**
 * Created by tencent on 2018/6/28.
 */
public class Demo {
```

©2013-2024 Tencent Cloud. All rights reserved.

```
    public static void main(String[] args){
        SparkSession spark = SparkSession
                .builder()
                .appName("Java Spark Hive Example")
                .enableHiveSupport()
                .getOrCreate();

        Dataset<Row> df = spark.read().json(args[0]);

        RDD<Row> test = df.rdd();

        test.saveAsTextFile(args[1]);
    }
}
```

Compiling code and packaging it for upload

Use the local command prompt to enter the project directory and run the following command to compile and package the project:

```
mvn package
```

"Build success" indicates that package is successfully created. You can see the generated .jar package in the target folder under the project directory.
Use the scp or sftp tool to upload the compressed .jar package to the EMR cluster. Run the following command in your local shell:

```
scp $localfile root@public IP address:$remotefolder
```

Here, $localfile is the path and the name of your local file; root is the CVM instance username. You can look up the public IP address in the node information in the EMR or CVM Console. $remotefolder is the path where you want to store the file in the CVM instance. After the upload is completed, you can check whether the file is in the corresponding folder on the EMR command line.

4. Preparing the Data and Running the Demo

You can use SparkSQL to process data stored in HDFS. First, upload the data to HDFS. The built-in file people.json stored in `/usr/local/service/spark/exa-mples/src/main/resources/` is used as an example here. Run the following command to upload it to HDFS:

```
[hadoop@10 hadoop]$ hadoop fs -put
/usr/local/service/spark/examples/src/main/resources/people.json
/user/hadoop
```

©2013-2024 Tencent Cloud. All rights reserved.

You can choose a different test file. Here, `/user/hadoop/` is a folder under HDFS, which you can create if it does not exist.

Run the demo. First, log in to a master node of the EMR cluster and switch to the Hadoop user, as shown in the interactive SparkSQL Console. Run the following command:

```
[hadoop@10spark]$ bin/spark-submit --class Demo --master yarn-client
$yourjarpackage /
/user/hadoop/people.json  /user/hadoop/$output
```

Here, --class is the executed entry class. In this example, Demo is the class, which is also the name of the Java Class you created when adding Hadoop dependencies and sample code. --master is the master URL of the cluster, $yourjarpackage is the package name, and $output is the output folder (**if the $output folder already exists before the command is executed, the program will fail**).

After the program is successfully executed, you can see the result in `/user/hadoop/$output` :

```
[hadoop@172 spark]$ hadoop fs -cat /user/hadoop/$output/part-00000
[null,Michael]
[30,Andy]
[19,Justin]
```

For more spark-submit parameters, run the following commands, or see the official documentation.

```
[hadoop@10spark]$ spark-submit -h
```

©2013-2024 Tencent Cloud. All rights reserved.

Integrating Spark Streaming with Ckafka

Last updated：2025-01-03 14:50:17

Tencent Cloud Elastic MapReduce (EMR) allows you to realize the following streaming applications with CKafka:
Log information stream processing
User behavior record stream processing
Alarm information collection and processing
Messaging

1. Preparations for Development

This job is required to access to CKafka, so you need to create a CKafka instance first. For more information, please see CKafka.
Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, select the Spark component on the software configuration page.

2. Using Kafka Toolkit in EMR Cluster

First, you need to check the private IP and port number of CKafka. Log in to the CKafka Console, select the CKafka instance you want to use, and view its private IP as $kafkaIP in the basic information section, and the port number is generally defaulted to 9092. Create a topic named spark_streaming_test on the topic management page.
Log in to any node (preferably a master one) in the EMR cluster. For more information on how to log in to EMR, please see Logging in to Linux Instances. Here, you can choose to log in with WebShell. Click "Log in" on the right of the desired CVM instance to enter the login page. The default username is `root`, and the password is the one you set when creating the EMR cluster. Once the correct credentials are entered, you can enter the command line interface.
Run the following command in EMR command-line interface to switch to the Hadoop user and go to the directory `/usr/local/service/spark`:

```
[root@172 ~]# su hadoop
[root@172 root]$ cd / usr/local/service/spark
```

Download the installation package Kafka's official website. A Kafka client is recommended as it is most compatible with Tencent Cloud CKafka. Then, Decompress the package and move the extracted folder to the `/opt` directory:

```
[hadoop@172 data]$ tar -xzvf kafka_2.10-0.10.2.0.tgz
[hadoop@172 data]$ mv kafka_2.10-0.10.2.0 /opt/
```

©2013-2024 Tencent Cloud. All rights reserved.

Once the package is decompressed, you can use Kafka. Run the `telnet` command to see whether the EMR cluster is connected to the CKafka instance:

```
[hadoop@172 kafka_2.10-0.10.2.0]$ telnet $kafkaIP 9092
Trying $kafkaIP...
Connected to $kafkaIP.
```

$kafkaIP is the private IP address of the CKafka instance you created.
The following example describes how to test the Kafka toolkit. Log in to the EMR cluster in two WebShell terminals, switch to the Hadoop user, and go to the Kafka installation path:

```
[root@172 ~]# su hadoop
[hadoop@172 root]$ cd /opt/kafka_2.10-0.10.2.0/
```

Connect to CKafka on the first terminal and send the following message:

```
[hadoop@172 kafka_2.10-0.10.2.0]$ bin/kafka-console-producer.sh --broker-list
$kafkaIP:9092
--topic spark_streaming_test
hello world
this is a message
```

Connect to CKafka on the other terminal. Now, as a consumer, you are able to access or consume records from a Kafka cluster:

```
[hadoop@172 kafka_2.10-0.10.2.0]$ bin/kafka-console-consumer.sh --bootstrap-
server
$kafkaIP:9092 --from-beginning --new-consumer --topic spark_streaming_test
hello world
this is a message
```

3. Connecting Spark Streaming to CKafka

On the consumer side, Spark Streaming is used to continuously pull data from CKafka for word frequency counting, i.e. performing the WordCount job on the streaming data. On the producer side, a program is used to constantly generate data which is continuously delivered to CKafka.
Download and install Maven first and then configure its environment variables. If you are using IDE, please configure Maven-related items in your IDE.

Creating a Spark Streaming consumer project

Enter the directory for your Maven project, such as `D://mavenWorkplace` , by running the following commands:

©2013-2024 Tencent Cloud. All rights reserved.

```
mvn   archetype:generate   -DgroupId=$yourgroupID   -
DartifactId=$yourartifactID
-DarchetypeArtifactId=maven-archetype-quickstart
```

Here, $yourgroupID is your package name, $yourartifactID is your project name, and maven-archetype-quickstart indicates to create a Maven Java project. Some files need to be downloaded during the process, so please keep the Internet connected.

After successfully creating the project, you will see a folder named $yourartifactID in the `D://mavenWorkplace` directory. The files included in the folder have the following structure:

```
simple
     ---pom.xml       Core configuration, under the project root directory
     ---src
         ---main
             ---java         Java source code directory
            ---resources    Java configuration file directory
       ---test
           ---java         Test source code directory
           ---resources    Test configuration directory
```

Among the files above, pay extra attention to the pom.xml file and the Java folder under the main directory. The pom.xml file is primarily used to create dependencies and package configurations; the Java folder is used to store your source code.

First, add the Maven dependencies to the pom.xml file:

```
<dependencies>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-core_2.11</artifactId>
            <version>2.0.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-streaming_2.11</artifactId>
            <version>2.0.2</version>
        </dependency>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-streaming-kafka-0-10_2.11</artifactId>
            <version>2.0.2</version>
        </dependency>
</dependencies>
```

Then, add the packaging and compiling plugins to the pom.xml file:

©2013-2024 Tencent Cloud. All rights reserved.

```
<build>
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
      <source>1.8</source>
      <target>1.8</target>
      <encoding>utf-8</encoding>
    </configuration>
  </plugin>
  <plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <configuration>
      <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
      </descriptorRefs>
    </configuration>
    <executions>
      <execution>
        <id>make-assembly</id>
        <phase>package</phase>
        <goals>
          <goal>single</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</plugins>
</build>
```

Note:

Replace $yourgroupID and $yourartifactID with your real information.

Then, add the sample code by creating a Java Class named KafkaTest.java in the main>Java folder and adding the following code to it:

```
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.spark.SparkConf;
import org.apache.spark.api.java.JavaSparkContext;
import org.apache.spark.streaming.Durations;
import org.apache.spark.streaming.api.java.JavaInputDStream;
import org.apache.spark.streaming.api.java.JavaPairDStream;
import org.apache.spark.streaming.api.java.JavaStreamingContext;
import org.apache.spark.streaming.kafka010.ConsumerStrategies;
import org.apache.spark.streaming.kafka010.KafkaUtils;
import org.apache.spark.streaming.kafka010.LocationStrategies;
```

©2013-2024 Tencent Cloud. All rights reserved.

```
import scala.Tuple2;

import java.util.*;
import java.util.concurrent.TimeUnit;

/**
 * Created by tencent on 2018/7/3.
 */
public class KafkaTest {
    public static void main(String[] args) throws InterruptedException {
        String brokers = "$kafkaIP:9092";
        String topics = "spark_streaming_test1";  // Subscribed topics;
multiple topics should be separated by ','
        int durationSeconds = 60;  // Interval
        SparkConf conf = new SparkConf().setAppName("spark streaming word
count");
        JavaSparkContext sc = new JavaSparkContext(conf);
        JavaStreamingContext ssc = new JavaStreamingContext(sc,
Durations.seconds(durationSeconds));
        Collection<String> topicsSet = new HashSet<>
(Arrays.asList(topics.split(",")));
        // Kafka-related parameter
        Map<String, Object> kafkaParams = new HashMap<>();
        kafkaParams.put("metadata.broker.list", brokers) ;
        kafkaParams.put("bootstrap.servers", brokers);
        kafkaParams.put("group.id", "group1");
        kafkaParams.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        kafkaParams.put("key.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        kafkaParams.put("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");
        // Create a connection
        JavaInputDStream<ConsumerRecord<Object,Object>> lines =
KafkaUtils.createDirectStream(
                ssc,
                LocationStrategies.PreferConsistent(),
                ConsumerStrategies.Subscribe(topicsSet, kafkaParams)
        );
        // wordcount logic
        JavaPairDStream<String, Integer> counts = lines
                .flatMap(x -> Arrays.asList(x.value().toString().split("
")).iterator())
                .mapToPair(x -> new Tuple2<String, Integer>(x, 1))
                .reduceByKey((x, y) -> x + y);
        // Save the result
        counts.dstream().saveAsTextFiles("$hdfsPath","result");
```

©2013-2024 Tencent Cloud. All rights reserved.

```
//
        ssc.start();
        ssc.awaitTermination();
        ssc.close();
    }
}
```

Pay attention to the following settings in the code:

The brokers variable should be set to the private IP of the CKafka instance found in step 2.

The topics variable should be set to the name of the topic you created, e.g., spark_streaming_test1 here.

durationSeconds is the interval for the program to consume the data in CKafka, e.g., 60 seconds here.

$hdfsPath is the path in HDFS to which the result will be output.

Use the local command prompt to enter the project directory and run the following command to compile and package the project:

```
mvn package
```

"Build success" indicates that package is successfully created. You can see the generated .jar package in the target folder under the project directory.

Upload the package file to the EMR cluster with the scp or sftp tool. Be sure to include the dependencies in the .jar package to be uploaded:

```
scp $localfile root@public IP address:$remotefolder
```

Here, $localfile is the path and the name of your local file; root is the CVM instance username. You can look up the public IP address in the node information in the EMR or CVM Console. $remotefolder is the path where you want to store the file in the CVM instance. After the upload is completed, you can check whether the file is in the corresponding folder on the EMR command line.

Creating a Spark Streaming producer project

Enter the directory for your Maven project, such as `D://mavenWorkplace` , by running the following commands:

```
mvn archetype:generate -DgroupId=$yourgroupID -DartifactId=$yourartifactID
-DarchetypeArtifactId=maven-archetype-quickstart
```

First, add the Maven dependencies to the pom.xml file:

```
<dependencies>
        <dependency>
            <groupId>org.apache.kafka</groupId>
            <artifactId>kafka_2.11</artifactId>
            <version>0.10.1.0</version>
        </dependency>
        <dependency>
```

©2013-2024 Tencent Cloud. All rights reserved.

```
            <groupId>org.apache.kafka</groupId>
            <artifactId>kafka-clients</artifactId>
            <version>0.10.1.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.kafka</groupId>
            <artifactId>kafka-streams</artifactId>
            <version>0.10.1.0</version>
        </dependency>
    </dependencies>
```

Then, add the packaging and compiling plugins to the pom.xml file:

```
<build>
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
      <source>1.8</source>
      <target>1.8</target>
      <encoding>utf-8</encoding>
    </configuration>
  </plugin>
  <plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <configuration>
      <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
      </descriptorRefs>
    </configuration>
    <executions>
      <execution>
        <id>make-assembly</id>
        <phase>package</phase>
        <goals>
          <goal>single</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</plugins>
</build>
```

Note:

Replace $yourgroupID and $yourartifactID with your real information.

©2013-2024 Tencent Cloud. All rights reserved.

Then, add the sample code by creating a Java Class named SendData.java in the main>Java folder and adding the following code to it:

```
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;
import java.util.Properties;


/**
 * Created by tencent on 2018/7/4.
 */
public class SendData {
    public static void main(String[] args) {

        Properties props = new Properties();
        props.put("bootstrap.servers", "$kafkaIP:9092");
        props.put("acks", "all");
        props.put("retries", 0);
        props.put("batch.size", 16384);
        props.put("linger.ms", 1);
        props.put("buffer.memory", 33554432);
        props.put("key.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        props.put("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");
        // The producer sends a message
        String topic = "spark_streaming_test1";
        org.apache.kafka.clients.producer.Producer<String, String> procuder =
new KafkaProducer<String,String>(props);
        while(true){
            int num = (int)((Math.random())*10);
            for (int i = 0; i <= 10; i++) {
                int tmp = (num+i)%10;
                String value = "value_" + tmp;
                ProducerRecord<String, String> msg = new ProducerRecord<String,
String>(topic, value);
                procuder.send(msg);
            }

            try {Thread.sleep(1000*10);}
            catch (InterruptedException e) {}
        }
    }
}
```

Replace $kafkaIP with the private IP address of your CKafka instance.

©2013-2024 Tencent Cloud. All rights reserved.

This program sends 10 messages from value_0 to value_9 to CKafka every 10 seconds, starting at a random order. For more information on the parameters in the program, please see the consumer program.
Use the local command prompt to enter the project directory and run the following command to compile and package the project:

```
mvn package
```

"Build success" indicates that package is successfully created. You can see the generated .jar package in the target folder under the project directory.
Upload the package file to the EMR cluster with the scp or sftp tool. Be sure to include the dependencies in the .jar package to be uploaded:

```
scp $localfile root@public IP address:$remotefolder
```

Using a program to consume CKafka data

Use two interfaces to log in to the WebShell of the EMR cluster.
In the first interface: log in to a master node of the EMR cluster and switch to the Hadoop user, as shown in section 2. Run the following command to run the demo:

```
[hadoop@172 ~]$ bin/spark-submit --class KafkaTest --master yarn-cluster
$consumerpackage
```

The parameters are as follows:
--class indicates the entry class to be executed, e.g., KafkaTest in this example
--master is the master URL of the cluster.
$consumerpackage is the package name of the packaged consumer program.
After the program is started, it will run continuously in the Yarn cluster. Run the following command to view the status of the program running:

```
[hadoop@172 ~]$ yarn application -list
```

In the second interface: log in to the WebShell of EMR and run the producer program, so that Spark Streaming can retrieve the data for consumption.

```
[hadoop@172 spark]$ bin/spark-submit --class SendData $producerpackage
```

Here, $producerpackage is the package name of the packaged producer program. The result of the wordcount job will be output to the specified HDFS folder in a while. You can view in HDFS the result of Spark Streaming's consumption of the CKafka data:

```
[hadoop@172 root]$ hdfs dfs -ls /user
Found 9 items
drwxr-xr-x - hadoop supergroup  0 2018-07-03 16:37 /user/hadoop
```

©2013-2024 Tencent Cloud. All rights reserved.

```
drwxr-xr-x - hadoop supergroup  0 2018-06-19 10:10 /user/hive
-rw-r--r-- 3 hadoop supergroup 0 2018-06-29 10:19 /user/pythontest.txt
drwxr-xr-x - hadoop supergroup 0 2018-07-05 20:25 /user/sparkstreamingtest-
1530793500000.result

[hadoop@172 root]$ hdfs dfs -cat /user/sparkstreamingtest-
1530793500000.result/*
(value_6,16)
(value_7,22)
(value_8,18)
(value_0,18)
(value_9,17)
(value_1,18)
(value_2,17)
(value_3,17)
(value_4,16)
(value_5,17)
```

Finally, exit the KafkaTest program in the Yarn cluster:

```
[hadoop@172 ~]$ yarn application -kill $Application-Id
```

Here, $Application-Id is the ID found by running the `yarn application -list` command.

For more information on Kafka, please see the official documentation.

©2013-2024 Tencent Cloud. All rights reserved.

Practices on Dynamic Scheduling of Spark Resources

Last updated：2025-01-03 14:50:17

Preparations for Development

Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, you need to select the spark_hadoop component on the software configuration page.
Spark is installed in the `/usr/local/service/` path (`/usr/local/service/spark`) in the CVM instance for the EMR cluster.

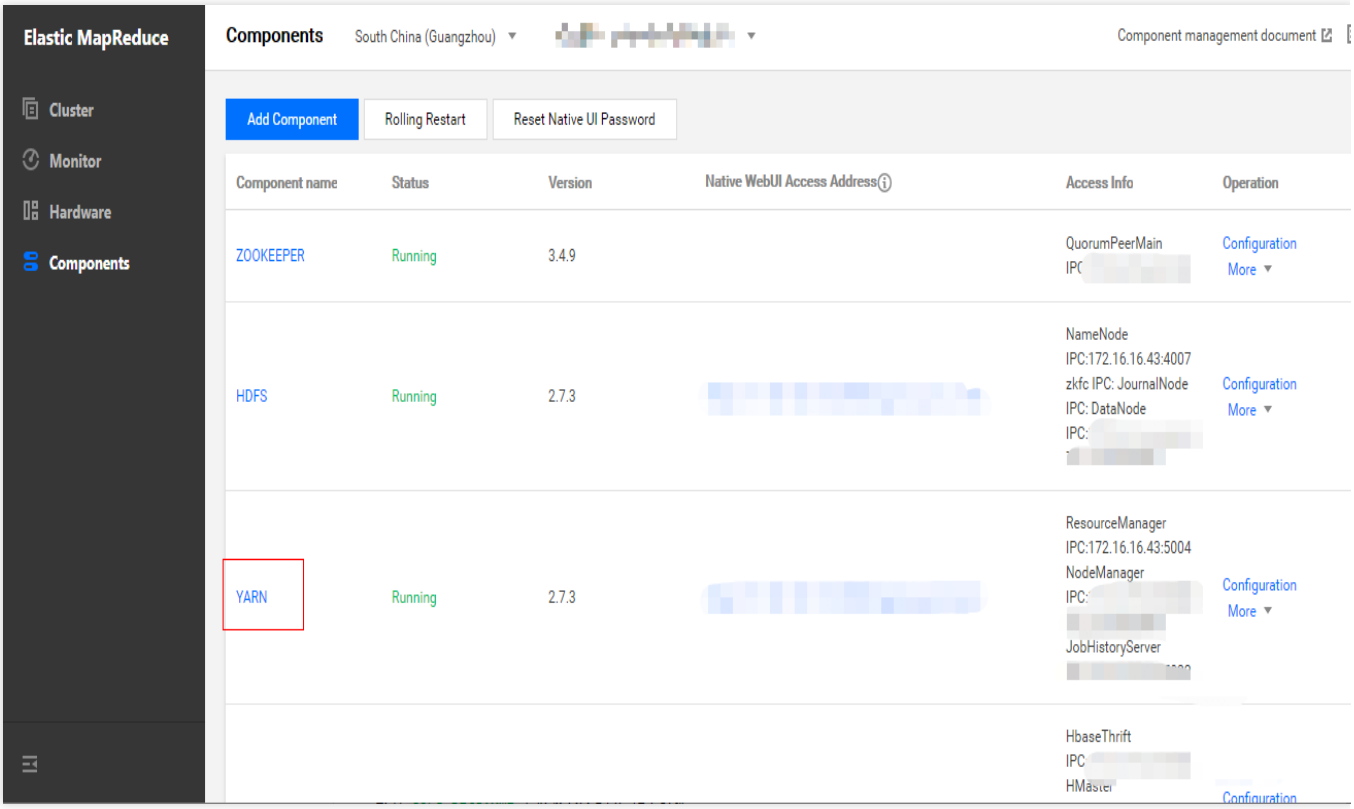
Copying JAR Package

You need to copy `spark-<version>-yarn-shuffle.jar` to the `/usr/local/service/hadoop/share/hadoop/yarn/lib` directory of all nodes in the cluster.

Method 1. Use the SSH Console

1. In **Cluster Service** > **YARN**, select **Operation** > **Role Management** and confirm the IP of the node where NodeManager resides.

©2013-2024 Tencent Cloud. All rights reserved.



2. Log in to the nodes where NodeManager resides one by one.

You need to log in to any node (preferably a master one) in the EMR cluster. For more information on how to log in to EMR, please see Logging in to Linux Instance. Here, you can log in by using XShell.

Use SSH to log in to other nodes where NodeManager resides. The used command is `ssh $user@$ip` , where `$user` is the login username, and `$ip` is the remote server IP (i.e., IP address confirmed in step 1).

Verify that the switch is successful.

©2013-2024 Tencent Cloud. All rights reserved.

```
[root@172 ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 1[redacted] netmask 255.255.240.0  broadcast 1[redacted]
        ether 52:54:00:0d:ad:e3  txqueuelen 1000  (Ethernet)
        RX packets 212406  bytes 110895121 (105.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 178536  bytes 26768271 (25.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 1[redacted]  netmask 255.0.0.0
        loop  txqueuelen 1  (Local Loopback)
        RX packets 378979  bytes 167242376 (159.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 378979  bytes 167242376 (159.4 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

3. Search for the path of the `spark-<version>-yarn-shuffle.jar` file.

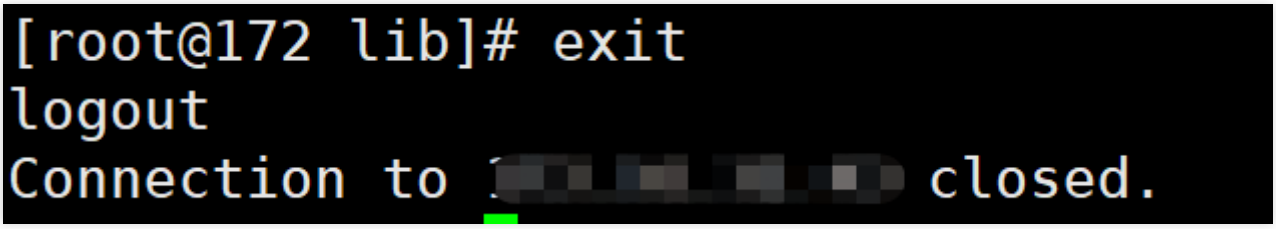
```
[root@172 ~]# find / -name *shuffle.jar
/usr/local/service/hadoop/share/hadoop/yarn/spark-2.3.2-yarn-shuffle.jar
/usr/local/service/spark/yarn/spark-2.3.2-yarn-shuffle.jar
/usr/local/service/hive/spark/yarn/spark-2.0.2-yarn-shuffle.jar
/usr/local/service/apps/kylin-2.5.2/spark/yarn/spark-2.1.2-yarn-shuffle.j
```

4. Copy `spark-<version>-yarn-shuffle.jar` to `/usr/local/service/hadoop/share/hadoop/yarn/lib`.

©2013-2024 Tencent Cloud. All rights reserved.

```
[root@172 ~]# cd /usr/local/service/hadoop/share/hadoop/yarn/lib/
[root@172 lib]# cp /usr/local/service/spark/yarn/spark-2.3.2-yarn-shuffle.jar spark-2.3.2-yarn-shuffle.jar
[root@172 lib]# ls
activation-1.1.jar             hadoop-lzo-0.4.20.jar          jetty-util-6.1.26.jar
aopalliance-1.0.jar            jackson-core-asl-1.9.13.jar    json-io-2.5.1.jar
asm-3.2.jar                    jackson-jaxrs-1.9.13.jar       jsr305-3.0.0.jar
commons-cli-1.2.jar            jackson-mapper-asl-1.9.13.jar  leveldbjni-all-1.8.jar
commons-codec-1.4.jar          jackson-xc-1.9.13.jar          log4j-1.2.17.jar
commons-collections-3.2.2.jar  javassist-3.18.1-GA.jar        netty-3.6.2.Final.jar
commons-compress-1.4.1.jar     java-util-1.9.0.jar            protobuf-java-2.5.0.jar
commons-io-2.4.jar             javax.inject-1.jar             ranger-plugin-classloader-0.7.1.jar
commons-lang-2.6.jar           jaxb-api-2.2.2.jar             ranger-yarn-plugin-impl
commons-logging-1.1.3.jar      jaxb-impl-2.2.3-1.jar          ranger-yarn-plugin-shim-0.7.1.jar
commons-math-2.2.jar           jersey-client-1.9.jar          servlet-api-2.5.jar
curator-client-2.7.1.jar       jersey-core-1.9.jar            spark-2.3.2-yarn-shuffle.jar
curator-test-2.7.1.jar         jersey-guice-1.9.jar           stax-api-1.0-2.jar
fst-2.50.jar                   jersey-json-1.9.jar            xz-1.0.jar
guava-11.0.2.jar               jersey-server-1.9.jar          zookeeper-3.4.6.jar
guice-3.0.jar                  jettison-1.1.jar               zookeeper-3.4.6-tests.jar
guice-servlet-3.0.jar          jetty-6.1.26.jar
[root@172 lib]# ls | grep spark-*
spark-2.3.2-yarn-shuffle.jar
```

5. Log out and switch to other nodes.



Method 2. Use batch deployment script

You need to log in to any node (preferably a master one) in the EMR cluster. For more information on how to log in to EMR, please see Logging in to Linux Instance. Here, you can log in by using XShell.

Write the following Shell script for batch file transfer. When there are many nodes in a cluster, to avoid entering the password for multiple times, you can use sshpass for file transfer. sshpass provides password-free transfer to eliminate your need to enter the password repeatedly; however, the password plaintext is prone to disclosure and can be found with the `history` command.

1. Install sshpass for password-free transfer.

```
[root@172 ~]# yum install sshpass
```

Write the following script:

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

©2013-2024 Tencent Cloud. All rights reserved.

```
nodes=(ip1 ip2 … ipn) # List of IPs of all nodes in the cluster separated by
spaces
len=${#nodes[@]}
password=<your password>
file=" spark-2.3.2-yarn-shuffle.jar "
source_dir="/usr/local/service/spark/yarn"
target_dir="/usr/local/service/hadoop/share/hadoop/yarn/lib"
echo $len
for node in ${nodes[*]}
do
  echo $node;
  sshpass -p $password scp "$source_dir/$file"root@$node:"$target_dir";
done
```

2. Transfer files in a non-password-free manner.

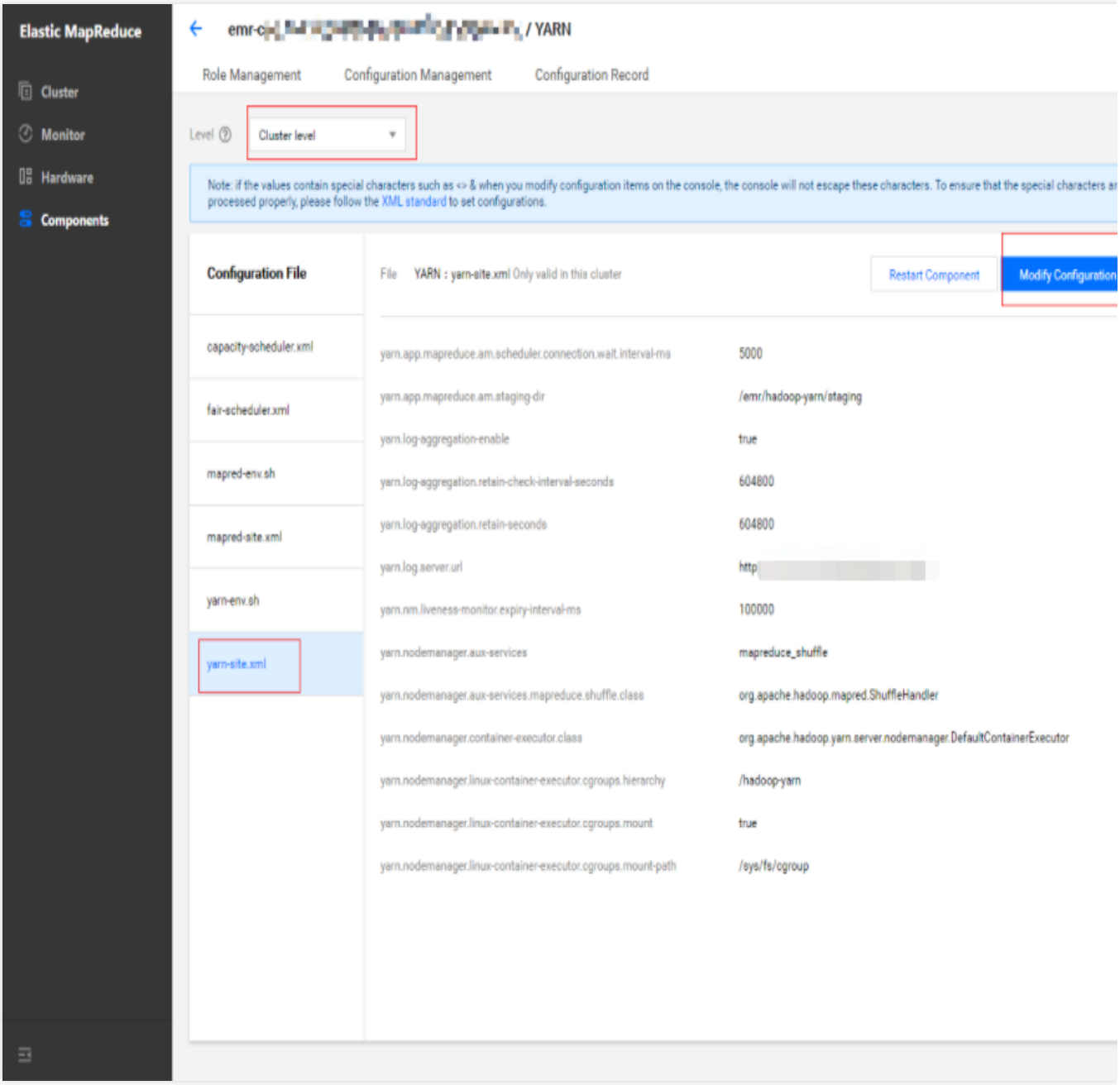
Write the following script:

```
#!/bin/bash
nodes=(ip1 ip2 … ipn) # List of IPs of all nodes in the cluster separated by
spaces
len=${#nodes[@]}
password=<your password>
file=" spark-2.3.2-yarn-shuffle.jar "
source_dir="/usr/local/service/spark/yarn"
target_dir="/usr/local/service/hadoop/share/hadoop/yarn/lib"
echo $len
for node in ${nodes[*]}
do
    echo $node;
    scp "$source_dir/$file" root@$node:"$target_dir";
done
```

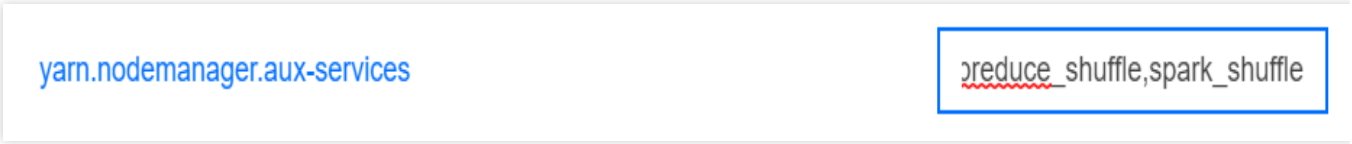
Modifying YARN Configuration

1. In **Cluster Service** > **YARN**, select **Operation** > **Configuration Management**. Select the configuration file `yarn-site.xml` and select "cluster level" as the **level** (modifications of configuration items at the cluster level will be applied to all nodes in the cluster).

©2013-2024 Tencent Cloud. All rights reserved.

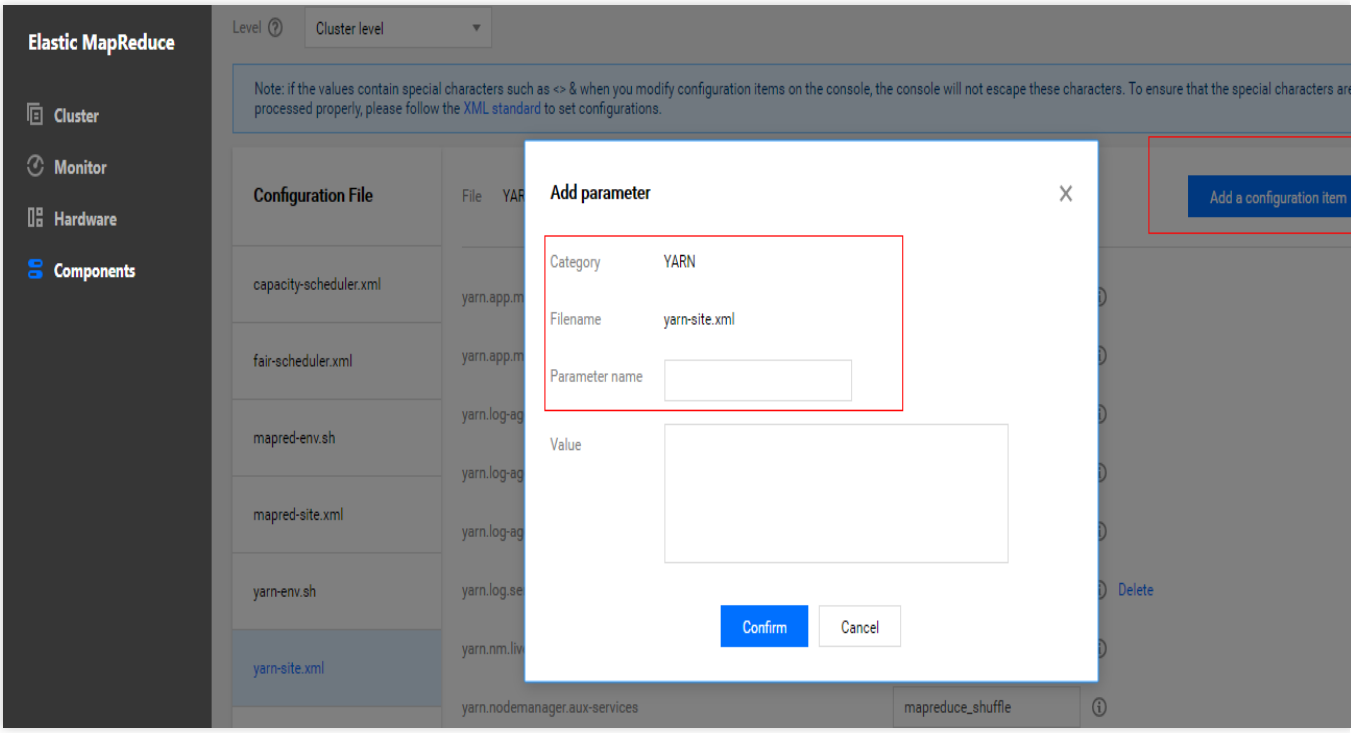


2. Modify the `yarn.nodemanager.aux-services` configuration item and add `spark_shuffle` .

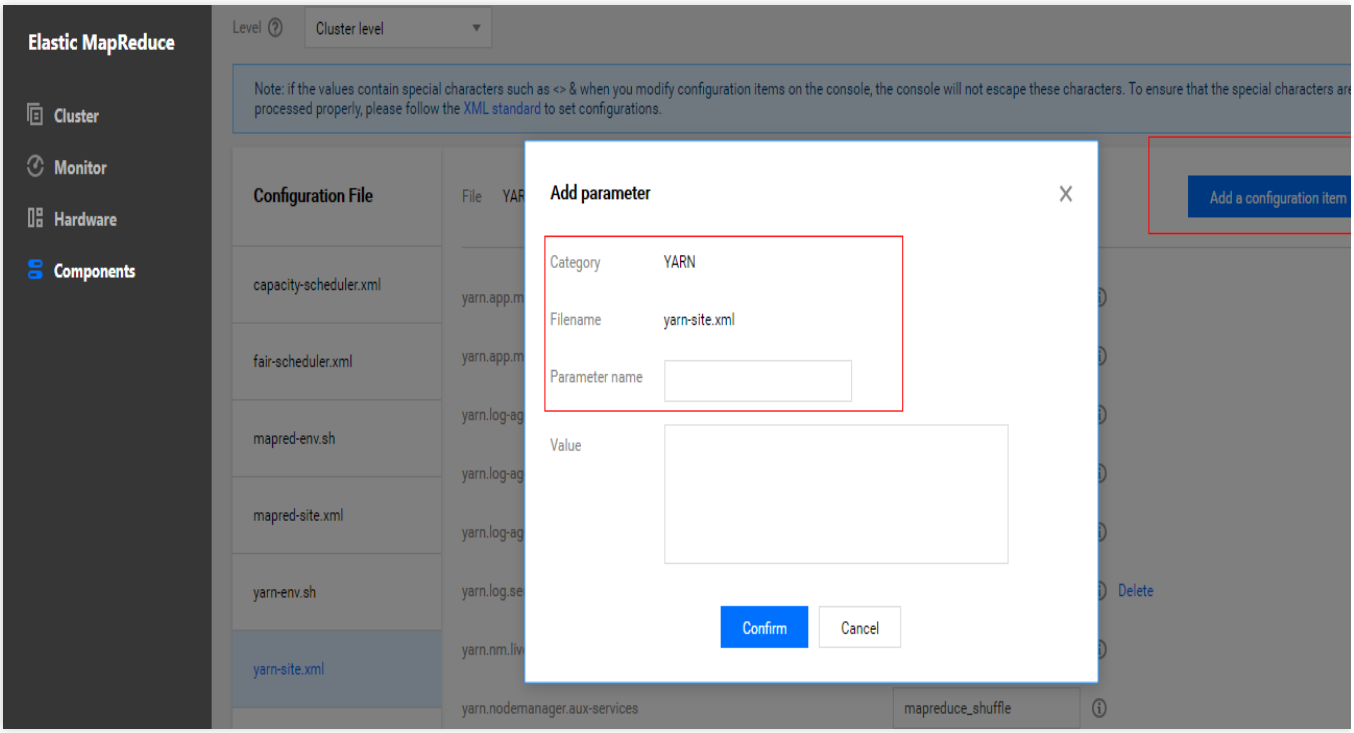


3. Add the configuration item `yarn.nodemanager.aux-services.spark_shuffle.class` and set it to `org.apache.spark.network.yarn.YarnShuffleService` .

©2013-2024 Tencent Cloud. All rights reserved.



4. Add the configuration item `spark.yarn.shuffle.stopOnFailure` and set it to `false`.

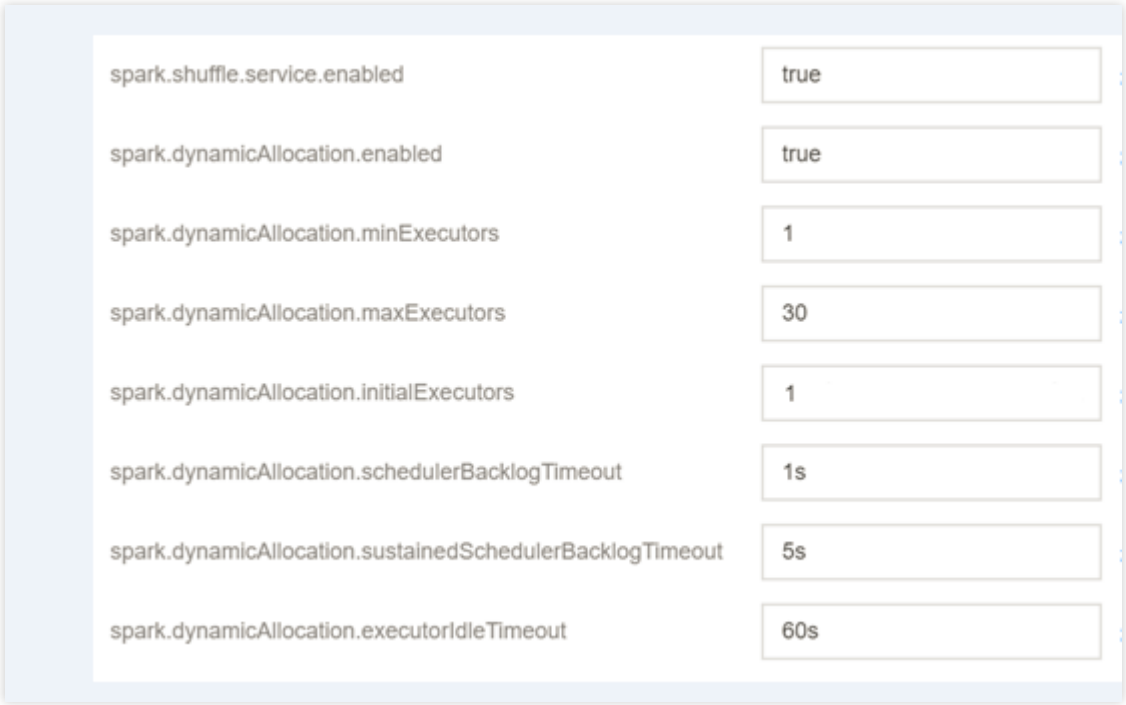


5. Save and distribute the settings. Restart the YARN component for the configuration to take effect.

Modifying Spark Configuration

©2013-2024 Tencent Cloud. All rights reserved.

1. In **Cluster Service** > **SPARK**, select **Operation** > **Configuration Management**.
2. Select the configuration file **spark-defaults.conf**, click **Modify Configuration**, and create configuration items as shown below:



Configuration Item	Value	Remarks
spark.shuffle.service.enabled	true	It starts the shuffle service.
spark.dynamicAllocation.enabled	true	It starts dynamic resource allocation.
spark.dynamicAllocation.minExecutors	1	It specifies the minimum number of executors allocated for each application
spark.dynamicAllocation.maxExecutors	30	It specifies the maximum number of executors allocated for each application
spark.dynamicAllocation.initialExecutors	1	Generally, its value is the same as that of `spark.dynamicAllocation.minExecutors
spark.dynamicAllocation.schedulerBacklogTimeout	1s	If there are pending jobs backlogged for more than this duration, new executors will be requested.
spark.dynamicAllocation.sustainedSchedulerBacklogTimeout	5s	If the queue of pending jobs still exists, will be triggered again once every several seconds. The number of executors requested per round grows

©2013-2024 Tencent Cloud. All rights reserved.

		exponentially compared to the previous round.
spark.dynamicAllocation.executorIdleTimeout	60s	If an executor has been idle for more than this duration, it will be deleted by the application.

3. Save and distribute the configuration and restart the component.

Testing Dynamic Scheduling of Spark Resources

1. Resource configuration description of the testing environment

In the testing environment, there are two nodes where NodeManager is deployed, and each node has a 4 CPU cores and 8 GB memory. The total resources of the cluster are 8 CPU cores and 16 GB memory.

2. Testing job description

Test 1

In the EMR Console, enter the `/usr/local/service/spark` directory, switch to the "hadoop" user, and run `spark-submit` to submit a job. The data needs to be stored in HDFS.

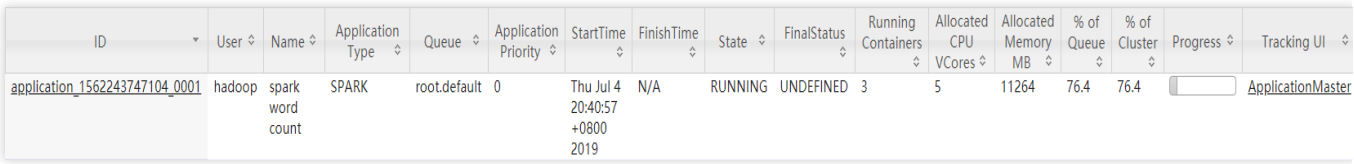
```
[root@172 ~]# cd /usr/local/service/spark/
[root@172 spark]# su hadoop
[hadoop@172 spark]$  hadoop fs -put ./README.md /
[hadoop@172 spark]$ spark-submit --class
org.apache.spark.examples.JavaWordCount --master yarn-client --num-executors 10
--driver-memory 4g --executor-memory 4g --executor-cores 2
./examples/jars/spark-examples_2.11-2.3.2.jar /README.md /output
```

In the "Application" panel of the WebUI of the YARN component, you can view the container and CPU allocation before and after the configuration.

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	% of Queue	% of Cluster	Progress	Tracking UI
application_1562242321020_0001	hadoop	spark word count	SPARK	default	0	Thu Jul 4 20:17:09 +0800 2019	N/A	RUNNING	UNDEFINED	3	3	9920	168.2	67.3		ApplicationMaste

Before dynamic resource scheduling is configured, at most 3 CPUs can be allocated.

©2013-2024 Tencent Cloud. All rights reserved.



ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	% of Queue	% of Cluster	Progress	Tracking UI
application_1562243747104_0001	hadoop	spark word count	SPARK	root.default	0	Thu Jul 4 20:40:57 +0800 2019	N/A	RUNNING	UNDEFINED	3	5	11264	76.4	76.4		ApplicationMaster

After dynamic resource scheduling is configured, up to 5 CPUs can be allocated.

Conclusion: after dynamic resource scheduling is configured, the scheduler will allocate more resources based on the real-time needs of applications.

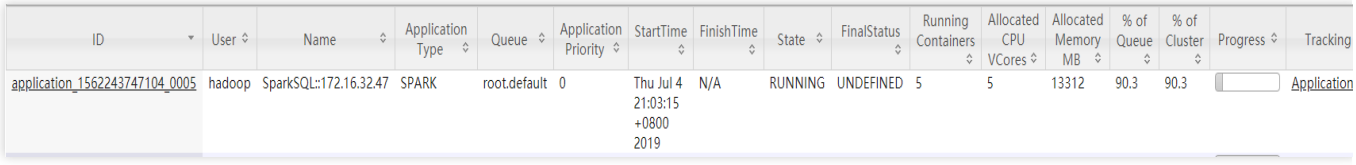
Test 2

In the EMR Console, enter the `/usr/local/service/spark` directory, switch to the "hadoop" user, and run `spark-sql` to start the interactive SparkSQL Console, which is set to use most of the resources in the testing cluster. Configure dynamic resource scheduling and check resource allocation before and after the configuration.

```
[root@172 ~]# cd /usr/local/service/spark/
[root@172 spark]# su hadoop
[hadoop@172 spark]$ spark-sql --master yarn-client --num-executors 5 --driver-
memory 4g --executor-memory 2g --executor-cores 1
```

Use the example for calculating the pi that comes with Spark 2.3.0 as the testing job. When submitting the job, set the number of executors to 5, the driver memory to 4 GB, the executor memory to 4 GB, and the number of executor cores to 2.

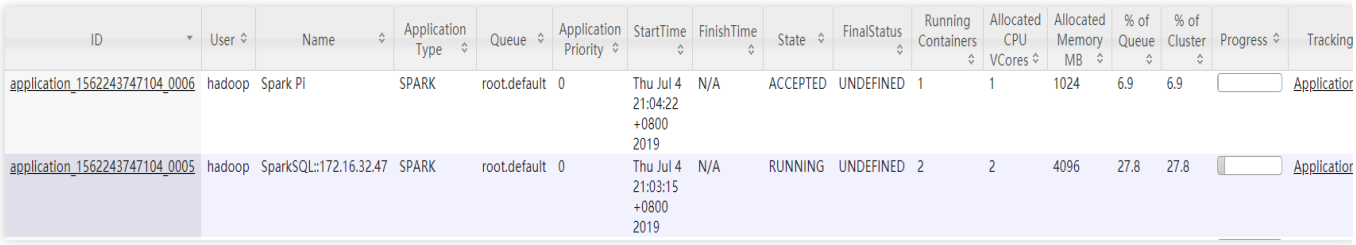
```
[root@172 ~]# cd /usr/local/service/spark/
[root@172 spark]# su hadoop
[hadoop@172 spark]$ spark-submit --class org.apache.spark.examples.SparkPi --
master yarn-client --num-executors 5 --driver-memory 4g --executor-memory 4g --
executor-cores 2 examples/jars/spark-examples_2.11-2.3.2.jar 500
```



ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	% of Queue	% of Cluster	Progress	Tracking
application_1562243747104_0005	hadoop	SparkSQL::172.16.32.47	SPARK	root.default	0	Thu Jul 4 21:03:15 +0800 2019	N/A	RUNNING	UNDEFINED	5	5	13312	90.3	90.3		Application

The resource utilization when only the SparkSQL job is running is 90.3%.

©2013-2024 Tencent Cloud. All rights reserved.



ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	% of Queue	% of Cluster	Progress	Tracking
application_1562243747104_0006	hadoop	Spark Pi	SPARK	root.default	0	Thu Jul 4 21:04:22 +0800 2019	N/A	ACCEPTED	UNDEFINED	1	1	1024	6.9	6.9		Applicatior
application_1562243747104_0005	hadoop	SparkSQL::172.16.32.47	SPARK	root.default	0	Thu Jul 4 21:03:15 +0800 2019	N/A	RUNNING	UNDEFINED	2	2	4096	27.8	27.8		Applicatior

After the SparkPi job is submitted, the resource utilization of SparkSQL becomes 27.8%.

Conclusion: although the SparkSQL job applies for a large amount of resources during submission, no analysis jobs are executed; therefore, there are a lot of idle resources actually. When the idle duration exceeds the limit set by `spark.dynamicAllocation.executorIdleTimeout`, idle executors will be released, and other jobs will get resources. In this test, the cluster resource utilization of the SparkSQL job decreases from 90% to 28%, and idle resources are allocated to the pi calculation job; therefore, automatic scheduling is effective.

Note:

The value of the configuration item `spark.dynamicAllocation.executorIdleTimeout` affects the speed of dynamic resource scheduling. In the test, it is found that the resource scheduling duration is basically the same as this value. You are recommended to adjust this value based on your actual needs for optimal performance.

©2013-2024 Tencent Cloud. All rights reserved.

Spark Integration with Kafka

Last updated：2025-01-03 14:50:17

Dependencies

Starting from v2.3, Spark no longer supports Kafka 0.8.2. As Spark 2.4.3 and later are integrated into EMR in the production environment, you need to integrate Kafka 0.10.0 and later.

How to view

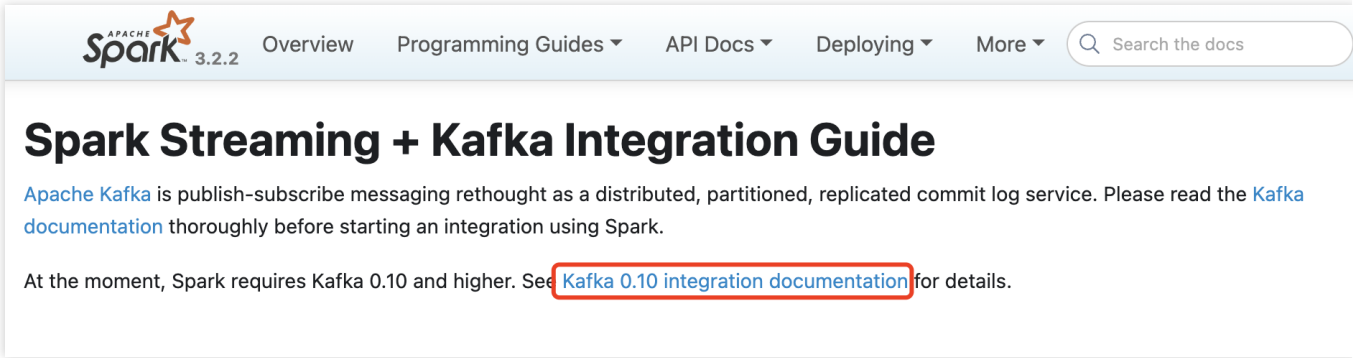
1. Enter the version number in the following URL and open it:

```
https://spark.apache.org/docs/{spark.version}/streaming-kafka-integration.html
```

Replace `{spark.version}` with the actual Spark version. For example, to view the dependencies of v3.2.2, the URL should be as follows:

```
https://spark.apache.org/docs/3.2.2/streaming-kafka-integration.html
```

2. Click the appropriate link to view detailed integration instructions.



©2013-2024 Tencent Cloud. All rights reserved.

Spark Dependencies in Each EMR Version

Last updated：2025-01-03 14:50:16

Dependencies

Spark	Scala	Python	R	Java
2.4.3	2.12.x	2.7+/3.4+	3.1+	8+
3.0.0	2.12.x	2.7+/3.4+	3.1+	8/11
3.0.2	2.12.x	2.7+/3.4+	3.5+	8/11
3.2.1	2.12.x/2.13.x	3.6+	3.5+	8/11
3.2.2	2.12.x/2.13.x	3.6+	3.5+	8/11

How to view

1. Enter the version number in the following URL and open it:

```
https://spark.apache.org/docs/{spark.version}/index.html
```

Replace `{spark.version}` with the actual Spark version. For example, to view the dependencies of v3.2.2, the URL should be as follows:

```
https://spark.apache.org/docs/3.2.2/index.html
```

2. View the dependencies.

©2013-2024 Tencent Cloud. All rights reserved.

Note: Kafka 0.8 support is deprecated as of Spark 2.3.0.

	spark-streaming-kafka-0-8	spark-streaming-kafka-0-10
Broker Version	0.8.2.1 or higher	0.10.0 or higher
API Maturity	Deprecated	Stable
Language Support	Scala, Java, Python	Scala, Java
Receiver DStream	Yes	No
Direct DStream	Yes	Yes
SSL / TLS Support	No	Yes
Offset Commit API	No	Yes
Dynamic Topic Subscription	No	Yes

©2013-2024 Tencent Cloud. All rights reserved.

Hbase Development Guide
Using HBase Through API

Last updated：2025-02-12 16:39:57

HBase is an open-source, high-reliability, high-performance, column-oriented, scalable distributed storage system developed based on Google BigTable. It uses Hadoop file system (HDFS) as the file storage system, Hadoop MapReduce for processing massive amounts of data in HBase, and ZooKeeper for collaboration.
HBase consists of ZooKeeper, HMaster, and HRegionServer. ZooKeeper prevents single points of failure on HMaster, and its master election mechanism guarantees that there is always a master node available in the cluster.
HMaster manages the CRUD operations on the tables and the load balancing of the HRegionServers. In addition, when an HRegionServer exits, it moves the HRegion of that HRegionServer to another.
HRegionServer is the core module in HBase and responsible for reading and writing data from and to HDFS based on user's I/O requests. HRegionServer internally manages a series of HRegion objects, each of which corresponds to a Region and consists of multiple Stores. Each Store corresponds to the storage in the Column Family.
This development guide describes how to use an EMR cluster for development from a technical perspective. For data security reasons, only VPC access is currently supported for EMR.

1. Development Preparations

Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, select the HBase and ZooKeeper components on the software configuration page.

2. Using HBase Shell

Log in to a master node of the EMR cluster first before using HBase Shell. For more information on how to log in to EMR, please see Logging in to Linux Instance Using Standard Login Method. You can choose to log in with WebShell. Click "Log in" on the right of the desired CVM instance to enter the login page. The default username is `root` , and the password is the one you set when creating the EMR cluster. Once the correct information is entered, you can enter the EMR command line interface.
Run the following command on the EMR command line interface to switch to the Hadoop user and go to the directory `/usr/local/service/hbase` :

```
[root@172 ~]# su hadoop
[hadoop@10root]$ cd /usr/local/service/hbase
```

You can enter HBase Shell by running the following command:

©2013-2024 Tencent Cloud. All rights reserved.

```
[hadoop@10hbase]$ bin/hbase shell
```

Enter `help` in HBase Shell to see basic usage information and command examples. Next, create a table by running the following command:

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

Once the table is created, you can run the `list` command to see whether it exists.

```
hbase(main):002:0> list 'test'
TABLE
test
1 row(s) in 0.0030 seconds

=> ["test"]
```

Run the `put` command to add elements to the table you created:

```
hbase(main):003:0> put 'test', 'row1', 'cf:a', 'value1'
0 row(s) in 0.0850 seconds

hbase(main):004:0> put 'test', 'row2', 'cf:b', 'value2'
0 row(s) in 0.0110 seconds

hbase(main):005:0> put 'test', 'row3', 'cf:c', 'value3'
0 row(s) in 0.0100 seconds
```

Three values are added to the created table. The first is "value1" inserted into row "row1" column "cf:a", and so on.
Run the `scan` command to traverse the entire table:

```
hbase(main):006:0> scan 'test'
ROW  COLUMN+CELL
row1   column=cf:a, timestamp=1530276759697, value=value1
row2   column=cf:b, timestamp=1530276777806, value=value2
row3   column=cf:c, timestamp=1530276792839, value=value3
3 row(s) in 0.2110 seconds
```

Run the `get` command to get the value of the specified row in the table:

```
hbase(main):007:0> get 'test', 'row1'
COLUMN  CELL
 cf:a       timestamp=1530276759697, value=value
1 row(s) in 0.0790 seconds
```

Run the `drop` command to delete a table, which should be disabled first before deletion by running the `disable` command:

©2013-2024 Tencent Cloud. All rights reserved.

```
hbase(main):010:0> disable 'test'
hbase(main):011:0> drop 'test'
```

Finally, run the `quit` command to close HBase Shell.

For more HBase Shell commands, please see the official documentation.

3. Using HBase with APIs

Download and install Maven first and then configure its environment variables. If you are using IDE, please configure Maven-related items in your IDE.

Creating Maven project

Enter the directory of the Maven project, such as `D://mavenWorkplace` , and create the project by running the following commands:

```
mvn archetype:generate -DgroupId=$yourgroupID -
DartifactId=$yourartifactID
-DarchetypeArtifactId=maven-archetype-quickstart
```

Here, `$yourgroupID` is your package name, `$yourartifactID` is your project name, and `maven-archetype-quickstart` indicates to create a Maven Java project. Some files need to be downloaded during the project creation, so please keep the network connected.

After successfully creating the project, you will see a folder named `$yourartifactID` in the `D://mavenWorkplace` directory. The files included in the folder have the following structure:

```
simple
     ---pom.xml       Core configuration, under the project root directory
     ---src
        ---main
           ---java         Java source code directory
           ---resources    Java configuration file directory
       ---test
           ---java         Test source code directory
           ---resources    Test configuration directory
```

Among the files above, pay extra attention to the `pom.xml` file and the Java folder under the main directory. The `pom.xml` file is primarily used to create dependencies and package configurations; the Java folder is used to store your source code.

Adding Hadoop dependencies and sample code

First, add the Maven dependencies to the `pom.xml` file:

©2013-2024 Tencent Cloud. All rights reserved.

```
<dependencies>
         <dependency>
                 <groupId>org.apache.hbase</groupId>
                 <artifactId>hbase-client</artifactId>
                 <version>1.2.4</version>
        </dependency>
</dependencies>
```

Then, add the packaging and compiling plugins to the `pom.xml` file:

```
<build>
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
      <source>1.8</source>
      <target>1.8</target>
      <encoding>utf-8</encoding>
    </configuration>
  </plugin>
  <plugin>
    <artifactId>maven-assembly-plugin</artifactId>
    <configuration>
      <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
      </descriptorRefs>
    </configuration>
    <executions>
      <execution>
        <id>make-assembly</id>
        <phase>package</phase>
        <goals>
          <goal>single</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</plugins>
</build>
```

Before adding the sample code, you need to get the ZooKeeper address of the HBase cluster. Log in to any master or core node in EMR, go to the `/usr/local/service/hbase/conf` directory, and view the hbase.zookeeper.quorum configuration in the `hbase-site.xml` file for ZooKeeper's IP address `$quorum` and the `hbase.zookeeper.property.clientPort` configuration for the port number `$clientPort` .

©2013-2024 Tencent Cloud. All rights reserved.

Then, add the sample code by creating a Java Class named `PutExample.java` in the main>java folder and adding the following code to it:

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.*;
import org.apache.hadoop.hbase.client.*;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.hbase.io.compress.Compression.Algorithm;

import java.io.IOException;

/**
 * Created by tencent on 2018/6/30.
 */
public class PutExample {
    public static void main(String[] args) throws IOException {
        Configuration conf = HBaseConfiguration.create();
        conf.set("hbase.zookeeper.quorum","$quorum");
        conf.set("hbase.zookeeper.property.clientPort","$clientPort");
        conf.set("zookeeper.znode.parent", "$znodePath");

        Connection connection = ConnectionFactory.createConnection(conf);
        Admin admin = connection.getAdmin();

        HTableDescriptor table = new
HTableDescriptor(TableName.valueOf("test1"));
        table.addFamily(new
HColumnDescriptor("cf").setCompressionType(Algorithm.NONE));

        System.out.print("Creating table. ");
        if (admin.tableExists(table.getTableName())) {
            admin.disableTable(table.getTableName());
            admin.deleteTable(table.getTableName());
        }
        admin.createTable(table);

        Table table1 = connection.getTable(TableName.valueOf("test1"));
        Put put1 = new Put(Bytes.toBytes("row1"));
        put1.addColumn(Bytes.toBytes("cf"), Bytes.toBytes("a"),
                Bytes.toBytes("value1"));
        table1.put(put1);
        Put put2 = new Put(Bytes.toBytes("row2"));
        put2.addColumn(Bytes.toBytes("cf"), Bytes.toBytes("b"),
                Bytes.toBytes("value2"));
        table1.put(put2);
        Put put3 = new Put(Bytes.toBytes("row3"));
        put3.addColumn(Bytes.toBytes("cf"), Bytes.toBytes("c"),
```

©2013-2024 Tencent Cloud. All rights reserved.

```
                Bytes.toBytes("value3"));
        table1.put(put3);

        System.out.println(" Done.");
    }
}
```

Compiling code and packaging it for upload

Use the local command prompt to enter the project directory and run the following command to compile and package the project:

```
mvn package
```

"Build success" indicates that package is successfully created. You can see the generated .jar package in the target folder under the project directory.
Upload the package file to the EMR cluster with the scp or sftp tool. **Be sure to include the dependencies in the .jar package to be uploaded.** Run the following command in local command line mode:

```
scp $localfile root@public IP address:$remotefolder
```

Here, `$localfile` is the path and the name of your local file; `root` is the CVM instance username. You can look up the public IP address in the node information in the EMR or CVM console. `$remotefolder` is the path where you want to store the file in the CVM instance. After the upload is completed, you can check whether the file is in the corresponding folder on the EMR command line.

4. Running Demo

Log in to a master node of the EMR cluster and switch to the Hadoop user. Run the following command to run the demo:

```
[hadoop@10 hadoop]$ java -jar $package.jar
```

If the console outputs "Done", all operations are completed. You can switch to HBase Shell and run the `list` command to see whether the HBase table is successfully created with the API, and if yes, you can run the `scan` command to see the detailed content of the table.

```
[hadoop@10hbase]$ bin/hbase shell
hbase(main):002:0> list 'test1'
TABLE
Test1
1 row(s) in 0.0030 seconds
```

©2013-2024 Tencent Cloud. All rights reserved.

```
=> ["test1"]
hbase(main):006:0> scan 'test1'
ROW  COLUMN+CELL
row1   column=cf:a, timestamp=1530276759697, value=value1
row2   column=cf:b, timestamp=1530276777806, value=value2
row3   column=cf:c, timestamp=1530276792839, value=value3
3 row(s) in 0.2110 seconds
```

For more information on API usage, please see the official documentation.

©2013-2024 Tencent Cloud. All rights reserved.

Using Hbase with Thrift

Last updated：2025-02-12 16:39:58

Apache Thrift is a software framework used for scalable cross-language services development. It allows you to define data types and service interfaces in a Thrift File through interface definition language (IDL). The Thrift compiler generates your Thrift File into source code which is to be used to build different clients and servers that communicate seamlessly across programming languages.
The Apache Thrift software framework, for scalable cross-language services development, combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Go, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, and OCaml languages.
Thrift server is a Hive-compatible interface for HBase used to support multi-language APIs. The HBase Thrift interface allows other languages to access HBase over Thrift by connecting to a Thrift server that interfaces with the Java client. This section will describe how to connect HBase with Python and Thrift.

1. Prerequisites

Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, select the HBase component on the software configuration page.

2. Using HBase with Python API

HBase on EMR is integrated with Thrift by default, and the Thrift server is started on the Master1 node (the node with a public IP).
Log in to any node (preferably a master one) in the EMR cluster. For information on how to log in to EMR, please see Logging in to Linux Instance Using Standard Login Method. Here, you can use WebShell to log in. Click **Login** on the right of the desired CVM instance to go to the login page. The default username is `root` , and the password is the one you set when creating the EMR cluster. Once your credentials are validated, you can enter the command line interface.
Run the following commands to switch to the Hadoop user and go to the Hbase installation folder:

```
[root@172 ~]# su hadoop
[hadoop@172 root]$ cd /usr/local/service/hbase/
[hadoop@172 hbase]$
```

View the IP address and port number of Thrift in HBase's configuration file:

```
[hadoop@172 hbase]$ vim conf/hbase-site.xml
```

©2013-2024 Tencent Cloud. All rights reserved.

```
<property>
        <name>hbase.master.hostname</name>
        <value>$thriftIP</value>
</property>
<property>
        <name>hbase.regionserver.thrift.port</name>
        <value>$port</value>
</property>
```

Here, `$port` is the port number of the Thrift server.

By default, HBase is connected with Thrift for EMR clusters. So you don't need to install and configure Thrift. Run the following command to check whether the Thrift server has been started:

```
[hadoop@172 hbase]$ jps

4711 ThriftServer
```

The message above indicates that the Thrift server is already running in the background. At this time, you can operate HBase directly with Python.

Load balancing

An HA cluster has two master nodes, and both nodes start Thrift server by default. If load balancing is required, the client code needs a custom policy to distribute requests to the two Thrift servers which are completely independent of each other with no communication.

Preparing data

Use HBase Shell to create an HBase table. If you have already created one through HBase on EMR, skip this step:

```
[hadoop@172 hbase]$ hbase shell

hbase(main):001:0> create 'thrift_test', 'cf'
hbase(main):005:0> list
thrift_test
1 row(s) in 0.2270 seconds

hbase(main):001:0> quit
```

Viewing table in HBase with Python

First, you need to install the Python dependencies. Switch to the root user with the password that is same as the one for EMR cluster, install the python-pip tool first and then dependencies:

```
[hadoop@172 hbase]$ su
```

©2013-2024 Tencent Cloud. All rights reserved.

```
Password: ********
[root@172 hbase]# yum install python-pip
[root@172 hbase]# pip install hbase-thrift
```

Then, switch back to the Hadoop user, create a Python file `Hbase_client.py` , and add the following code to it:

```
#! /usr/bin/env python
#coding=utf-8

from thrift.transport import TSocket,TTransport
from thrift.protocol import TBinaryProtocol
from hbase import Hbase

socket = TSocket.TSocket('$thriftIP ', $port)
socket.setTimeout(5000)

transport = TTransport.TBufferedTransport(socket)
protocol = TBinaryProtocol.TBinaryProtocol(transport)

client = Hbase.Client(protocol)
transport.open()

print client.getTableNames()
```

Note:

Here, `$thriftIP` is the IP address of the master node on the private network, and `$port` is the port number of ThriftService.

Save and run the file, and the table in HBase will be shown in the console:

```
[hadoop@172 hbase]$ python Hbase_client.py
['thrift_test']
```

Creating HBase table with Python

Create a Python file `Create_table.py` and add the following code to it:

```
#! /usr/bin/env python
#coding=utf-8

from thrift import Thrift
from thrift.transport import TSocket,TTransport
from thrift.protocol import TBinaryProtocol
from hbase import Hbase
from hbase.ttypes import ColumnDescriptor,Mutation,BatchMutation,TRegionInfo
from hbase.ttypes import IOError,AlreadyExists
```

©2013-2024 Tencent Cloud. All rights reserved.

```
socket = TSocket.TSocket('$thriftIP ',$port)
socket.setTimeout(5000)

transport = TTransport.TBufferedTransport(socket)
protocol = TBinaryProtocol.TBinaryProtocol(transport)

client = Hbase.Client(protocol)
transport.open()

new_table = ColumnDescriptor(name = 'cf:',maxVersions = 1)
client.createTable('thrift_test_1',[new_table])

tables = client.getTableNames()
socket.close()

print tables
```

The program will add a new table `thrift_test_1` in HBase and output all existing tables:

```
[hadoop@172 hbase]$ python Create_table.py
['thrift_test', 'thrift_test_1']
```

Inserting data into HBase table with Python

Create a Python file `Insert.py` and add the following code to it:

```
#! /usr/bin/env python
#coding=utf-8

from thrift import Thrift
from thrift.transport import TSocket,TTransport
from thrift.protocol import TBinaryProtocol
from hbase import Hbase
from hbase.ttypes import ColumnDescriptor,Mutation,BatchMutation,TRegionInfo
from hbase.ttypes import IOError,AlreadyExists

socket = TSocket.TSocket('$thriftIP ', $port)
socket.setTimeout(5000)

transport = TTransport.TBufferedTransport(socket)
protocol = TBinaryProtocol.TBinaryProtocol(transport)

client = Hbase.Client(protocol)
transport.open()

mutation1 = [Mutation(column = "cf:a",value = "value1")]
client.mutateRow('thrift_test_1',"row1",mutation1)
```

©2013-2024 Tencent Cloud. All rights reserved.

```
mutation2 = [Mutation(column = "cf:b",value = "value2")]
client.mutateRow('thrift_test_1',"row1",mutation2)

mutation1 = [Mutation(column = "cf:a",value = "value3")]
client.mutateRow('thrift_test_1',"row2",mutation1)

mutation2 = [Mutation(column = "cf:b",value = "value4")]
client.mutateRow('thrift_test_1',"row2",mutation2)

socket.close()
```

The program will add two rows of data to the `thrift_test_1` table in HBase, each with two data entries, which can be viewed in HBase Shell:

```
hbase(main):005:0> scan 'thrift_test_1'
ROW       COLUMN+CELL
row1       column=cf:a, timestamp=1530697238581, value=value1
row1       column=cf:b, timestamp=1530697238587, value=value2
row2       column=cf:a, timestamp=1530704886969, value=value3
row2       column=cf:b, timestamp=1530704886975, value=value4
2 row(s) in 0.0190 seconds
```

Viewing data in HBase table with Python

You can view the data by row or scan the entire dataset. Create a Python file `Scan_table.py` and add the following code to it:

```
#! /usr/bin/env python
#coding=utf-8

from thrift import Thrift
from thrift.transport import TSocket,TTransport
from thrift.protocol import TBinaryProtocol
from hbase import Hbase
from hbase.ttypes import ColumnDescriptor,Mutation,BatchMutation,TRegionInfo
from hbase.ttypes import IOError,AlreadyExists

socket = TSocket.TSocket('$thriftIP ', $port)
socket.setTimeout(5000)

transport = TTransport.TBufferedTransport(socket)
protocol = TBinaryProtocol.TBinaryProtocol(transport)

client = Hbase.Client(protocol)
transport.open()
```

©2013-2024 Tencent Cloud. All rights reserved.

```
result1 = client.getRow("thrift_test_1","row1")
print result1
for r in result1:
        print 'the rowname is ',r.row
        print 'the frist value is ',r.columns.get('cf:a').value
        print 'the second value is ',r.columns.get('cf:b').value

scanId = client.scannerOpen('thrift_test_1',"",["cf"])
result2 = client.scannerGetList(scanId,10)
print result2

client.scannerClose(scanId)
socket.close()
```

Use GetRow to get the data of one row, or use scannerGetList to get all the data in the table. The output of the program is as follows:

```
[hadoop@172 hbase]$ python Scan_table.py
[TRowResult(columns={'cf:a': TCell(timestamp=1530697238581, value='value1'),
'cf:b': TCell(timestamp=1530697238587, value='value2')}, row='row1')]
the rowname is  row1
the frist value is  value1
the second value is  value2

[TRowResult(columns={'cf:a': TCell(timestamp=1530697238581, value='value1'),
'cf:b': TCell(timestamp=1530697238587, value='value2')}, row='row1'),
TRowResult(columns={'cf:a': TCell(timestamp=1530704886969, value='value3'),
'cf:b': TCell(timestamp=1530704886975, value='value4')}, row='row2')]
```

As you can see, the data of the first row and the data of the entire table are outputted separately.

Deleting data from HBase with Python

Create a Python file `Delete_row.py` and add the following code to it:

```
#! /usr/bin/env python
#coding=utf-8

from thrift import Thrift
from thrift.transport import TSocket,TTransport
from thrift.protocol import TBinaryProtocol
from hbase import Hbase
from hbase.ttypes import *

socket = TSocket.TSocket('$thriftIP ',$port)
socket.setTimeout(5000)
```

©2013-2024 Tencent Cloud. All rights reserved.

```
transport = TTransport.TBufferedTransport(socket)
protocol = TBinaryProtocol.TBinaryProtocol(transport)

client = Hbase.Client(protocol)
transport.open()

client.deleteAllRow("thrift_test_1","row2")

socket.close()
```

The program will delete the second row of data in the test table. After the program is executed, you can view the table in HBase Shell:

```
[hadoop@172 hbase]$ python Delete_row.py
[hadoop@172 hbase]$ hbase shell

hbase(main):004:0> scan 'thrift_test_1'
ROW     COLUMN+CELL
 row1     column=cf:a, timestamp=1530697238581, value=value1
 row1     column=cf:b, timestamp=1530697238587, value=value2
1 row(s) in 0.2050 seconds
```

At this time, the table contains only the data in the first row

For more information on Thrift operations, please see How to Use Thrift.

©2013-2024 Tencent Cloud. All rights reserved.

Spark on Hbase

Last updated：2025-02-12 16:44:06

For more information on Spark on HBase operations, please see Spark-HBase Connector on GitHub.

©2013-2024 Tencent Cloud. All rights reserved.

MapReduce on Hbase

Last updated：2025-02-12 16:44:06

For more information on MapReduce on Hbase operations (e.g., read and write), please see Use Cases.

©2013-2024 Tencent Cloud. All rights reserved.

Phoenix on Hbase Development Guide

Phoenix Client Usage

Last updated：2025-02-12 16:46:40

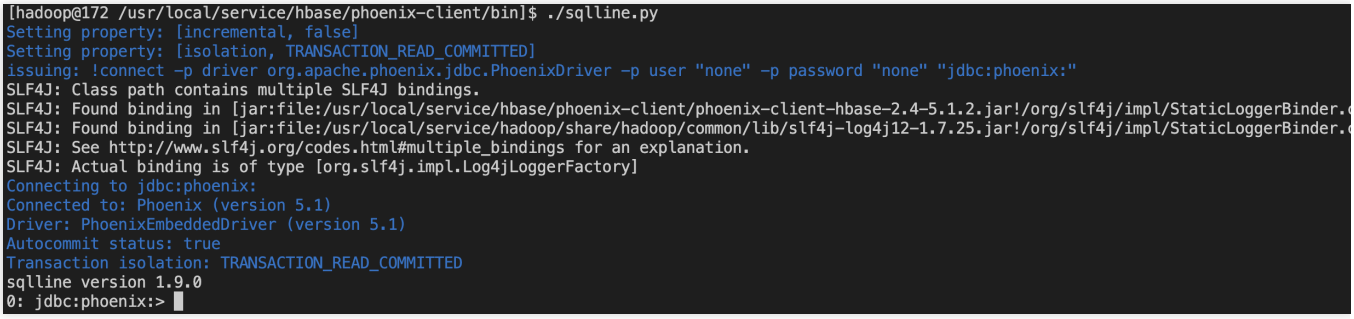
Apache Phoenix is a massively parallel relational database engine supporting OLTP for Hadoop with Apache HBase as its backing store. Phoenix compiles simple queries in just milliseconds and queries millions of rows of data in seconds. By default, the Phoenix client is integrated in HBase-enabled EMR clusters.

1. Start the client.

Switch to the `hadoop` user, go to the `/usr/local/service/hbase/phoenix-client/bin` directory, and use Phoenix's Python command line tool:

```
./sqlline.py
```

If the execution succeeds, the following result will be displayed:



2. Phoenix supports SQL queries. The following are some common operations:

Create a table

```
0: jdbc:phoenix:> CREATE TABLE IF NOT EXISTS TEST (
host char(50) not null,
txn_count bigint
CONSTRAINT pk PRIMARY KEY (host)
);
```

Insert data

```
0: jdbc:phoenix:>UPSERT INTO TEST(host,txn_count) VALUES('192.168.1.1',1);
0: jdbc:phoenix:>UPSERT INTO TEST(host,txn_count) VALUES('192.168.1.2',2);
```

Query data

```
0: jdbc:phoenix:>SELECT * FROM TEST;
```

©2013-2024 Tencent Cloud. All rights reserved.

Delete a data table

```
0: jdbc:phoenix:>DROP TABLE IF EXISTS TEST;
```

For more operations and instructions, see Grammar.

©2013-2024 Tencent Cloud. All rights reserved.

Phoenix JDBC Usage

Last updated：2025-02-12 16:46:40

Adding Maven dependencies

```
<dependency>
     <groupId>org.apache.phoenix</groupId>
     <artifactId>phoenix-core</artifactId>
     <version>${phoenix.version}</version>
</dependency>
```

Here, `phoenix.version` should be consistent with the Phoenix version in the cluster.

Creating a JDBC object

```
Class.forName("org.apache.phoenix.jdbc.PhoenixDriver");
        // Connect to the database
    connection = DriverManager.getConnection("jdbc:phoenix:10.0.0.3:2181,10.0.0.5:2
```

Running a query

```
private static void instertPhoenix(Connection connection)throws Exception{
     String sql="upsert into album_subscribe_log(id,album_id,user_id,op_time,sub_fl
     +" values(?,?,?,?,?,?,?,?,?,?,?,?)";
     PreparedStatement ps=connection.prepareStatement(sql);
     ps.setLong(0,1);
     ps.setLong(1,3);
     ps.setLong(2,1);
     ps.setString(3,"2017-09-05 14:00:00");
     ps.setInt(4,1);
     ps.setString(5,"1");
     ps.setInt(6,3);
     ps.setInt(7,5);
     ps.setInt(8,6);
     ps.setInt(9,7);
     ps.setString(10,"1");
     ps.setString(11,"1");
```

©2013-2024 Tencent Cloud. All rights reserved.

```
        ps.setString(12,"1");
        ps.executeUpdate();
        ps.close();
        connection.commit();
    }
```

©2013-2024 Tencent Cloud. All rights reserved.

Phoenix Practical Tutorial

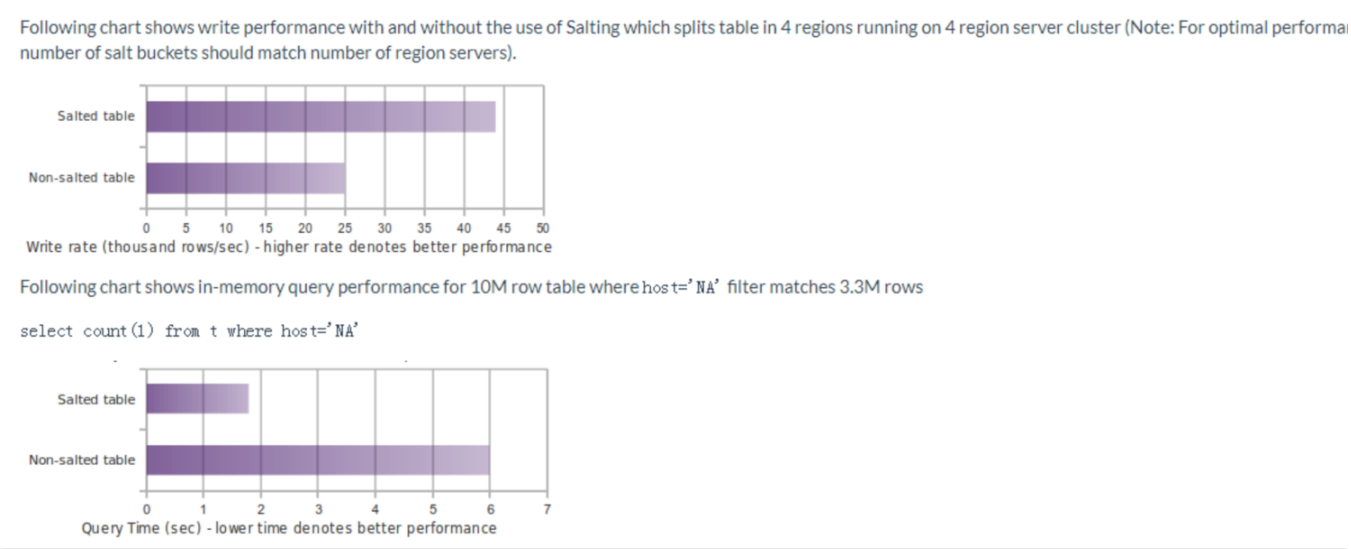
Last updated：2024-10-21 17:28:12

Using Phoenix Salted Table

HBase sequential write may suffer from region server hotspotting if your row key is monotonically increasing. Phoenix provides a way to transparently salt the row key with a salting byte for a particular table. You need to specify this in table creation time by specifying a table property "SALT_BUCKETS" with a value from 1 to 256. Like this:

```
0: jdbc:phoenix:>CREATE TABLE table (a_key VARCHAR PRIMARY KEY, a_col VARCHAR) SALT
```

Salting the row key provides a way to mitigate the problem caused by HBase sequential write. With salted tables, you don't need to be knowledgeable about the row key design in HBase. Below is Phoenix's official performance comparison of read and write performance between salted and non-salted tables:



For more information on salting performance or directions, please see the community documentation for salted tables in Phoenix.

Phoenix Secondary Indexing

Secondary indexes are an orthogonal way to access data from its primary access path. In HBase, you have a single index that is lexicographically sorted on the primary row key. Access to records in any way other than through the primary row requires scanning over potentially all the rows in the table to test them against your filter. With secondary

©2013-2024 Tencent Cloud. All rights reserved.

indexing, the columns or expressions you index form an alternate row key to allow point lookups and range scans along this new axis.

Configuring Secondary Indexing in Phoenix

Phoenix on EMR supports Phoenix's secondary indexing. If you need to use non-transactional, variable indexing, just follow the steps below to configure it. Go to the component management page in the EMR console, click **HBase**, select **Configuration** > **Configuration Management**, and add three configuration items in hbase-site.xml: `hbase.regionserver.wal.codec` , `hbase.region.server.rpc.scheduler.factory.class` , and `hbase.rpc.controllerfactory.class` . The detailed configuration is as follows:

```
    <property>
      <name>hbase.regionserver.wal.codec</name>
      <value>org.apache.hadoop.hbase.regionserver.wal.IndexedWALEditCodec</value>
      </property>
    <property>
      <name>hbase.region.server.rpc.scheduler.factory.class</name>
      <value>org.apache.hadoop.hbase.ipc.PhoenixRpcSchedulerFactory</value>
      <description>Factory to create the Phoenix RPC Scheduler that uses separate qu
    </property>
    <property>
      <name>hbase.rpc.controllerfactory.class</name>
      <value>org.apache.hadoop.hbase.ipc.controller.ServerRpcControllerFactory</valu
      <description>Factory to create the Phoenix RPC Scheduler that uses separate qu
    </property>
```

Using Secondary Indexing in Phoenix

To create a secondary index, run the following command:

```
0: jdbc:phoenix:>CREATE INDEX my_index ON my_table (v1) INCLUDE (v2);
```

For more information on secondary indexing, please see the community documentation of Phoenix secondary indexing.

©2013-2024 Tencent Cloud. All rights reserved.

Hive Development Guide
Hive Overview

Last updated：2024-10-30 11:30:16

Hive is a data warehouse architecture built on the Hadoop file system, offering various features for data warehouse management, including ETL (Extract, Transform, Load) tools, data storage management, and capabilities for querying and analyzing large datasets. Hive also defines a SQL-like development language that allows users to map structured data files to a database table and provides simple SQL query features.
In EMR, Hive is installed in the /usr/local/service/hive path under EMR nodes.
For more details about Hive, see the Apache Hive Official Website.

Hive Service Roles

Role Name	Description
HiveServer2	The ThriftServer service of Hive is used to receive client query requests, perform SQL compilation and parsing, and support multiple client concurrency and authentication. An EMR cluster can deploy multiple HiveServer2 instances, which supports scaling to Router nodes and configuring load balancing.
Hive MetaStore	Hive's metadata service maintains metadata information for Hive databases and Hive tables. The metadata management capability of this module is also integrated with engines such as Spark and Trino. An EMR cluster can deploy multiple Hive MetaStore instances, with support for expansion to Router nodes.
Hive Client	The Hive client provides applications like Beeline and JDBC, allowing users to submit SQL jobs to HiveServer2. Hive service is installed on all nodes where the service is deployed.
Hive WebHCat	WebHCat is a service that provides a REST API for HCatalog, allowing the execution of Hive commands and submission of MapReduce tasks through REST APIs. Multiple WebHCat instances can be deployed within a cluster, with support for scaling to Router nodes.

Internal Table and External Table in Hive

Internal Table: Hive manages both the metadata and the actual data of internal tables. When you use the DROP command to delete an internal table, both the metadata and the corresponding data are deleted. After an internal table

©2013-2024 Tencent Cloud. All rights reserved.

is created, HDFS files are mapped into a table, and Hive's data warehouse generates a corresponding directory. The default warehouse path in EMR is /usr/hive/warehouse/${tablename}, where ${tablename} is the name of the table you create, located on HDFS.

External Table: External tables in Hive are similar to internal tables, but their data is not stored in the directory associated with the table itself; instead, it is stored elsewhere. The benefit of this is that if you delete the external table, the data it points to will not be deleted; only the metadata corresponding to the external table will be removed.

Hive Syntax

Hive in EMR is fully compatible with the open-source community syntax. For more details, see the HiveQL Community Syntax Manual.

©2013-2024 Tencent Cloud. All rights reserved.

Basic Hive Operations

Basic Hive Operations

Last updated：2024-10-30 11:25:22

This document demonstrates how to use Hive on EMR to create databases and tables, import data, and perform basic queries.

Development Preparation

Make sure you have activated Tencent Cloud and created an EMR cluster. For more details, see Creating a Cluster.
During the creation of an EMR cluster, select the Hive component in the software configuration interface.
In the example, there are optional contents that require access to Tencent Cloud Object Storage (COS). You can see Creating a Bucket to create a bucket in COS, and enable COS authorization on the Instance Information page of the EMR console.
Note:
You can see Logging in to a Linux Instance for the method to log in to an EMR node. On the cluster details page, select **Cluster Resources** > **Resource Management**, click the corresponding node resource ID to enter the CVM list, and click **Login** on the right to use WebShell to log in to the instance.
The default username for logging in to a Linux instance is root, and the password is the one set by the user during the creation of the EMR. After entering the correct information, you will be taken to the command line interface.
All operations in this document are performed as the hadoop user. Switch the user identity after logging in to the command line interface.

Preparing Sample Data

Log in to the Master node and use the following command in the EMR command line to switch to the hadoop user and navigate to the Hive folder:

```
su hadoop
cd /usr/local/service/hive
```

Create a bash script file named gen_data.sh and add the following code to it:

```
#!/bin/bash
MAXROW=1000000 #Specifies the number of rows of data to generate.
for((i = 0; i < $MAXROW; i++))
do
```

©2013-2024 Tencent Cloud. All rights reserved.

