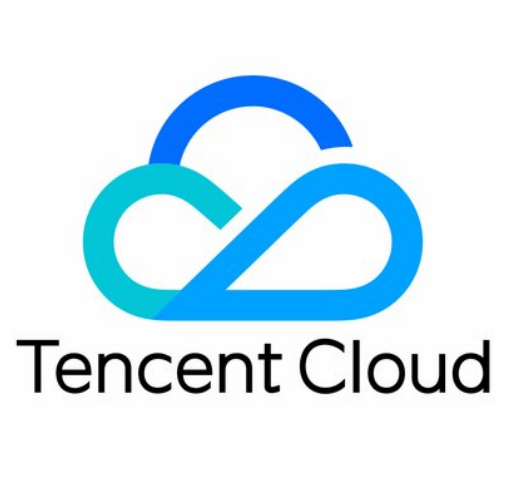
Cloud HDFS

Practical Tutorial

Product Documentation



Contents

Practical Tutorial
- Using CHDFS as Druid's Deep Storage
- Migrating Data from Native HDFS to CHDFS
- Importing or Exporting CHDFS with DataX
- Guide to Configuring CHDFS for CDH
- CHDFS Ranger Permission System Solution

Practical Tutorial
Using CHDFS as Druid's Deep Storage

Last updated：2022-03-30 09:30:26

Environment Dependencies

CHDFS_JAR.

Druid version: Druid 0.12.1

Download and Installation

Getting CHDFS JAR

Download CHDFS JAR.

Installing CHDFS JAR

Using CHDFS as Druid's deep storage requires `Druid-hdfs-extension` .
After downloading the CHDFS JAR, copy `chdfs_hadoop_plugin_network-1.7.jar` to the Druid installation path `extensions/druid-hdfs-storage` and `hadoop-dependencies/hadoop-client/2.x.x` .

Directions

Modifying configuration

1. Modify the `conf/druid/_common/common.runtime.properties` file at the Druid installation path, add the `extension` of HDFS to `druid.extensions.loadList` , specify HDFS as Druid's deep storage, and enter the path of the CHDFS instance:

```
properties
druid.extensions.loadList=["druid-hdfs-storage"]
druid.storage.type=hdfs
druid.storage.storageDirectory=ofs://<mountpoint>/<druid-path>
```

2. In the `conf/druid/_common/` directory, create the HDFS configuration file `hdfs-site.xml` and enter the configuration information of the CHDFS instance:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
```

```
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at
    http://www.apache.org/licenses/LICENSE-2.0
  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
 <property>
    <name>fs.AbstractFileSystem.ofs.impl</name>
    <value>com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter</value>
 </property>
 <property>
    <name>fs.ofs.impl</name>
    <value>com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter</value>
 </property>
 <!--Temporary directory of the local cache. For data read/write, data will be
written to the local disk when the memory cache is insufficient. This path will
be created automatically if it does not exist-->
 <property>
    <name>fs.ofs.tmp.cache.dir</name>
    <value>/data/chdfs_tmp_cache</value>
 </property>
 <!--You need to replace `appId` with your own `appid`, which can be obtained
at https://console.intl.cloud.tencent.com/cam/capi-->
 <property>
    <name>fs.ofs.user.appid</name>
    <value>125000001</value>
 </property>
</configuration>
```

The supported items of the above configuration are completely consistent with those as described at the CHDFS website. For more information, please see Mounting CHDFS Instance.

Starting to use

Start the Druid processes in turn, and Druid data can be loaded into CHDFS.

Migrating Data from Native HDFS to CHDFS

Last updated：2022-03-30 09:30:26

Preparations

1. Create a CHDFS instance and a CHDFS mount point and configure the permission information at the Tencent Cloud official website.
2. Access the created CHDFS instance from a CVM instance in a VPC. For more information, please see Creating CHDFS Instance.
3. After the mount is successful, open the Hadoop command line tool and run the following command to verify whether the CHDFS instance works properly.

```
hadoop fs -ls ofs://f4xxxxxxxxxxxxxx.chdfs.ap-beijing.myqcloud.com/
```

If you can see the output similar to the following, the CHDFS instance works properly.

Migration

After the preparations are completed, you can use the standard DistCp tool in the Hadoop community to perform full or incremental HDFS data migration. For more information, please see DistCp.

Notes

The Hadoop DistCp tool provides some parameters that are incompatible with CHDFS. If you specify some parameters in the following table, they will not take effect.

Parameter	Description	Status
-p[rbax]	r: replication; b: block-size; a: ACL, x: XATTR	Not effective

Samples

1. When the CHDFS instance is ready, run the following Hadoop command to perform data migration.

```
hadoop distcp hdfs://10.0.1.11:4007/testcp ofs://f4xxxxxxxx-xxxx.chdfs.ap-beijing.myqcloud.com/
```

Here, `f4xxxxxxxx-xxxx.chdfs.ap-beijing.myqcloud.com` is the mount point domain name, which needs to be replaced with the information of your actual mount point.

2. After the Hadoop command is executed, the details of the migration will be printed in the log as shown below:

```
2019-12-31 10:59:31 [INFO ] [main:13300] [org.apache.hadoop.mapreduce.Job:] [Job.ja
 Counters: 38
 File System Counters
     FILE: Number of bytes read=0
     FILE: Number of bytes written=387932
     FILE: Number of read operations=0
     FILE: Number of large read operations=0
     FILE: Number of write operations=0
     HDFS: Number of bytes read=1380
     HDFS: Number of bytes written=74
     HDFS: Number of read operations=21
     HDFS: Number of large read operations=0
     HDFS: Number of write operations=6
     OFS: Number of bytes read=0
     OFS: Number of bytes written=0
     OFS: Number of read operations=0
     OFS: Number of large read operations=0
     OFS: Number of write operations=0
 Job Counters
     Launched map tasks=3
     Other local map tasks=3
     Total time spent by all maps in occupied slots (ms)=419904
     Total time spent by all reduces in occupied slots (ms)=0
     Total time spent by all map tasks (ms)=6561
     Total vcore-milliseconds taken by all map tasks=6561
     Total megabyte-milliseconds taken by all map tasks=6718464
 Map-Reduce Framework
     Map input records=3
     Map output records=2
     Input split bytes=408
     Spilled Records=0
     Failed Shuffles=0
     Merged Map outputs=0
     GC time elapsed (ms)=179
     CPU time spent (ms)=4830
     Physical memory (bytes) snapshot=1051619328
```

```
        Virtual memory (bytes) snapshot=12525191168
        Total committed heap usage (bytes)=1383071744
    File Input Format Counters
        Bytes Read=972
    File Output Format Counters
        Bytes Written=74
    org.apache.hadoop.tools.mapred.CopyMapper$Counter
        BYTESSKIPPED=5
        COPY=1
        SKIP=2
```

Importing or Exporting CHDFS with DataX

Last updated：2022-03-30 09:30:26

Environment Dependencies

CHDFS_JAR.
DataX version: DataX 3.0

Download and Installation

Getting CHDFS JAR

Download CHDFS JAR.

Getting DataX package

Download DataX.

Installing CHDFS JAR

After downloading the CHDFS JAR, copy `chdfs_hadoop_plugin_network-1.7.jar` to the DataX decompression path `plugin/reader/hdfsreader/libs/` and `plugin/writer/hdfswriter/libs/` .

Directions

Configuring DataX

Modifying `datax.py` script

Open the `bin/datax.py` script in the DataX decompression directory and modify the `CLASS_PATH` variable in it as follows:

