Cloud HDFS

プラクティスチュートリアル

製品ドキュメント

著作権声明

商標に関する声明

Tencent Cloud

およびその他のTencent Cloudサービスに関連する商標は、すべてTencentグループ下の関連会社主体により所有しています。また、本ドキュメントに記載されている第三者主体の商標は、法に基づき権利者により所有しています。

サービス声明

本ドキュメントは、お客様にTencent Cloudの全部また一部の製品・サービスの概要をご紹介することを目的としておりますが、一部の製品・サービス内容は変更される可能性があります。お客様がご購入されるTencent Cloud製品・サービスの種類やサービス基準などは、お客様とTencent Cloudとの間の締結された商業契約に基づきます。別段の合意がない限り、Tencent Cloudは本ドキュメントの内容に関して、明示または黙示の一切保証もしません。

カタログ:

プラクティスチュートリアル DruidのDeep storageとしてCHDFSを使用します

最終更新日：：2024-01-19 16:58:09

環境の依存

- CHDFS_JAR。
- Druidバージョン：Druid-0.12.1。

ダウンロードとインストール

CHDFS JARの取得

公式GithubでCHDFS_JARをダウンロードします。

CHDFS JARのインストール

DruidのDeep StorageとしてCHDFSを使用するには、Druid-hdfs-extensionによって実現する必要があります。CHDFS JARをダウンロードした後、`chdfs_hadoop_plugin_network-1.7.jar`をDruidインストールパス`extensions/druid-hdfs-storage`および`hadoop-dependencies/hadoop-client/2.x.x`にコピーします。

使用方法

設定の変更

1. Druidインストールパスの`conf/druid/_common/common.runtime.properties`ファイルを変更し、hdfsのextensionを`druid.extensions.loadList`に追加すると同時に、hdfsをDruidのdeep storageとして指定します。そして、パスはCHDFSのパスとして入力します。

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

2. このディレクトリ`conf/druid/_common/`下に、hdfsの設定ファイルhdfs-site.xmlを新規作成し、CHDFSの設定情報などを入力します。

```
<?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>
 <!--ローカルcacheの一時ディレクトリ。データの読み取りと書き込みの場合、メモリcache
が不足すると、ローカルディスクに書き込まれます。このパスが存在しない場合は、自動的に作
成されます-->
 <property>
    <name>fs.ofs.tmp.cache.dir</name>
    <value>/data/chdfs_tmp_cache</value>
 </property>
 <!--appIdユーザーは、ご自身のappidに変更する必要があります。
https://console.intl.cloud.tencent.com/cam/capiから取得できます-->
 <property>
    <name>fs.ofs.user.appid</name>
    <value>125000001</value>
 </property>
</configuration>
```

上記の設定のサポート項目は、CHDFSの公式ウェブサイトの説明と全く同じです。詳細については、CHDFSのマウント ドキュメントをご参照ください。

使用の開始

Druidプロセスを順次起動すると、DruidデータをCHDFSにロードできます。

ネイティブHDFSデータをTencent Cloud CHDFSに移行します

最終更新日：：2024-01-19 16:58:09

準備作業

1. Tencent Cloudの公式ウェブサイトでCHDFSファイルシステムとCHDFSマウントポイントを作成し、アクセス権限情報を設定します。
2. Tencent Cloud VPC環境のCVMマシンを介して、作成済みのCHDFSにアクセスします。詳細については、CHDFSの作成 をご参照ください。
3. マウントが成功したら、hadoopコマンドラインツールを開き、以下のコマンドを実行して、CHDFS検証機能が正常かどうかを検証します。

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

以下のような出力が表示される場合は、Cloud HDFS機能のすべてが正常であることを意味します。

移行

準備ができたら、hadoopコミュニティ標準のDistcpツールを使用して、完全または増分のHDFSデータの移行を行うことができます。詳細については、Distcp公式ガイドラインドキュメント をご参照ください。

注意事項

hadoop distcpツールには、CHDFSと互換性のないパラメータがいくつか提供されています。次の表のパラメータのうちいくつかは、指定しても有効になりません。

パラメータ	説明	状態
-p[rbax]	r：replication，b：block-size，a：ACL，x：XATTR	無効

事例の説明

1. CHDFSの準備ができたら、以下のhadoopコマンドを実行してデータを移行します。

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

そのうち `f4xxxxxxxx-xxxx.chdfs.ap-beijing.myqcloud.com` はマウントポイントのドメイン名であり、実際に申請したマウントポイント情報に基づいて置き換える必要があります。

2. Hadoopコマンドが実行された後、移行の具体的な詳細がログに印刷されます。以下に例を示します。

```
2019-12-31 10:59:31 [INFO ] [main:13300]
[org.apache.hadoop.mapreduce.Job:] [Job.java:1385]
 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
```