```
 echo $RANDOM, \\"$RANDOM\\"
done
```

Grant permissions and execute the script as follows. This script will generate 1,000,000 pairs of random numbers and save them to the file hive_test.data:

```
chmod +x script name
./gen_data.sh > hive_test.data
```

Use the following command to upload the generated test data to HDFS, where ${hdfspath} is the path on HDFS where you want to store the file:

```
hdfs dfs -put ./hive_test.data /${hdfspath}
```

You can also use data stored in COS. Upload the data to COS. If the data is stored locally, you can use the COS console to upload it. If the data is in the EMR cluster, use the following command to upload the data, where ${bucketname} is the name of the COS bucket you created:

```
hdfs dfs -put ./hive_test.data cosn://${bucketname}/
```

Hive Basic Operations

Log in to the Master node of the EMR cluster, switch to the hadoop user, and enter the Hive command line using the Hive client:

```
hive
```

Creating Databases and Tables

Use the SHOW syntax to display all current databases:

```
hive> show databases;
OK
default
Time taken: 0.26 seconds, Fetched: 1 row(s)
```

Use the CREATE DATABASE syntax to create a database named test:

```
hive> create database if not exists test;
OK
Time taken: 0.176 seconds
```

Use the USE syntax to switch to the created test database:

©2013-2024 Tencent Cloud. All rights reserved.

```
hive> use test;
OK
Time taken: 0.176 seconds
```

Use the CREATE TABLE syntax to create an internal table named hive_test under the test database:

```
hive> create table hive_test (a int, b string)
hive> ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
-- Create the hive_test data table and specify the column delimiter as commas (,).
OK
Time taken: 0.204 seconds
```

Finally, use the SHOW TABLES syntax to check if the table has been created successfully:

```
hive> show tables;
OK
hive_test
Time taken: 0.176 seconds, Fetched: 1 row(s)
```

Importing Data

For data stored in HDFS, use the following command to import it into the table:

```
hive> load data inpath "/${hdfspath}/hive_test.data" into table hive_test;
```

For data stored in COS, use the following command to import it into the table:

```
hive> load data inpath "cosn://${bucketname}/hive_test.data" into table hive_test;
```

Data stored locally in the EMR cluster can also be imported into Hive using the following command:

```
hive>load data local inpath "/${localpath}/hive_test.data" into table hive_test;
```

Note:

${hdfspath} is the path where your file is stored on HDFS, ${bucketname} is your COS bucket name, and ${localpath} is the path where your data is stored locally on the EMR cluster. After the import is completed, the source data will be deleted.

Executing Queries

Query the first 10 elements in the table:

```
hive> select * from hive_test limit 10;
OK
30847   "31583"
```

©2013-2024 Tencent Cloud. All rights reserved.

```
14887   "32053"
19741   "16590"
8104    "20321"
29030   "32724"
27274   "5231"
10028   "22594"
924     "32569"
10603   "27927"
4018    "30518"
Time taken:2.133 seconds, Fetched:10 row(s)
```

Count the total number of rows in the table:

```
hive> select count(*) from hive_test;
OK
1000000
Time taken:18.504 seconds, Fetched:1 row(s)
```

Deleting Databases and Tables

Use the DROP TABLE syntax to delete a Hive table:

```
hive> drop table if exists hive_test;
Moved: 'hdfs://HDFS/usr/hive/warehouse/hive_test' to trash at: hdfs://HDFS/user/had
OK
Time taken: 2.327 seconds
```

Use the DROP DATABASE syntax to delete a Hive database:

```
hive> drop database if exists test;
OK
Time taken: 0.531 seconds
```

©2013-2024 Tencent Cloud. All rights reserved.

Hive Connection Methods

Last updated：2024-11-18 16:46:14

This document introduces three ways to connect to Hive in EMR: using the Hive client, Beeline, and Java.

Development Preparation

Make sure you have activated Tencent Cloud and created an EMR cluster. For details, see Create Cluster.
During the creation of an EMR cluster, select the Hive component in the software configuration interface.
Note:
You can see Logging in to Linux Instance for the method to log in to an EMR node. On the cluster details page, select **Cluster Resources > Resource Management**, click the corresponding node resource ID to enter the CVM list, and click **Login** on the right to use WebShell to log in to the instance.
The default username for logging in to a Linux instance is root, and the password is the one set by the user during the creation of the EMR. After entering the correct information, you will be taken to the command line interface.
All operations in this document are performed as the hadoop user. Switch the user identity by using the su hadoop command after logging in to the command line interface.

Connecting to Hive

The Hive service is by default deployed on the Master node. You can also see Scaling Out a Cluster to deploy HiveServer2 on a Router node. This document uses the Master node as an example to connect to the Hive service.

Method I: Using the Hive Client

Log in to the EMR cluster's Master node, switch to the Hadoop user, and execute the following command to enter the Hive command line:

```
hive
```

You can also use the -h parameter to get basic information about Hive commands.

Method II: Connecting to HiveServer2 via Beeline

Log in to the EMR cluster's Master node and connect to Hive using the beeline command:

```
beeline -u "jdbc:hive2://${hs2_ip}:${hs2_port}" -n hadoop
```

Note:

©2013-2024 Tencent Cloud. All rights reserved.

1. ${hs2_ip} is the private IP address of the node where the HiveServer2 service is deployed. You can view it on the cluster details page under **Cluster Services > Hive > Role Management**.
2. ${hs2_port} is the port number of the HiveServer2 in the cluster, with a default value of 7001. You can view it on the cluster details page under **Cluster Services > Hive > Configuration Management** by checking the hive.server2.thrift.port setting in the hive-site.xml configuration file.

Method III: Connecting to Hive via Java

This document uses Maven as an example to manage your project. Maven is a project management tool that helps you easily manage project dependencies. It retrieves jar files based on the configuration in the pom.xml file, eliminating the need for manual addition.

First, download and install Maven locally, and configure the Maven environment variables. If you use an IDE, set up the related Maven configuration within the IDE.

In the local shell, navigate to the directory where you want to create a project, for example, /tmp/mavenWorkplace, and enter the following command to create a Maven project:

```
mvn archetype:generate -DgroupId=${yourgroupID} -DartifactId=${yourartifactID}
-DarchetypeArtifactId=maven-archetype-quickstart
```

Note:

1. ${yourgroupID} represents your package name, and ${yourartifactID} represents your project name.
2. maven-archetype-quickstart indicates that you are creating a Maven Java project. During the project creation process, some files need to be downloaded, so ensure a stable internet connection.

Among these, we primarily focus on the pom.xml file and the java folder under main. The pom.xml file is mainly used for dependency and packaging configuration, while the Java folder contains your source code.

First, configure the project dependencies (hadoop-common and hive-jdbc) in the pom.xml file:

```
<dependencies>
        <dependency>
            <groupId>org.apache.hive</groupId>
            <artifactId>hive-jdbc</artifactId>
            <version>${hive_version}</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>${hadoop_version}</version>
        </dependency>
</dependencies>
```

Note:

${hive_version} is the version of Hive in your cluster, and ${hadoop_version} is the version of Hadoop in your cluster.

Next, add the packaging and compilation plugins to the pom.xml file:

©2013-2024 Tencent Cloud. All rights reserved.

```
<build>
<plugins>
 <plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-compiler-plugin</artifactId>
   <configuration>
     <source>1.8</source>
     <target>1.8</target>
     <encoding>utf-8</encoding>
   </configuration>
 </plugin>
 <plugin>
   <artifactId>maven-assembly-plugin</artifactId>
   <configuration>
     <descriptorRefs>
     <descriptorRef>jar-with-dependencies</descriptorRef>
     </descriptorRefs>
   </configuration>
   <executions>
     <execution>
       <id>make-assembly</id>
       <phase>package</phase>
       <goals>
         <goal>single</goal>
       </goals>
     </execution>
   </executions>
 </plugin>
</plugins>
</build>
```

Right-click under src > main > java to create a Java class, enter your class name (in this example, App.java), and add the sample code to the class:

```
package org.example;

import java.sql.*;
/**
 * Created by tencent on 2023/8/11.
 */
public class App {
    private static final String DRIVER_NAME = "org.apache.hive.jdbc.HiveDriver";
    public static void main(String[] args) throws SQLException {
        try {
            // Load hive-jdbc driver
            Class.forName(DRIVER_NAME);
        } catch (ClassNotFoundException e) {
```

©2013-2024 Tencent Cloud. All rights reserved.

```
            e.printStackTrace();
            System.exit(1);
        }
        // Get the connection using the provided connection information and account
        Connection conn = DriverManager.getConnection("jdbc:hive2://$hs2_ip:$hs2_po
        // Create a statement (use conn.prepareStatement(sql) to prevent SQL inject
        Statement stmt = conn.createStatement();
        // The following are basic table creation, data insertion, and data query o
        String tableName = "hive_test";
        stmt.execute("drop table if exists " + tableName);
        stmt.execute("create table " + tableName + " (key int, value string)");
        System.out.println("Create table success!");
        // sshow tables
        String sql = "show tables '" + tableName + "'";
        System.out.println("Running: " + sql);
        ResultSet res = stmt.executeQuery(sql);
        if (res.next()) {
            System.out.println(res.getString(1));
        }
        // describe table
        sql = "describe " + tableName;
        System.out.println("Running: " + sql);
        res = stmt.executeQuery(sql);
        while (res.next()) {
            System.out.println(res.getString(1) + "\\t" + res.getString(2));
        }
        sql = "insert into " + tableName + " values (42,\\"hello\\"),(48,\\"world\\
        stmt.execute(sql);
        sql = "select * from " + tableName;
        System.out.println("Running: " + sql);
        res = stmt.executeQuery(sql);
        while (res.next()) {
            System.out.println(res.getInt(1) + "\\t" + res.getString(2));
        }
        sql = "select count(1) from " + tableName;
        System.out.println("Running: " + sql);
        res = stmt.executeQuery(sql);
        while (res.next()) {
            System.out.println(res.getString(1));
        }
    }
}
```

The program will first connect to the HiveServer2 service, and then create a table named hive_test in the default database. Afterward, it will insert two elements into the table and output the entire content of the table.

©2013-2024 Tencent Cloud. All rights reserved.

If your Maven configuration is correct and the dependencies have been successfully imported, the entire project can be compiled directly. Navigate to the project directory in the local shell and run the following command to package the entire project:

```
mvn clean package -DskipTests
```

During the process, some files may need to be downloaded; the appearance of build success indicates successful packaging. You can then find the packaged jar file in the target folder under the project directory.

Uploading and Running the Program

First, you need to upload the compressed jar file to the EMR cluster using scp or sftp tools. Under the local shell, run the following command (enter yes and then enter the password for verification)

```
scp ${localfile} root@${master_pubilc_ip}:/usr/local/service/hive
```

Note:

1. ${localfile} is the path and name of your local file, root is the CVM server username, and the public IP address can be found in the EMR console's node information or in the CVM console.
2. ${master_public_ip} is the public IP address of your cluster's Master node.

Upload the packaged jar file to the /home/hadoop/ directory in the EMR cluster. After the upload is completed, you can check the corresponding directory in the EMR command line to verify if the file is present.**Make sure to upload JAR packages with dependencies.**

Log in to the EMR cluster, switch to the hadoop user, and navigate to the /home/hadoop/ directory. Run the program:

```
yarn jar ./hive-test-1.0-SNAPSHOT-jar-with-dependencies.jar org.example.App
```

Note:

hive-test-1.0-SNAPSHOT-jar-with-dependencies.jar is the path and name of your jar file, and org.example.App is the package name and class name of the Java class.

The results are as follows:

```
Create table success!
Running: show tables 'hive_test'
hive_test
Running: describe hive_test
key     int
value   string
Running: select * from hive_test
42      hello
48      world
Running: select count(1) from hive_test
2
```

©2013-2024 Tencent Cloud. All rights reserved.

Method IV: Connecting to Hive via Java

This document takes the PyHive Project to connect to Hive through Python 3.

First, log in to the EMR cluster's Master node, switch to the root user, and navigate to the /usr/local/service/hive/ directory. Run the following commands to install the required tools and dependencies:

```
pip3 install sasl
pip3 install thrift
pip3 install thrift-sasl
pip3 install pyhive
```

After the installation is completed, switch back to the hadoop user. Then, create a Python file named hivetest.py under the /usr/local/service/hive/ directory and add the following code:

```
from pyhive import hive

import sys

default_encoding = 'utf-8'

conn = hive.connect(host='${hs2_host}',
                    port='${hs2_port}',
                    username='hadoop',
                    password='hadoop',
                    database='default',
                    auth="CUSTOM",)


tablename = 'HiveByPython'
cur = conn.cursor()

print("\\n")
print('show the tables in default: ')
cur.execute('show tables')
for i in cur.fetchall():
        print(i)

cur.execute('drop table if exists ' + tablename)
cur.execute('create table ' + tablename + ' (key int,value string)')

print("\\n")
print('show the new table: ')
cur.execute('show tables ' +"'" +tablename+"'")
for i in cur.fetchall():
        print(i)

print("\\n")
```

©2013-2024 Tencent Cloud. All rights reserved.

```
print("contents from " + tablename + ":")
cur.execute('insert into ' + tablename + ' values (42,"hello"),(48,"world")')
cur.execute('select * from ' + tablename)
for i in cur.fetchall():
        print(i)
```

After connecting to HiveServer2, the program first outputs all the databases and then displays the tables in the default database. It creates a table named hivebypython, inserts two records into the table, and outputs the results.

Note:

1. ${hs2_host} is the host ID of the HiveServer2 in the cluster. You can find it in the hive.server2.thrift.bind.host setting in the hive-site.xml configuration file under **Cluster Services > Hive > Configuration Management** on the cluster details page.
2. ${hs2_port} is the port number of the HiveServer2 in the cluster, with a default value of 7001. You can view it on the cluster details page under **Cluster Services > Hive > Configuration Management** by checking the hive.server2.thrift.port setting in the hive-site.xml configuration file.

After the file is saved, run the program directly:

```
python3 hivetest.py
```

You will see the following information output in the command line:

```
show the tables in default:

show the new table:
('hivebypython',)

contents from HiveByPython:
(42, 'hello')
(48, 'world')
```

©2013-2024 Tencent Cloud. All rights reserved.

Configuring Hive Execution Engine

Last updated：2024-10-30 11:34:16

Hive supports using MapReduce, Tez, and Spark as execution engines, with MapReduce as the default in EMR. This document will show you how to configure the execution engine in Hive.
It is generally recommended to use the Tez execution engine instead of MapReduce for improved computation efficiency.

Development Preparation

Make sure you have activated Tencent Cloud and created an EMR cluster. For more details, see Creating a Cluster. When an EMR cluster is created, select the Hive component in the software configuration interface. If you need to switch to the Tez or Spark execution engine, check the corresponding component when creating the cluster.

Configuring Engines

Log in to the Hive client, following the directions in Hive Connection Method. The execution engine for Hive is configured using the hive.execution.engine parameter, with the default engine being mr. You can use the following command to check the current execution engine:

```
hive> set hive.execution.engine;
hive.execution.engine=mr
```

Switching Execution Engine in the Command Line

Use the following command to modify the execution engine for the current client session:
Modify the execution engine to Tez:

```
hive> set hive.execution.engine=tez;
```

Modify the execution engine to Spark:

```
hive> set hive.execution.engine=spark;
```

After execution, you can enter the following command to verify:

```
hive> set hive.execution.engine;
```

If the returned information matches the execution engine you set, it indicates that the configuration is successful.

©2013-2024 Tencent Cloud. All rights reserved.

Note:

The above execution engine switch operation is only effective for the current client session; the default execution engine will be used when accessing by other clients or after disconnecting and reconnecting.

Switching Execution Engine in the Configuration File

You can globally switch the default execution engine through the EMR console by navigating to **Cluster Services > HIVE > Configuration Management**. Locate the hive-site.xml configuration file and set the following parameter value:

Modify the execution engine to Tez:

```
hive.execution.engine = tez
```

Modify the execution engine to Spark:

```
hive.execution.engine = sparkP
```

Perform **Save Configuration > Save and Deploy,** and **Restart HiveServer2**. After the restart is completed, the default execution engine of Hive will be modified.

©2013-2024 Tencent Cloud. All rights reserved.

Advanced Usage
Configuring LDAP Authentication

Last updated：2024-10-30 11:35:19

This document introduces the configuration and use of LDAP in Hive on EMR.

Development Preparation

Make sure you have activated Tencent Cloud and created an EMR cluster. For more details, see Creating a Cluster. Create a Hadoop default scene cluster and select the Hive component in the software configuration interface.

Enabling LDAP Authentication

Enter the EMR console, click **Cluster Services > HIVE > Configuration Management**, select the hive-site.xml configuration file, add the following configuration items, set parameter values, save the configuration and deploy it, and then restart HiveServer2:

Parameter	Value	Remarks
hive.server2.authentication	LDAP	Set the authentication mechanism to LDAP.
hive.server2.authentication.ldap.url	ldap://$l{dap_ip}:389	Specify the URL for the LDAP service. ${ldap_ip} represents the IP address of the OPENLDAP service node, which can be found under **Cluster Services >**

©2013-2024 Tencent Cloud. All rights reserved.

		OPENLDAP in the EMR console. For a self-built LDAP service, fill in the details according to your actual setup.
hive.server2.authentication.ldap.baseDN	ou=People,dc=emr,dc=cloud,dc=tencent,dc=com	The Base DN where the LDAP service user is located in EMR. For a self-built LDAP service, fill in the details according to your actual setup.
hive.server2.authentication.ldap.guidKey	cn	The LDAP service username format in EMR. For a self-built LDAP service, fill in the details according to your actual setup.

Accessing HiveServer2

After enabling LDAP authentication, you will need to provide the LDAP username and password to access HiveServer2.

©2013-2024 Tencent Cloud. All rights reserved.

Connecting to Hive with Beeline Client

```
beeline -u "jdbc:hive2://${hs2_ip}:${hs2_port}" -n ${user} -p ${password}
```

Connecting to Hive with JDBC

```
jdbc:hive2://${hs2_ip}:${hs2_port}/default;user=${user};password=${password}
```

Note:

1. ${user} represents the LDAP username.
2. ${password} represents the LDAP password.
3. ${hs2_ip} is the private IP address of the node where the HiveServer2 service is deployed. You can view it on the cluster details page under **Cluster Services > Hive > Role Management**.
4. ${hs2_port} is the port number of the HiveServer2 in the cluster, with a default value of 7001. You can view it on the cluster details page under **Cluster Services > Hive > Configuration Management** by checking the hive.server2.thrift.port setting in the hive-site.xml configuration file.

Note:

After integrating Hive with EMR OpenLDAP and adding a new user, grant 644 permissions to the /emr/hive directory under HDFS for the new user to access.

©2013-2024 Tencent Cloud. All rights reserved.

HiveServer2 CLB

Last updated：2024-10-30 11:36:28

When an EMR cluster has multiple HiveServer2 services, load balancing for HiveServer2 can be achieved using the ZooKeeper service. This document provides a detailed introduction to using HiveServer2 load balancing.

Development Preparation

Make sure you have activated Tencent Cloud and created an EMR cluster. For more details, see Creating a Cluster. Create a **High Availability** Hadoop default scene cluster, and select the Hive component in the software configuration interface.

Achieving HiveServer2 Load Balancing through ZooKeeper

The high-availability cluster in the default Hadoop scene has ZooKeeper service installed by default. You can use the following connection method to connect to HiveServer2, leveraging ZooKeeper to achieve load balancing:

```
beeline -u
'jdbc:hive2://${hive.zookeeper.quorum}/;serviceDiscoveryMode=zooKeeper;zooKeepe
rNamespace=<hive.server2.zookeeper.namespace>' -n ${user} -p ${password}
```

Note:

1. ${hive.zookeeper.quorum} and ${hive.server2.zookeeper.namespace} are related configurations for ZooKeeper Server in Hive. You can find the corresponding values of the hive.zookeeper.quorum and hive.server2.zookeeper.namespace parameters in the hive-site.xml configuration file under EMR console **Cluster Services > HIVE > Configuration Management**.
2. user and password refer to the LDAP username and login password you set.

©2013-2024 Tencent Cloud. All rights reserved.

Hive Metadata Management

Last updated：2024-10-30 10:29:49

When you deploy the Hive component, Hive metadata can be stored in two ways: the first option is the default cluster, where metadata is stored in a separately purchased MetaDB within the cluster; the second option is to associate an external Hive Metastore, where you can choose to link to EMR-MetaDB or a self-built MySQL database, with metadata stored in the associated database, which will not be destroyed when the cluster is terminated.
The default cluster automatically purchases a separate MetaDB CloudDB instance as the metadata storage unit, storing metadata together with other components. This MetaDB CloudDB will be terminated along with the cluster. To retain metadata, you need to manually save it in the CloudDB beforehand.

Note

1. Hive metadata is stored together with the metadata of Druid, Superset, Hue, Ranger, Oozie, and Presto components.
2. The cluster requires a separate purchase of a MetaDB as a metadata storage unit.
3. The MetaDB is terminated along with the cluster, meaning that the metadata is also terminated with the cluster.

Associating EMR-MetaDB to Share Hive Metadata

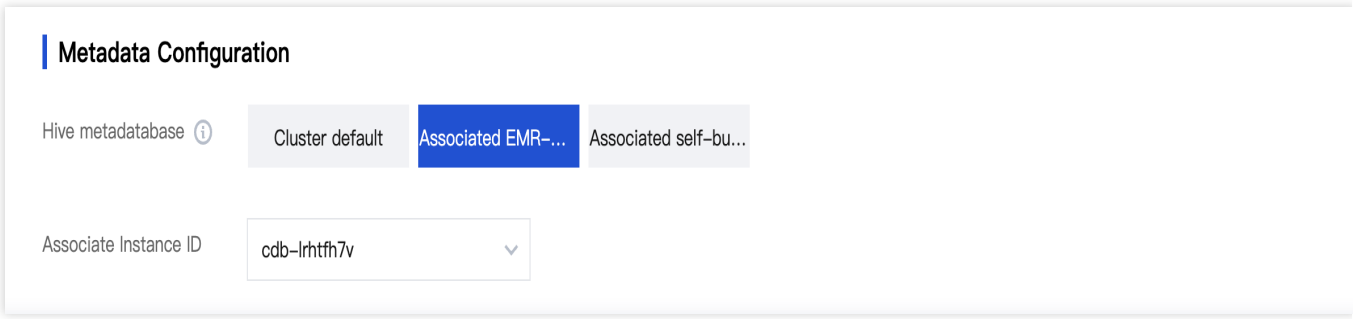
When you create a cluster, the system will pull an available MetaDB from the cloud to store the metadata for the new cluster’s Hive component. This eliminates the need for a separate MetaDB purchase, reducing costs. Moreover, Hive metadata will not be terminated along with the current cluster.

Note

1. The available MetaDB instance ID should be one of the MetaDBs existing under the same account within EMR clusters.
2. When you select one or more of the following components such as Hue, Ranger, Oozie, Druid, and Superset, the system will automatically purchase a MetaDB for storing metadata of components other than Hive.
3. To terminate an associated EMR-MetaDB, you need to do so via CloudDB. Once it is terminated, the Hive Metastore cannot be recovered.
4. Ensure that the network of the associated EMR-MetaDB is in the same network environment as that of the created cluster.

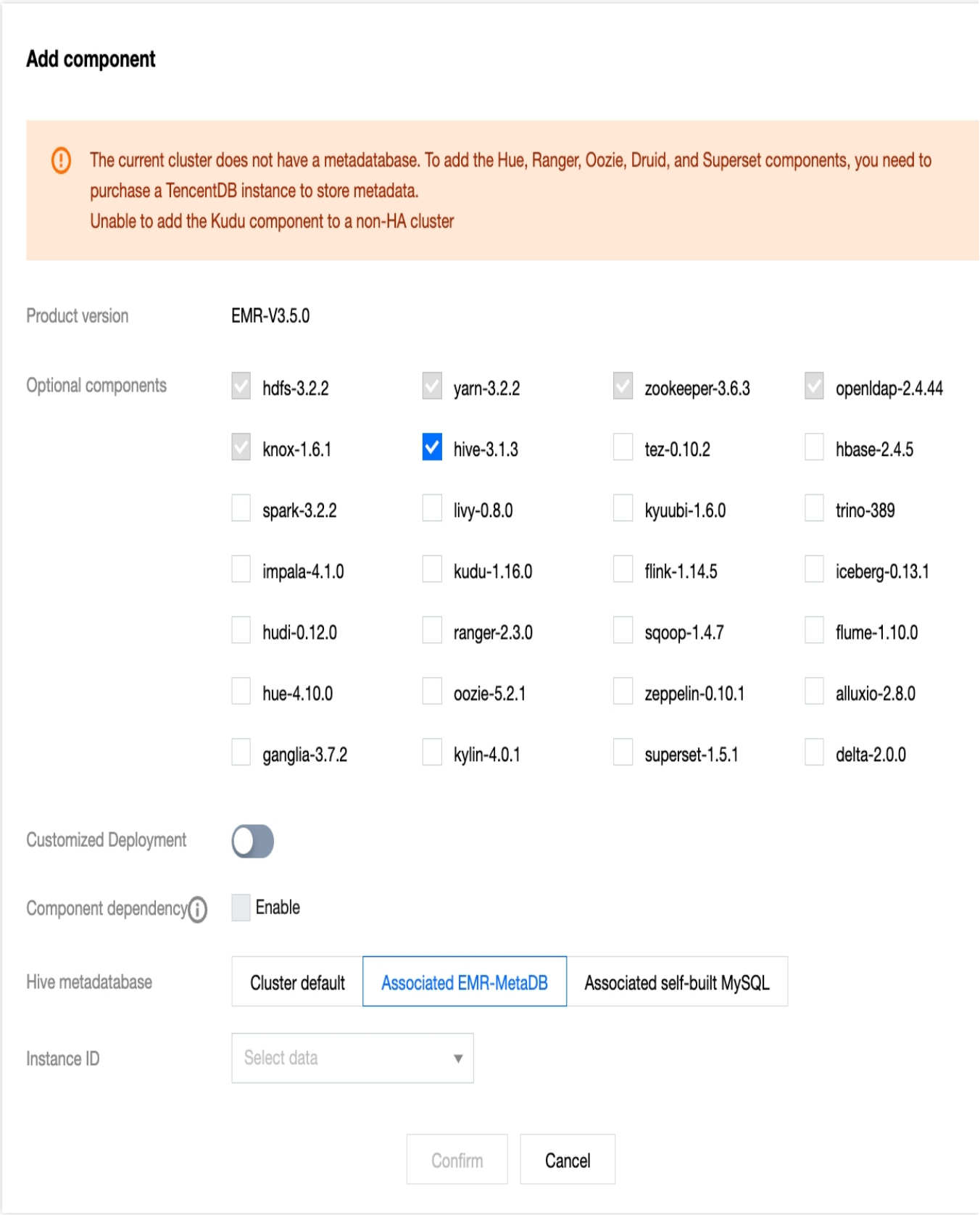
1. After a cluster is created and a Hive component is selected, click **Next** and choose the associated EMR-MetaDB:

©2013-2024 Tencent Cloud. All rights reserved.



2. For clusters without the Hive component installed, when the Hive component is added, select the associated EMR-MetaDB:

©2013-2024 Tencent Cloud. All rights reserved.



Associating Self-Built MySQL to Share Hive Metadata

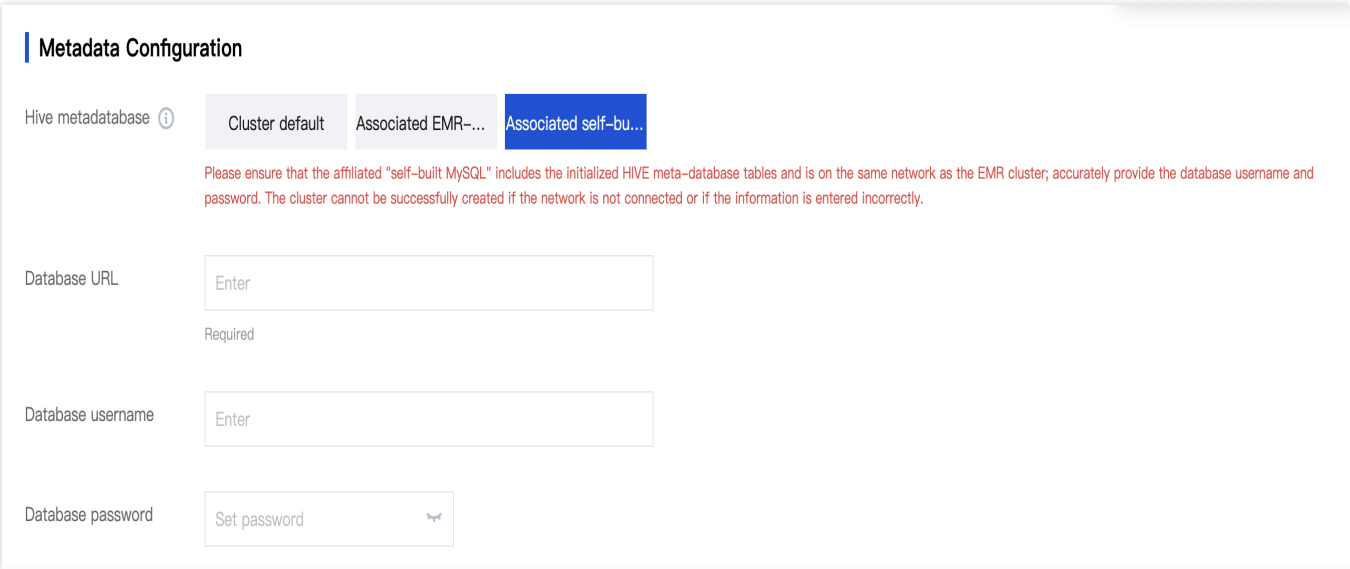
©2013-2024 Tencent Cloud. All rights reserved.

Associating a self-built local MySQL database as Hive metadata storage also avoids the need to purchase a separate MetaDB for storing Hive metadata, thus saving costs. You should accurately enter the local address starting with jdbc:mysql://, the database name, and the database login password, and ensure the network is connected with the current cluster.

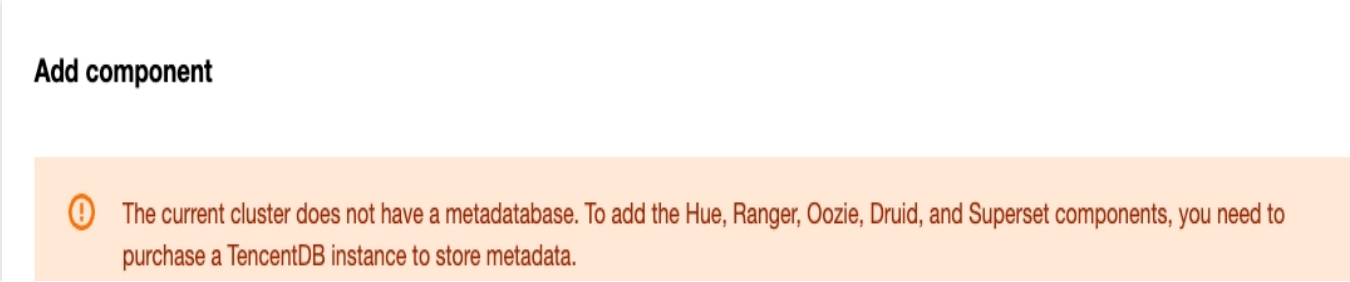
Note

1. Ensure that the self-built database and the EMR cluster are in the same network.
2. Enter the correct database username and password.
3. When you select one or more components such as Hue, Ranger, Oozie, Druid, or Superset, the system will automatically purchase a MetaDB for storing metadata, excluding Hive.
4. Ensure that the Hive metadata version in the custom database is greater than or equal to the Hive version in the new cluster.
5. Ensure that the self-built MySQL contains an initialized Hive Metastore and table.

1. After a cluster is created and a Hive component is selected, click **Next** and link the self-built MySQL database:



2. For clusters without the Hive component installed, when the Hive component is added, link the self-built MySQL database:



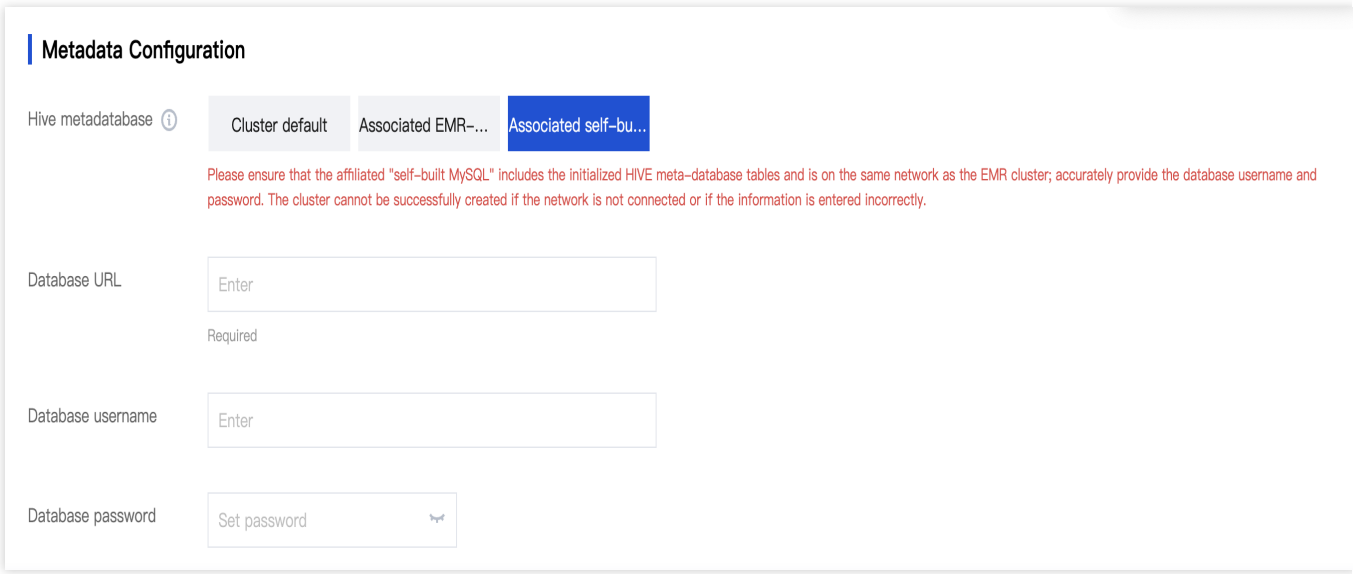
©2013-2024 Tencent Cloud. All rights reserved.

©2013-2024 Tencent Cloud. All rights reserved.

Fixing Issues with HIVE Linking Self-Built Metadata

Selecting a self-built MySQL without HIVE metadata during the creation of an EMR cluster can result in HIVE process issues.

Issue Reproduction



Solutions

For Hive metadata without data, follow the directions below:

Description

Replace ${ip}, ${port}, and ${database} with the actual values used by the user.

1. Stop HiveServer2 (hs2) and metastore for Hive in the console.

2. Modify the hive-site.xml to proto-hive-site.xml for the Hive component and issue it accordingly.Configuration item: javax.jdo.option.ConnectionURL

```
jdbc:mysql://${ip}:${port}/${database}?
useSSL=false&amp;createDatabaseIfNotExist=true&amp;characterEncodin
g=UTF-8
```

3. Delete the database in the CDB:

```
drop database ${database};
```

4. Execute the following command as the hadoop user:

```
/usr/local/service/hive/bin/schematool -dbType mysql -initSchema
```

©2013-2024 Tencent Cloud. All rights reserved.

5. Start hs2 and metastore for Hive from the console.

6. Check if Hive is functioning normally.

If there are character exceptions, execute the following command in CDB:

```
alter database ${database} character set latin1;
flush privileges;
```

©2013-2024 Tencent Cloud. All rights reserved.

Custom Functions UDF

Last updated：2024-10-30 11:37:25

This document introduces custom functions (UDF) as well as their development and usage process.

UDF Classification

UDF Classification	Description
UDF (User Defined Scalar Function)	A custom scalar function, commonly referred to as UDF. It has a one-to-one relationship between input and output, meaning that it reads one row of data and writes out a single output value.
UDTF (User Defined Table-valued Function)	A custom table-valued function, which is used in scenes where a single function call outputs multiple rows of data. It is also the only type of custom function that can return multiple fields.
UDAF (User Defined Aggregation Function)	A custom aggregation function where the relationship between input and output is many-to-one. It aggregates multiple input records into a single output value and can be used in conjunction with the GROUP BY statement in SQL.

For more details, see the community documentation: UDF,UDAF, and UDTF.

Developing UDF

Use an IDE to create a Maven project. The basic project information is as follows; you can customize the groupId and artifactId:

```
<groupId>org.example</groupId>
<artifactId>hive-udf</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
```

Add pom dependency:

```
<!-- https://mvnrepository.com/artifact/org.apache.hive/hive-exec -->
<dependency>
 <groupId>org.apache.hive</groupId>
 <artifactId>hive-exec</artifactId>
 <version>3.1.3</version>
```

©2013-2024 Tencent Cloud. All rights reserved.

```
 <exclusions>
 <exclusion>
 <groupId>org.pentaho</groupId>
 <artifactId>*</artifactId>
 </exclusion>
 </exclusions>
</dependency>
```

Create a class with a name you can customize. This document takes nvl as an example:

Method 1: Extend UDF and override the evaluate method:

```
package org.example;

import org.apache.hadoop.hive.ql.exec.UDF;

public class nvl extends UDF {
 public String evaluate(final String s) {
 if (s == null) { return null; }
 return s + ":HelloWorld";
 }
}
```

Method 2 (recommended for scenes with complex parameters): Extend GenericUDF and override initialize, evaluate, and getDisplayString methods:

```
package org.example;

import org.apache.hadoop.hive.ql.exec.Description;
import org.apache.hadoop.hive.ql.exec.UDFArgumentException;
import org.apache.hadoop.hive.ql.exec.UDFArgumentTypeException;
import org.apache.hadoop.hive.ql.metadata.HiveException;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDF;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDFUtils;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;

@Description(name = "nvl",
        value = "nvl(value, default_value) - Returns default value if value is null
        extended = "Example: SELECT nvl(null, default_value);")
public class MyUDF extends GenericUDF {

    private GenericUDFUtils.ReturnObjectInspectorResolver returnOIResolver;
    private ObjectInspector[] argumentOIs;

    /**
     * Determine the return type based on the parameter types of the function.
     */
    public ObjectInspector initialize(ObjectInspector[] arguments) throws UDFArgume
```

©2013-2024 Tencent Cloud. All rights reserved.

```
        argumentOIs = arguments;
        if(arguments.length != 2) {
            throw new UDFArgumentException("The operator 'NVL' accepts 2 arguments.
        }
        returnOIResolver = new GenericUDFUtils.ReturnObjectInspectorResolver(true);
        if(!(returnOIResolver.update(arguments[0]) && returnOIResolver.update(argum
            throw new UDFArgumentTypeException(2, "The 1st and 2nd args of function
                    + "but they are different: \\""+arguments[0].getTypeName()+"\\"
        }
        return returnOIResolver.get();
    }

    /**
     * Calculate the result. The final result's data type will be determined based
     */
    public Object evaluate(DeferredObject[] arguments) throws HiveException {
        Object retVal = returnOIResolver.convertIfNecessary(arguments[0].get(), arg
        if(retVal == null) {
            retVal = returnOIResolver.convertIfNecessary(arguments[1].get(), argume
        }
        return retVal;
    }

    /**
     * Get the string to display in the explain
     */
    public String getDisplayString(String[] children) {
        StringBuilder builder = new StringBuilder();
        builder.append("if ");
        builder.append(children[0]);
        builder.append(" is null ");
        builder.append("returns ");
        builder.append(children[1]);
        return builder.toString();
    }
}
```

For method 2, package the custom code into a jar file. Execute the following command in the directory containing pom.xml to create the jar file.

```
mvn clean package -DskipTests
```

The target directory will contain the hive-udf-1.0-SNAPSHOT.jar file, indicating that the UDF development work is complete.

©2013-2024 Tencent Cloud. All rights reserved.

Using UDF

Upload the generated JAR to the EMR cluster Master node:

```
scp ./target/hive-udf-1.0-SNAPSHOT.jar
root@${master_public_ip}:/usr/local/service/hive
```

Switch to the Hadoop user and execute the following command to upload the JAR to HDFS:

```
su hadoop
hadoop fs -put ./hive-udf-1.0-SNAPSHOT.jar /
```

View the jar uploaded to HDFS:

```
hadoop fs -ls /
Found 5 items
drwxr-xr-x - hadoop supergroup 0 2023-08-22 09:20 /data
drwxrwx--- - hadoop supergroup 0 2023-08-22 09:20 /emr
-rw-r--r-- 2 hadoop supergroup 3235 2023-08-22 15:39 /hive-udf-1.0-SNAPSHOT.jar
drwx-wx-wx - hadoop supergroup 0 2023-08-22 09:20 /tmp
drwxr-xr-x - hadoop supergroup 0 2023-08-22 09:20 /user
```

Connect to Hive:

```
hive
```

Execute the following command to create a function using the generated JAR package:

```
hive> create function nvl as "org.example.MyUDF" using jar "hdfs:///hive-udf-1.0-SN
```

Note:

1. nvl is the name of the UDF function.

2. org.example.MyUDF is the fully qualified name of the class created in the project.

3. hdfs:///user/hive/warehouse/hiveudf-1.0-SNAPSHOT.jar is the path to the JAR package uploaded to HDFS.

If the following information appears, it indicates that the creation was successful:

```
Added [/data/emr/hive/tmp/1b0f12a6-3406-4700-8227-37dec721297b_resources/hive-udf-1
Added resources: [hdfs:///hive-udf-1.0-SNAPSHOT.jar]
OK
Time taken: 1.549 seconds
```

You can also verify whether the function was successfully created by executing the command SHOW FUNCTIONS LIKE '*nvl*'.

Execute the following command to use the UDF function. The function can be accessed in the same way as that of built-in functions, by directly using the function name:

©2013-2024 Tencent Cloud. All rights reserved.

```
hive> select nvl("tur", "def");
OK
tur
Time taken:0.344 seconds, Fetched:1 row(s)
hive> select nvl(null, "def");
OK
def
Time taken:0.471 seconds, Fetched:1 row(s)
```

©2013-2024 Tencent Cloud. All rights reserved.

Practical Tutorial
Mapping Hbase Tables

Last updated：2024-10-30 11:40:08

You can use Hive to map HBase tables. By doing so, you can read data in HBase with Hive and run Hive-SQL statements to perform operations such as query and insertion on HBase tables.

Preparations for Development

Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, select the Hive and HBase components on the software configuration page.
Hive and its dependencies are installed under the EMR cluster directory `/usr/local/service/`

Creating an HBase Table

First, you need to log in to any node (preferably a master one) in the EMR cluster. For more information on how to log in to EMR, please see Logging in to Linux Instances. Here, you can choose to log in with WebShell. Click "Log in" on the right of the desired CVM instance to enter the login page. The default username is `root` , and the password is the one you set when creating the EMR cluster. Once the correct credentials are entered, you can enter the command line interface.
Run the following command in EMR command-line interface to switch to the Hadoop user and go to the HBase folder to enter HBase Shell:

```
[root@172 ~]# su hadoop
[hadoop@172 ~]# cd /usr/local/service/hbase
[hadoop@10hbase]$ bin/hbase shell
```

Create a table in HBase as shown below:

```
hbase(main):001:0> create 'test', 'cf'
hbase(main):003:0> put 'test', 'row1', 'cf:a', 'value1'
hbase(main):004:0> put 'test', 'row1', 'cf:b', 'value2'
hbase(main):005:0> put 'test', 'row1', 'cf:c', 'value3'
```

For more information on HBase operations, please see the Hbase Operation Guide or official documentation.
After the creation is completed, you can use the `list` and `scan` operations to view the newly created table.

```
hbase(main):001:0> list 'test'
```

©2013-2024 Tencent Cloud. All rights reserved.

```
TABLE
test
1 row(s) in 0.0030 seconds
=> ["test"]

hbase(main):002:0> scan 'test'
ROW  COLUMN+CELL
row1   column=cf:a, timestamp=1530276759697, value=value1
row2   column=cf:b, timestamp=1530276777806, value=value2
row3   column=cf:c, timestamp=1530276792839, value=value3
3 row(s) in 0.2110 seconds
```

Mapping a Hive Table

Switch to the Hive folder and connect to Hive:

```
[hadoop@172 hive]$ cd /usr/local/service/hive/
[hadoop@172 hive]$ bin/hive
```

Next, create a Hive external table and map it to the HBase table created in step 2:

```
hive> CREATE EXTERNAL TABLE hive_test (
    > rowkey string,
    > a string,
    > b string,
    > c string
    > ) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH
    > SERDEPROPERTIES("hbase.columns.mapping" = ":key,cf:a,cf:b,cf:c")
    > TBLPROPERTIES("hbase.table.name" = "test");
OK
Time taken: 2.086 seconds
```

Now, a mapping from the Hive table to the HBase is created. You can run the following command to view the elements in the Hive table:

```
hive> select * from hive_test;
OK
row1 value1     value2 value3
Time taken: 0.305 seconds, Fetched: 1 row(s)
```

©2013-2024 Tencent Cloud. All rights reserved.

Practices on Loading JSON Data to Hive

Last updated：2025-02-12 16:16:58

1. Connect to Hive

Log in to a master node of the EMR cluster, switch to the "hadoop" user, go to the Hive directory, and connect to Hive by running the following command:

```
[root@10 ~]# su hadoop
[hadoop@10 root]$ cd /usr/local/service/hive
```

2. Prepare data

Create a data file in JSON format. Compile the following code and save:

```
vim test.data
{"name":"Mary","age":12,"course":[{"name":"math","location":"b208"},
{"name":"english","location":"b702"}],"grade":[99,98,95]}
{"name":"Bob","age":20,"course":[{"name":"music","location":"b108"},
{"name":"history","location":"b711"}],"grade":[91,92,93]}
```

Store the data file in HDFS:

```
hadoop fs -put ./test.data /
```

3. Create a table

Connect to Hive:

```
[hadoop@10 hive]$ hive
```

Create a table based on the mapping:

```
hive> CREATE TABLE test (name string, age int, course
array<map<string,string>>, grade array<int>) ROW FORMAT SERDE
'org.apache.hive.hcatalog.data.JsonSerDe' STORED AS TEXTFILE;
```

4. Import data

```
hive>LOAD DATA INPATH '/test.data' into table test;
```

5. Check whether data import is successful

Query all data:

©2013-2024 Tencent Cloud. All rights reserved.

```
hive> select * from test;
OK
Mary    12      [{"name":"math","location":"b208"},
{"name":"english","location":"b702"}]    [99,98,95]
Bob     20      [{"name":"music","location":"b108"},
{"name":"history","location":"b711"}]   [91,92,93]
Time taken: 0.153 seconds, Fetched: 2 row(s)
```

Query the first score of each record:

```
hive> select grade[0] from test;
OK
99
91
Time taken: 0.374 seconds, Fetched: 2 row(s)
```

Query the name and location of the first course of each record:

```
hive> select course[0]['name'], course[0]['location'] from test;
OK
math    b208
music   b108
Time taken: 0.162 seconds, Fetched: 2 row(s)
```

©2013-2024 Tencent Cloud. All rights reserved.

Accessing Iceberg Data with Hive

Last updated：2024-10-30 11:41:59

Development Preparation

Make sure you have activated Tencent Cloud and created an EMR cluster. For more details, see Creating a Cluster. During the creation of an EMR cluster, select the Hive, Spark, and Iceberg components in the software configuration interface.

Using Spark to Create an Iceberg Table

Log in to the Master node, switch to the hadoop user, and execute the following command to start SparkSQL:

```
spark-sql --master local[*] --conf
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExt
ensions --conf spark.sql.catalog.local=org.apache.iceberg.spark.SparkCatalog --
conf spark.sql.catalog.local.type=hadoop --conf
spark.sql.catalog.local.warehouse=/usr/hive/warehouse --jars
/usr/local/service/iceberg/iceberg-spark-runtime-3.2_2.12-0.13.0.jar
```

Note:

The Iceberg-related packages are located in the /usr/local/service/iceberg/ directory. The versions of the dependency packages used by --jars may vary between different EMR versions, so check and use the correct dependency packages.

Create a table:

```
spark-sql> CREATE TABLE local.default.t1 (id int, name string) USING iceberg;
Time taken: 2.752 seconds
```

Insert data:

```
spark-sql> INSERT INTO local.default.t1 values(1, "tom");
Time taken: 2.71 seconds
```

Query data:

```
spark-sql> SELECT * from local.default.t1;
1 tom
Time taken: 0.558 seconds, Fetched 1 row(s)
```

©2013-2024 Tencent Cloud. All rights reserved.

Using Hive to View Iceberg Data

Log in to the Master node, switch to the hadoop user, and execute the following command to connect to Hive:

```
hive
```

Add the Iceberg dependency package:

```
hive> add jar /usr/local/service/iceberg/iceberg-hive-runtime-0.13.0.jar;
```

Create an external table:

```
hive> CREATE EXTERNAL TABLE t1
STORED BY 'org.apache.iceberg.mr.hive.HiveIcebergStorageHandler'
LOCATION '/usr/hive/warehouse/default/t1'
TBLPROPERTIES ('iceberg.catalog'='location_based_table');
```

Query the record count of the t1 table:

```
hive> select count(*) from t1;
OK
1
Time taken: 26.255 seconds, Fetched: 1 row(s)
```

©2013-2024 Tencent Cloud. All rights reserved.

Accessing Hudi Data with Hive

Last updated：2024-10-30 11:43:08

Development Preparation

Make sure you have activated Tencent Cloud and created an EMR cluster. For more details, see Creating a Cluster. During the creation of an EMR cluster, select the Hive, Spark, and Hudi components in the software configuration interface.

Reading and Writing Hudi with Spark

Log in to the master node, switch to the hadoop user, and use SparkSQL with the HoodieSparkSessionExtension extension to read and write data:

```
spark-sql --master yarn \\
--num-executors 2 \\
--executor-memory 1g \\
--executor-cores 2 \\
--jars /usr/local/service/hudi/hudi-bundle/hudi-spark3.3-bundle_2.12-0.13.0.jar
\\
--conf 'spark.serializer=org.apache.spark.serializer.KryoSerializer' \\
--conf
'spark.sql.extensions=org.apache.spark.sql.hudi.HoodieSparkSessionExtension' \\
--conf
'spark.sql.catalog.spark_catalog=org.apache.spark.sql.hudi.catalog.HoodieCatalo
g'
```

Note:

Among them, --master specifies your master URL, --num-executors specifies the number of executors, and --executor-memory specifies the executor memory capacity. You can modify these parameters based on your actual requirements. The dependency package versions used by --jars may vary across different EMR versions. **Check and use the correct dependency package located in the /usr/local/service/hudi/hudi-bundle directory.**

Create a table:

```
-- Create a partition table


spark-sql> create table hudi_cow_nonpcf_tbl (
 uuid int,
 name string,
```

©2013-2024 Tencent Cloud. All rights reserved.

```
 price double
) using hudi
tblproperties (
 primaryKey = 'uuid'
);


-- Create a partition table


spark-sql> create table hudi_cow_pt_tbl (
 id bigint,
 name string,
 ts bigint,
 dt string,
 hh string
) using hudi
tblproperties (
 type = 'cow',
 primaryKey = 'id',
 preCombineField = 'ts'
 )
partitioned by (dt, hh);


-- Create a MOR partition table


spark-sql> create table hudi_mor_tbl (
 id int,
 name string,
 price double,
 ts bigint,
 dt string
) using hudi
tblproperties (
 type = 'mor',
 primaryKey = 'id',
 preCombineField = 'ts'
)
partitioned by (dt);
```

Write data:

```
-- insert into non-partitioned table
spark-sql> insert into hudi_cow_nonpcf_tbl select 1, 'a1', 20;
```

©2013-2024 Tencent Cloud. All rights reserved.

```
-- insert dynamic partition
spark-sql> insert into hudi_cow_pt_tbl partition (dt, hh) select 1 as id, 'a1' as n


-- insert static partition
spark-sql> insert into hudi_cow_pt_tbl partition(dt = '2021-12-09', hh='11') select
spark-sql> insert into hudi_mor_tbl partition(dt = '2021-12-09') select 1, 'a1', 20
```

Using Hive to Query Hudi Table

Log in to the Master node, switch to the hadoop user, and execute the following command to connect to Hive:

```
hive
```

Add the Hudi dependency package:

```
hive> add jar /usr/local/service/hudi/hudi-bundle/hudi-hadoop-mr-bundle-0.13.0.jar;
```

View the table:

```
hive> show tables;
OK
hudi_cow_nonpcf_tbl
hudi_cow_pt_tbl
hudi_mor_tbl
hudi_mor_tbl_ro
hudi_mor_tbl_rt
Time taken:0.023 seconds, Fetched:5 row(s)
```

Query data:

```
hive> select * from hudi_cow_nonpcf_tbl;
OK
20230905170525412 20230905170525412_0_0 1 8d32a1cc-11f9-437f-9a7b-8ba9532223d3-0_0-
Time taken:1.447 seconds, Fetched:1 row(s)

hive> set hive.input.format=org.apache.hadoop.hive.ql.io.HiveInputFormat;
hive> select * from hudi_mor_tbl_ro;
OK
20230808174602565       20230808174602565_0_1   id:1    dt=2021-12-09   af40667d-1d
Time taken:0.159 seconds, Fetched:1 row(s)


hive> set hive.vectorized.execution.enabled=false;
```

©2013-2024 Tencent Cloud. All rights reserved.

```
hive> select name, count(*) from hudi_mor_tbl_rt group by name;
a1      1
Time taken:17.618 seconds, Fetched:1 row(s)
```

©2013-2024 Tencent Cloud. All rights reserved.

Creating Databases and Tables in COS/CHDFS with Hive

Last updated：2024-10-30 11:43:59

This document introduces how to create databases and tables in Hive on COS and CHDFS.

Development Preparation

Make sure you have activated Tencent Cloud and created an EMR cluster. When creating an EMR cluster, you need to select the Hive component in the software configuration interface.
The example includes content that requires access to Tencent COS. You can see Creating a Bucket to create a bucket in COS and enable COS authorization on the Instance Information page of the EMR console.
The example includes content that requires access to Tencent Cloud CHDFS. You can see Mounting CHDFS to create a mount point and mount it to the EMR cluster.

Using Hive to Create Databases and Tables in COS

Note:
EMR has integrated Hadoop-COS by default. After enabling COS authorization in the EMR console, you may use a temporary key tied to the Tencent Cloud EMR instance's role to access COS, which is more secure compared to using a fixed key.
For other configuration methods, see COS-Hadoop Tools.

Method I: Creating the Entire Database on COS

Log in to the EMR cluster's Master node, switch to the Hadoop user, and execute the following command to enter the Hive command line:

```
hive
```

Execute the following command to create a database named hivewithcos in your COS bucket.

```
hive> create database hivewithcos location 'cosn://${bucketname}/${path}';
```

Note:
${bucketname} is the name of the COS bucket you created, and ${path} is the storage path.
View the execution result, and you will see the hivewithcos database created on COS.

©2013-2024 Tencent Cloud. All rights reserved.

```
hive> show databases;
OK
default
hivewithcos
Time taken: 0.094 seconds, Fetched: 2 row(s)
```

Create a data table named record (the method to load data into the table is the same as on HDFS):

```
hive> use hivewithcos;
hive> create table record(id int, name string) row format delimited fields terminat
```

View the table:

```
hive> show tables;
OK
record
Time taken: 0.063 seconds, Fetched: 1 row(s)
```

Method II: Placing a Specific Table in COS

Create a database in Hive:

```
hive> create database test;
hive> use test;
```

Execute the following statement to create a table named record in the specified path of the COS bucket (the method to load data into the table is the same as on HDFS):

```
hive> create table record(id int, name string) row format delimited fields terminat
```

View the table:

```
hive> show tables;
OK
record
Time taken: 0.063 seconds, Fetched: 1 row(s)
```

Using Hive to Create Databases and Tables on CHDFS

Method I: Creating the Entire Database on CHDFS

Log in to the EMR cluster's Master node, switch to the Hadoop user, and execute the following command to enter the Hive command line:

©2013-2024 Tencent Cloud. All rights reserved.

```
hive
```

Execute the following command to create a database named hivewithofs in your CHDFS directory:

```
hive> create database hivewithofs location 'ofs://${mountpoint}/${path}';
```

Note:

${mountpoint} is the CHDFS mount address you created, and ${path} is the directory path.

View the execution result, and you will see the hivewithofs database created on CHDFS.

```
hive> show databases;
OK
default
hivewithofs
Time taken: 0.094 seconds, Fetched: 2 row(s)
```

Create a data table named record (the method to load data into the table is the same as on HDFS).

```
hive> use hivewithofs;
hive> create table record(id int, name string) row format delimited fields terminat
```

View the table:

```
hive> show tables;
OK
record
Time taken: 0.063 seconds, Fetched: 1 row(s)
```

Method II: Placing a Specific Table in CHDFS

Create a database test2 in Hive:

```
hive> create database test2;
hive> use test2;
```

Execute the following statement to create a table named record in CHDFS:

```
hive> create table record(id int, name string) row format delimited fields terminat
```

View the table:

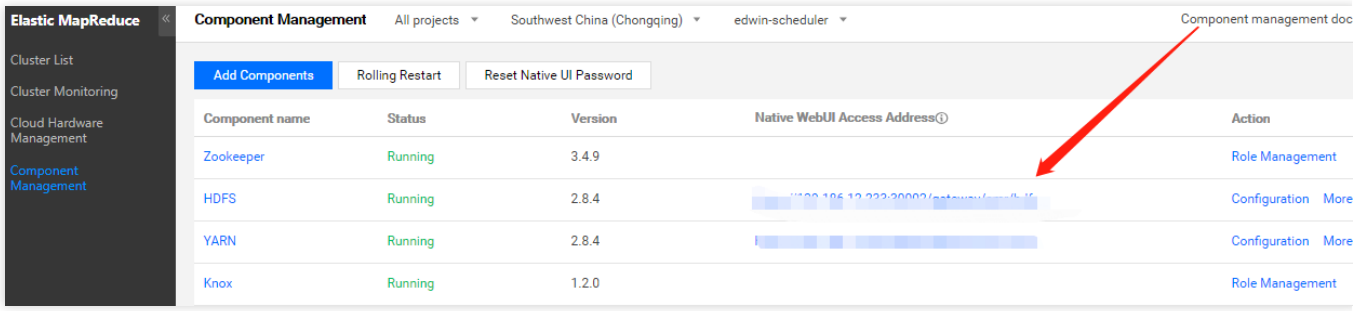
```
hive> show tables;
OK
record
Time taken: 0.063 seconds, Fetched: 1 row(s)
```

©2013-2024 Tencent Cloud. All rights reserved.

Presto Development Guide
Presto Web UI

Last updated：2025-02-12 16:49:06

Presto is an open source distributed SQL query engine for running interactive analytic queries against data sources of all sizes ranging from gigabytes to petabytes. Presto was designed and written from the ground up for interactive analysis and approaches the speed of commercial data warehouses.
Presto allows querying data where it lives, including Hive, Cassandra, relational databases or even proprietary data stores. A single Presto query can combine data from multiple sources, allowing for analysis across your entire organization.
EMR integrates the native web UI of Presto, which you can see in the EMR Console. Log in to the EMR Console, click a cluster instance ID to enter the instance management page. Click **Cluster Service** on the left sidebar to view the entry to the **WebUI Address** page and then click the entry to Presto. The username for login is `root`, and the password is the one set when the cluster was created as shown below:



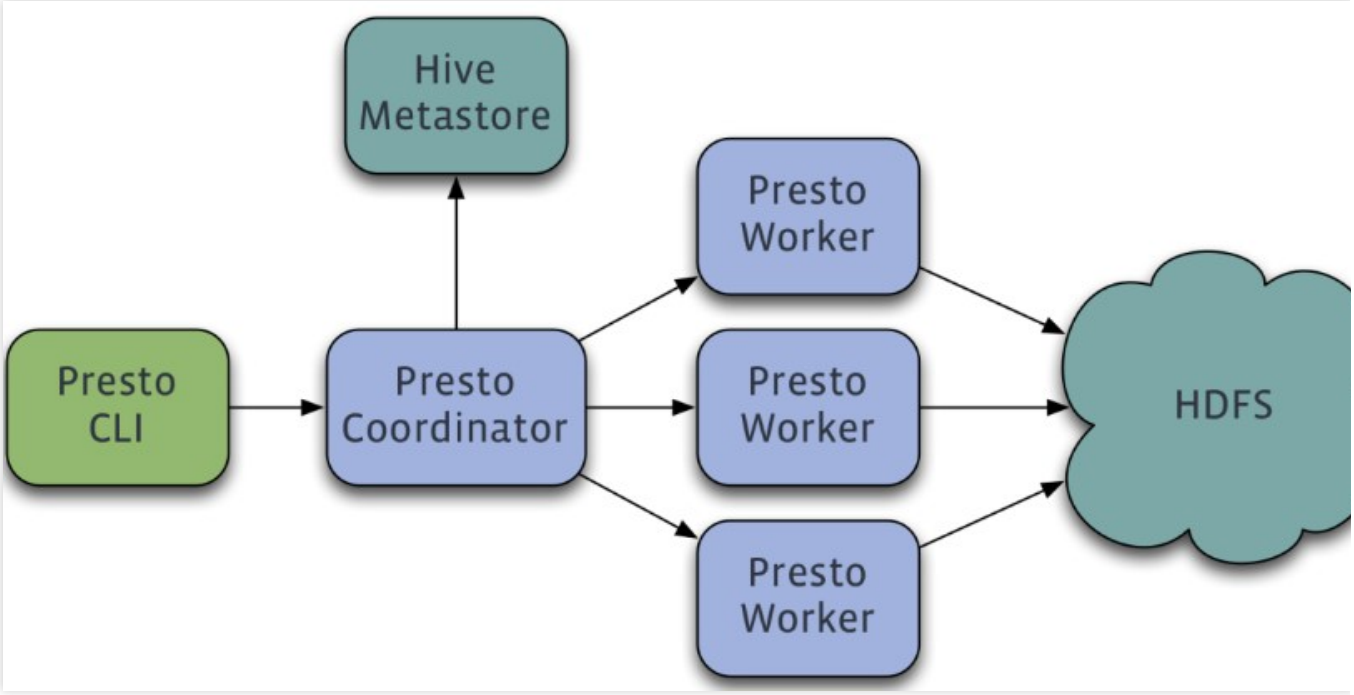
It is required to authenticate the accesses. The username is root, and the default password is the one entered when the cluster was created. To change the password, click **Reset Native UI Password**.

©2013-2024 Tencent Cloud. All rights reserved.

Connector

Last updated：2021-07-09 11:05:00

Presto is a distributed SQL query engine developed by Facebook that is designed to perform high-speed, real-time data analysis for interactive analytical queries of gigabytes to petabytes of data. It supports standard ANSI SQL, including complex queries, aggregations, joins, and window functions. Implemented by Java, it supports various data sources such as Hive, HBase, relational databases, and even proprietary data stores. Below is its architecture diagram:



Presto adopts a distributed system running on multiple servers in a master-slave architecture, which consists of one master node (coordinator) and multiple slave nodes (worker). The client (Presto CLI) is responsible for submitting a query to the Coordinator node which is responsible for parsing the SQL statement, generating the query execution plan, and managing the worker nodes. A worker node is responsible for actually executing the query job.
Presto on EMR is pre-configured with connectors such as Hive, MySQL, and Kafka. In this section, the Hive connector is used as an example to illustrate how Presto reads the data in a Hive table for query. The EMR cluster is configured with environment variables related to presto-client, so you can switch directly to the Hadoop user and use Presto CLI.

1. Preparations for Development

©2013-2024 Tencent Cloud. All rights reserved.

Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, select the Presto component on the software configuration page.

```
Relevant software programs such as Presto are installed in the
`/usr/local/service/` directory of the CVM instance for the EMR cluster.
```

2. Using a Connector to Manipulate Hive

First, you need to log in to any node (preferably a master one) in the EMR cluster. For more information on how to log in to EMR, please see Logging in to Linux Instances. Here, you can choose to log in with WebShell. Click "Log in" on the right of the desired CVM instance to enter the login page. The default username is `root`, and the password is the one you set when creating the EMR cluster. Once the correct credentials are entered, you can enter the command line interface.

Run the following command in EMR command-line interface to switch to the Hadoop user and go to the Presto folder:

```
[root@172 ~]# su hadoop
[hadoop@172 ~]# cd /usr/local/service/presto
```

View the value of uri in the `etc/config.properties` configuration file:

```
[hadoop@172 presto]$ vim etc/config.properties
http-server.http.port=$port
discovery.uri=http://$host:$port
```

Here, $host is your host address, and $port is your port number. Then, switch to the presto-client folder and connect to Hive through Presto:

```
[hadoop@172 presto]# cd /usr/local/service/presto-client
[hadoop@172 presto-client]$ ./presto --server $host:$port --catalog hive --
schema default
```

Here, --catalog indicates the type of database to be manipulated, and --schema the database name (the default database "default" is entered here). For more information on the parameters, run the `presto -h` command or see the official documentation.

After successful execution, you can enter the Presto interface and go directly to the specified database. You can use Hive-SQL to view the table in the Hive database:

```
presto:default> show tables;
     Table
---------------
hive_from_cos
test
```

©2013-2024 Tencent Cloud. All rights reserved.

```
(2 rows)

Query 20180702_140619_00006_c4qzg, FINISHED, 2 nodes
Splits: 2 total, 2 done (100.00%)
0:00 [3 rows, 86B] [17 rows/s, 508B/s]
```

The table hive_from_cos is the one created in the Hive Development Guide.

For more information on Presto usage, please see the official documentation.

©2013-2024 Tencent Cloud. All rights reserved.

Analyzing Data in COS

Last updated：2021-07-08 10:43:44

This document describes more options on using COS-Hive connectors, where the data is from direct-insert, COS, and LZO.

1. Development Preparations

This task requires access to COS, so you need to create a bucket in COS first.
Create an EMR cluster. When creating the EMR cluster, you need to select the Presto component on the software configuration page and enable access to COS on the basic configuration page.
Relevant software programs such as Presto are installed in the `/usr/local/service/` directory of the CVM instance for the EMR cluster.

2. Data Preparations

First, log in to any node (preferably a master one) in the EMR cluster. For information about how to log in to EMR, see Logging in to Linux Instance Using Standard Login Method. Here, you can use WebShell to log in. Click **Login** on the right of the desired CVM instance to go to the login page. The default username is `root` , and the password is the one you set when creating the EMR cluster. Once your credentials are validated, you can enter the command line interface.
Run the following commands to switch to the Hadoop user and go to the Hive installation folder:

```
[root@172 ~]# su hadoop
[hadoop@172 ~]# cd /usr/local/service/hive
```

Create a file named `cos.txt` and add the following data to it:

```
5,cos_patrick
6,cos_stone
```

Upload the file to COS by running the following HDFS command. Here, `$bucketname` is the name plus path of the bucket you created.

```
[hadoop@172 hive]# hdfs dfs -put cosn://$bucketname/
```

Create a file named `lzo.txt` and add the following data to it:

```
10,lzo_pop
```

©2013-2024 Tencent Cloud. All rights reserved.

```
11,lzo_tim
```

Compress it into an .lzo file:

```
[hadoop@172 hive]$ lzop -v lzo.txt
compressing hive_test.data into lzo.txt.lzo
```

Note:

To create an .lzo compressed file, you need to install lzo and lzop first by running this command: `yum -y install lzo lzop` .

3. Creating a Hive Table and Using Presto for Query

A script file is used here to create a Hive database and table. Create a script file named presto_on_cos_test.sql and add the following program to it:

```
create database if not exists test;
use test;
create external table if not exists presto_on_cos (id int,name string) ROW
FORMAT DELIMITED FIELDS TERMINATED BY ',';
insert into presto_on_cos values (12,'hello'),(13,'world');
load data inpath "cosn://$bucketname/cos.txt" into table presto_on_cos;
load data local inpath "/$yourpath/lzo.txt.lzo" into table presto_on_cos;
```

Here, $bucketname is the name plus path of your COS bucket, and $yourpath is the path where the lzo.txt.lzo file is stored.

The script file creates a database named "test" first and then a table named "presto_on_cos" in the created database. Here, the data is loaded in three steps: first, insert the data directly; then, insert the data stored in COS; and finally, insert the data stored in the .lzo package.

It is recommended to use an external table for Hive testing as shown in this example so as to avoid deleting important data by mistake. Run this script with Hive CLI:

```
[hadoop@172 hive]$ hive -f "presto_on_cos_test.sql"
```

Once the execution is completed, you can enter Presto to view the data in the table. Use the same method as described in the previous section to enter Presto, but you should modify the `schema` parameter.

```
[hadoop@172 presto-client]$ ./presto --server $host:$port --catalog hive --
schema test
```

Query the Hive table you just created:

```
presto:test> select * from presto_on_cos ;
```

©2013-2024 Tencent Cloud. All rights reserved.

```
 id |    name
----+-------------
  5 | cos_patrick
  6 | cos_stone
 10 | lzo_pop
 11 | lzo_tim
 12 | hello
 13 | world
(6 rows)

Query 20180702_150000_00011_c4qzg, FINISHED, 3 nodes
Splits: 4 total, 4 done (100.00%)
0:03 [6 rows, 127B] [1 rows/s, 37B/s]
```

For more information about Presto usage, see the official documentation.

©2013-2024 Tencent Cloud. All rights reserved.

Sqoop Development Guide
Import/Export of Relational Database and HDFS

Last updated：2020-11-23 17:13:14

Sqoop is an open-source tool for transferring data between Hadoop and traditional databases such as MySQL and PostgreSQL. It can import data in a relational database (e.g., MySQL, Oracle, and Postgres) into Hadoop's HDFS, and vice versa. One of the highlights of Sqoop is its ability to import data in a relational database into HDFS through Hadoop MapReduce.
This document describes how to use Sqoop on EMR to import and export data between MySQL and HDFS.

1. Prerequisites

You have signed up for a Tencent Cloud account and created an EMR cluster. When creating the EMR cluster, select the Sqoop component on the software configuration page.
Relevant software programs such as Sqoop are installed in the `/usr/local/service/` directory of the CVM instance for the EMR cluster.

2. Creating a MySQL Table

Connect to the created MySQL database first. Enter the EMR Console and copy the instance ID of the destination cluster, i.e., the cluster name. Then, enter the TencentDB for MySQL Console, use Ctrl+F to find the MySQL database corresponding to the cluster, and view the private IP address of the database ($mysqlIP).
Log in to any node (preferably a master one) in the EMR cluster. For more information on how to log in to EMR, please see Logging in to Linux Instances. Here, you can choose to log in with WebShell. Click "Log in" on the right of the desired CVM instance to enter the login page. The default username is `root`, and the password is the one you set when creating the EMR cluster. Once the correct credentials are entered, you can enter the command line interface.
Run the following command in EMR command-line interface to switch to the Hadoop user and go to the Sqoop folder:

```
[root@172 ~]# su hadoop
[hadoop@172 ~]# cd /usr/local/service/sqoop
```

Connect to the MySQL database:

```
[hadoop@172 sqoop]$ mysql -h $mysqlIP -p
Enter password:
```

©2013-2024 Tencent Cloud. All rights reserved.

The password is the one you set when you created the EMR cluster.

After connecting to the MySQL database, enter the test database and create a table. You can also choose the destination database:

```
mysql> use test;
Database changed

mysql> create table sqoop_test(id int not null primary key auto_increment,
title varchar(64), time timestamp, content varchar(255));
Query ok , 0 rows affected(0.00 sec)
```

This command creates a MySQL table with a primary key of ID and three columns of title, time, and content. Insert data entries into the table as follows:

```
mysql> insert into sqoop_test values(null, 'first', now(), 'hdfs');
Query ok, 1 row affected(0.00 sec)

mysql> insert into sqoop_test values(null, 'second', now(), 'mr');
Query ok, 1 row affected (0.00 sec)

mysql> insert into sqoop_test values(null, 'third', now(), 'yarn');
Query ok, 1 row affected(0.00 sec)
```

Run the following command to view the data in the table:

```
Mysql> select * from sqoop_test;
+----+--------+---------------------+---------+
| id | title  | time                | content |
+----+--------+---------------------+---------+
|  1 | first  | 2018-07-03 15:29:37 | hdfs    |
|  2 | second | 2018-07-03 15:30:57 | mr      |
|  3 | third  | 2018-07-03 15:31:07 | yarn    |
+----+--------+---------------------+---------+
3 rows in set (0.00 sec)
```

Exit the MySQL database:

```
Mysql> exit;
```

3. Importing MySQL Data into HDFS

Run the sqoop-import command to import the data from the sqoop_test table created in the previous step into HDFS:

©2013-2024 Tencent Cloud. All rights reserved.

```
[hadoop@172 sqoop]$ bin/sqoop-import --connect jdbc:mysql://$mysqlIP/test --
username root
-P --table sqoop_test --target-dir /sqoop
```

Here, --connect is used to connect to the MySQL database, test can be replaced with your database name, -P indicates that a password is required, --table is the name of the database you want to export, --target-dir is the path in HDFS to import into. **Make sure that the `/sqoop` folder does not exist before the command is run; otherwise, a failure will occur.**

After you press Enter, you will be asked for a password, which is the one set when you created the EMR cluster.

After successful execution, you can view the imported data in the corresponding path in HDFS:

```
[hadoop@172 sqoop]$ hadoop fs -cat /sqoop/*
1, first, 2018-07-03 15:29:37.0,hdfs
2, second, 2018-07-03 15:30:57.0,mr
3, third, 2018-07-03 15:31:07.0,yarn
```

4. Importing HDFS Data into MySQL

You need to create a table in MySQL first to store the data from HDFS:

```
[hadoop@172 sqoop]$ mysql -h $mysqlIP -p
Enter password:
mysql> use test;
Database changed

mysql> create table sqoop_test_back(id int not null primary key auto_increment,
title varchar(64), time timestamp, content varchar(255));
Query ok , 0 rows affected(0.00 sec)
```

Check whether the table is created successfully and then exit MySQL:

```
mysql> show tables;
+-----------------+
| Tables_in_test  |
+-----------------+
| sqoop_test      |
| sqoop_test_back |
+-----------------+
2 rows in set (0.00 sec)

mysql> exit;
```

Run the sqoop-export command to export the data imported into HDFS in the previous step into MySQL:

©2013-2024 Tencent Cloud. All rights reserved.

```
[hadoop@172 sqoop]$ bin/sqoop-export --connect jdbc:mysql://$mysqlIP/test --
username
root -P --table sqoop_test_back --export-dir /sqoop
```

The parameters are similar to those for sqoop-import, except --export-dir, which is the path of the data in HDFS. You also need to enter the password after pressing Enter.

After successful execution, you can verify the data in the database sqoop_test_back:

```
[hadoop@172 sqoop]$ mysql -h $mysqlIP -p
Enter password:
mysql> use test;
Database changed

mysql> select * from sqoop_test_back;
+----+---------+---------------------+---------+
| id | title   | time                | content |
+----+---------+---------------------+---------+
|  1 | first   | 2018-07-03 15:29:37 | hdfs    |
|  2 | second  | 2018-07-03 15:30:57 | mr      |
|  3 | third   | 2018-07-03 15:31:07 | yarn    |
+----+---------+---------------------+---------+
3 rows in set (0.00 sec)
```

For more information about Sqoop usage, please see the official documentation.

©2013-2024 Tencent Cloud. All rights reserved.

Incremental Data Import into HDFS

Last updated：2025-02-12 16:52:06

Sqoop is an open-source tool for transferring data between Hadoop and traditional databases such as MySQL and PostgreSQL. It can import data in a relational database (e.g., MySQL, Oracle, and Postgres) into Hadoop's HDFS, and vice versa. One of the highlights of Sqoop is its ability to import data in a relational database into HDFS through Hadoop MapReduce.
This document describes the incremental import operation of Sqoop, i.e., importing only new or updated data in the database into HDFS. This can be done in either append or lastmodified mode. The former can be used in the case where there is only new but not modified data in the database, while the latter can be used in either case.

1. Preparations for Development

Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, select the Sqoop component on the software configuration page.
Relevant software programs such as Sqoop are installed in the `/usr/local/service/` directory of the CVM instance for the EMR cluster.

2. Using the append Mode

This section will continue to use the use case from the previous section.
Enter the EMR Console and copy the instance ID of the target cluster, i.e., the cluster name. Then, enter the TencentDB for MySQL Console, use Ctrl+F to find the MySQL database corresponding to the cluster, and view the private IP address of the database ($mysqlIP).
Log in to any node (preferably a master one) in the EMR cluster. For more information on how to log in to EMR, please see Logging in to Linux Instances. Here, you can choose to log in with WebShell. Click "Log in" on the right of the desired CVM instance to enter the login page. The default username is `root`, and the password is the one you set when creating the EMR cluster. Once the correct credentials are entered, you can enter the command line interface.
Run the following command in EMR command-line interface to switch to the Hadoop user and go to the Sqoop folder:

```
[root@172 ~]# su hadoop
[hadoop@172 ~]# cd /usr/local/service/sqoop
```

Connect to the MySQL database:

```
[hadoop@172 sqoop]$ mysql -h $mysqlIP -p
Enter password:
```

©2013-2024 Tencent Cloud. All rights reserved.

The password is the one you set when you created the EMR cluster.

After connecting to the MySQL database, add a new data entry to the table sqoop_test, as shown below:

```
mysql> use test;
Database changed

mysql> insert into sqoop_test values(null, 'forth', now(), 'hbase');
Query ok, 1 row affected(0.00 sec)
```

View the data in the table:

```
Mysql> select * from sqoop_test;
+----+--------+---------------------+---------+
| id | title  | time                | content |
+----+--------+---------------------+---------+
|  1 | first  | 2018-07-03 15:29:37 | hdfs    |
|  2 | second | 2018-07-03 15:30:57 | mr      |
|  3 | third  | 2018-07-03 15:31:07 | yarn    |
|  4 | forth  | 2018-07-03 15:39:38  | hbase    |
+----+--------+---------------------+---------+
4 rows in set (0.00 sec)
```

Sync the new data entry in append mode to the HDFS path where the data in the previous section is stored:

```
[hadoop@172 sqoop]$ bin/sqoop-import --connect jdbc:mysql://$mysqlIP/test --
username
root -P --table sqoop_test --check-column id  --incremental append --last-value
3 --target-dir
/sqoop
```

Here, $mysqlIP is the private IP address of your MySQL database.

To run the command, you need to enter the password for the database, which is defaulted to the password you set when you created the EMR cluster. There are more parameters than the normal sqoop-import command, where --check-column is the reference data during import, --incremental is the import mode (i.e., append in this example), and --last-value is the value of the reference data. All data entries that are newer than this value are imported into HDFS.

After successful execution, you can view the updated data in the corresponding directory of HDFS:

```
[hadoop@172 sqoop]$ hadoop fs -cat /sqoop/*
1, first, 2018-07-03 15:29:37.0,hdfs
2, second, 2018-07-03 15:30:57.0,mr
3, third, 2018-07-03 15:31:07.0,yarn
4,forth,2018-07-03 15:39:38.0,hbase
```

Using a Sqoop job

©2013-2024 Tencent Cloud. All rights reserved.

To sync data to HDFS in append mode, you need to manually enter --last-value each time, but you can also use the Sqoop job method, where Sqoop automatically saves the value of last-value in the last successful import. To use it, you need to start the sqoop-metastore process in the following steps:

Start the sqoop-metastore process in conf/sqoop-site.xml first:

```
<property>
  <name>sqoop.metastore.client.enable.autoconnect</name>
  <value>true</value>
</property>
```

Then, start the sqoop-metastore service in the bin directory:

```
./sqoop-metastore &
```

Create a Sqoop job by running the following command:

Note

This command is applicable to Sqoop 1.4.6.