```
CLASS_PATH = ("%s/lib/*:%s/plugin/reader/hdfsreader/libs/*:%s/plugin/writer/hdfswri
```

Configuring `hdfsreader` and `hdfswriter` in JSON configuration file

Below is a JSON sample:

```
{
    "job": {
        "setting": {
```

```
            "speed": {
                "byte": 10485760
            },
            "errorLimit": {
                "record": 0,
                "percentage": 0.02
            }
        },
        "content": [{
            "reader": {
                "name": "hdfsreader",
                "parameter": {
                    "path": "testfile",
                    "defaultFS": "ofs://f4xxxxxxxxx-hxT9.chdfs.ap-beijing.myqcloud.
                    "column": ["*"],
                    "fileType": "text",
                    "encoding": "UTF-8",
                    "hadoopConfig": {
                        "fs.AbstractFileSystem.ofs.impl": "com.qcloud.chdfs.fs.CHDF
                        "fs.ofs.impl": "com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAd
                        "fs.ofs.tmp.cache.dir": "/data/chdfs_tmp_cache",
                        "fs.ofs.user.appid": "1250000000"
                    },
                    "fieldDelimiter": ","
                }
            },
            "writer": {
                "name": "hdfswriter",
                "parameter": {
                    "path": "/user/hadoop/",
                    "fileName": "testfile1",
                    "defaultFS": "ofs://f4xxxxxxxxx-hxT9.chdfs.ap-beijing.myqcloud.
                    "column": [{
                            "name": "col",
                            "type": "string"
                        },
                        {
                            "name": "col1",
                            "type": "string"
                        },
                        {
                            "name": "col2",
                            "type": "string"
                        }
                    ],
                    "fileType": "text",
                    "encoding": "UTF-8",
```

```
                    "hadoopConfig": {
                        "fs.AbstractFileSystem.ofs.impl": "com.qcloud.chdfs.fs.CHDF
                        "fs.ofs.impl": "com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAd
                        "fs.ofs.tmp.cache.dir": "/data/chdfs_tmp_cache",
                        "fs.ofs.user.appid": "1250000000"
                    },
                    "fieldDelimiter": ":",
                    "writeMode": "append"
                }
            }
        }]
    }
}
```

Here, configure `hadoopConfig` as the configuration required by the CHDFS instance and enter the path of the CHDFS instance as `defaultFS`, such as `ofs://f4xxxxxxxx-hxT9.chdfs.ap-beijing.myqcloud.com/`. Other configuration items are the same as the HDFS configuration items.

Performing data migration

Save the configuration file as `hdfs_job.json`, place it in the `job` directory, and run the following command:

```
bin/datax.py job/hdfs_job.json
```

You can see that the output is as follows:

```
2020-03-09 16:49:59.543 [job-0] INFO  JobContainer -
         [total cpu info] =>
                averageCpu                     | maxDeltaCpu                    | m
                -1.00%                         | -1.00%                         | -


         [total gc info] =>
                 NAME                 | totalGCCount       | maxDeltaGCCount    | m
                 PS MarkSweep         | 1                  | 1                  | 1
                 PS Scavenge          | 1                  | 1                  | 1

2020-03-09 16:49:59.543 [job-0] INFO  JobContainer - PerfTrace not enable!
2020-03-09 16:49:59.543 [job-0] INFO  StandAloneJobContainerCommunicator - Total 2
2020-03-09 16:49:59.544 [job-0] INFO  JobContainer -
Task start time                    : 2020-03-09 16:49:48
Task end time                      : 2020-03-09 16:49:59
Total task duration                :                 11s
Average task traffic               :                 3 B/s
Record write speed                 :              0 rec/s
Read records                       :                    2
Failed reads/writes                :                    0
```

Guide to Configuring CHDFS for CDH

Last updated：2022-03-30 09:30:26

Overview

Cloudera's Distribution Including Apache Hadoop (CDH) is a popular Hadoop distribution. This document describes how to use CHDFS on CDH to separate big data computing and storage and provide a flexible big data solution at low costs.

CHDFS supports the following big data components:

Component	Supported by CHDFS	Service Component Restart Required
YARN	Yes	NodeManager
YARN	Yes	NodeManager
Hive	Yes	HiveServer and HiveMetastore
Spark	Yes	NodeManager
Sqoop	Yes	NodeManager
Presto	Yes	HiveServer, HiveMetastore, and Presto
Flink	Yes	No
Impala	Yes	No
EMR	Yes	No
Self-built component	To be supported in the future	No
HBase	Not recommended	No

Version Dependency

The dependent component versions used in this document are as follows:

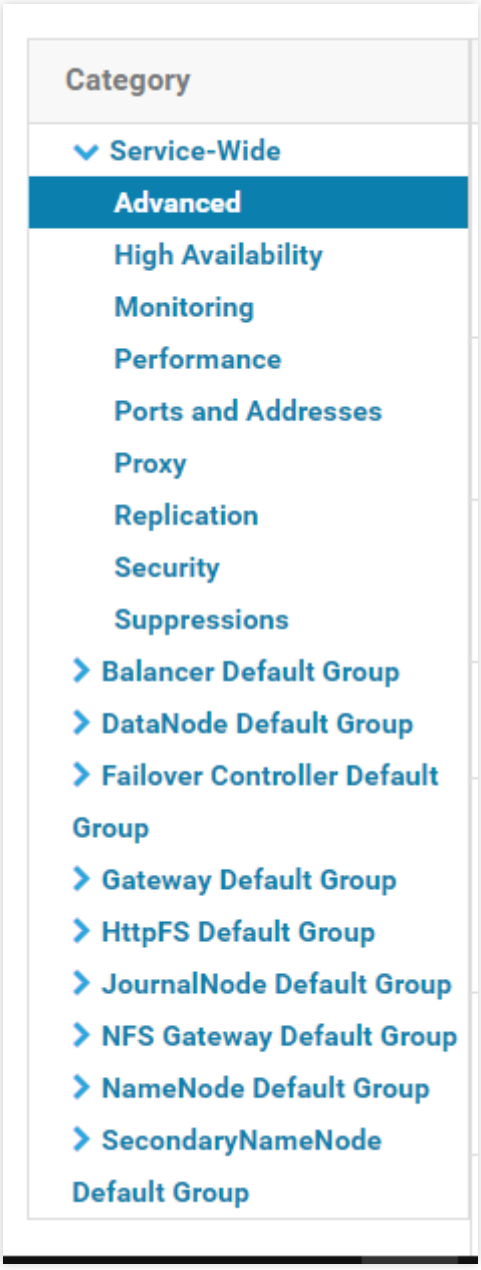
CDH 5.16.1

Hadoop 2.6.0

Directions

Storage environment configuration

1. Log in to Cloudera Manager (CDH management page).
2. On the system homepage, select **Configuration** > **Service-Wide** > **Advanced** to enter the advanced configuration code snippet page as shown below:



3. Enter the CHDFS configuration in the code box of `Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml`.