DataXを使用してCHDFSをインポートまたはエクスポートします

最終更新日：：2025-09-26 10:17:22

環境の依存

- CHDFS_JAR。
- DataXバージョン：DataX-3.0。

ダウンロードとインストール

CHDFS JARの取得

公式GithubでCHDFS_JARをダウンロードします。

DataXソフトウェアパッケージの取得

公式GithubでDataXをダウンロードします。

CHDFS JARのインストール

CHDFS JARをダウンロードした後、`chdfs_hadoop_plugin_network-1.7.jar`をDatax解凍パス`plugin/reader/hdfsreader/libs/`および`plugin/writer/hdfswriter/libs/`にコピーします。

使用方法

DataXの設定

datax.pyスクリプトの変更

DataX解凍ディレクトリ下のbin/datax.pyスクリプトを開き、スクリプトのCLASS_PATH変数を次のように変更します。

```
CLASS_PATH =
("%s/lib/*:%s/plugin/reader/hdfsreader/libs/*:%s/plugin/writer/hdfswrite
r/libs/*:.") % (DATAX_HOME, DATAX_HOME, DATAX_HOME)
```

JSON設定ファイルでhdfsreaderとhdfswriterを設定します

サンプルJSONは次のとおりです。

```
{
```

```
    "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.com/",
                    "column": ["*"],
                    "fileType": "text",
                    "encoding": "UTF-8",
                    "hadoopConfig": {
                        "fs.AbstractFileSystem.ofs.impl":
"com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter",
                        "fs.ofs.impl":
"com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter",
                        "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.com/",
                    "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.CHDFSDelegateFSAdapter",
                        "fs.ofs.impl":
"com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter",
                        "fs.ofs.tmp.cache.dir": "/data/chdfs_tmp_cache",
                        "fs.ofs.user.appid": "1250000000"
                    },
                    "fieldDelimiter": ":",
                    "writeMode": "append"
                }
            }
        }]
    }
}
```

そのうち、hadoopConfigの設定はCHDFSに必要な設定であり、defaultFSはCHDFSのパスとして記入します。例えば、`ofs://f4xxxxxxxxx-hxT9.chdfs.ap-beijing.myqcloud.com/` の場合、他の設定はhdfsの設定項目と同様でかまいません。

データマイグレーションの実行

設定ファイルをhdfs_job.jsonとしてjobディレクトリ下に保存し、以下のコマンドラインを実行します。

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

次のように画面の通常出力を観察します。

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


         [total gc info] =>
                 NAME                 | totalGCCount       |
maxDeltaGCCount    | minDeltaGCCount    | totalGCTime        |
maxDeltaGCTime     | minDeltaGCTime
                 PS MarkSweep         | 1                  | 1
| 1                  | 0.024s             | 0.024s             | 0.024s
                 PS Scavenge          | 1                  | 1
| 1                  | 0.014s             | 0.014s             | 0.014s