```
[hadoop@172 sqoop]$ bin/sqoop job --create job1 -- import --connect
jdbc:mysql://$mysqlIP/test --username root -P --table sqoop_test --check-column
id
--incremental append --last-value 4 --target-dir /sqoop
```

Here, $mysqlIP is the private IP address of your MySQL database. With this command, a Sqoop job is successfully created, and each execution will automatically use the last modified value of last-value.

Add a new data entry to the table sqoop_test in the MySQL database:

```
mysql> insert into sqoop_test values(null, 'fifth', now(), 'hive');
Query ok, 1 row affected(0.00 sec)

Mysql> select * from sqoop_test;
+----+--------+---------------------+---------+
| id | title  | time                | content |
+----+--------+---------------------+---------+
|  1 | first  | 2018-07-03 15:29:37 | hdfs    |
|  2 | second | 2018-07-03 15:30:57 | mr      |
|  3 | third  | 2018-07-03 15:31:07 | yarn    |
|  4 | forth  | 2018-07-03 15:39:38  | hbase    |
|  5 | fifth  | 2018-07-03 16:02:29   | hive    |
+----+--------+---------------------+---------+
5 rows in set (0.00 sec)
```

Then, execute the Sqoop job:

```
[hadoop@172 sqoop]$ bin/sqoop job --exec job1
```

©2013-2024 Tencent Cloud. All rights reserved.

To run this command, you need to enter the password for your MySQL database. After successful execution, you can view the updated data in the corresponding directory of HDFS:

```
[hadoop@172 sqoop]$ hadoop fs -cat /sqoop/*
1, first, 2018-07-03 15:29:37.0,hdfs
2, second, 2018-07-03 15:30:57.0,mr
3, third, 2018-07-03 15:31:07.0,yarn
4,forth,2018-07-03 15:39:38.0,hbase
5,fifth,2018-07-03 16:02:29.0,hive
```

3. Using the lastmodified Mode

Create a Sqoop job in lastmodified mode of sqoop-import directly to first query the last modified time in sqoop_test:

```
mysql> select max(time) from sqoop_test;
```

Create a Sqoop job:

```
[hadoop@172 sqoop]$ bin/sqoop job --create job2 -- import --connect
jdbc:mysql://$mysqlIP/test --username root -P --table sqoop_test --check-column
time --incremental lastmodified --merge-key id --last-value '2018-07-03
16:02:29' --target-dir /sqoop
```

Parameter description

`$mysqlIP` : refers to the private IP address of your MySQL database

`--check-column` : must use a timestamp

`--incremental` : selects the lastmodified mode

`--merge-key` : selects the ID

`--last-value` : refers to the last modified time in the table that is queried. All modifications made after this time will be synced to HDFS, and the Sqoop job will automatically save and update the value each time.

Add data to the table sqoop_test in the MySQL database and make changes:

```
mysql> insert into sqoop_test values(null, 'sixth', now(), 'sqoop');
Query ok, 1 row affected(0.00 sec)

mysql> update sqoop_test set time=now(), content='spark' where id = 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 changed: 1 warnings: 0

Mysql> select * from sqoop_test;
+----+--------+---------------------+---------+
| id | title  | time                | content |
+----+--------+---------------------+---------+
```

©2013-2024 Tencent Cloud. All rights reserved.

```
|  1 | first  | 2018-07-03 16:07:46 | spark    |
|  2 | second | 2018-07-03 15:30:57 | mr       |
|  3 | third  | 2018-07-03 15:31:07 | yarn     |
|  4 | forth  | 2018-07-03 15:39:38  | hbase    |
|  5 | fifth  | 2018-07-03 16:02:29   | hive    |
|  6 | fifth  | 2018-07-03 16:09:58   | sqoop    |
+----+--------+---------------------+---------+
6 rows in set (0.00 sec)
```

Execute the Sqoop job:

```
[hadoop@172 sqoop]$ bin/sqoop job --exec job2
```

To run this command, you need to enter the password for your MySQL database. After successful execution, you can view the updated data in the corresponding directory of HDFS:

```
[hadoop@172 sqoop]$ hdfs dfs -cat /sqoop/*
1,first,2018-07-03 16:07:46.0,spark
2,second,2018-07-03 15:30:57.0,mr
3,third,2018-07-03 15:31:07.0,yarn
4,forth,2018-07-03 15:39:38.0,hbase
5,fifth,2018-07-03 16:02:29.0,hive
6,sixth,2018-07-03 16:09:58.0,sqoop
```

For more information on Sqoop usage, please see the official documentation.

©2013-2024 Tencent Cloud. All rights reserved.

Importing and Exporting Data Between Hive and TencentDB for MySQL

Last updated：2025-02-12 16:52:07

This document describes how to use Sqoop on EMR to import and export data between MySQL and Hive.

1. Prerequisites

Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, you need to select the Sqoop and Hive components on the software configuration page.
Relevant software programs such as Sqoop are installed in the `/usr/local/service/` directory of the CVM instance for the EMR cluster.

2. Importing Data from TencentDB for MySQL into Hive

This section will continue to use the use case from the previous section.
Enter the Elastic MapReduce Console and copy the instance ID of the target cluster, i.e., the cluster name. Then, enter the TencentDB for MySQL Console, use Ctrl+F to find the MySQL database corresponding to the cluster, and view the private IP address of the database ($mysqlIP).
Log in to any node (preferably a master one) in the EMR cluster. For more information on how to log in to EMR, please see Logging in to Linux Instances. Here, you can choose to log in with WebShell. Click "Log in" on the right of the desired CVM instance to enter the login page. The default username is `root`, and the password is the one you set when creating the EMR cluster. Once the correct credentials are entered, you can enter the command line interface.
Run the following command on the EMR command line to switch to the Hadoop user and go to the Hive folder:

```
[root@172 ~]# su hadoop
[hadoop@172 ~]# cd /usr/local/service/hive
```

Create a Hive database:

```
[hadoop@172 hive]$ hive
hive> create database hive_from_sqoop;
OK
Time taken: 0.167 seconds
```

Import the MySQL database created in the previous section into Hive by running the `sqoop-import` command:

```
[hadoop@172 hive]# cd /usr/local/service/sqoop
```

©2013-2024 Tencent Cloud. All rights reserved.

```
[hadoop@172 sqoop]$ bin/sqoop-import --connect  jdbc:mysql://$mysqlIP/test --
username
root -P --table sqoop_test_back --hive-database hive_from_sqoop --hive-import -
-hive-table hive_from_sqoop
```

$mysqlIP: private IP of the TencentDB instance.

test: MySQL database name.

--table: name of the MySQL table to be exported.

--hive-database: Hive database name.

--hive-table: name of the Hive table to be imported.

Running this command requires the password of your MySQL database, which is the one you set when you created the EMR cluster. After successful execution, you can view the imported database in Hive:

```
hive> select * from hive_from_sqoop;
OK
1       first   2018-07-03 16:07:46.0   spark
2       second  2018-07-03 15:30:57.0   mr
3       third   2018-07-03 15:31:07.0   yarn
4       forth   2018-07-03 15:39:38.0   hbase
5       fifth   2018-07-03 16:02:29.0   hive
6       sixth   2018-07-03 16:09:58.0   sqoop
Time taken: 1.245 seconds, Fetched: 6 row(s)
```

3. Importing Data from Hive into TencentDB for MySQL

Sqoop supports importing data from Hive tables into TencentDB for MySQL. To do so, create a table in Hive first and then import data.

Log in to any node (preferably a master one) in the EMR cluster. Run the following command on the EMR command line to switch to the Hadoop user and go to the Hive folder:

```
[root@172 ~]# su hadoop
[hadoop@172 ~]# cd /usr/local/service/hive
```

Create a bash script file named gen_data.sh and add the following code to it:

```
#!/bin/bash
MAXROW=1000000 # Specify the number of data rows to be generated
for((i = 0; i < $MAXROW; i++))
do
     echo $RANDOM, \\"$RANDOM\\"
done
```

Run it as follows:

©2013-2024 Tencent Cloud. All rights reserved.

```
[hadoop@172 hive]$ ./gen_data.sh > hive_test.data
```

This script file will generate 1,000,000 random number pairs and save them to the `hive_test.data` file.

Run the following command to upload the generated test data to HDFS first:

```
[hadoop@172 hive]$ hdfs dfs -put ./hive_test.data /$hdfspath
```

Here, `$hdfspath` is the path to your file on HDFS.

Connect to Hive and create a test table:

```
[hadoop@172 hive]$ bin/hive
hive> create database hive_to_sqoop;          # Create the database
`hive_to_sqoop`
OK
Time taken: 0.176 seconds
hive> use hive_to_sqoop;                              # Switch databases
OK
Time taken: 0.176 seconds
hive> create table hive_test (a int, b string)
hive> ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
                          # Create a data table named `hive_test` and specify
the column separator as `,`
OK
Time taken: 0.204 seconds
hive> load data inpath "/$hdfspath/hive_test.data" into table hive_test;   #
Import the data
```

`$hdfspath` is the path to your file stored in HDFS.

After success, you can run the `quit` command to exit the Hive data warehouse. Then, connect to TencentDB for MySQL and create a corresponding table:

```
[hadoop@172 hive]$ mysql -h $mysqlIP -p
Enter password:
```

Here, `$mysqlIP` is the private IP address of this database, and the password is the one you set when creating the cluster.

Create a table named `test` in MySQL. **Note that the field names in the MySQL table must be the same as those in the Hive table:**

```
mysql> create table table_from_hive (a int,b varchar(255));
```

You can exit MySQL after successfully creating the table.

There are two ways to import data from a Hive data warehouse into a TencentDB for MySQL database by using Sqoop: using the Hive data stored in HDFS directly or using HCatalog to import the data.

©2013-2024 Tencent Cloud. All rights reserved.

Using Hive data stored in HDFS

Switch to the Sqoop folder and export the data from the Hive database to the TencentDB for MySQL database by running the following command:

```
[hadoop@172 hive]$ cd  ../sqoop/bin
[hadoop@172 bin]$ ./sqoop-export --connect jdbc:mysql://$mysqlIP/test --
username root -P
--table table_from_hive --export-dir
/usr/hive/warehouse/hive_to_sqoop.db/hive_test
```

Here, `$mysqlIP` is the private IP address of your TencentDB for MySQL instance, `test` is the name of the MySQL database, `--table` is followed by the name of the table in the MySQL database, and `--export-dir` is followed by the location of the Hive table data stored in HDFS.

Importing data by using HCatalog

Switch to the Sqoop folder and export the data from the Hive database to the TencentDB for MySQL database by running the following command:

```
[hadoop@172 hive]$ cd  ../sqoop/bin
[hadoop@172 bin]$ ./sqoop-export --connect jdbc:mysql://$mysqlIP/test --
username root -P
--table table_from_hive --hcatalog-database hive_to_sqoop --hcatalog-table
hive_test
```

Here, `$mysqlIP` is the private IP address of your TencentDB for MySQL instance, `test` is the name of the MySQL database, `--table` is followed by the name of the table in the MySQL database, `-hcatalog-database` is followed by the name of the database where the Hive table to be exported is stored, and `--hcatalog-table` is followed by the name of the Hive table to be exported.

After the operation is completed, you can enter the MySQL database to see whether the import is successful:

```
[hadoop@172 hive]$ mysql -h $mysqlIP -p                  # Connect to MySQL
Enter password:
mysql> use test;
Database changed
mysql> select count(*) from table_from_hive;     # Now there are 1,000,000 data
entries in the table
+----------+
| count(*) |
+----------+
| 1000000 |
+----------+
1 row in set (0.03 sec)
mysql> select * from table_from_hive limit 10;    # View the first 10 entries
in the table
```

©2013-2024 Tencent Cloud. All rights reserved.

```
+-------+----------+
| a     | b        |
+-------+----------+
| 28523 |  "3394"  |
| 31065 |  "24583" |
|   399 |  "23629" |
| 18779 |  "8377"  |
| 25376 |  "30798" |
| 20234 |  "22048" |
| 30744 |  "32753" |
| 21423 |  "6117"  |
| 26867 |  "16787" |
| 18526 |  "5856"  |
+-------+----------+
10 rows in set (0.00 sec)
```

You can view more parameters about the `sqoop-export` command by running the following command:

```
[hadoop@172 bin]$ ./sqoop-export --help
```

4. Importing a Hive Table in ORC Format into TencentDB for MySQL

ORC is a columnar storage format that can greatly improve the performance of Hive. This section describes how to create an ORC table, load data into it, and then use the Sqoop on EMR to export the data stored in ORC format in Hive to TencentDB for MySQL.

Note:

After importing a Hive table in ORC format to TencentDB for MySQL, you cannot use the data stored in HDFS directly; instead, you have to use HCatalog.

This section will continue to use the use case from the previous section.

Log in to a master node of the EMR cluster and run the following command on the EMR command line to switch to the Hadoop user and go to the Hive folder:

```
[root@172 ~]# su hadoop
[hadoop@172 ~]# cd /usr/local/service/hive
```

Create a table in the database `hive_from_sqoop` created in the previous section:

```
[hadoop@172 hive]$ hive
hive> use hive_to_sqoop;
OK
Time taken: 0.013 seconds
hive> create table if not exists orc_test(a int,b string) ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',' stored as orc;
```

©2013-2024 Tencent Cloud. All rights reserved.

You can view the storage format of the data in the table by running the following command:

```
hive> show create table orc_test;
OK
CREATE TABLE `orc_test`(
  `a` int,
  `b` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
WITH SERDEPROPERTIES (
  'field.delim'=',',
  'serialization.format'=',')
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat'
LOCATION
  'hdfs://HDFS2789/usr/hive/warehouse/hive_to_sqoop.db/orc_test'
TBLPROPERTIES (
  'COLUMN_STATS_ACCURATE'='{\\"BASIC_STATS\\":\\"true\\"}',
  'numFiles'='0',
  'numRows'='0',
  'rawDataSize'='0',
  'totalSize'='0',
  'transient_lastDdlTime'='1533563293')
Time taken: 0.041 seconds, Fetched: 21 row(s)
```

It can be seen from the returned data that the data storage format in the table is ORC.

There are several ways to import data into a Hive table in ORC format, but only one of them is described below, i.e., importing data into an ORC table by creating a temporary Hive table in text storage format. The table `hive_test` created in the previous section is used here as the temporary table, and the following command is used to import the data:

```
hive> insert into table orc_test select * from hive_test;
```

After successful import, you can view the data in the table by running the `select` command.

Then, use Sqoop to export the Hive table in ORC format to MySQL. Connect to TencentDB for MySQL and create a corresponding table. The specific way to connect is as described above.

```
[hadoop@172 hive]$ mysql -h $mysqlIP -p
Enter password:
```

Here, `$mysqlIP` is the private IP address of this database, and the password is the one you set when creating the cluster.

©2013-2024 Tencent Cloud. All rights reserved.

Create a table named `test` in MySQL. **Note that the field names in the MySQL table must be the same as those in the Hive table:**

```
mysql> create table table_from_orc (a int,b varchar(255));
```

You can exit MySQL after successfully creating the table.

Switch to the Sqoop folder and export the data in ORC format from the Hive database to the TencentDB for MySQL database by running the following command:

```
[hadoop@172 hive]$ cd  ../sqoop/bin
[hadoop@172 bin]$ ./sqoop-export --connect jdbc:mysql://$mysqlIP/test --
username root -P
--table table_from_orc --hcatalog-database hive_to_sqoop --hcatalog-table
orc_test
```

Here, `$mysqlIP` is the private IP address of your TencentDB for MySQL instance, `test` is the name of the MySQL database, `--table` is followed by the name of the table in the MySQL database, `-hcatalog-database` is followed by the name of the database where the Hive table to be exported is stored, and `--hcatalog-table` is followed by the name of the Hive table to be exported.

After successful import, you can view the data in the corresponding table in the MySQL database:

```
mysql> select count(*) from table_from_orc;
+----------+
| count(*) |
+----------+
| 1000000 |
+----------+
1 row in set (0.24 sec)
mysql> select * from table_from_orc limit 10;
+-------+----------+
| a     | b        |
+-------+----------+
| 28523 |  "3394"  |
| 31065 |  "24583" |
|   399 |  "23629" |
| 18779 |  "8377"  |
| 25376 |  "30798" |
| 20234 |  "22048" |
| 30744 |  "32753" |
| 21423 |  "6117"  |
| 26867 |  "16787" |
| 18526 |  "5856"  |
+-------+----------+
10 rows in set (0.00 sec)
```

©2013-2024 Tencent Cloud. All rights reserved.

For more information on Sqoop usage, please see the official documentation.

©2013-2024 Tencent Cloud. All rights reserved.

Hue Development Guide

Hue Overview

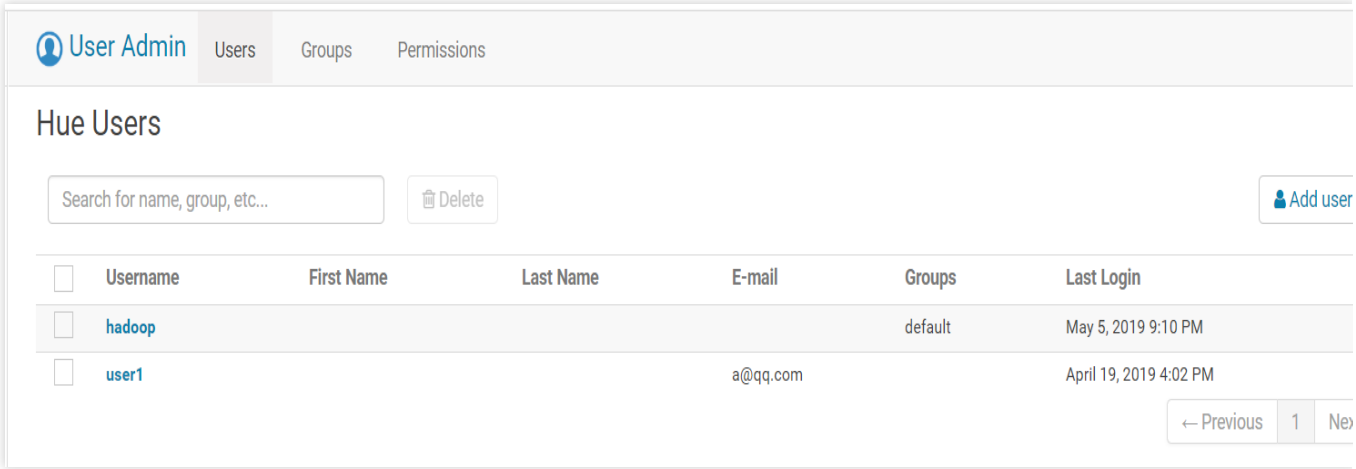
Last updated：2025-02-12 16:52:06

Hue is an open-source Apache Hadoop UI system that evolved from Cloudera Desktop, which Cloudera contributed to the Hadoop project of the Apache Software Foundation. Hue is implemented on the basis of Django, a Python web framework. By using Hue, you can interact with Hadoop clusters in the web-based console on a browser, such as manipulating HDFS data, running MapReduce jobs, executing Hive SQL statements, and browsing HBase databases.

Accessing Hue WebUI

To use the Hue component to manage workflows, log in to the Hue console first:

1. Log in to the EMR console and click the **ID/Name** of the target cluster in the cluster list to go to the cluster details page. Then, click **Cluster Service**.
2. Find the Hue component on the list page and click **WebUI URL** to go to the Hue page.
3. When logging in to the Hue console for the first time, use the hadoop account and the password set when you created the cluster.



Note:

EMR-v2.5.0 and earlier, and EMR-v3.1.0 and earlier are not integrated with OpenLDAP. When you log in to the Hue console for the first time, you must use the root account for login and then create the hadoop account on the WebUI. This is because the default component account upon startup in EMR is hadoop and all subsequent jobs should be submitted by using the hadoop account. For more information about how to create a hadoop account, see the official document of [Hue](https://docs.gethue.com/administrator/administration/ user-management/).

©2013-2024 Tencent Cloud. All rights reserved.

Managing User Permissions

1. Add a user.

1.1 Log in to the EMR console and add a user on the User Management page.

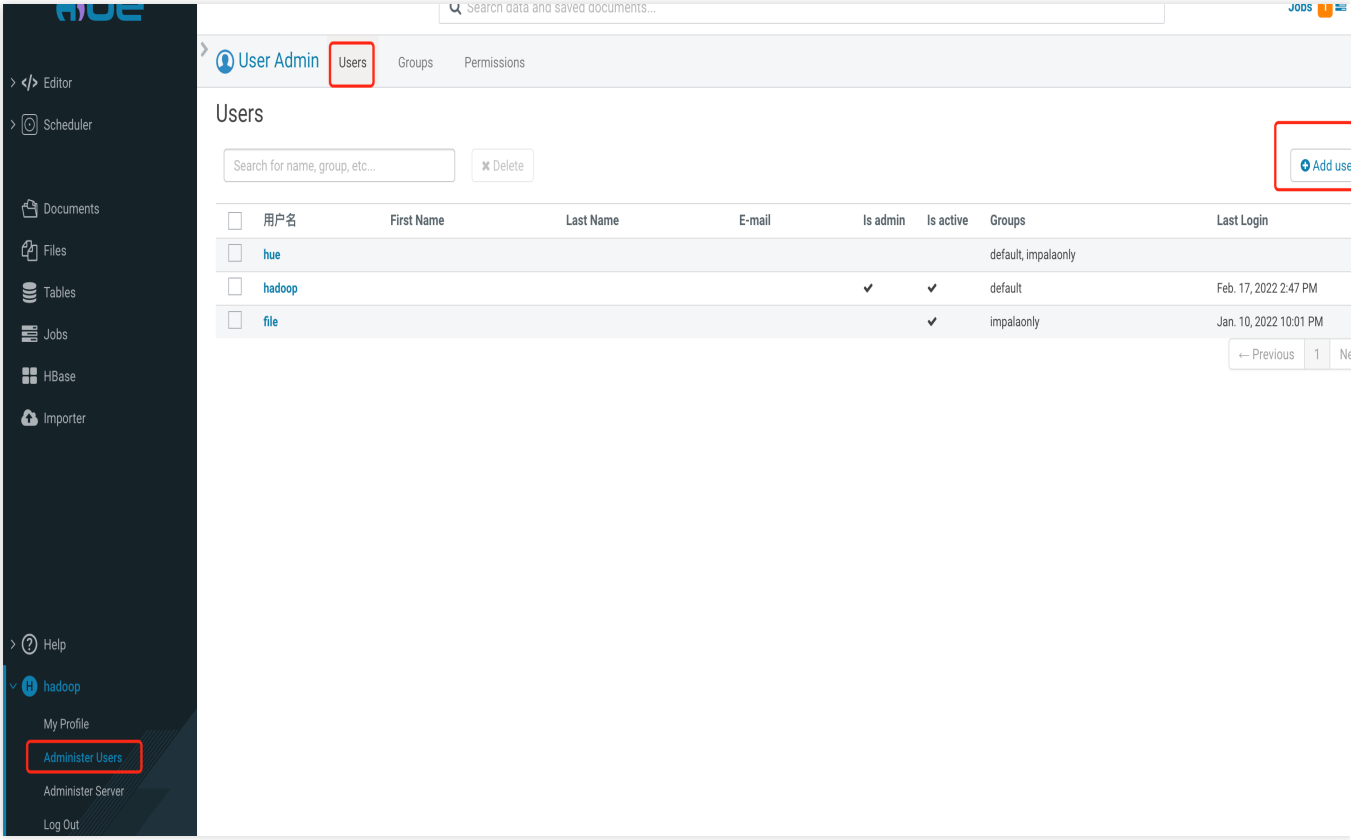
1.2 After adding a user, if you have deployed Ranger in your cluster, you must manually trigger the delivery of the configuration in `ranger-ugsync-site.xml` to restart the EnableUnixAuth service for user synchronization. For more information, see [User Management](https://intl.cloud.tencent.com /document/product/1026/43326). Then, you can go to the Ranger WebUI to set access permissions of the new user.

1.3 Find the Hue component on the list page, click **WebUI URL** to go to the Hue page and log in to Hue.

2. Perform permission control.

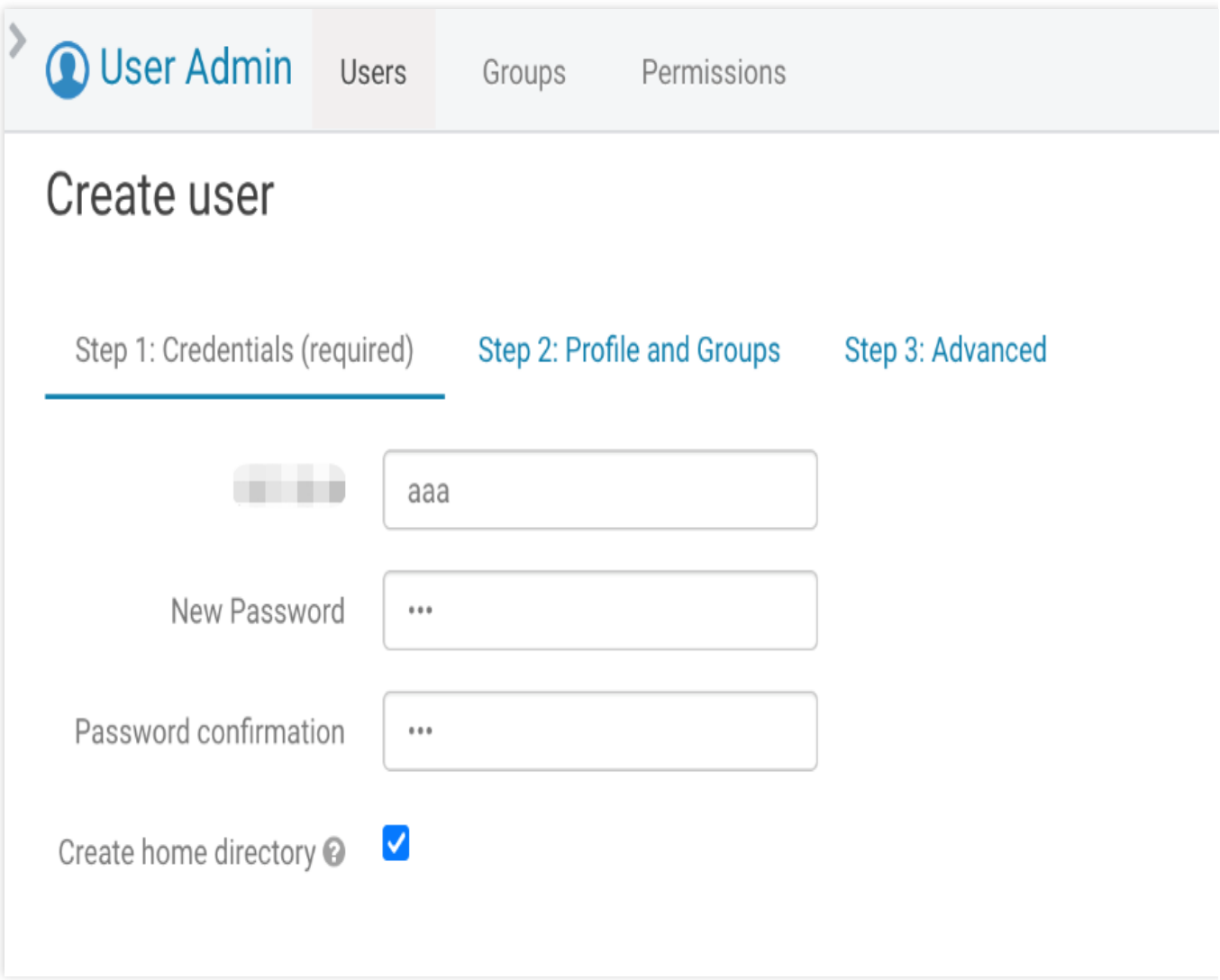
You can assign different permissions to groups through Hue and add users to groups to get specific permissions.

2.1 Click **Groups** at the top of the user management page and then click **Add group** on the right.



2.2 Enter the user group information, select the users to be added to the group, specify the permissions for the group, and click **Add group**.

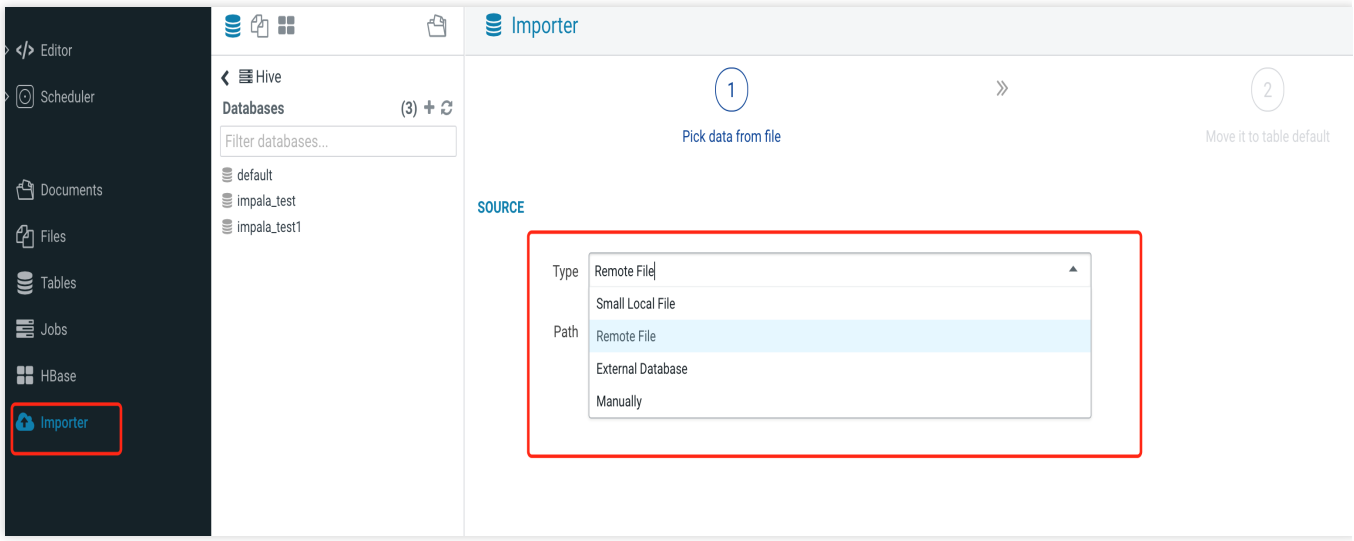
©2013-2024 Tencent Cloud. All rights reserved.



Importing Data

Hue allows you to import data from a local file, HDFS file, external database, or manually.

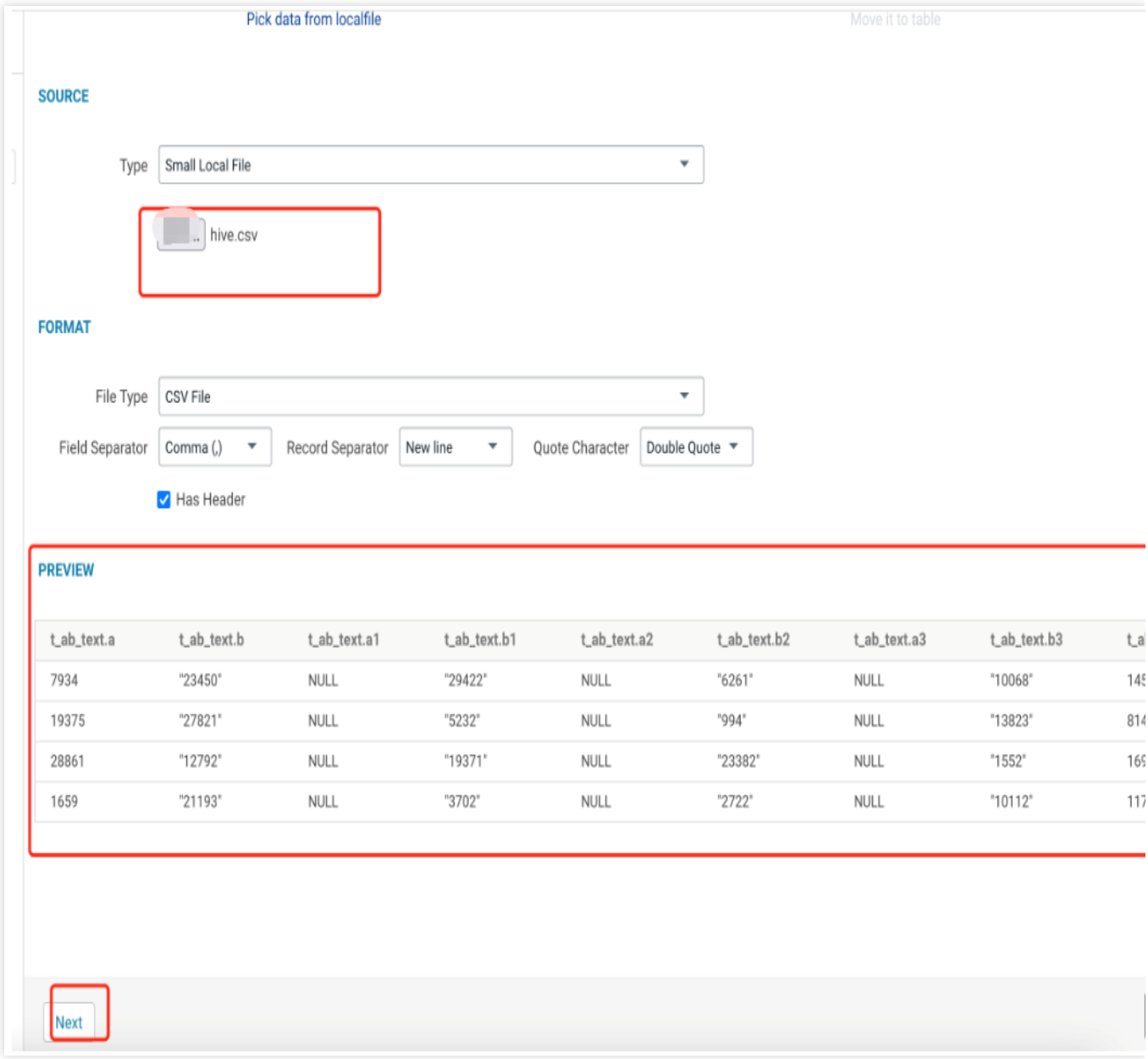
©2013-2024 Tencent Cloud. All rights reserved.



1. Import a local file.

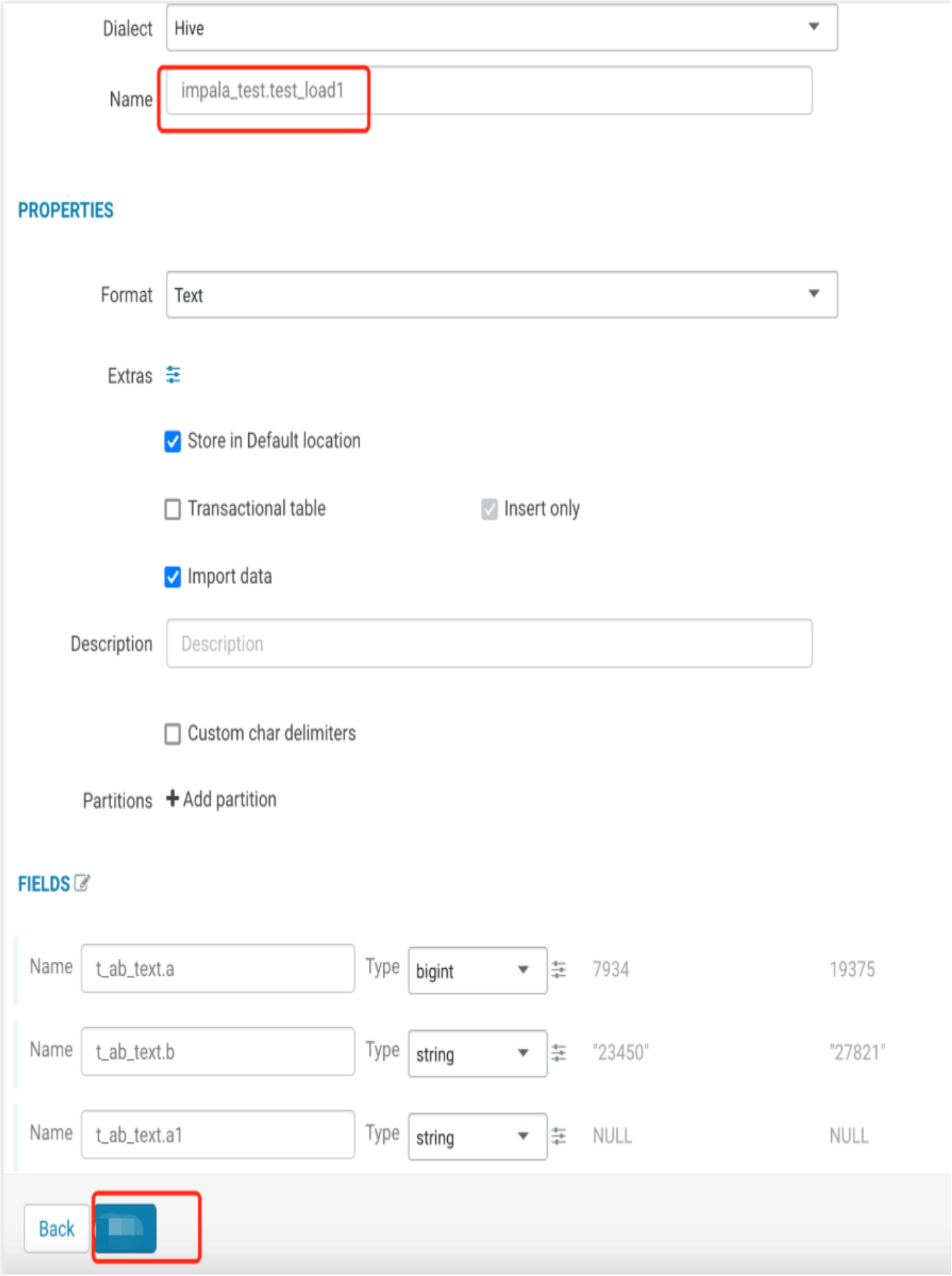
1.1 Click **Choose file** and select a CSV file. Hue will automatically recognize the delimiter and generate a preview. Click **Next** to import the file to a table.

©2013-2024 Tencent Cloud. All rights reserved.



1.2 Enter the table information and click **Save**.

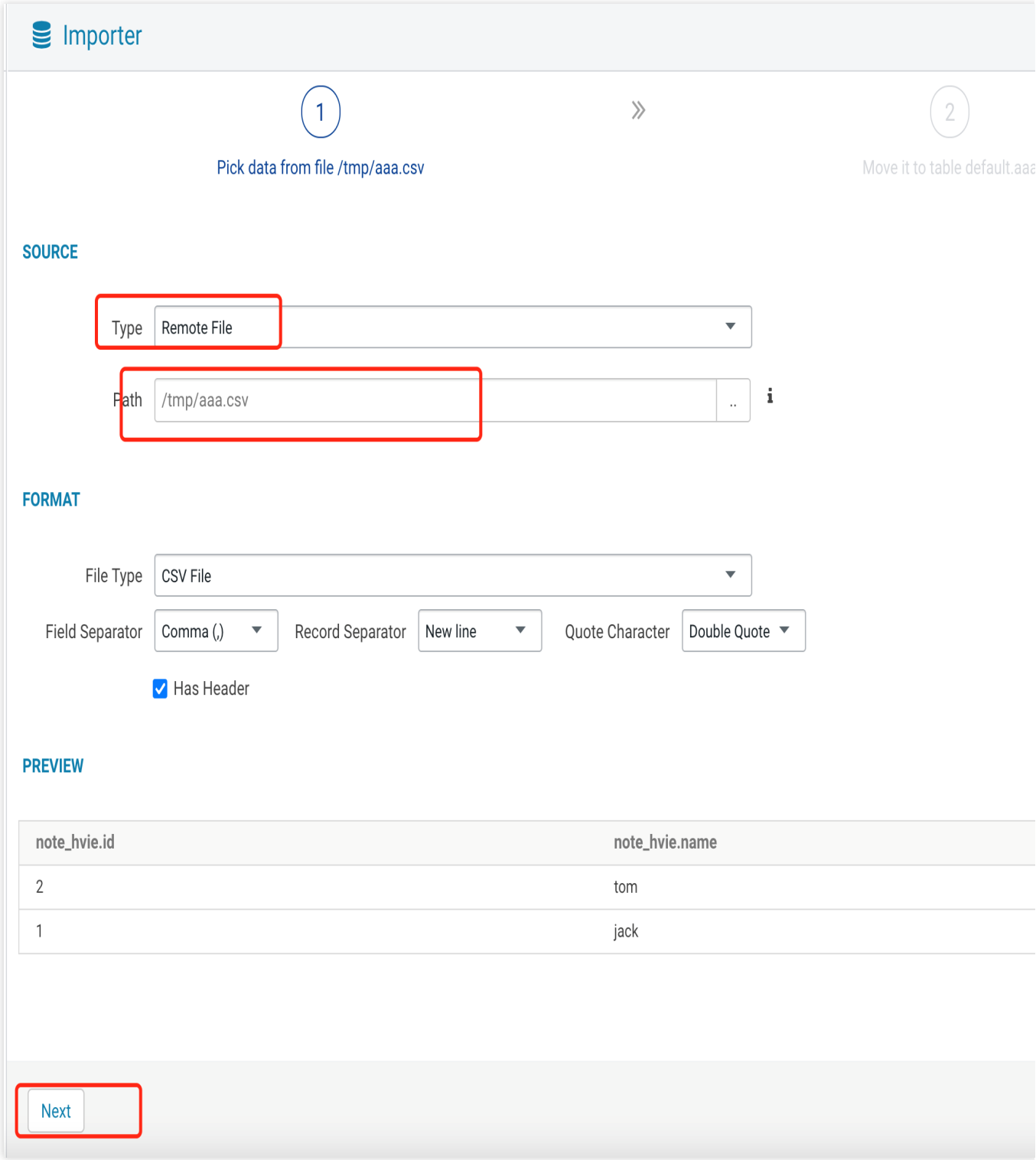
©2013-2024 Tencent Cloud. All rights reserved.



©2013-2024 Tencent Cloud. All rights reserved.

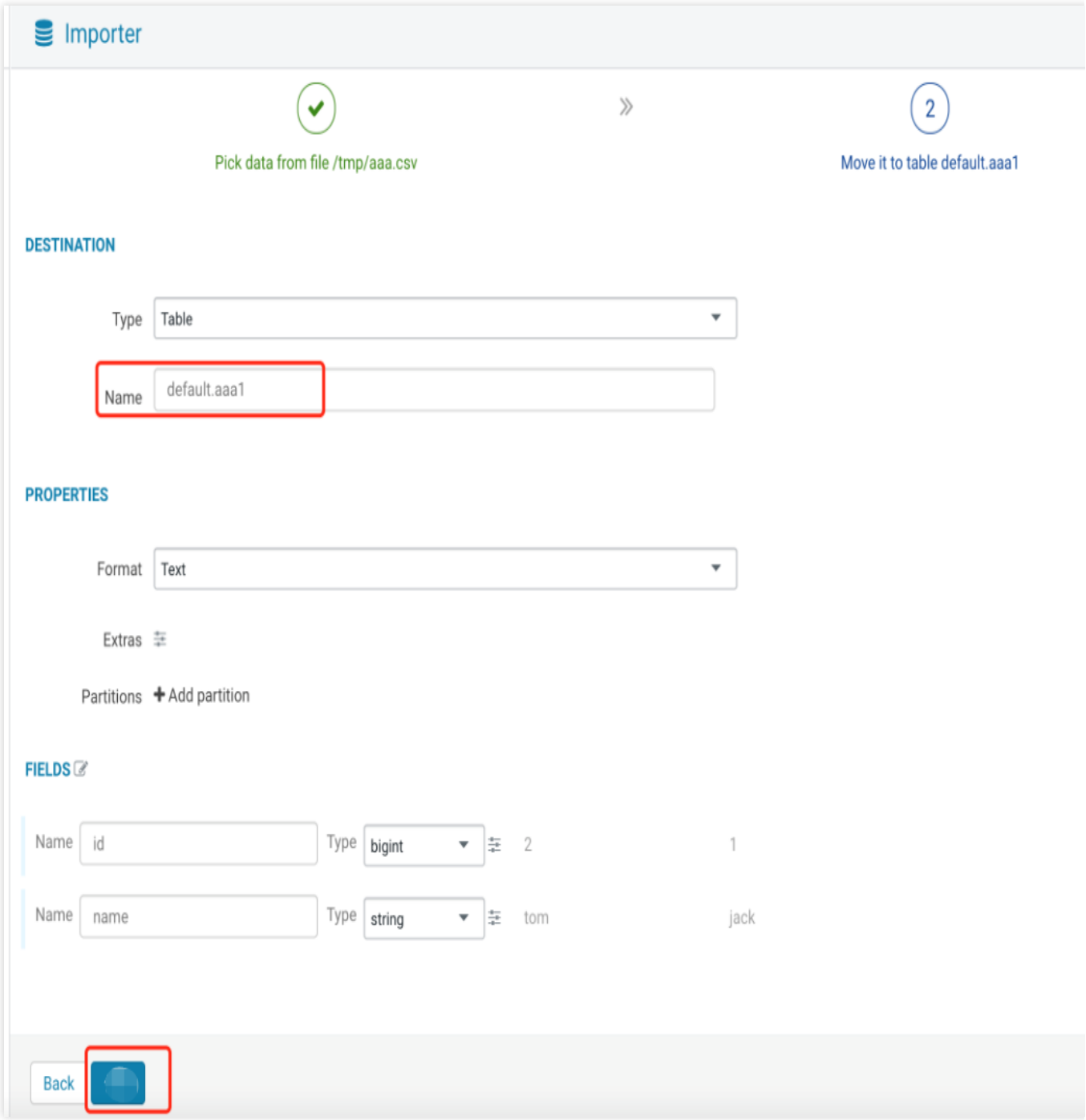
2. Import an HDFS file.

2.1 Select a CSV file from HDFS.



2.2 Enter the table information and click **Save**.

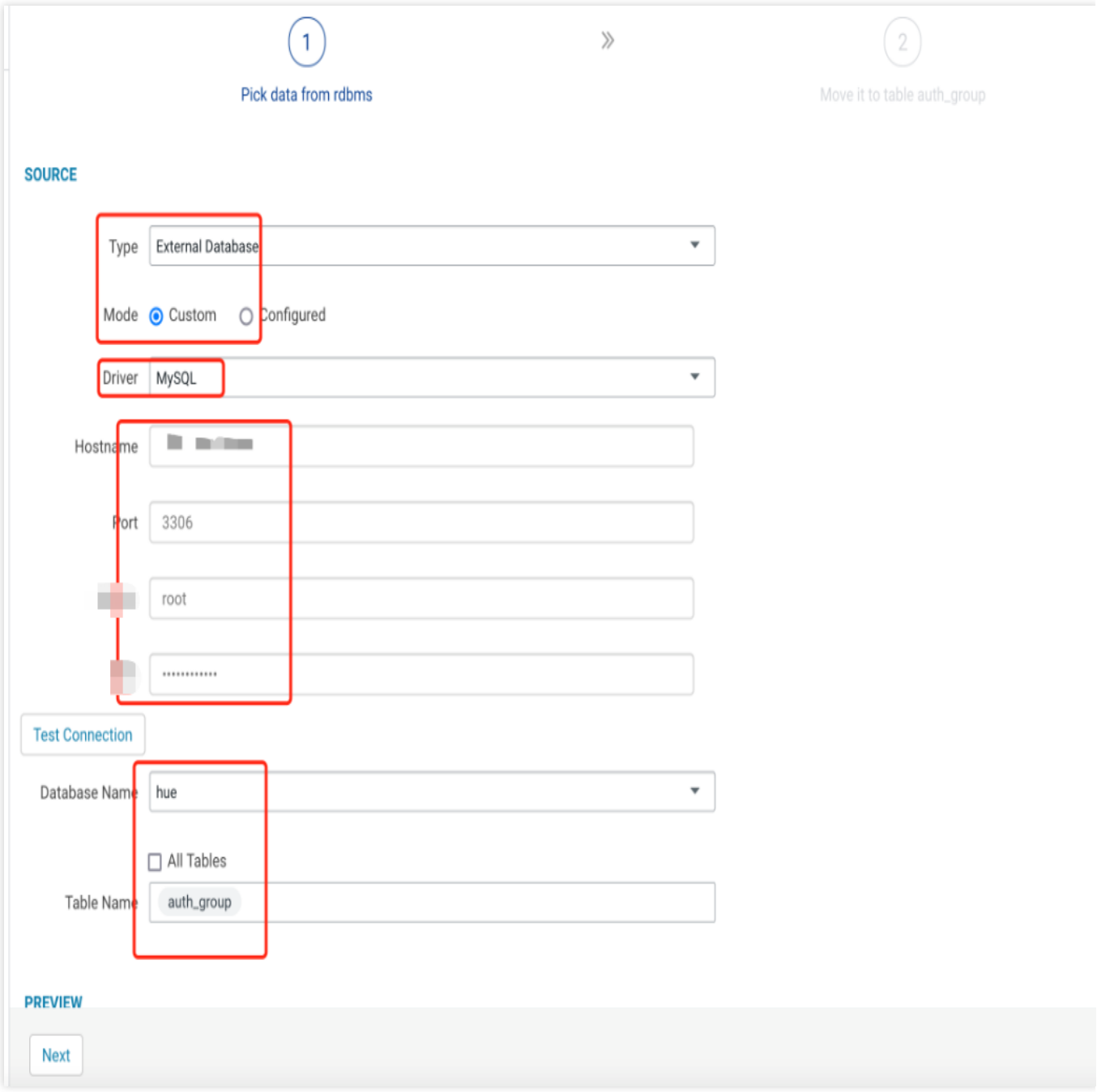
©2013-2024 Tencent Cloud. All rights reserved.



3. Import an external database.

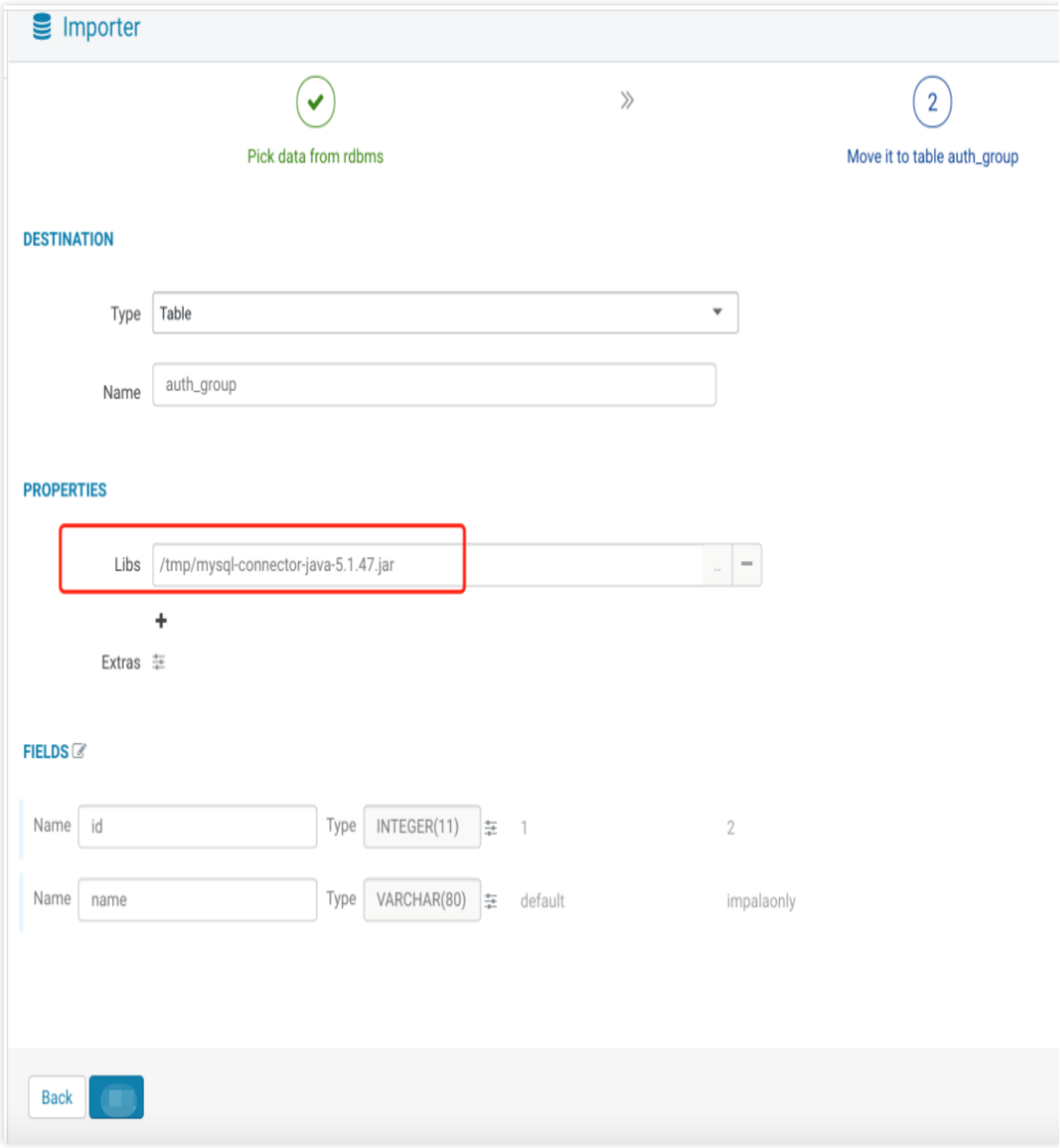
3.1 Enter the external database information, click **Test Connection** to get the database information, select the database and table, and click **Next**.

©2013-2024 Tencent Cloud. All rights reserved.



3.2 Enter the information of the target table, click **Libs**, select the MySQL driver, and click **Save**.

©2013-2024 Tencent Cloud. All rights reserved.



Job Management

Click the **Jobs** tab on the right to enter the job management page. Then, click a job type tab at the top to view and manage jobs.

©2013-2024 Tencent Cloud. All rights reserved.

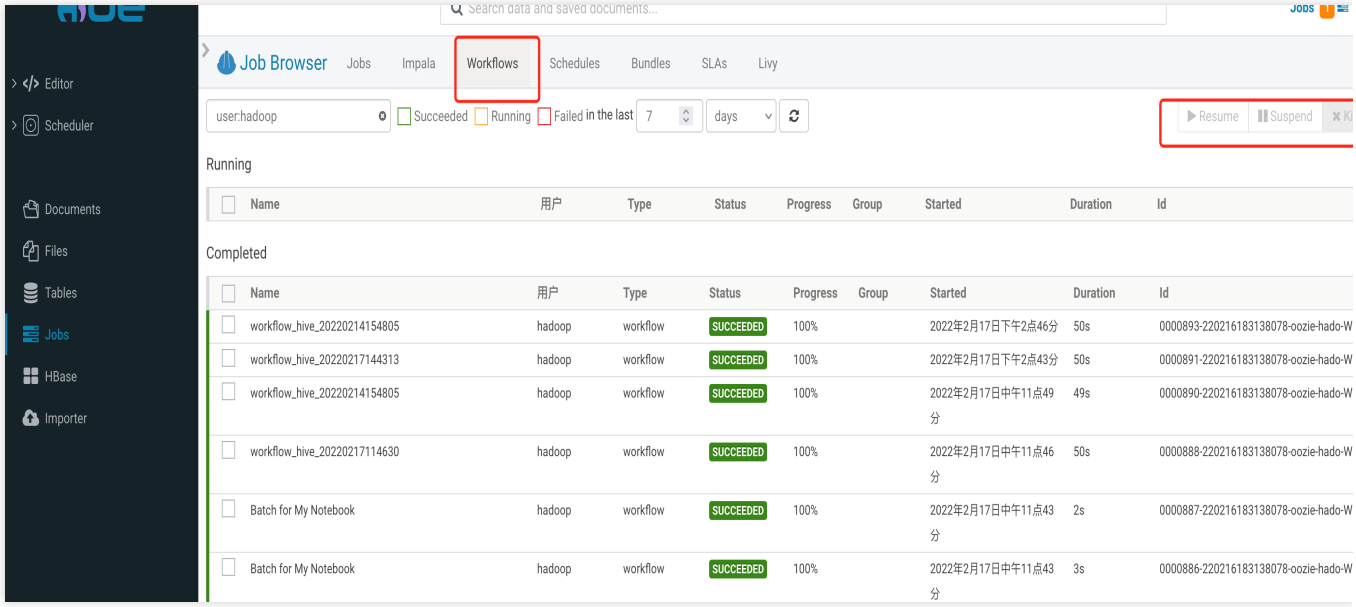
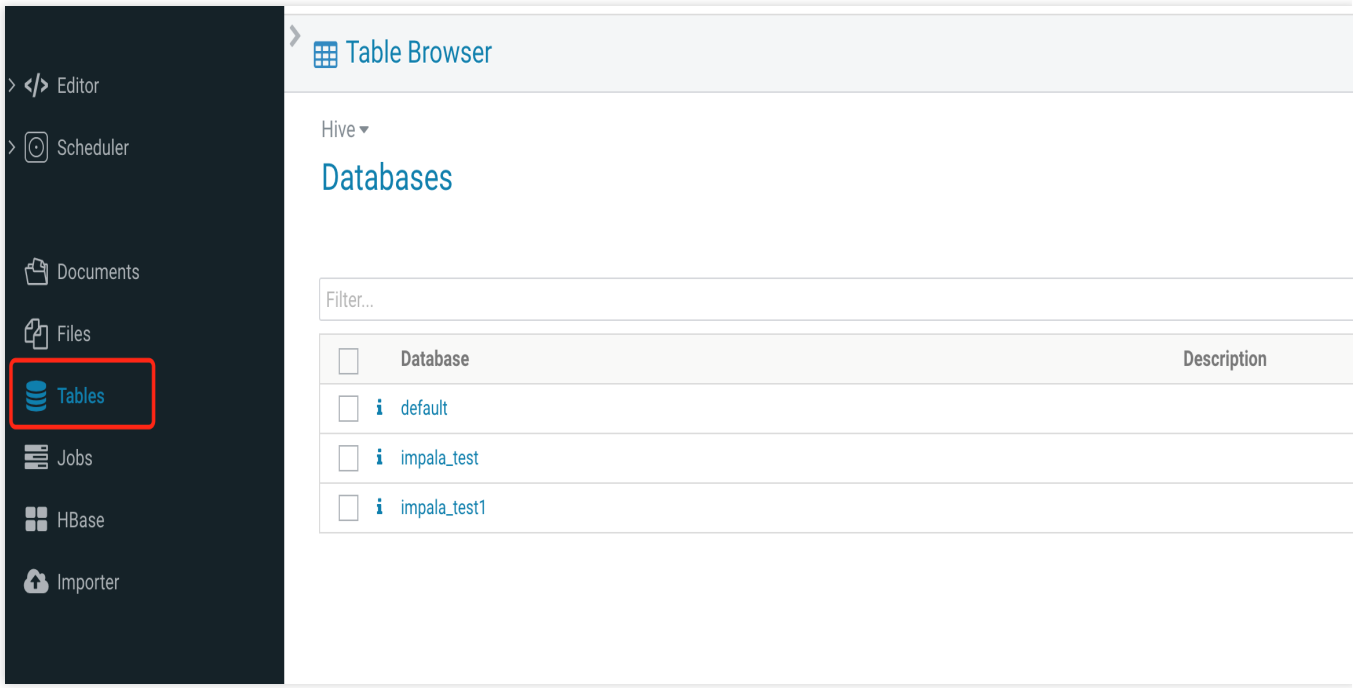


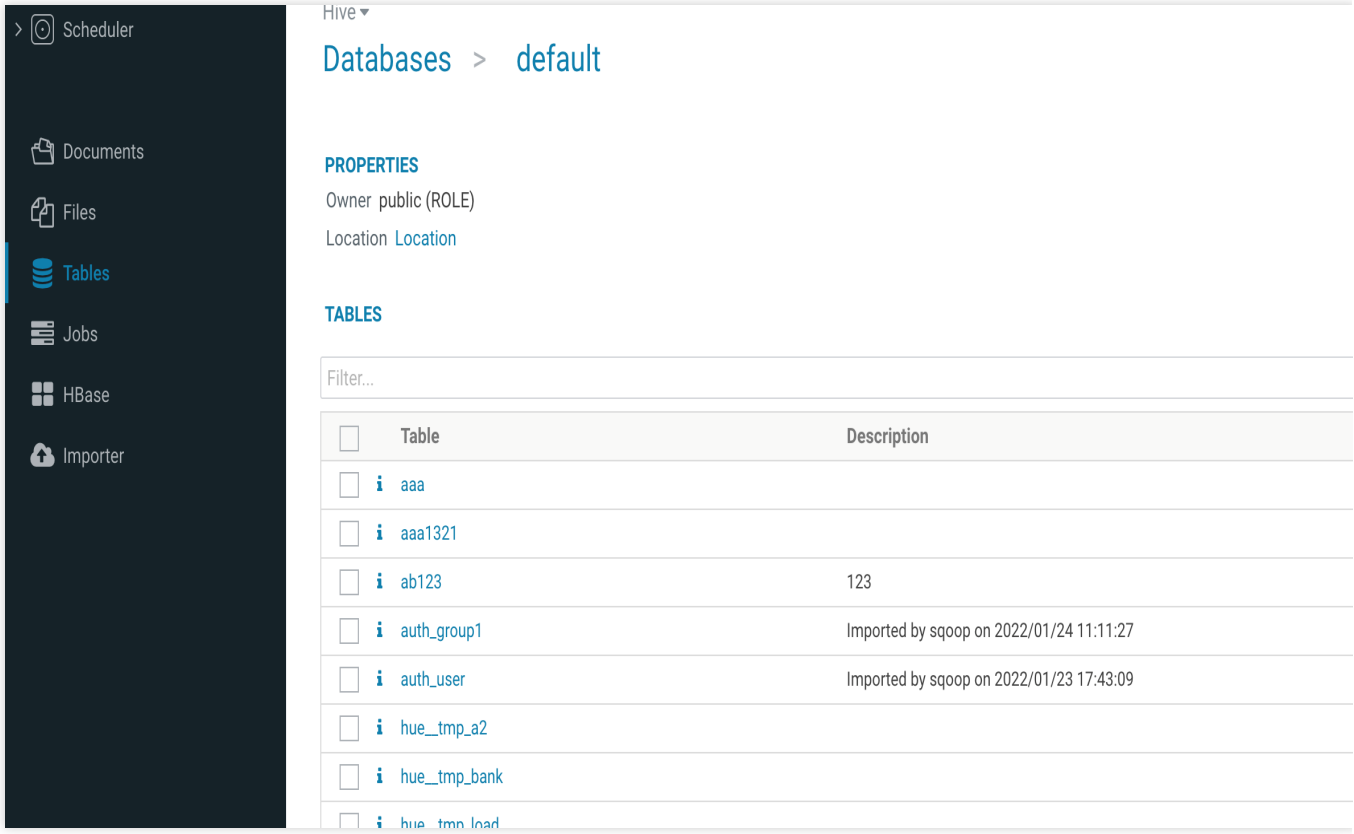
Table Management

1. Click **Tables** on the right to enter the table management page and view the basic database information.

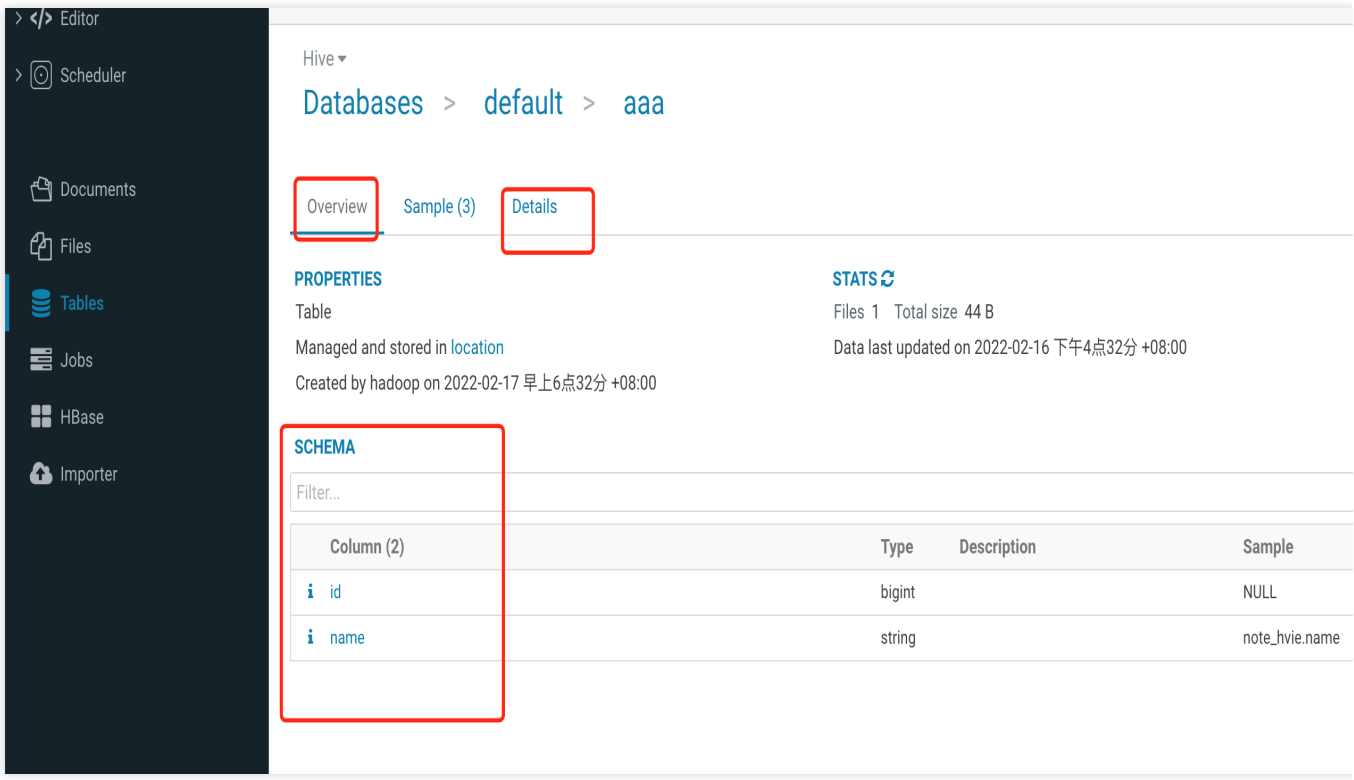


2. Click a database to view the information of its tables.

©2013-2024 Tencent Cloud. All rights reserved.



3. Click a table to view its details.



©2013-2024 Tencent Cloud. All rights reserved.

Hue Practical Tutorial

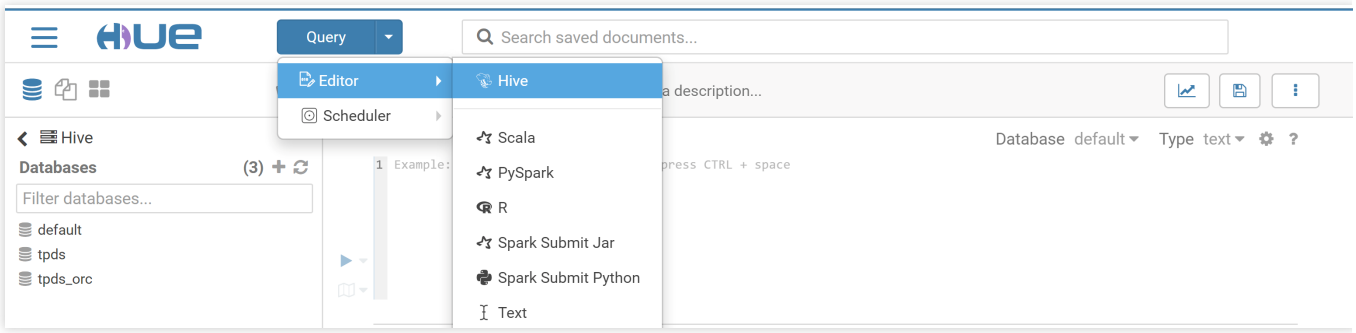
Last updated：2024-10-21 17:53:19

This document describes how to use Hue.

Hive SQL Query

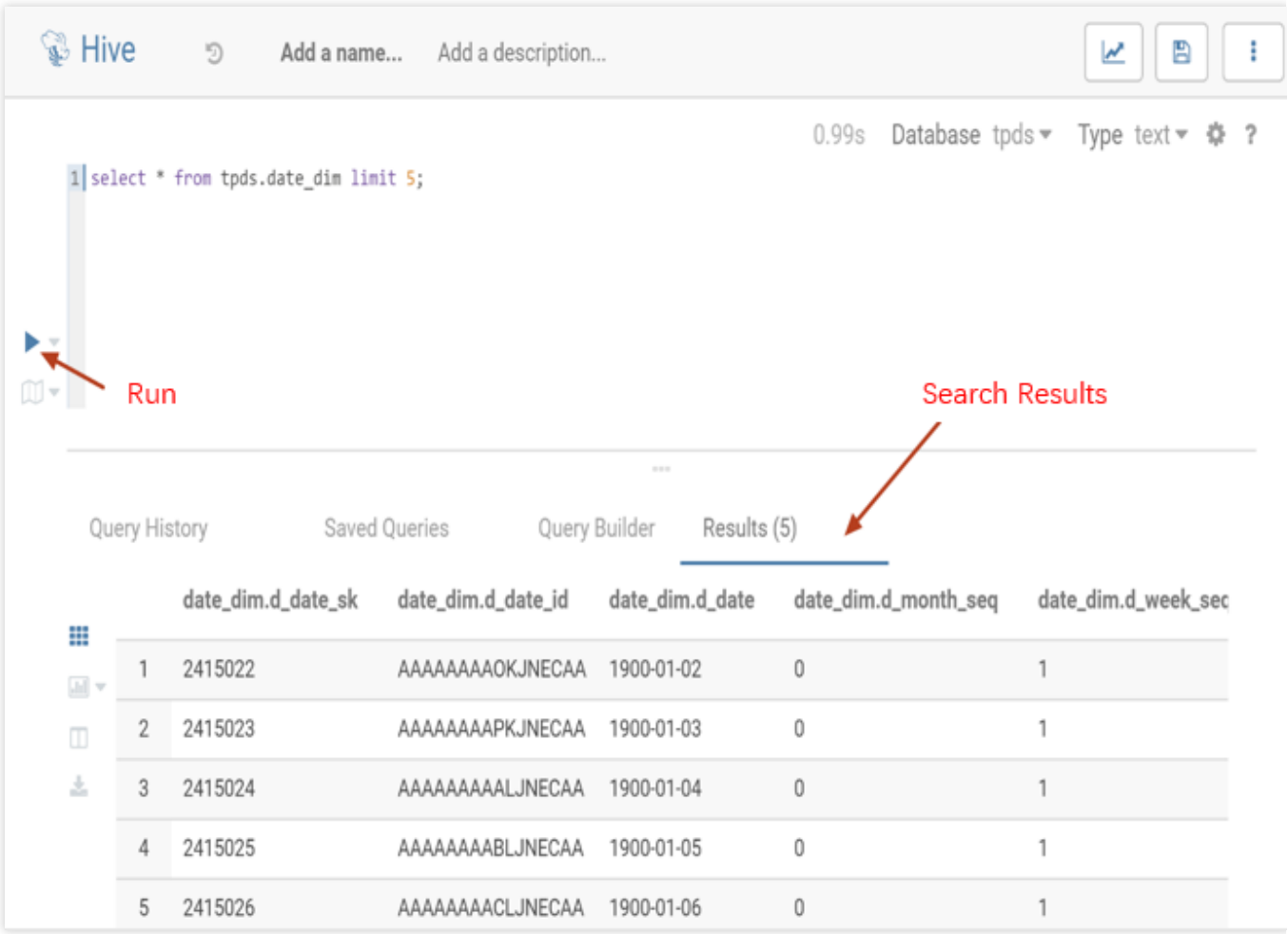
Hue's Beeswax app provides user-friendly and convenient Hive query capabilities, enabling you to select different Hive databases, write HQL statements, submit query tasks, and view results with ease.

1. At the top of the Hue console, select **Query** > **Editor** > **Hive**.



2. Enter the statement to be executed in the statement input box and click **Run** to run it.

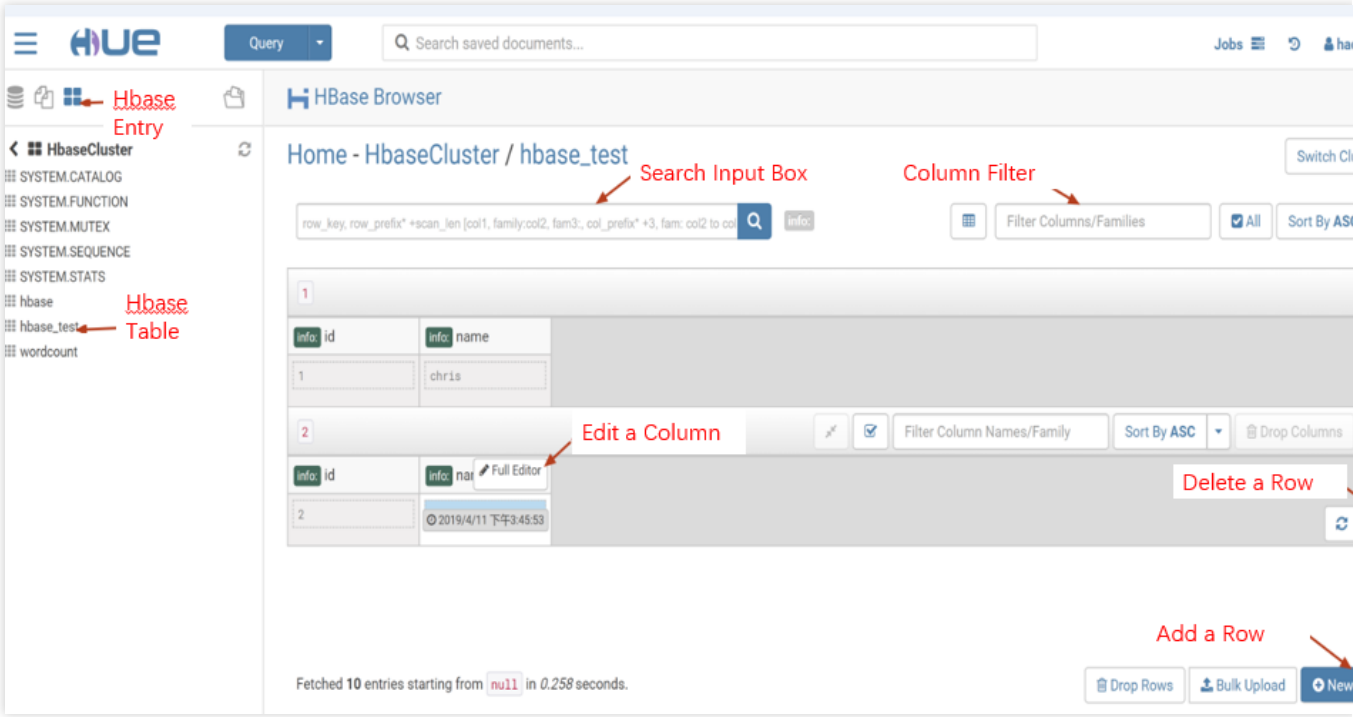
©2013-2024 Tencent Cloud. All rights reserved.



HBase Data Query, Modification, and Display

You can use HBase Browser to query, modify, and display data from tables in an HBase cluster.

©2013-2024 Tencent Cloud. All rights reserved.

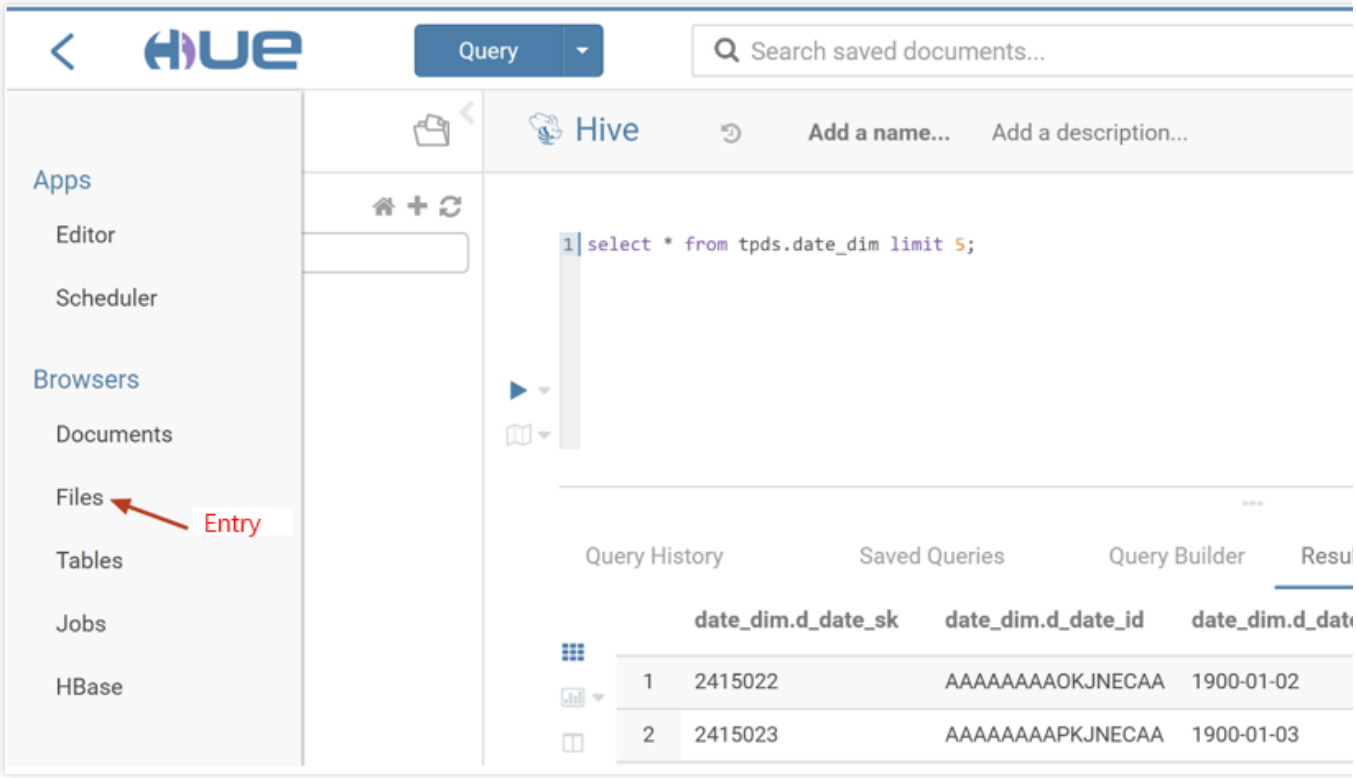


HDFS Access and File Browsing

Hue's web UI makes it easy to view files and folders in HDFS and perform operations such as creation, download, upload, copy, modification, and deletion.

1. On the left sidebar in the Hue console, select **Browsers** > **Files** to browse HDFS files.

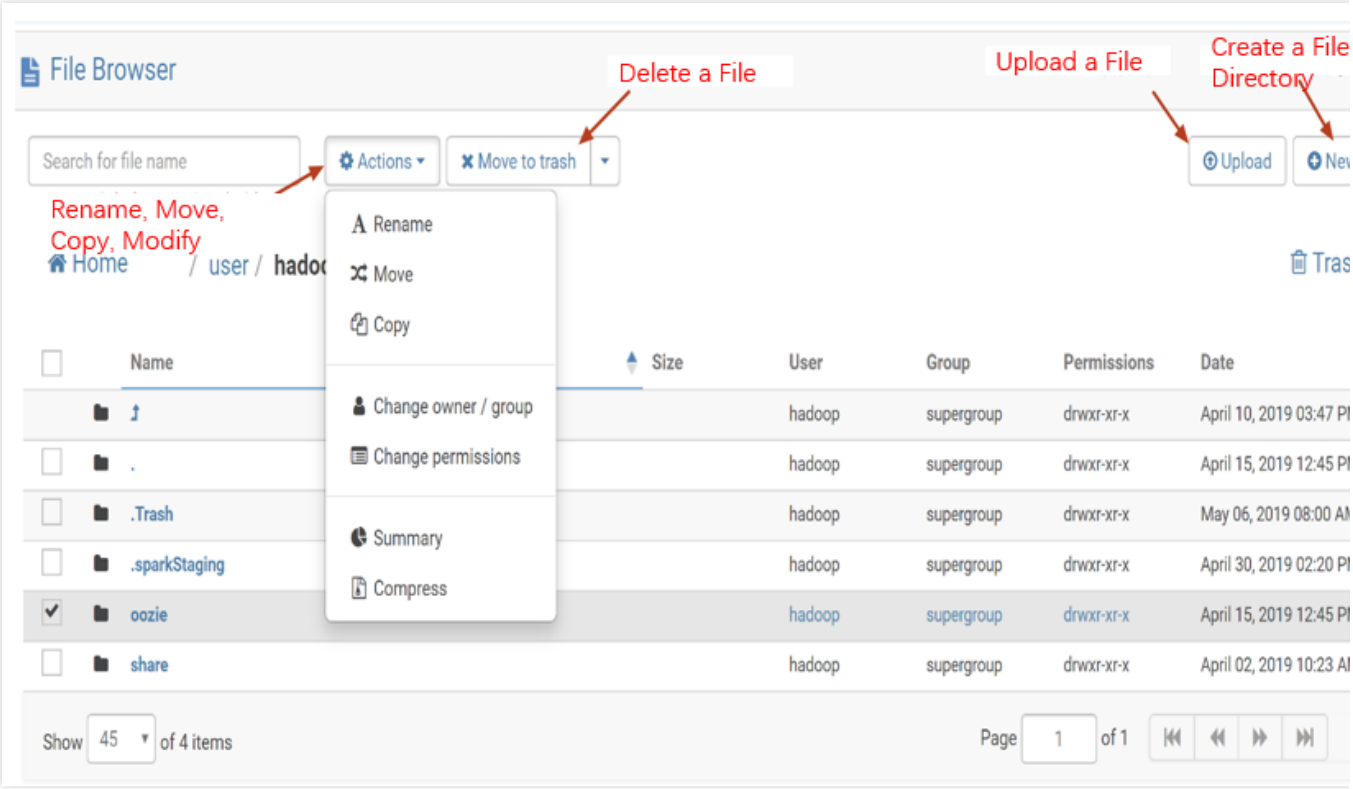
©2013-2024 Tencent Cloud. All rights reserved.



1.1