```
<property>
```

```
<name>fs.AbstractFileSystem.ofs.impl</name>
<value>com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter</value>
</property>
<property>
<name>fs.ofs.impl</name>
<value>com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter</value>
</property>
<!--Temporary directory of the local cache. For data read/write, data will be
written to the local disk when the memory cache is insufficient. This path will
be created automatically if it does not exist-->
<property>
<name>fs.ofs.tmp.cache.dir</name>
<value>/data/emr/hdfs/tmp/chdfs/</value>
</property>
<!--appId-->
<property>
<name>fs.ofs.user.appid</name>
<value>1250000000</value>
</property>
```

The following are the required CHDFS configuration items (which need to be added to `core-site.xml`). For other CHDFS configuration items, please see Mounting CHDFS Instance.

CHDFS Configuration Item	Value	Description
fs.ofs.user.appid	1250000000	User `appid`
fs.ofs.tmp.cache.dir	/data/emr/hdfs/tmp/chdfs/	Temporary directory
fs.ofs.impl	com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter	CHDFS implementat `com.qcloud.chd`
fs.AbstractFileSystem.ofs.impl	com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter	CHDFS implementat is fixed at `com.qcloud.chd`

4. Click the HDFS service to deploy the client configuration, and the above `core-site.xml` configuration will be updated to the servers in the cluster.
5. Place the latest CHDFS SDK package under the JAR package path of the CDH HDFS service as shown below, where the path should be replaced with the actual value:

```
cp chdfs_hadoop_plugin_network-2.0.jar /opt/cloudera/parcels/CDH-5.16.1-
1.cdh5.16.1.p0.3/lib/hadoop-hdfs/
```

Note:

You need to place the SDK package at the same location on each server in the cluster.

Data migration

Migrate the data from CDH HDFS to CHDFS with the Hadoop DistCp tool. For more information, please see Migrating Data from Native HDFS to CHDFS.

Using CHDFS for big data suites

1. MapReduce

Directions

(1) Configure HDFS as instructed in Data migration and place the CHDFS SDK JAR package in the corresponding directory of HDFS.

(2) On the CDH system homepage, find YARN and restart NodeManager (you don't need to restart NodeManager for TeraGen but need to do so for TeraSort due to internal business logic; therefore, we recommend you restart NodeManager in both cases).

Sample

The following uses TeraGen and TeraSort in a standard Hadoop test as an example:

```
hadoop jar ./hadoop-mapreduce-examples-2.7.3.jar teragen -Dmapred.map.tasks=4
1099 cosn://examplebucket-1250000000/teragen_5/

hadoop jar ./hadoop-mapreduce-examples-2.7.3.jar terasort  -Dmapred.map.tasks=4
cosn://examplebucket-1250000000/teragen_5/ cosn://examplebucket-
1250000000/result14
```

Note:

Please replace the part after `ofs://  schema` with the mount point path of your CHDFS instance.

2. Hive

2.1 MR engine

Directions

(1) Configure HDFS as instructed in Data migration and place the CHDFS SDK JAR package in the corresponding directory of HDFS.

(2) On the CDH homepage, find Hive and restart the HiveServer2 and HiveMetastore roles.

Sample

For a user's real business query, for example, run the `Hive` command line to create a location as a sharded table on CHDFS:

```
CREATE TABLE `report.report_o2o_pid_credit_detail_grant_daily`(
  `cal_dt` string,
  `change_time` string,
  `merchant_id` bigint,
  `store_id` bigint,
  `store_name` string,
  `wid` string,
  `member_id` bigint,
  `meber_card` string,
  `nickname` string,
  `name` string,
  `gender` string,
  `birthday` string,
  `city` string,
  `mobile` string,
  `credit_grant` bigint,
  `change_reason` string,
  `available_point` bigint,
  `date_time` string,
  `channel_type` bigint,
  `point_flow_id` bigint)
PARTITIONED BY (
  `topicdate` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
STORED AS INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'
    OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat'
LOCATION
  'cosn://examplebucket-1250000000/user/hive/warehouse/report.db/report_o2o_pid_cre
TBLPROPERTIES (
  'last_modified_by'='work',
  'last_modified_time'='1589310646',
  'transient_lastDdlTime'='1589310646')
```

Run the following SQL query:

```
select count(1) from report.report_o2o_pid_credit_detail_grant_daily;
```

You will see the following results:

```
hive> select count(1) from report.report_o2o_pid_credit_detail_grant_daily;
Query ID = root_20200713121818_3a911a0c-2496-4e7e-b59d-b2a26266a6ab
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1594351711155_0014, Tracking URL = http://hadoop01:8088/prox
y/application_1594351711155_0014/
Kill Command = /opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/lib/hadoop/bin
/hadoop job  -kill job_1594351711155_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-07-13 12:18:19,189 Stage-1 map = 0%,  reduce = 0%
2020-07-13 12:18:25,391 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 8.89 s
ec
2020-07-13 12:18:30,544 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.8
 sec
MapReduce Total cumulative CPU time: 10 seconds 800 msec
Ended Job = job_1594351711155_0014
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.8 sec   HDFS Read: 17383
HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 800 msec
OK
27677
Time taken: 19.128 seconds, Fetched: 1 row(s)
```

2.2 Tez engine

For the Tez engine, you need to import the CHDFS JAR package into the compressed package of Tez. The following takes `apache-tez.0.8.5` as an example for description:

Directions

(1) Find the Tez package installed on the CDH cluster and decompress it, such as `/usr/local/service/tez/tez-0.8.5.tar.gz` .

(2) Place the CHDFS JAR package in the decompressed directory, recompress it, and output a compressed package.

(3) Upload the new compressed package to the path specified by `tez.lib.uris` (if it already exists, directly replace it).

(4) On the CDH homepage, find Hive and restart HiveServer and HiveMetastore.

3. Spark

Directions

(1) Configure HDFS as instructed in Data migration and place the CHDFS SDK JAR package in the corresponding directory of HDFS.

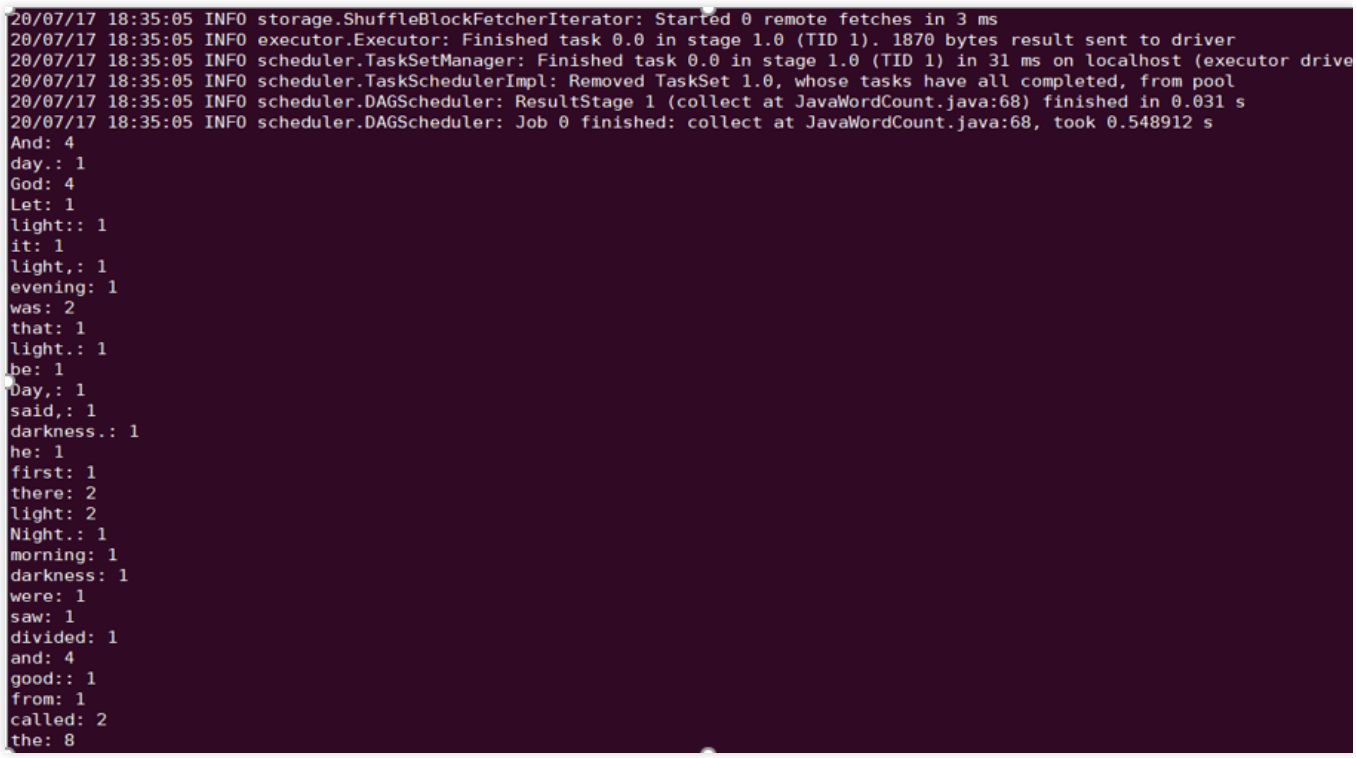
(2) Restart NodeManager.

Sample

The following takes the `Spark example word count` test conducted with CHDFS as an example.

```
spark-submit  --class org.apache.spark.examples.JavaWordCount --executor-memory
4g --executor-cores 4  ./spark-examples-1.6.0-cdh5.16.1-hadoop2.6.0-
cdh5.16.1.jar cosn://examplebucket-1250000000/wordcount
```

The execution result is as follows:



4. Sqoop

Directions

(1) Configure HDFS as instructed in Data migration and place the CHDFS SDK JAR package in the corresponding directory of HDFS.

(2) Place the CHDFS SDK JAR package in the `sqoop` directory (such as `/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/lib/sqoop/`).

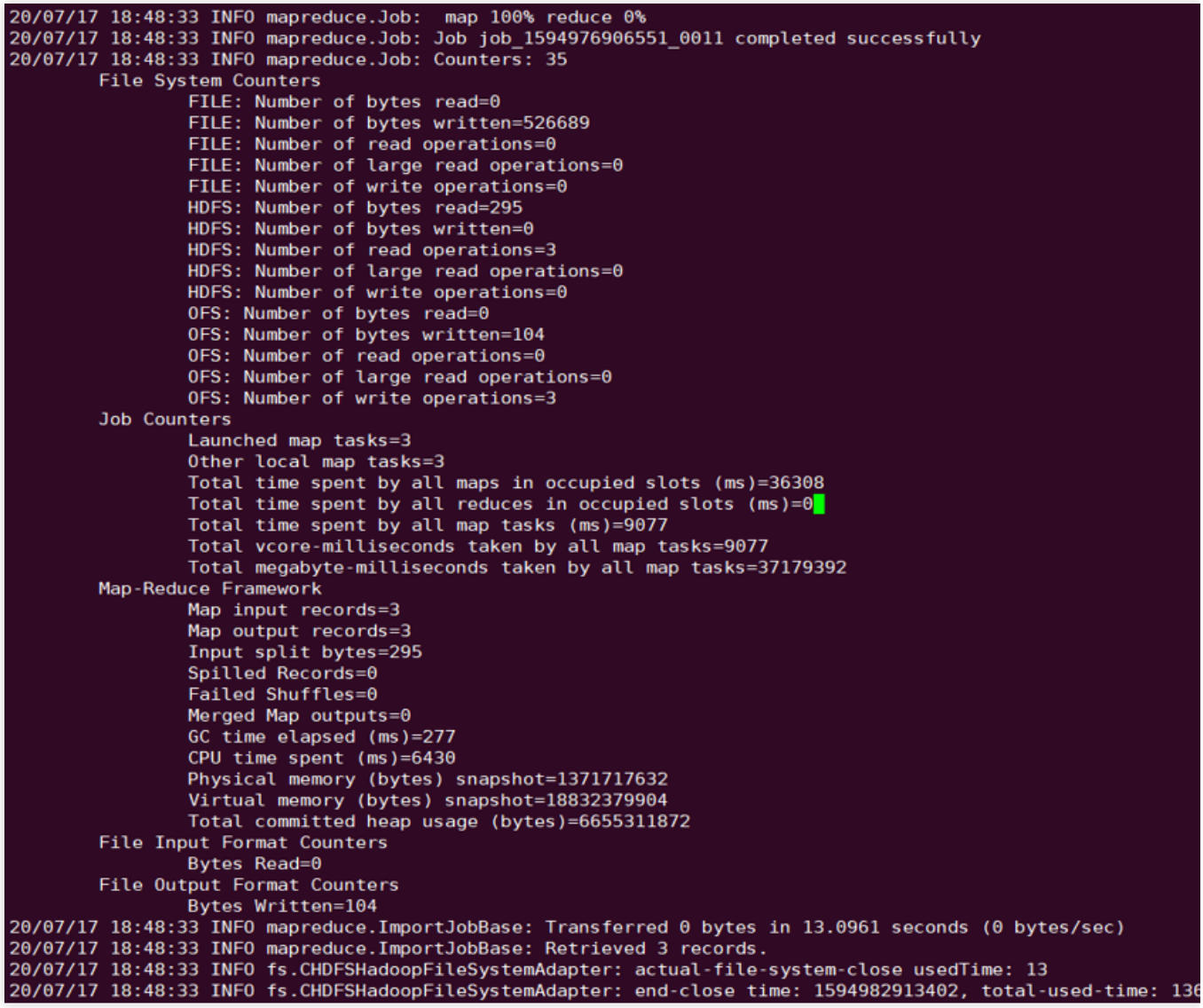
(3) Restart NodeManager.

Sample

The test is performed by taking exporting MySQL tables to CHDFS as instructed in Import/Export of Relational Database and HDFS as an example.