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 records, 33 bytes | Speed 3B/s, 0 records/s | Error 0 records,
0 bytes |  All Task WaitWriterTime 0.000s |  All Task WaitReaderTime
0.033s | Percentage 100.00%
2020-03-09 16:49:59.544 [job-0] INFO  JobContainer -
タスク開始時刻                    : 2020-03-09 16:49:48
タスク終了時刻                    : 2020-03-09 16:49:59
タスク総所要時間                    :                11s
タスク平均トラフィック                       :               3B/s
記録書き込み速度                    :              0rec/s
読み出し記録総数                    :                   2
読み取り・書き込み失敗総数                       :                   0
```

CDHのCHDFS設定ガイドライン

最終更新日：：2025-11-11 17:40:22

概要

CDH(Cloudera's Distribution, including Apache Hadoop)は、業界で流行りのHadoopディストリビューションです。ここでは、CDH環境でTencent Cloud CHDFSサービスを使用して、ビッグデータの計算とストレージの分離を実現し、フレキシブルかつ低コストのビッグデータソリューションを提供する方法についてご説明します。
CHDFSビッグデータコンポーネントのサポートは次のとおりです。

コンポーネント名	CHDFSビッグデータコンポーネントのサポート状況	サービスコンポーネントの再起動の要否
Yarn	サポート	NodeManagerの再起動
Yarn	サポート	NodeManagerの再起動
Hive	サポート	HiveServerおよびHiveMetastoreの再起動
Spark	サポート	NodeManagerの再起動
Sqoop	サポート	NodeManagerの再起動
Presto	サポート	HiveServerおよびHiveMetastoreとPrestoの再起動
Flink	サポート	いいえ
Impala	サポート	いいえ
EMR	サポート	いいえ
セルフビルドコンポーネント	フォローアップサポート	なし
HBase	推奨しない	なし

バージョンの依存関係

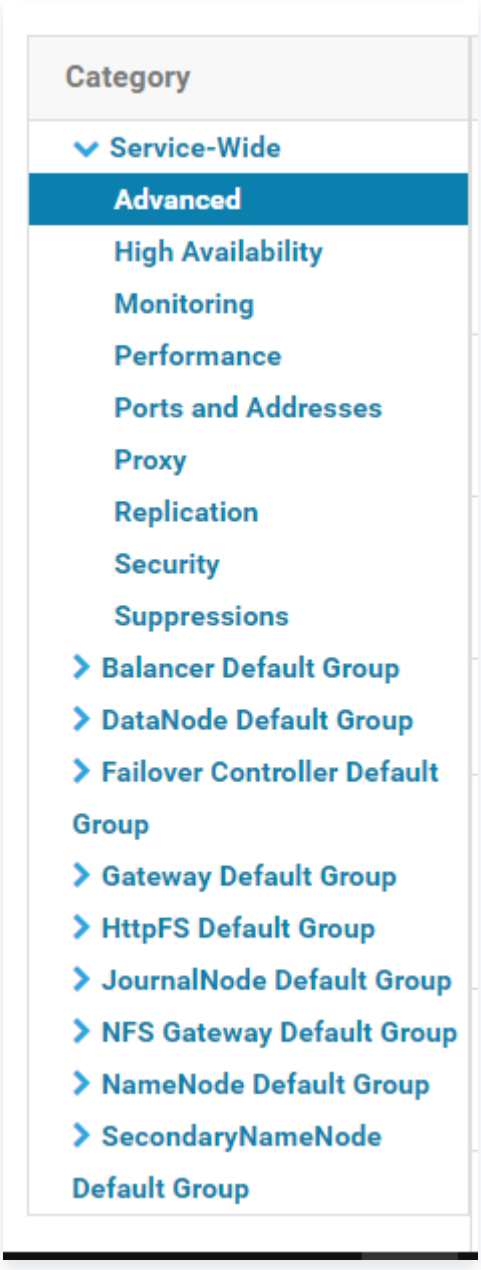
依存するコンポーネントのバージョンは次のとおりです。

- CDH 5.16.1
- Hadoop 2.6.0

使用方法

ストレージ環境の設定

1. CDH管理ページにログインします。
2. 下図に示すとおり、**設定** > *サービス範囲** > **高度**を選択して、コードセグメントの高度な設定ページに進みます。



3. Cluster-wide Advanced Configuration Snippet(Safety Valve)for core-site.xmlのコードボックスに、CHDFS設定を入力します。

```
<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>
<!--ローカルcacheの一時ディレクトリ。データの読み取りと書き込みの場合、メモリcacheが不足すると、ローカルディスクに書き込まれます。このパスが存在しない場合は、自動的に作成されます-->
<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>
```

以下は、入力必須のCHDFS設定項目です（core-site.xmlに追加する必要があります）。CHDFSの他の設定については、CHDFSのマウント をご参照ください。

CHDFS設定項目	値	意味
fs.ofs.user.appid	1250000000	ユーザーappid
fs.ofs.tmp.cache.dir	/data/emr/hdfs/tmp/chdfs/	ローカルcacheの一時ディレクトリ
fs.ofs.impl	com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter	chdfsのileSystemに対する実装クラスは、com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapterに固定されます
fs.AbstractFileSystem.ofs.impl	com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter	chdfsのAbstractFileSystemに対する実装クラスは、com.qcloud.chdfs.fs.CHDFSDelegateFSAdapterに固定されます

4. HDFSサービスを操作し、「クライアント設定のデプロイ」をクリックします。この時点で、上記のcore-site.xmlの設定がクラスター内のマシンに更新されます。
5. CHDFSの最新のSDKパッケージをCDH HDFSサービスのjarパッケージパス下に配置します。実際の値に従って置き換えてください。次に例を示します。

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

> **注意:**
> クラスター内の各マシンは、SDKパッケージを同じ場所に配置する必要があります。

データマイグレーション

Hadoop Distcpツールを使用して、CDH HDFSデータをCHDFSに移行します。詳細については、以下をご参照ください
ネイティブHDFSデータのTencent Cloud CHDFSへの移行 。

ビッグデータスイートによるCHDFSの使用

1. MapReduce

操作手順
（1）[データマイグレーション]#1)の章に従って、HDFSの関連設定を行い、CHDFSのSDK jarパッケージをHDFSの対応するディレクトリに配置します。
（2）CDHシステムのホームページでYARNを見つけてNodeManagerサービスを再起動します（TeraGen コマンドを再起動する必要はありませんが、TeraSortは内部ビジネスロジックのためにNodeMangerを再起動する必要があります。NodeManagerサービスを一律に再起動することをお勧めします）。
事例
以下に、Hadoop標準テストでのTeraGenとTeraSortの例を示します。