```
Perform various operations.
```

©2013-2024 Tencent Cloud. All rights reserved.



Oozie Job Development

1. Prepare workflow data: Hue's job scheduling is based on workflows. First, create a workflow containing a Hive script with the following content:

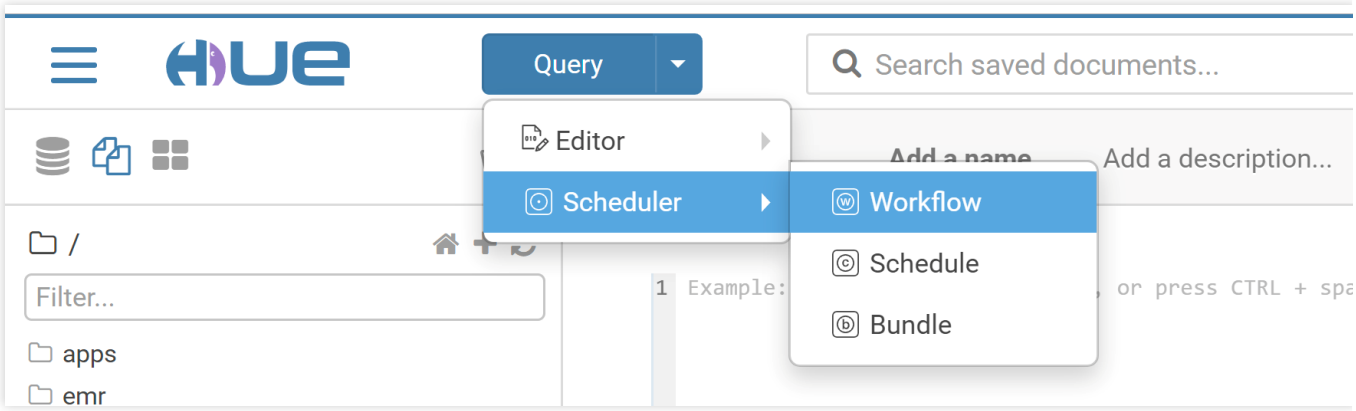
```
create database if not exists hive_sample;
show databases;
use hive_sample;
show tables;
create table if not exists hive_sample (a int, b string);
show tables;
insert into hive_sample select 1, "a";
select * from hive_sample;
```

Save the above content as a file named `hive_sample.sql`. The Hive workflow also requires a `hive-site.xml` configuration file, which can be found on the cluster node where the Hive component is installed. The specific path is `/usr/local/service/hive/conf/hive-site.xml`. Copy the `hive-site.xml` file and then upload the Hive script file and `hive-site.xml` to a directory in HDFS, such as `/user/hadoop`.

2. Create a workflow.

2.1 Switch to the `hadoop` user. At the top of the Hue console, select **Query** > **Scheduler** > **Workflow**.

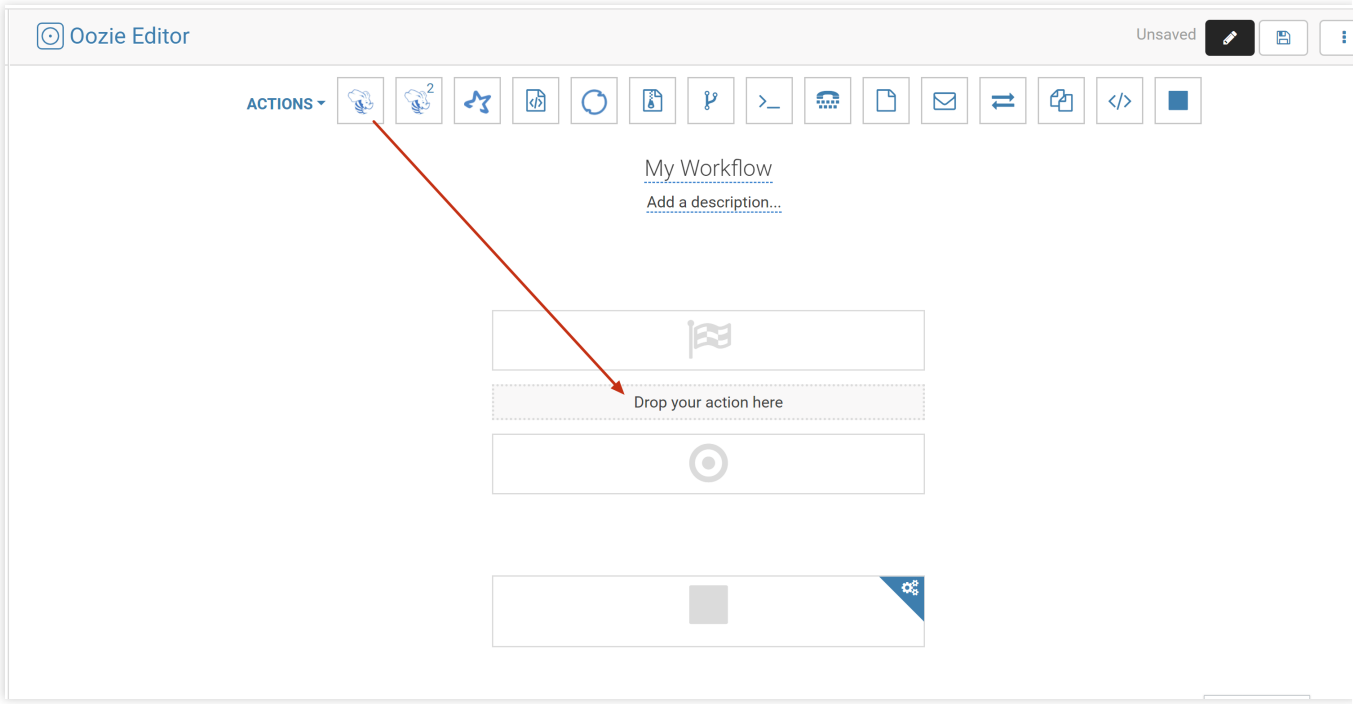
©2013-2024 Tencent Cloud. All rights reserved.



2.2 Drag a Hive script into the workflow editing page.

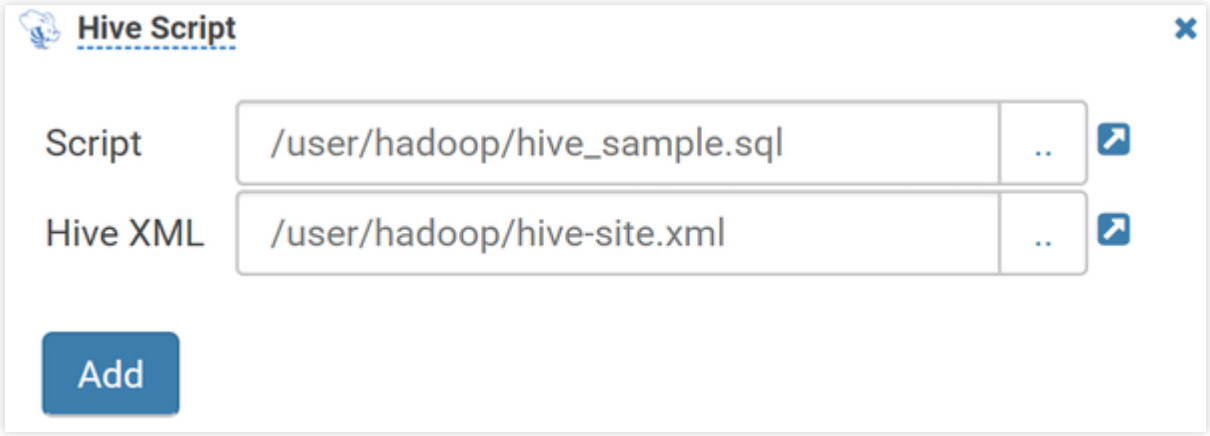
Caution

This document uses the installation of Hive v1 as an example, and the configuration parameter is `HiveServer1` . If it is deployed with other Hive versions (i.e., configuring configuration parameters of other versions), an error will be reported.

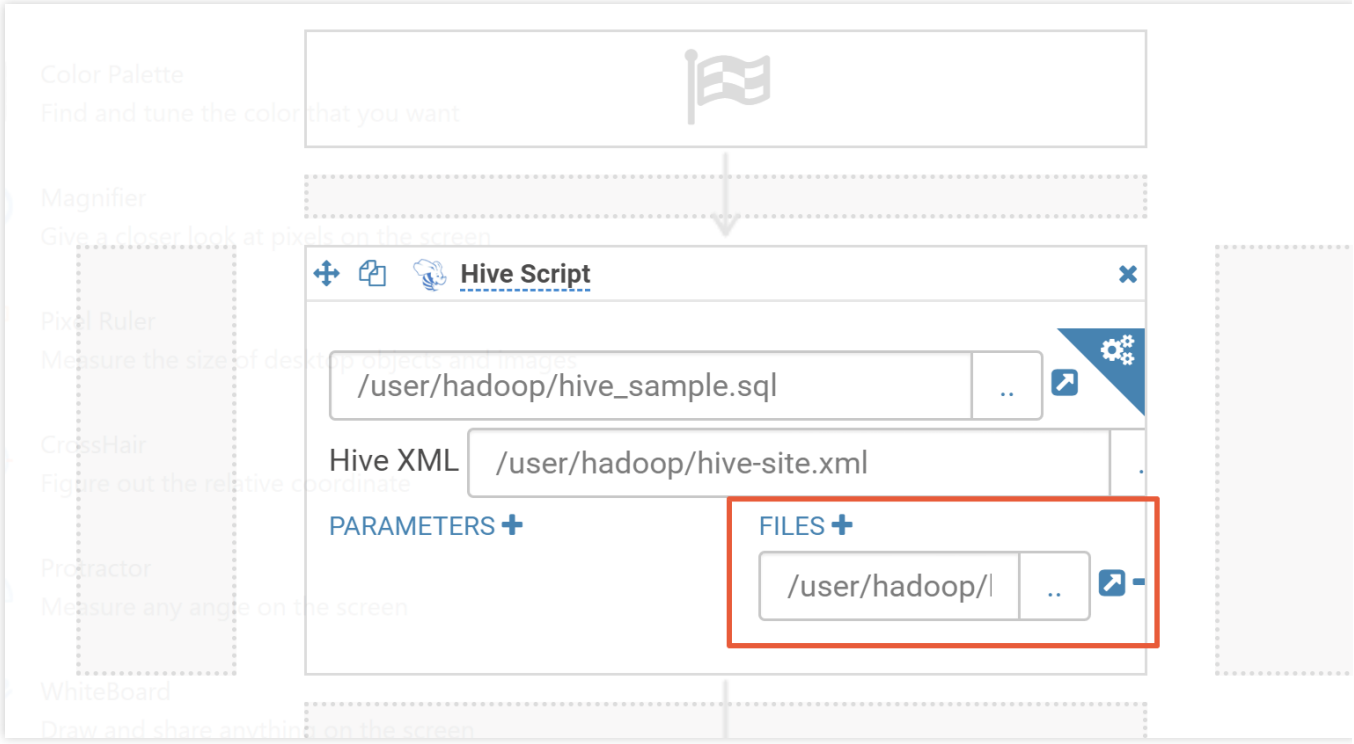


3. Select the Hive script and `hive-site.xml` files you just uploaded.

©2013-2024 Tencent Cloud. All rights reserved.

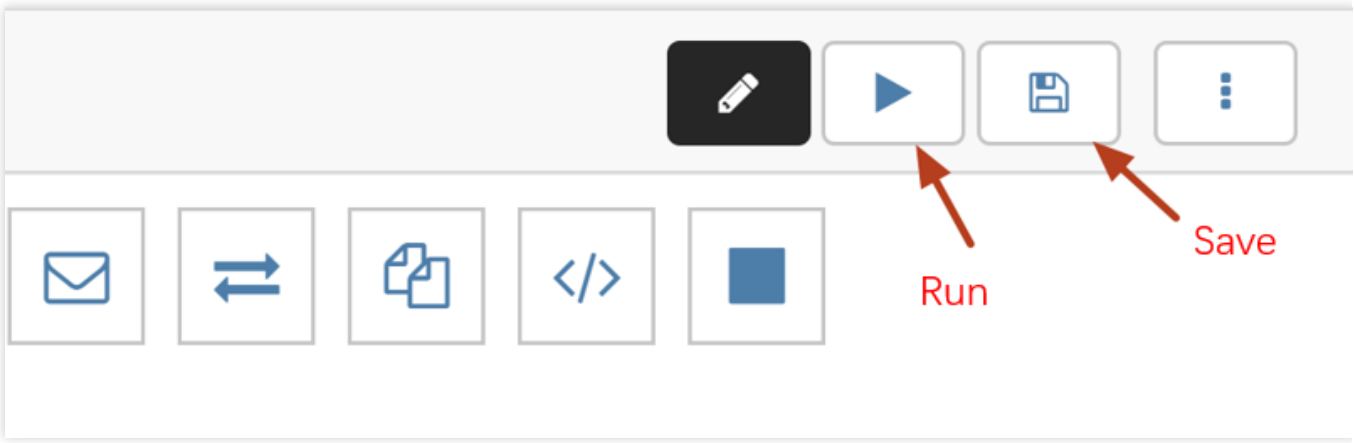


4. Click **Add** and specify the Hive script file in `FILES` .



5. Click **Save** in the top-right corner and then click **Run** to run the workflow.

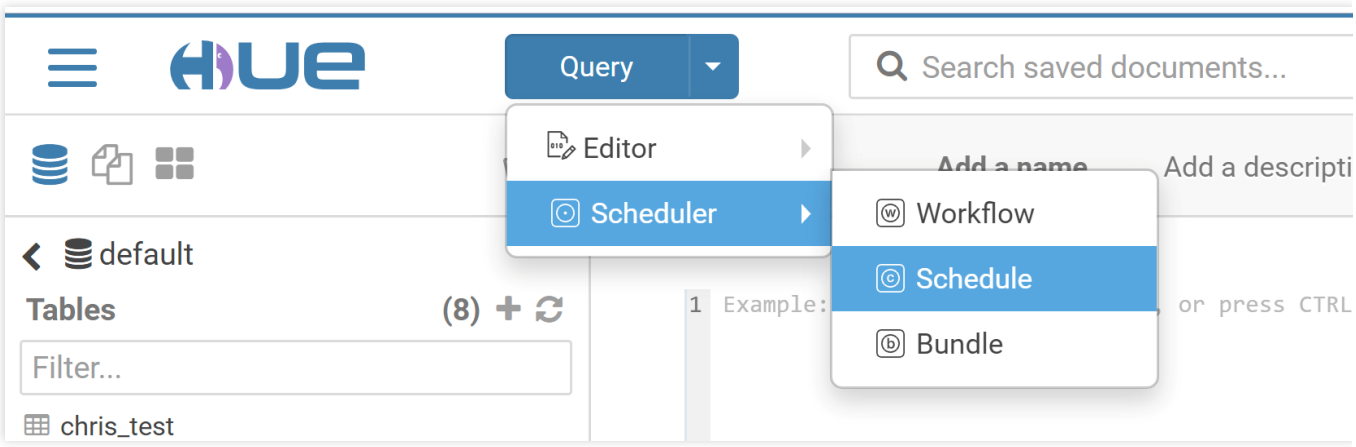
©2013-2024 Tencent Cloud. All rights reserved.



3. Create a scheduled job.

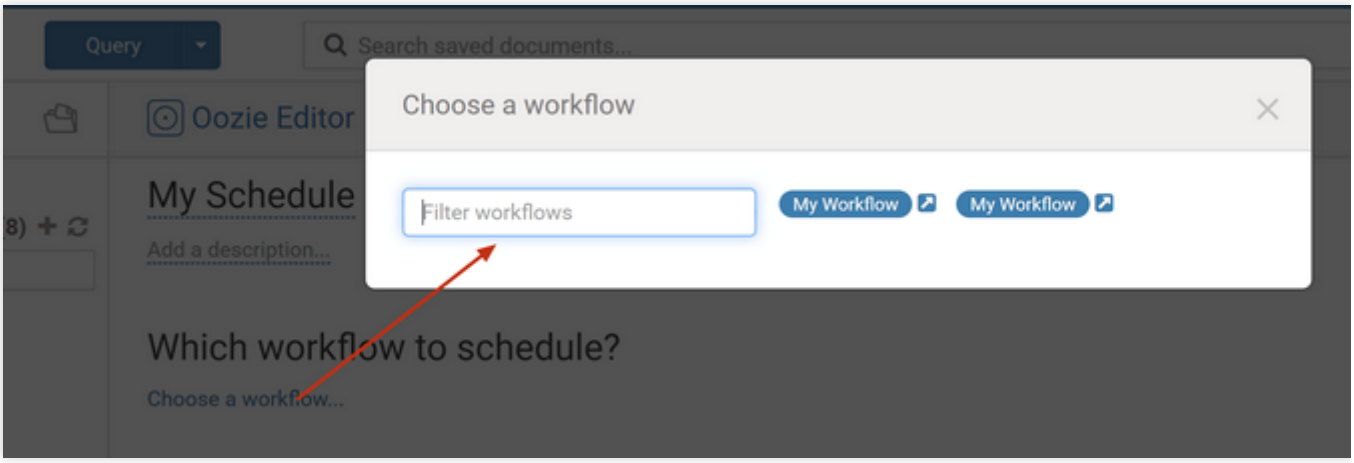
The scheduled job in Hive is "schedule", which is similar to the crontab in Linux. The supported scheduling granularity can be down to the minute level.

3.1 Select **Query** > **Scheduler** > **Schedule** to create a schedule.

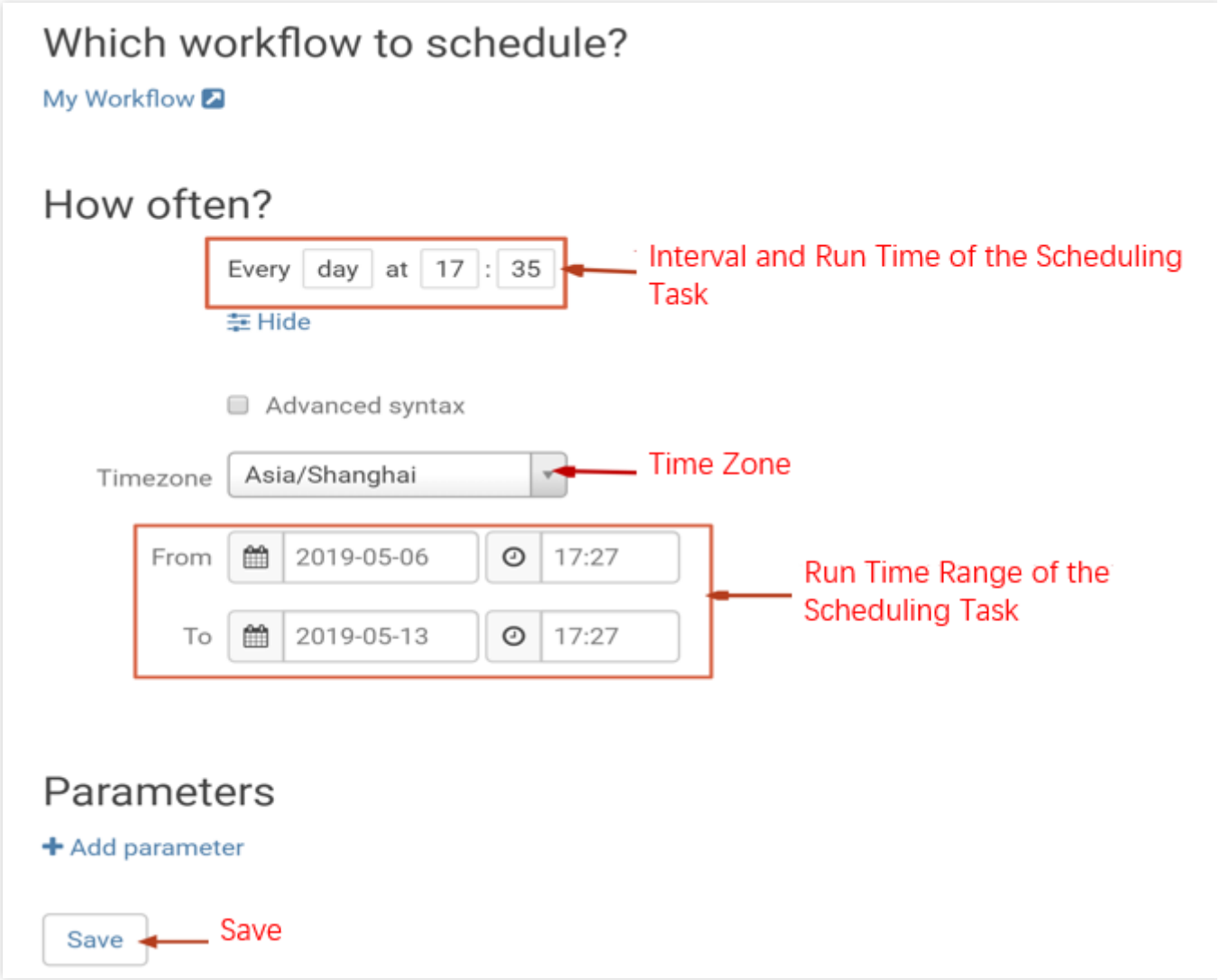


3.2 Click **Choose a workflow** to select a created workflow.

©2013-2024 Tencent Cloud. All rights reserved.



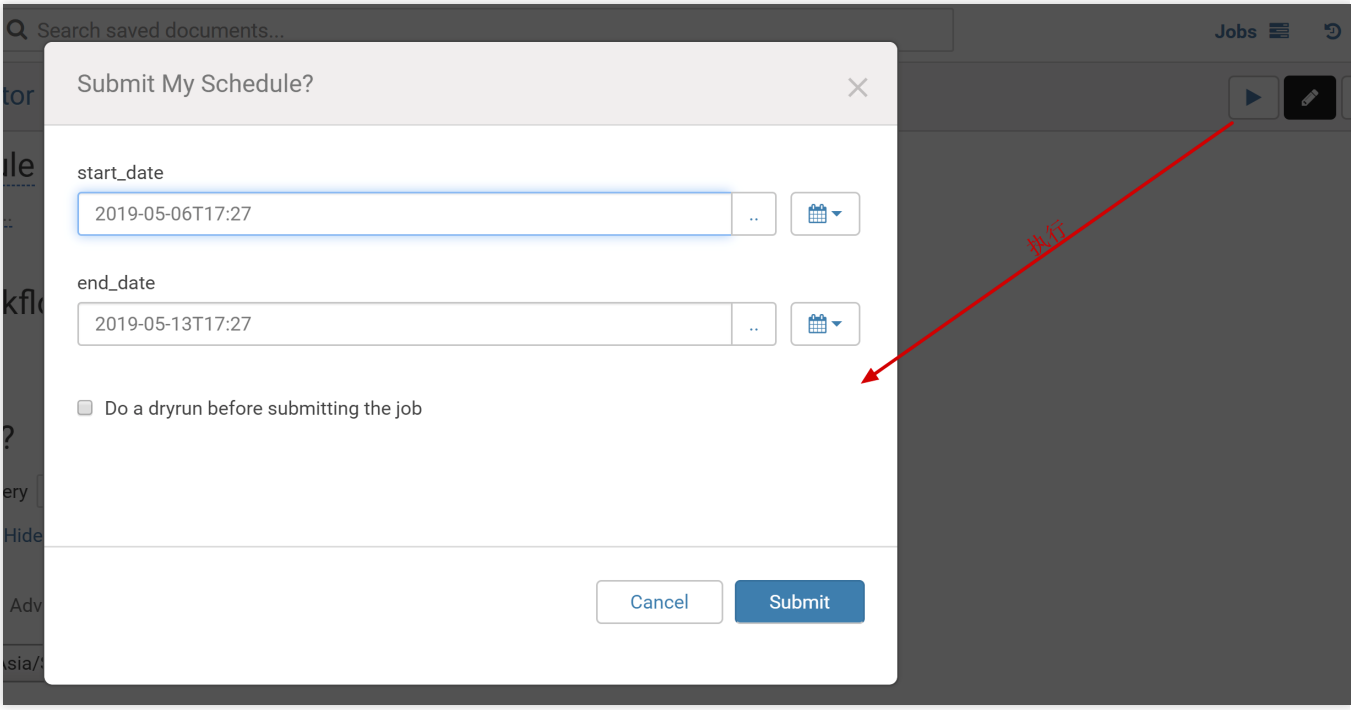
3.3 Select the execution time, frequency, time zone, start time, and end time of the schedule and click **Save**.



4. Create a scheduled job.

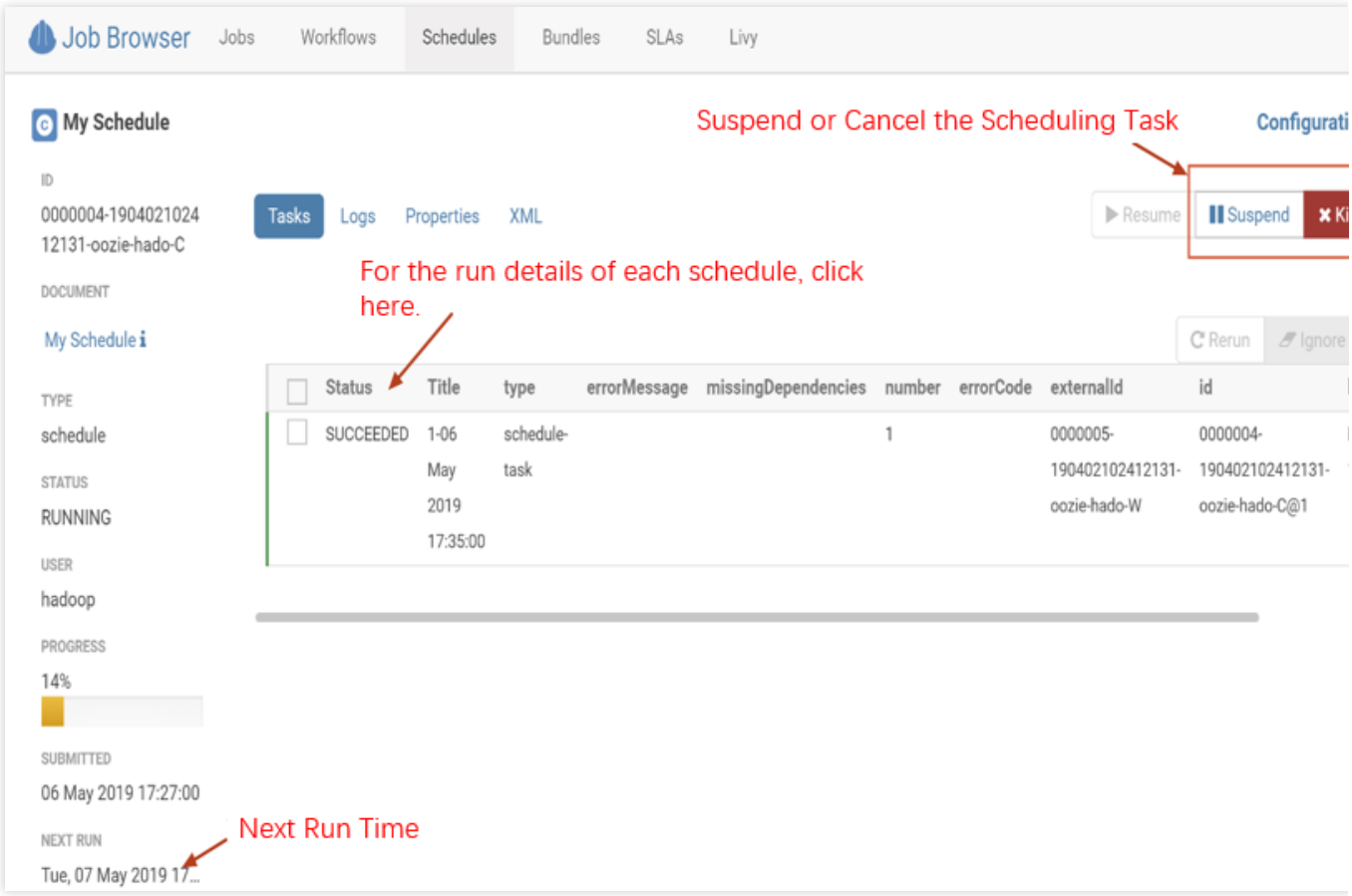
©2013-2024 Tencent Cloud. All rights reserved.

4.1 Click **Submit** in the top-right corner to submit the schedule.



4.2 You can view the scheduling status on the monitoring page of the schedulers.

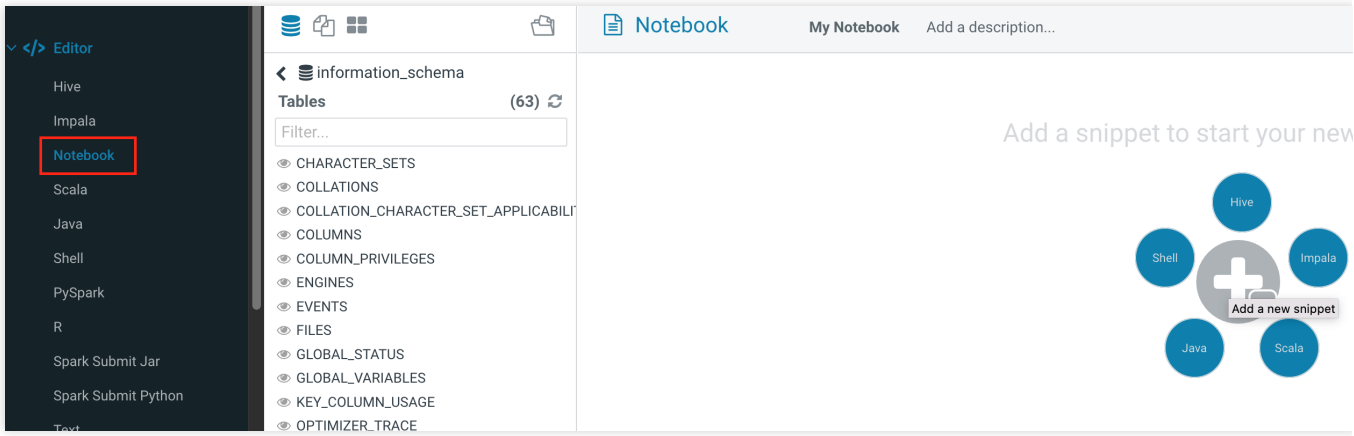
©2013-2024 Tencent Cloud. All rights reserved.



Notebook Query and Comparative Analysis

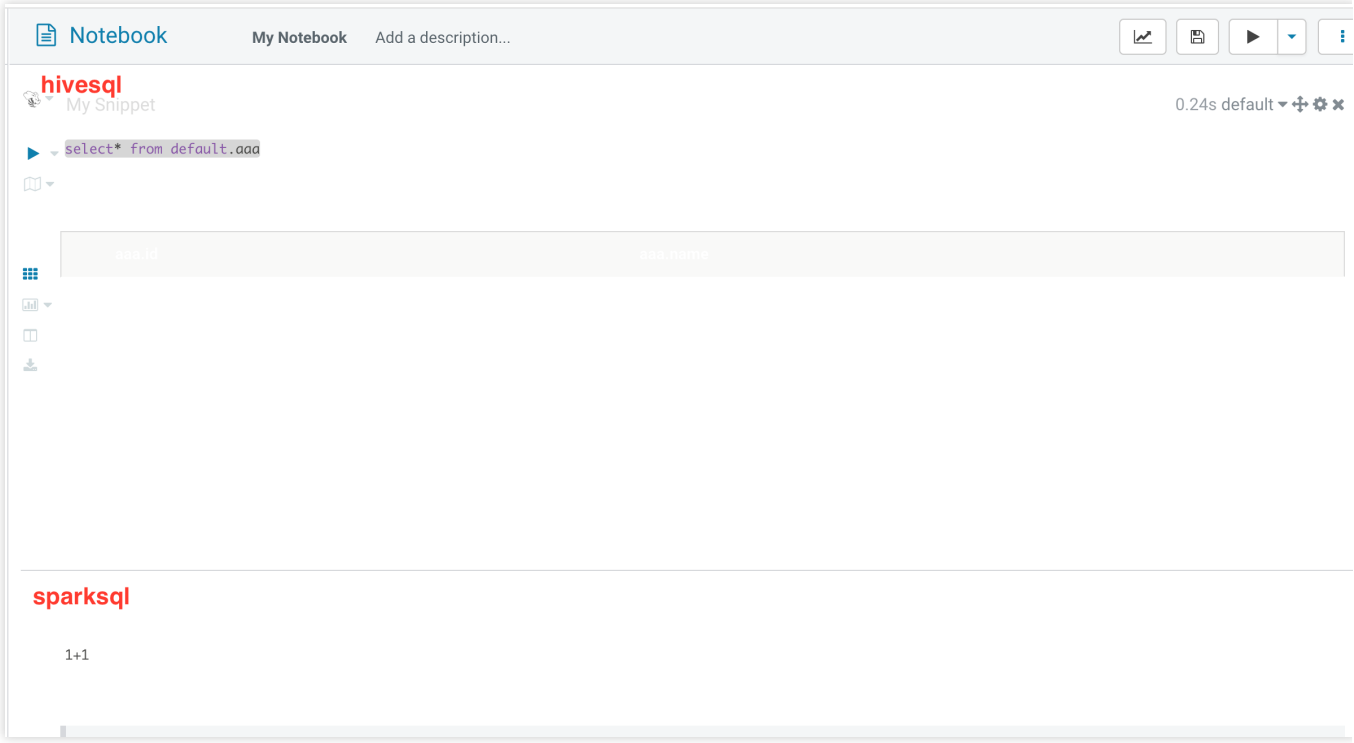
Notebooks can quickly build access requests and queries and put the query results together for comparative analysis. It supports five types: Hive, Impala, Spark, Java, and Shell.

1. Click **Editor**, **Notebook**, and **+** to add the required query.



©2013-2024 Tencent Cloud. All rights reserved.

2. Click **Save** to save the added notebook and click **Run** to run the entire notebook.



©2013-2024 Tencent Cloud. All rights reserved.

Oozie Development Guide

Last updated：2025-02-12 16:49:07

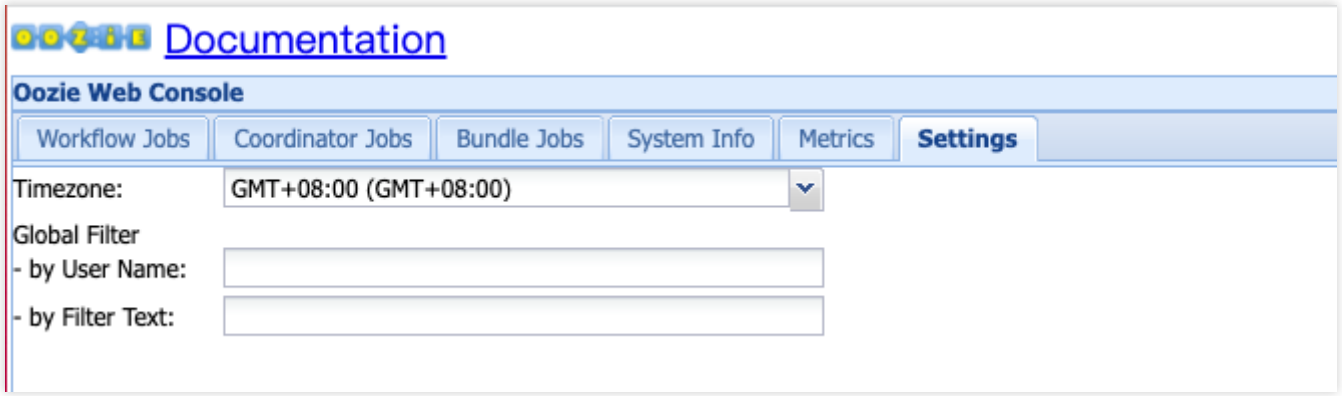
Apache Oozie is an open-source workflow engine. It is designed to orchestrate the tasks of Hadoop ecosystem components into workflows and then schedule, execute, and monitor them. This document briefly describes how to use Oozie in EMR. For detailed directions, visit the website. Here, we recommend you use Oozie through Hue's GUI as instructed in the Hue development documentation.

Prerequisites

You have created an EMR Hadoop cluster and selected the Oozie service. For more information, see Creating EMR Cluster.

Accessing Oozie WebUI

If you have enabled public network access for cluster nodes during cluster purchase, you can click the WebUI link in the EMR console for access.
If you are in the Chinese mainland, we recommend you set the WebUI time zone to GMT+08:00.



Updating ShareLib

As the EMR cluster is preinstalled with ShareLib, you no longer need to install it when using Oozie to submit a workflow job. Of course, you can edit and update ShareLib as instructed below:

```
cd /usr/local/service/oozie
Add `tar -xf oozie-sharelib.tar.gz` to `bin/oozie-setup.sh sharelib create -fs
hdfs://active-namenode-ip:4007 -locallib shareoozie admin --oozie http://oozie-
```

©2013-2024 Tencent Cloud. All rights reserved.

```
server-ip:12000/oozie -sharelibupdate` in the directory of the action to be
supported in the `share` directory generated by decompressing the JAR package.
```

Submitting Workflow in Non-Kerberos Environment

Decompress the `oozie-examples.tar.gz` file in the Oozie installation directory `/usr/local/service/oozie`, which provides the sample workflows of the components supported by Oozie:

```
tar -xf oozie-examples.tar.gz
```

Take `action hive2` as an example:

su hadoop.

cd examples/apps/hive2/.

Modify `job.properties`:

Set the value of `namenode` to the value of `fs.defaultFS` in `core-site.xml`.

Set the value of **resourceManager** to the value of `yarn.resourcemanager.ha.rm-ids` in `yarn-site.xml` in HA mode, or to the value of `yarn.resourcemanager.address` in non-HA mode.

The value of **jdbcURL** is `jdbc:hive2://hive2-server:7001/default`.

hadoop fs -put examples.

oozie job -debug -oozie http://oozie-server-ip:12000/oozie -config examples/apps/hive2/job.properties -run.

oozie job -info the job ID returned in the previous step (or viewed on the WebUI).

Submitting Workflow in Kerberos Environment

Take `action hive2` as an example again. Check the README file in the `hive2` directory for other notes.

kinit -kt /var/krb5kdc/emr.keytab hadoop's principal && su hadoop.

cd examples/apps/hive2/.

mv job.properties.security job.properties && mv workflow.xml.security workflow.xml.

Modify `job.properties`:

Set the value of `namenode` to the value of `fs.defaultFS` in `core-site.xml`.

Set the value of **resourceManager** to the value of `yarn.resourcemanager.ha.rm-ids` in `yarn-site.xml` in HA mode, or to the value of `yarn.resourcemanager.address` in non-HA mode.

The value of `jdbcURL` is `jdbc:hive2://hive2-server:7001/default`.

The value of `jdbcPrincipal` is the value of `hive.server2.authentication.kerberos.principal`.

hadoop fs -put examples.

oozie job -debug -oozie http://oozie-server-ip:12000/oozie -config examples/apps/hive2/job.properties -run.

©2013-2024 Tencent Cloud. All rights reserved.

oozie job -info the job ID returned in the previous step (or viewed on the WebUI).

©2013-2024 Tencent Cloud. All rights reserved.

Flume Development Guide
Flume Overview

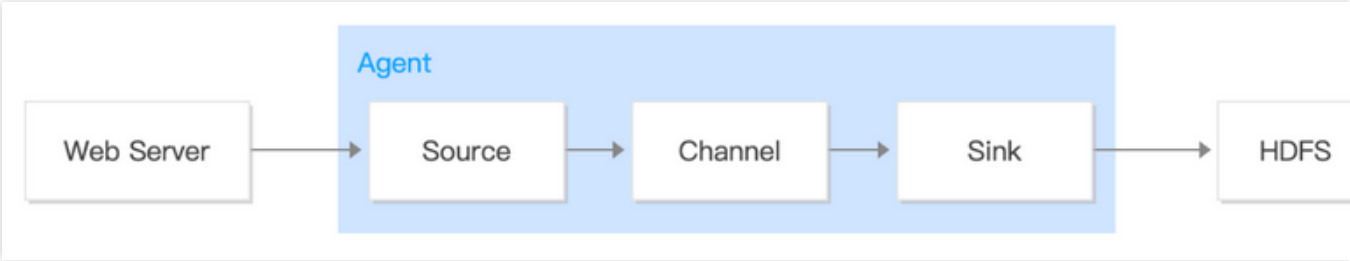
Last updated：2025-02-12 16:42:48

Flume Overview

Apache Flume is a highly available, distributed, and configurable tool/service that collects and aggregates large amounts of data such as logs and events from different sources. It is designed to collect data flows (e.g., log data) from various web servers and store them in centralized data storage system like HDFS and HBase.

Flume Architecture

A Flume event is defined as a unit of data flow. A Flume agent is a JVM process that contains the components required for completing a task. Among them, Source, Channel, and Sink are the core ones.



Source
It consumes an event passed to it by an external source (e.g., web servers or other sources) and save it to one or more channels.
Channel
Located between a source and a sink, a channel is used to cache incoming events. After the sink successfully sends the events to the channel at the next hop or the final destination, the events are removed from the channel.
Sink
A sink is responsible for transferring the events to the next hop or final destination and removing them from the channel upon transfer completion.

Instructions

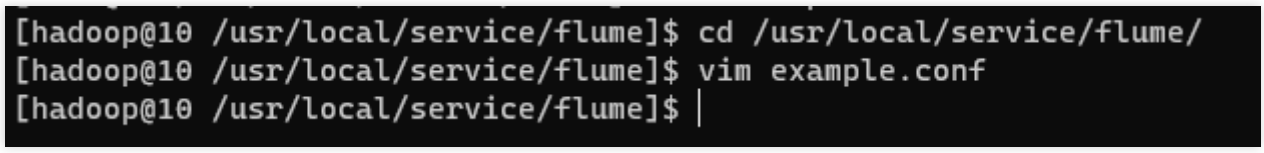
©2013-2024 Tencent Cloud. All rights reserved.

Preparations

Create an EMR cluster. When creating the EMR cluster, you need to select the Flume component on the software configuration page.
Install Flume in the `/usr/local/service/flume` path on the core and task nodes (CVM instances) of the EMR cluster. The installation path for master nodes is `/usr/local/service/apps/` .

Configuring Flume

Go to the `/usr/local/service/flume` folder and create a file named `example.conf` .




```
# example.conf: A single-node Flume configuration

# Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444

# Describe the sink
a1.sinks.k1.type = logger

# Use a channel which buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

Starting Flume

```
bin/flume-ng agent --conf conf --conf-file example.conf --name a1 -
Dflume.root.logger=INFO,console
```

©2013-2024 Tencent Cloud. All rights reserved.

Configuring a test sample

After successful configuration, you will see the Flume agent started previously printing to the terminal.

```
telnet localhost 44444
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
Hello world! <ENTER>
OK
```

©2013-2024 Tencent Cloud. All rights reserved.

Storing Kafka Data in Hive Through Flume

Last updated：2025-02-12 16:42:49

Scenario Description

Data in Kafka can be collected through Flume and stored in Hive.

Preparations for Development

As this job requires access to CKafka, you need to create a CKafka instance first. For more information, please see CKafka.
Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, you need to select the Flume component on the software configuration page.

Using the Kafka Toolkit in the EMR Cluster

First, you need to check the private IP and port number of CKafka. Log in to the CKafka Console, select the CKafka instance you want to use, and view its private IP as `$kafkaIP` in the basic information section, and the port number is generally 9092 by default. Create a topic named `kafka_test` on the topic management page.

Configuring Flume

1. Create the Flume configuration file `hive_kafka.properties` .

```
vim hive_kafka.properties
agent.sources = kafka_source
agent.channels = mem_channel
agent.sinks = hive_sink
# The following code is used to configure the source
agent.sources.kafka_source.type = org.apache.flume.source.kafka.KafkaSource
agent.sources.kafka_source.channels = mem_channel
agent.sources.kafka_source.batchSize = 5000
agent.sources.kafka_source.kafka.bootstrap.servers = $kafkaIP:9092
agent.sources.kafka_source.kafka.topics = kafka_test
# The following code is used to configure the sink
agent.sinks.hive_sink.channel = mem_channel
agent.sinks.hive_sink.type = hive
```

©2013-2024 Tencent Cloud. All rights reserved.

```
agent.sinks.hive_sink.hive.metastore = thrift://172.16.32.51:7004
agent.sinks.hive_sink.hive.database = default
agent.sinks.hive_sink.hive.table = weblogs
agent.sinks.hive_sink.hive.partition = asia,india,%y-%m-%d-%H-%M
agent.sinks.hive_sink.useLocalTimeStamp = true
agent.sinks.hive_sink.round = true
agent.sinks.hive_sink.roundValue = 10
agent.sinks.hive_sink.roundUnit = minute
agent.sinks.hive_sink.serializer = DELIMITED
agent.sinks.hive_sink.serializer.delimiter = ","
agent.sinks.hive_sink.serializer.serdeSeparator = ','
agent.sinks.hive_sink.serializer.fieldnames =id,msg
# The following code is used to configure the channel
agent.channels.mem_channel.type = memory
agent.channels.mem_channel.capacity = 100000
agent.channels.mem_channel.transactionCapacity = 100000
```

You can confirm Hive Metastore in the following way:

```
grep "hive.metastore.uris" -C 2 /usr/local/service/hive/conf/hive-site.xml
```

```
<property>
<name>hive.metastore.uris</name>
<value>thrift://172.16.32.51:7004</value>
</property>
```

2. Create a Hive table.

```
create table weblogs ( id int , msg string )
partitioned by (continent string, country string, time string)
clustered by (id) into 5 buckets
stored as orc TBLPROPERTIES ('transactional'='true');
```

Note:

All the following conditions must be met: it must be a table with partitions and buckets, the storage format is ORC, and `TBLPROPERTIES ('transactional'='true')` is set.

3. Enable the Hive transaction.

In the console, add the following configuration items to `hive-site.xml` .

```
<property>
<name>hive.support.concurrency</name>
<value>true</value>
</property>
<property>
<name>hive.exec.dynamic.partition.mode</name>
<value>nonstrict</value>
```

©2013-2024 Tencent Cloud. All rights reserved.

```
</property>
<property>
<name>hive.txn.manager</name>
<value>org.apache.hadoop.hive.ql.lockmgr.DbTxnManager</value>
</property>
<property>
<name>hive.compactor.initiator.on</name>
<value>true</value>
</property>
<property>
<name>hive.compactor.worker.threads</name>
<value>1</value>
</property>
<property>
<name>hive.enforce.bucketing</name>
<value>true</value>
</property>
```

Note:

After the configuration is distributed and Hive is restarted, the `hadoop-hive` log will prompt that the Metastore cannot be connected to. Please ignore this error. Because of the startup order of the processes, Metastore will be started before HiveServer2.

4. Copy `hive-hcatalog-streaming-xxx.jar` of Hive to the `lib` directory of Flume.

```
cp -ra /usr/local/service/hive/hcatalog/share/hcatalog/hive-hcatalog-streaming-
2.3.3.jar /usr/local/service/flume/lib/
```

5. Run Flume.

```
./bin/flume-ng agent --conf ./conf/ -f hive_kafka.properties -n agent -
Dflume.root.logger=INFO,console
```

6. Run the Kafka producer.

```
[hadoop@172 kafka]$ ./bin/kafka-console-producer.sh --broker-list $kafkaIP:9092
--topic kafka_test
1,hello
2,hi
```

Test

Enter information on the client of the Kafka producer and press Enter.

Check whether there is corresponding data in the Hive table.

©2013-2024 Tencent Cloud. All rights reserved.

Reference Documentation

Hive Sink Configuration Description

Hive Log Configuration Description

©2013-2024 Tencent Cloud. All rights reserved.

Storing Kafka Data in HDFS or COS Through Flume

Last updated：2025-02-12 16:42:49

Scenario Description

Collecting and saving the data in Kafka to HDFS or COS via Flume

Development Preparations

This task requires access to CKafka, so you need to create a CKafka instance first. For more information, see Message Queue CKafka.
Create an EMR cluster. When creating the EMR cluster, you need to select the Flume component on the software configuration page and enable access to COS on the basic configuration page.

Using the Kafka Toolkit in the EMR Cluster

First, you need to check the private IP and port number of CKafka. Log in to the CKafka console, select the CKafka instance you want to use, and find the private IP (`$kafkaIP`) and port number (generally `9092`) in the basic information section. Create a topic named `kafka_test` on the topic management page.

Configuring Flume

1. Create a Flume configuration file `kafka.properties` .

```
vim kafka.properties
agent.sources = kafka_source
agent.channels = mem_channel
agent.sinks = hdfs_sink
# The following is for source configuration.
agent.sources.kafka_source.type = org.apache.flume.source.kafka.KafkaSource
agent.sources.kafka_source.channels = mem_channel
agent.sources.kafka_source.batchSize = 5000
agent.sources.kafka_source.kafka.bootstrap.servers = $kafkaIP:9092
agent.sources.kafka_source.kafka.topics = kafka_test
```

©2013-2024 Tencent Cloud. All rights reserved.

```
# The following is for sink configuration.
agent.sinks.hdfs_sink.type = hdfs
agent.sinks.hdfs_sink.channel = mem_channel
agent.sinks.hdfs_sink.hdfs.path = /data/flume/kafka/%Y%m%d (or
cosn://bucket/xxx)
agent.sinks.hdfs_sink.hdfs.rollSize = 0
agent.sinks.hdfs_sink.hdfs.rollCount = 0
agent.sinks.hdfs_sink.hdfs.rollInterval = 3600
agent.sinks.hdfs_sink.hdfs.threadsPoolSize = 30
agent.sinks.hdfs_sink.hdfs.fileType=DataStream
agent.sinks.hdfs_sink.hdfs.useLocalTimeStamp=true
agent.sinks.hdfs_sink.hdfs.writeFormat=Text
# The following is for channel configuration.
agent.channels.mem_channel.type = memory
agent.channels.mem_channel.capacity = 100000
agent.channels.mem_channel.transactionCapacity = 10000
```

2. Run Flume.

```
./bin/flume-ng agent --conf ./conf/ -f kafka.properties -n agent -
Dflume.root.logger=INFO,console
```

3. Run Kafka producer.

```
[hadoop@172 kafka]$ ./bin/kafka-console-producer.sh --broker-list $kafkaIP:9092
--topic kafka_test
test
hello
```

Testing

Enter information on the Kafka producer client and press Enter.

Check whether the corresponding directory and file `hadoop fs -ls /data/flume/kafka/` have been generated in HDFS.

References

Kafka Source Configuration Description

©2013-2024 Tencent Cloud. All rights reserved.

Storing Kafka Data in Hive Through Flume

Last updated：2025-02-12 16:42:49

Scenario Description

Data in Kafka can be collected through Flume and stored in HBase.

Preparations for Development

As this job requires access to CKafka, you need to create a CKafka instance first. For more information, please see CKafka.
Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, you need to select the Flume component on the software configuration page.

Using the Kafka Toolkit in the EMR Cluster

First, you need to check the private IP and port number of CKafka. Log in to the CKafka Console, select the CKafka instance you want to use, and view its private IP as `$kafkaIP` in the basic information section, and the port number is generally 9092 by default. Create a topic named `kafka_test` on the topic management page.

Configuring Flume

1. Create the Flume configuration file `hbase_kafka.properties` .

```
vim hbase_kafka.properties
agent.sources = kafka_source
agent.channels = mem_channel
agent.sinks = hbase_sink
# The following code is used to configure the source
agent.sources.kafka_source.type = org.apache.flume.source.kafka.KafkaSource
agent.sources.kafka_source.channels = mem_channel
agent.sources.kafka_source.batchSize = 5000
agent.sources.kafka_source.kafka.bootstrap.servers = $kafkaIP:9092
agent.sources.kafka_source.kafka.topics = kafka_test
# The following code is used to configure the sink
agent.sinks.hbase_sink.channel = mem_channel
agent.sinks.hbase_sink.table = foo_table
```

©2013-2024 Tencent Cloud. All rights reserved.

```
agent.sinks.hbase_sink.columnFamily = cf
agent.sinks.hbase_sink.serializer =
org.apache.flume.sink.hbase.RegexHbaseEventSerializer
# The following code is used to configure the channel
agent.channels.mem_channel.type = memory
agent.channels.mem_channel.capacity = 100000
agent.channels.mem_channel.transactionCapacity = 10000
```

2. Create an HBase table.

```
hbase shell
create 'foo_table','cf'
```

3. Run Flume.

```
./bin/flume-ng agent --conf ./conf/ -f hbase_kafka.properties -n agent -
Dflume.root.logger=INFO,console
```

4. Run the Kafka producer.

```
[hadoop@172 kafka]$ ./bin/kafka-console-producer.sh --broker-list $kafkaIP:9092
--topic kafka_test
hello
hbase_test
```

Test

Enter information on the client of the Kafka producer and press Enter.

Check whether there is corresponding data in the HBase table.

Reference Documentation

HBaseSink Configuration Description

©2013-2024 Tencent Cloud. All rights reserved.

Kerberos Development Guide
Kerberos Overview

Last updated：2025-01-03 14:56:32

Currently, only EMR v2.1.0 supports the creation of secure clusters, i.e., the open-source components in the cluster are launched in Kerberos secure mode. In this security environment, only authenticated clients can access the services (such as HDFS) of the cluster.

Below lists the components in EMR v2.1.0 that currently supports Kerberos.

Component Name	Version
Hadoop	2.8.4
Hbase	1.3.1
Hive	2.3.3
Hue	4.4.0
Ooize	4.3.1
Presto	0.7.1
Zookeeper	3.4.9

Important Concepts

KDC

Full name: Key distribution center

Role: Generating and managing tickets in the entire authentication process, including two services: AS and TGS.

AS

Full name: Authentication service

Role: Generating TGTs for a client.

TGS

Name: Ticket granting service

Role: Granting a client tickets for a specified service.

AD

Name: Account database

Role: Storing the allowlist of all clients, and only clients in the allowlist can successfully apply for TGTs.

©2013-2024 Tencent Cloud. All rights reserved.

TGT

Name: Ticket-granting ticket

Role: A ticket used to obtaining tickets.

client

A client that wants to access a server.

server

A server that provide a service for a certain business.

Other Concepts

principal

A subject of authentication, i.e., the username.

realm

Realm is a little bit like a namespace in programming languages. In programming languages, a variable name only makes sense in a namespace. Similarly, a principal only makes sense in a realm. Generally, a realm can be seen as a "container" or "space" of a principal.

Correspondingly, the naming rule for principals is what_name_you_like@realm.

In Kerberos, it is customary to use uppercase letters to name a realm, such as EXAMPLE.COM.

password

A password of a user, corresponding to a master_key in Kerberos. It can be stored in a keytab file; therefore, in all scenarios that require passwords in Kerberos, a keytab can be used as the input.

credential

A credential is a proof that "proves that someone is truly of the claimed identity or that something can really happen".

The specific meaning of the credential varies slightly by usage scenario:

For an individual principal, a credential is a password.

In the Kerberos authentication process, a credential means a variety of tickets.

Authentication Process

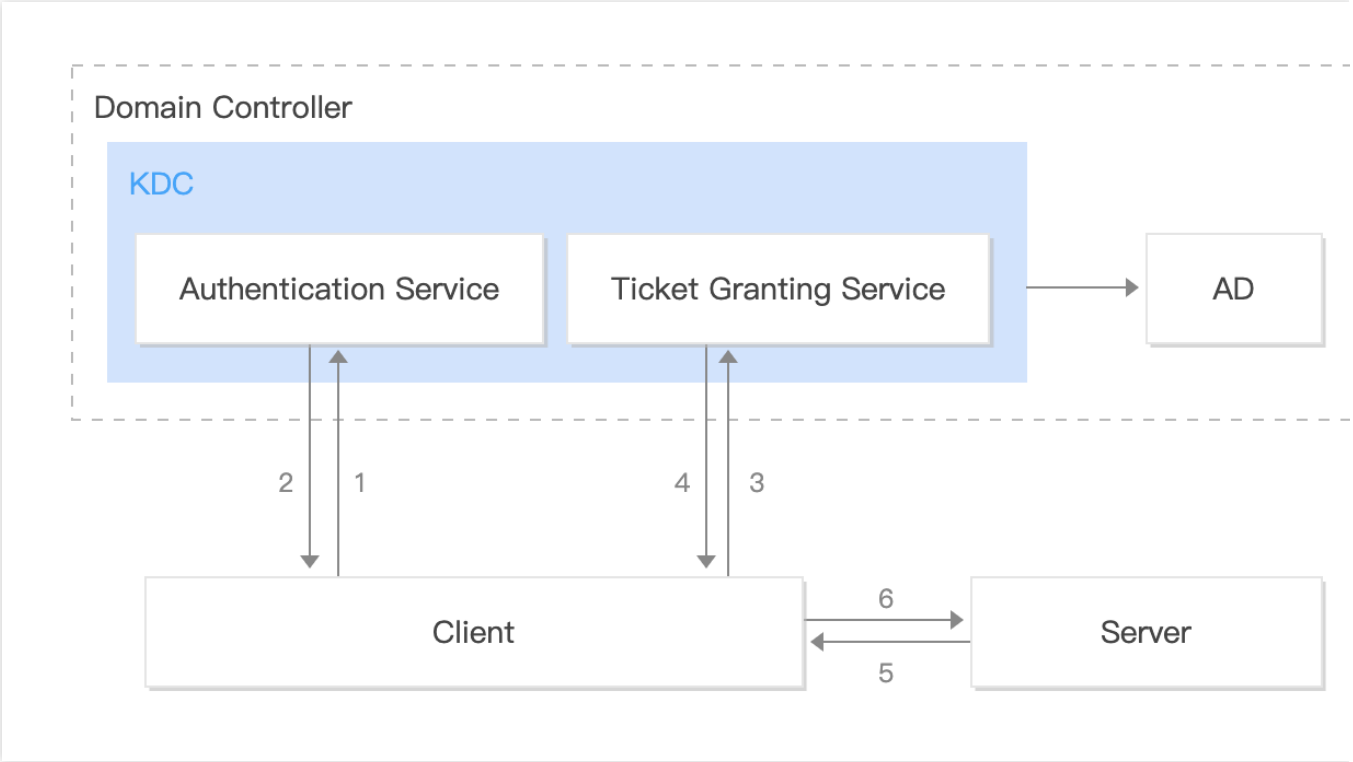
During the process of accessing a server by a client, to ensure that both the client and the server are reliable, it is necessary to introduce a third-party authentication platform. Therefore, two services, AS and TGS, are designed. They are usually in the same service process and provided by a KDC for the implementation of MIT Kerberos.

The authentication process is divided into the following three steps:

1. A client requests access to a server from the Kerberos service. Kerberos first determines whether the client is trustworthy by checking whether it is in the blocklist and allowlist stored in the AD. After success, the AS returns a TGT to the client.

©2013-2024 Tencent Cloud. All rights reserved.

2. After receiving the TGT, the client continues to request access to the server from Kerberos. Kerberos uses the TGT in the message submitted by the client to determine that the client has the permissions, and then grants the server access ticket to the client.
3. After receiving the ticket, the client can access the server, but the ticket is only for the specified server. To access other servers, the client needs to apply to the TGS again.



©2013-2024 Tencent Cloud. All rights reserved.

Knox Development Guide

Knox Development Guide

Last updated：2025-01-03 14:56:32

Currently, EMR v1.3.1 and v2.0.1 support Apache Knox. After completing the following preparations, you can access the web UIs of services such as Yarn and HDFS on the internet.

Preparations

You have signed up for a Tencent Cloud account and created an EMR cluster.
In EMR version 1.3.1 and 2.0.1, Knox is a required component by default when a cluster is created. If you use a legacy version, please contact our customer service to help you install Knox.

Accessing Knox

Access by using the public IP address of the cluster. You are recommended to modify the CVM security group rule of this IP to limit the accessing IP address on the TCP:30002 port to your own IP address.
1. View the public IP address in the cluster details.
2. Access the URL of the corresponding service in a browser.
HDFS UI: https://{public IP address of the cluster}:30002/gateway/emr/hdfs
Yarn UI: https://{public IP address of the cluster}:30002/gateway/emr/yarn
Hive UI: https://{public IP address of the cluster}:30002/gateway/emr/hive
HBase UI: https://{public IP address of the cluster}:30002/gateway/emr/hbase/webui
Hue UI: https://{public IP address of the cluster}:30002/gateway/emr/hue
Storm UI: https://{public IP address of the cluster}:30002/gateway/emr/stormui
Ganglia UI: https://{public IP address of the cluster}:30002/gateway/emr/ganglia/
Presto UI: https://{public IP address of the cluster}:30002/gateway/emr/presto/
Oozie UI: https://{public IP address of the cluster}:30002/gateway/emr/oozie/
3. The browser may display an error message saying that **your connection is not private**. This is because that the Knox service uses a self-signed certificate. Please confirm again that you are accessing the public IP address of your own cluster and the port is 30002. Then, select **Advanced** > **Proceed**.
4. In the login box that pops up, the username is root, and the default password is the one entered when the cluster was created. **It is recommended to change the password by clicking Reset Native UI Password on the page.**

©2013-2024 Tencent Cloud. All rights reserved.

Alluxio Development Guide
Alluxio Development Documentation

Last updated：2025-01-03 14:56:32

Background

Alluxio provides access to data through a filesystem interface. Files in Alluxio offer write-once semantics: they become immutable after they have been written in their entirety and cannot be read before being completed. Alluxio provides two different Filesystem APIs, the Alluxio API and a Hadoop compatible API. The Alluxio API provides additional functionality, while the Hadoop compatible API gives you the flexibility of leveraging Alluxio without having to modify existing code written through Hadoop's API.
All resources with the Alluxio Java API are specified through an AlluxioURI which represents the path to the resource.

Getting a Filesystem Client

To obtain an Alluxio filesystem client in Java code, use:

```
FileSystem fs = FileSystem.Factory.get();
```

Creating a File

All metadata operations as well as opening a file for reading or creating a file for writing are executed through the FileSystem object. Since Alluxio files are immutable once written, the idiomatic way to create files is to use `FileSystem#createFile(AlluxioURI)` , which returns a stream object that can be used to write the file. For example:

```
FileSystem fs = FileSystem.Factory.get();
AlluxioURI path = new AlluxioURI("/myFile");
// Create a file and get its output stream
FileOutStream out = fs.createFile(path);
// Write data
out.write(...);
// Close and complete file
out.close();
```

©2013-2024 Tencent Cloud. All rights reserved.

Specifying Operation Options

For all FileSystem operations, an additional options field may be specified, which allows you to specify non-default settings for the operation. For example:

```
FileSystem fs = FileSystem.Factory.get();
AlluxioURI path = new AlluxioURI("/myFile");
// Generate options to set a custom blocksize of 128 MB
CreateFileOptions options = CreateFileOptions.defaults().setBlockSize(128 *
Constants.MB);
FileOutStream out = fs.createFile(path, options);
```

IO Options

Alluxio uses two different storage types: Alluxio managed storage and under storage. Alluxio managed storage is the memory, SSD, and/or HDD allocated to Alluxio workers. Under storage is the storage resource managed by the underlying storage system, such as S3, Swift, or HDFS. You can specify the interaction with Alluxio managed storage through ReadType and WriteType. ReadType specifies the data read behavior when reading a file. WriteType specifies the data write behavior when writing a new file, e.g., whether the data should be written in Alluxio storage. Below is a table of the expected behaviors of ReadType. Reads will always prefer Alluxio storage over the under storage system.

Read Type	Behavior
CACHE_PROMOTE	Data is moved to the highest tier in the worker where the data was read. If the data was not in the Alluxio storage of the local worker, a replica will be added to the local Alluxio worker. If `alluxio.user.file.cache.partially.read.block` is set to true, data blocks that are not completely read will also be **entirely** stored in Alluxio. In contrast, a data block can be cached only when it is completely read.
CACHE	If the data was not in the Alluxio storage of the local worker, a replica will be added to the local Alluxio worker. If `alluxio.user.file.cache.partially.read.block` is set to true, data blocks that are not completely read will also be **entirely** stored in Alluxio. In contrast, a data block can be cached only when it is completely read.
NO_CACHE	Data is read without storing a replica in Alluxio.

Below is a table of the expected behaviors of WriteType.

Write Type	Behavior

©2013-2024 Tencent Cloud. All rights reserved.

CACHE_THROUGH	Data is written synchronously to an Alluxio worker and the under storage system.
MUST_CACHE	Data is written synchronously to an Alluxio worker. No data will be written to the under storage. This is the default write type.
THROUGH	Data is written synchronously to the under storage. No data will be written to Alluxio.
ASYNC_THROUGH	Data is written synchronously to an Alluxio worker and asynchronously to the under storage system. This is experimental.

Location Policy

Alluxio provides location policy to choose which workers to store the blocks of a file.
Using Alluxio's Java API, you can set the policy in CreateFileOptions for writing files into Alluxio and OpenFileOptions for reading files from Alluxio.
You can simply override the default policy class in the configuration file at property
`alluxio.user.file.write.location.policy.class. The built-in policies include:
LocalFirstPolicy (alluxio.client.file.policy.LocalFirstPolicy)
It returns the local node first, and if the local worker doesn't have enough capacity of a block, it randomly selects a worker from the active workers list. This is the default policy.
MostAvailableFirstPolicy (alluxio.client.file.policy.MostAvailableFirstPolicy)
It returns the worker with the most available bytes.
RoundRobinPolicy (alluxio.client.file.policy.RoundRobinPolicy)
It chooses the worker for the next block in a round-robin manner and skips workers that do not have enough capacity.
SpecificHostPolicy (alluxio.client.file.policy.SpecificHostPolicy)
It returns a worker with the specified node name. This policy cannot be set as default policy.
Alluxio supports custom policies, so you can also develop your own policy appropriate for your workload by implementing interface `alluxio.client.file.policyFileWriteLocationPolicy` .

Note:

The default policy must have an empty constructor. To use the ASYNC_THROUGH write type, all the blocks of a file must be written to the same worker.
Alluxio allows a client to select a tier preference when writing blocks to a local worker. Currently, this policy preference exists only for local workers but not remote workers; remote workers will write blocks to the highest tier.
By default, data is written to the top tier. You can modify the default setting through the `alluxio.user.file.write.tier.default` configuration property or override it through an option to the `FileSystem#createFile(AlluxioURI)API` call.
All operations on existing files or directories require you to specify the AlluxioURI. With the AlluxioURI, you may use any of the methods of FileSystem to access the resource.

©2013-2024 Tencent Cloud. All rights reserved.

An AlluxioURI can be used to perform Alluxio FileSystem operations, such as modifying the file metadata, ttl, or pin state, or getting an input stream to read the file. For example, to read a file:

```
FileSystem fs = FileSystem.Factory.get();
AlluxioURI path = new AlluxioURI("/myFile");
// Open the file for reading
FileInStream in = fs.openFile(path);
// Read data
in.read(...);
// Close file relinquishing the lock
in.close();
```

©2013-2024 Tencent Cloud. All rights reserved.

Common Alluxio Commands

Last updated：2025-01-03 14:56:32

Operation	Syntax	Description
cat	cat "path"	Print the content of a file in Alluxio to the console.
checkConsistency	checkConsistency "path"	Check the metadata consistency between Alluxio and the under storage.
checksum	checksum "path"	Calculate the md5 checksum for a file.
chgrp	chgrp "group" "path"	Change the group of a file or directory in Alluxio.
chmod	chmod "permission" "path"	Change the permission of the directory or file in Alluxio.
chown	chown "owner" "path"	Change the owner of a file or directory in Alluxio.
copyFromLocal	copyFromLocal "source path""remote path"	Copy the file specified by "source path" to the path in Alluxio specified by "remote path". This command will fail if "remote path" already exists.
copyToLocal	copyToLocal "remote path" "local path"	Copy the file specified by "remote path" in Alluxio to a local destination.
count	count "path"	Display the number of folders and files matching the specified prefix in "path".
cp	cp "src" "dst"	Copy a file or directory within the Alluxio file system.
du	du "path"	Display the size of the file or directory specified by the input path.
fileInfo	fileInfo "path"	Print the information of the blocks of a specified file.
free	free "path"	Free a file or all files under a directory from Alluxio. If the file/directory is also in under storage, it will still be available there.
getCapacityBytes	getCapacityBytes	Get the capacity of the Alluxio file system.
getUsedBytes	getUsedBytes	Get the number of bytes used in the Alluxio file system.
help	help "cmd"	Print help information for the given command. If no command is given, print help information for all supported commands.

©2013-2024 Tencent Cloud. All rights reserved.

leader	leader	Print the current Alluxio leader master node name.
load	load "path"	Load the data of a file or a directory from under storage into Alluxio.
loadMetadata	loadMetadata "path"	Load the metadata of a file or directory from under storage into Alluxio.
location	location "path"	Output a list of node that have the specified file data.
ls	ls "path"	List all the files and directories directly under the given path with information such as size.
masterInfo	masterInfo	Print information regarding Alluxio master fault tolerance such as leader address, list of master addresses, and configured Zookeeper address.
mkdir	mkdir "path1" ... "pathn"	Create one or more directories under the given paths, along with any necessary parent directories. Multiple paths are separated by spaces or tabs. This command will fail if any of the given paths already exist.
mount	mount "path" "uri"	Mount the underlying file system path "uri" into the Alluxio namespace as "path". The "path" is assumed not to exist and is created by the operation. No data or metadata is loaded from under storage into Alluxio. After a path is mounted, operations on objects under the mounted path are mirror to the mounted under storage.
mv	mv "source" "destination"	Move a file or directory specified by "source" to a new location "destination". This command will fail if "destination" already exists.
persist	persist "path1" ... "pathn"	Persist files or directories currently stored only in Alluxio to the underlying file system.
pin	pin "path"	Pin the given file to avoid evicting it from memory. If the given path is a directory, it recursively pins all the files contained and any new files created within this directory.
report	report "path"	Report to the master that a file is lost.
rm	rm "path"	Remove a file. This command will fail if the given path is a directory rather than a file.
setTtl	setTtl "path" "time"	Set the TTL (time to live) in milliseconds for a file.
stat	stat "path"	Display information of the specified file or directory.

©2013-2024 Tencent Cloud. All rights reserved.

tail	tail "path"	Print the last 1 KB of the specified file to the console.
test	test "path"	Test a property of a path, returning 0 if the property is true, or 1 otherwise.
touch	touch "path"	Create a 0-byte file at the specified location.
unmount	unmount "path"	Unmount the underlying file system path mounted in the Alluxio namespace as "path". Alluxio objects under "path" are removed from Alluxio, but they still exist in the previously mounted under storage.
unpin	unpin "path"	Unpin the given file to allow Alluxio to evict this file again. If the given path is a directory, it recursively unpins all files contained and any new files created within this directory.
unsetTtl	unsetTtl "path"	Remove the TTL (time to live) setting from a file.

cat

Background

The cat command prints the entire contents of a file in Alluxio to the console. This can be useful for verifying the file. Use the copyToLocal command to copy the file to your local file system.

Operation example

When trying out a new computation job, cat can be used as a quick way to check the output:

```
$ ./bin/alluxio fs cat /output/part-00000
```

checkConsistency

Background

The checkConsistency command compares Alluxio and under storage metadata for a given path. If the path is a directory, the entire subtree will be compared. The command returns a message listing each inconsistent file or directory. The system administrator should reconcile the differences of these files at their discretion. To avoid metadata inconsistencies between Alluxio and under storages, design your systems to modify files and directories through the Alluxio and avoid directly modifying state in the underlying storage.
If the -r option is used, the checkConsistency command will repair all inconsistent files and directories under the given path. If an inconsistent file or directory exists only in under storage, its metadata will be added to Alluxio. If an

©2013-2024 Tencent Cloud. All rights reserved.

inconsistent file exists in Alluxio and its data is fully present in Alluxio, its metadata will be loaded to Alluxio again.

Note:

This command requires a read lock on the subtree being checked, meaning writes and updates to files or directories in the subtree cannot be completed until this command completes.

Operation example

checkConsistency can be used to periodically validate the integrity of the namespace.
List each inconsistent file or directory:

```
$ ./bin/alluxio fs checkConsistency /
```

Repair the inconsistent files or directories:

```
$ ./bin/alluxio fs checkConsistency -r /
```

checksum

Background

The checksum command outputs the md5 value of a file in Alluxio.

Operation example

checksum can be used to verify the content of a file stored in Alluxio matches the content stored in an under file system or local file system.

```
$ ./bin/alluxio fs checksum /LICENSE
```

chgrp

Background

The chgrp command changes the group of the file or directory in Alluxio. Alluxio supports file authorization with Posix file permission. Group is an authorizable entity in Posix file permission model. The file owner or super-user can execute this command to change the group of the file or directory.
Adding -R option also changes the group of child file and child directory recursively.

Operation example

chgrp can be used as a quick way to change the group of file:

```
$ ./bin/alluxio fs chgrp alluxio-group-new /input/file1
```

©2013-2024 Tencent Cloud. All rights reserved.

chmod

Background

The chmod command changes the permission of file or directory in Alluxio. Currently, octal mode is supported: the numerical format accepts three octal digits which refer to permissions for the file owner, the group and other users. Here is the number-permission mapping table:

Number	Permission	rwx
7	read, write and execute	rwx
6	read and write	rw-
5	read and execute	r-x
4	read only	r--
3	write and execute	-wx
2	write only	-w-
1	execute only	--x
0	none	---

Adding -R option also changes the permission of child file and child directory recursively.

Operation example

chmod can be used as a quick way to change the permission of file:

```
$ ./bin/alluxio fs chmod 755 /input/file1
```

chown

Background

The chown command changes the owner of the file or directory in Alluxio. For obvious security reasons, the ownership of a file can only be altered by a super-user.
Adding -R option also changes the owner of child file and child directory recursively.

Sample

©2013-2024 Tencent Cloud. All rights reserved.

chown can be used as a quick way to change the owner of file:

```
$ ./bin/alluxio fs chown alluxio-user /input/file1
```

copyFromLocal

Background

The copyFromLocal command copies the contents of a file in your local file system into Alluxio. If the node you run the command from has an Alluxio worker, the data will be available on that worker. Otherwise, the data will be copied to a random remote node running an Alluxio worker. If a directory is specified, the directory and all its contents will be copied recursively.

Operation example

copyFromLocal can be used as a quick way to inject data into the system for processing:

```
$ ./bin/alluxio fs copyFromLocal /local/data /input
```

copyToLocal

Background

The copyToLocal command copies the contents of a file in Alluxio to a file in your local file system. If a directory is specified, the directory and all its contents will be downloaded recursively.

Operation example

copyToLocal can be used as a quick way to download output data for additional investigation or debugging.

```
$ ./bin/alluxio fs copyToLocal /output/part-00000 part-00000
```

count

Background

The count command outputs the number of files and folders matching a prefix as well as the total size of the files. count works recursively and accounts for any nested directories and files. count is best utilized when you have some predefined naming conventions for their files.

Operation example

©2013-2024 Tencent Cloud. All rights reserved.

If data files are stored by their date, count can be used to determine the number of data files and their total size for any date, month, or year.

```
$ ./bin/alluxio fs count /data/2014
```

cp

Background

The cp command copies a file or directory in the Alluxio file system or between local file system and Alluxio file system. Scheme file indicates the local file system and scheme alluxio or no scheme indicates the Alluxio file system.
If the -R option is used and the source designates a directory, cp copies the entire subtree at source to the destination.

Operation example

cp can be used to copy files between Under file systems.

```
$ ./bin/alluxio fs cp /hdfs/file1 /s3/
```

du

Background

The du command outputs the size of a file. If a directory is specified, it will output the aggregate size of all files in the directory and its children directories.

Operation example

If the Alluxio space is unexpectedly over utilized, du can be used to detect which folders are taking up the most space.

```
$ ./bin/alluxio fs du /\\\\*
```

fileInfo

Background

The fileInfo command is deprecated since Alluxio version 1.5. Please use the stat command instead.
The fileInfo command dumps the FileInfo representation of a file to the console. It is primarily intended to assist users in debugging their system. Generally, viewing the file information in the UI will be much easier to understand.

Operation example

©2013-2024 Tencent Cloud. All rights reserved.

fileInfo can be used to debug the block locations of a file. This is useful when trying to achieve locality for compute workloads.

```
$ ./bin/alluxio fs fileInfo /data/2015/logs-1.txt
```

free

Background

The free command sends a request to the master to evict all blocks of a file from the Alluxio workers. If the argument to free is a directory, it will recursively free all files. This request is not guaranteed to take effect immediately, as readers may be currently by using the blocks of the file. free will return immediately after the request is acknowledged by the master. Note that, files must be persisted already in under storage before being freed, or the free command will fail; also any pinned files cannot be freed unless -f option is specified. The free command does not delete any data from the under storage system, but only removing the blocks of those files in Alluxio space to reclaim space. In addition, metadata will not be affected by this operation, meaning the freed file will still show up if an ls command is run.

Operation example

free can be used to manually manage Alluxio's data caching.

```
$ ./bin/alluxio fs free /unused/data
```

getCapacityBytes

Background

The getCapacityBytes command returns the maximum number of bytes Alluxio is configured to store.

Operation example

getCapacityBytes can be used to verify if your cluster is set up as expected.

```
$ ./bin/alluxio fs getCapacityBytes
```

getUsedBytes

Background

©2013-2024 Tencent Cloud. All rights reserved.

The getUsedBytes command returns the number of used bytes in Alluxio.

Operation example

getUsedBytes can be used to monitor the health of your cluster.

```
$ ./bin/alluxio fs getUsedBytes
```

leader

Background

The `leader` command prints the current Alluxio leader master node name.

Operation example

```
$ ./bin/alluxio fs leader
```

load

Background

The load command moves data from the under storage system into Alluxio storage. If there is an Alluxio worker on the machine this command is run from, the data will be loaded to that worker. Otherwise, a random worker will be selected to serve the data. If the data is already loaded into Alluxio, load is a no-op. If load is run on a directory, files in the directory will be recursively loaded.

Operation example

load can be used to prefetch data for analytics jobs.

```
$ ./bin/alluxio fs load /data/today
```

loadMetadata

Background

The loadMetadata command queries the under storage system for any file or directory matching the given path and then creates a mirror of the file in Alluxio backed by that file. Only the metadata, such as the file name and size, are loaded this way and no data transfer occurs.

©2013-2024 Tencent Cloud. All rights reserved.

Operation example

loadMetadata can be used when other systems output to the under storage directly (bypassing Alluxio), and the application running on Alluxio needs to use the output of those systems.

```
$ ./bin/alluxio fs loadMetadata /hdfs/data/2015/logs-1.txt
```

location

Background

The location command returns the addresses of all the Alluxio workers which contain blocks belonging to the given file.

Operation example

location can be used to debug data locality when running jobs by using a compute framework.

```
$ ./bin/alluxio fs location /data/2015/logs-1.txt
```

ls

Background

The ls command lists all the immediate children in a directory and displays the file size, last modification time, and in memory status of the files. Using ls on a file will only display the information for that specific file. The ls command will also load the metadata for any file or immediate children of a directory from the under storage system to Alluxio namespace, if it does not exist in Alluxio yet. ls queries the under storage system for any file or directory matching the given path and then creates a mirror of the file in Alluxio backed by that file. Only the metadata, such as the file name and size are loaded this way and no data transfer occurs.

Options:

-d option lists the directories as plain files. For example, `ls -d /` shows the attributes of root directory.

f option forces loading metadata for immediate children in a directory. By default, it loads metadata only at the first time at which a directory is listed.

-h option displays file sizes in human-readable formats.

-p option lists all pinned files.

-R option also recursively lists child directories, displaying the entire subtree starting from the input path.

Operation example

ls can be used to browse the file system.

©2013-2024 Tencent Cloud. All rights reserved.

```
$ ./bin/alluxio fs mount /cos/data cosn://data-bucket/
```

Verify:

```
$ ./bin/alluxio fs ls /s3/data/
```

masterInfo

Background

The masterInfo command prints information regarding master fault tolerance such as leader address, list of master addresses, and the configured Zookeeper address. If Alluxio is running in single master mode, masterInfo will print the master address. If Alluxio is running in fault tolerance mode, the leader address, list of master addresses and the configured Zookeeper address will be printed.

Operation example

masterInfo can be used to print information regarding master fault tolerance.

```
$ ./bin/alluxio fs masterInfo
```

mkdir

Background

The mkdir command creates a new directory in Alluxio space. It is recursive and will create any nonexistent parent directories. Note that the created directory will not be created in the under storage system until a file in the directory is persisted to the underlying storage. Using mkdir on an invalid or already existing path will fail.

Operation example

mkdir can be used by an admin to set up the basic folder structures.

```
$ ./bin/alluxio fs mkdir /users
$ ./bin/alluxio fs mkdir /users/Alice
$ ./bin/alluxio fs mkdir /users/Bob
```

mount

Background

©2013-2024 Tencent Cloud. All rights reserved.

The mount command links an under storage path to an Alluxio path, and files and folders created in Alluxio space under the path will be backed by a corresponding file or folder in the under storage path. For more details, please see Unified Namespace.
Options:
--readonly option sets the mount point to be readonly in Alluxio
--option `<key>=<val>` option passes a property to this mount point (e.g., S3 credential)

Operation example

mount can be used to make data in another storage system available in Alluxio.

```
$ ./bin/alluxio fs mount /mnt/hdfs hdfs://host1:9000/data/
```

mv

Background

The mv command moves a file or directory to another path in Alluxio. The destination path must not exist or be a directory. If it is a directory, the file or directory will be placed as a child of the directory. mv is purely a metadata operation and does not affect the data blocks of the file. mv cannot be done between mount points of different under storage systems.

Operation example

mv can be used to move older data into a non working directory.

```
$ ./bin/alluxio fs mv /data/2014 /data/archives/2014
```

persist

Background

The persist command persists data in Alluxio storage into the under storage system. This is a data operation and will take time depending on how large the file is. After persist is complete, the file in Alluxio will be backed by the file in the under storage, make it still valid if the Alluxio blocks are evicted or otherwise lost.

Operation example

persist can be used after filtering a series of temporary files for the ones containing useful data.

```
$ ./bin/alluxio fs persist /tmp/experimental-logs-2.txt
```

©2013-2024 Tencent Cloud. All rights reserved.

pin

Background

The pin command marks a file or folder as pinned in Alluxio. This is a metadata operation and will not cause any data to be loaded into Alluxio. If a file is pinned, any blocks belonging to the file will never be evicted from an Alluxio worker. If there are too many pinned files, Alluxio workers may run low on storage space preventing other files from being cached.

Operation example

pin can be used to manually ensure performance if the administrator understands the workloads well.

```
$ ./bin/alluxio fs pin /data/today
```

report

Background

The report command marks a file as lost to the Alluxio master. This command should only be used with files created by using the Lineage API. Marking a file as lost will cause the master to schedule a recomputation job to regenerate the file.

Operation example

report can be used to force recomputation of a file.

```
$ ./bin/alluxio fs report /tmp/lineage-file
```

rm

Background

The rm command removes a file from Alluxio space and the under storage system. The file will be unavailable immediately after this command returns, but the actual data may be deleted a while later.
Add -R option will delete all contents of the directory and then the directory itself. Add -U option to not check whether the UFS contents being deleted are in-sync with Alluxio before attempting to delete persisted directories.

Operation example

rm can be used to remove temporary files which are no longer needed.

©2013-2024 Tencent Cloud. All rights reserved.

```
$ ./bin/alluxio fs rm /tmp/unused-file
```

setTtl

Background

The setTtl command sets the time-to-live of a file or a directory, in milliseconds. If set ttl to a directory, all the children inside that directory will set too. So a directory's TTL expires, all the children inside that directory will also expire. Action parameter will indicate the action to perform once the current time is greater than the TTL + creation time of the file. Action delete (default) will delete file or directory from both Alluxio and the under storage system, whereas action free will just free the file from Alluxio even they are pinned.

Operation example

setTtl with action delete can be used to clean up files the administrator knows are unnecessary after a period of time, or with action free just remove the contents from Alluxio to make room for more space in Alluxio.

```
$ ./bin/alluxio fs setTtl -action free /data/good-for-one-day 86400000
```

stat

Background

The stat command dumps the FileInfo representation of a file or a directory to the console. It is primarily intended to assist users in debugging their system. Generally, viewing the file information in the UI will be much easier to understand.
You can specify -f to display information in a given format:
"%N": name of the file
"%z": size of the file in bytes
"%u": owner
"%g": group name of the owner
"%y" or "%Y": modification time, `%y shows 'yyyy-MM-dd HH:mm:ss' (the UTC date), %Y` shows milliseconds since January 1, 1970 UTC
"%b": number of blocks allocated for the file

Operation example

stat can be used to debug the block locations of a file. This is useful when trying to achieve locality for compute workloads.

©2013-2024 Tencent Cloud. All rights reserved.

```
$ ./bin/alluxio fs stat /data/2015/logs-1.txt
$ ./bin/alluxio fs stat /data/2015
$ ./bin/alluxio fs stat -f %z /data/2015/logs-1.txt
```

tail

Background

The tail command outputs the last 1 KB of data in a file to the console.

Operation example

tail can be used to verify the output of a job is in the expected format or contains expected values.

```
$ ./bin/alluxio fs tail /output/part-00000
```

test

Background

The test command tests a property of a path, returning 0 if the property is true, or 1 otherwise.
Options:
-d option tests whether the path is a directory.
-e option tests whether the path exists.
-f option tests whether the path is a file.
-s option tests whether the directory is empty.
-z option tests whether the file is zero length.

Operation example

```
$ ./bin/alluxio fs test -d /someDir
```

touch

Background

The touch command creates a 0-byte file. Files created with touch cannot be overwritten and are mostly useful as flags.

Operation example

©2013-2024 Tencent Cloud. All rights reserved.

touch can be used to create a file signifying the completion of analysis on a directory.

```
$ ./bin/alluxio fs touch /data/yesterday/_DONE_
```

unmount

Background

The unmount command disassociates an Alluxio path with an under storage directory. Alluxio metadata for the mount point will be removed along with any data blocks, but the under storage system will retain all metadata and data. See Unified Namespace for more details.

Operation example

unmount can be used to remove an under storage system when you no longer need data from that system.

```
$ ./bin/alluxio fs unmount /s3/data
```

unpin

Background

The unpin command unmarks a file or directory in Alluxio as pinned. This is a metadata operation and will not evict or delete any data blocks. Once a file is unpinned, its data blocks can be evicted from various Alluxio workers containing the block.

Operation example

unpin can be used when the administrator knows there is a change in the data access pattern.

```
$ ./bin/alluxio fs unpin /data/yesterday/join-table
```

unsetTtl

Background

The unsetTtl command will remove the TTL of a file in Alluxio. This is a metadata operation and will not evict or store blocks in Alluxio. The TTL of a file can later be reset with setTtl.

Operation example

©2013-2024 Tencent Cloud. All rights reserved.

unsetTtl can be used if a regularly managed file requires manual management due to some special case.

```
$ ./bin/alluxio fs unsetTtl /data/yesterday/data-not-yet-analyzed
```

©2013-2024 Tencent Cloud. All rights reserved.

Mounting File System to Unified Alluxio File System

Last updated：2025-01-03 14:56:32

Background

Alluxio provides a unified namespace mechanism that allows other file systems to be mounted to the file system of Alluxio. In addition, it allows upper-layer applications to use the unified namespace to access data scattered across different systems.