```
sqoop import --connect "jdbc:mysql://IP:PORT/mysql" --table sqoop_test --
username root --password 123  --target-dir cosn://examplebucket-
1250000000/sqoop_test
```

The execution result is as follows:



5. Presto

Directions

(1) Configure HDFS as instructed in Data migration and place the CHDFS SDK JAR package in the corresponding directory of HDFS.

(2) Place the CHDFS SDK JAR package in the `presto` directory (such as `/usr/local/services/cos_presto/plugin/hive-hadoop2`).

(3) As Presto doesn't load `gson-2.*.*.jar` under `hadoop common`, you also need to place `gson-2.*.*.jar` in the `presto` directory (such as `/usr/local/services/cos_presto/plugin/hive-hadoop2`. Only CHDFS depends on Gson).

(4) Restart HiveServer, HiveMetaStore, and Presto.

Sample

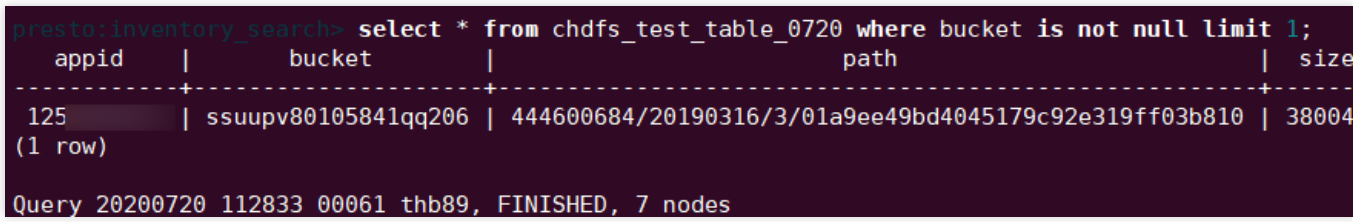
The following takes creating a location with Hive as a CHDFS table query as an example:

```
select * from chdfs_test_table where bucket is not null limit 1;
```

Note:

`chdfs_test_table` is the table whose location is `ofs scheme` .

The query result is as follows:



CHDFS Ranger Permission System Solution

Last updated：2022-03-30 09:30:26

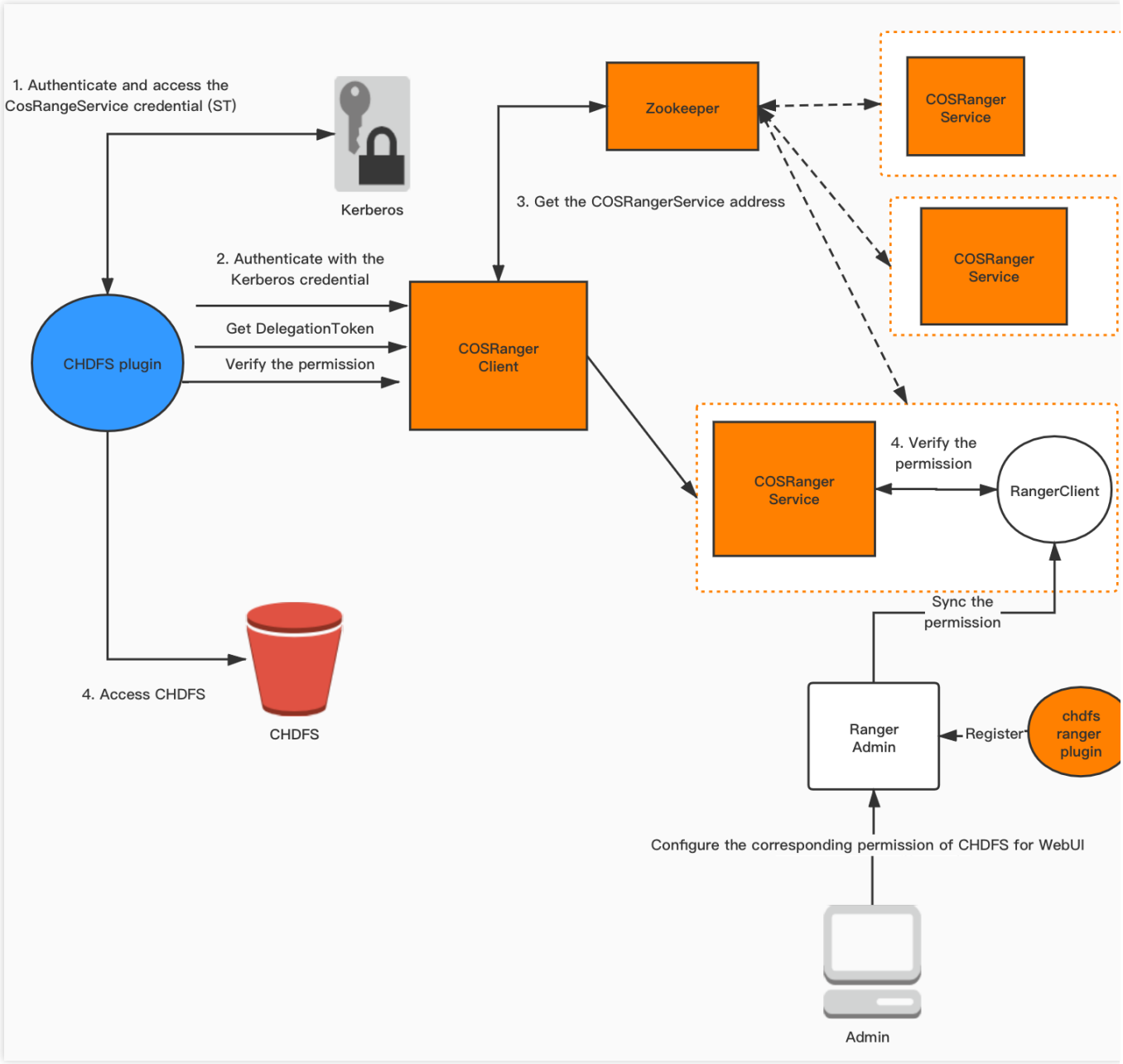
Background

After you adopt storage-compute separation, you will host your data on CHDFS. CHDFS provides a permission system similar to that of native HDFS. In addition to HDFS permissions, Hadoop Ranger implements more refined permission control, including user group permission settings and permission settings for specific prefixes. Plus, as a one-stop permission system solution, Hadoop Ranger supports permission control for not only storage services but also components such as YARN and Hive. Therefore, to suit your use habits, we provide a CHDFS-Ranger connection solution to make it easy for you to control the permissions of CHDFS with Ranger.

Strengths

Fine-grained permission control, which suits the use habits with Hadoop permissions.
Support for unified management of big data component and cloud-hosted storage permissions.

Solution Architecture



In the Hadoop permission system, authentication is provided by Kerberos and authorized by Ranger. In addition to this, we provide the following components to support the Ranger permission scheme for CHDFS.

CHDFS-Ranger-Plugin: it provides a service definition plugin on the Ranger server and description of the CHDFS service on the Ranger side. After this plugin is deployed, you can enter the corresponding permission policy on the Ranger control page.

COSRangerService: it integrates the Ranger client, periodically syncs permission policies from the Ranger server, and verifies the permission locally after receiving an authentication request. In addition, it also provides `DelegationToken` generation and renewal APIs in Hadoop, all of which are defined through Hadoop IPC.

CosRangerClient: it is dynamically loaded by the COSN plugin to forward permission verification requests to `CosRangerService` .

Environment Deployment

Hadoop environment

ZooKeeper, Ranger, and Kerberos (if there are authentication requirements)

Note:

As the above services are mature open-source components, you can install them on your own.

Component Deployment

Deploy CHDFS-Ranger-Plugin

Deploy COSRangerService

Deploy COSRangerClient

Deploy CHDFS

`CHDFS-Ranger-Plugin` extends the service types in the Ranger Admin console. You can set the operation permissions related to CHDFS in the Ranger console.

Code address

You can get the code from the `ranger-plugin` directory at GitHub.

Version

v1.2 or above

Deployment steps

1. Create a `COS` directory in the service definition directory of Ranger (note: make sure that the directory permissions include at least `x` and `r` permissions).

1. For an EMR environment, the path is `ranger/ews/webapp/WEB-INF/classes/ranger-plugins` .

2. For a self-built Hadoop environment, you can find the components connected to the Ranger service through `find hdfs` in the `ranger` directory in order to find the location of the `ranger-plugins` directory.