Mounting COS

Sample: Mounting a COS bucket to an Alluxio directory

```
bin/alluxio fs mount --option fs.cos.access.key=<COS_SECRET_ID> \\
   --option fs.cos.secret.key=<COS_SECRET_KEY> \\
   --option fs.cos.region=<COS_REGION> \\
   --option fs.cos.app.id=<COS_APP_ID> \\
   /cos cos://<COS_BUCKET>/
```

Configure the COS information in each `--option` .

Configuration Item	Description
fs.cos.access.key	The COS secret ID
fs.cos.secret.key	The COS secret key
fs.cos.region	The COS region name, such as `ap-beijing`
fs.cos.app.id	Your `AppID`
COS_BUCKET	The COS bucket name **without the `AppID` suffix**

This command mounts the COS directory specified by `cos://bucket/xxx` to the `/cos` directory in Alluxio.

Mounting HDFS

Sample: Mounting an HDFS directory to an Alluxio directory

```
`bin/alluxio fs mount /hdfs hdfs://data`
```

©2013-2024 Tencent Cloud. All rights reserved.

This command mounts the `/data` directory of HDFS to the `/hdfs` subdirectory of Alluxio.

After the mount is successful, the mounted content can be viewed by running the `alluxio fs ls` command.

Mounting CHDFS

Sample: Mounting CHDFS to Alluxio through `mount`

Note

This is supported only for EMR 2.5.0 or later + Alluxio 2.3.0 or later.

```
alluxio fs mount   \\
--option
alluxio.underfs.hdfs.configuration=/usr/local/service/hadoop/etc/hadoop/core-
site.xml  \\
/chdfs ofs://f4modr7kmvw-wMqw.chdfs.ap-chongqing.myqcloud.com
```

©2013-2024 Tencent Cloud. All rights reserved.

Using Alluxio in Tencent Cloud

Last updated：2025-01-03 14:56:32

Overview

Tencent Cloud EMR comes with the ready-to-use Alluxio service, helping you accelerate distributed memory-level caching and simplify data management. You can also use the configuration delivery feature to configure multi-level caching and manage metadata via the EMR console or APIs. In addition, EMR offers one-stop monitoring and alarming.

Preparations

Tencent Cloud EMR Hadoop Standard v2.1.0 or above.
For specific Alluxio versions supported in EMR, see Component Version.

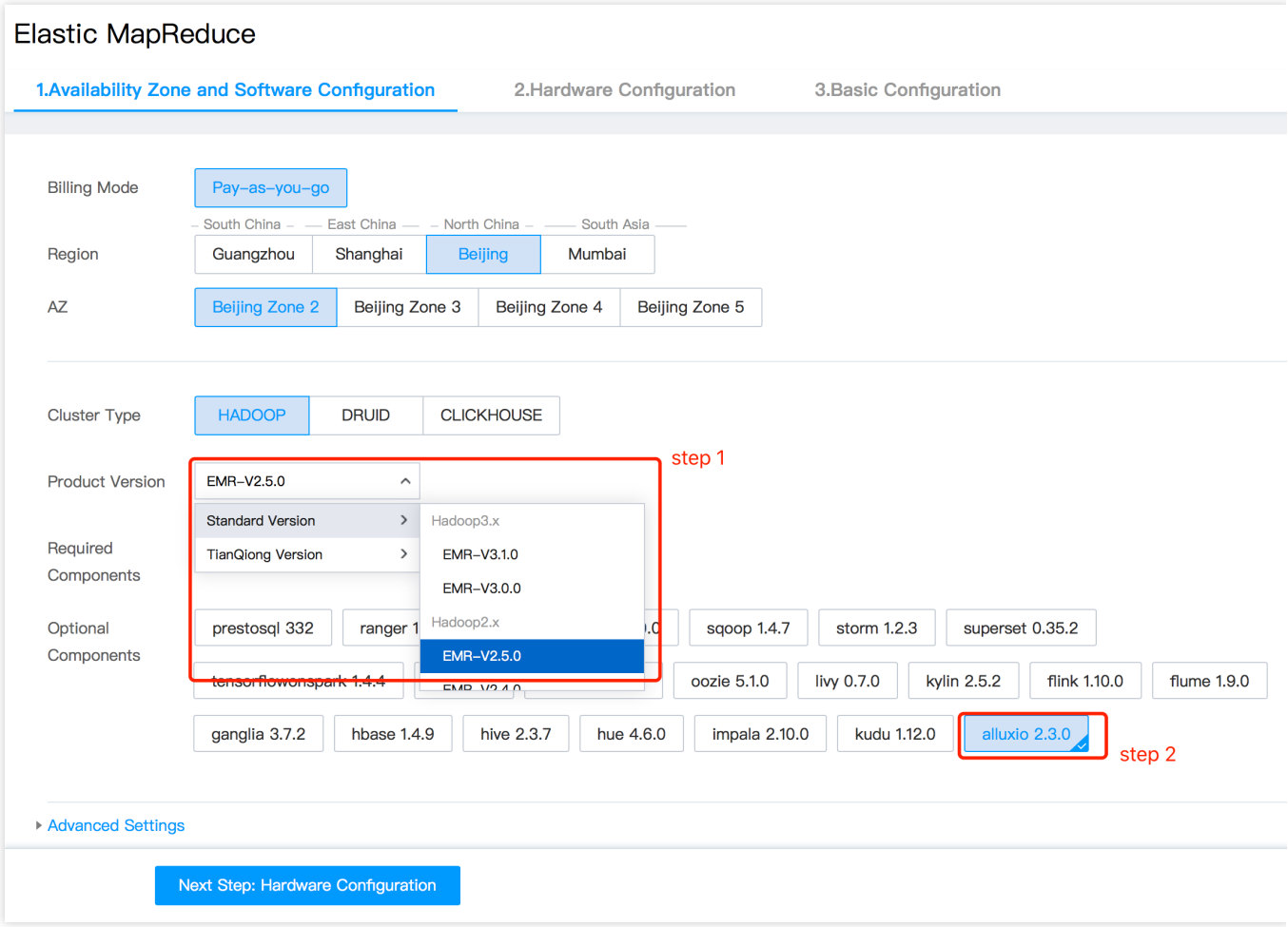
Creating an Alluxio-based EMR Cluster

This section describes how to create a ready-to-use Alluxio-based EMR cluster. You can create an EMR cluster via the purchase page or API.

Creating a cluster via the purchase page

Go to the EMR purchase page, choose an Alluxio-supported version, and select the Alluxio component in **Optional Components**.

©2013-2024 Tencent Cloud. All rights reserved.



Select other options as needed to meet your business needs. For reference, see Creating EMR Cluster.

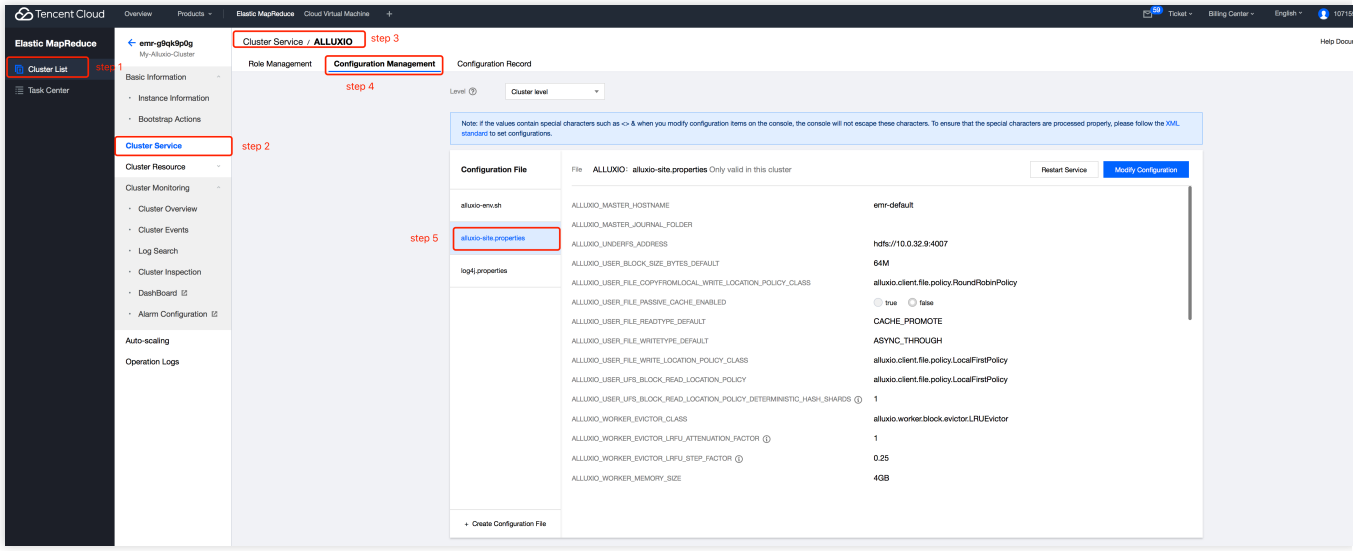
Creating a cluster via API

Tencent Cloud EMR also allows you to build a big data cluster based on Alluxio. For details, see DescribeClusterNodes.

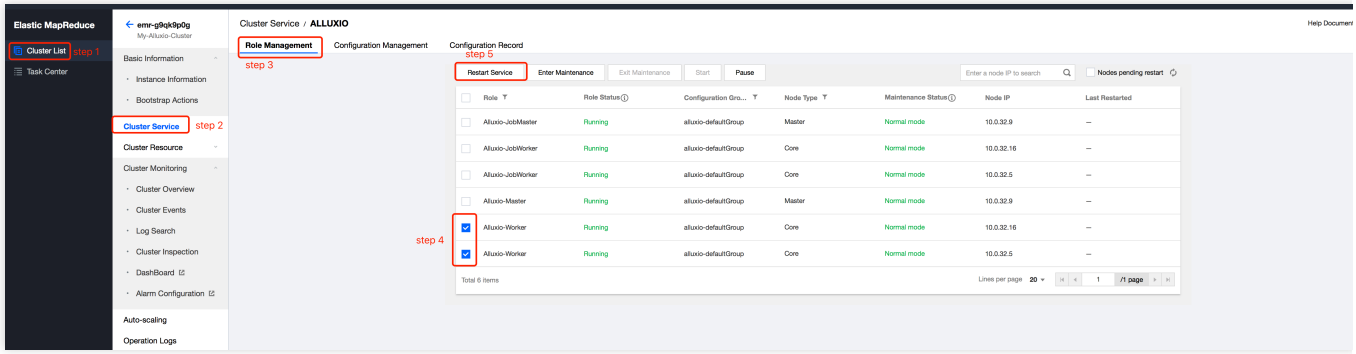
Basic Configurations

When you create an EMR cluster containing the Alluxio component, HDFS will be mounted to Alluxio and memory will be used for single-level (level 0) storage by default. You can use the configuration delivery feature to change the storage mode to multi-level storage or make other optimizations.

©2013-2024 Tencent Cloud. All rights reserved.



After delivering configurations, you need to restart the Alluxio service for some configurations to take effect.



For more details on configuration delivery and restarting policies, see Configuration Management and Restarting Services.

Storage and compute separation based on Alluxio acceleration

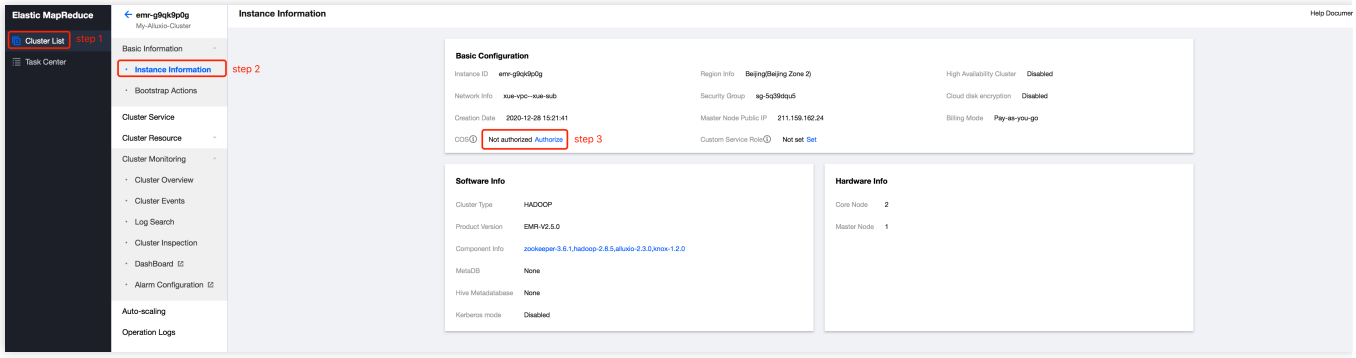
Tencent Cloud EMR provides the compute and storage separation capability based on Tencent Cloud COS. By default, when directly accessing the data in COS, applications do not have node-level data locality or cross-application caching. Alluxio acceleration helps alleviate these issues.

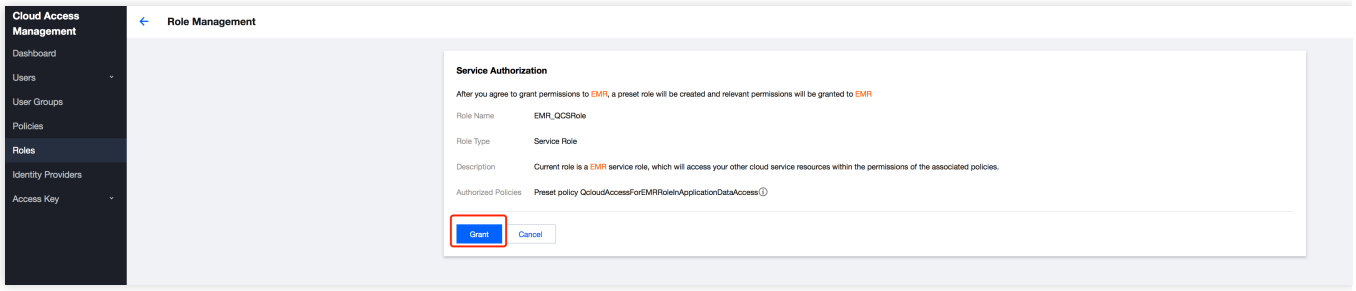
COS is deployed on Tencent Cloud EMR clusters by default and serves as the dependent JAR package of UFS. You only need to grant EMR clusters the permission to access COS and mount COS to Alluxio.

Authorization

©2013-2024 Tencent Cloud. All rights reserved.

If COS is not enabled for the current cluster, you can go to **CAM console > Roles** to grant permission. After authorization, EMR nodes can access the data in COS using temporary keys.





Mounting

Log in to any machine of EMR and mount COS to Alluxio.

```
bin/alluxio fs mount <alluxio-path> <source-path>
//TODO,
```

For more information on using Alluxio in Tencent Cloud EMR, see Alluxio Development Documentation.

©2013-2024 Tencent Cloud. All rights reserved.

Support for COS Transparent-URI

Last updated：2025-01-03 14:56:32

Alluxio users often have existing applications accessing their existing storage systems (Under-FileSystem). Alluxio can be added to existing ecosystems, but one thing that must always be changed is the URI used by the application. The Transparent-URI feature allows users to access their existing storage systems without changing URIs at the application level.

Supported Versions and URI Configuration

1. Supported service component version: Alluxio 2.8.0.
2. Product version: Hadoop 3.x Standard EMR 3.4.0.
3. In order to use Alluxio Transparent-URI, a new Hadoop compatible file system client implementation should be configured. This new ShimFileSystem will replace existing FileSystem whenever the client is configured for receiving foreign URIs. The APIs of Hadoop FileSystem (a Hadoop compatible compute framework) define the mapping from FileSystem scheme to FileSystem implementation. To configure ShimFileSystem, make sure that the following items are configured in the `core-site.xml` file:

Configuration Item	Value
fs.cosn.impl	alluxio.hadoop.ShimFileSystem

4. In order to be compatible with the Transparent-URI schema, Alluxio needs to perform schema conversion. Therefore, make sure that the following items are configured in the `alluxio-site.properties` file:

Configuration Item	Value
alluxio.master.uri.translator.impl	alluxio.master.file.uritranslator.AutoMountUriTranslator
alluxio.user.shimfs.bypass.ufs.impl.list	fs.cosn.impl:org.apache.hadoop.fs.cosnative.NativeCosFileSystem

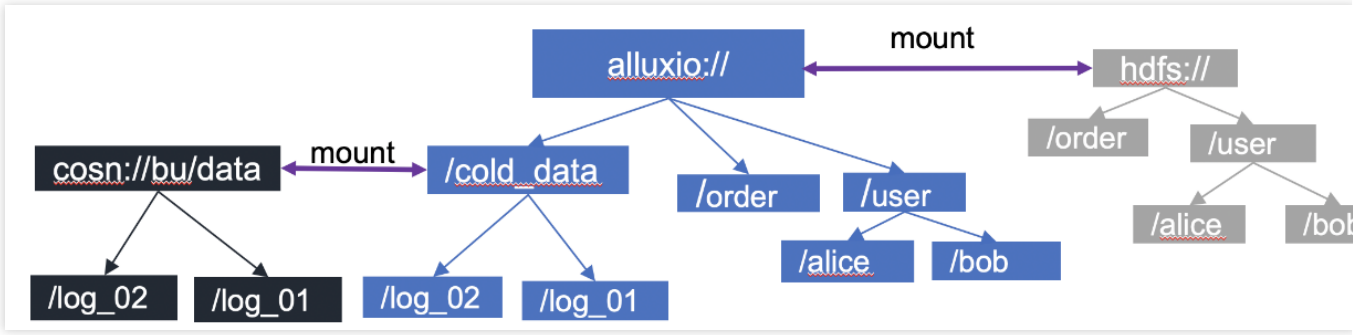
Note

You need to restart the Alluxio service after modifying the configuration in `alluxio-site.properties`.
Once ShimFileSystem is configured, master will need to route URIs that are native to external storage system, to Alluxio namespace. This requires COSN to have been mounted to Alluxio namespace.
To disable the Transparent-URI feature, roll back the `fs.cosn.impl` configuration item in `core-site.xml`.

©2013-2024 Tencent Cloud. All rights reserved.

Mounting

The `mount` command is one of Alluxio's most distinctive commands. It is similar to Linux's `mount` command, through which users can load disks, SSDs and other storage devices to the local file system of Linux. In Alluxio, the concept of mounting is further extended to the distributed system level, that is, users can mount one or more other storage systems/cloud storage services (such as HDFS and COS) to the Alluxio distributed file system using the `mount` command. So they can run distributed applications on Alluxio, such as Spark, Presto, and MapReduce, without any adaptation or even knowledge of the specific data access protocol and path. Users only need to know the path corresponding to the data in the Alluxio file system. It greatly facilitates application development and maintenance.



EMR-Alluxio uses HDFS as the root directory mount point by default

Beginning from EMR-Alluxio v2.5.1, Alluxio's UFS supports the COSN protocol. COS UFS has relatively poor read/write performance and is unstable. In order to solve this issue, the community provides the underlying COSN UFS. Both COS UFS and COSN UFS are used to access Tencent Cloud COS. COSN UFS is a great improvement compared with COS UFS. Its read/write performance are twice as good as that of COS UFS, and it has better stability. Therefore, COSN UFS is strongly recommended. The maintenance for COS UFS will stop after EMR-Alluxio v2.6.0.

Example of mounting COSN:

```
  alluxio fs mount --option fs.cosn.userinfo.secretId=xx \\
        --option fs.cosn.userinfo.secretKey=xx \\
        --option fs.cosn.bucket.region=ap-xx \\
        --option
fs.cosn.impl=org.apache.hadoop.fs.cosnative.NativeCosFileSystem \\
        --option fs.AbstractFileSystem.cosn.impl=org.apache.hadoop.fs.CosN \\
        --option fs.cosn.userinfo.appid=xx \\
        /cosn cosn://COS_BUCKET/path
```

Configure the COS information in each `--option` .

Configuration Item	Description

©2013-2024 Tencent Cloud. All rights reserved.

fs.cosn.userinfo.secretId	COS secret ID
fs.cosn.userinfo.secretKey	COS secret key
fs.cosn.impl	Fixed value: `org.apache.hadoop.fs.CosFileSystem`.
fs.AbstractFileSystem.cosn.impl	Fixed value: `org.apache.hadoop.fs.CosN`.
fs.cosn.bucket.region cos region	Region name such as `ap-beijing`
fs.cosn.userinfo.appid	The AppID of root account
COS_BUCKET COS BUCKET	Bucket name without the AppID suffix

©2013-2024 Tencent Cloud. All rights reserved.

Support for Authentication

Last updated：2025-01-03 14:56:32

Authentication is not required when Alluxio users access data from COS, HDFS, or CHDFS in the existing unified namespace or access the data cached in Alluxio through Transparent-URI; that is, any user can get the data as long as they get the URI. In view of this, EMR-Alluxio improves authentication based on Ranger and COSRanger.

Note

To configure the authentication feature, make sure that the cluster is integrated with the following components:
If only HDFS is mounted to Alluxio, you need to integrate the Ranger component.
If COS and CHDFS are mounted to Alluxio, you need to integrate the COSRanger component.

Supported Versions

Supported service component version: Alluxio v2.8.0.
Product version: Hadoop 3.x Standard EMR v3.4.0.

Configuring Authentication

Prerequisite configuration

```
# Add the `ranger-hive-security.xml` configuration item in the Hive component
ranger.plugin.hive.urlauth.filesystem.schemes==hdfs:,file:,wasb:,adl:,alluxio:

# Add the `hive.properties` configuration item in the Presto component
hive.hdfs.authentication.type=NONE
hive.metastore.authentication.type=NONE
hive.hdfs.impersonation.enabled=true
hive.metastore.thrift.impersonation.enabled=true
```

Note

The above prerequisite configuration items need to be configured based on the existing components in your cluster.

HDFS authentication

Create a soft link to the Ranger configuration file as follows:

```
[hadoop@172 conf]$ pwd
/usr/local/service/alluxio/conf
[hadoop@172 conf]$ ln -s /usr/local/service/hadoop/etc/hadoop/ranger-hdfs-
audit.xml
```

©2013-2024 Tencent Cloud. All rights reserved.

```
ranger-hdfs-audit.xml
[hadoop@172 conf]$ ln -s /usr/local/service/hadoop/etc/hadoop/ranger-hdfs-
security.xml ranger-hdfs-security.xml
```

Configure `alluxio-site.properties`

We recommend you deliver the cluster configuration in the EMR console.

```
# Authentication switch (`false` by default)
alluxio.security.authorization.plugins.enabled=true
alluxio.security.authorization.plugin.name=ranger
alluxio.security.authorization.plugin.paths=/usr/local/service/alluxio/conf
alluxio.underfs.security.authorization.plugin.name=ranger
alluxio.underfs.security.authorization.plugin.paths=/usr/local/service/alluxio/
conf
alluxio.master.security.impersonation.hadoop.users=*
alluxio.security.login.impersonation.username=_HDFS_USER_
```

Note

You need to restart the Alluxio service after the delivery is completed.

COS and CHDFS authentication

```
# Add the `core-site.xml` configuration item
fs.ofs.ranger.enable.flag=true
```

Configure `alluxio-site.properties`

We recommend you deliver the cluster configuration in the EMR console.

```
# Authentication switch (`false` by default)
# Authentication switch (`false` by default)
alluxio.security.authorization.plugins.enabled=true
alluxio.security.authorization.plugin.name=ranger
alluxio.security.authorization.plugin.paths=/usr/local/service/alluxio/conf
alluxio.underfs.security.authorization.plugin.name=ranger
alluxio.underfs.security.authorization.plugin.paths=/usr/local/service/alluxio/
conf
alluxio.cos.qcloud.object.storage.ranger.service.config.dir=/usr/local/service/
cosranger/conf
alluxio.master.security.impersonation.hadoop.users=*
alluxio.security.login.impersonation.username=_HDFS_USER_
# The number of retries is 5 by default.
alluxio.cos.qcloud.object.storage.permission.check.max.retry=5
```

Note

You need to restart the Alluxio service after the delivery is completed.

©2013-2024 Tencent Cloud. All rights reserved.

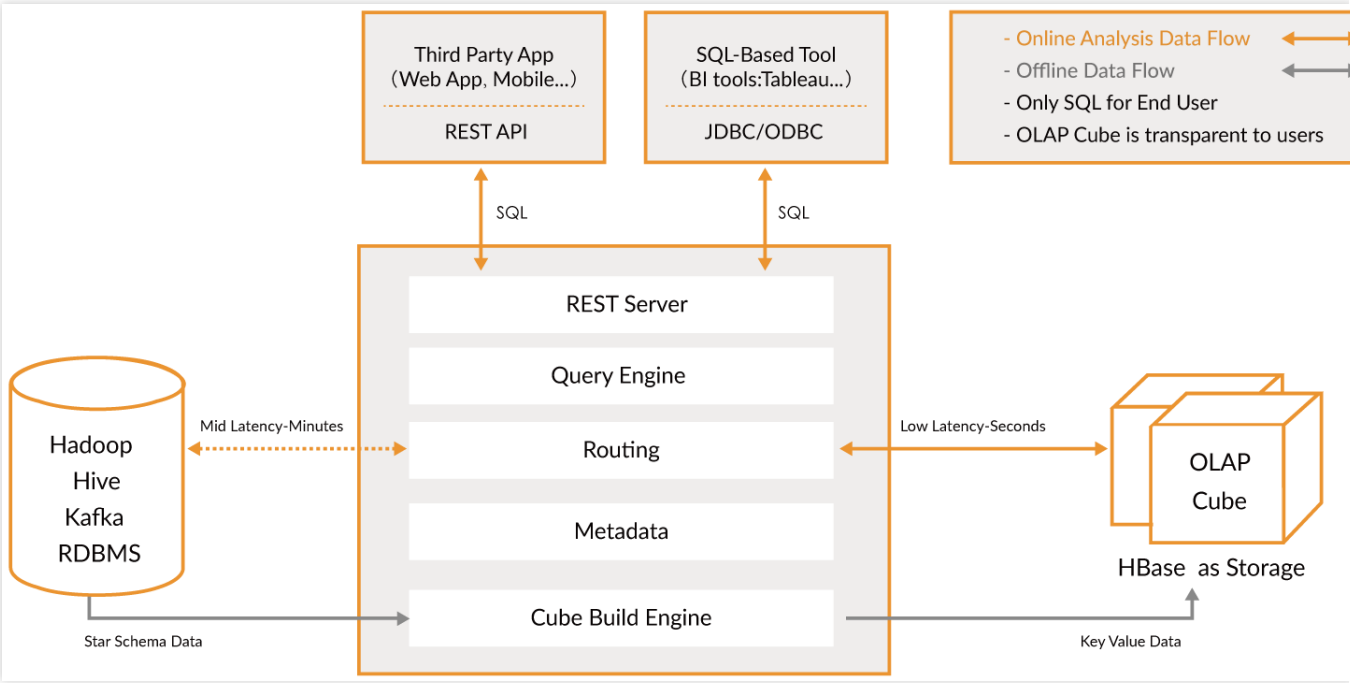
Kylin Development Guide
Kylin Overview

Last updated：2025-01-03 15:02:25

Originally developed by eBay and then contributed to the open source community, Apache Kylin™ is an open-source and distributed analytical data warehouse designed to provide SQL interface and multi-dimensional analysis (OLAP) for Hadoop and Spark. It supports extremely large-scale datasets and can query huge tables in sub-seconds.

Overview of Kylin Framework

The key that enables Kylin to provide a sub-second latency is pre-calculation, which involves pre-calculating the measures of a data cube with a star topology in a combination of dimensions, saving the results in HBase, and then providing query APIs such as JDBC, ODBC, and RESTful APIs to implement real-time queries.



Core concepts of Kylin

Table: it is defined in Hive and is the data source of a data cube which must be synced into Kylin before a cube is built.

Model: it describes a star schema data structure and defines the connection and filtering relationship between a fact table and a number of lookup tables.

©2013-2024 Tencent Cloud. All rights reserved.

Cube descriptor: it describes the definition and configuration of a cube instance, including which data model to use, what dimensions and measures to have, how to partition data, how to handle auto-merge, etc.
Cube Instance: it is built from a cube descriptor and consists of one or more cube segments.
Partition: you can use a DATE/STRING column in a cube descriptor to separate a cube into several segments by date.
Cube segment: it is the carrier of cube data. A segment maps to a table in HBase. After a cube instance is built, a new segment will be created. In case of any changes in the raw data of a built cube, you simply need to refresh the associated segments.
Aggregation group: each aggregation group is a subset of dimensions and builds cuboid with combinations inside.
Job: after a request to build a cube instance is initiated, a job will be generated, which records information about each step in building the cube instance. The status of the job reflects the result of cube instance creation. For example, RUNNING indicates that the cube instance is being built; FINISHED indicates that the cube instance has been successfully built; and ERROR indicates that cube instance creation failed.
Dimension and measure types
Mandatory: this is a dimension that all cuboids must have.
Hierarchy: there is a hierarchical relationship between dimensions. Only hierarchical cuboids need to be retained.
Derived: in lookup tables, some dimensions can be derived from their primary key, so they will not be involved in cuboid creation.
Count Distinct(HyperLogLog): immediate COUNT DISTINCT is hard to calculate, so an approximate algorithm (HyperLogLog) is introduced to keep error rate at a lower level.
Count Distinct(Precise): precise COUNT DISTINCT will be pre-calculated based on RoaringBitmap. Currently, only `int` and `bigint` are supported.
Cube action types
BUILD: this action builds a new cube segment at the time interval specified by a partition.
REFRESH: this action rebuilds a cube segment over some intervals.
MERGE: this action merges multiple cube segments. It can be set to be automated when building a cube.
PURGE: this action clears segments under a cube instance. However, it will not delete tables from HBase.
Job status
NEW: this indicates that the job has been created.
PENDING: this indicates that the job has been submitted by the job scheduler and is waiting for execution resources.
RUNNING: this indicates that the job is running.
FINISHED: this indicates that the job has been successfully finished.
ERROR: this indicates that the job has exited due to an error.
DISCARDED: this indicates that the job has been canceled by the user.
Job execution
RESUME: this action will continue to execute a failed job from the last successful checkpoint.
DISCARD: regardless of the status of the job, the user can end it and release the resources.

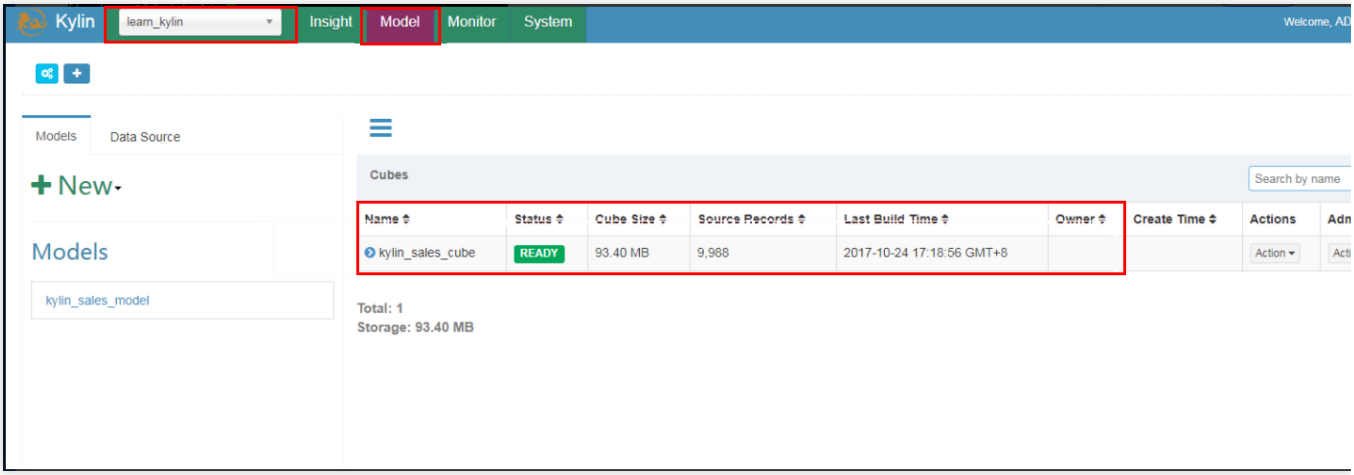
©2013-2024 Tencent Cloud. All rights reserved.

Getting Started with Cubes

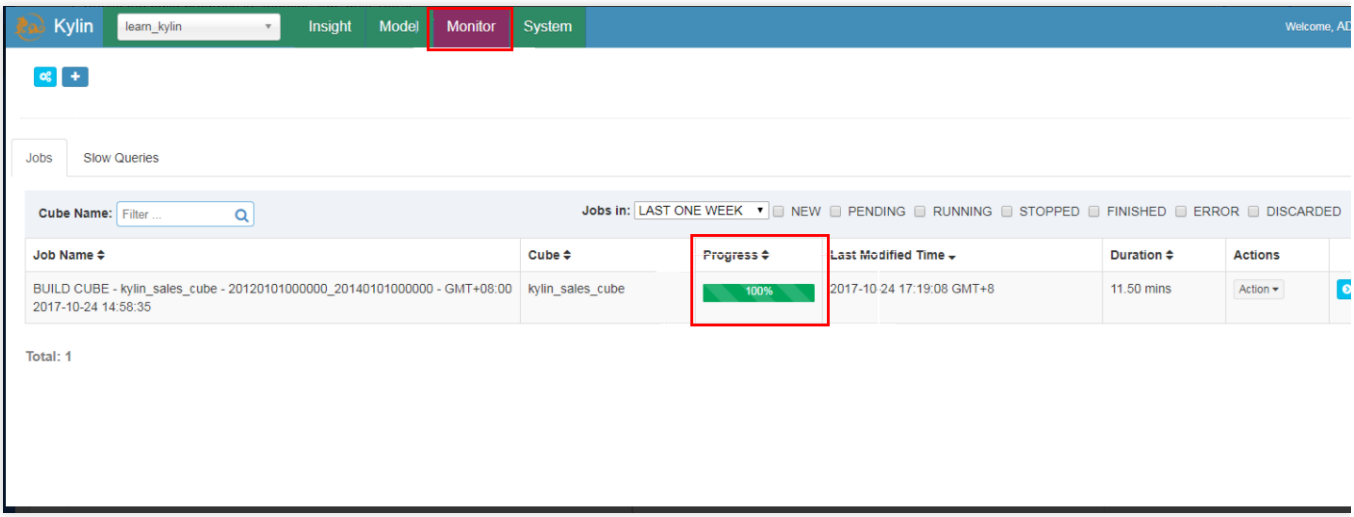
Run the script to restart the Kylin server to purge the cache.

```
/usr/local/service/kylin/bin/sample.sh
```

Log in at the Kylin website with the default username and password (ADMIN/KYLIN), select the `learn_kylin` project from the project drop-down list in the top-left corner, select the sample cube named `kylin_sales_cube`, click **Actions** > **Build**, and select a date after January 1, 2014 (overwriting all 10000 sample records).

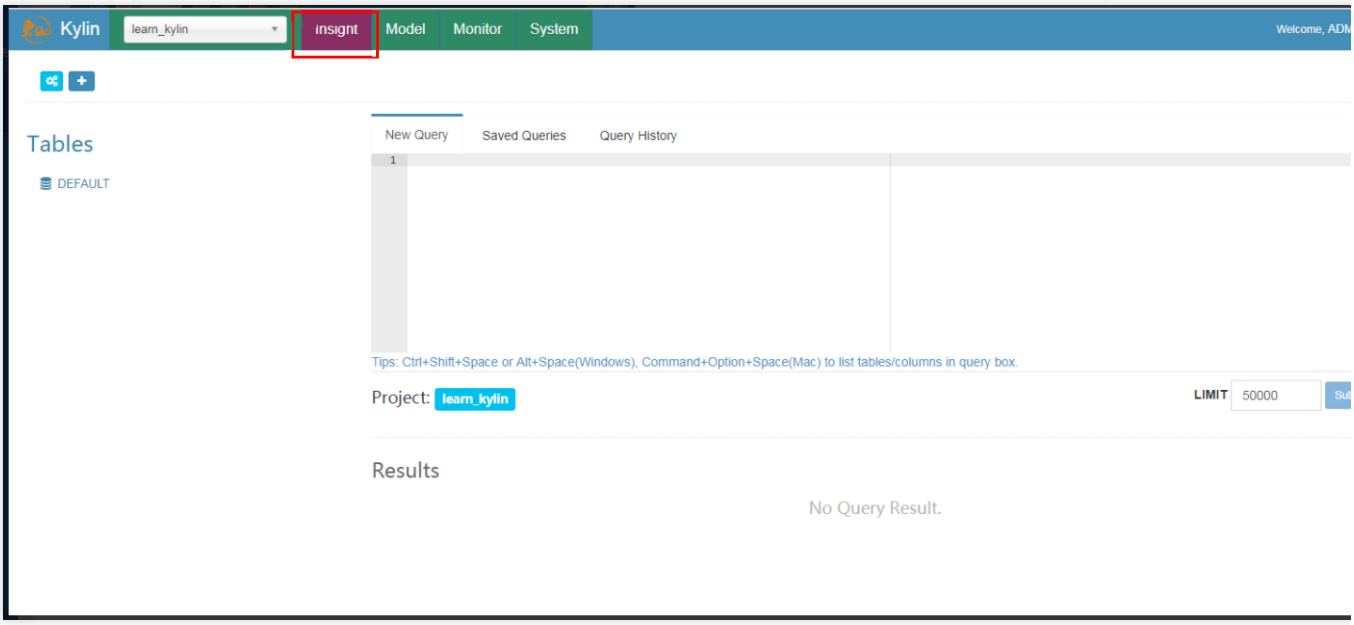


Click the **Monitor** to view the building progress until 100%.



Click the **Insight** to execute SQLs; for example:

©2013-2024 Tencent Cloud. All rights reserved.




```
select part_dt, sum(price) as total_sold, count(distinct seller_id) as sellers
from kylin_sales group by part_dt order by part_dt
```

Building Cube with Spark

1. Set the `kylin.env.hadoop-conf-dir` property in `kylin.properties` .

```
kylin.env.hadoop-conf-dir=/usr/local/service/hadoop/etc/hadoop
```

2. Check the Spark configuration.

Kylin embeds a Spark binary (v2.1.2) in `$KYLIN_HOME/spark` , and all Spark properties prefixed with `kylin.engine.spark-conf.` can be managed in `$KYLIN_HOME/conf/kylin.properties` . These properties will be extracted and applied when a submitted Spark job is executed; for example, if you configure `kylin.engine.spark-conf.spark.executor.memory=4G` , Kylin will use `-conf spark.executor.memory=4G` as a parameter when executing `spark-submit` .

Before you run Spark cubing, you are recommended to take a look at these configurations and customize them based on your cluster. Below is the recommended configuration with Spark dynamic resource allocation enabled:

```
kylin.engine.spark-conf.spark.master=yarn
kylin.engine.spark-conf.spark.submit.deployMode=cluster
kylin.engine.spark-conf.spark.dynamicAllocation.enabled=true
kylin.engine.spark-conf.spark.dynamicAllocation.minExecutors=1
kylin.engine.spark-conf.spark.dynamicAllocation.maxExecutors=1000
kylin.engine.spark-conf.spark.dynamicAllocation.executorIdleTimeout=300
```

©2013-2024 Tencent Cloud. All rights reserved.

```
kylin.engine.spark-conf.spark.yarn.queue=default
kylin.engine.spark-conf.spark.driver.memory=2G
kylin.engine.spark-conf.spark.executor.memory=4G
kylin.engine.spark-conf.spark.yarn.executor.memoryOverhead=1024
kylin.engine.spark-conf.spark.executor.cores=1
kylin.engine.spark-conf.spark.network.timeout=600
kylin.engine.spark-conf.spark.shuffle.service.enabled=true
#kylin.engine.spark-conf.spark.executor.instances=1
kylin.engine.spark-conf.spark.eventLog.enabled=true
kylin.engine.spark-conf.spark.hadoop.dfs.replication=2
kylin.engine.spark-
conf.spark.hadoop.mapreduce.output.fileoutputformat.compress=true
kylin.engine.spark-
conf.spark.hadoop.mapreduce.output.fileoutputformat.compress.codec=org.apache.h
adoop.io.compress.DefaultCodec
kylin.engine.spark-
conf.spark.io.compression.codec=org.apache.spark.io.SnappyCompressionCodec
kylin.engine.spark-conf.spark.eventLog.dir=hdfs\\:///kylin/spark-history
kylin.engine.spark-conf.spark.history.fs.logDirectory=hdfs\\:///kylin/spark-
history
## Uncommenting for HDP
#kylin.engine.spark-conf.spark.driver.extraJavaOptions=-Dhdp.version=current
#kylin.engine.spark-conf.spark.yarn.am.extraJavaOptions=-Dhdp.version=current
#kylin.engine.spark-conf.spark.executor.extraJavaOptions=-Dhdp.version=current
```

For running on the Hortonworks platform, you need to specify `hdp.version` as the Java option for Yarn container; therefore, you should uncomment the last three lines in `kylin.properties`.

Besides, in order to avoid repeatedly uploading Spark jars to Yarn, you can manually upload them once and then configure the jar's HDFS path. **The HDFS path must be a full path name.**

```
jar cv0f spark-libs.jar -C $KYLIN_HOME/spark/jars/ .
hadoop fs -mkdir -p /kylin/spark/
hadoop fs -put spark-libs.jar /kylin/spark/
```

Then, configure `kylin.properties` as follows:

```
kylin.engine.spark-
conf.spark.yarn.archive=hdfs://sandbox.hortonworks.com:8020/kylin/spark/spark-
libs.jar
```

All the `kylin.engine.spark-conf.*` parameters can be overwritten at the cube or project level, which gives you more flexibility.

3. Create and modify a sample cube.

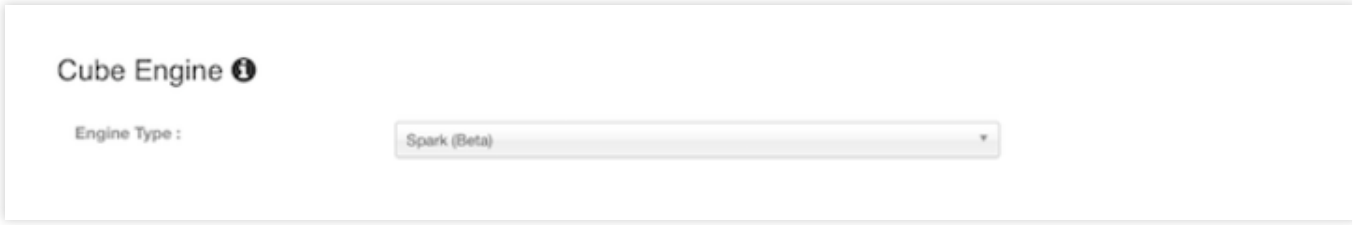
Run `sample.sh` to create a sample cube and then start the Kylin server:

```
/usr/local/service/kylin/bin/sample.sh
```

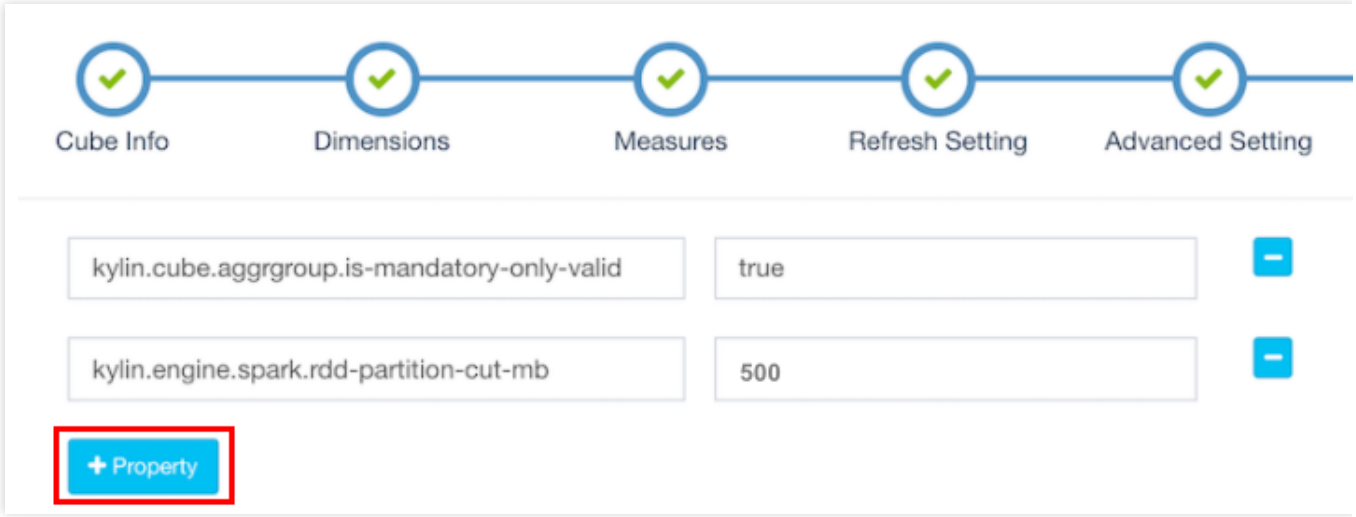
©2013-2024 Tencent Cloud. All rights reserved.

```
/usr/local/service/kylin/bin/kylin.sh start
```

After Kylin is started, access the Kylin website and edit the `kylin_sales` cube on the "Advanced Setting" page by changing **Cube Engine** from **MapReduce** to **Spark (Beta)**:



Click **Next** to enter the "Configuration Overwrites" page, and click **+Property** to add the `kylin.engine.spark.rdd-partition-cut-mb` property with a value of 500.



The sample cube has two memory hungry measures: `COUNT DISTINCT` and `TOPN(100)` . When the source data is small, their estimated size will be much larger than their actual size, thus causing more RDD partitions to be split and slowing down the building process. 500 is a reasonable number. Click **Next** and **Save** to save the cube.

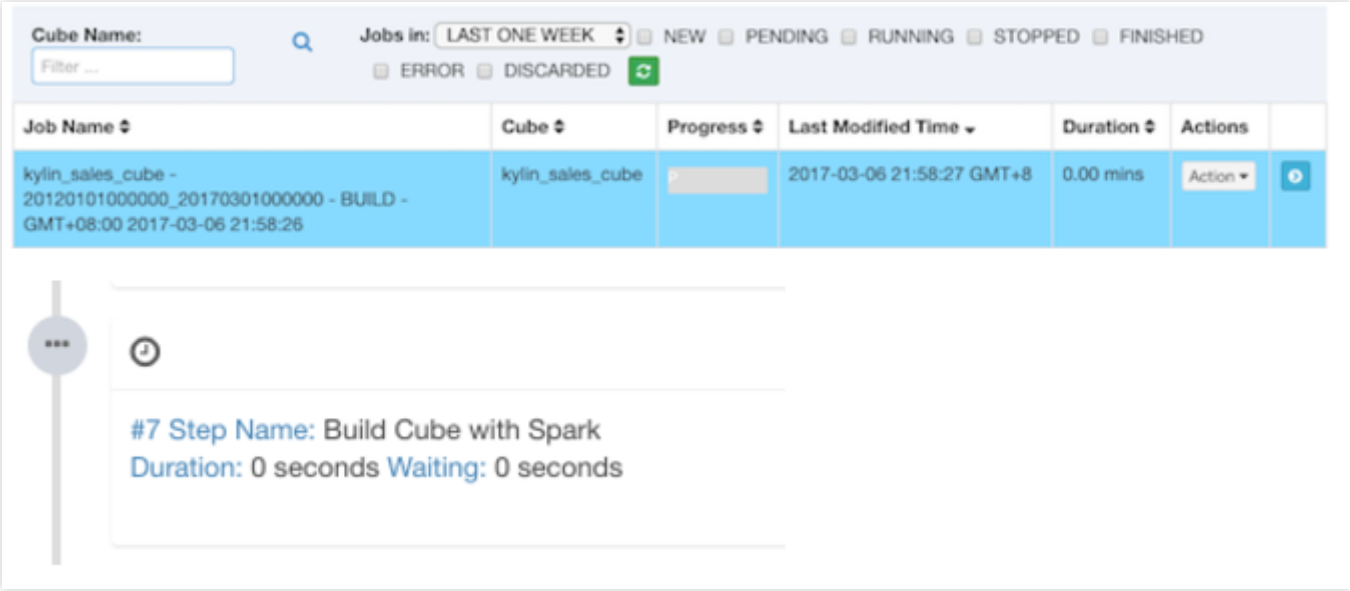
> For cubes without `COUNT DISTINCT` and `TOPN` , please keep the default configuration.

3. Build a cube with Spark.

Click **Build** and select the current date as the end date. Kylin will generate a building job on the "Monitor" page, in

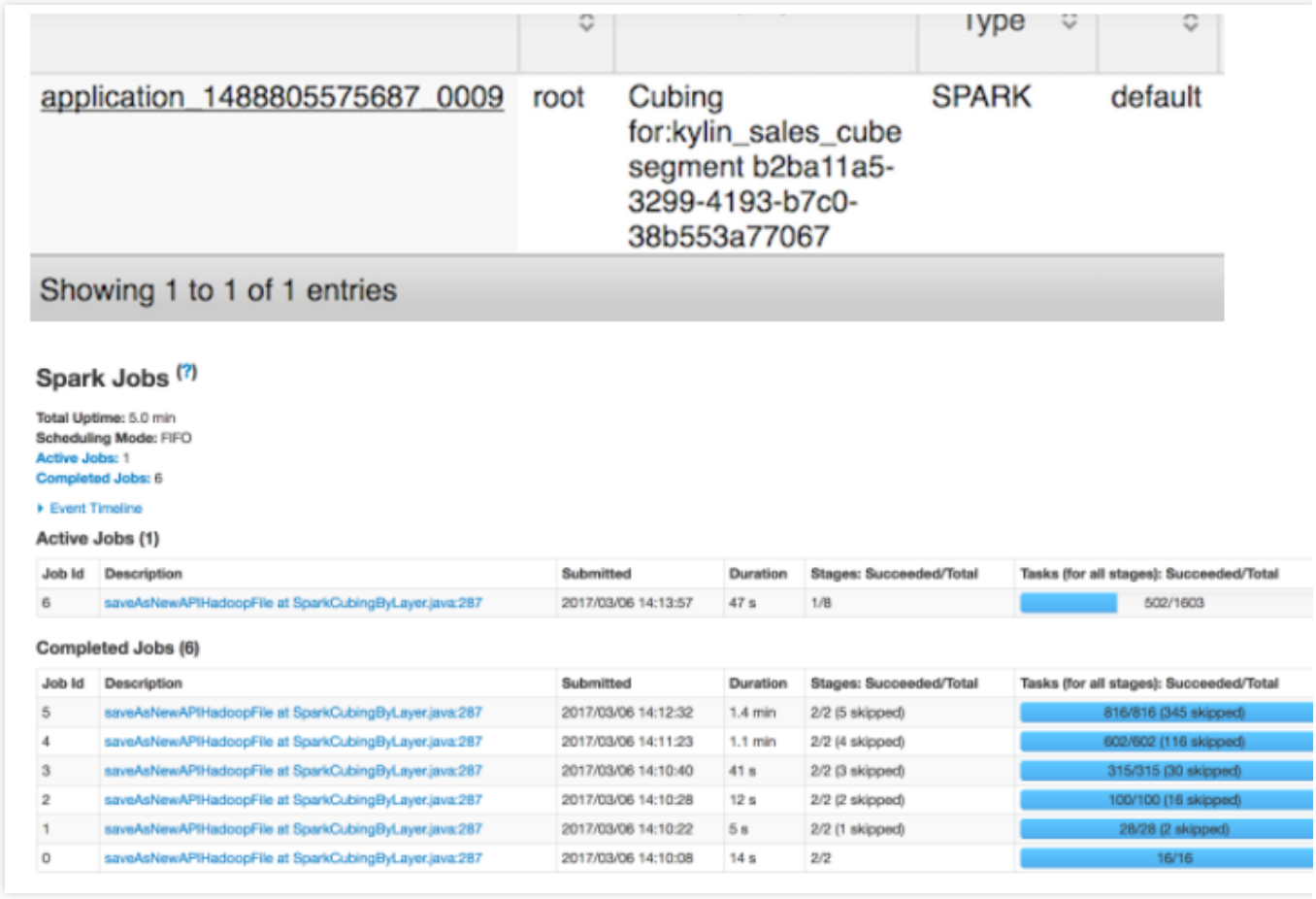
©2013-2024 Tencent Cloud. All rights reserved.

which the 7th step is Spark cubing. The job engine will start to execute the steps in sequence.



When Kylin executes this step, you can monitor the status in the Yarn resource manager. Click the "Application Master" link to open the web UI of Spark, which will display the progress and details of each stage.

©2013-2024 Tencent Cloud. All rights reserved.



After all the steps are successfully performed, the cube will become "Ready", and you can perform queries.

©2013-2024 Tencent Cloud. All rights reserved.

Livy Development Guide
Livy Overview

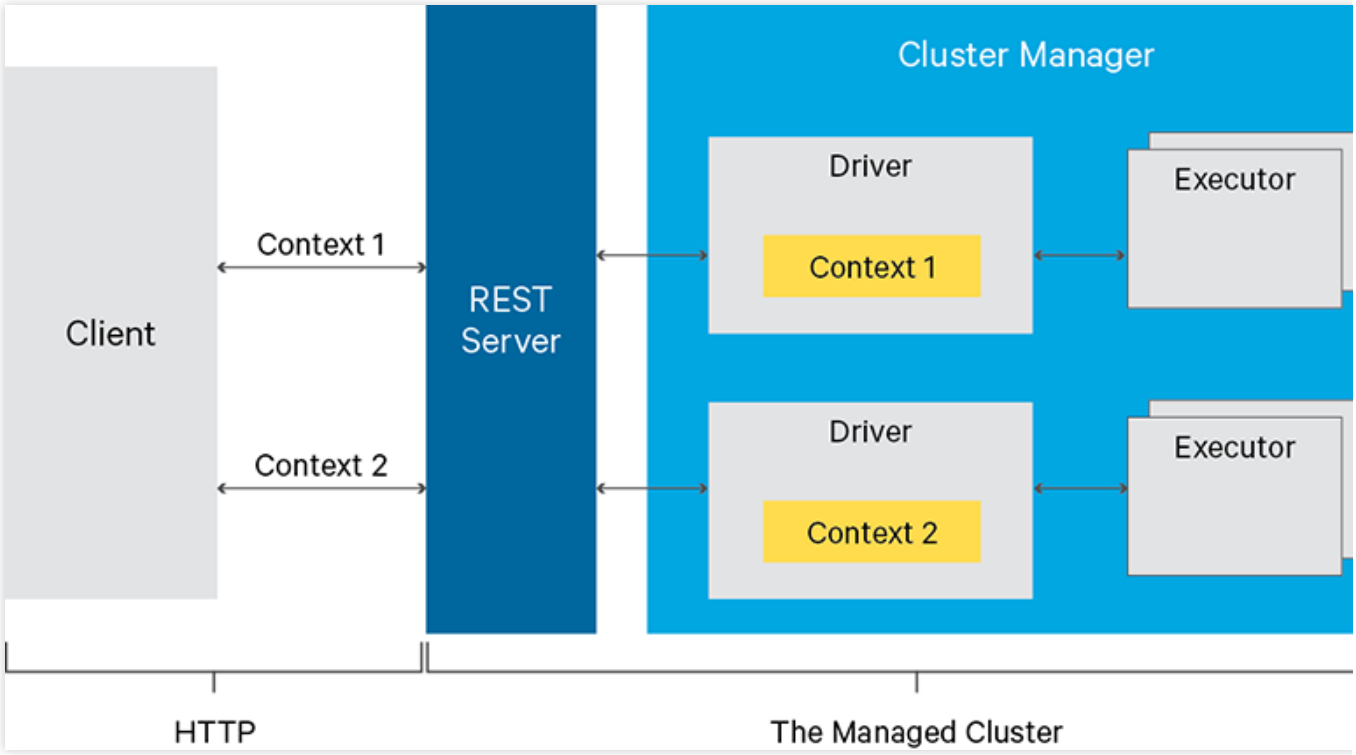
Last updated：2025-01-03 15:02:25

Apache Livy is a service that enables easy interaction with a Spark cluster over a REST interface. It enables easy submission of Spark jobs or snippets of Spark code, synchronous or asynchronous result retrieval, as well as Spark Context management, all via a simple REST interface or an RPC client library. Apache Livy also simplifies the interaction between Spark and application servers, thus enabling the use of Spark for interactive web/mobile applications.

Livy Features

Additional features include:
Have long running Spark Contexts that can be used for multiple Spark jobs, by multiple clients.
Share cached RDDs or Dataframes across multiple jobs and clients.
Multiple Spark Contexts can be managed simultaneously, and the Spark Contexts run on the cluster (YARN/Mesos) instead of the Livy Server, for good fault tolerance and concurrency.
Jobs can be submitted as precompiled jars, snippets of code or via java/scala client API
Ensure security via secure authenticated communication.

©2013-2024 Tencent Cloud. All rights reserved.



Using Livy

1. Access `http://IP:8998/ui` to enter the UI of Livy (**this IP is the public IP. You need to apply for a public IP for the server where Livy is installed and set the security group policy to open the port for access**).

2. Create an interactive session.

```
curl -X POST --data '{"kind":"spark"}' -H "Content-Type:application/json"
IP:8998/sessions
```

3. View the alive sessions on Livy.

```
curl IP:8998/sessions
```

4. Execute the code snippet to perform a simple addition operation (here, specify session 0, or specify another session if there are multiple sessions).

```
curl -X POST IP:8998/sessions/0/statements -H "Content-Type:application/json" -
d '{"code":"1+1"}'
```

5. Calculate the pi (execute the JAR package).

Step 1. Upload the JAR package to HDFS, such as to `/usr/local/spark-examples_2.11-2.4.3.jar` .

©2013-2024 Tencent Cloud. All rights reserved.

Step 2. Run the following command:

```
curl -H "Content-Type: application/json" -X POST -d
'{ "file":"/usr/local/spark-examples_2.11-2.4.3.jar",
"className":"org.apache.spark.examples.SparkPi" }' IP:8998/batches
```

6. Check whether the code snippet has been successfully executed. You can also view it on the UI at `http://IP:8998/ui/session/0` .

```
curl IP:8998/sessions/0/statements/0
```

7. Delete the session.

```
curl -X DELETE IP:8998/sessions/0
```

Notes

Open configuration files

Currently, open configuration files include `livy.conf` and `livy-env.sh` , both of which can be used to modify the configuration through configuration distribution. Check the EMR console for the actual open configuration files.

Changing the port used by Livy

By default, Livy runs on port 8998, which can be changed with the `livy.server.port` configuration option in the `livy.conf` configuration file.

If Hue is installed in the cluster, due to the connectivity between Hue and Livy, the port for Hue also needs to be changed with the `livy_server_port=8998` configuration option in the `pseudo-distributed.ini` configuration file at `/usr/local/service/hue/desktop/conf` . The service needs to be restarted after the change.

Unless necessary, we don't recommend you modify the port used by Livy. Instead, you can control the access by using a security group. Any change may cause other potential problems.

Livy deployment

Currently, Livy is deployed on all master nodes by default. You can also deploy it on router nodes by scaling out router nodes.

©2013-2024 Tencent Cloud. All rights reserved.

Kyuubi Development Guide
Kyuubi Overview

Last updated：2025-01-03 15:02:25

Apache Kyuubi (Incubating) is a distributed multi-tenant Thrift JDBC/ODBC service. Currently, it has been connected to Apache Spark and is being connected to Apache Flink and Trino. It can be used for diverse big data scenarios within enterprises, such as ETL and BI reporting.

Use Cases

You can replace HiveServer2 to increase the performance by 10–100 times.
Kyuubi is highly compatible with HiveServer2 APIs and supports seamless migration.

©2013-2024 Tencent Cloud. All rights reserved.

The layered architecture of Kyuubi eliminates client compatibility issues and supports imperceptible upgrade.
Kyuubi supports full-linkage Spark SQL optimization and enhancement for an excellent performance.
Kyuubi offers various enterprise-grade features, such as high availability, multi-tenancy, and fine-grained authentication.
You can build a Serverless Spark platform.
The goal of serverless Spark is definitely not to let you call Spark APIs or continue to write Spark jobs.
The built-in engine module of Kyuubi makes Spark easier to use, and you don't even need to understand the logic.
You can focus on your business development by manipulating data through JDBC and SQL. The resources are elastically scalable and Ops-free.
Resource managers such as Kubernetes and YARN as well as engine lifecycle are supported. Spark allocates resources dynamically and elastically at three different granularities.
You can schedule resources with multiple resource managers such as YARN and Kubernetes simultaneously. This guarantees the secure migration of historical jobs to the cloud.
Spark Adaptive Query Execution (AQE) and Kyuubi AQE Plus robustly empower data operations.
You can build a unified data lake insight, analysis, and management platform (with Kyuubi 1.5 or later).
All Spark and third-party data sources are supported.
Metadata can be managed through Spark DSv2 to visually build and manage data lakes.
All popular data lake frameworks are supported, including Apache Iceberg, Apache Hudi, and Delta Lake.
With one copy of data for one API and one engine, Kyuubi provides a unified query, analysis, data ingestion, and data lake management platform.
Kyuubi integrates batch processing and stream processing and supports stream operations (upcoming).

©2013-2024 Tencent Cloud. All rights reserved.

Kyuubi Practical Tutorial

Last updated：2025-01-03 15:02:25

Connecting Beeline to Kyuubi

Log in to a master node in the EMR cluster, switch to the Hadoop user, and go to the Kyuubi directory:

```
[root@172 ~]# su hadoop
[hadoop@172 root]$ cd /usr/local/service/kyuubi
```

Connect to Kyuubi:

```
[hadoop@10kyuubi]$ bin/beeline -u
"jdbc:hive2://${zkserverip1}:${zkport},${zkserverip2}:${zkport},${zkserverip3}:
${zkport}/default;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=kyuubi" -n
hadoop
```

Or

```
[hadoop@10kyuubi]$ bin/beeline -u
"jdbc:hive2://${kyuubiserverip}:${kyuubiserverport}" -n hadoop
```

For more information on `${zkserverip}:${zkport}`, see the `kyuubi.ha.zookeeper.quorum` configuration item of `kyuubi-defaults.conf`.
For more information on `${kyuubiserverport}`, see the `kyuubi.frontend.bind.port` configuration item of `kyuubi-defaults.conf`.

Creating and viewing database

Create a new table in the database you just created and view the table:

```
0: jdbc:hive2://ip:port> create database sparksql;
+---------+
| Result  |
+---------+
+---------+
No rows selected (0.326 seconds)
```

Insert two rows of data into the table and view them:

```
0: jdbc:hive2://ip:port> use sparksql;
+---------+
| Result  |
+---------+
```

©2013-2024 Tencent Cloud. All rights reserved.

```
+---------+
No rows selected (0.077 seconds)
0: jdbc:hive2://ip:port> create table sparksql_test(a int,b string);
+---------+
| Result  |
+---------+
+---------+
No rows selected (0.402 seconds)
0: jdbc:hive2://ip:port> show tables;
+-----------+----------------+--------------+
| database  |   tableName    | isTemporary  |
+-----------+----------------+--------------+
| sparksql  | sparksql_test  | false        |
+-----------+----------------+--------------+
1 row selected (0.108 seconds)
```

Connecting Hue to Kyuubi

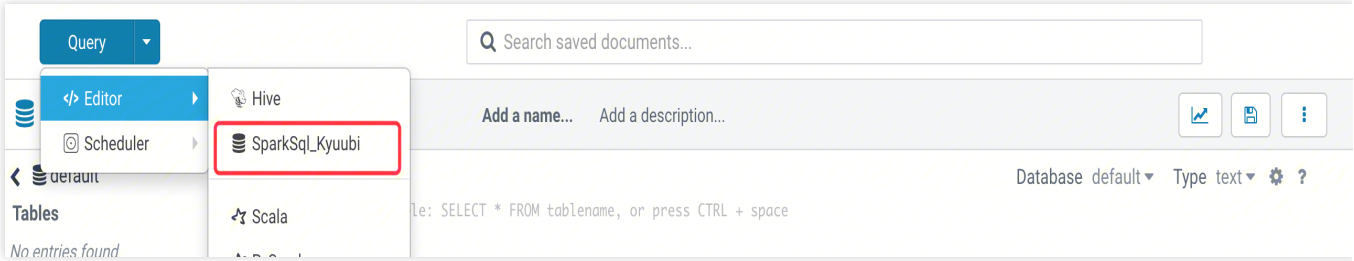
Prerequisites

If Kyuubi is installed in the existing cluster, you need to perform the following operations before you can use Kyuubi on Hue:

1. Go to the **Configuration Management** page of HDFS, add the `hadoop.proxyuser.hue.groups` and `hadoop.proxyuser.hue.hosts` configuration items to `core-site.xml`, and set their values to `*`.
2. Restart Kyuubi and Hue.
3. Access the Hue console. For more information, see Hue Development Guide.

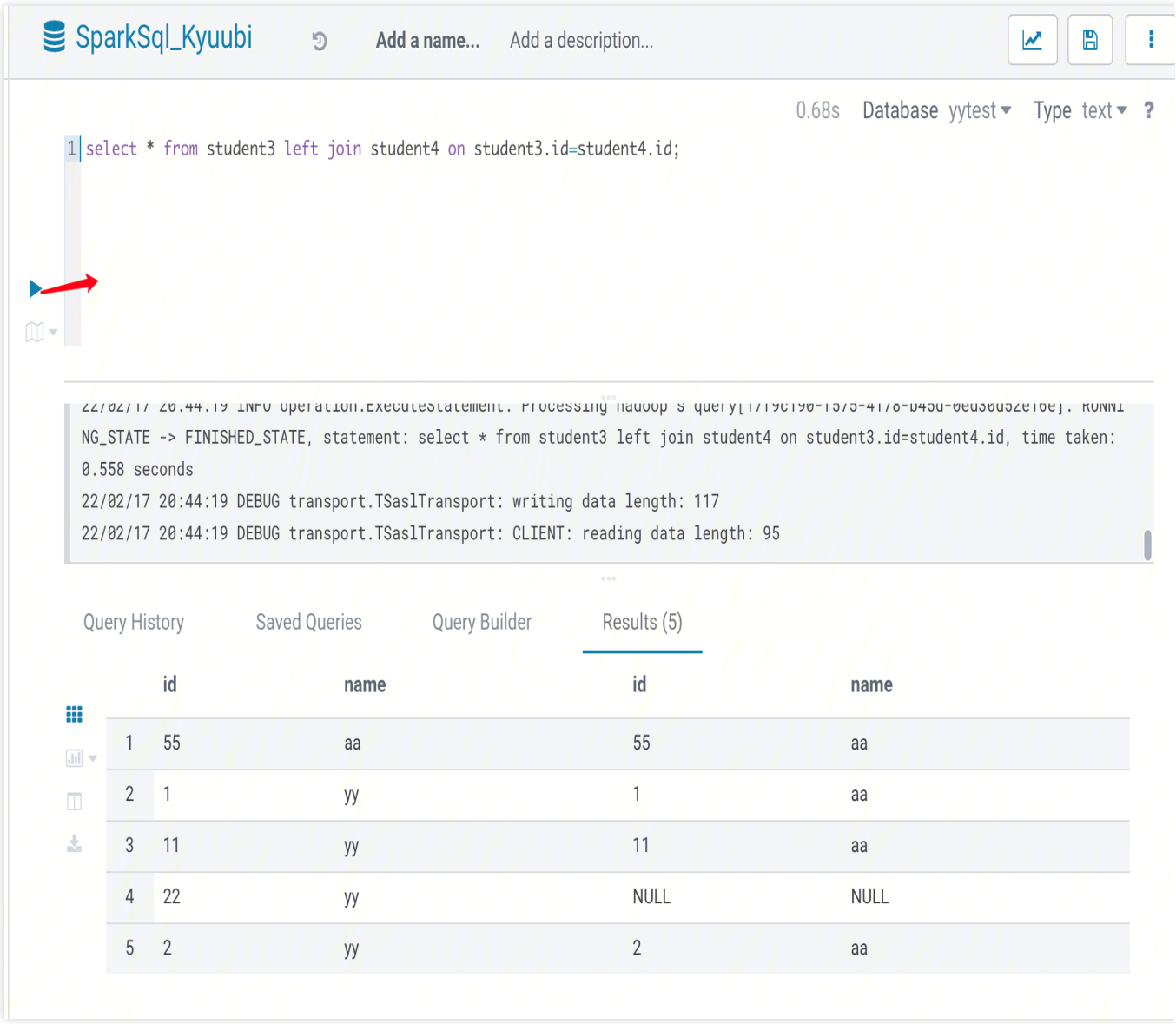
Kyuubi query

1. At the top of the Hue console, select **Query** > **Editor** > **SparkSql_Kyuubi**.



2. Enter the statement to be executed in the statement input box and click **Run** to run it.

©2013-2024 Tencent Cloud. All rights reserved.



Connecting to Kyuubi Through Java

KyuubiServer is integrated with the Thrift service. Thrift, created by Facebook, is an interface definition language and binary communication protocol used for defining and creating services for numerous languages. Apache Kyuubi is based on Thrift, allowing many languages such as Java and Python to call Kyuubi's APIs. Additionally, the Hive JDBC driver for Kyuubi enables Java applications to interact with Kyuubi. This section describes how to connect to Kyuubi through Java code.

1. Prepare for development.

Confirm that you have activated EMR and created an EMR cluster. When creating the EMR cluster, select the Kyuubi and Spark components on the software configuration page.

Kyuubi and its dependencies are installed in the EMR cluster directory `/usr/local/service/` .

©2013-2024 Tencent Cloud. All rights reserved.

2. Use Maven to create a project.

Maven is recommended for project management, as it can help you manage project dependencies with ease. Specifically, it can get JAR packages through the configuration of the `pom.xml` file, eliminating your need to add them manually. Download and install Maven locally first and then configure its environment variables. If you are using the IDE, set the Maven-related configuration items in the IDE.

In the local shell environment, enter the directory where you want to create the Maven project, such as `D://mavenWorkplace` , and enter the following command to create it:

```
mvn archetype:generate -DgroupId=$yourgroupID -DartifactId=$yourartifactID -
DarchetypeArtifactId=maven-archetype-quickstart
```

Here, `$yourgroupID` is your package name, `$yourartifactID` is your project name, and `maven-archetype-quickstart` indicates to create a Maven Java project. Some files need to be downloaded for creating the project, so stay connected to the internet. After successfully creating the project, you will see a folder named `$yourartifactID` in the `D://mavenWorkplace` directory. The files included in the folder have the following structure:

```
simple
---pom.xml       Core configuration, under the project root directory
---src
 ---main
     ---java         Java source code directory
     ---resources    Java configuration file directory
 ---test
     ---java         Test source code directory
     ---resources    Test configuration directory
```

Among the above files, pay extra attention to the `pom.xml` file and the Java folder under the main directory. The `pom.xml` file is primarily used to create dependencies and package configurations; the Java folder is used to store your source code. First, add the Maven dependencies to `pom.xml` :

```
<dependencies>
 <dependency>
     <groupId>org.apache.kyuubi</groupId>
     <artifactId>kyuubi-hive-jdbc-shaded</artifactId>
     <version>1.4.1-incubating</version>
 </dependency>
 <dependency>
     <groupId>org.apache.hadoop</groupId>
     <artifactId>hadoop-common</artifactId>
     <!-- keep consistent with the build hadoop version -->
     <version>2.8.5</version>
 </dependency>
</dependencies>
```

©2013-2024 Tencent Cloud. All rights reserved.

Then, add the packaging and compiling plugins to pom.xml:

```
<build>
<plugins>
 <plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<configuration>
  <source>1.8</source>
  <target>1.8</target>
  <encoding>utf-8</encoding>
</configuration>
 </plugin>
 <plugin>
<artifactId>maven-assembly-plugin</artifactId>
<configuration>
  <descriptorRefs>
  <descriptorRef>jar-with-dependencies</descriptorRef>
  </descriptorRefs>
</configuration>
<executions>
  <execution>
    <id>make-assembly</id>
    <phase>package</phase>
    <goals>
      <goal>single</goal>
    </goals>
  </execution>
</executions>
 </plugin>
</plugins>
</build>
```

Right-click in src>main>Java and create a Java Class. Enter the Class name (e.g., `KyuubiJDBCTest.java` here) and add the sample code to the Class:

```
import java.sql.*;

public class KyuubiJDBCTest {
 private static String driverName =
         "org.apache.kyuubi.jdbc.KyuubiHiveDriver";
 public static void main(String[] args)
         throws SQLException {
     try {
         Class.forName(driverName);
     } catch (ClassNotFoundException e) {
```

©2013-2024 Tencent Cloud. All rights reserved.

```
            e.printStackTrace();
            System.exit(1);
        }
        Connection con = DriverManager.getConnection(
                "jdbc:hive2://$kyuubiserverhost:$kyuubiserverport/default",
"hadoop", "");
        Statement stmt = con.createStatement();
        String tableName = "KyuubiTestByJava";
        stmt.execute("drop table if exists " + tableName);
        stmt.execute("create table " + tableName +
                " (key int, value string)");
        System.out.println("Create table success!");
        // show tables
        String sql = "show tables '" + tableName + "'";
        System.out.println("Running: " + sql);
        ResultSet res = stmt.executeQuery(sql);
        if (res.next()) {
            System.out.println(res.getString(1));
        }
        // describe table
        sql = "describe " + tableName;
        System.out.println("Running: " + sql);
        res = stmt.executeQuery(sql);
        while (res.next()) {
            System.out.println(res.getString(1) + "\\t" + res.getString(2));
        }
        sql = "insert into " + tableName + " values (42,\\"hello\\"),
(48,\\"world\\")";
        stmt.execute(sql);
        sql = "select * from " + tableName;
        System.out.println("Running: " + sql);
        res = stmt.executeQuery(sql);
        while (res.next()) {
            System.out.println(String.valueOf(res.getInt(1)) + "\\t"
                    + res.getString(2));
        }
        sql = "select count(1) from " + tableName;
        System.out.println("Running: " + sql);
        res = stmt.executeQuery(sql);
        while (res.next()) {
            System.out.println(res.getString(1));
        }
    }
}
```

Note:

©2013-2024 Tencent Cloud. All rights reserved.

The `$kyuubiserverhost` and `$kyuubiserverport` parameters in the program should be replaced with the values of the IP and port number of the KyuubiServer you queried.

If your Maven is configured correctly and its dependencies are imported successfully, the project can be compiled directly. Enter the project directory in the local shell and run the following command to package the entire project:

```
mvn package
```

3. Upload and run the program.

First, use the SCP or SFTP tool to upload the compressed JAR package to the EMR cluster. Run the following command in your local shell:

```
scp $localfile root@public IP address:/usr/local/service/kyuubi
```

Be sure to upload a JAR package that contains the dependencies. Log in to the EMR cluster, switch to the Hadoop user, and go to the `/usr/local/service/kyuubi` directory. Then, you can run the following program:

```
[hadoop@172 kyuubi]$ yarn jar $package.jar KyuubiJDBCTest
```

Here, `$package.jar` is the path plus name of your JAR package, and `KyuubiJDBCTest` is the name of the previously created Java Class. The result is as follows:

```
Create table success!
Running: show tables 'KyuubiTestByJava'
default
Running: describe KyuubiTestByJava
key     int
value   string
Running: select * from KyuubiTestByJava
42      hello
48      world
Running: select count(1) from KyuubiTestByJava
2
```

©2013-2024 Tencent Cloud. All rights reserved.

Zeppelin Development Guide
Zeppelin Overview

Last updated：2025-01-03 15:02:25

Apache Zeppelin is a web-based notebook that enables interactive data analysis. It allows you to create interactive collaborative documents with various prebuilt language backends (or interpreters), such as Scala (Apache Spark), Python (Apache Spark), Spark SQL, Hive, and Shell.

Note

The Flink, HBase, Kylin, Livy, and Spark interpreters are configured for EMR v3.3.0 or later and EMR v2.6.0 or later by default. For configuring interpreters of other components for other versions of EMR, see Documentation and configure them based on the Zepplin version.

Prerequisites

You have created a cluster and selected the Zeppelin service. For more information, see Creating EMR Cluster.
In the EMR security group of the cluster, ports 22, 30001, and 18000 and necessary private network IP ranges are enabled (ports 22 and 30001 enabled for a new cluster by default). A new security group must be named in the format of "emr-xxxxxxxx_yyyyMMdd", and the name cannot be modified manually.
Services are added as needed, such as Spark, Flink, HBase, and Kylin.

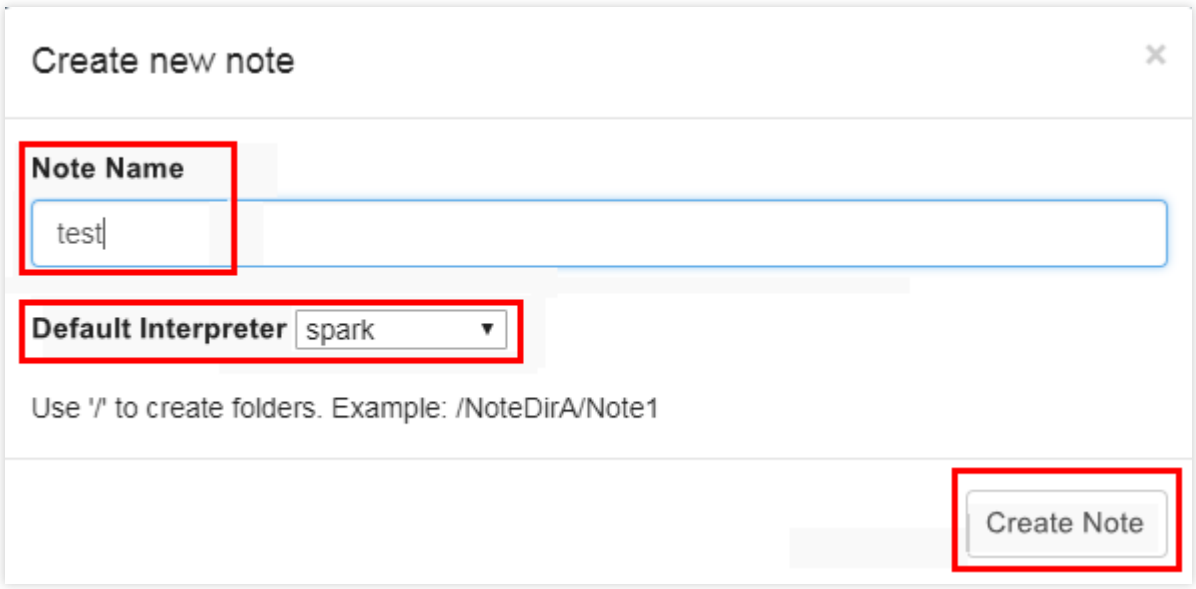
Logging In to Zeppelin

1. Create a cluster and select the Zeppelin service. For more information, see Creating EMR Cluster.
2. On the left sidebar in the EMR console, select **Cluster Services**.
3. Click the Zeppelin block and click **WebUI address** to access the WebUI.
4. For EMR v2.5.0 or earlier and EMR 3.2.1 or earlier, the default login permission is set, and both the username and password are `admin`. To change the password, you can modify the `users` and `roles` options in the configuration file `/usr/local/service/zeppelin-0.8.2/conf/shiro.ini`. For more configuration instructions, see here.
5. In EMR v2.6.0 or later and EMR v3.3.0 or later, Zeppelin login is integrated into the OpenLDAP account, so you can log in only with an OpenLDAP account and password. After a cluster is created, the default OpenLDAP accounts are `root` and `hadoop`, and the default password is the cluster password. Only the `root` account has the Zeppelin admin permissions and thus the access to the interpreter configuration page.

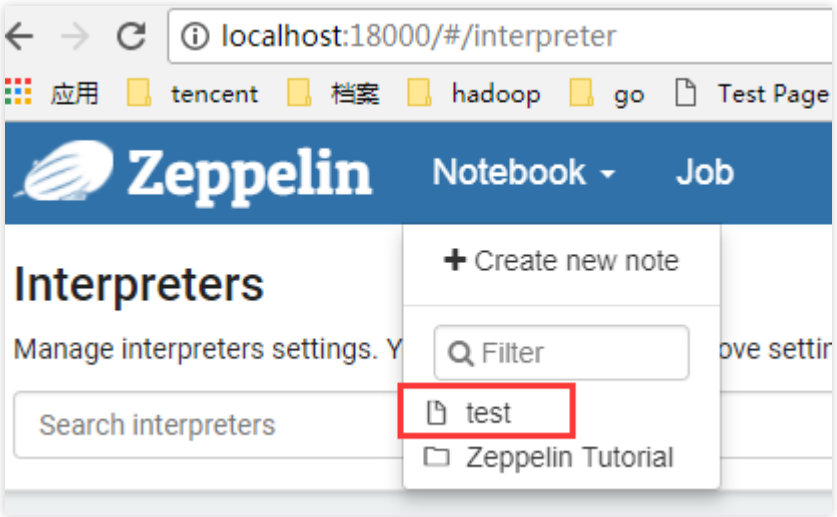
©2013-2024 Tencent Cloud. All rights reserved.

Performing Wordcount Using Spark

1. Click **Create new note** on the left and create a notebook on the pop-up page.



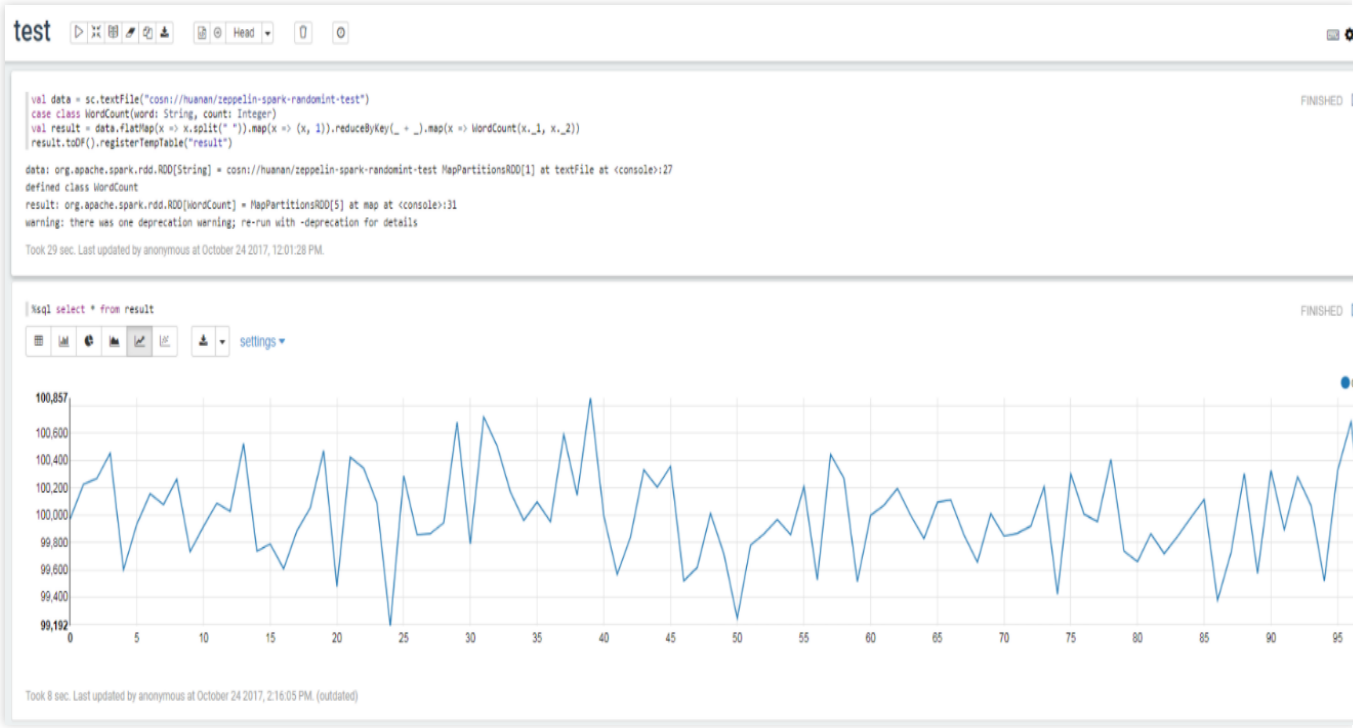
2. For EMR v3.3.0 or later and EMR v2.6.0 or later, clusters connecting Spark to EMR (Spark on YARN) are configured by default.
If you are using EMR v3.1.0, EMR v2.5.0, or EMR v2.3.0, see Documentation to configure the Spark interpreter.
If you are using EMR v3.2.1, see Documentation to configure the Spark interpreter.
3. Go to your own notebook.