```
[hadoop@10 ranger-plugins]$ ll
total 68
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 atlas
drwxr-xr-x 2 hadoop hadoop 4096 Dec 15 10:57 chdfs
drwxr-xr-x 2 hadoop hadoop 4096 Dec 15 10:57 cos
drwxr-xr-x 2 hadoop hadoop 4096 Feb 25  2020 hbase
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 hdfs
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 hive
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 kafka
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 kms
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 knox
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 kylin
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 nifi
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 nifi-registry
drwxr-xr-x 2 hadoop hadoop 4096 Aug  5  2020 presto
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 solr
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 sqoop
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 storm
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 yarn
```

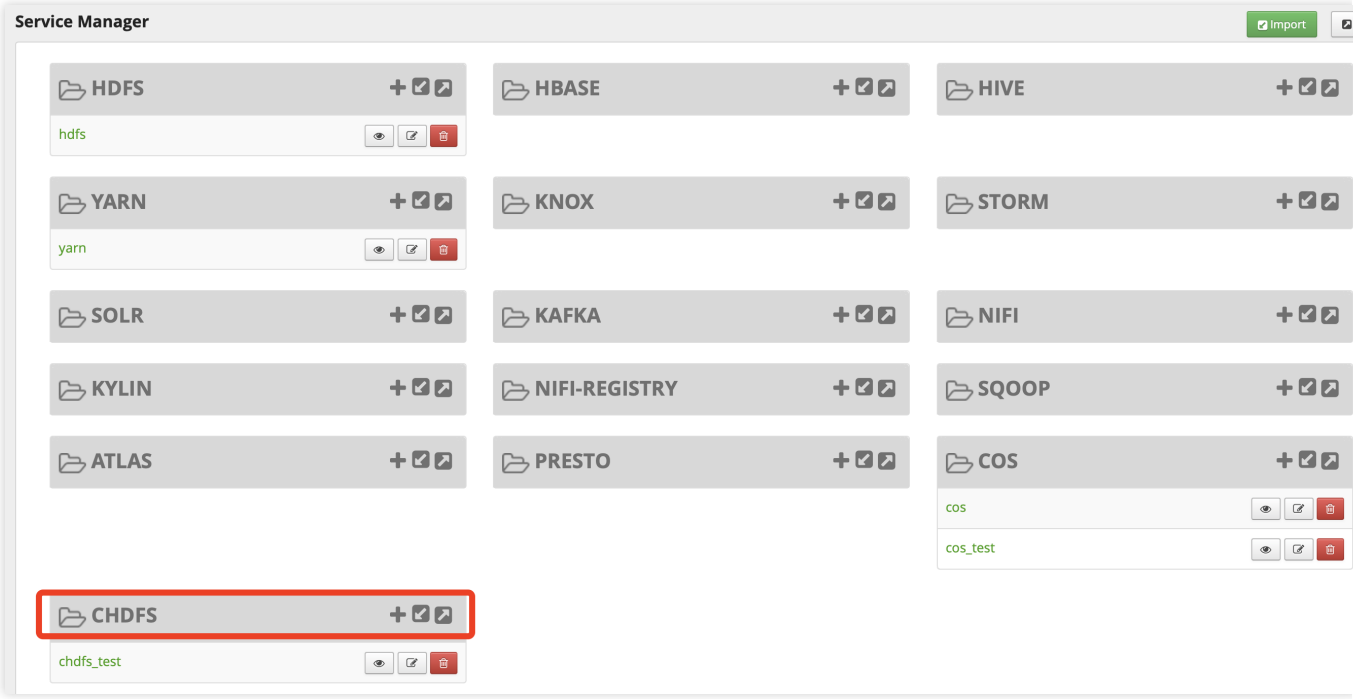
3. Place `cos-chdfs-ranger-plugin-xxx.jar` in the CHDFS directory (note: the JAR package should at least have the `r` permission).

4. Restart Ranger.

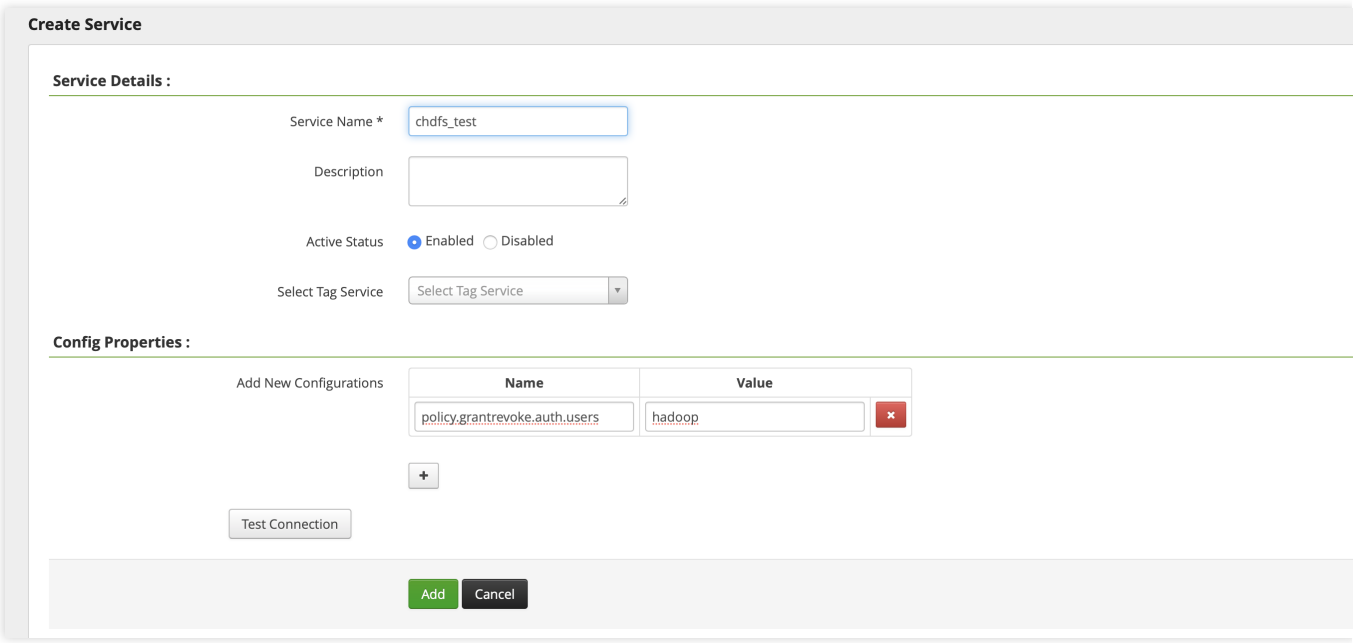
5. Register the CHDFS service on Ranger. You can refer to the following command:

```
## To generate a service, you need to pass in the password of the Ranger admin account and the address of the Ranger service.
## For an EMR cluster, the admin user is `root`, and the password is the root password set when the EMR cluster is created. Replace the IP of the Ranger service with the master node IP of EMR.
adminUser=root
adminPasswd=xxxxxx
rangerServerAddr=10.0.0.1:6080
curl -v -u${adminUser}:${adminPasswd} -X POST -H "Accept:application/json" -H "Content-Type:application/json" -d @./chdfs-ranger.json http://${rangerServerAddr}/service/plugins/definitions
## To delete the service just defined, you need to pass in the service ID returned during creation.
serviceId=102
curl -v -u${adminUser}:${adminPasswd} -X DELETE -H "Accept:application/json" -H "Content-Type:application/json" http://${rangerServerAddr}/service/plugins/definitions/${serviceId}
```

6. After the service is successfully created, you can see the CHDFS service in the Ranger console as shown below:

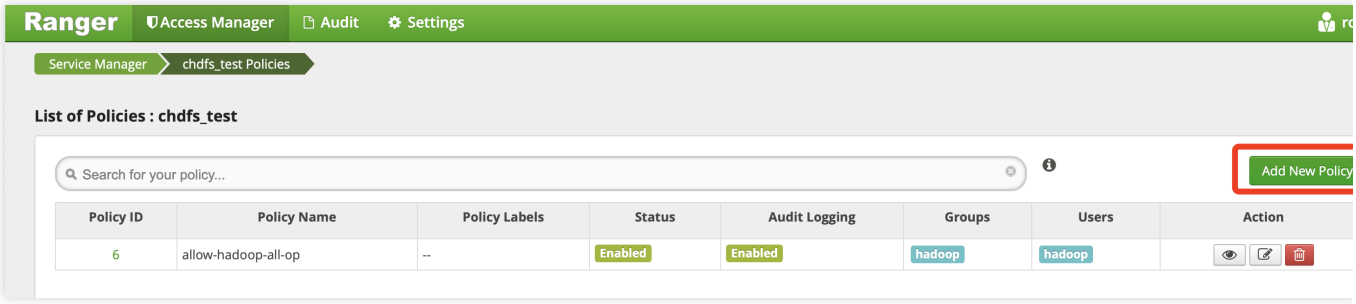


7. Click + next to the CHDFS service to define a new service instance. The service instance name is customizable; for example, you can enter `chdfs` or `chdfs_test`. The service configuration is as shown below:

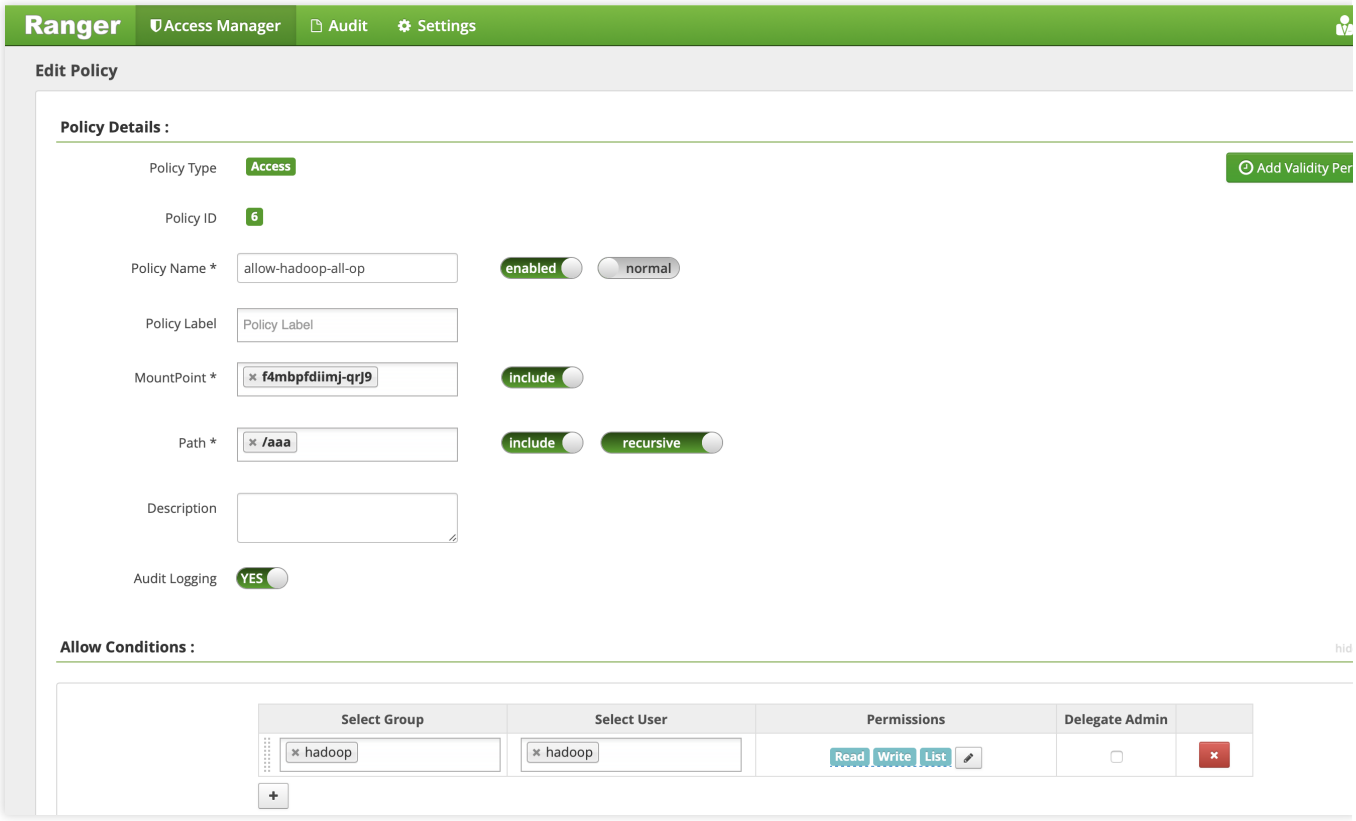


You need to set the username subsequently used to start the `COSRangerService` service for `policy.grantrevoke.auth.users`. We generally recommend you set it to `hadoop`.

8. Click the generated CHDFS service instance to add a policy as shown below:



9. On the displayed page, configure the following parameters as detailed below:



MountPoint: mount point name in the format of `f4mxxxxxx-yyyy`. You can log in to the CHDFS console to view it.

Path: path of CHDFS, which must start with `/`.

include: indicates whether the set permission applies to the specified path itself or other paths except it.

recursive: indicates that the permission applies to not only the specified path but also the subpaths under it (i.e., recursive subpaths). It is usually used when the path is set as a directory.

Select Group/**Select User**: username and user group in logical OR relationship; that is, the operation is authorized as long as the username or user group condition is met.

Permissions:

Read: read operation, which corresponds to the GET and HEAD operations in COS, such as downloading objects and querying object metadata.
Write: write operation, which corresponds to the PUT operation in COS, such as uploading objects.
Delete: delete operation, which corresponds to the object deletion operation in COS. To rename a path in Hadoop, you need to have the deletion permission for the original path and write permission for the new path.
List: traversal permission, which corresponds to the `List Object` operation in COS.
COSRangerService is the core of the entire permission system. It is responsible for integrating the Ranger client and receiving its authentication requests, token generation and renewal requests, and temporary key generation requests. It is also where sensitive information (Tencent Cloud key information) resides. Generally, it is deployed on a bastion host, and only the cluster admin is allowed to manipulate it and view its configuration.
COSRangerService supports one-primary-multiple-secondary HA deployment. The `DelegationToken` status is persistently stored in HDFS. The leader role is determined by ZooKeeper lock grabbing. The service that gets the leader role will write its address to ZooKeeper, so that COSRangerClient can perform address routing.

Code address

You can get the code from the `cos-ranger-server` directory at GitHub.

Version

v1.2 or above

Deployment steps

1. Copy the code of COSRangerService to the servers in the cluster. We recommend you configure at least two servers (one primary server and one secondary server) in the production environment. As sensitive information is involved, we recommend you use bastion hosts or servers with tight permission control.
2. Modify the relevant configuration items in the `cos-ranger.xml` file. Those that must be modified are as follows. For the description of the configuration items, please see the comments in the file.
qcloud.object.storage.rpc.address
qcloud.object.storage.status.port
qcloud.object.storage.enable.chdfs.ranger
qcloud.object.storage.zk.address
3. Modify the relevant configuration items in the `ranger-chdfs-security.xml` file. Those that must be modified are as follows. For the description of the configuration items, please see the comments in the file.
ranger.plugin.chdfs.policy.cache.dir
ranger.plugin.chdfs.policy.rest.url
ranger.plugin.chdfs.service.name
4. Modify the `hadoop_conf_path` and `java.library.path` configuration items in `start_rpc_server.sh`, which point to the directory of the Hadoop configuration file (such as `core-site.xml` and `hdfs-site.xml`) and the `hadoop native lib` path, respectively.

5. Run the following command to start the service.