4. Write a wordcount program and run the following commands:

©2013-2024 Tencent Cloud. All rights reserved.

```
val data = sc.textFile("cosn://huanan/zeppelin-spark-randomint-test")
case class WordCount(word: String, count: Integer)
val result = data.flatMap(x => x.split(" ")).map(x => (x, 1)).reduceByKey(_ +
_).map(x => WordCount(x._1, x._2))
result.toDF().registerTempTable("result")
%sql select * from result
```



©2013-2024 Tencent Cloud. All rights reserved.

Zeppelin Interpreter Configuration

Last updated：2025-01-03 15:02:25

This document describes how to configure and verify common interpreters for Zeppelin (v0.91 or later is used as an example).

Spark Interpreter

Configuration

```
SPARK_HOME: /usr/local/service/spark
spark.master: yarn
spark.submit.deployMode: cluster
spark.app.name: zeppelin-spark
```

Verification

1. Upload the `wordcount.txt` file to the `/tmp` path of `emr hdfs` first.
2. Find the `hdfs://HDFS45983` value of the `fs.defaultFS` configuration item in `core-site.xml`.
3. Run the Spark code in the notebook.

```
%spark
val data = sc.textFile("hdfs://HDFS45983/tmp/wordcount.txt")
case class WordCount(word: String, count: Integer)
val result = data.flatMap(x => x.split(" ")).map(x => (x, 1)).reduceByKey(_ +
_).map(x => WordCount(x._1, x._2))
result.toDF().registerTempTable("result")

%sql
select * from result
```

Flink Interpreter

Configuration

```
FLINK_HOME: /usr/local/service/flink

flink.execution.mode: yarn
```

©2013-2024 Tencent Cloud. All rights reserved.

Verification

```
%flink
val data = benv.fromElements("hello world", "hello flink", "hello hadoop")
data.flatMap(line => line.split("\\\\s"))
              .map(w => (w, 1))
              .groupBy(0)
              .sum(1)
              .print()
```

HBase Interpreter

Configuration

```
hbase.home: /usr/local/service/hbase

hbase.ruby.sources: lib/ruby

zeppelin.hbase.test.mode: false
```

Note:

As the JAR packages depended on by this interpreter have been integrated into the `/usr/local/service/zeppelin/local-repo` path of the cluster, you don't need to configure dependencies. They are required only if you want to define JAR packages.

Verification

```
%hbase
help 'get'

%hbase
list
```

Livy Interpreter

Configuration

```
zeppelin.livy.url: http://ip:8998
```

Verification

©2013-2024 Tencent Cloud. All rights reserved.

```
%livy.spark
sc.version

%livy.pyspark
print "1"

%livy.sparkr
hello <- function( name ) {
    sprintf( "Hello, %s", name );
}
hello("livy")
```

Kylin Interpreter

Configuration

1. Create a default project in the Kylin console.
2. Configure the Kylin interpreter for Zeppelin.

```
kylin.api.url: http://ip:16500/kylin/api/query

kylin.api.user: ADMIN

kylin.api.password: KYLIN

kylin.query.project: default
```

Verification

```
%kylin(default)

select count(*) from table1
```

JDBC Interpreters

1. MySQL interpreter configuration

```
default.url: jdbc:mysql://ip:3306

default.user: xxx
```

©2013-2024 Tencent Cloud. All rights reserved.

```
default.password: xxx

default.driver: com.mysql.jdbc.Driver
```

Note:

As the JAR packages depended on by this interpreter have been integrated into the `/usr/local/service/zeppelin/local-repo` path of the cluster, you don't need to configure dependencies. They are required only if you want to define JAR packages.

Verification

```
%mysql
show databases
```

2. Hive interpreter configuration

```
default.url: jdbc:hive2://ip:7001

default.user: hadoop

default.password:

default.driver: org.apache.hive.jdbc.HiveDriver
```

Note:

As the JAR packages depended on by this interpreter have been integrated into the `/usr/local/service/zeppelin/local-repo` path of the cluster, you don't need to configure dependencies. They are required only if you want to define JAR packages.

Verification

```
%hive
show databases

%hive
use default;
show tables;
```

3. Presto interpreter configuration

```
default.url: jdbc:presto://ip:9000?user=hadoop
```

©2013-2024 Tencent Cloud. All rights reserved.

```
default.user: hadoop

default.password:

default.driver: io.prestosql.jdbc.PrestoDriver
```

Note:

As the JAR packages depended on by this interpreter have been integrated into the `/usr/local/service/zeppelin/local-repo` path of the cluster, you don't need to configure dependencies. They are required only if you want to define JAR packages.

Verification

```
%presto
show catalogs;

%presto
show schemas from hive;

%presto
show tables from hive.default;
```

For more versions and interpreter configuration, see Zeppelin Documentation.

©2013-2024 Tencent Cloud. All rights reserved.

Hudi Development Guide
Hudi Overview

Last updated：2025-01-03 15:02:25

Apache Hudi provides streaming primitives over HDFS datasets for upsert and incremental pull.
In most cases, we store a large amount of data in HDFS. As new data is incrementally written to the storage, old data is rarely changed, especially if the data is cleansed and stored in, for example, a Hive warehouse. Limited support is provided for updates and the computing costs are high. Further, tools like Hive, Presto, and HBase cannot natively analyze the new data written in a specific period of time. Instead, the data is filtered based on the timestamp before the analysis.
Hudi is a solution to those issues because it supports record-level updates and incremental queries. You can use Apache Hive, Presto, and Spark to directly query the data managed by Hudi.
As a general-purpose big data storage system, Hudi provides the following key features:
Ingests snapshots from query engines, such as Hive, Presto, and Spark.
Supports rollback and savepoint for dataset recovery.
Automatically manages file sizes and layouts to optimize query performance and provide near-real-time data for queries.
Asynchronously compresses real-time data and column data.

Timeline

At its core, Hudi maintains a **timeline** of all actions performed on the dataset at different **instants** of time. Therefore, Hudi provides views of the dataset at different points in time.
A Hudi instant consists of the following components:
Instant action: The type of action performed on the dataset.
Instant time: Instant time is a timestamp, such as 20190117010349, which monotonically increases in the order of the start time of actions.
State: The state of the instant.

File Layout

Hudi organizes a dataset on DFS into a directory structure under a `base path` . The dataset is divided into partitions, which are folders containing data files for that partition. This is much like a Hive table.
Each partition is identified by a specific `partition path` , which is relative to the base path. In each partition, files are organized into file groups, which are uniquely identified by file IDs. Each file group contains multiple file slices.

©2013-2024 Tencent Cloud. All rights reserved.

Each file slice contains a base file `*.parquet`, which is a columnar file generated at a certain commit or compaction instant time, and a set of `*.log*` files that contain upserts to the base file since it was generated. Hudi works on the Multi-Version Concurrency Control (MVCC) mechanism. Logs and base files are compacted to produce new file slices, and unused and older file slices are cleared to reclaim space on DFS. Hudi provides efficient upserts by mapping a given hoodie key (record key + partition path) to a file group through an indexing mechanism. Once the first version of a record is written to a file, the mapping between the record key and the file group or file ID never changes. In other words, a mapped file group contains all versions of a group of records.

Storage Types

Hudi supports the following storage types:
Copy on Write: Data is stored only in a columnar file format, such as Parquet. The version is updated and the file rewritten by performing a synchronous merge during the data write.
Merge on Read: Data is stored in both columnar (such as Parquet) and row-based (such as Avro) file formats. Updates are recorded in incremental files, which are synchronously or asynchronously compacted to produce new versions of columnar files.
The following table summarizes the trade-offs between these two storage types:

Trade-Off	Copy on Write	Merge on Read
Data latency	Higher	Lower
Update cost (I/O)	Higher (rewrite the entire Parquet file)	Lower (append to incremental logs)
Parquet file size	Smaller (high update (I/O))	Larger (low update cost)
Write amplification	Higher	Lower (depending on compaction strategies)

EMR Underlying Storage Supported by Hudi

HDFS
COS

Installing Hudi

Go to the EMR purchase page, select **EMR-V2.2.0** for **Product Version**, and select the **hudi 0.5.1** optional component. By default, Hudi is installed on the master and router nodes.
Note:

©2013-2024 Tencent Cloud. All rights reserved.

Hudi relies on Hive and Spark. If you install Hudi, EMR automatically installs Hive and Spark.

Examples

The following examples are applicable to Hudi 0.11.0 and later. For more information about examples applicable to other versions, see the Docker Demo on Hudi's official website.

1. Log in to the master node and switch to the `hadoop` user.

2. Load the Spark configuration.

```
cd /usr/local/service/hudi
ln -s /usr/local/service/spark/conf/spark-defaults.conf
/usr/local/service/hudi/demo/config/spark-defaults.conf
```

Upload the configuration to HDFS:

```
hdfs dfs -mkdir -p /hudi/config
hdfs dfs -copyFromLocal demo/config/* /hudi/config/
```

3. Modify the Kafka data source.

```
/usr/local/service/hudi/demo/config/kafka-source.properties
bootstrap.servers=kafka_ip:kafka_port
```

Upload the first batch of data:

```
cat demo/data/batch_1.json | kafkacat -b [kafka_ip] -t stock_ticks -P
```

4. Ingest the first batch of data.

```
spark-submit --class
org.apache.hudi.utilities.deltastreamer.HoodieDeltaStreamer --master yarn
./hudi-utilities-bundle_2.11-0.5.1-incubating.jar   --table-type COPY_ON_WRITE
--source-class org.apache.hudi.utilities.sources.JsonKafkaSource --source-
ordering-field ts  --target-base-path /usr/hive/warehouse/stock_ticks_cow --
target-table stock_ticks_cow --props /hudi/config/kafka-source.properties --
schemaprovider-class org.apache.hudi.utilities.schema.FilebasedSchemaProvider
spark-submit --class
org.apache.hudi.utilities.deltastreamer.HoodieDeltaStreamer  --master yarn
./hudi-utilities-bundle_2.11-0.5.1-incubating.jar  --table-type MERGE_ON_READ -
-source-class org.apache.hudi.utilities.sources.JsonKafkaSource --source-
ordering-field ts  --target-base-path /usr/hive/warehouse/stock_ticks_mor --
target-table stock_ticks_mor --props /hudi/config/kafka-source.properties --
schemaprovider-class org.apache.hudi.utilities.schema.FilebasedSchemaProvider -
-disable-compaction
```

©2013-2024 Tencent Cloud. All rights reserved.

5. View the HDFS data.

```
 hdfs dfs -ls /usr/hive/warehouse/
```

6. Synchronize the Hive metadata.

```
bin/run_sync_tool.sh  --jdbc-url jdbc:hive2://[hiveserver2_ip:hiveserver2_port]
--user hadoop --pass [password] --partitioned-by dt --base-path
/usr/hive/warehouse/stock_ticks_cow --database default --table stock_ticks_cow
bin/run_sync_tool.sh  --jdbc-url jdbc:hive2://[hiveserver2_ip:hiveserver2_port]
--user hadoop --pass [password]--partitioned-by dt --base-path
/usr/hive/warehouse/stock_ticks_mor --database default --table stock_ticks_mor
--skip-ro-suffix
```

7. Use a compute engine to query data.

Hive engine

```
beeline -u jdbc:hive2://[hiveserver2_ip:hiveserver2_port] -n hadoop --hiveconf
hive.input.format=org.apache.hadoop.hive.ql.io.HiveInputFormat --hiveconf
hive.stats.autogather=false
```

Or Spark engine

```
spark-sql --master yarn --conf spark.sql.hive.convertMetastoreParquet=false
```

Execute the following SQL statements in the Hive or Spark engine:

```
select symbol, max(ts) from stock_ticks_cow group by symbol HAVING symbol =
'GOOG';
select `_hoodie_commit_time`, symbol, ts, volume, open, close  from
stock_ticks_cow where  symbol = 'GOOG';
select symbol, max(ts) from stock_ticks_mor group by symbol HAVING symbol =
'GOOG';
select `_hoodie_commit_time`, symbol, ts, volume, open, close  from
stock_ticks_mor where  symbol = 'GOOG';
select symbol, max(ts) from stock_ticks_mor_rt group by symbol HAVING symbol =
'GOOG';
select `_hoodie_commit_time`, symbol, ts, volume, open, close  from
stock_ticks_mor_rt where  symbol = 'GOOG';
```

Presto engine

```
/usr/local/service/presto-client/presto --server localhost:9000 --catalog hive
--schema default --user Hadoop
```

To query a field with underscores (_) in the Presto engine, you must enclose the field with double quotation marks(" "). Example: `"_hoodie_commit_time"` . Execute the following SQL statements:

©2013-2024 Tencent Cloud. All rights reserved.

```
select symbol, max(ts) from stock_ticks_cow group by symbol HAVING symbol =
'GOOG';
select "_hoodie_commit_time", symbol, ts, volume, open, close  from
stock_ticks_cow where  symbol = 'GOOG';
select symbol, max(ts) from stock_ticks_mor group by symbol HAVING symbol =
'GOOG';
select "_hoodie_commit_time", symbol, ts, volume, open, close  from
stock_ticks_mor where  symbol = 'GOOG';
select symbol, max(ts) from stock_ticks_mor_rt group by symbol HAVING symbol =
'GOOG';
```

8. Upload the second batch of data.

```
cat demo/data/batch_2.json | kafkacat -b 10.0.1.70 -t stock_ticks -P
```

9. Ingest the second batch of incremental data.

```
spark-submit --class
org.apache.hudi.utilities.deltastreamer.HoodieDeltaStreamer --master yarn
./hudi-utilities-bundle_2.11-0.5.1-incubating.jar   --table-type COPY_ON_WRITE
--source-class org.apache.hudi.utilities.sources.JsonKafkaSource --source-
ordering-field ts  --target-base-path /usr/hive/warehouse/stock_ticks_cow --
target-table stock_ticks_cow --props /hudi/config/kafka-source.properties --
schemaprovider-class org.apache.hudi.utilities.schema.FilebasedSchemaProvider
spark-submit --class
org.apache.hudi.utilities.deltastreamer.HoodieDeltaStreamer  --master yarn
./hudi-utilities-bundle_2.11-0.5.1-incubating.jar  --table-type MERGE_ON_READ -
-source-class org.apache.hudi.utilities.sources.JsonKafkaSource --source-
ordering-field ts  --target-base-path /usr/hive/warehouse/stock_ticks_mor --
target-table stock_ticks_mor --props /hudi/config/kafka-source.properties --
schemaprovider-class org.apache.hudi.utilities.schema.FilebasedSchemaProvider -
-disable-compaction
```

10. Query the incremental data. The query method is the same as step 7.

11. Use the hudi-cli tool.

```
 cli/bin/hudi-cli.sh
connect --path /usr/hive/warehouse/stock_ticks_mor
compactions show all
compaction schedule
Compact execution plans.
 compaction run --compactionInstant [requestID] --parallelism 2 --sparkMemory
1G  --schemaFilePath /hudi/config/schema.avsc --retry 1
```

12. Specify Tez or Spark as the query engine.

©2013-2024 Tencent Cloud. All rights reserved.

```
beeline -u jdbc:hive2://[hiveserver2_ip:hiveserver2_port] -n hadoop --hiveconf
hive.input.format=org.apache.hadoop.hive.ql.io.HiveInputFormat --hiveconf
hive.stats.autogather=false
set hive.execution.engine=tez;
set hive.execution.engine=spark;
```

Execute the SQL query. For more information about the query method, see step 7.

Using Hudi with COS

Like HDFS, when using Hudi with COS, you must add `cosn://[bucket]` before the storage path. Example:

```
bin/kafka-server-start.sh config/server.properties &
cat     demo/data/batch_1.json | kafkacat -b kafkaip -t stock_ticks -P
cat     demo/data/batch_2.json | kafkacat -b kafkaip -t stock_ticks -P
kafkacat -b kafkaip  -L
hdfs dfs -mkdir -p cosn://[bucket]/hudi/config
hdfs dfs -copyFromLocal demo/config/*  cosn://[bucket]/hudi/config/

spark-submit --class
org.apache.hudi.utilities.deltastreamer.HoodieDeltaStreamer --master yarn
./hudi-utilities-bundle_2.11-0.5.1-incubating.jar   --table-type COPY_ON_WRITE
--source-class org.apache.hudi.utilities.sources.JsonKafkaSource --source-
ordering-field ts  --target-base-path
cosn://[bucket]/usr/hive/warehouse/stock_ticks_cow --target-table
stock_ticks_cow --props cosn://[bucket]/hudi/config/kafka-source.properties --
schemaprovider-class org.apache.hudi.utilities.schema.FilebasedSchemaProvider

spark-submit --class
org.apache.hudi.utilities.deltastreamer.HoodieDeltaStreamer  --master yarn
./hudi-utilities-bundle_2.11-0.5.1-incubating.jar  --table-type MERGE_ON_READ -
-source-class org.apache.hudi.utilities.sources.JsonKafkaSource --source-
ordering-field ts  --target-base-path
cosn://[bucket]/usr/hive/warehouse/stock_ticks_mor --target-table
stock_ticks_mor --props cosn://[bucket]/hudi/config/kafka-source.properties --
schemaprovider-class org.apache.hudi.utilities.schema.FilebasedSchemaProvider -
-disable-compaction

bin/run_sync_tool.sh  --jdbc-url jdbc:hive2://[hiveserver2_ip:hiveserver2_port]
--user hadoop --pass isd@cloud --partitioned-by dt --base-path
cosn://[bucket]/usr/hive/warehouse/stock_ticks_cow --database default --table
stock_ticks_cow
```

©2013-2024 Tencent Cloud. All rights reserved.

```
bin/run_sync_tool.sh  --jdbc-url jdbc:hive2://[hiveserver2_ip:hiveserver2_port]
--user hadoop --pass hive --partitioned-by dt --base-path
cosn://[bucket]/usr/hive/warehouse/stock_ticks_mor --database default --table
stock_ticks_mor --skip-ro-suffix


beeline -u jdbc:hive2://[hiveserver2_ip:hiveserver2_port] -n hadoop --hiveconf
hive.input.format=org.apache.hadoop.hive.ql.io.HiveInputFormat --hiveconf
hive.stats.autogather=false

spark-sql --master yarn --conf spark.sql.hive.convertMetastoreParquet=false

hivesqls:
select symbol, max(ts) from stock_ticks_cow group by symbol HAVING symbol =
'GOOG';
select `_hoodie_commit_time`, symbol, ts, volume, open, close  from
stock_ticks_cow where  symbol = 'GOOG';
select symbol, max(ts) from stock_ticks_mor group by symbol HAVING symbol =
'GOOG';
select `_hoodie_commit_time`, symbol, ts, volume, open, close  from
stock_ticks_mor where  symbol = 'GOOG';
select symbol, max(ts) from stock_ticks_mor_rt group by symbol HAVING symbol =
'GOOG';
select `_hoodie_commit_time`, symbol, ts, volume, open, close  from
stock_ticks_mor_rt where  symbol = 'GOOG';

prestosqls:
/usr/local/service/presto-client/presto --server localhost:9000 --catalog hive
--schema default --user Hadoop
select symbol, max(ts) from stock_ticks_cow group by symbol HAVING symbol =
'GOOG';
select "_hoodie_commit_time", symbol, ts, volume, open, close  from
stock_ticks_cow where  symbol = 'GOOG';
select symbol, max(ts) from stock_ticks_mor group by symbol HAVING symbol =
'GOOG';
select "_hoodie_commit_time", symbol, ts, volume, open, close  from
stock_ticks_mor where  symbol = 'GOOG';
select symbol, max(ts) from stock_ticks_mor_rt group by symbol HAVING symbol =
'GOOG';
select "_hoodie_commit_time", symbol, ts, volume, open, close  from
stock_ticks_mor_rt where  symbol = 'GOOG';

cli/bin/hudi-cli.sh
connect --path cosn://[bucket]/usr/hive/warehouse/stock_ticks_mor
compactions show all
compaction schedule
```

©2013-2024 Tencent Cloud. All rights reserved.

```
compaction run --compactionInstant [requestid]  --parallelism 2 --sparkMemory
1G  --schemaFilePath cosn://[bucket]/hudi/config/schema.avsc --retry 1
```

©2013-2024 Tencent Cloud. All rights reserved.

Superset Development Guide
Superset Overview

Last updated：2025-03-05 16:41:12

Apache Superset is a web-based data browsing and visualization application. Superset on EMR supports MySQL, Hive, Presto, Impala, Kylin, Druid, and ClickHouse.

Superset Features

Supports almost all major databases such as MySQL, PostgresSQL, Oracle, SQL Server, SQLite, and Spark SQL as well as Druid.
Provides a wide variety of visual displays and allows you to create custom dashboards.
Makes data display controllable and enables customization of displayed fields, aggregated data, and data sources.

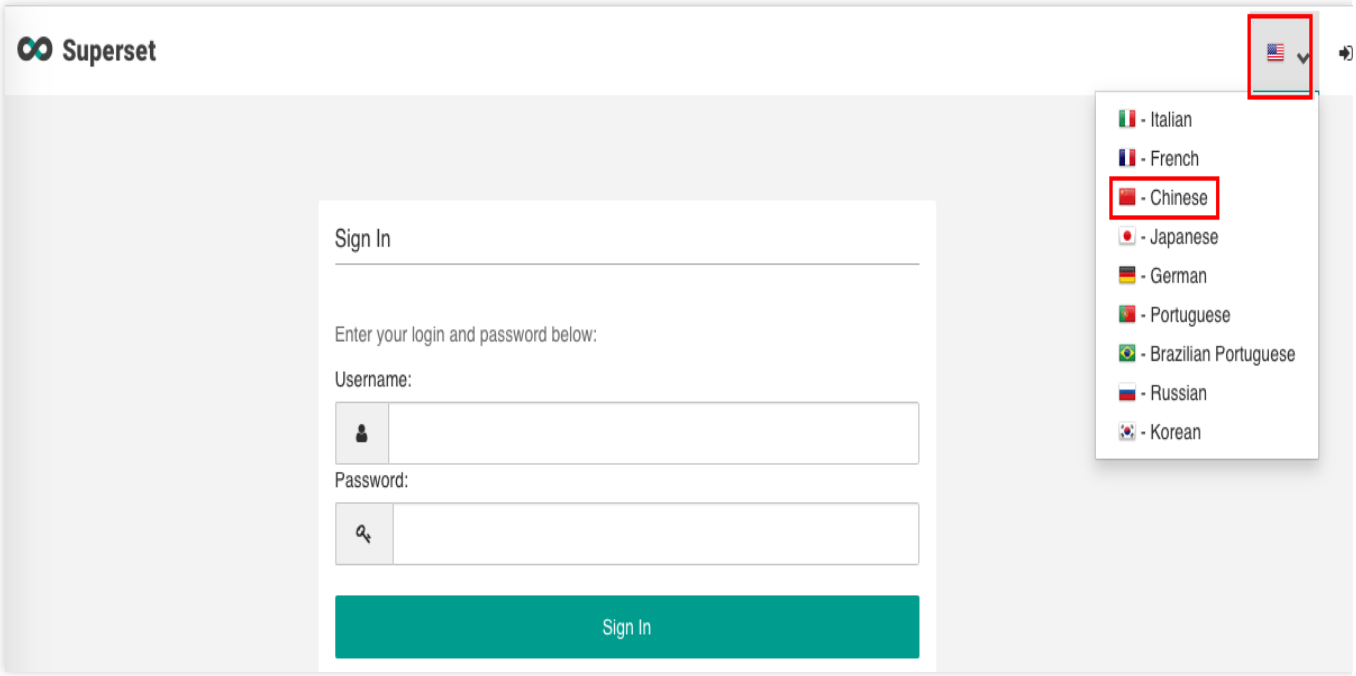
Prerequisites

1. You have created an EMR Hadoop or Druid cluster and selected the Superset service. For more information, see Creating EMR Cluster.
2. By default, Superset is installed on the master node of your cluster. Enable the security group policy for the master node and make sure that your network can access port 18088 of the master node.

Login

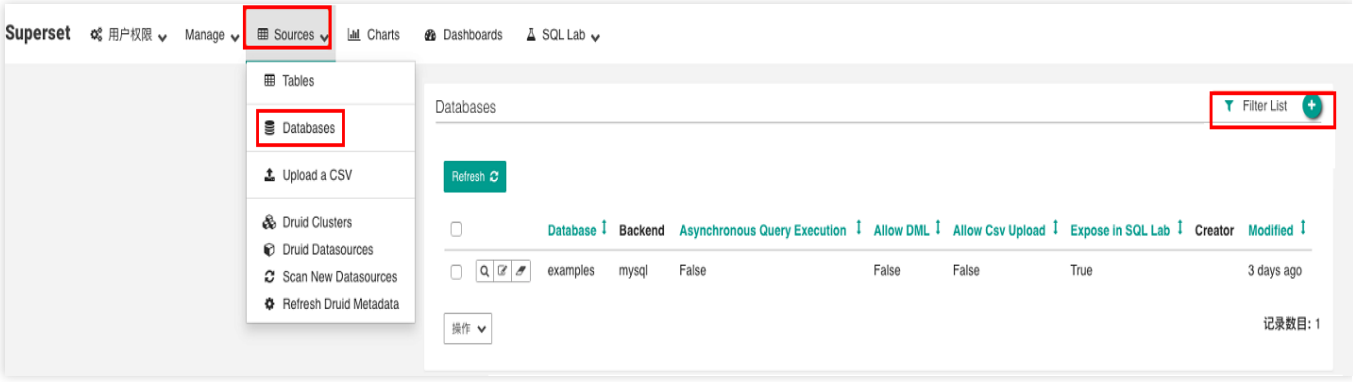
Enter `http://${master_ip}:18088` in your browser (or go to the **EMR console** > **Cluster Service**) to open the login page of Supserset. The default username is `admin`, and the password is the one you set when creating the cluster.

©2013-2024 Tencent Cloud. All rights reserved.

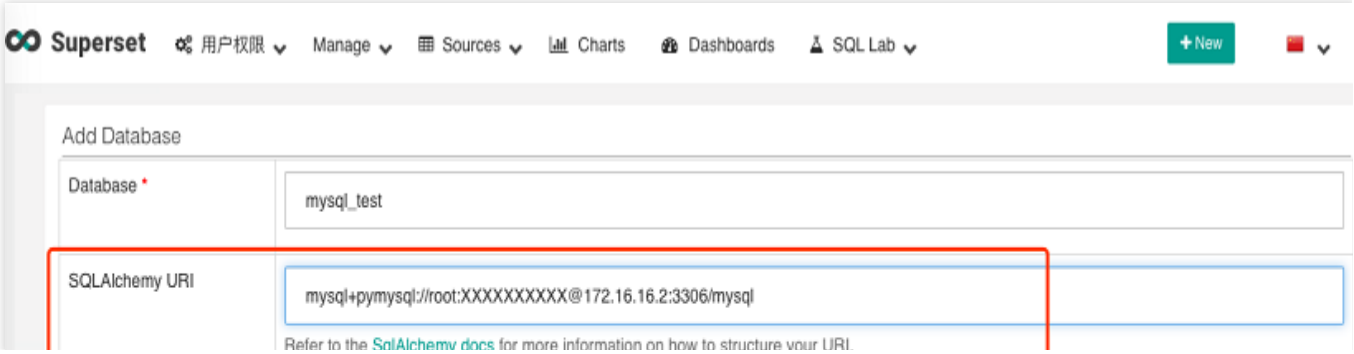


Adding Databases

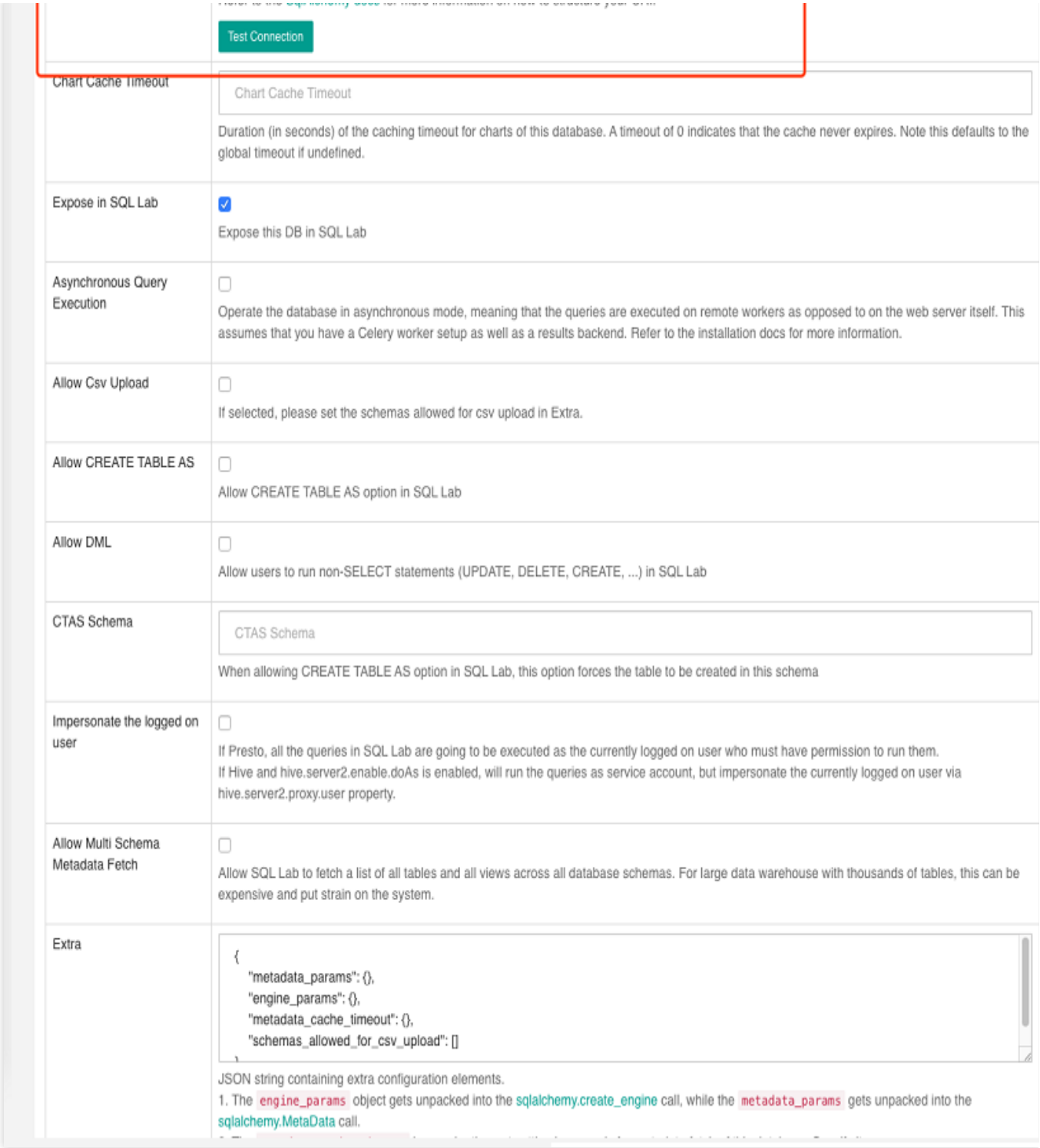
Go to **Sources** > **Databases** and click **Filter List**.



On the following page, add the URI of the component to be added in **SQLAlchemy URI**.



©2013-2024 Tencent Cloud. All rights reserved.



The SQLAlchemy URI for each database is as follows:

Name	SQLAlchemy URI	Remarks
MySQL	mysql+pymysql://:@:/	mysqlname: Username used to connect to MySQL.password: MySQL password.your_databas The MySQL database to be connected to.
Hive	hive://hadoop@<master_ip>:7001/default?auth=NONE	master_ip: Master IP of the EMR cluster.

©2013-2024 Tencent Cloud. All rights reserved.

Presto	presto://hive@:9000/hive/	Master_ip: master_ip of the EMR cluster hive_db_name: Name of the database in Hive. If this parameter is left empty, it will be default by default
Impala	impala://:27000	core_ip: core IP of EMR cluster.
Kylin	kylin://:@:16500/	kylin_user: Kylin username password: Kylin password master_ip: master_ip of the EMR cluster kylin_project: Kylin project
ClickHouse	clickhouse://:@:8123/	clickhouse://default:password@localhost:8123/defa user_name: Username password: Password clickhouse-server-endpoint: ClickHouse service endpoint database_name: Name of the database to be accessed

Adding New Databases on Your Own

Superset supports databases. To install another database, follow the steps below:

1. Log in to the server where the master node of EMR cluster resides.
2. Run the `source /usr/local/service/superset/bin/activate` command.
3. Install the corresponding Python library with pip3.
4. Restart Superset.

©2013-2024 Tencent Cloud. All rights reserved.

Impala Development Guide
Impala Overview

Last updated：2025-01-03 15:02:25

Apache Impala provides high-performance and low-latency SQL queries on data stored in Apache Hadoop file formats. Its fast response to queries enables interactive exploration and fine-tuning of analytical queries rather than long batch jobs traditionally associated with SQL-on-Hadoop technologies.
Impala differs from Hive in that Hive uses the MapReduce engine for execution and involves batch processing, while Impala streams intermediate results over the internet instead of writing them to the disk, which greatly reduces the I/O overheads of nodes.
Impala integrates with Apache Hive database to share databases and tables between both components. The high level of integration with Hive and compatibility with the HiveQL syntax enable you to use either Impala or Hive to create tables, initiate queries, load data, and do more.

Prerequisites

Confirm that you have activated Tencent Cloud and created an EMR cluster. When creating the EMR cluster, select the Impala component on the software configuration page.
Impala is installed in the `/data/Impala` directory of the CVM instance for the EMR cluster.

Data Preparations

Log in to any node (preferably a master one) in the EMR cluster first. For more information on how to log in to EMR, see Logging In To Linux Instance (Web Shell). You can choose to log in with WebShell. Click **Login** on the right of the desired CVM instance to enter the login page. The default username is `root`, and the password is the one you set when creating the EMR cluster. Once the correct credentials are entered, you can enter the command line interface.
Run the following commands to switch to the Hadoop user and go to the Impala folder:

```
[root@10 ~]# su hadoop
[hadoop@10 root]$ cd /data/Impala/
```

Create a bash script file named `gen_data.sh` and add the following code to it:

```
#!/bin/bash
MAXROW=1000000 # Specify the number of data rows to be generated
for((i = 0; i < $MAXROW; i++))
do
```

©2013-2024 Tencent Cloud. All rights reserved.

```
        echo $RANDOM, \\"$RANDOM\\"
done
```

Then, run the following command:

```
[hadoop@10 ~]$ ./gen_data.sh > impala_test.data
```

This script file will generate 1,000,000 random number pairs and save them to the `impala_test.data` file. Then, upload the generated test data to HDFS and run the following commands:

```
[hadoop@10 ~]$ hdfspath="/impala_test_dir"
[hadoop@10 ~]$ hdfs dfs -mkdir $hdfspath
[hadoop@10 ~]$ hdfs dfs -put ./impala_test.data $hdfspath
```

Here, `$hdfspath` is the path of your file in HDFS. Finally, you can run the following command to verify whether the data has been properly put in HDFS.

```
[hadoop@10 ~]$ hdfs dfs -ls $hdfspath
```

Basic Impala Operations

The `impala-shell` path varies by the community component API protocol and default path of the Impala version as shown below:

Impala Version	impala-shell Path	Default Communication Port of impala-shell
4.1.0/4.0.0	/data/Impala/shell	27009
3.4.0	/data/Impala/shell	27001
2.10.0	/data/Impala/bin	27001

The following takes Impala 3.4.0 as an example:

Connecting to Impala

Log in to a master node in the EMR cluster, switch to the Hadoop user, go to the Impala directory, and connect to Impala:

```
[root@10 Impala]# cd /data/Impala/shell;./impala-shell -i $core_ip:27001
```

Here, `core_ip` is the IP of the core node of the EMR cluster. The IP of a task node can also be used. After login succeeds, the following will be displayed:

©2013-2024 Tencent Cloud. All rights reserved.

```
Connected to $core_ip:27001
Server version: impalad version 3.4.1-RELEASE RELEASE (build Could not obtain git h
***********************************************************************************
Welcome to the Impala shell.
(Impala Shell 3.4.1-RELEASE (ebled66) built on Tue Nov 20 17:28:10 CST 2021)

The SET command shows the current value of all shell and query options.
***********************************************************************************
[$core_ip:27001] >
```

You can also directly connect to Impala by executing the following statement after logging in to the core node or task node:

```
cd /data/Impala/shell;./impala-shell -i localhost:27001
```

Creating an Impala database

Run the following statement in Impala to view the database:

```
[10.1.0.215:27001] > show databases;
Query: show databases
+------------------+-----------------------------------------------+
| name             | comment                                       |
+------------------+-----------------------------------------------+
| _impala_builtins | System database for Impala builtin functions  |
| default          | Default Hive database                         |
+------------------+-----------------------------------------------+
Fetched 2 row(s) in 0.09s
```

Run the `create` command to create a database:

```
[localhost:27001] > create database experiments;
Query: create database experiments
Fetched 0 row(s) in 0.41s
```

Run the `use` command to go to the `test` database you just created:

```
[localhost:27001] > use experiments;
Query: use experiments
```

View the current database and execute the following statement:

```
select current_database();
```

Creating an Impala table

©2013-2024 Tencent Cloud. All rights reserved.

Run the `create` command to create an internal table named `impala_test` in the `experiments` database:

```
[localhost:27001] > create table t1 (a int, b string) ROW FORMAT DELIMITED FIELDS T
Query: create table t1 (a int, b string)
Fetched 0 row(s) in 0.13s
```

View all tables:

```
[localhost:27001] > show tables;
Query: show tables
+------+
| name |
+------+
| t1   |
+------+
Fetched 1 row(s) in 0.01s
```

View the table structure:

```
[localhost:27001] > desc t1;
Query: describe t1
+------+--------+---------+
| name | type   | comment |
+------+--------+---------+
| a    | int    |         |
| b    | string |         |
+------+--------+---------+
Fetched 2 row(s) in 0.01s
```

Importing data into a table

For data stored in HDFS, run the following command to import it into the table:

```
LOAD DATA INPATH '$hdfspath/impala_test.data' INTO TABLE t1;
```

Here, `$hdfspath` is the path of your file in HDFS. After the import is completed, the source data file in the import path in HDFS will be deleted and then stored in the `/usr/hive/warehouse/experiments.db/t1` path of the Impala internal table. You can also create an external table by running the following statement:

Note:

There is only one command. If you do not enter the semicolon ";", you can put one command in multiple lines for input.

```
CREATE EXTERNAL TABLE t2
(
   a INT,
   b string
)
```

©2013-2024 Tencent Cloud. All rights reserved.

```
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/impala_test_dir';
```

Running a query

```
[localhost:27001] > select count(*) from experiments.t1;
Query: select count(*) from experiments.t1
Query submitted at: 2019-03-01 11:20:20 (Coordinator: http://10.1.0.215:20004)
Query progress can be monitored at: http://10.1.0.215:20004/query_plan?query_id=f14
+----------+
| count(*) |
+----------+
| 1000000  |
+----------+
Fetched 1 row(s) in 0.63s
```

The final output is 1000000.

Deleting a table

```
[localhost:27001] > drop table experiments.t1;
Query: drop table experiments.t1
```

For more information on Impala operations, see the official documentation.

Connecting to Impala Through JDBC

Impala can also be connected through Java code by following the steps similar to those described in Connecting to Hive Through Java.

The only difference is `$hs2host` and `$hsport`, where `$hs2host` is the IP of any core node or task node in the EMR cluster and `$hsport` can be viewed in the `conf/impalad.flgs` configuration file under the Impala directory of the corresponding node.

```
[root@10 ~]# su hadoop
[hadoop@10 root]$ cd /data/Impala/
[hadoop@10 Impala]$ grep hs2_port conf/impalad.flgs
```

Mapping an HBase Table

©2013-2024 Tencent Cloud. All rights reserved.

Impala uses Hive metadata, and all tables in Hive can be read in Impala. For more information on how to map an HBase table in Impala, see Mapping HBase Tables.

©2013-2024 Tencent Cloud. All rights reserved.

Impala OPS Manual

Last updated：2025-01-03 15:02:25

Impala failed to start as the data volume increased

Background

When there is too much metadata (such as hundreds of databases or tens of thousands of tables) in Impala, Impala needs to broadcast such metadata to all nodes when starting, with a timeout period of 10 seconds by default. If there is a large amount of metadata and the broadcasting is easy to trigger, you can set `-statestore_subscriber_timeout_seconds=100` in the `/data/Impala/conf/impalad.flgs` launch configuration file to fix this problem.

Troubleshooting

Generally, when this issue occurs, the following content will appear in the Impala log at `/data/emr/impala/logs`：

```
Connection with state-store lost
Trying to re-register with state-store
```

Impala queries are slow due to a low configuration

Although Impala is not an in-memory database, it is still necessary to allocate more physical memory to Impala when dealing with large tables or high volumes of data. You are generally recommended to use a memory of 128 GB or more and allocate 80% of it to the Impala process.

A SELECT statement failed

Possible reasons:
1. Timeout was caused by a performance, capacity, or network issue with a particular node. View the Impala log to identify the node and check whether the problem persists after changing the node network.
2. Automatic cancellation of queries was caused due to excessive memory usage by `join` queries. Check whether the `join` statement is appropriate or increase the server memory.
3. The way how a node generates native code to process a specific `WHERE` clause in a query was incorrect, such as server instructions that are not supported by the processor that can generate a specific node. If the error message in

©2013-2024 Tencent Cloud. All rights reserved.

the log indicates that the cause is an invalid instruction, please consider disabling native code generation before trying a query again.

4. The input data format is incorrect, such as text data files with very long lines or delimiters that do not match the characters specified in the `FIELDS TERMINATED BY` clause of the `CREATE TABLE` statement. Check whether there is extra-long data and whether correct delimiters are used in the `CREATE TABLE` statement.

Setting a limit on the memory usage of queries

```
[localhost:27001] > set mem_limit=3000000000;
MEM_LIMIT set to 3000000000
[localhost:27001] > select 5;
Query: select 5
+---+ |5 | +---+ |5 | +---+
[localhost:27001] > set mem_limit=3g;
MEM_LIMIT set to 3g
[localhost:27001] > select 5;
Query: select 5
+---+ |5 | +---+ |5 | +---+
[localhost:27001] > set mem_limit=3gb;
MEM_LIMIT set to 3gb
[localhost:27001] > select 5;
+---+
|5 | +---+ |5 | +---+
[localhost:27001] > set mem_limit=3m;
MEM_LIMIT set to 3m
[localhost:27001] > select 5;
+---+
|5 |
+---+
|5 |
+---+
[localhost:27001] > set mem_limit=3mb; MEM_LIMIT set to 3mb [localhost:21000] >
select 5;
+---+ |5 | +---+
```

©2013-2024 Tencent Cloud. All rights reserved.

Analyzing Data on COS/CHDFS

Last updated：2025-01-03 15:02:25

This document describes more ways to use Impala based on COS with data from direct data insertion and COS.

Development Preparations

1. This task requires access to COS, so you need to create a bucket in COS first.
2. Create an EMR cluster. When creating the EMR cluster, you need to select the Impala component on the software configuration page and enable access to COS on the basic configuration page.
3. Relevant software programs such as Impala are installed in the `/usr/local/service/` directory of the CVM instance for the EMR cluster.

Directions

Log in to any node (preferably a master one) in the EMR cluster first. For more information on how to log in to EMR, see Logging in to Linux Instance Using Standard Login Method. You can choose to log in with WebShell. Click **Login** on the right of the desired CVM instance to enter the login page. The default username is `root` , and the password is the one you set when creating the EMR cluster. Once the correct information is entered, you can enter the command line interface.
Run the following commands on the EMR command line to switch to the Hadoop user and connect to Impala:

```
[root@172 ~]# su hadoop
[hadoop@172 ~]$ impala-shel.sh -i $host:27001
```

`$host` is the private IP of your Impala data node.

Step 1. Create a table (record)

```
[$host:27001 ] > create table record(id int, name string) row format delimited
fields terminated by ',' stored as textfile location 'cosn://$bucketname/';
Query: create table record(id int, name string) row format delimited fields
terminated by ',' stored as textfile location 'cosn://$bucketname/'
Fetched 0 row(s) in 3.07s
Here, `$bucketname` is the name plus path of your COS bucket. If you use CHDFS,
replace the `location` value with `ofs://$mountname/`, where `$mountname` is
your CHDFS instance mount address plus path.
View the table information and confirm that `location` is the COS path.
[$host:27001 ] > show create table record2;
```

©2013-2024 Tencent Cloud. All rights reserved.

```
Query: show create table record2
+------------------------------------------------------------------------------+
| result                                                                       |
+------------------------------------------------------------------------------+
| CREATE TABLE default.record2 (                                               |
|   id INT,                                                                    |
|   name STRING                                                                |
| )                                                                            |
| ROW FORMAT DELIMITED FIELDS TERMINATED BY ','                                |
| WITH SERDEPROPERTIES ('field.delim'=',', 'serialization.format'=',')         |
| STORED AS TEXTFILE                                                           |
| LOCATION 'cosn://$bucketname'                                        |
| TBLPROPERTIES ('numFiles'='19', 'totalSize'='1870')                          |
+------------------------------------------------------------------------------+
Fetched 1 row(s) in 5.90s
```

Step 2. Insert data into the table

```
[$host:27001] > insert into record values(1,"test");
Query: insert into record values(1,"test")
Query submitted at: 2020-08-03 11:29:16 (Coordinator: http://$host:27004)
Query progress can be monitored at: http:/$host:27004/query_plan?
query_id=b246d3194efb7a8f:bc60721600000000
Modified 1 row(s) in 0.64s
```

Step 3. Use Impala to query the table

```
[$host:27001] > select * from record;
Query: select * from record
Query submitted at: 2020-08-03 11:29:31 (Coordinator:
http://172.30.1.136:27004)
Query progress can be monitored at: http://$host:27004/query_plan?
query_id=8148da96f8c0d369:4b26432a00000000
+----+---------+
| id | name    |
+----+---------+
| 1  | test    |
+----+---------+
Fetched 1 row(s) in 0.37s
```

©2013-2024 Tencent Cloud. All rights reserved.

Druid Development Guide
Druid Overview

Last updated：2025-01-03 15:02:25

Apache Druid is a distributed data processing system supporting real-time and multi-dimensional online analytical processing (OLAP). It is used to implement quick and interactive query and analysis for large data sets.

Basic Characteristics

Characteristics of Apache Druid:
It supports interactive queries with a subsecond response time and has various features such as multi-dimensional filtering, ad hoc attribute grouping, and quick data aggregation.
It supports highly concurrent and real-time data ingestion to ensure real-timeliness for data ingestion and query.
It features high scalability. With the distributed shared-nothing architecture, it supports quick processing of petabytes of data with hundreds of billions of events and sustains thousands of concurrent queries per second.
It allows simultaneous online queries by multiple tenants.
It supports high availability (HA) and rolling update.

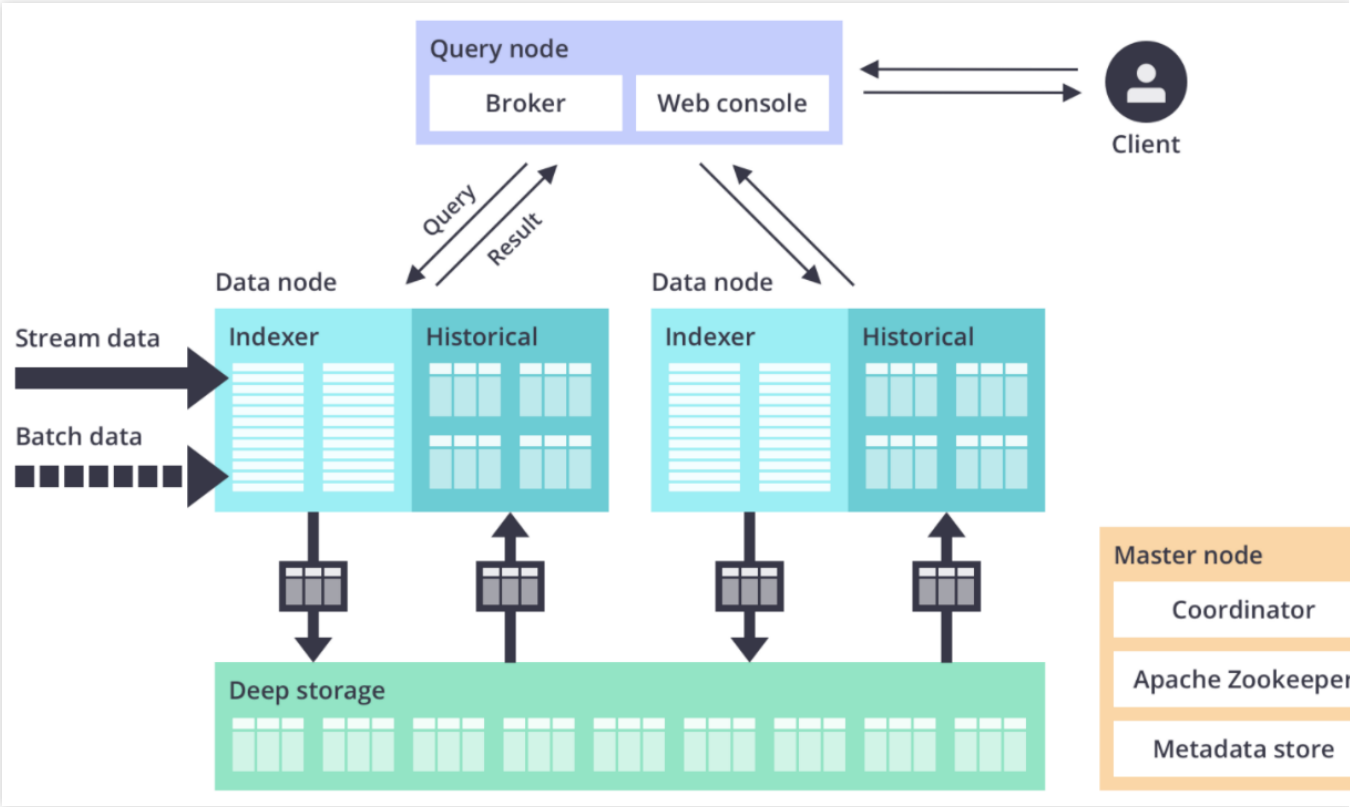
Use Cases

Druid is most frequently used for flexible, quick, multi-dimensional OLAP analysis on big data. In addition, as it supports ingestion of pre-aggregated data and analysis of aggregated data based on timestamps, it is usually used in time-series data processing and analysis, such as ad platform, real-time metric monitoring, recommendation model, and search model.

System and Architecture

Druid uses a microservice-based architecture. All core services in it can be deployed on different hardware devices either separately or jointly.

©2013-2024 Tencent Cloud. All rights reserved.



Enhanced EMR Druid

A lot of improvements have been made on EMR Druid based on Apache Druid, including integration with EMR Hadoop and relevant Tencent Cloud ecosystem, convenient monitoring and OPS, and easy-to-use product APIs, so that you can use it out of the box in an OPS-free manner.

Currently, EMR Druid supports the following features:

Easy integration with EMR Hadoop cluster

Easy and quick elastic scalability

HA

Using COS as deep storage

Using COS file as data source for batch indexing

Metadata storage in TencentDB

Integration with tools such as Superset

Various monitoring metrics and alarm rules

Failover

High security

©2013-2024 Tencent Cloud. All rights reserved.

Druid Usage

Last updated：2025-01-03 15:02:25

EMR allows you to deploy an E-MapReduce Druid cluster as an independent cluster based on the following considerations:
Use case: E-MapReduce Druid can be used without Hadoop to adapt to different business use cases.
Resource preemption: E-MapReduce Druid has high requirements for the memory, especially with the Broker and Historical nodes. The resource usage of E-MapReduce Druid is not scheduled by Hadoop YARN; therefore, resource preemption tends to occur during operations.
Cluster size: As an infrastructure, Hadoop generally has a large size, while E-MapReduce Druid is relatively small. When they are deployed in the same cluster, resources may be wasted due to their different sizes. Therefore, separate deployment is more flexible.

Purchase suggestions

To purchase a Druid cluster, select Druid as the cluster type when creating the EMR cluster. The Druid cluster has built-in Hadoop HDFS and YARN services integrated with Druid, which are recommended for testing only. **We strongly recommend you use a dedicated Hadoop cluster in the production environment.**
To disable the built-in Hadoop services for the Druid cluster, go to the EMR console, select the target service pane on the **Cluster services** page, and click **Operation > Pause service** to suspend the service.

Configuring connectivity between Hadoop and Druid clusters

This section describes how to configure the connectivity between the Hadoop and Druid clusters. If you use the built-in Hadoop cluster in the Druid cluster (which is not recommended for the production environment), they can be properly connected with no additional settings required, and you can skip this section.
If you want to store the index data in the HDFS of another independent Hadoop cluster (which is recommended for the production environment), you need to configure the connectivity between the two clusters in the following steps:
1. Make sure that the Druid and Hadoop clusters can properly communicate with each other.
The two clusters should be in the same VPC. If they are in different VPCs, the two VPCs should be able to communicate with each other (through CCN or Peering Connection, for example).
2. Copy the `core-site.xml`, `hdfs-site.xml`, `yarn-site.xml`, and `mapred-site.xml` files in `/usr/local/service/hadoop/etc/hadoop` in the Hadoop cluster and paste them in `/usr/local/service/druid/conf/druid/_common` on each node in the E-MapReduce Druid cluster.
Note:

©2013-2024 Tencent Cloud. All rights reserved.

As the Druid cluster has a built-in Hadoop cluster, the relevant soft links to the files above already exist in the Druid path. You need to delete them first before copying the configuration files of another Hadoop cluster. In addition, you need to make sure that the file permissions are correct so that the files can be accessed by the `hadoop` user.

3. Modify the `common.runtime.properties` configuration file in Druid configuration management, save the change, and restart the Druid cluster services.

druid.storage.type: It defaults to `hdfs` and does not need to be modified

druid.storage.storageDirectory:

```
If the target Hadoop cluster is non-HA: hdfs://{namenode_ip}:4007
If the target Hadoop cluster is HA: hdfs://HDFSXXXXX
Configure the full path, which can be found in the `fs.defaultFS` configuration
item in the `core-site.xml` file of the target Hadoop cluster.
```

Using COS

E-MapReduce Druid can use COS as the deep storage. This section describes how to configure COS as the deep storage of the Druid cluster.

First, you need to make sure that COS has been activated for both the Druid cluster and the target Hadoop cluster. You can activate COS when purchasing the clusters or configure COS in the EMR console after purchasing them.

1. Modify the `common.runtime.properties` configuration file in Druid configuration management:

druid.storage.type: `hdfs`

druid.storage.storageDirectory: `cosn://{bucket_name}/druid/segments`

You can create the `segments` directory on COS and set its permissions in advance.

2. Modify the `core-site.xml` configuration file in HDFS configuration management:

Set `fs.cosn.impl` to `org.apache.hadoop.fs.CosFileSystem` .

Add a new configuration item `fs.AbstractFileSystem.cosn.impl` and set it to `org.apache.hadoop.fs.CosN` .

3. Put the JAR packages related to hadoop-cos (such as `cos_api-bundle-5.6.69.jar` and `hadoop-cos-2.8.5-8.1.6.jar`) into the `/usr/local/service/druid/extensions/druid-hdfs-storage` , `/usr/local/service/druid/hadoopdependencies/hadoop-client/2.8.5` , and `/usr/local/service/hadoop/share/hadoop/common/lib/` directories on each node of the cluster.

Save the configuration and restart the Druid cluster services.

Modifying Druid parameters

©2013-2024 Tencent Cloud. All rights reserved.

After you create the E-MapReduce Druid cluster, a set of configuration items will be generated automatically. However, we recommend you modify the memory configuration as needed to achieve the optimal performance. You can do so on the [Configurations]https://intl.cloud.tencent.com/document/product/1026/31109) page in the EMR console.
When modifying the configuration, make sure that the modification is correct:

```
MaxDirectMemorySize >= druid.processing.buffer.sizeByte *
(druid.processing.numMergeBuffers + druid.processing.numThreads + 1)
```

Modification suggestion:

```
druid.processing.numMergeBuffers = max(2, druid.processing.numThreads / 4)
druid.processing.numThreads =  Number of cores - 1 (or 1)
druid.server.http.numThreads = max(10, (Number of cores * 17) / 16 + 2) + 30
```

For more information on the configuration, see Configuration reference.

Using a router as a query node

Currently, a Druid cluster deploys the Broker process on the EMR master node by default. As there are many processes deployed on the master node, they may interfere with each other, which may lead to insufficient memory and compromise the query efficiency. In addition, many businesses require that the query nodes and core nodes be separately deployed. In this case, you can add one or more router nodes in the console and install the Broker processes so as to scale out the query nodes of the Druid cluster.

Accessing the web

You can access the Druid cluster in the console through the port 18888 on the master node and configure the public IP on your own. After opening port 18888 in the security group and setting the bandwidth, you can access the cluster at `[http://{masterIp}:18888]()` .

©2013-2024 Tencent Cloud. All rights reserved.

Ingesting Data from Hadoop in Batches

Last updated：2025-01-03 15:02:26

This document describes how to load data files from a remote Hadoop cluster into a Druid cluster in batches. Operations in this document are all performed by the `Hadoop` user; therefore, please switch to the `Hadoop` user in both the Druid and Hadoop clusters.

Loading Data into Druid Cluster in Batches

1. Run the following commands to create directories as the `Hadoop` user in the remote Hadoop cluster:

```
hdfs dfs -mkdir /druid
hdfs dfs -mkdir /druid/segments
hdfs dfs -mkdir /quickstart
hdfs dfs -chmod 777 /druid
hdfs dfs -chmod 777 /druid/segments
hdfs dfs -chmod 777 /quickstart
```

Note:

If the Druid and Hadoop clusters are both self-deployed clusters, the directories need to be created on the corresponding Hadoop cluster (you also need to perform subsequent operations in the correct cluster). If the Druid and Hadoop clusters are the same cluster in the testing environment, you can perform the operations in the same cluster.

2. Upload the testing package.

The Druid cluster comes with a sample dataset named `Wikiticker` (located in `/usr/local/service/druid/quickstart/tutorial/wikiticker-2015-09-12-sampled.json.gz` by default). The operation of uploading the dataset in the Druid cluster to the corresponding remote Hadoop cluster **should be performed on the remote Hadoop cluster.**

```
hdfs dfs -put wikiticker-2015-09-12-sampled.json.gz /quickstart/wikiticker-
2015-09-12-sampled.json.gz
```

3. Compile the index file.

Prepare an index file by running the following commands, which is still the Druid cluster's sample file `/usr/local/service/druid/quickstart/tutorial/wikipedia-index-hadoop.json`:

```
{
  "type" : "index_hadoop",
  "spec" : {
 "dataSchema" : {
```

©2013-2024 Tencent Cloud. All rights reserved.

```
    "dataSource" : "wikipedia",
    "parser" : {
      "type" : "hadoopyString",
      "parseSpec" : {
        "format" : "json",
        "dimensionsSpec" : {
          "dimensions" : [
            "channel",
            "cityName",
            "comment",
            "countryIsoCode",
            "countryName",
            "isAnonymous",
            "isMinor",
            "isNew",
            "isRobot",
            "isUnpatrolled",
            "metroCode",
            "namespace",
            "page",
            "regionIsoCode",
            "regionName",
            "user",
            { "name": "added", "type": "long" },
            { "name": "deleted", "type": "long" },
            { "name": "delta", "type": "long" }
          ]
        },
        "timestampSpec" : {
          "format" : "auto",
          "column" : "time"
        }
      }
    },
    "metricsSpec" : [],
    "granularitySpec" : {
      "type" : "uniform",
      "segmentGranularity" : "day",
      "queryGranularity" : "none",
      "intervals" : ["2015-09-12/2015-09-13"],
      "rollup" : false
    }
  },
  "ioConfig" : {
    "type" : "hadoop",
    "inputSpec" : {
      "type" : "static",
```

©2013-2024 Tencent Cloud. All rights reserved.

```
      "paths" : "/quickstart/wikiticker-2015-09-12-sampled.json.gz"
    }
  },
  "tuningConfig" : {
    "type" : "hadoop",
    "partitionsSpec" : {
      "type" : "hashed",
      "targetPartitionSize" : 5000000
    },
    "forceExtendableShardSpecs" : true,
    "jobProperties" : {
      "yarn.nodemanager.vmem-check-enabled" : "false",
      "mapreduce.map.java.opts" : "-Duser.timezone=UTC -Dfile.encoding=UTF-8",
      "mapreduce.job.user.classpath.first" : "true",
      "mapreduce.reduce.java.opts" : "-Duser.timezone=UTC -Dfile.encoding=UTF-
8",
      "mapreduce.map.memory.mb" : 1024,
      "mapreduce.reduce.memory.mb" : 1024
    }
  }
   },
   "hadoopDependencyCoordinates": ["org.apache.hadoop:hadoop-client:2.8.5"]
}
```

Note:

`hadoopDependencyCoordinates` is the dependent Hadoop version.

`spec.ioConfig.inputSpec.paths` is the input file path. If you have already set cluster connectivity in the `common.runtime.properties` configuration item, you can use the relative path (for more information, please see Druid Usage); otherwise, you need to use a relative path starting with `hdfs://` or `cosn://` based on the actual conditions.

The `tuningConfig.jobProperties` parameter is used to set MapReduce job parameters.

4. Submit the indexing task.

Submit the task in the Druid cluster to ingest the data. Run the following command as the `Hadoop` user under the Druid directory:

```
./bin/post-index-task --file quickstart/tutorial/wikipedia-index-hadoop.json --
url http://localhost:8090
```

If the command succeeds, an output similar to the one below will be displayed:

```
...
Task finished with status: SUCCESS
Completed indexing data for wikipedia. Now loading indexed data onto the
cluster...
wikipedia loading complete! You may now query your data
```

©2013-2024 Tencent Cloud. All rights reserved.

Data Query

Druid supports SQL-like and native JSON queries as described below. For more information, please see Tutorial: Querying data.

Querying with SQL

Druid supports multiple SQL query methods:

Perform queries in the `Query` menu on the web UI.

```
SELECT page, COUNT(*) AS Edits
FROM wikipedia
WHERE TIMESTAMP '2015-09-12 00:00:00' <= "__time" AND "__time" < TIMESTAMP
'2015-09-13 00:00:00'
GROUP BY page
ORDER BY Edits DESC
LIMIT 10
```

Use the command line tool `bin/dsql` for interactive queries on a query node.

```
[hadoop@172 druid]$ ./bin/dsql
Welcome to dsql, the command-line client for Druid SQL.
Connected to [http://localhost:8082/].
Type "\\h" for help.
dsql> SELECT page, COUNT(*) AS Edits FROM wikipedia WHERE "__time" BETWEEN
TIMESTAMP '2015-09-12 00:00:00' AND TIMESTAMP '2015-09-13 00:00:00' GROUP BY
page ORDER BY Edits DESC LIMIT 10;
┌──────────────────────────────────────────────────────────┬───────┐
| page                                                     | Edits |
├──────────────────────────────────────────────────────────┼───────┤
| Wikipedia:Vandalismusmeldung                             |    33 |
| User:Cyde/List of candidates for speedy deletion/Subpage |    28 |
| Jeremy Corbyn                                            |    27 |
| Wikipedia:Administrators' noticeboard/Incidents          |    21 |
| Flavia Pennetta                                          |    20 |
| Total Drama Presents: The Ridonculous Race               |    18 |
| User talk:Dudeperson176123                               |    18 |
| Wikipédia:Le Bistro/12 septembre 2015                    |    18 |
| Wikipedia:In the news/Candidates                         |    17 |
| Wikipedia:Requests for page protection                   |    17 |
└──────────────────────────────────────────────────────────┴───────┘
Retrieved 10 rows in 0.06s.
```

Submit SQL queries over HTTP.

©2013-2024 Tencent Cloud. All rights reserved.

```
curl -X 'POST' -H 'Content-Type:application/json' -d
@quickstart/tutorial/wikipedia-top-pages-sql.json
http://localhost:18888/druid/v2/sql
```

The formatted output is as follows:

```
[
  {
      "page":"Wikipedia:Vandalismusmeldung",
      "Edits":33
  },
  {
      "page":"User:Cyde/List of candidates for speedy deletion/Subpage",
      "Edits":28
  },
  {
      "page":"Jeremy Corbyn",
      "Edits":27
  },
  {
      "page":"Wikipedia:Administrators' noticeboard/Incidents",
      "Edits":21
  },
  {
      "page":"Flavia Pennetta",
      "Edits":20
  },
  {
      "page":"Total Drama Presents: The Ridonculous Race",
      "Edits":18
  },
  {
      "page":"User talk:Dudeperson176123",
      "Edits":18
  },
  {
      "page":"Wikipédia:Le Bistro/12 septembre 2015",
      "Edits":18
  },
  {
      "page":"Wikipedia:In the news/Candidates",
      "Edits":17
  },
  {
      "page":"Wikipedia:Requests for page protection",
      "Edits":17
```

©2013-2024 Tencent Cloud. All rights reserved.

```
  }
]
```

Querying through native JSON

Directly enter JSON queries in the `Query` menu on the web UI.

```
{
   "queryType" : "topN",
   "dataSource" : "wikipedia",
   "intervals" : ["2015-09-12/2015-09-13"],
   "granularity" : "all",
   "dimension" : "page",
   "metric" : "count",
   "threshold" : 10,
   "aggregations" : [
       {
          "type" : "count",
          "name" : "count"
       }
   ]
}
```

Submit the queries over HTTP under the Druid directory on a query node.

```
curl -X 'POST' -H 'Content-Type:application/json' -d
@quickstart/tutorial/wikipedia-top-pages.json http://localhost:18888/druid/v2?
pretty
```

The output is as follows:

```
[ {
"timestamp" : "2015-09-12T00:46:58.771Z",
"result" : [ {
  "count" : 33,
  "page" : "Wikipedia:Vandalismusmeldung"
}, {
  "count" : 28,
  "page" : "User:Cyde/List of candidates for speedy deletion/Subpage"
}, {
  "count" : 27,
  "page" : "Jeremy Corbyn"
}, {
  "count" : 21,
  "page" : "Wikipedia:Administrators' noticeboard/Incidents"
}, {
  "count" : 20,
```

©2013-2024 Tencent Cloud. All rights reserved.

```
    "page" : "Flavia Pennetta"
  }, {
    "count" : 18,
    "page" : "Total Drama Presents: The Ridonculous Race"
  }, {
    "count" : 18,
    "page" : "User talk:Dudeperson176123"
  }, {
    "count" : 18,
    "page" : "Wikipédia:Le Bistro/12 septembre 2015"
  }, {
    "count" : 17,
    "page" : "Wikipedia:In the news/Candidates"
  }, {
    "count" : 17,
    "page" : "Wikipedia:Requests for page protection"
  } ]
  } ]
```


©2013-2024 Tencent Cloud. All rights reserved.

Ingesting Data from Kafka in Real Time

Last updated：2025-01-03 15:02:25

This document describes how to consume Kafka data in real time by using the Apache Druid Kafka indexing service. Before performing operations described in this document, just like in a Hadoop cluster, you need to make sure that the Kafka and Druid clusters can properly communicate with each other.

Note

The two clusters should be in the same VPC. If they are in different VPCs, the two VPCs should be able to communicate with each other (through CCN or Peering Connection, for example).

If necessary, you should configure the Druid cluster with the host information of the Kafka cluster.

Using Command Line

1. Start the Kafka broker in the Kafka cluster.

```
./bin/kafka-server-start.sh config/server.properties
```

2. Create a Kafka topic named mytopic.

```
./bin/kafka-topics.sh --create --zookeeper {kafka_zk_ip}:2181 --replication-
factor 1 --partitions 1 --topic mytopic
Output:
Created topic "mytopic".
```

`{kafka_zk_ip}:2181` is the ZooKeeper address of the Kafka cluster.

3. Prepare a data description file kafka-mytopic.json in the Druid cluster.

```
{
  "type": "kafka",
  "dataSchema": {
      "dataSource": "mytopic-kafka",
      "parser": {
          "type": "string",
          "parseSpec": {
              "timestampSpec": {
                  "column": "time",
                  "format": "auto"
              },
              "dimensionsSpec": {
                  "dimensions": ["url", "user"]
              },
              "format": "json"
```

©2013-2024 Tencent Cloud. All rights reserved.

```
            }
        },
        "granularitySpec": {
            "type": "uniform",
            "segmentGranularity": "hour",
            "queryGranularity": "none"
        },
        "metricsSpec": [{
                "type": "count",
                "name": "views"
            },
            {
                "name": "latencyMs",
                "type": "doubleSum",
                "fieldName": "latencyMs"
            }
        ]
    },
    "ioConfig": {
        "topic": "mytopic",
        "consumerProperties": {
            "bootstrap.servers": "{kafka_ip}:9092",
            "group.id": "kafka-indexing-service"
        },
        "taskCount": 1,
        "replicas": 1,
        "taskDuration": "PT1H"
    },
    "tuningConfig": {
        "type": "kafka",
        "maxRowsInMemory": "100000"
    }
  }
```

`{kafka_ip}:9092` is the IP and port of the `bootstrap.servers` in your Kafka cluster.

4. Add the Kafka supervisor on the master nodes in the Druid cluster.

```
curl -XPOST -H 'Content-Type: application/json' -d @kafka-mytopic.json
http://{druid_master_ip}:8090/druid/indexer/v1/supervisor
Output:
{"id":"mytopic-kafka"}
```

`{druid_master_ip}:8090` is the node where the `overload` process is deployed, which is generally a master node.

5. Enable a console producer in the Kafka cluster.

©2013-2024 Tencent Cloud. All rights reserved.

```
./bin/kafka-console-producer.sh --broker-list {kafka_ip}:9092 --topic mytopic
```

`{kafka_ip}:9092` is the IP and port of the `bootstrap.servers` in your Kafka cluster.

6. Prepare a query file named `query-mytopic.json` in the Druid cluster.

```
{
  "queryType" : "search",
  "dataSource" : "mytopic-kafka",
  "intervals" : ["2020-03-13T00:00:00.000/2020-03-20T00:00:00.000"],
  "granularity" : "all",
  "searchDimensions": [
      "url",
      "user"
  ],
  "query": {
      "type": "insensitive_contains",
      "value": "roni"
  }
 }
```

7. Enter some data on Kafka in real time.

```
{"time": "2020-03-19T09:57:58Z", "url": "/foo/bar", "user": "brozo",
"latencyMs": 62}
{"time": "2020-03-19T16:57:59Z", "url": "/", "user": "roni", "latencyMs": 15}
{"time": "2020-03-19T17:50:00Z", "url": "/foo/bar", "user": "roni",
"latencyMs": 25}
```

Timestamp generation command:

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

8. Perform a query in the Druid cluster.

```
curl -XPOST -H 'Content-Type: application/json' -d @query-mytopic.json
http://{druid_ip}:8082/druid/v2/?pretty
```

`{druid_ip}:8082` is the broker node of your Druid cluster, which generally resides on a master or router node.

Query result:

```
[ {
  "timestamp" : "2020-03-19T16:00:00.000Z",
  "result" : [ {
 "dimension" : "user",
 "value" : "roni",
```

©2013-2024 Tencent Cloud. All rights reserved.

```
  "count" : 2
   } ]
} ]
```

Using Web-Based Visualization

You can ingest data from a Kafka cluster and perform queries in the Druid Web UI Console in a visualized manner. For detailed directions, please see Loading data with the data loader.

©2013-2024 Tencent Cloud. All rights reserved.

TensorFlow Development Guide
TensorFlow Overview

Last updated：2025-01-03 15:02:25

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state of the art in machine learning and developers easily build and deploy machine learning-powered applications.

Easy model building

Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.

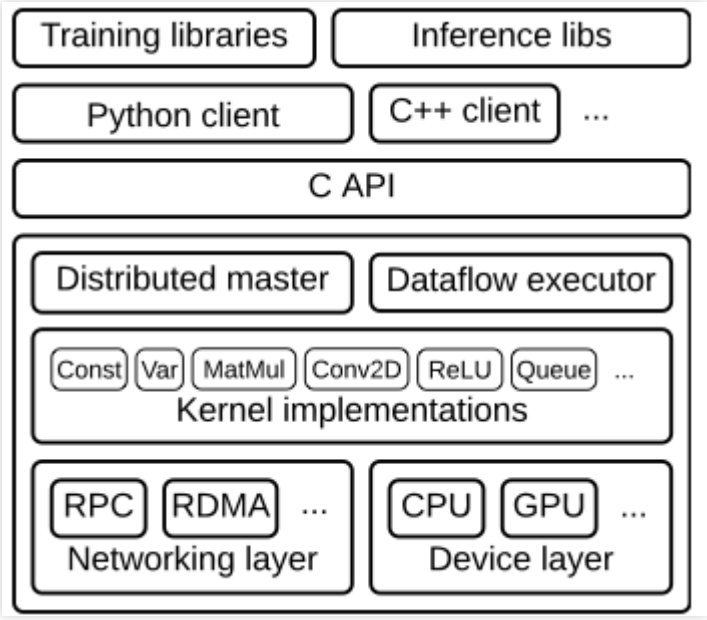
Reliable machine learning production anywhere

Easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language you use.

Powerful experimentation for research

A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

TensorFlow Architecture



Client

It defines the computation process as a data flow graph and initializes graph execution by using `_Session_` .

©2013-2024 Tencent Cloud. All rights reserved.

Distributed master
It prunes specific subgraphs in a graph, i.e, parameters defined in `Session.run()`, partitions a subgraph into multiple parts that run in different processes and devices, and distributes the graph parts to different worker services, which initialize subgraph computation.
Worker service (one for each task)
It schedules the execution of graph operations by using kernel implementations appropriate to the available hardware (CPUs, GPUs, etc.) and sends/receives operation results to/from other worker services.
Kernel implementation
It performs the computation for individual graph operations.

EMR's Support for TensorFlow

TensorFlow version: v1.14.0
Currently, TensorFlow can only run on CPU models instead of GPU models.
TensorFlowOnSpark can be used for distributed training.

TensorFlow Development Sample

Write code: `test.py`

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print sess.run(hello)
a = tf.constant(10)
b = tf.constant(111)
print sess.run(a+b)
exit()
```

Run the following command:

```
python test.py
```

For more usage, please visit the TensorFlow official website.

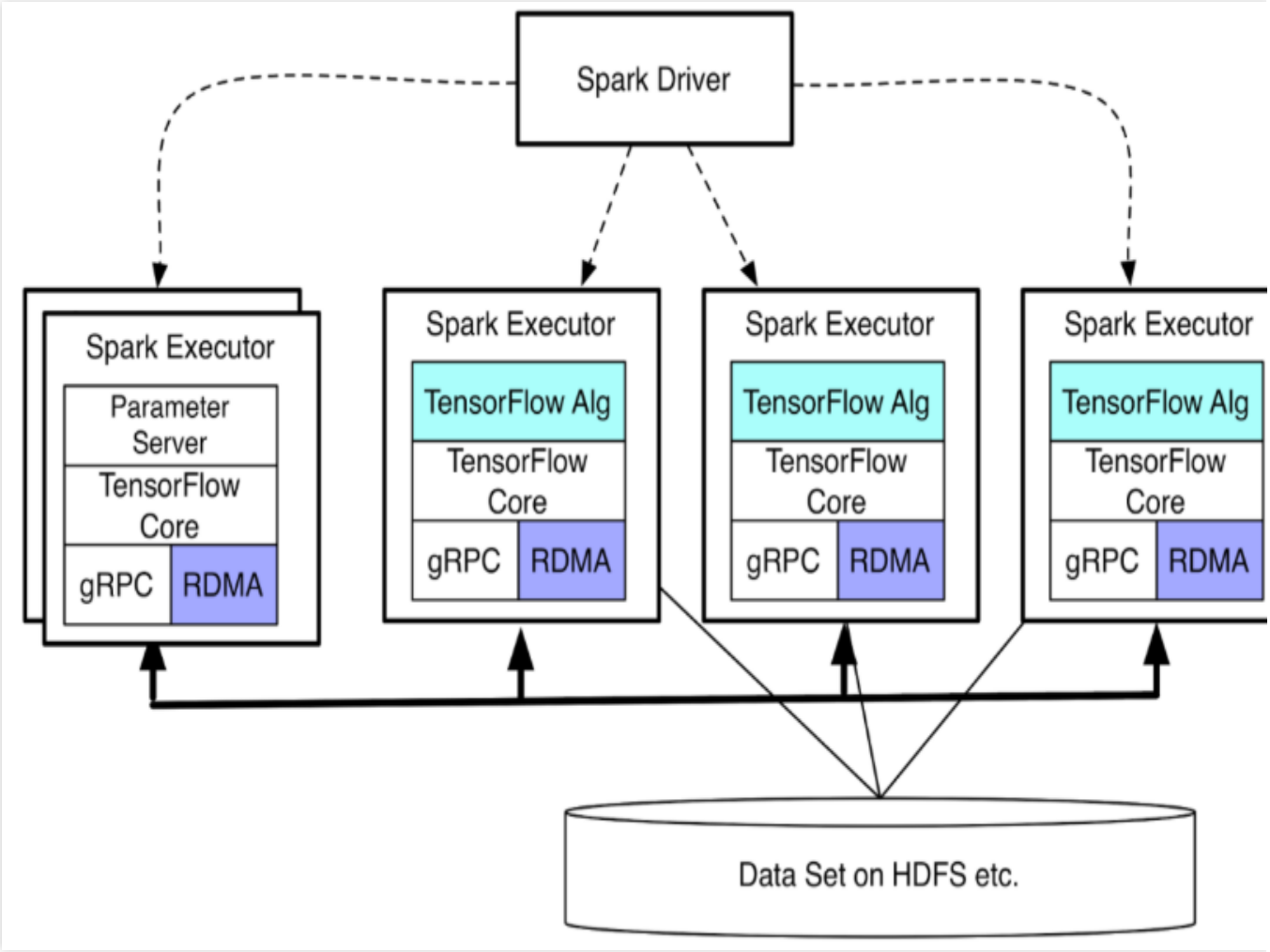
©2013-2024 Tencent Cloud. All rights reserved.

TensorFlowOnSpark Overview

Last updated：2025-01-03 15:02:25

TensorFlowOnSpark brings scalable deep learning to Apache Hadoop and Apache Spark clusters with support for TensorFlow programs in all types, async/sync training and inferencing, model parallelism, and parallel data processing. For more information, please visit TensorFlowOnSpark official website.

TensorFlowOnSpark Architecture Diagram



TensorFlowOnSpark supports direct tensor communication among TensorFlow processes (workers and parameter servers). Process-to-process direct communication enables TensorFlowOnSpark programs to scale easily by adding machines. As TensorFlowOnSpark doesn't involve Spark drivers in tensor communication, it can achieve similar scalability as standalone TensorFlow clusters.

©2013-2024 Tencent Cloud. All rights reserved.

Installing TensorFlowOnSpark

1. Enter the EMR purchase page and select EMR v2.3.0 or above.
2. Select the `tensorflowonspark 1.4.4` component in the **Optional Component** list.
3. TensorFlowOnSpark will be installed in the `/usr/local/service/tensorflowonspark` directory by default.

Note:

The components depended on by TensorFlowOnSpark include Hive, Spark, etc., which will be installed together with TensorFlowOnSpark.

Use Cases

There is complete sample code in the directory of the installed TensorFlowOnSpark component. You can use TensorFlowOnSpark in the following steps:

Download testing data

Run the following command in the `/usr/local/service/tensorflowonspark` directory as the `hadoop` user:

```
sh mnist_download.sh
cat mnist_download.sh
mkdir ${HOME}/mnist
pushd ${HOME}/mnist >/dev/null
curl -O "http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz"
curl -O "http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz"
curl -O "http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz"
curl -O "http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz"
zip -r mnist.zip *
popd >/dev/null
```

Upload the original data and dependency packages

```
hdfs dfs -mkdir -p /mnist/tools/
hdfs dfs -put ~/mnist/mnist.zip /mnist/tools
hdfs dfs -mkdir /tensorflow
hdfs dfs -put TensorFlowOnSpark/tensorflow-hadoop-1.10.0.jar /tensorflow
```

Prepare the feature data

```
sh prepare_mnist.sh
```

You can see that the feature data has been prepared:

©2013-2024 Tencent Cloud. All rights reserved.

```
hdfs dfs -ls /user/hadoop/mnist
Found 2 items
drwxr-xr-x - hadoop supergroup 0 2020-05-21 11:40 /user/hadoop/mnist/csv
drwxr-xr-x - hadoop supergroup 0 2020-05-21 11:41 /user/hadoop/mnist/tfr
```

Train the model based on InputMode.SPARK

```
sh mnist_train_with_spark_cpu.sh
```

View the trained model:

```
[hadoop@10 tensorflow-on-spark]$ hdfs dfs -ls /user/hadoop/mnist_model
Found 10 items
-rw-r--r-- 1 hadoop supergroup 128 2020-05-21 11:46
/user/hadoop/mnist_model/checkpoint
-rw-r--r-- 1 hadoop supergroup 243332 2020-05-21 11:46
/user/hadoop/mnist_model/events.out.tfevents.1590032704.10.0.0.114
-rw-r--r-- 1 hadoop supergroup 164619 2020-05-21 11:45
/user/hadoop/mnist_model/graph.pbtxt
-rw-r--r-- 1 hadoop supergroup 814168 2020-05-21 11:45
/user/hadoop/mnist_model/model.ckpt-0.data-00000-of-00001
-rw-r--r-- 1 hadoop supergroup 375 2020-05-21 11:45
/user/hadoop/mnist_model/model.ckpt-0.index
-rw-r--r-- 1 hadoop supergroup 64658 2020-05-21 11:45
/user/hadoop/mnist_model/model.ckpt-0.meta
-rw-r--r-- 1 hadoop supergroup 814168 2020-05-21 11:46
/user/hadoop/mnist_model/model.ckpt-595.data-00000-of-00001
-rw-r--r-- 1 hadoop supergroup 375 2020-05-21 11:46
/user/hadoop/mnist_model/model.ckpt-595.index
-rw-r--r-- 1 hadoop supergroup 64658 2020-05-21 11:46
/user/hadoop/mnist_model/model.ckpt-595.meta
drwxr-xr-x - hadoop supergroup 0 2020-05-21 11:46
/user/hadoop/mnist_model/train
```

Predict the model based on InputMode.SPARK

```
sh mnist_inference_with_spark_cpu.sh
```

View the prediction result:

```
hdfs dfs -cat /user/hadoop/predictions/part-00000 |more
2020-05-21T11:49:56.561506 Label: 7, Prediction: 7
2020-05-21T11:49:56.561535 Label: 2, Prediction: 2
2020-05-21T11:49:56.561541 Label: 1, Prediction: 1
2020-05-21T11:49:56.561545 Label: 0, Prediction: 0
2020-05-21T11:49:56.561550 Label: 4, Prediction: 4
2020-05-21T11:49:56.561555 Label: 1, Prediction: 1
```

©2013-2024 Tencent Cloud. All rights reserved.