```
chmod +x start_rpc_server.sh
nohup ./start_rpc_server.sh &> nohup.txt &
```

6. If the start fails, check whether the error log contains an error message.

7. COSRangerService supports displaying the HTTP port status (the port name is `qcloud.object.storage.status.port`, which is 9998 by default). You can run the following command to get the status information (such as whether the leader is contained and the authentication statistics).

```
# Replace `10.xx.xx.xxx` below with the IP of the server where the Ranger
service is deployed
# Replace 9998 with the value of `qcloud.object.storage.status.port`
curl -v http://10.xx.xx.xxx:9998/status
```

COSRangerClient is dynamically loaded by the Hadoop CHDFS plugin and proxies all COSRangerService access requests, such as token acquisition and authentication.

Code address

You can get the code from the `cos-ranger-client` directory at GitHub.

Version

v1.0 or above

Deployment method

1. Copy the `cos-ranger-client` JAR package to the same directory as the CHDFS plugin (please choose to copy the JAR package consistent with the major version of your Hadoop).
2. Add the following configuration items in `core-site.xml`:

```
```
<configuration>
          <!--*****Required********-->
          <!-- ZooKeeper address. The client finds the address of the Ranger
service by ZooKeeper query -->
          <property>
              <name>qcloud.object.storage.zk.address</name>
              <value>10.0.0.8:2121</value>
          </property>

          <!--***Optional****-->
          <!-- Set the Kerberos credentials used by the COSRangerService.
Please refer to the configuration of the COSRangerService, which must be
consistent. If there are no authentication needs, you can skip this
configuration -->
```
```

```
            <property>
                      <name>qcloud.object.storage.kerberos.principal</name>
                      <value>hadoop/_HOST@EMR-XXXX</value>
            </property>

          <!--***Optional****-->
          <!-- Ranger server IP path recorded on ZooKeeper. The default value is
used here. The configuration must be consistent with that of COSRangerService -
->
           <property>
                     <name>qcloud.object.storage.zk.leader.ip.path</name>
                     <value>/ranger_qcloud_object_storage_leader_ip</value>
           </property>
</configuration>
```

Version

v2.1 or above

Deployment method

For the method of deploying the CHDFS plugin, please see Mounting CHDFS Instance. Open Ranger and adding the following:

```
    <property>
        <name>fs.ofs.ranger.enable.flag</name>
        <value>true</value>
    </property>
```

Verification

1. Use `hadoop cmd` to perform operations related to accessing CHDFS as shown below:

```
# Replace the mount point, path, and other parameters with your actual information.
hadoop fs -ls ofs://f4mxxxxyyyy-zzzz.chdfs.ap-guangzhou.myqcloud.com/doc
hadoop fs -put ./xxx.txt ofs://f4mxxxxyyyy-zzzz.chdfs.ap-guangzhou.myqcloud.com/doc
hadoop fs -get ofs://f4mxxxxyyyy-zzzz.chdfs.ap-guangzhou.myqcloud.com/exampleobject
hadoop fs -rm ofs://f4mxxxxyyyy-zzzz.chdfs.ap-guangzhou.myqcloud.com/exampleobject.
```

2. Use an MR job to verify. Relevant services such as YARN and Hive must be restarted before verification.

FAQs

Do I have to install Kerberos?

If users in the cluster are trustworthy, such as in a cluster that is only used internally, you can install Kerberos to support authentication. If the users only perform authentication operations, in order to avoid maloperations by unauthorized clients, you can choose not to install Kerberos and only use Ranger for authentication.
Installing Kerberos will incur some performance loss. Please consider your own security and performance requirements. If authentication is required, after enabling Kerberos, you need to set the related configuration items of COSRangerService and COSRangerClient.

If Ranger is enabled, but no policy is configured, or no policy is matched, what will happen?

If no policy is matched, the operation will be denied by default.

After Ranger is enabled, will CHDFS still perform POSIX authentication?

Ranger authentication is performed in the client environment. Requests authenticated by Ranger will be sent to the CHDFS server, which will perform POSIX authentication by default. Therefore, if the permissions are controlled by Ranger, please disable the POSIX permission in the CHDFS console.