```
2020-05-21T11:49:56.561559 Label: 4, Prediction: 4
2020-05-21T11:49:56.561564 Label: 9, Prediction: 9
2020-05-21T11:49:56.561568 Label: 5, Prediction: 6
2020-05-21T11:49:56.561573 Label: 9, Prediction: 9
2020-05-21T11:49:56.561578 Label: 0, Prediction: 0
2020-05-21T11:49:56.561582 Label: 6, Prediction: 6
2020-05-21T11:49:56.561587 Label: 9, Prediction: 9
2020-05-21T11:49:56.561603 Label: 0, Prediction: 0
2020-05-21T11:49:56.561608 Label: 1, Prediction: 1
2020-05-21T11:49:56.561612 Label: 5, Prediction: 5
```

Train the model based on InputMode.TENSORFLOW

```
sh mnist_train_with_tf_cpu.sh
```

View the model:

```
hdfs dfs -ls mnist_model
Found 25 items
-rw-r--r-- 1 hadoop supergroup 265 2020-05-21 14:58 mnist_model/checkpoint
-rw-r--r-- 1 hadoop supergroup 40 2020-05-21 14:53
mnist_model/events.out.tfevents.1590044017.10.0.0.144
-rw-r--r-- 1 hadoop supergroup 40 2020-05-21 14:57
mnist_model/events.out.tfevents.1590044221.10.0.0.144
-rw-r--r-- 1 hadoop supergroup 40 2020-05-21 14:57
mnist_model/events.out.tfevents.1590044227.10.0.0.144
-rw-r--r-- 1 hadoop supergroup 40 2020-05-21 14:57
mnist_model/events.out.tfevents.1590044232.10.0.0.144
-rw-r--r-- 1 hadoop supergroup 40 2020-05-21 14:57
mnist_model/events.out.tfevents.1590044238.10.0.0.144
-rw-r--r-- 1 hadoop supergroup 40 2020-05-21 14:58
mnist_model/events.out.tfevents.1590044303.10.0.0.114
-rw-r--r-- 1 hadoop supergroup 198078 2020-05-21 14:58 mnist_model/graph.pbtxt
drwxr-xr-x - hadoop supergroup 0 2020-05-21 14:58 mnist_model/inference
-rw-r--r-- 1 hadoop supergroup 814168 2020-05-21 14:57 mnist_model/model.ckpt-
238.data-00000-of-00001
-rw-r--r-- 1 hadoop supergroup 375 2020-05-21 14:57 mnist_model/model.ckpt-
238.index
-rw-r--r-- 1 hadoop supergroup 76255 2020-05-21 14:57 mnist_model/model.ckpt-
238.meta
-rw-r--r-- 1 hadoop supergroup 814168 2020-05-21 14:57 mnist_model/model.ckpt-
277.data-00000-of-00001
-rw-r--r-- 1 hadoop supergroup 375 2020-05-21 14:57 mnist_model/model.ckpt-
277.index
-rw-r--r-- 1 hadoop supergroup 76255 2020-05-21 14:57 mnist_model/model.ckpt-
277.meta
```

©2013-2024 Tencent Cloud. All rights reserved.

```
-rw-r--r-- 1 hadoop supergroup 814168 2020-05-21 14:57 mnist_model/model.ckpt-
315.data-00000-of-00001
-rw-r--r-- 1 hadoop supergroup 375 2020-05-21 14:57 mnist_model/model.ckpt-
315.index
-rw-r--r-- 1 hadoop supergroup 76255 2020-05-21 14:57 mnist_model/model.ckpt-
315.meta
-rw-r--r-- 1 hadoop supergroup 814168 2020-05-21 14:57 mnist_model/model.ckpt-
354.data-00000-of-00001
-rw-r--r-- 1 hadoop supergroup 375 2020-05-21 14:57 mnist_model/model.ckpt-
354.index
-rw-r--r-- 1 hadoop supergroup 76255 2020-05-21 14:57 mnist_model/model.ckpt-
354.meta
-rw-r--r-- 1 hadoop supergroup 814168 2020-05-21 14:58 mnist_model/model.ckpt-
393.data-00000-of-00001
-rw-r--r-- 1 hadoop supergroup 375 2020-05-21 14:58 mnist_model/model.ckpt-
393.index
-rw-r--r-- 1 hadoop supergroup 76255 2020-05-21 14:58 mnist_model/model.ckpt-
393.meta
drwxr-xr-x - hadoop supergroup 0 2020-05-21 14:53 mnist_model/train
```

Predict the model based on InputMode.TENSORFLOW

```
sh mnist_train_with_tf_cpu.sh
```

View the prediction result:

```
hdfs dfs -cat predictions/part-00000 |more
9 4
9 9
4 4
1 1
4 4
8 8
9 9
2 2
3 5
6 6
9 9
2 2
6 6
0 0
7 7
5 5
3 3
```

©2013-2024 Tencent Cloud. All rights reserved.

Kudu Development Guide
Kudu Overview

Last updated：2025-01-03 15:02:25

Apache Kudu is a distributed and horizontally scalable columnar storage system. It improves the storage layer of Hadoop and can quickly analyze rapidly changing data.

Basic Kudu Features

Fast processing of OLAP workloads.
Integration with MapReduce, Spark, and other Hadoop ecosystem components.
Tight integration with Apache Impala, making it a good and mutable alternative to using HDFS with Apache Parquet.
Flexible consistency model.
Strong performance for running sequential and random workloads simultaneously.
High data availability and storage reliability backed by the Raft protocol.
Structured data model.

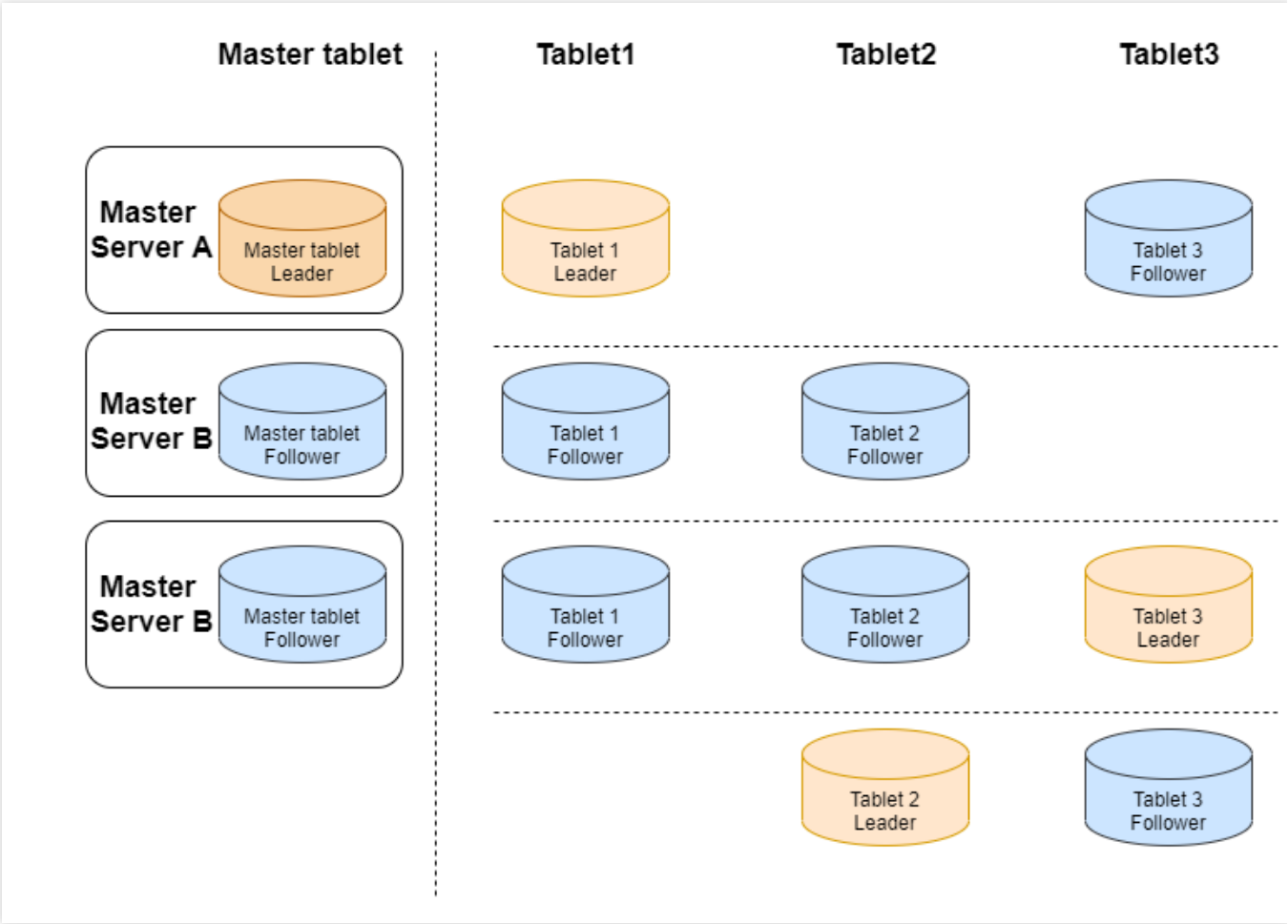
Kudu Use Cases

Complex scenarios involving both random access and batch data scanning.
Scenarios with high computational load.
Application of real-time predication models, which supports periodic model update based on all historical data.
Data update, which avoids repeated data migration.
Cross-region real-time data backup and query.

Basic Kudu Architecture

Kudu contains the following two types of components:
Master, which is mainly responsible for managing metadata information, listening on servers, and reassigning tablets in case of server failures.
Tablet server, which is mainly responsible for tablet storage and data CRUD.

©2013-2024 Tencent Cloud. All rights reserved.



Kudu Usage

EMR 2.4.0 supports the Kudu component. If you check the Kudu component when creating a Hadoop cluster, a Kudu cluster will be created. By default, it contains 3 Kudu masters, and high availability is enabled for it.

Note

All IPs used below are private IPs.

Integrate Impala with Kudu

```
[172.30.0.98:27001] > CREATE TABLE t2(id BIGINT,name STRING,PRIMARY
KEY(id))PARTITION BY HASH PARTITIONS 2 STORED AS KUDU TBLPROPERTIES (
'kudu.master_addresses' =
'172.30.0.240,172.30.1.167,172.30.0.96,172.30.0.94,172.30.0.214',
'kudu.num_tablet_replicas' = '1');
Query: create TABLE t2 (id BIGINT,name STRING,PRIMARY KEY(id)) PARTITION BY
HASH PARTITIONS 2 STORED AS KUDU TBLPROPERTIES (
'kudu.master_addresses' =
'172.30.0.240,172.30.1.167,172.30.0.96,172.30.0.94,172.30.0.214',
```

©2013-2024 Tencent Cloud. All rights reserved.

```
'kudu.num_tablet_replicas' = '1')
Fetched 0 row(s) in 0.12s
[hadoop@172 root]$ /usr/local/service/kudu/bin/kudu table list
172.30.0.240,172.30.1.167,172.30.0.96,172.30.0.94,172.30.0.214
impala::default.t2
```

Insert data

```
[172.30.0.98:27001] > insert into t2 values(1, 'test');
Query: insert into t2 values(1, 'test')
Query submitted at: 2020-08-10 20:07:21 (Coordinator: http://172.30.0.98:27004)
Query progress can be monitored at: http://172.30.0.98:27004/query_plan?
query_id=b44fe203ce01254d:b055e98200000000
Modified 1 row(s), 0 row error(s) in 5.63s
```

Query data based on Impala

```
[172.30.0.98:27001] > select * from t2;
Query: select * from t2
Query submitted at: 2020-08-10 20:09:47 (Coordinator: http://172.30.0.98:27004)
Query progress can be monitored at: http://172.30.0.98:27004/query_plan?
query_id=ec4c9706368f135d:f20ccb6e00000000
+----+------+
| id | name |
+----+------+
| 1  | test |
+----+------+
Fetched 1 row(s) in 0.20s
```

Other commands

i. Perform health check for the cluster

```
[hadoop@172 root]$ /usr/local/service/kudu/bin/kudu cluster ksck
172.30.0.240,172.30.1.167,172.30.0.96,172.30.0.94,172.30.0.214
```

ii. Create a table

```
 [hadoop@172 root]$ /usr/local/**service**/**kudu**/bin/kudu table create
'172.30.0.240,172.30.1.167,172.30.0.96,172.30.0.94,172.30.0.214'
'{"table_name":"test","schema":{"columns":
[{"column_name":"id","column_type":"INT32","default_value":"1"},
{"column_name":"key","column_type":"INT64","is_nullable":false,"comment":"range
key"},
{"column_name":"name","column_type":"STRING","is_nullable":false,"comment":"use
r name"}],"key_column_names":["id","key"]},"partition":{"hash_partitions":
[{"columns":["id"],"num_buckets":2,"seed":100}],"range_partition":{"columns":
["key"],"range_bounds":[{"upper_bound":
```

©2013-2024 Tencent Cloud. All rights reserved.

```
{"bound_type":"inclusive","bound_values":["2"]}},{"lower_bound":
{"bound_type":"exclusive","bound_values":["2"]},"upper_bound":
{"bound_type":"inclusive","bound_values":["3"]}}]}},"extra_configs":{"configs":
{"kudu.table.history_max_age_sec":"3600"}},"num_replicas":1}'
```

iii. Query the created `test` table

```
[hadoop@172 root]$ /usr/local/service/kudu/bin/kudu table list
172.30.0.240,172.30.1.167,172.30.0.96,172.30.0.94,172.30.0.214
test
```

iv. View table structure

```
[hadoop@172 root]$ /usr/local/service/kudu/bin/kudu table describe
172.30.0.240,172.30.1.167,172.30.0.96,172.30.0.94,172.30.0.214 test
TABLE test (
    id INT32 NOT NULL,
    key INT64 NOT NULL,
    name STRING NOT NULL,
    PRIMARY KEY (id, key)
)
HASH (id) PARTITIONS 2 SEED 100,
RANGE (key) (
    PARTITION VALUES < 3,
    PARTITION 3 <= VALUES < 4
)
REPLICAS 1
```

©2013-2024 Tencent Cloud. All rights reserved.

Data Migration Guide for Kudu Node Scale-In

Last updated：2025-01-03 15:02:25

This document describes how to migrate data when removing core nodes tservers from a Kudu cluster.

Note

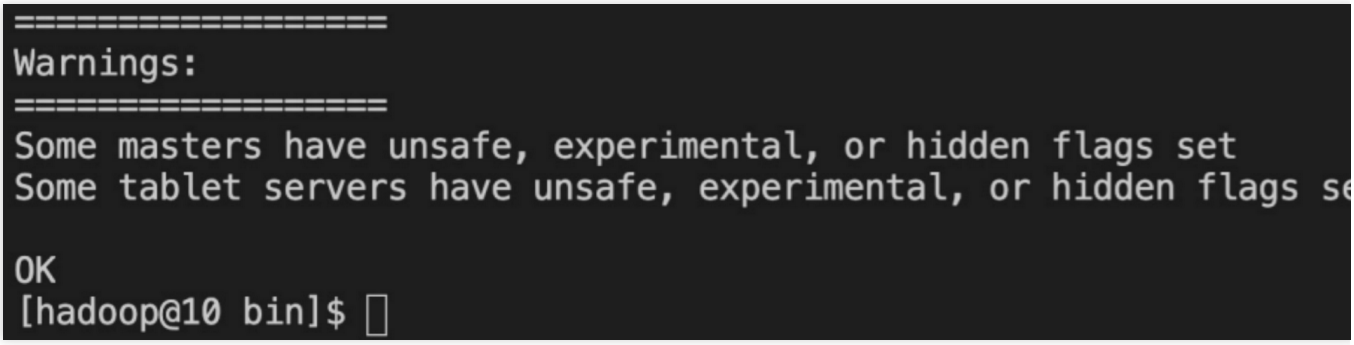
Core nodes can be removed from a Kudu cluster. This feature is not available by default. If you need to use it, submit a ticket for application.

Before disconnecting tservers, you can use the rebalancing tool for data migration. Note that you can disconnect only one tserver at a time. To disconnect multiple ones, repeat the following steps.

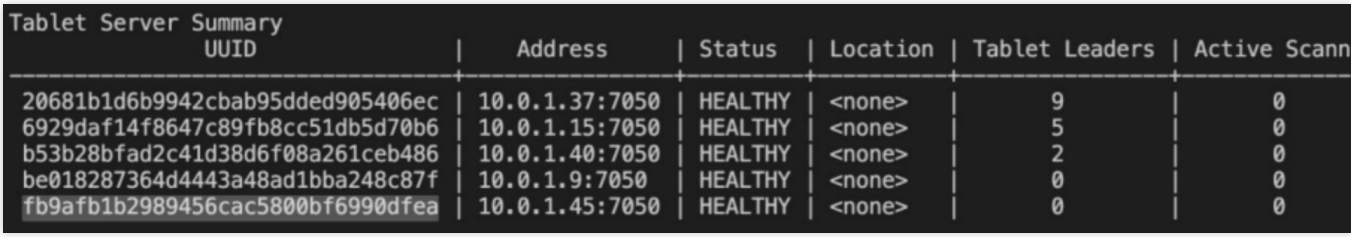
Kudu migration based on rebalancing tool

1. Make sure that the cluster status is **OK**.

```
/usr/local/service/kudu/bin/kudu cluster ksck
10.0.1.29:7051,10.0.1.16:7051,10.0.1.36:7051
```



2. Run the `ksck` command in step 1 to get the `uid` of the disconnected nodes.



Take the `fb9afb1b2989456cac5800bf6990dfea` node as an example.

©2013-2024 Tencent Cloud. All rights reserved.

3. Switch the `fb9afb1b2989456cac5800bf6990dfea` node to the maintenance mode.

```
/usr/local/service/kudu/bin/kudu tserver state enter_maintenance
10.0.1.29:7051,10.0.1.16:7051,10.0.1.36:7051 fb9afb1b2989456cac5800bf6990dfea
```

4. Run the following `rebalance` command.

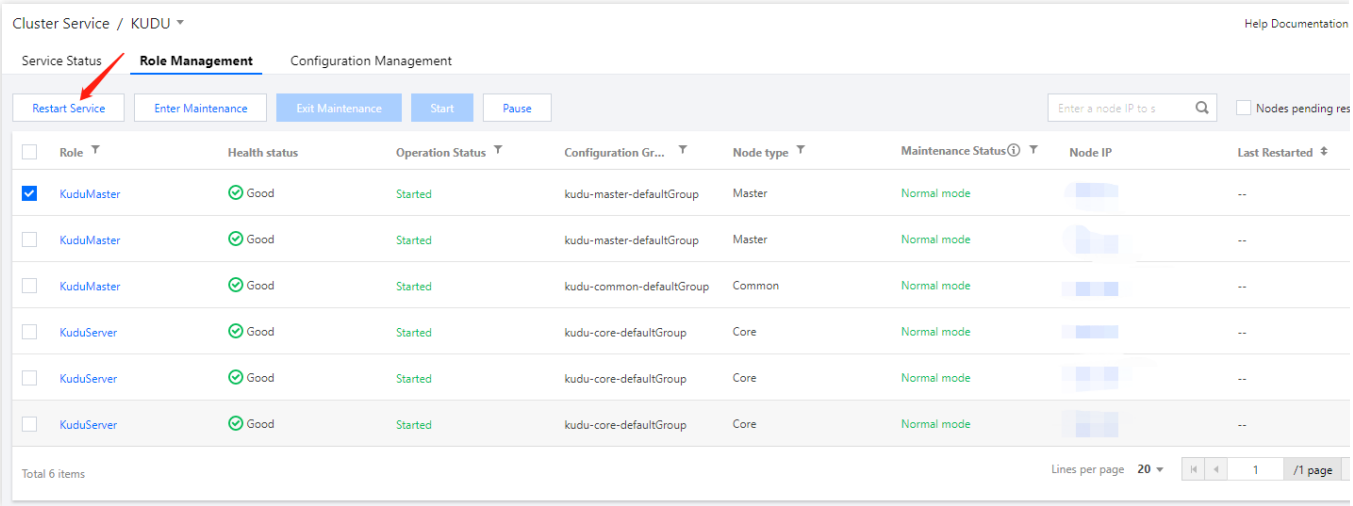
```
/usr/local/service/kudu/bin/kudu cluster rebalance
10.0.1.29:7051,10.0.1.16:7051,10.0.1.36:7051 --ignored_tservers
fb9afb1b2989456cac5800bf6990dfea --move_replicas_from_ignored_tservers
```

After the command is executed, run the `ksck` command again to check whether the status is **OK** before proceeding.

5. Suspend the tserver process at `10.0.1.45` on the `fb9afb1b2989456cac5800bf6990dfea` node. Note that at this point, if you run the `ksck` command, you will see that the cluster is unhealthy, and you need to restart the tmasters.

```
Tablet Server Summary
                UUID                |     Address     |   Status    | Location | Tablet Leaders | Active Scanners
------------------------------------+-----------------+-------------+----------+----------------+-----------------
 20681b1d6b9942cbab95dded905406ec   | 10.0.1.37:7050  | HEALTHY     | <none>   |              9 |               0
 6929daf14f8647c89fb8cc51db5d70b6   | 10.0.1.15:7050  | HEALTHY     | <none>   |              5 |               0
 b53b28bfad2c41d38d6f08a261ceb486   | 10.0.1.40:7050  | HEALTHY     | <none>   |              2 |               0
 be018287364d4443a48ad1bba248c87f   | 10.0.1.9:7050   | HEALTHY     | <none>   |              0 |               0
 fb9afb1b2989456cac5800bf6990dfea   | 10.0.1.45:7050  | UNAVAILABLE | <none>   | n/a            | n/a
```

6. Restart the masters in the EMR console one by one (rolling restart in the console is not recommended). Then, run the `ksck` command to check whether the cluster is healthy.



©2013-2024 Tencent Cloud. All rights reserved.

Ranger Development Guide
Ranger Overview

Last updated：2025-01-03 15:02:25

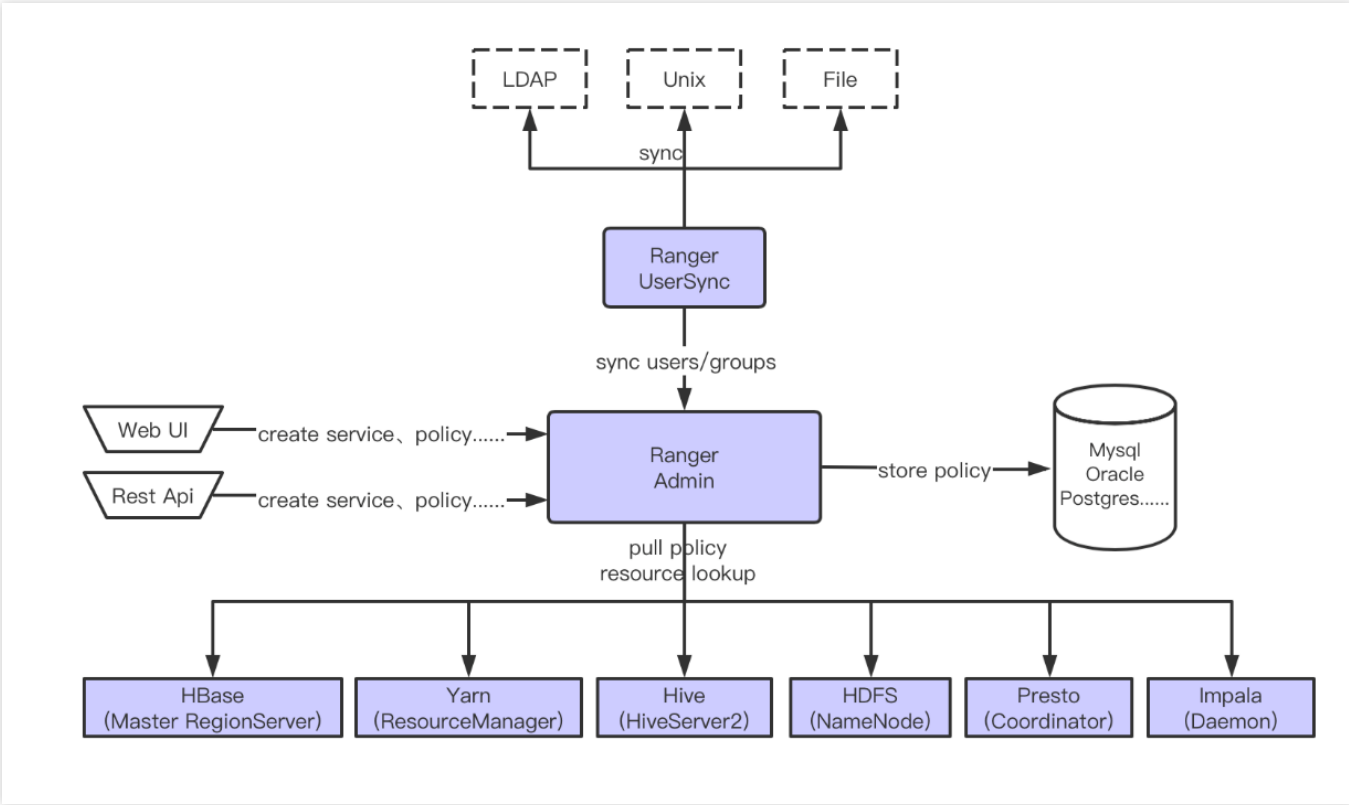
Ranger Overview

Ranger is a framework for centralized security management of Hadoop components in the field of big data. Users can use Ranger to securely access data in a cluster. It is mainly designed to monitor Hadoop components, and control service launch and resource access. The major goals of Ranger include:
Centralized security management of big data components in the web UI or using RESTful APIs provided by Ranger.
Authorization to perform specific operations with big data components based on roles and attributes.
Centralized auditing of user access and administrative operations (security-related) with big data components.

Ranger Architecture

Ranger is mainly composed of Ranger Admin, Ranger UserSync, and Ranger Plugin. Both Ranger Admin and Ranger UserSync are a separate JVM process, while Ranger Plugin needs to be installed on different nodes depending on different components.

©2013-2024 Tencent Cloud. All rights reserved.



Ranger Admin is used to manage configured policies, created services, and audit logs and reports. It also persistently saves the policies and services to the database for regular queries from Ranger Plugin.
Ranger UserSync is used to synchronize information from LDAP, File, and Unix to Ranger Admin, for example, user and group information from users' Unix or LDAP directory access systems. To synchronize user and group information from Unix, you need to enable the `unixAuthenticationService` process and persistently store the synchronized information.
Ranger Plugin will be deployed on service nodes as needed and periodically synchronize policy information from Ranger Admin.
The following table lists the components that can be integrated with Ranger.

Service	Installation Nodes	EMR Versions
HDFS	NameNode	EMR v2.0.1 and above
HBase	Master, RegionServer	EMR v2.0.1 and above
Hive	HiveServer2	EMR v2.0.1 and above
YARN	ResourceManager	EMR v2.0.1 and above
Presto	All coordinators	EMR v2.0.1 and above
Impala	All daemons	EMR v2.2.0 and above

©2013-2024 Tencent Cloud. All rights reserved.

Kudu	All masters	EMR v3.2.0

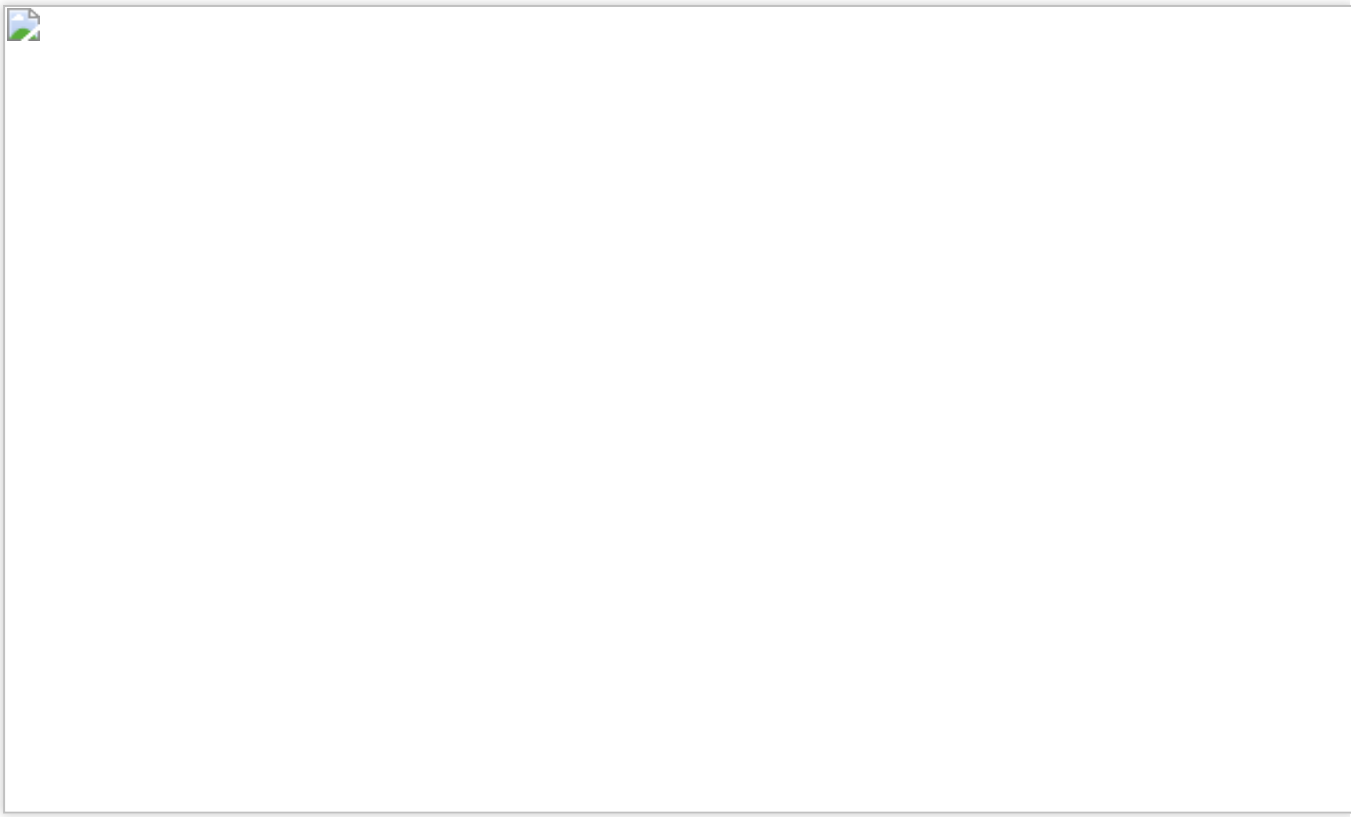
©2013-2024 Tencent Cloud. All rights reserved.

Ranger User Guide
Integrating HDFS with Ranger

Last updated：2025-01-03 15:02:25

Preparations

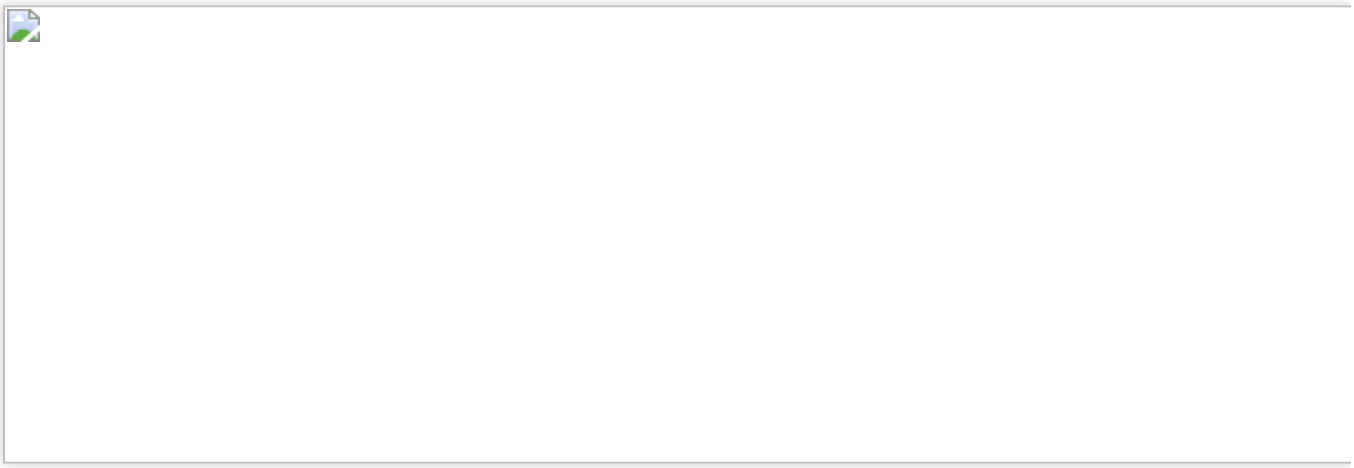
Ranger is available only when it is selected in **Optional Components** when you purchase a cluster. If you add the Ranger component after purchasing the cluster, the Web UI may be inaccessible. By default, when Ranger is installed, Ranger Admin and Ranger UserSync are deployed on the master node, and Ranger Plugin is deployed on the main daemon node of the embedded component.
When creating a cluster of the Hadoop type, you can select Ranger in **Optional Components**. The Ranger version varies depending on the EMR version you choose.
Note:
When the cluster type is Hadoop and the Ranger optional component is selected, EMR-Ranger will create services for HDFS and YARN by default and set default policies.



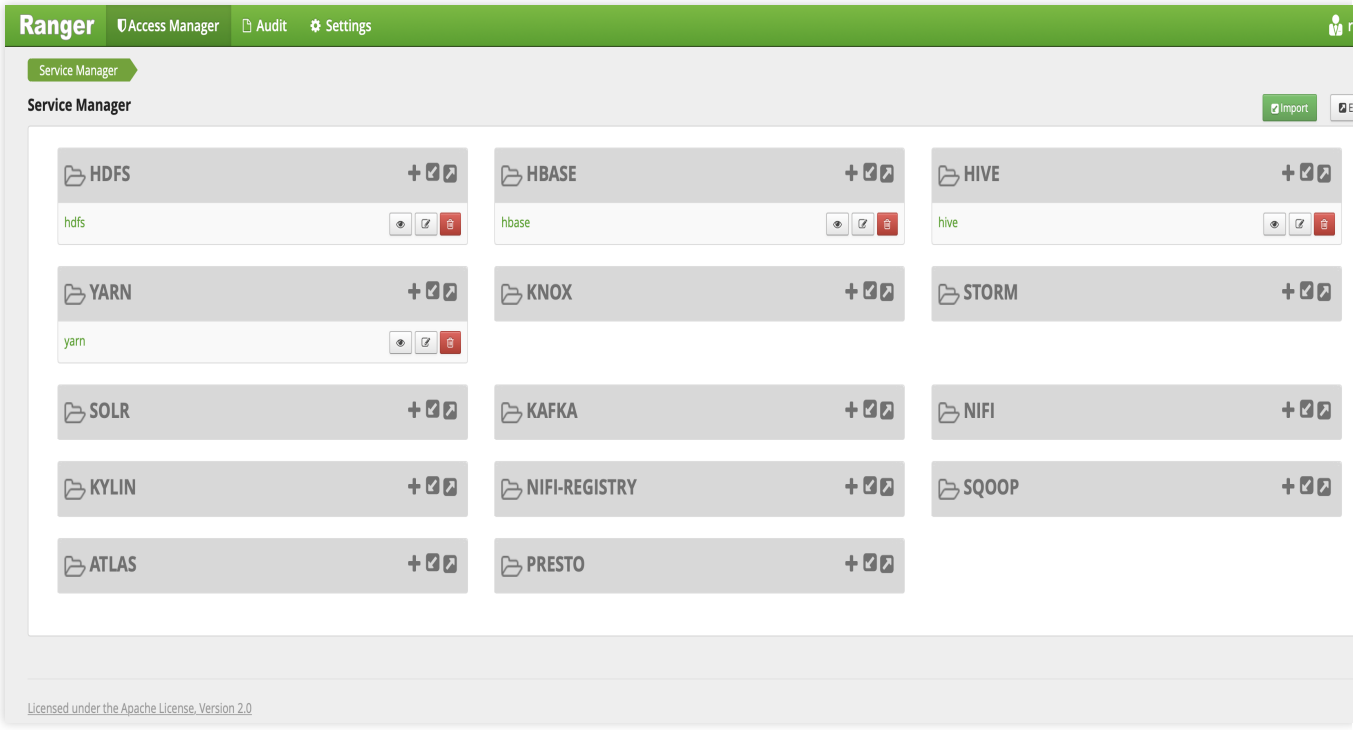
©2013-2024 Tencent Cloud. All rights reserved.

Ranger Web UI

Before accessing the Ranger Web UI, make sure that the current cluster is configured with a public IP and click the Ranger Web UI URL on the **Cluster Service** page.



After you are redirected, enter the username and password that you set when you purchased the cluster.



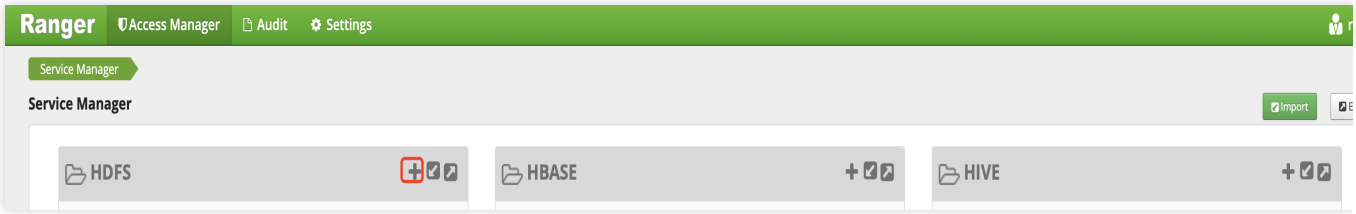
Integrating HDFS with Ranger

Note:

©2013-2024 Tencent Cloud. All rights reserved.

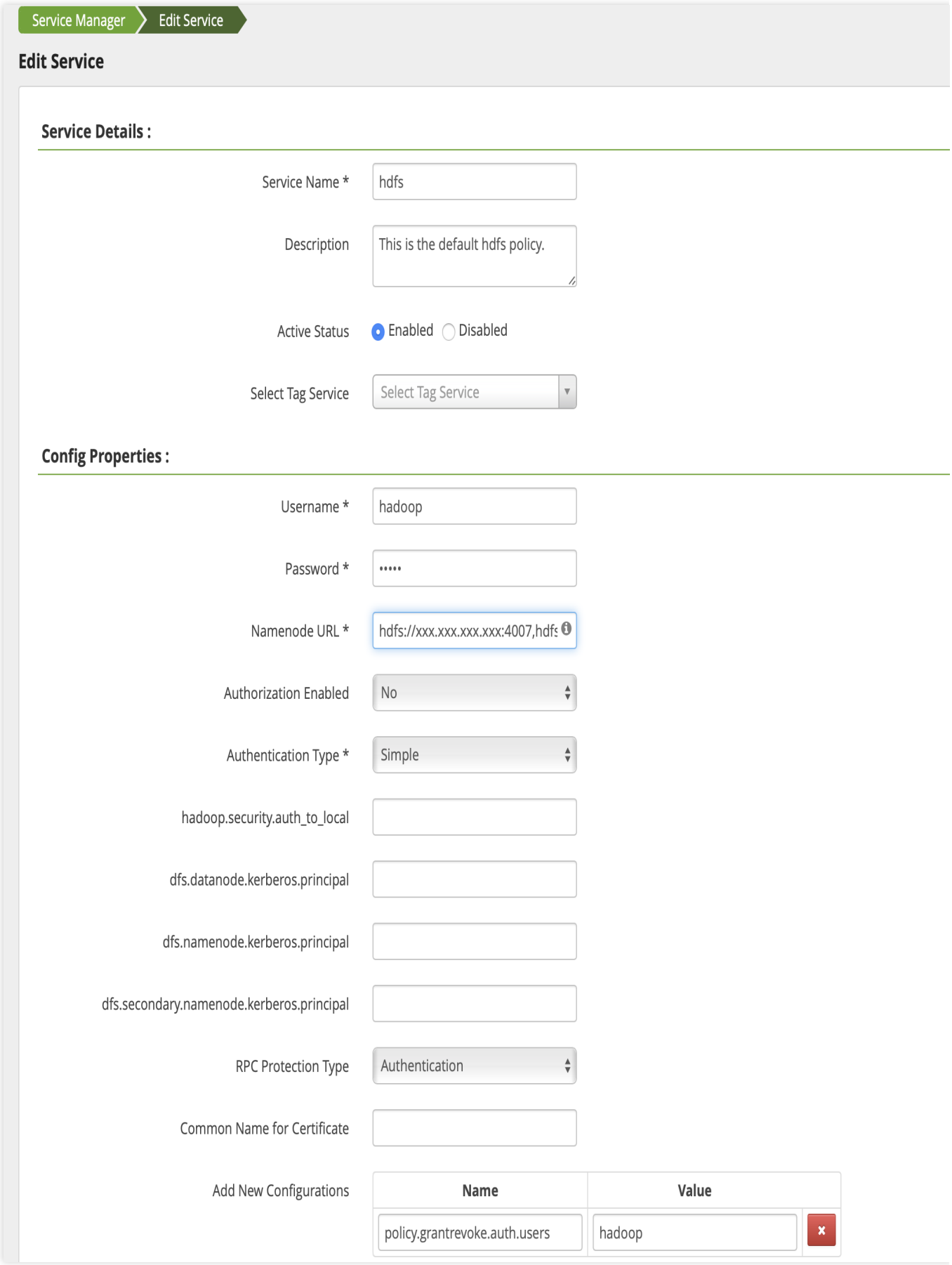
Make sure that HDFS related services are running normally and Ranger has been installed in the current cluster.

1. Add an EMR Ranger HDFS service on the EMR Ranger Web UI.



2. Configure EMR Ranger HDFS service parameters.

©2013-2024 Tencent Cloud. All rights reserved.

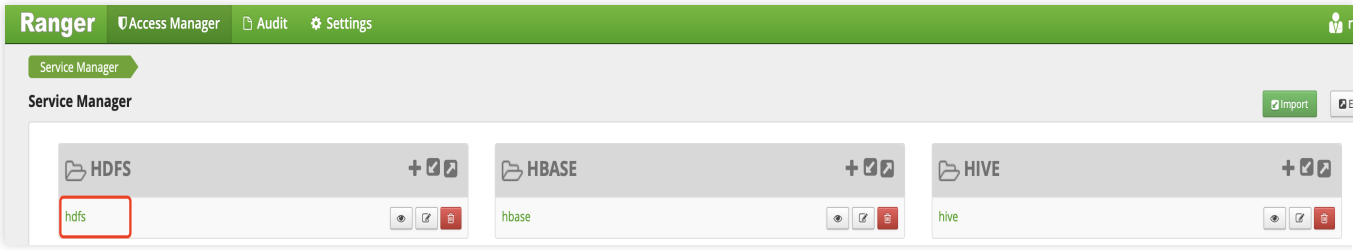


©2013-2024 Tencent Cloud. All rights reserved.

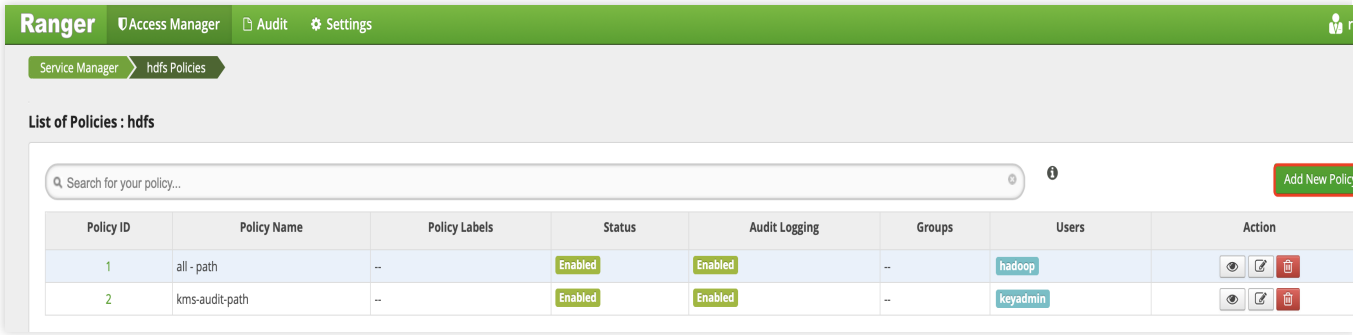
Parameter	Required	Description
Service Name	Yes	Service name, which is displayed on the main HDFS component on the Ranger Web UI
Description	No	Service description
Active Status	Default	Service status, which is **Enabled** by default
Username	Yes	Username of the resource
Password	Yes	User password
NameNode URL	Yes	HDFS URL
Authorization Enabled	Default	Select **No** for standard clusters and **Yes** for high-security clusters.
Authorization Type	Yes	**Simple**: standard cluster; **Kerberos**: high-security cluster

3. Configure EMR Ranger HDFS resource permissions.

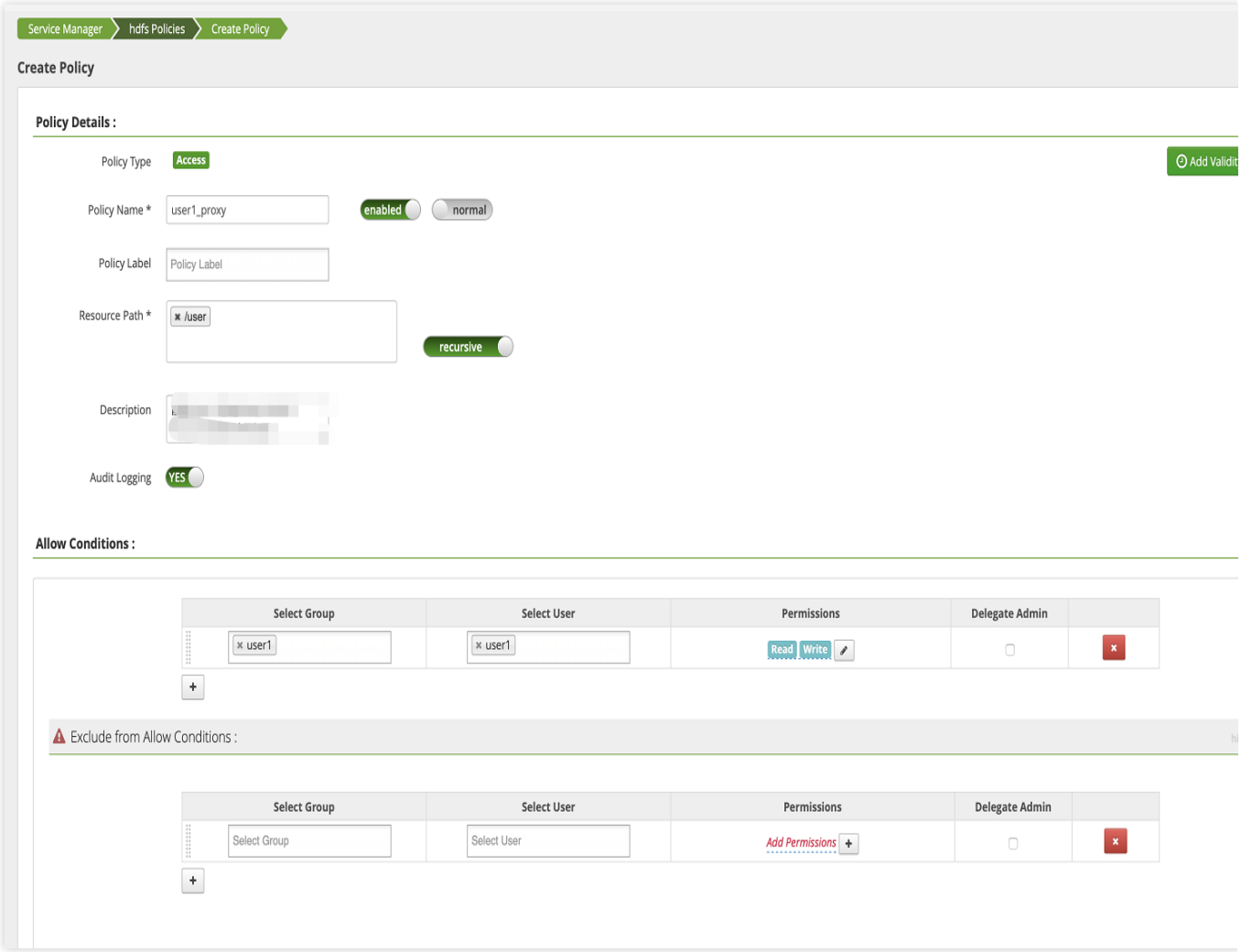
Click the configured EMR Ranger HDFS service.



Configure a policy.



©2013-2024 Tencent Cloud. All rights reserved.



4. After the policy is added, it will take effect in about 30 seconds, then you can use **user1** to read and write the /**user** of the HDFS file system.

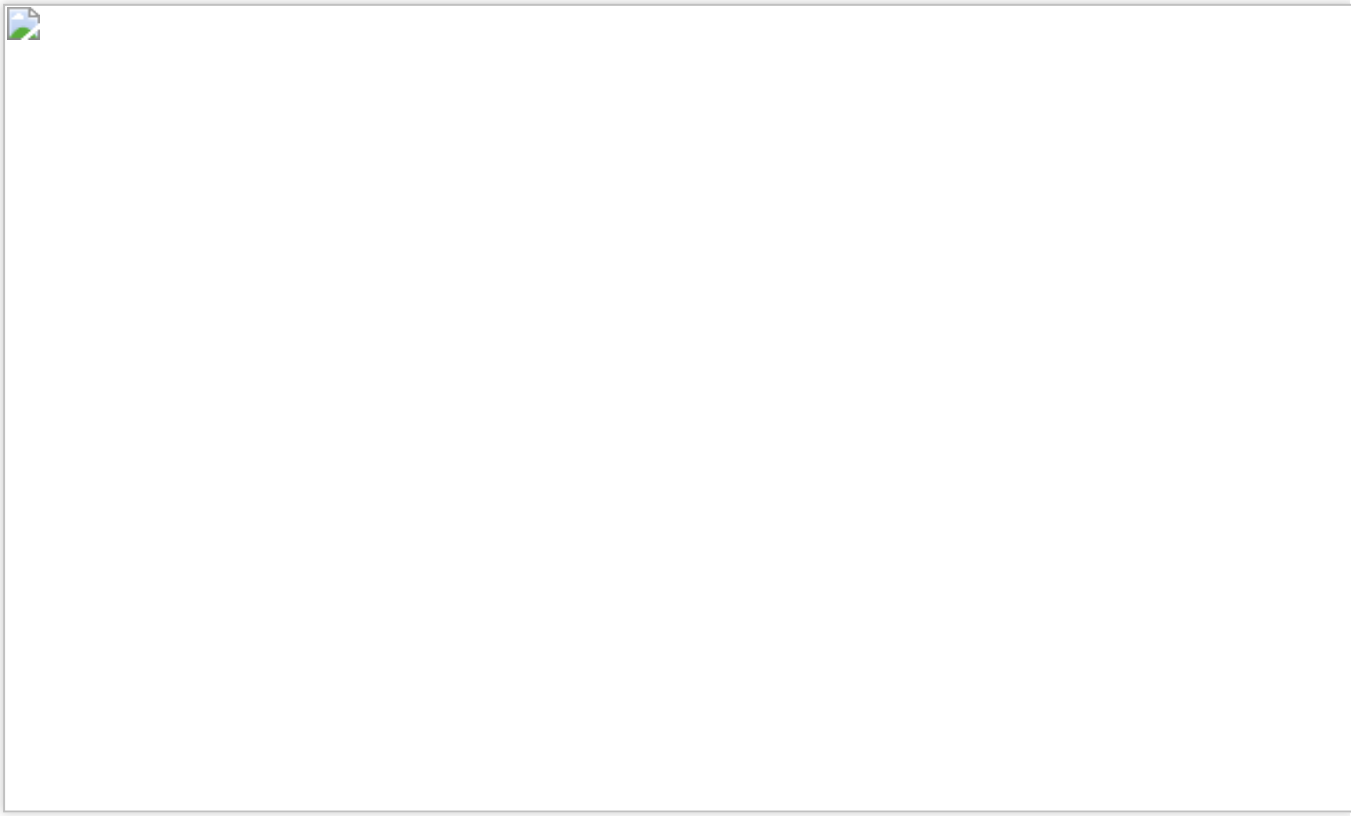
©2013-2024 Tencent Cloud. All rights reserved.

Integrating YARN with Ranger

Last updated：2025-01-03 15:02:25

Preparations

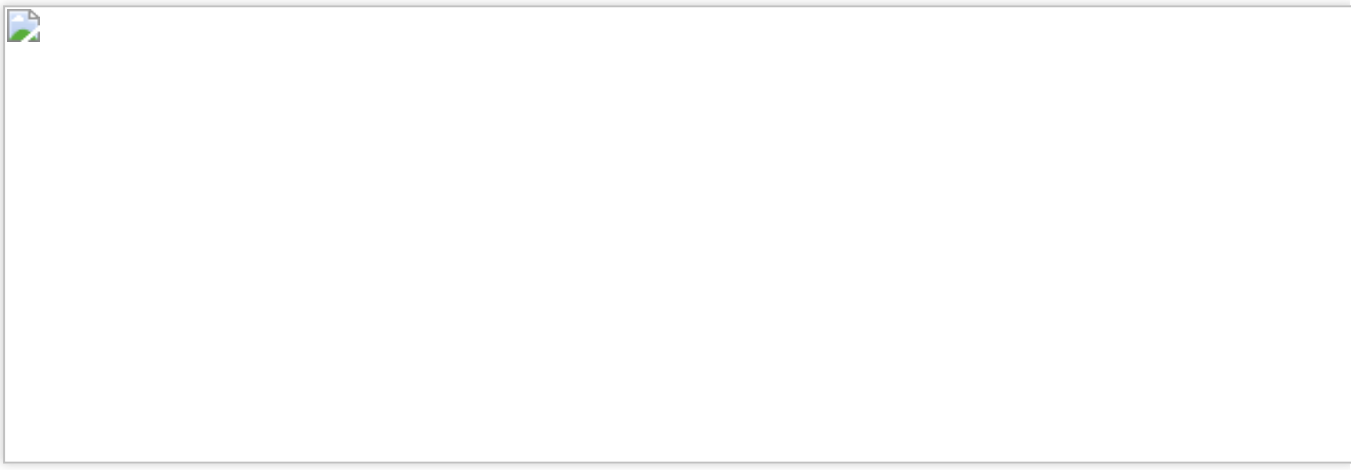
Ranger is available only when it is selected in **Optional Components** when you purchase a cluster. If you add the Ranger component after purchasing the cluster, the Web UI may be inaccessible. By default, when Ranger is installed, Ranger Admin and Ranger UserSync are deployed on the master node, and Ranger Plugin is deployed on the main daemon node of the embedded component.
When creating a cluster of the Hadoop type, you can select Ranger in **Optional Components**. The Ranger version varies depending on the EMR version you choose.
Note:
When the cluster type is Hadoop and the Ranger optional component is selected, EMR-Ranger will create services for HDFS and YARN by default and set default policies.



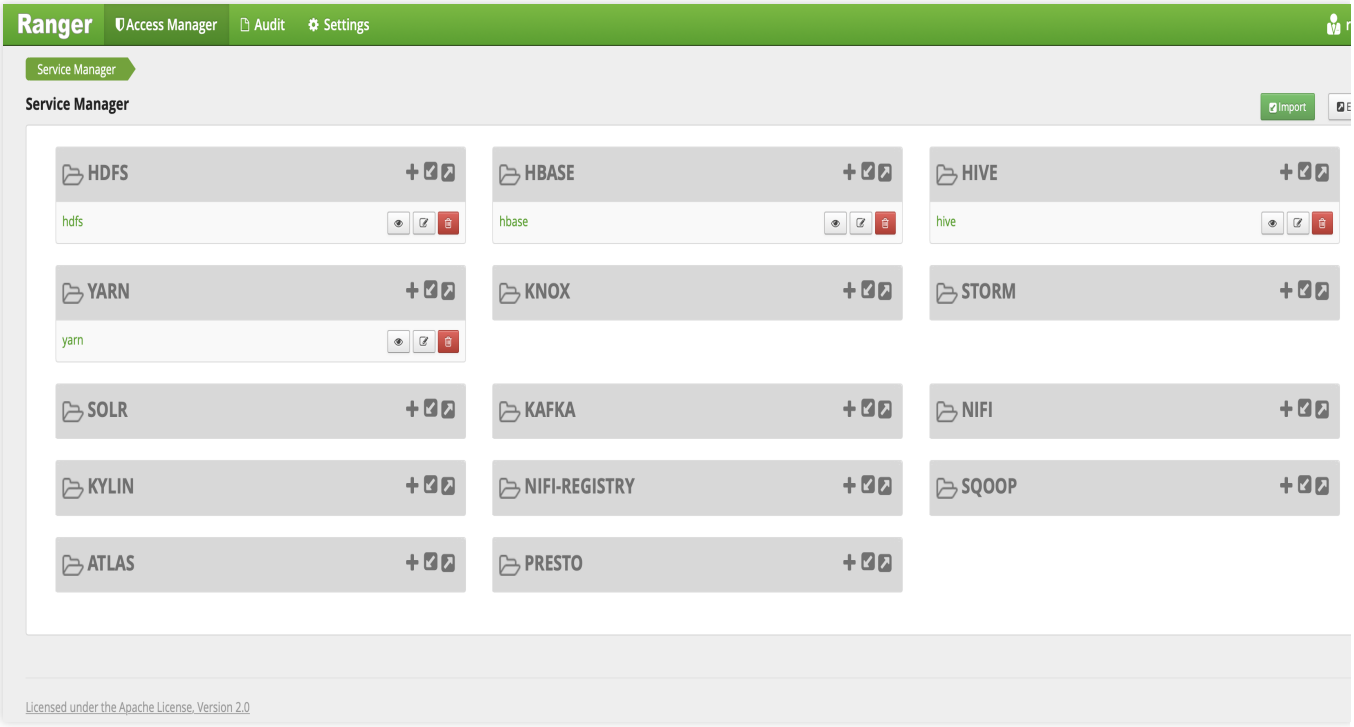
Ranger Web UI

©2013-2024 Tencent Cloud. All rights reserved.

Before accessing the Ranger Web UI, make sure that the current cluster is configured with a public IP and click the Ranger Web UI URL on the **Cluster Service** page.



After you are redirected, enter the username and password that you set when you purchased the cluster.



Integrating YARN with Ranger

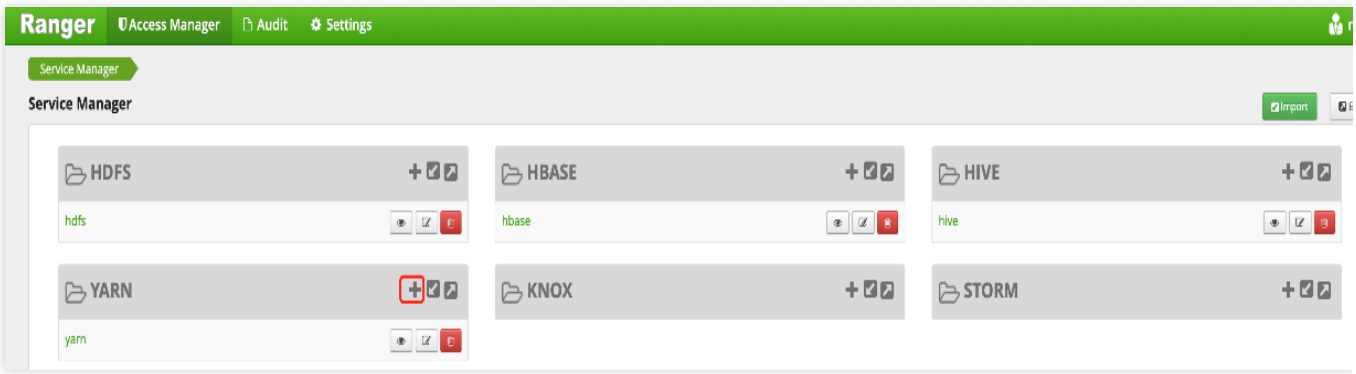
Note:

Make sure that YARN related services are running normally and Ranger has been installed in the current cluster.

©2013-2024 Tencent Cloud. All rights reserved.

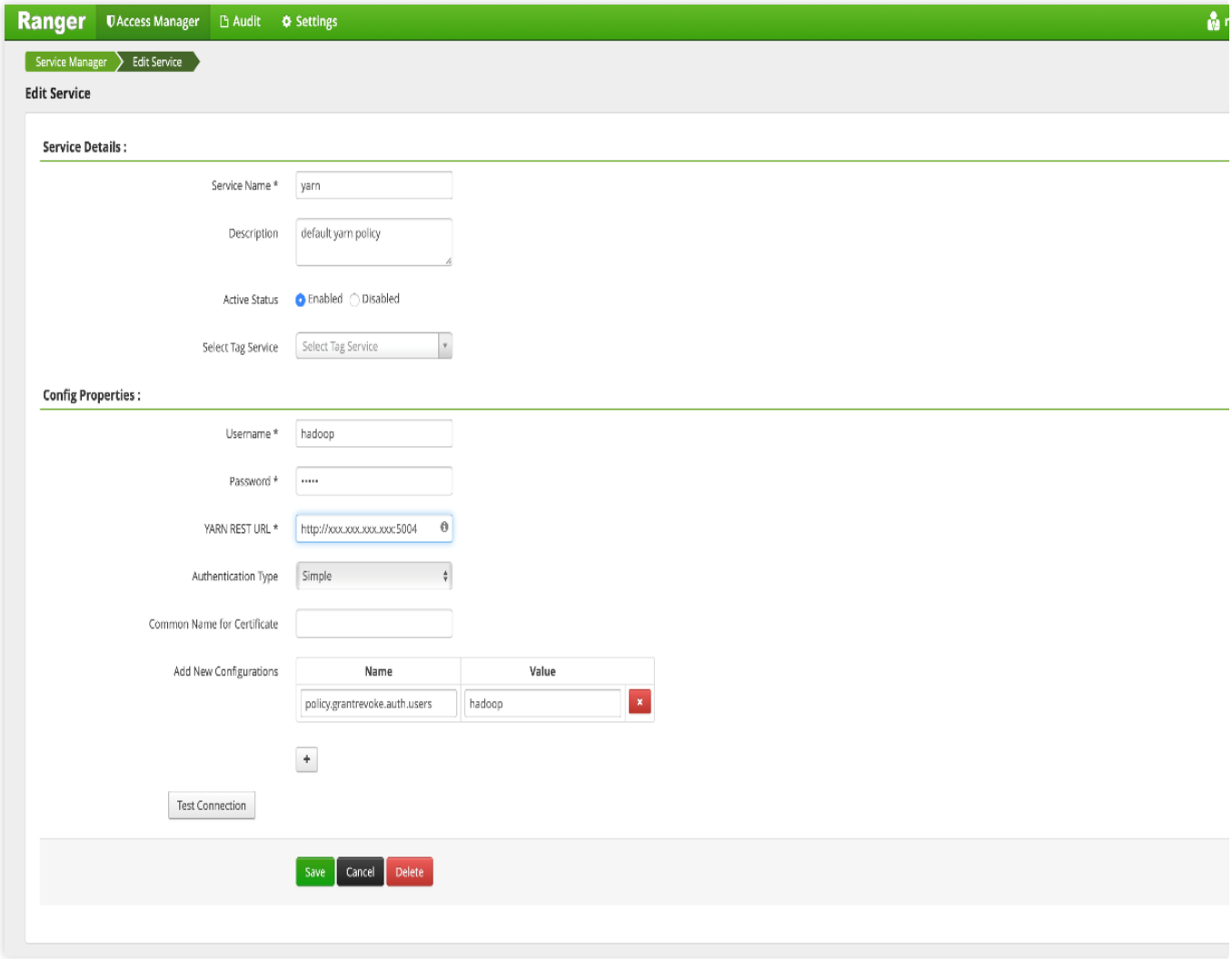
Currently, EMR Ranger YARN only supports ACLs for Capacity Scheduler queues, not for Fair Scheduler queues. Ranger YARN's queue ACLs take effect together with YARN's built-in Capacity Scheduler configuration, but with lower priority. Ranger YARN permissions will be verified only when YARN's built-in Capacity Scheduler configuration denies verification. **You are advised to set ACLs via Ranger, instead of in the configuration files.**

1. Add an EMR Ranger YARN service on the EMR Ranger Web UI.



2. Configure EMR Ranger YARN service parameters.

©2013-2024 Tencent Cloud. All rights reserved.

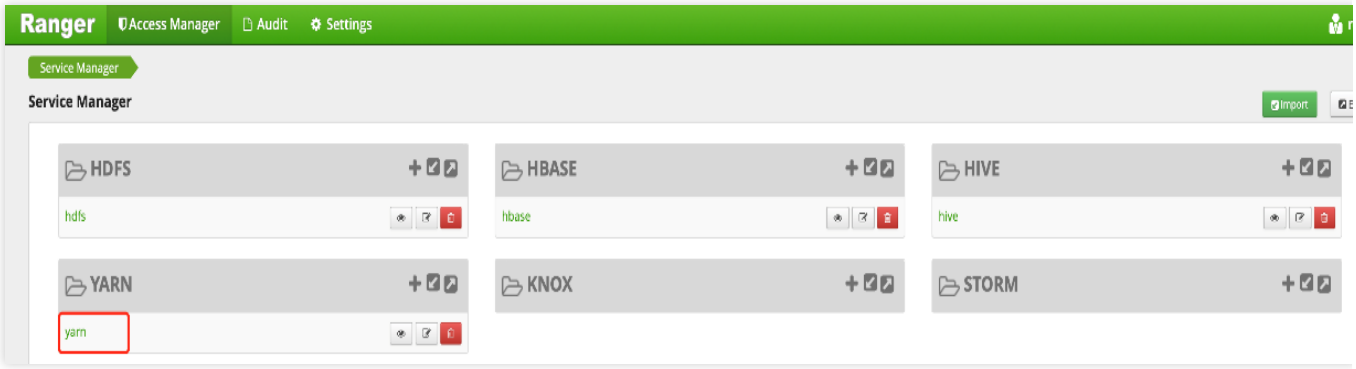


Parameter	Required	Description
Service Name	Yes	Service name, which is displayed on the main YARN component on the Ranger Web UI
description	No	Service description
Active Status	Default	Service status, which is **Enabled** by default
Username	Yes	Username of the resource
Password	Yes	User password
NameNode URL	Yes	YARN URL
Authorization Enabled	Default	Select **No** for standard clusters and **Yes** for high-security clusters.
Authorization Type	Yes	**Simple**: standard cluster; **Kerberos**: high-security cluster

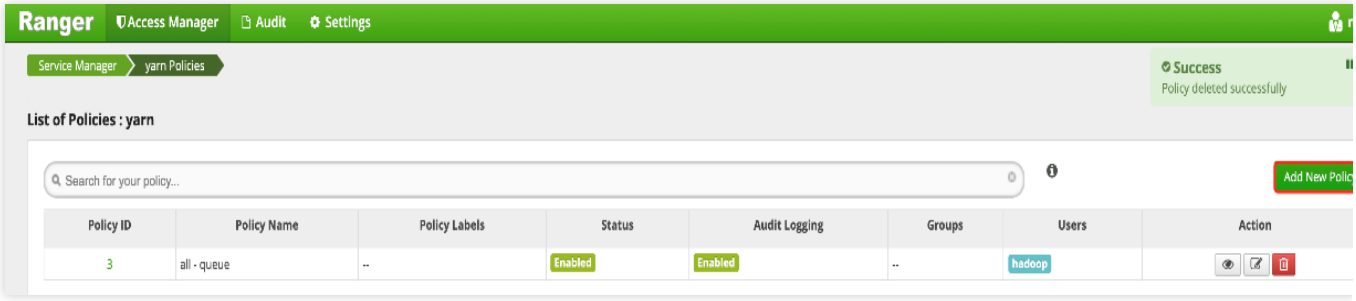
©2013-2024 Tencent Cloud. All rights reserved.

3. Configure EMR Ranger YARN resource permissions.

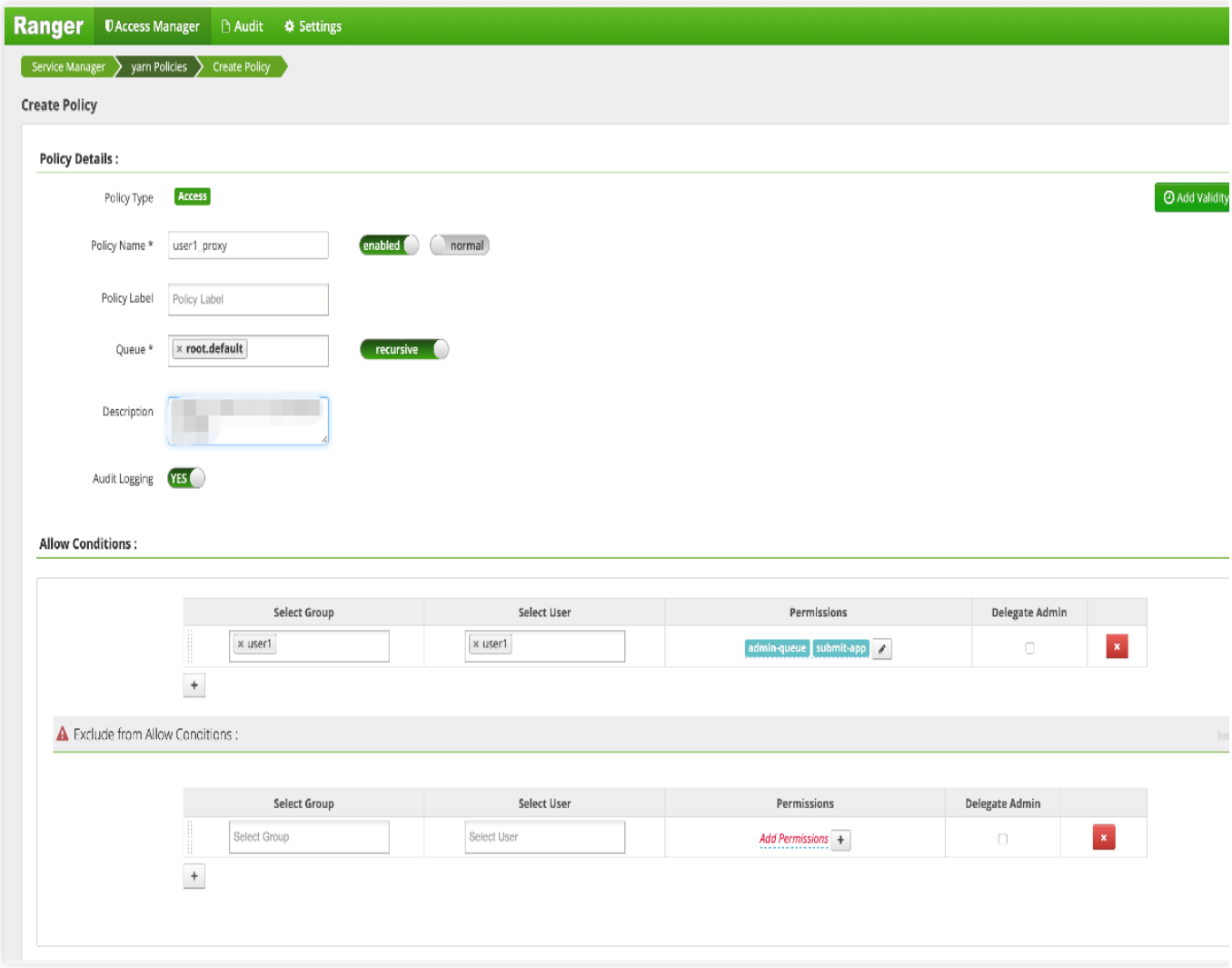
Click the configured EMR Ranger YARN service.



Configure a policy.



©2013-2024 Tencent Cloud. All rights reserved.



4. After the policy is added, it will take effect in about 30 seconds, then you can use **user1** to submit, kill, or query jobs in the `root.default` queue of YARN.

Note:

When configuring Ranger YARN services and policies, make sure that there are no YARN jobs during this period; otherwise, issues about job submission permissions may occur.

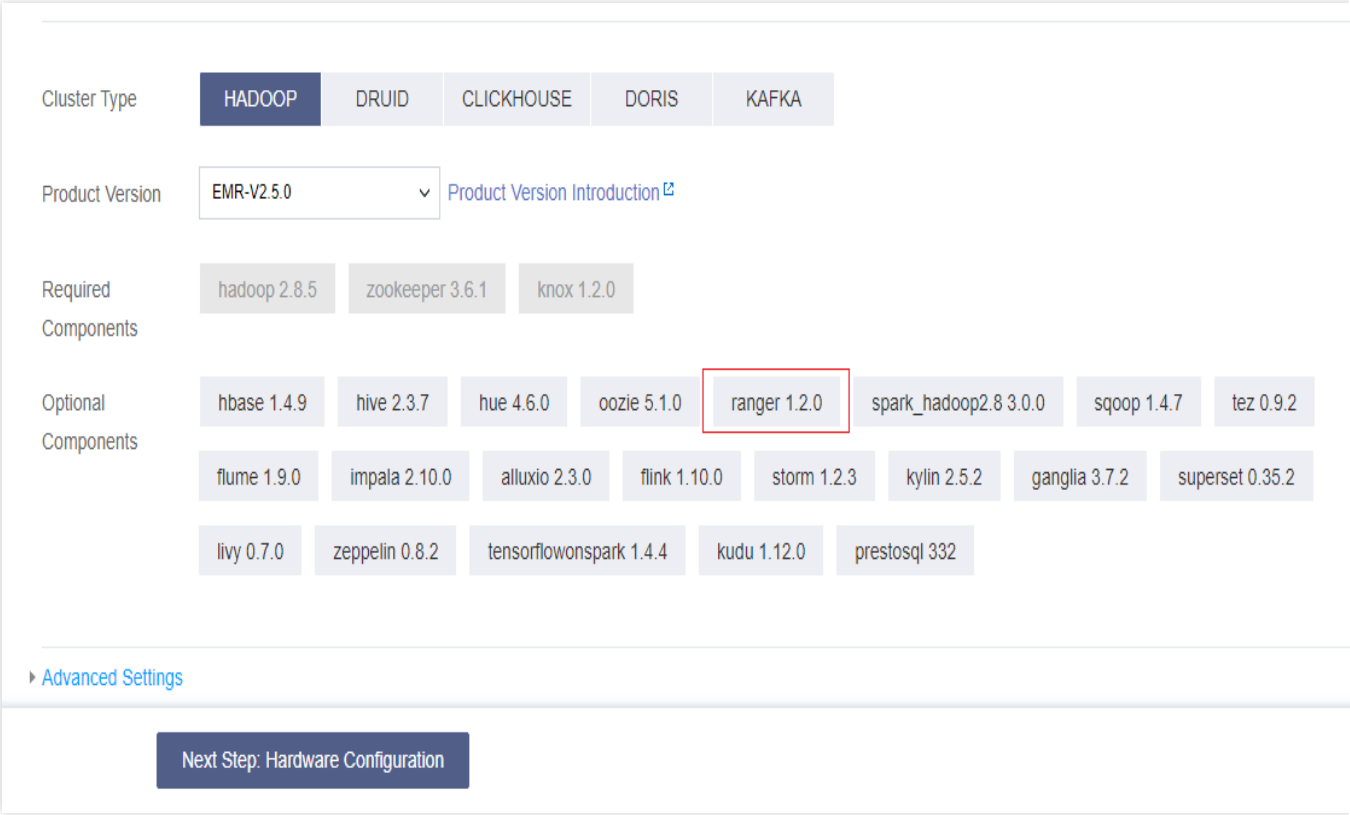
©2013-2024 Tencent Cloud. All rights reserved.

Integrating HBase with Ranger

Last updated：2025-01-03 15:02:25

Preparations

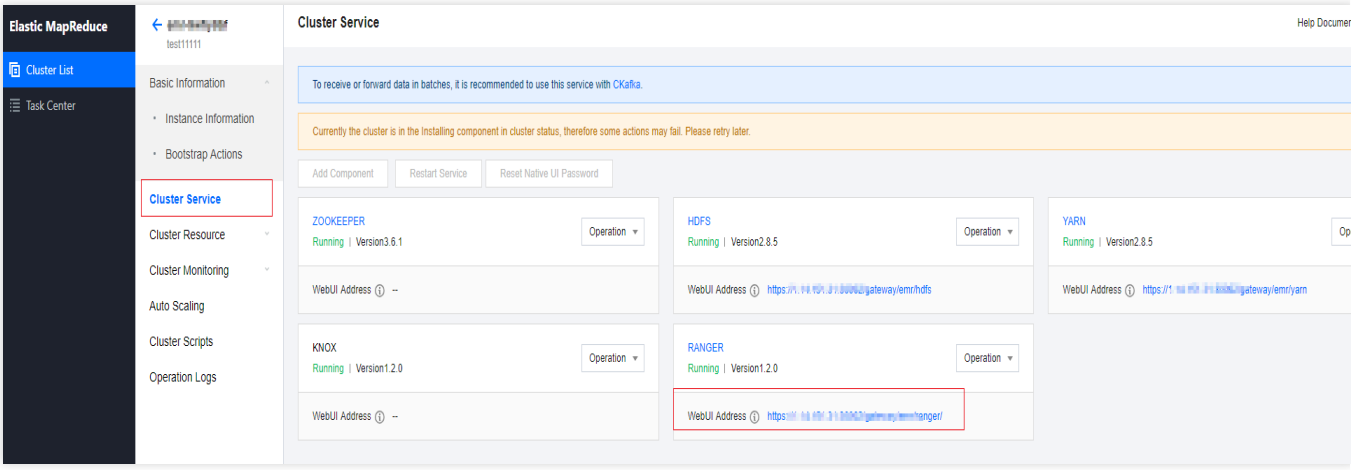
Ranger is available only when it is selected in **Optional Components** when you purchase a cluster. If you add the Ranger component after purchasing the cluster, the Web UI may be inaccessible. By default, when Ranger is installed, Ranger Admin and Ranger UserSync are deployed on the master node, and Ranger Plugin is deployed on the main daemon node of the embedded component.
When creating a cluster of the Hadoop type, you can select Ranger in **Optional Components**. The Ranger version varies depending on the EMR version you choose.
Note:
When the cluster type is Hadoop and the Ranger optional component is selected, EMR-Ranger will create services for HDFS and YARN by default and set default policies.



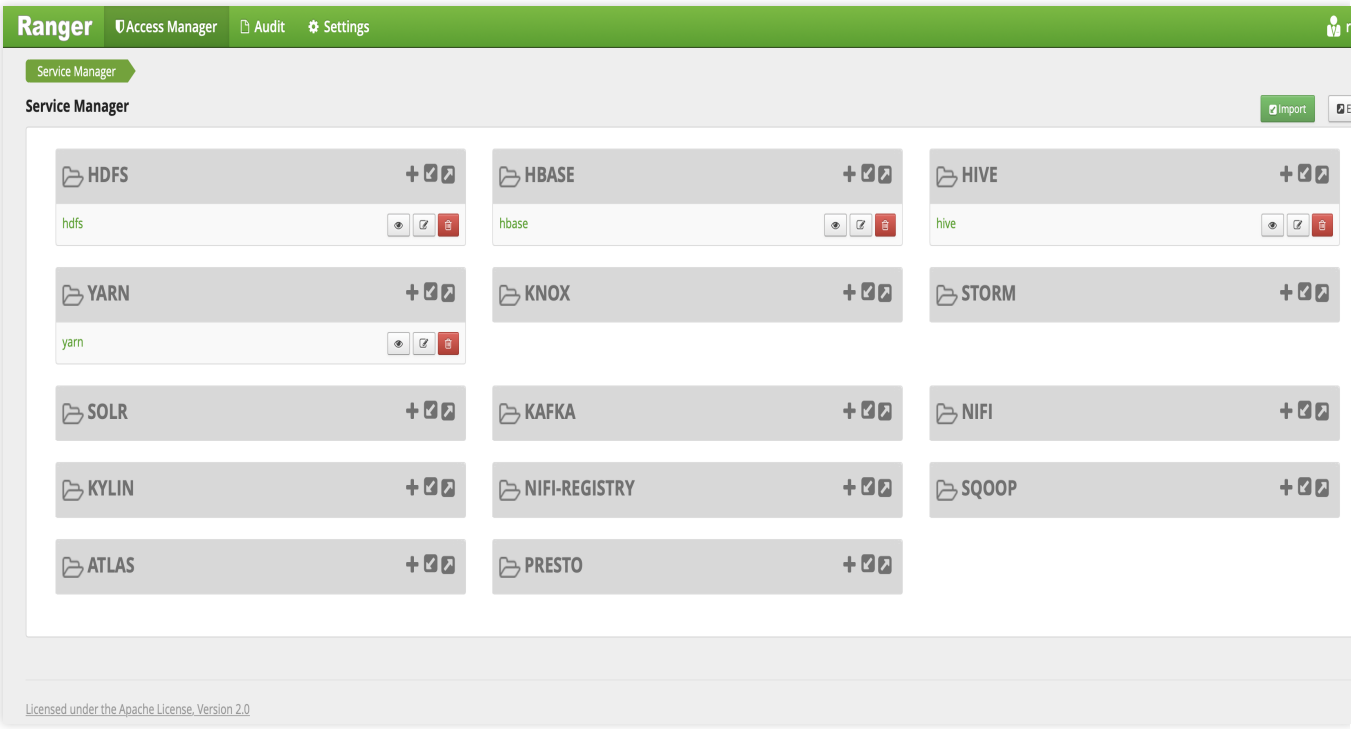
Ranger Web UI

©2013-2024 Tencent Cloud. All rights reserved.

Before accessing the Ranger Web UI, make sure that the current cluster is configured with a public IP and click the Ranger Web UI URL on the **Cluster Service** page.



After you are redirected, enter the username and password that you set when you purchased the cluster.



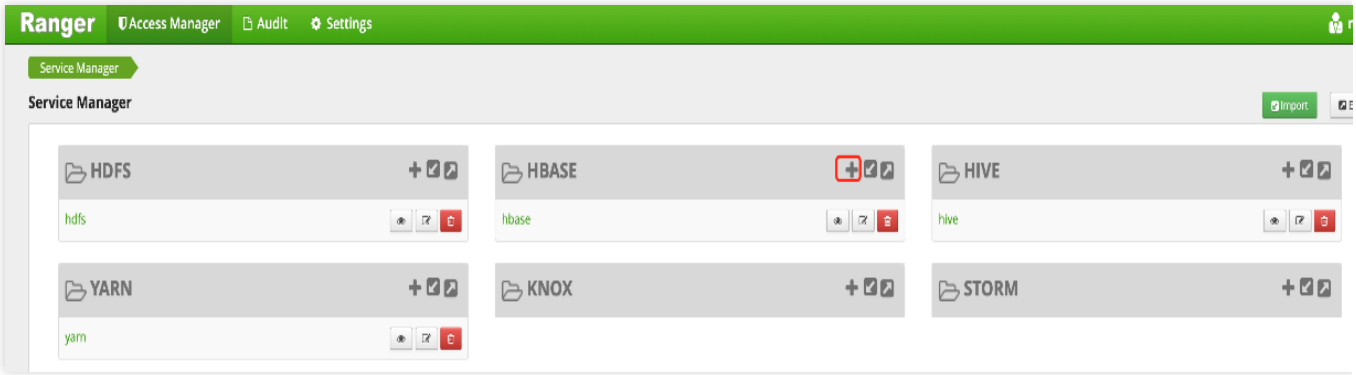
Integrating HBase with Ranger

Note:

Make sure that HBase related services are running normally and Ranger has been installed in the current cluster.

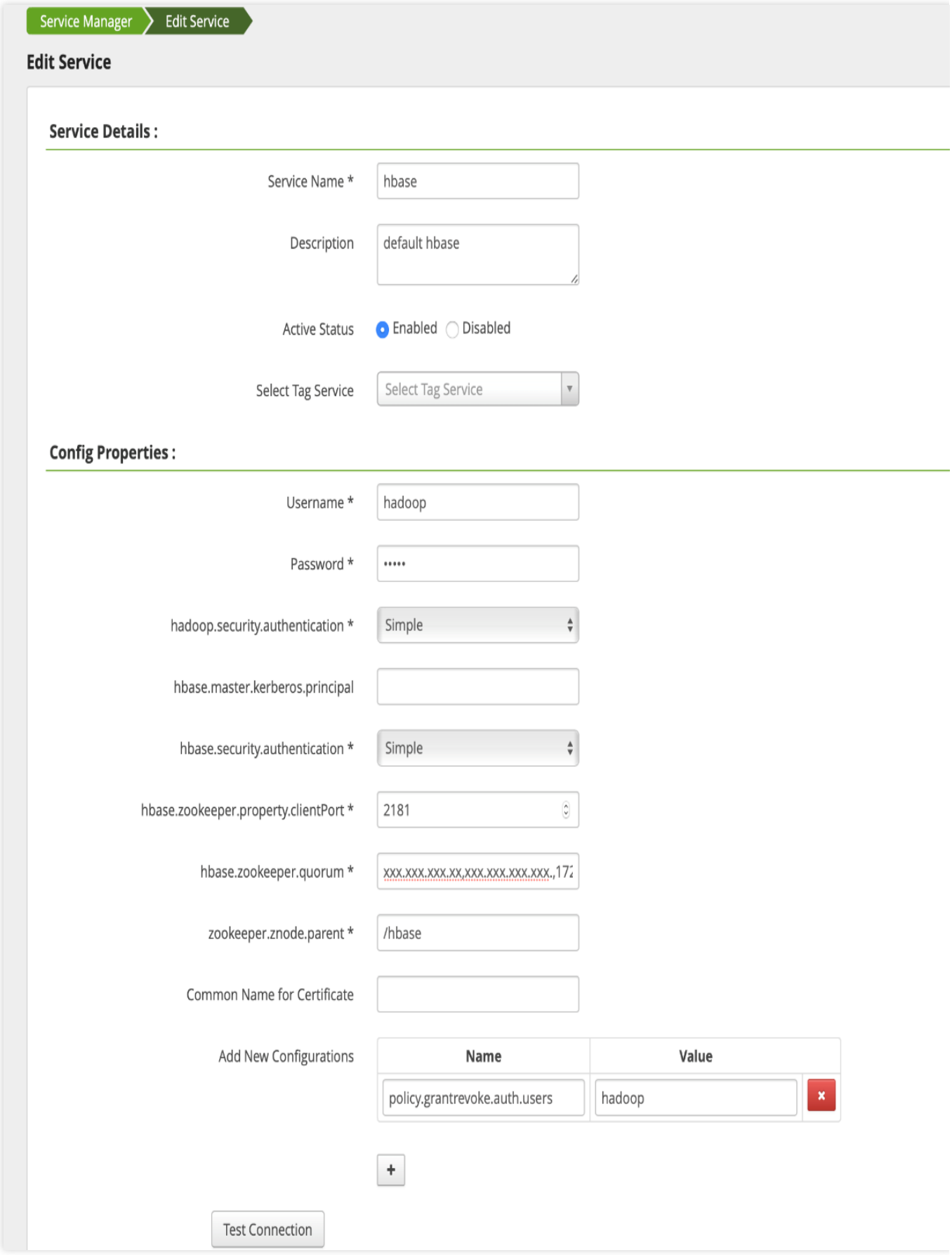
1. Add an EMR Ranger HBase service on the EMR Ranger Web UI.

©2013-2024 Tencent Cloud. All rights reserved.



2. Configure EMR Ranger HBase service parameters.

©2013-2024 Tencent Cloud. All rights reserved.

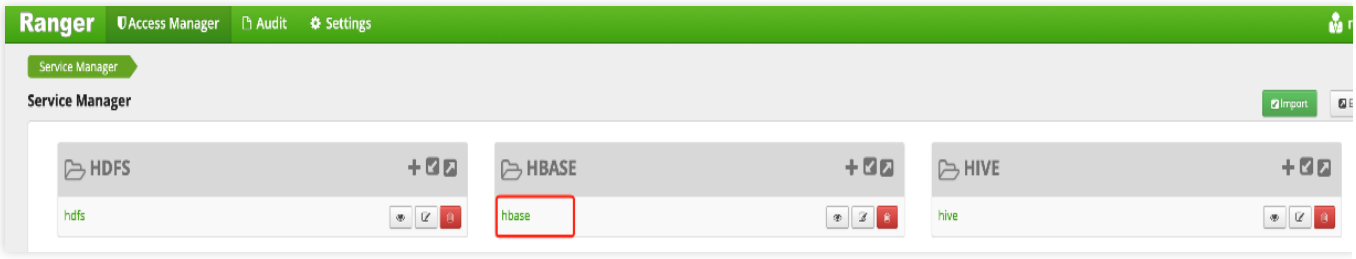


Parameter	Required	Description

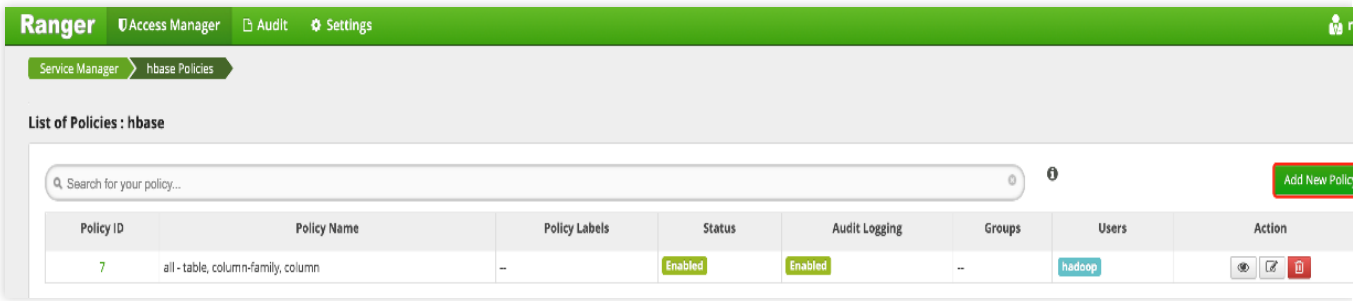
©2013-2024 Tencent Cloud. All rights reserved.

Service Name	Yes	Service name, which is displayed on the main HBase component on the Ranger Web UI
Description	No	Service description
Active Status	Default	Service status, which is **Enabled** by default
Username	Yes	Username of the resource
Password	Yes	User password
Hbase.zookeeper.property.clientPort	Yes	Request port of the ZooKeeper client
Hbase.zookeeper.quorum	Yes	ZooKeeper cluster IP
Zookeeper.znode.parent	Yes	ZooKeeper node information

3. Configure EMR Ranger HBase resource permissions.
Click the configured EMR Ranger HBase service.

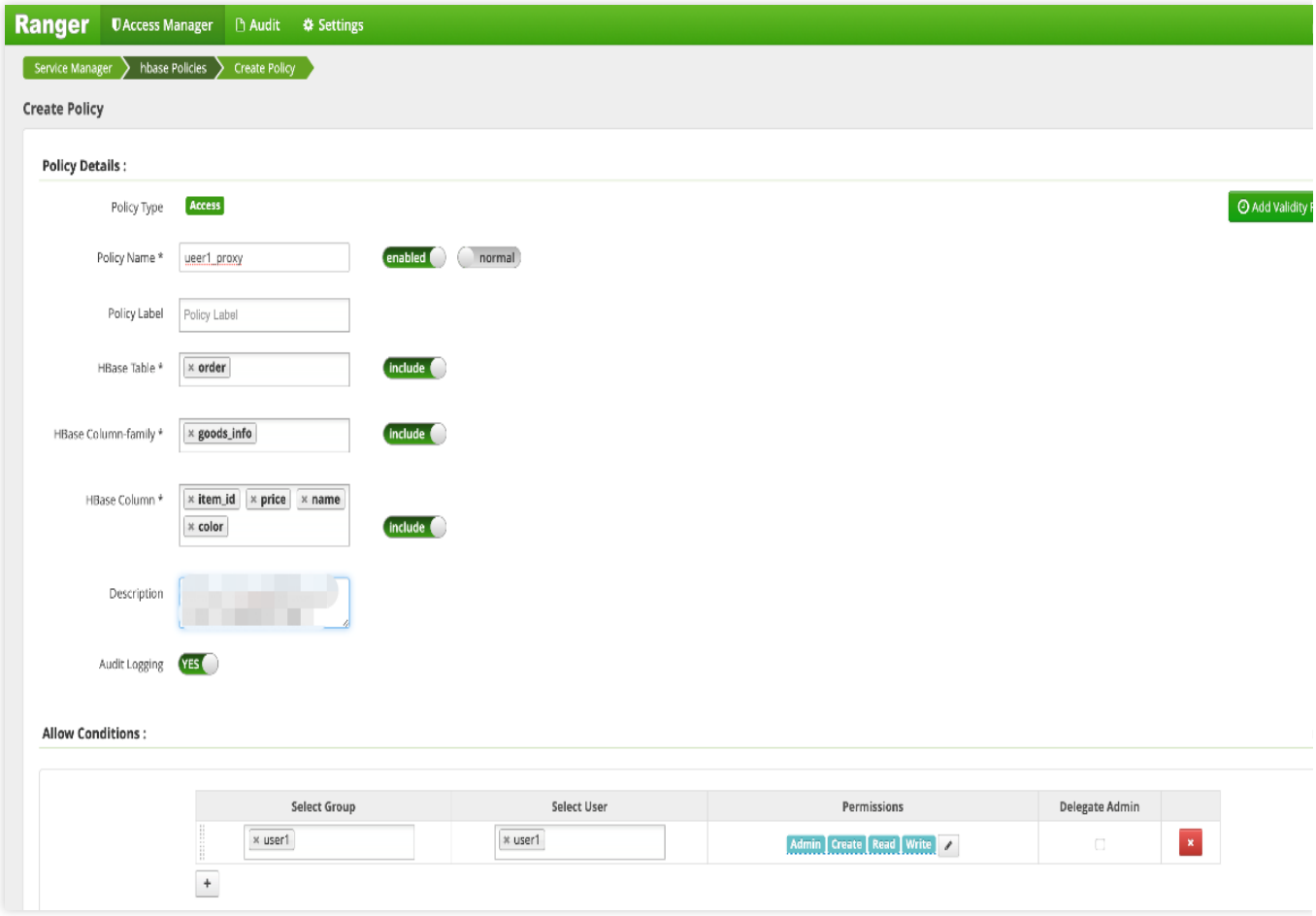


Configure a policy.



In the above figure, the **Users** is **hadoop** and **Policy Name** is **all-table, column-family, column**, meaning HBase users have Region Balance, MemeStore Flush, Compaction, and Split permissions. **Make sure that the created service has these permissions.**

©2013-2024 Tencent Cloud. All rights reserved.



Parameter	Required	Description
HBase Table	Yes	HBase table name
HBase Column-family	Yes	Column family of the HBase table
HBase Column	Yes	Qualifiers of the column family

4. After the policy is added, it will take effect in about 30 seconds, then you can use **user1** to perform operations on the column family and qualifiers of the **order** table.

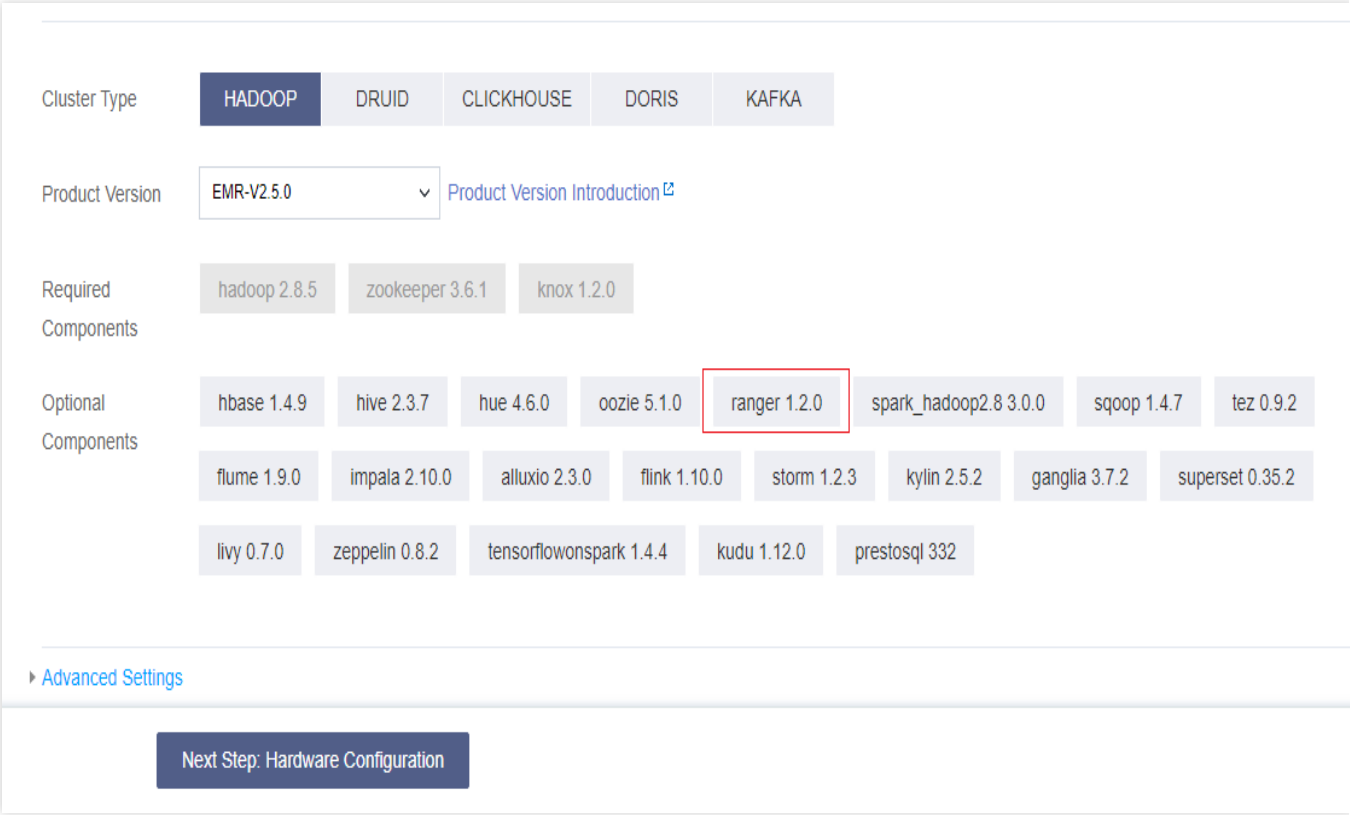
©2013-2024 Tencent Cloud. All rights reserved.

Integrating Presto with Ranger

Last updated：2025-01-03 15:02:25

Preparations

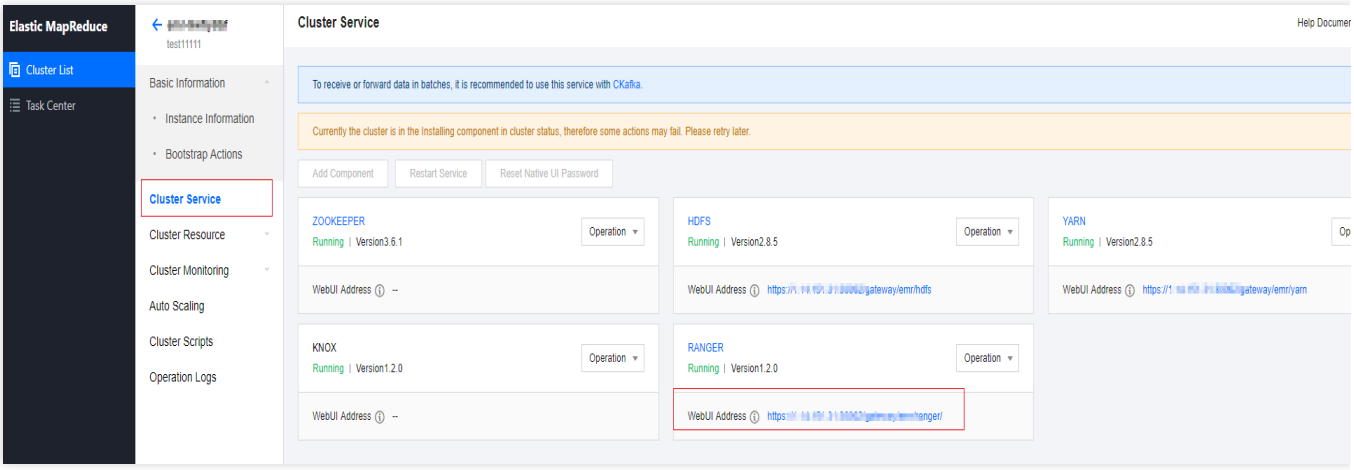
Ranger is available only when it is selected in **Optional Components** when you purchase a cluster. If you add the Ranger component after purchasing the cluster, the Web UI may be inaccessible. By default, when Ranger is installed, Ranger Admin and Ranger UserSync are deployed on the master node, and Ranger Plugin is deployed on the main daemon node of the embedded component.
When creating a cluster of the Hadoop type, you can select Ranger in **Optional Components**. The Ranger version varies depending on the EMR version you choose.
Note:
When the cluster type is Hadoop and the Ranger optional component is selected, EMR-Ranger will create services for HDFS and YARN by default and set default policies.



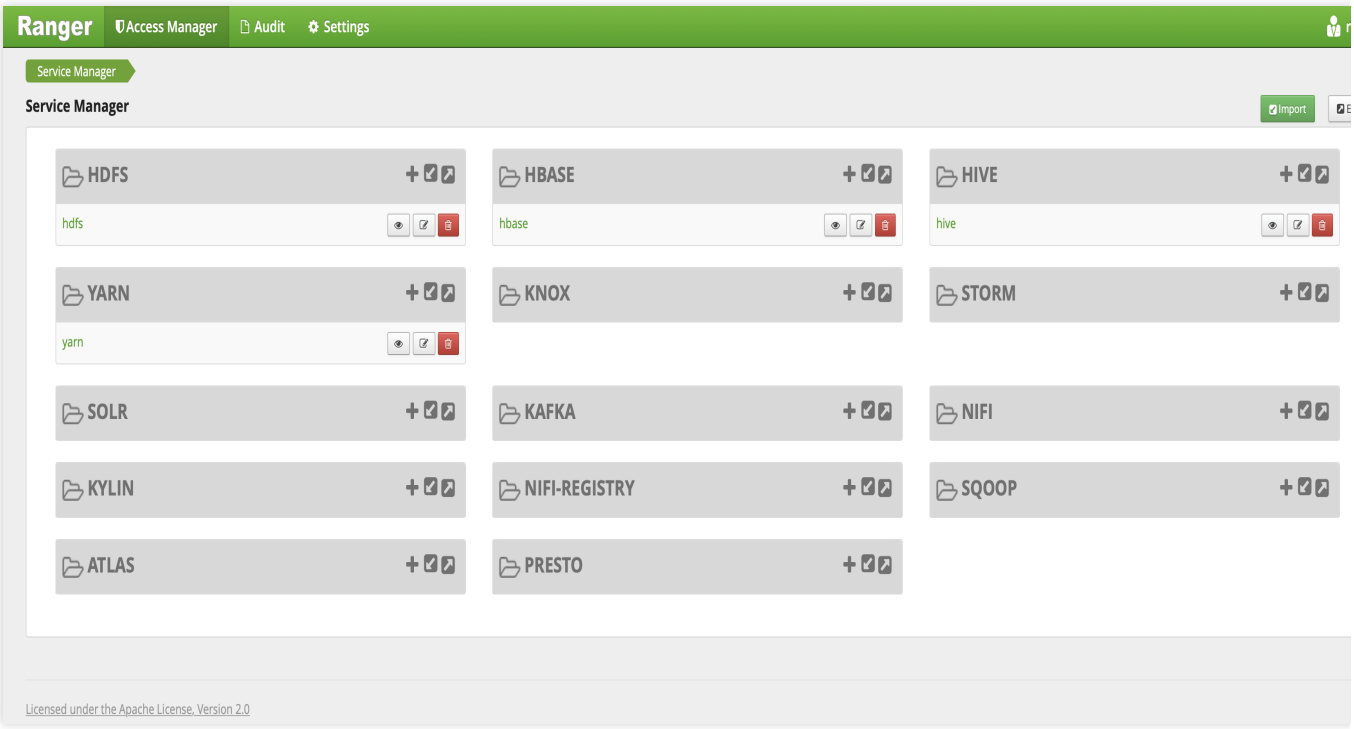
Ranger Web UI

©2013-2024 Tencent Cloud. All rights reserved.

Before accessing the Ranger Web UI, make sure that the current cluster is configured with a public IP and click the Ranger Web UI URL on the **Cluster Service** page.



After you are redirected, enter the username and password that you set when you purchased the cluster.



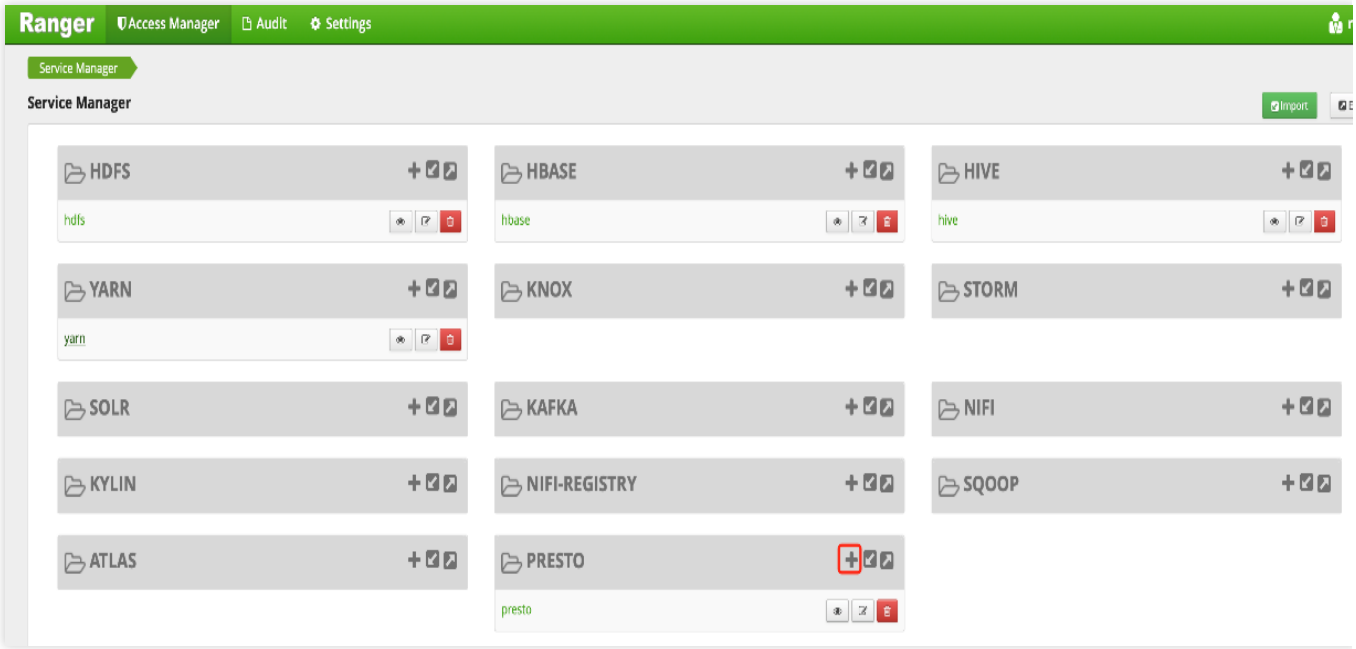
Integrating Presto with Ranger

Note:

Make sure that Presto related services are running normally and Ranger has been installed in the current cluster.

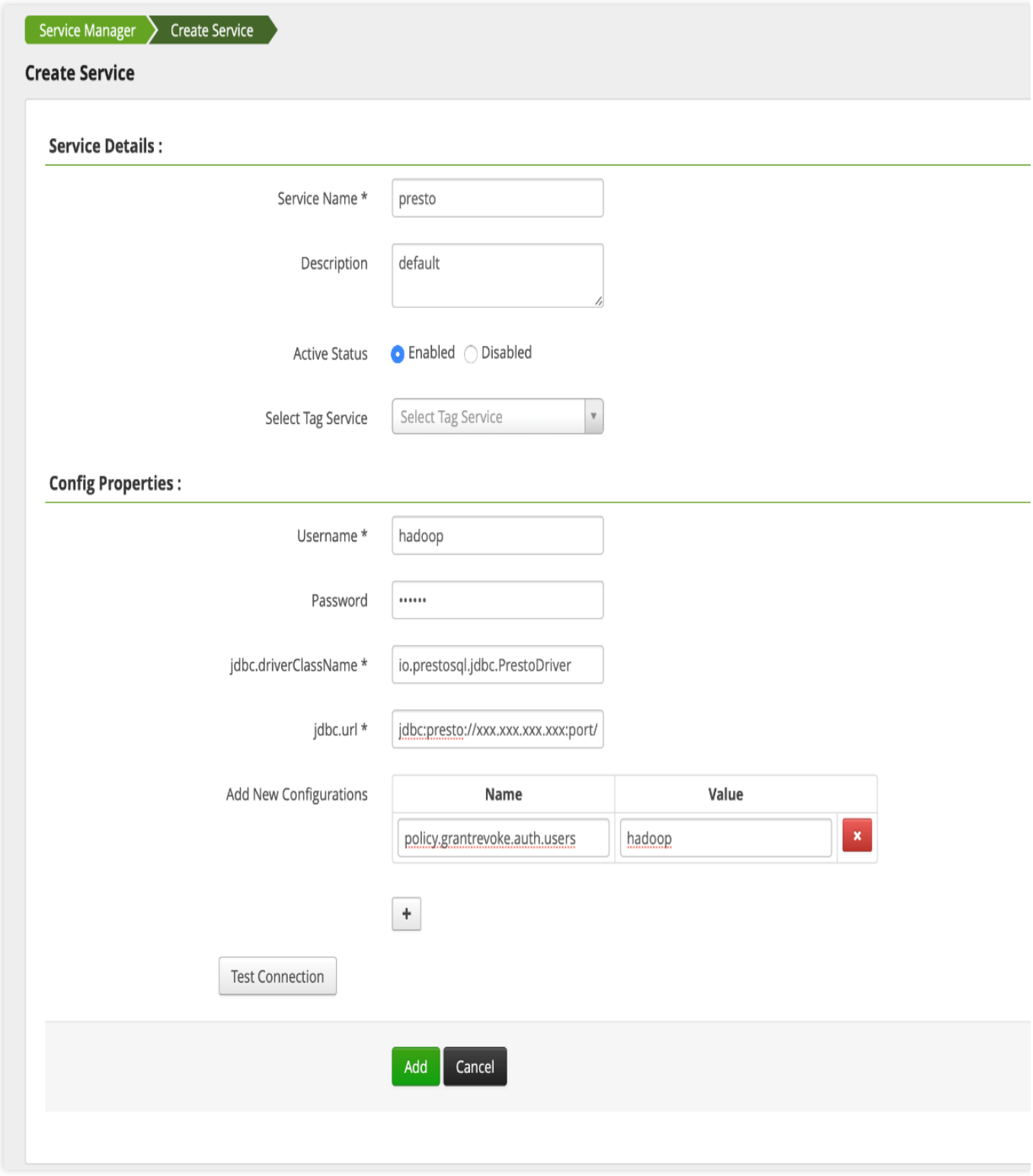
1. Add an EMR Ranger Presto service on the EMR Ranger Web UI.

©2013-2024 Tencent Cloud. All rights reserved.



2. Configure EMR Ranger Presto service parameters.

©2013-2024 Tencent Cloud. All rights reserved.



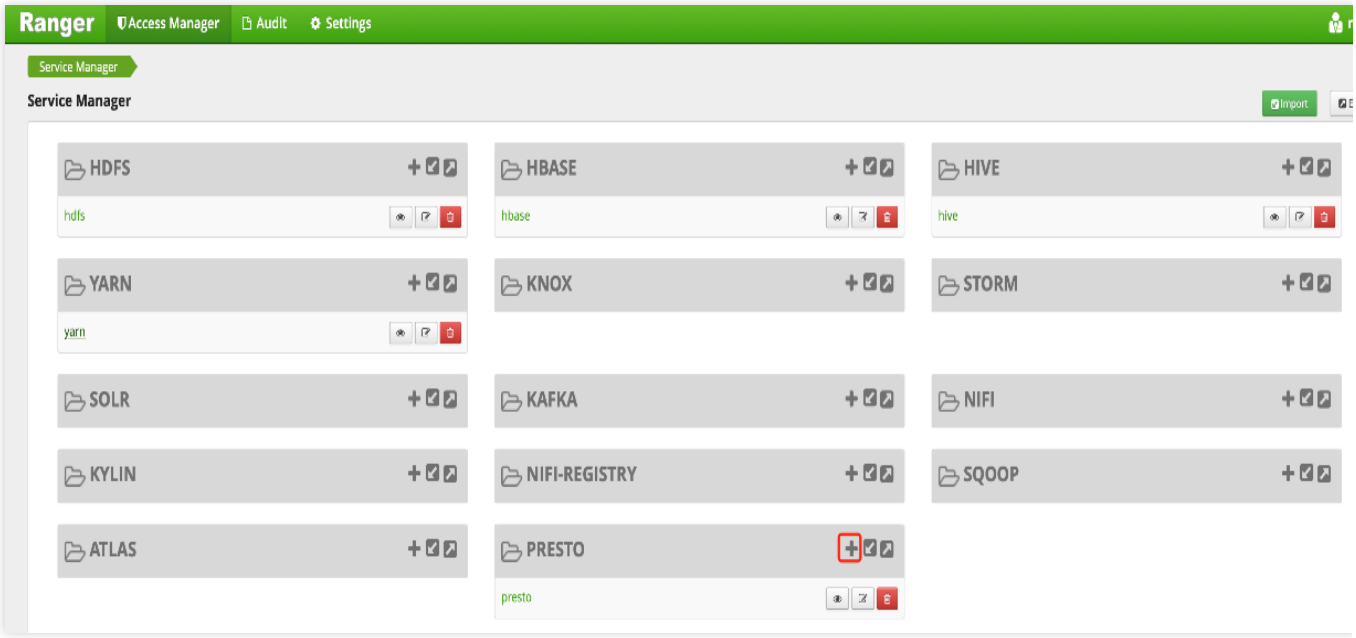
Parameter	Required	Description
Service Name	Yes	Service name, which is displayed on the main Presto component on the Ranger Web UI
Description	No	Service description

©2013-2024 Tencent Cloud. All rights reserved.

Active Status	Default	Service status, which is **Enabled** by default
Username	Yes	Username of the resource
Password	Yes	User password
JDBC.driverClassName	Yes	Full path of the drive class name
jdbc.url	Yes	Presto JDBC connection URL, for example, jdbc:presto://ip/hostname:port

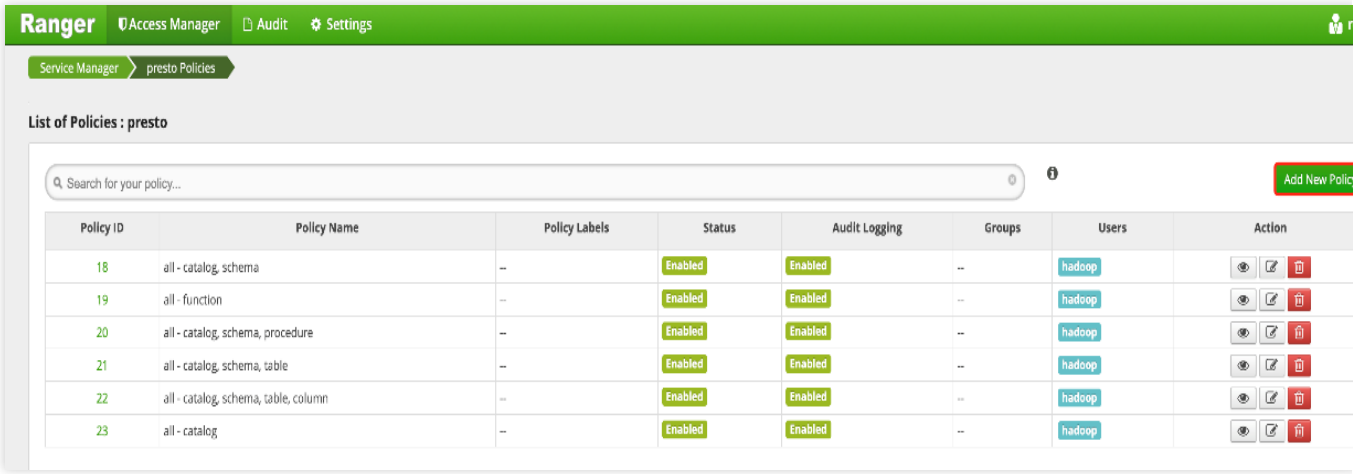
3. Configure EMR Ranger Presto resource permissions.

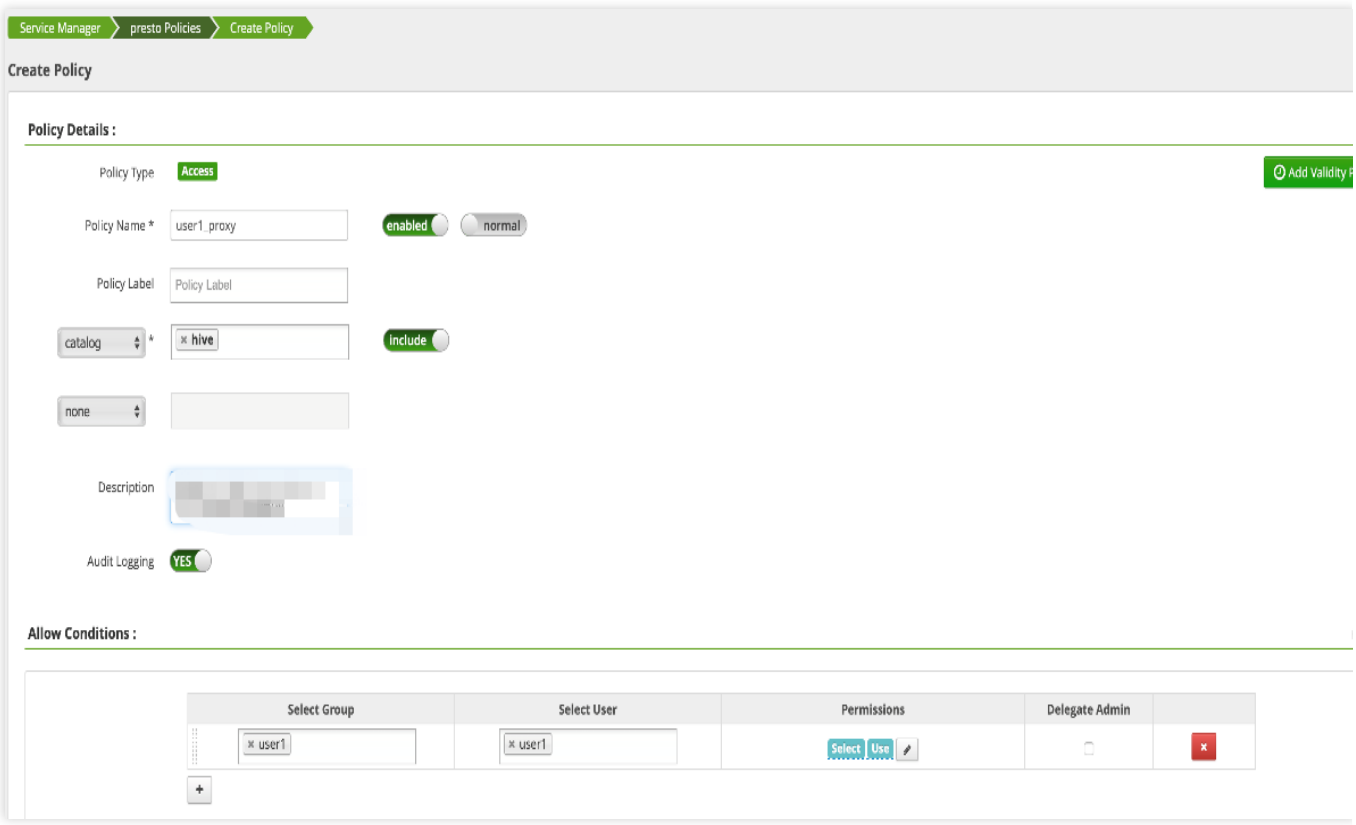
Click the configured EMR Ranger Presto service.



Configure a policy.

©2013-2024 Tencent Cloud. All rights reserved.





4. After the policy is added, it will take effect in about 30 seconds, then you can use **user1** to view and use the catalog of Presto.

©2013-2024 Tencent Cloud. All rights reserved.

Ranger Audit Log Guide
Storing Ranger Audit Logs in Solr

Last updated：2025-01-03 15:02:25

Ranger audit logs are disabled by default; you can enable the audit feature for the relevant components by modifying the configuration file as needed.

Note:

When you install Ranger in EMR-2.7.0 and later versions, a stand-alone Solr is installed by default on a single Master node to store Ranger's audit logs, with a retention period of 7 days.

1. To enable the audit feature, taking HDFS as an example, go to the EMR Console**> Cluster Services > HDFS > Configuration Management**, and modify the ranger-hdfs-audit.xml file as follows:

```
xasecure.audit.is.enabled=true
xasecure.audit.solr.is.enabled=true
```

2. In **Cluster Services > HDFS > Role Management**, restart the NameNode role.

3. If the customer needs to retain Ranger audit logs for a long term, it is recommended to store Ranger audit logs in Tencent Cloud ElasticSearch.

©2013-2024 Tencent Cloud. All rights reserved.

Storing Ranger Audit Logs in Tencent Cloud ElasticSearch

Last updated：2025-01-03 15:02:25

Prerequisites

1. Supports Ranger 2.1.0 and later versions, corresponding to EMR-2.7.0, EMR-3.2.1, and later.
2. Prepare a Tencent Cloud ElasticSearch cluster. For the creation process, see Creating ES cluster.
Capacity reference: Assuming the size of a single Hive SQL query is 1 KB, a single record will be approximately 20 KB, meaning 1 GB can store around 52,428 Hive SQL records.
3. Open port 9200 in the security group, allowing only private network access, requiring that the EMR cluster and ElasticSearch cluster are in the same VPC.
4. **Example:** Key information for the ElasticSearch cluster:
Username: elastic
Password: MyPassword
Private IP address: 10.206.48.118
Port: 9200
Protocol: http

EMR Console Directions

1. Log in to the EMR Console and go to **Cluster Services > RANGER > Configuration Management > ranger-admin-site.xml file**. Modify the configuration items as follows:

```
ranger.audit.source.type = elasticsearch
ranger.audit.elasticsearch.user = elastic
ranger.audit.elasticsearch.password = MyPassword
ranger.audit.elasticsearch.urls = 10.206.48.118
ranger.audit.elasticsearch.protocol = httpot
ranger.audit.elasticsearch.port = 9200
ranger.audit.elasticsearch.index = ranger_audits
ranger.audit.elasticsearch.bootstrap.enabled = true
```

2. Restart the Ranger service role EmbeddedServer.
3. Take HIVE as an example (other services are similar). Go to **Cluster Services > HIVE > Configuration Management** and modify the ranger-hive-audit.xml file with the following configuration, ensuring the values are consistent with those in the ranger-admin-site.xml file.

```
ranger.audit.source.type = elasticsearch
ranger.audit.elasticsearch.user = elastic
ranger.audit.elasticsearch.password = MyPassword
```

©2013-2024 Tencent Cloud. All rights reserved.

```
ranger.audit.elasticsearch.urls = 10.206.48.118
ranger.audit.elasticsearch.protocol = httpot
ranger.audit.elasticsearch.port = 9200
ranger.audit.elasticsearch.index = ranger_audits
ranger.audit.elasticsearch.bootstrap.enabled = true
```

4. Restart the HIVE role HiveServer2。

©2013-2024 Tencent Cloud. All rights reserved.

Kafka Development Guide
Kafka Overview

Last updated：2025-01-03 15:02:25

Tencent Cloud EMR-Kafka offers the cloud hosting service of open-source Kafka, with convenient Kafka cluster deployment, configuration modification, monitoring and alarming, and other features, providing enterprises and users with safe, stable OLAP solutions. Kafka data pipelines have been the most commonly used data sources and data sinks in stream computing systems. You can import streaming data into a certain topic in Kafka, process it through Flink operators, and output it to another topic in the same or a different Kafka instance. Kafka supports reading/writing data from/to multiple partitions in the same topic, which increases throughput and reduces data skew and hotspots.

Architecture

Both single-node and multi-node architectures are available for your choice based on business needs.

OPS

The console provides out-of-the-box services such as monitoring, log search, and parameter adjustment.

Features

Sending-receiving decoupling: the relationship between producers and consumers is effectively decoupled. Under the premise that the same API constraint is ensured, the processing between producers and consumers can be independently expanded or modified.
Flexibility: Kafka clusters are able to withstand sudden increases in requests without breakdown, effectively improving the robustness of the system.
Orderly reading and writing: Kafka clusters can guarantee the order of messages in a partition. Just like most message queue services, they can also ensure that data is processed in order, greatly improving disk efficiency.
Asynchronous communication: in scenarios where the business does not need to process messages immediately, Kafka clusters provide the asynchronous message processing mechanism. When the traffic is heavy, messages are put into the queue only, and they will be processed after the traffic become lighter, which relieves the system pressure.

Strengths

©2013-2024 Tencent Cloud. All rights reserved.

100% compatibility with open-source Kafka and easy migration

Kafka clusters are compatible with open-source Kafka v1.1.1.
The business system of Kafka clusters is based on the existing code of the open-source Apache Kafka ecosystem. Without any changes to your existing project, you can migrate to the cloud and enjoy the high-performance Kafka services provided by Tencent Cloud.

High performance

Tencent Cloud has improved the service performance, eliminating the need for complicated parameter configuration. You can upgrade or downgrade configurations on the UI and enjoy high-performance IaaS layer support.

High availability

Leveraging Tencent's years of experience in monitoring technologies, EMR offers comprehensive monitoring on clusters and has a professional OPS team in place that responds to alarms on a 24/7 basis to ensure the high availability of Kafka clusters.
Custom multi-AZ deployment in the same region is supported to improve disaster recovery ability.

High reliability

The disks are highly reliable, making it possible to keep services running even if 50% of the disks become faulty.
Two replicas are created by default, and up to three replicas can be used. The more replicas, the higher the reliability.

©2013-2024 Tencent Cloud. All rights reserved.

Use Cases

Last updated：2025-01-03 15:02:25

Webpage Behavior Analysis

Kafka clusters process website activities (PV, search, etc.) in real time and publish them to topics by type. These information flows can be used for real-time monitoring or offline statistical analysis.
A large amount of activity information is generated in each user's PV, therefore, website activity tracking requires high throughput. Kafka clusters perfectly meet the requirements of high throughput and offline processing.

Log Aggregation

Kafka clusters feature low-latency processing, supporting multiple data sources and distributed data processing (consumption). Compared with centralized log aggregation systems, Kafka provides better persistence and lower end-to-end latency while delivering the same performance.
The above features make Kafka clusters an ideal log collection center. Multiple servers/applications can asynchronously send operation logs in batches to Kafka clusters instead of saving them locally or in a DB. Kafka clusters can submit/compress messages in batches, and producers can hardly perceive the performance overhead. Consumers can use systematic storage and analysis systems such as Hadoop to analyze the pulled logs.

Online/Offline Analysis

In some big data scenarios, a large amount of concurrent data needs to be processed and aggregated. This requires clusters to have excellent processing performance and high scalability. Moreover, Kafka clusters' data distribution mechanism, in terms of disk space allocation, message format processing, server selection, and data compression, also makes them suitable for handling high numbers of real-time messages and aggregating distributed application data, which facilitates system OPS.
Kafka clusters can better aggregate, process, and analyze offline and streaming data.

©2013-2024 Tencent Cloud. All rights reserved.

Kafka Usage

Last updated：2025-01-03 15:02:25

Generating Data

By Java code

```
@Component
@Slf4j
public class KafkaProducer {

    @Autowired
    private KafkaTemplate<String, Object> kafkaTemplate;

    // Custom topics.
    public static final String TOPIC_TEST = "topic.test";

    //
    public static final String TOPIC_GROUP1 = "topic.group1";

    //
    public static final String TOPIC_GROUP2 = "topic.group2";

    public void send(Object obj) {
        String obj2String = JSONObject.toJSONString(obj);
        log.info("the message to send: {}", obj2String);
        // Send a message.
        ListenableFuture<SendResult<String, Object>> future =
kafkaTemplate.send(TOPIC_TEST, obj);
        future.addCallback(new ListenableFutureCallback<SendResult<String,
Object>>() {
            @Override
            public void onFailure(Throwable throwable) {
                // Returned result for failed sending
                log.info(TOPIC_TEST + " - the producer failed to send the
message:" + throwable.getMessage());
            }

            @Override
            public void onSuccess(SendResult<String, Object>
stringObjectSendResult) {
                // Returned result for successful sending
```

©2013-2024 Tencent Cloud. All rights reserved.

```
                log.info(TOPIC_TEST + " - the producer sent the message
successfully:" + stringObjectSendResult.toString());
            }
        });
    }
}
```

By command

```
bin/kafka-console-producer.sh --broker-list node86:9092 --topic t_cdr
```

Consuming Data

By Java code

```
@Component
@Slf4j
public class KafkaConsumer {

    @KafkaListener(topics = KafkaProducer.TOPIC_TEST, groupId =
KafkaProducer.TOPIC_GROUP1)
    public void topic_test(ConsumerRecord<?, ?> record, Acknowledgment ack,
@Header(KafkaHeaders.RECEIVED_TOPIC) String topic) {

        Optional message = Optional.ofNullable(record.value());
        if (message.isPresent()) {
            Object msg = message.get();
            log.info("topic_test consumed: Topic:" + topic + ",Message:" +
msg);
            ack.acknowledge();
        }
    }

    @KafkaListener(topics = KafkaProducer.TOPIC_TEST, groupId =
KafkaProducer.TOPIC_GROUP2)
    public void topic_test1(ConsumerRecord<?, ?> record, Acknowledgment ack,
@Header(KafkaHeaders.RECEIVED_TOPIC) String topic) {

        Optional message = Optional.ofNullable(record.value());
        if (message.isPresent()) {
            Object msg = message.get();
            log.info("topic_test1 consumed: Topic:" + topic + ",Message:" +
msg);
```

©2013-2024 Tencent Cloud. All rights reserved.

```
                ack.acknowledge();
            }
        }
    }
```

By command

```
bin/kafka-console-consumer.sh --zookeeper node01:2181 --topic t_cdr --from-
beginning
```

Adding a topic (by command)

```
bin/kafka-topics.sh --zookeeper node01:2181 --create --topic t_cdr --partitions
30  --replication-factor 2
```

For more information, see Kafka Documentation.

©2013-2024 Tencent Cloud. All rights reserved.

Iceberg Development Guide

Last updated：2025-01-03 15:02:25

Iceberg overview

Apache Iceberg is an open-source table format for large-scale data analytics and large, slow-moving tabular data storage. It is designed to improve the de facto standard table layout built into Hive, Trino (PrestoSQL), and Spark. Iceberg helps solve issues caused by the differences between data storage formats in lower layers and provides unified APIs for upper layers, so that different engines can access through the APIs.
Apache Iceberg capabilities:
Schema evolution: Supports five actions - Add, Drop, Update, Rename, and Reorder.
Partition layout evolution: Updates the layout of a table as data volume or query patterns change.
Hidden partitioning: With this capability, queries no longer depend on a table's physical layout. Through a separation between physical and logical, Iceberg tables can evolve partition schemes over time as data volume changes. Misconfigured tables can be fixed without an expensive migration.
Time travel: Enables reproducible queries that use exactly the same table snapshot, or lets users easily examine changes.
Version rollback: Allows users to quickly correct problems by resetting tables to a good state.
Iceberg boasts high reliability and performance. It can be used in production where a single table contain tens of petabytes of data and these huge tables can be read even without a distributed SQL engine.
Fast scan planning: A distributed SQL engine isn't needed to read a table or find files.
Advanced filtering: Data files are pruned with partition and column-level statistics, based on table metadata.
Iceberg is designed to solve correctness problems in eventually-consistent cloud object stores.
Works with any cloud store and reduces NameNode congestion in HDFS by avoiding listing and renaming.
Serializable isolation: Table changes are atomic and readers never see partial or uncommitted changes.
Multiple concurrent writers use optimistic concurrency control and will retry to ensure that compatible updates succeed, even when the writes conflict.
Iceberg is designed to manage data in tables in the form of snapshots. A snapshot is the state of a table at some time. Each snapshot is a complete list of files in a table at some time. Data files are stored across multiple manifest files, and the manifest files are listed in a single manifest list file. Manifest files can be shared among different manifest list files, and a manifest list file represents a snapshot.
A manifest list file is a metadata file that lists the manifest files. Each manifest file takes up a row.
A manifest file is a metadata file that lists the data files that make up a snapshot. Each row is a detailed description of each data file, including the file status, file path, partition information, column-level statistics (such as the maximum/minimum values and number of empty values in each column), file size, number of rows, etc.

©2013-2024 Tencent Cloud. All rights reserved.

A data file is where data is stored. Generally, data files are saved in the `data` directory under the table's data storage directory.

Example

For more examples, see here.

This document takes Iceberg 0.11.0 in EMR v3.3.0 as an example. The names of JAR packages may vary by EMR version.

1. Log in to the master node and switch to the `hadoop` user.
2. Iceberg packages are saved in `/usr/local/service/iceberg/` .
3. Use a compute engine to query data.

Spark engine

Spark-SQL interactive command line

```
spark-sql --master local[*] --conf
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExt
ensions --conf spark.sql.catalog.local=org.apache.iceberg.spark.SparkCatalog --
conf spark.sql.catalog.local.type=hadoop --conf
spark.sql.catalog.local.warehouse=/usr/hive/warehouse --jars
/usr/local/service/iceberg/iceberg-spark3-runtime-0.11.0.jar
```

Insert and query data.

```
CREATE TABLE local.default.t1 (id int, name string) USING iceberg;
INSERT INTO local.default.t1 values(1, "tom");
SELECT * from local.default.t1;
```

Hive engine

Use Beeline.

```
beeline -u jdbc:hive2://[hiveserver2_ip:hiveserver2_port] -n hadoop --hiveconf
hive.input.format=org.apache.hadoop.hive.ql.io.HiveInputFormat --hiveconf
hive.stats.autogather=false
```

Query data

```
ADD JAR /usr/local/service/iceberg/iceberg-hive-runtime-0.11.0.jar;
CREATE EXTERNAL TABLE t1 STORED BY
'org.apache.iceberg.mr.hive.HiveIcebergStorageHandler' LOCATION
'/usr/hive/warehouse/default/t1' TBLPROPERTIES
('iceberg.catalog'='location_based_table');

select count(*) from t1;
```

©2013-2024 Tencent Cloud. All rights reserved.

Flink engine

Download the corresponding version of the `flink-sql-connector-hive` package from the Maven repository based on the Flink and Hive versions. The following takes the Flink standalone mode as an example and uses the Flink Shell interactive command line.

```
wget https://repo1.maven.org/maven2/org/apache/flink/flink-sql-connector-hive-3.1.2_2.11/1.12.1/flink-sql-connector-hive-3.1.2_2.11-1.12.1.jar
/usr/local/service/flink/bin/start-cluster.sh
sql-client.sh embedded -j /usr/local/service/iceberg/iceberg-flink-runtime-0.11.0.jar -j flink-sql-connector-hive-3.1.2_2.11-1.12.1.jar shell
```

Query data

```
CREATE CATALOG hive_catalog WITH ('type'='iceberg','catalog-type'='hive','uri'='hivemetastore_ip:hivemetastore_port','clients'='5','property-version'='1','warehouse'='hdfs:///usr/hive/warehouse/');
CREATE DATABASE hive_catalog.iceberg_db;
CREATE TABLE hive_catalog.iceberg_db.t1 (id BIGINT COMMENT 'unique id',data STRING);
INSERT INTO hive_catalog.iceberg_db.t1 values(1, 'tom');
SELECT count(*) from hive_catalog.iceberg_db.t1;
```

©2013-2024 Tencent Cloud. All rights reserved.

StarRocks Development Guide
StarRocks Overview

Last updated：2025-01-03 15:02:25

StarRocks Overview

StarRocks is a new-generation and high-speed MPP database for nearly all data analytics scenarios.
StarRocks takes advantage of the relational Online Analytical Processing (OLAP) database and distributed storage system. Through architectural upgrades and functional optimization, StarRocks has developed into an enterprise-level product.
StarRocks is committed to accommodating multiple data analysis scenarios for enterprise users. It supports multiple data warehouse schemas (flat tables, pre-aggregations, star or snowflake schema) and various data import methods (batch and streaming) for up to 10,000 columns of data. It allows direct access to data from Hive, MySQL and Elasticsearch without importing.
StarRocks is compatible with the MySQL protocol. You can use the MySQL client and common Business Intelligence (BI) tools to connect to StarRocks for data analysis.
StarRocks uses a distributed architecture to divide the table horizontally and store it in multiple replications. The clusters are highly scalable and therefore support 10PB-level data analysis, Massively Parallel Processing (MPP), and data replication and elastic fault tolerance.
Leveraging a relational model, strong data typing, and a columnar storage engine, StarRocks reduces read-write amplification through encoding and compression techniques. Using vectorized query execution, it fully unleashes the power of parallel computing on multicore CPUs, therefore significantly improves query performance.

StarRocks Features

StarRocks integrates the design ideas of MPP databases and distributed systems into its architecture and has the following features:

Simplified architecture

StarRocks performs the specific execution of SQL through the MPP computing framework. The framework itself can make full use of the computing power of multiple nodes and execute the entire query in parallel, so as to deliver an excellent interactive analysis experience. StarRocks does not rely on any external systems. Its clean architecture makes it easier to deploy, maintain, and scale out. Its minimalist architectural design reduces its complexity and

©2013-2024 Tencent Cloud. All rights reserved.

maintenance cost and increases its reliability and scalability. Admins only need to focus on the StarRocks system itself, with no need to learn and manage other external systems.

Native vectorized SQL engine

The computing layer of StarRocks fully adopts the vectorization technology to systematically optimize all operators, functions, scanning, filtering, and import and export modules. Through the columnar memory layout and the SIMD instruction set adapted to CPU, it makes full use of the parallel computing power of CPU, achieving sub-second query returns in multidimensional analyses.

Query optimization

StarRocks can optimize complex queries through cost-based optimizer (CBO). The execution costs can be reasonably estimated based on the statistical information with no human intervention required. With a better execution plan, the data analysis efficiency in ad hoc and ETL scenarios can be greatly improved.

Query federation

StarRocks enables you to perform federated queries by using external tables. Currently, it supports tables from Hive, MySQL, and Elasticsearch. In this way, you can quickly query data without importing data.

Efficient update

StarRocks supports multiple data models. Among them, the update model can perform UPSERT/DELETE operations according to the primary key and achieve efficient query during concurrent updates through storage and indexing optimization. This better serves real-time data warehouses.

Intelligent materialized view

StarRocks supports intelligent materialized views. Users can create materialized views and generate pre-aggregated tables to speed up aggregate queries. StarRocks' materialized view automatically runs the aggregation when data is imported, keeping it consistent with the original table. When querying, users do not need to specify a materialized view, StarRocks can automatically select the best-materialized view to satisfy the query.

Standard SQL

StarRocks supports standard SQL syntax, including aggregation, join, sorting, window functions, and custom functions. It also fully supports 22 SQL queries from TPC-H and 99 SQL queries from TPC-DS. In addition, it is compatible with the MySQL protocol, so you can use various existing client tools and BI software programs to access StarRocks and perform data analysis with simple drag-and-drops in StarRocks.

Unified batch processing and streaming

StarRocks supports batch and streaming import of up to 10,000 columns of data in ORC, Parquet, and CSV formats from Kafka, HDFS, and local files. It can consume real-time Kafka data to import the data, which avoids data loss or

©2013-2024 Tencent Cloud. All rights reserved.

duplication (i.e., exactly once). It can also import data in batches from local or remote (HDFS) data sources.

High availability and scalability

StarRocks metadata and data are stored in multiple replicas. A StarRocks cluster provides hot standby services and can be deployed as multiple instances, eliminating single points of failure. It has the ability of self-healing and elastic recovery. Therefore, the overall stability of the cluster service will not be affected by node failures, disconnections, or exceptions. StarRocks adopts a distributed architecture that makes possible to horizontally scale the storage capacity and computing power. Specifically, a cluster can be expanded to hundreds of nodes to support up to 10 PB data storage. It can normally provide the query service during scaling. In addition, the table schema of StarRocks supports hot changes, so you can use a simple SQL command to dynamically modify the table definition, for example, adding or deleting a column or creating a materialized view. You can also import data into or query data from tables during schema changes.

Use Cases

StarRocks can meet a variety of analysis needs, including OLAP analysis, custom reporting, real-time data analysis, and ad hoc data analysis. Specific business scenarios include:
Multidimensional OLAP. User behavior analysis.
User profiling, tag analysis, and user tagging.
High-dimensional business metric reporting.
Self-service reporting.
Business problem identification and analysis.
Cross-topic business analysis.
Financial reporting.
System monitoring analysis.
Real-time data analysis. Ecommerce promotion data analysis.
Education live streaming quality analysis.
Waybill analysis.
Finance performance analysis and metric calculation.
Ad placement analysis.
Management cockpit.
Application performance management (APM).
High-concurrency queries. Advertiser report analysis.
Retail channel personnel analysis.
User-oriented analysis reporting in the SaaS industry.
Multi-page dashboard analysis.

©2013-2024 Tencent Cloud. All rights reserved.

Unified analysis. The same system is used to address the needs in different scenarios, such as multidimensional analysis, high-concurrency queries, precomputing, real-time analysis, and ad hoc query, simplifying the system and reducing the cost of multi-technology stack development and maintenance.

©2013-2024 Tencent Cloud. All rights reserved.

User Guide

Last updated：2025-01-03 15:02:25

Currently, StarRocks can be connected to in different ways. This document describes how to use a MySQL client to connect to StarRocks for development.

Logging in as Root User

Use the MySQL client to connect to the query port (9030) of an FE instance. StarRocks is preconfigured with the root user, whose password is the cluster password:

```
mysql -h fe_host -P9030 -u root -p
```

Clean up the environment:

```
mysql > drop database if exists example_db;

mysql > drop user test;
```

Viewing Deployed Nodes

1. View the FE node.

```
mysql> SHOW PROC '/frontends'\\G

*************************** 1. row ***********************
          Name: 172.16.139.24_9010_1594200991015
            IP: 172.16.139.24
      HostName: starrocks-sandbox01
   EditLogPort: 9010
      HttpPort: 8030
     QueryPort: 9030
       RpcPort: 9020
          Role: FOLLOWER
      IsMaster: true
     ClusterId: 861797858
          Join: true
         Alive: true
ReplayedJournalId: 64
 LastHeartbeat: 2020-03-23 20:15:07
```

©2013-2024 Tencent Cloud. All rights reserved.

```
      IsHelper: true
        ErrMsg:
1 row in set (0.03 sec)
```

The value of `Role` is `FOLLOWER`, indicating that the FE is eligible for master election. The value of `IsMaster` is `true`, indicating that the FE is the current master node.

2. Viewing the BE node.

```
mysql> SHOW PROC '/backends'\\G

********************* 1. row **********************
         BackendId: 10002
           Cluster: default_cluster
                IP: 172.16.139.24
          HostName: starrocks-sandbox01
     HeartbeatPort: 9050
            BePort: 9060
          HttpPort: 8040
          BrpcPort: 8060
     LastStartTime: 2020-03-23 20:19:07
     LastHeartbeat: 2020-03-23 20:34:49
             Alive: true
 SystemDecommissioned: false
ClusterDecommissioned: false
         TabletNum: 0
  DataUsedCapacity: .000
     AvailCapacity: 327.292 GB
     TotalCapacity: 450.905 GB
           UsedPct: 27.41 %
    MaxDiskUsedPct: 27.41 %
            ErrMsg:
           Version:
1 row in set (0.01 sec)
```

If the value of `isAlive` is `true`, the BE is connected to the cluster normally; if not, view the `be.WARNING` log file in the `log` directory to identify the cause.

3. View the broker node.

```
MySQL> SHOW PROC "/brokers"\\G
*************************** 1. row ***************************
       Name: broker1
         IP: 172.16.139.24
       Port: 8000
      Alive: true
 LastStartTime: 2020-04-01 19:08:35
LastUpdateTime: 2020-04-01 19:08:45
```

©2013-2024 Tencent Cloud. All rights reserved.

```
      ErrMsg:
1 row in set (0.00 sec)
```

If the value of `Alive` is `true`, the status is normal.

Creating User

Create an ordinary user with the following command:

```
mysql > create user 'test' identified by '123456';
```

Creating Database

In StarRocks, only the root account has the permission to create databases. Log in as the root user and create the `example\\_db` database:

```
mysql > create database example_db;
```

After creating the database, you can view the database information through `show databases`:

```
mysql > show databases;

+--------------------+
| Database           |
+--------------------+
| example_db         |
| information_schema |
+--------------------+
2 rows in set (0.00 sec)
```

`information_schema` exists to be compatible with the MySQL protocol. In practice, information may not be accurate. Therefore, we recommend you get the information of a specific database by directly querying the database.

Account Authorization

After creating the `example_db` database, you can grant its read/write permissions to the test account through the root account and then log in with the test account to manipulate the database.

```
mysql > grant all on example_db to test;
```

©2013-2024 Tencent Cloud. All rights reserved.

Log out of the root account and log in to the StarRocks cluster with the test account:

```
mysql > exit

mysql -h 127.0.0.1 -P9030 -utest -p123456
```

Creating Table

StarRocks supports two table creation methods: single partitioning and composite partitioning.
In composite partitioning:
The first level is called Partition, or partitioning. You can specify a dimension column as a partitioning column (currently only integer and time columns are supported) and specify the value range for each partition.
The second level is called Distribution, i.e., bucketing. You can specify multiple dimension columns (or specify none, in which case, all `KEY` columns will be selected) and the number of buckets for HASH distribution of data.
Composite partitioning is recommended for the following scenarios:
Scenarios involving time dimensions or similar dimensions with ordered values: You can use such dimension columns as a partitioning column and assess the partitioning granularity based on the import frequency and the amount of partitioned data.
Historical data deletion: If you want to retain only data from the past N days, you can delete historical partitions through composite partitioning or delete data by sending a `DELETE` statement to the specified partition.
Data skew elimination: You can specify the number of buckets for each partition separately. If you partition data by day, when the amount of data varies greatly every day, you can reasonably partition the data by specifying the number of buckets for the partitions. We recommend you select distinguishing columns as the bucket columns.
You can also use single partitioning instead of composite partitioning. Then, the data will be only distributed by HASH.
The following demonstrates the table creation statements for the two partitioning methods respectively:
1. Switch the database from `mysql` to `use example_db` .
2. Create a single-partitioned logical table named `table1` by assigning ten hash buckets based on `siteid` , with the schema as shown below:
`siteid` : The type is `INT` (4 bytes); the default value is `10` .
`city_code` : The type is `SMALLINT` (two bytes).
`username` : The type is `VARCHAR` . The maximum length is `32` . The default value is an empty string.
`pv` : The type is `BIGINT` (eight bytes), and the default value is `0` . It is a metric column and will be aggregated by StarRocks with the `SUM` method. The aggregation model is used here. In addition, StarRocks supports the detail model and update model. For more information, see Data Model.
The table creation statement is as follows:

```
mysql >
CREATE TABLE table1
```

©2013-2024 Tencent Cloud. All rights reserved.

```
(
 siteid INT DEFAULT '10',
 citycode SMALLINT,
 username VARCHAR(32) DEFAULT '',
 pv BIGINT SUM DEFAULT '0'
)
AGGREGATE KEY(siteid, citycode, username)
DISTRIBUTED BY HASH(siteid) BUCKETS 10
PROPERTIES("replication_num" = "1");
```

3. Create a composite partitioned table

Create a logical table named `table2` with the schema as shown below:

`event_day` : The type is `DATE` ; no default value.

`siteid` : The type is `INT` (4 bytes); the default value is `10` .

`city_code` : The type is `SMALLINT` (two bytes).

`username` : The type is `VARCHAR` . The maximum length is `32` . The default value is an empty string.

`pv` : The type is `BIGINT` (eight bytes), and the default value is `0` . It is a metric column and will be aggregated by StarRocks with the `SUM` method.

Use the `event_day` column as the partitioning column to create three partitions: p1, p2, and p3.

p1: Its value range is [minimum value, 2017-06-30].

p2: Its value range is [2017-06-30, 2017-07-31).

p3: Its value range is [2017-07-31, 2017-08-31).

Use `siteid` to assign ten hash buckets to each partition.

The table creation statement is as follows:

```
CREATE TABLE table2
(
event_day DATE,
siteid INT DEFAULT '10',
citycode SMALLINT,
username VARCHAR(32) DEFAULT '',
pv BIGINT SUM DEFAULT '0'
)
AGGREGATE KEY(event_day, siteid, citycode, username)
PARTITION BY RANGE(event_day)
(
PARTITION p1 VALUES LESS THAN ('2017-06-30'),
PARTITION p2 VALUES LESS THAN ('2017-07-31'),
PARTITION p3 VALUES LESS THAN ('2017-08-31')
)
DISTRIBUTED BY HASH(siteid) BUCKETS 10
PROPERTIES("replication_num" = "1");
```

©2013-2024 Tencent Cloud. All rights reserved.

After creating the table, you can view the information of the table in `example\\_db` :

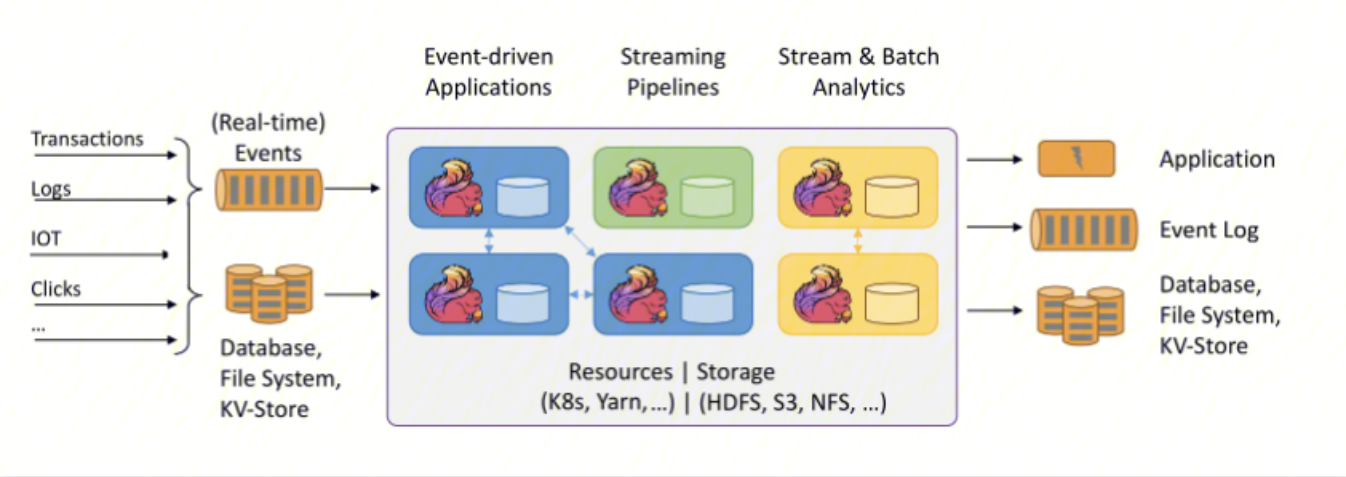
```
mysql> show tables;

+-------------------------+
| Tables_in_example_db    |
+-------------------------+
| table1                  |
| table2                  |
+-------------------------+
2 rows in set (0.01 sec)


mysql> desc table1;

+----------+-------------+------+-------+---------+-------+
| Field    | Type        | Null | Key   | Default | Extra |
+----------+-------------+------+-------+---------+-------+
| siteid   | int(11)     | Yes  | true  | 10      |       |
| citycode | smallint(6) | Yes  | true  | N/A     |       |
| username | varchar(32) | Yes  | true  |         |       |
| pv       | bigint(20)  | Yes  | false | 0       | SUM   |
+----------+-------------+------+-------+---------+-------+
4 rows in set (0.00 sec)


mysql> desc table2;

+-----------+-------------+------+-------+---------+-------+
| Field     | Type        | Null | Key   | Default | Extra |
+-----------+-------------+------+-------+---------+-------+
| event_day | date        | Yes  | true  | N/A     |       |
| siteid    | int(11)     | Yes  | true  | 10      |       |
| citycode  | smallint(6) | Yes  | true  | N/A     |       |
| username  | varchar(32) | Yes  | true  |         |       |
| pv        | bigint(20)  | Yes  | false | 0       | SUM   |
+-----------+-------------+------+-------+---------+-------+
5 rows in set (0.00 sec)
```

©2013-2024 Tencent Cloud. All rights reserved.

Flink Development Guide
Flink Overview

Last updated：2025-01-03 15:02:25

Flink is an open-source distributed, high-performance, highly available, and accurate data stream execution engine. It provides diverse features such as data distribution, data communication, and fault tolerance for distributed computations over data streams. Based on the stream execution engine, Flink provides APIs at a higher abstraction layer for you to write distributed jobs.
Distributed: Flink can run on multiple machines.
High-performance: Flink has a high processing performance.
Highly available: Flink supports an automatic application restart mechanism.
Accurate: Flink can ensure the accuracy of data processing.



As shown in the figure, the data source on the left contains real-time logs or data in the database, file system, or key-value storage system. Flink in the middle organizes the data and outputs the computed data to the destination on the right, which can be an application system or storage system. In summary, Flink has three core components:
Data source: The data source on the left.
Transformations: Operators, which process data.
Data sink: Output component, which outputs the computed data to other application systems.

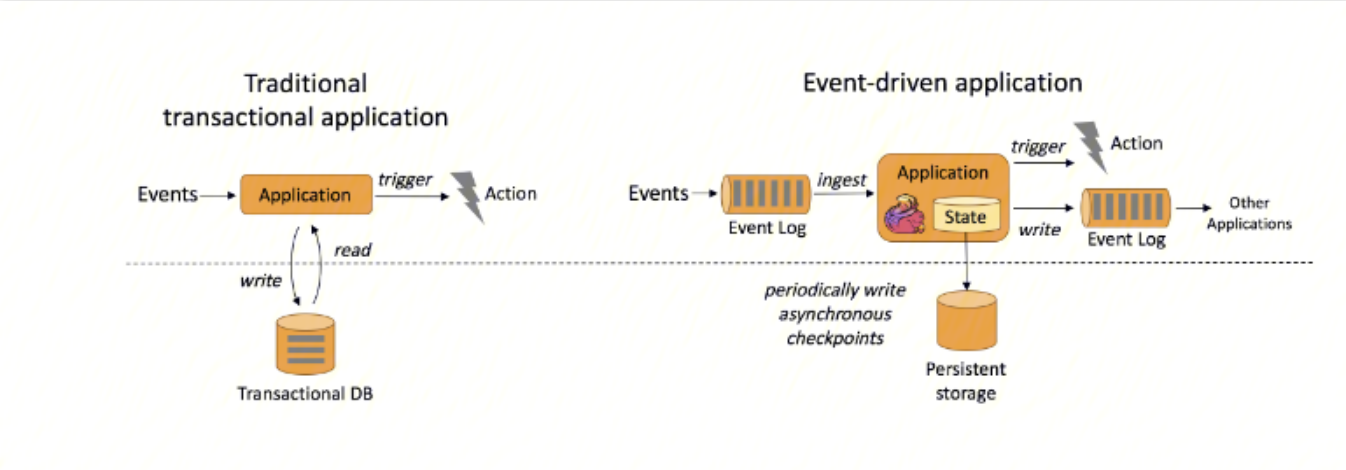
Use cases

Flink has the following three use cases:

©2013-2024 Tencent Cloud. All rights reserved.

1. Event-driven applications

An event-driven application is stateful. It ingests data from one or more event streams and triggers computations, status updates, or external actions based on the events.



In a traditional architecture (left), an application needs to read/write data from/to a remote transactional database, such as MySQL. For an event-driven application, data and computation are co-located. The application only needs to access the local memory or disk to get data, so it delivers a higher throughput and lower latency.

Flink supports event-driven applications by virtue of the following features:

Efficient status management: Flink comes with state backends, which store the intermediate status information.

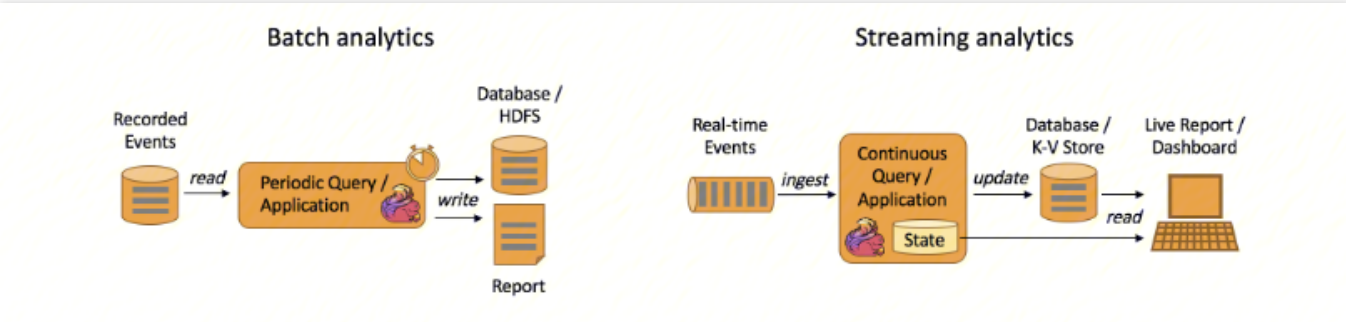
Rich windows: Flink has tumbling, sliding, and other windows.

Various notions of time: Flink supports event time, processing time, and ingestion time.

Flink supports "at-least-once" and "exactly-once" fault tolerance levels.

2. Real-time data analytics applications

Analytical jobs extract valuable information and metrics from raw data. Traditionally, batch queries are used, or events are recorded to form an application based on the limited data set. To get the analysis result of the latest data, this mode needs to add the data to the data set, perform the query or run the application again, and write the result to a storage system or generate a report.



©2013-2024 Tencent Cloud. All rights reserved.

3. Real-time data warehouse and ETL

Extract, Transform, and Load (ETL) is a process that extracts, transforms, and loads data in a business system to a data warehouse.

In traditional mode, the offline data warehouse centrally stores business data and performs ETL and runs other models at regular intervals based on the fixed computing logic to generate reports. It mainly builds T+1 offline data, pulls incremental data every day through periodic jobs, creates topic-level data related to different businesses, and provides the T+1 data query API externally.

The above figure compares the offline data warehouse ETL and real-time data warehouse. As can be seen, the offline mode is inferior in terms of computing and data real-timeness. Data becomes less valuable over time and needs to reach the user as soon as possible; therefore, real-time data warehouses are demanded.

For more information on API layers, see the following documents:

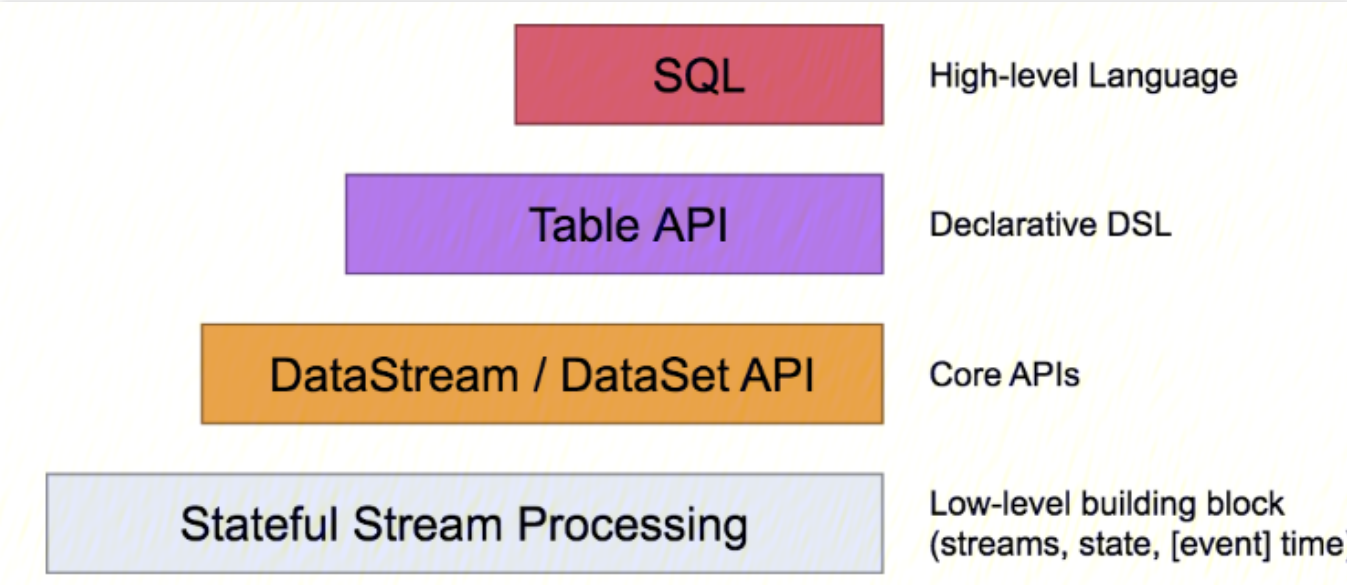


Table API & SQL: The Table API is tightly integrated with the DataSet or DataStream. You can create a table through a DataSet or DataStream and convert it into a DataSet or DataStream through operations such as FILTER, SUM,

©2013-2024 Tencent Cloud. All rights reserved.

JOIN, and SELECT. The SQL API is based on Apache Calcite at the underlying layer and is more flexible than other APIs, as Apache Calcite implements the SQL standard to allow for the direct use of SQL statements. The Table API and SQL API can work together as both of them return table objects.
DataStream API and DataSet API: They mainly process streaming data and batch data and encapsulate lower-level APIs to support higher-order functions such as FILTER, SUM, MAX, and MIN. They are easy to use and popular in actual applications.
Stateful Stream Processing: It provides time- and status-based control and is a bit complex and hard to use. It mainly applies to the logic of complex event processing.

Environment information

By default, Flink is deployed on the master and core nodes in a cluster. It is an out-of-the-box service.
After logging in, you can run the `su hadoop` command to switch to the `hadoop` user and then perform local tests.
The Flink software path is `/usr/local/service/flink` .
The log path is `/data/emr/flink/logs` .
For more information, see the community documentation.

©2013-2024 Tencent Cloud. All rights reserved.

Analyzing COS Data with Flink

Last updated：2025-01-03 15:02:25

Flink excels at processing unbounded and bounded data sets. Precise control of time and state enables Flink's runtime to run any kind of application on unbounded streams. Bounded streams are internally processed by algorithms and data structures that are specifically designed for fixed sized data sets, yielding excellent performance.
The following are the directions for bounded or unbounded data sets in COS. Here, the YARN Session Mode is used for job submission to better observe job running. Flink on YARN supports the Session Mode and Application Mode. For more information, see the community documentation.
The job submitted in this tutorial is a wordcount job, i.e., counting the number of words. You need to upload the file for counting to the cluster in advance.

Development Preparations

1. Create a bucket in COS for this job.
2. Create an EMR cluster. When creating the EMR cluster, you need to select the Flink component on the software configuration page and enable access to COS on the basic configuration page.
3. After purchasing the cluster, access COS with HDFS to ensure that its basic features are available. The specific commands are as follows:

```
[hadoop@10 ~]$ hdfs dfs -ls cosn://$BUCKET_NAME/path
Found 1 items
-rw-rw-rw-   1 hadoop hadoop      27040 2022-10-28 15:08
cosn://$BUCKET_NAME/path/LICENSE
```

Example

```
# `-n` indicates the number of containers applied for, i.e., the number of
TaskManagers.
# `-tm` indicates the memory size per TaskManager.
# `-s` indicates the number of slots per TaskManager.
# `-d` indicates to run as a backend application, which is followed by the
session name.
[hadoop@10 ~]$ yarn-session.sh -jm 1024 -tm 1024 -n 1 -s 1 -nm wordcount-
example -d
```

©2013-2024 Tencent Cloud. All rights reserved.

```
/usr/local/service/flink/bin/flink run -m yarn-cluster
/usr/local/service/flink/examples/batch/WordCount.jar --input
cosn://$BUCKET_NAME/path/LICENSE -output cosn://$BUCKET_NAME/path/wdp_test
[hadoop@10 ~]$ hdfs dfs -ls cosn://$BUCKET_NAME/path/wdp_test
-rw-rw-rw-   1 hadoop hadoop       7484 2022-11-04 00:47
cosn://$BUCKET_NAME/path/wdp_test
```

Maven Demo

Here, the demo that comes with the system is not used; instead, you need to create a project and compile, compress, and upload it to the EMR cluster on your own for execution. Maven is recommended for project management, as it can help you manage project dependencies with ease. Specifically, it can get .jar packages through the configuration of the `pom.xml` file, eliminating the need to add them manually.

1. Download and install Maven first and then configure its environment variables. If you are using the IDE, set the Maven configuration items in the IDE.
2. In the local shell environment, enter the directory where you want to create the Maven project, such as `D://mavenWorkplace`, and enter the following command to create it:

```
mvn archetype:generate -DgroupId=$yourgroupID -DartifactId=$yourartifactID -
DarchetypeArtifactId=maven-archetype-quickstart
```

Here, `yourgroupID` is your package name, `yourartifactID` is your project name, and `maven-archetype-quickstart` indicates to create a Maven Java project. Some files need to be downloaded during the project creation, so you should keep the network connected.

After successfully creating the project, you will see a folder named `$yourartifactID` in the `D://mavenWorkplace` directory. The files in the folder have the following structure:

```
simple
    ---pom.xml       Core configuration, under the project root directory
    ---src
        ---main
            ---java         Java source code directory
            ---resources    Java configuration file directory
        ---test
             ---java         Test source code directory
             ---resources    Test configuration directory
```

Among the files above, pay extra attention to the `pom.xml` file and the Java folder under the main directory. The `pom.xml` file is primarily used to create dependencies and package configurations; the Java folder is used to store

©2013-2024 Tencent Cloud. All rights reserved.

your source code.

First, add the Maven dependencies to pom.xml:

```
<properties>
    <scala.version>2.12</scala.version>
    <flink.version>1.14.3</scala.version>
</properties>

<dependencies>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-java</artifactId>
        <version>1.14.3</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-streaming-scala_${scala.version}</artifactId>
        <version>${flink.version}</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.apache.flink</groupId>
        <artifactId>flink-clients_${scala.version}</artifactId>
        <version>${flink.version}</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

Note

Use your actual `scala.version` and `flink.version` values.

Then, add the packaging and compiling plugins to `pom.xml` :

```
<build>
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
      <source>1.8</source>
      <target>1.8</target>
      <encoding>utf-8</encoding>
    </configuration>
  </plugin>
  <plugin>
    <artifactId>maven-assembly-plugin</artifactId>
```

©2013-2024 Tencent Cloud. All rights reserved.

```
    <configuration>
      <descriptorRefs>
      <descriptorRef>jar-with-dependencies</descriptorRef>
      </descriptorRefs>
    </configuration>
    <executions>
      <execution>
        <id>make-assembly</id>
        <phase>package</phase>
        <goals>
          <goal>single</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</plugins>
</build>
```

If your Maven is configured correctly and its dependencies are successfully imported, the project will be compiled directly. Enter the project directory in the local shell, and run the following command to package the entire project:

```
mvn package
```

Some files may need to be downloaded during the running process. "Build success" indicates that a package is successfully created. You can see the generated .jar package in the target folder under the project directory.

Data preparations

First, you need to upload the compressed .jar package to the EMR cluster using the scp or sftp tool by running the following command in local command line mode:

```
scp $localfile root@public IP address:$remotefolder
```

Here, `$localfile` is the path plus name of your local file; `root` is the CVM instance username. You can look up the public IP address in the node information in the EMR console or the CVM console. `remotefolder` is the path where you want to store the file in the CVM instance. After the upload is completed, you can check whether the file is in the corresponding folder on the EMR command line.

The file to be processed needs to be uploaded to COS in advance. If the file is in your local file system, you can upload it directly through the COS console; if it is in the EMR cluster, you can upload it by running the following Hadoop command:

```
[hadoop@10 hadoop]$ hadoop fs -put $testfile cosn://BUCKET_NAME/
```

Running the demo

©2013-2024 Tencent Cloud. All rights reserved.

First, log in to any node (preferably a master one) in the EMR cluster. For more information on how to log in to EMR, see Logging In To Linux Instance (Web Shell). Here, you can use WebShell to log in. Click **Login** on the right of the desired CVM instance to enter the login page. The default username is `root`, and the password is the one you set when creating the EMR cluster. Once your credentials are validated, you can enter the command line interface.
Run the following command in EMR command-line interface to switch to the Hadoop user:

```
[root@172 ~]# su hadoop
[hadoop@172 ~]$ flink  run  -m yarn-cluster -c com.tencent.flink.CosWordcount
./flink-example-1.0-SNAPSHOT.jar cosn://$BUCKET_NAME/test/data.txt
cosn://$BUCKET_NAME/test/result
[hadoop@172 ~]$ hdfs dfs -cat cosn://becklong-cos/test/result
(Flink,8)
(Hadoop,3)
(Spark,7)
(Hbase,3)
```

©2013-2024 Tencent Cloud. All rights reserved.