```
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
```

> **説明:**
> `ofs://schema` の後ろを、ユーザーCHDFSのマウントポイントパスに置き換えてください。

2. Hive

2.1 MRエンジン

操作手順

（1）データマイグレーションの章に従って、HDFSの関連設定を行い、CHDFSのSDK jarパッケージをHDFSの対応するディレクトリに配置します。
（2）CDHメインページでHIVEサービスを見つけ、Hiveserver2およびHiverMetastoreのロールを再起動します。

事例

例えば、あるユーザーの実際の業務の照会では、Hiveコマンドラインを実行してLocationを1つ作成し、CHDFSのパーティションテーブルとします。

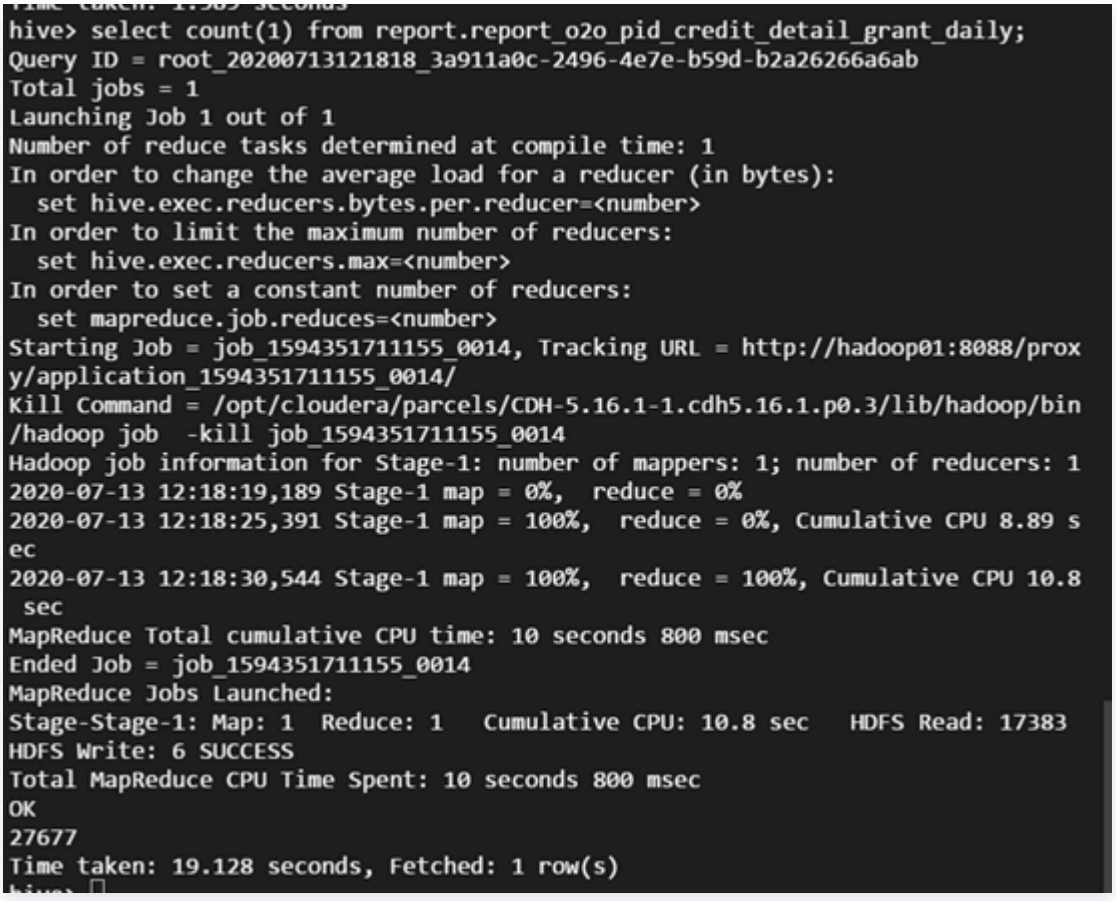
```
CREATE TABLE `report.report_o2o_pid_credit_detail_grant_daily`(
  `cal_dt` string,
  `change_time` string,
  `merchant_id` bigint,
  `store_id` bigint,
  `store_name` string,
  `wid` string,
  `member_id` bigint,
  `meber_card` string,
  `nickname` string,
  `name` string,
  `gender` string,
  `birthday` 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_credit_detail_gr
ant_daily'
```

```
TBLPROPERTIES (
  'last_modified_by'='work',
  'last_modified_time'='1589310646',
  'transient_lastDdlTime'='1589310646')
```

sql照会を実行します。

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

観察の結果は次のとおりです。



2.2 Tezエンジン

Tezエンジンは、CHDFSのjarパッケージをTezの圧縮パッケージにインポートする必要があります。以下は、apache-tez.0.8.5を例として説明しています。

操作手順

（1）CDHクラスターによってインストールされたtezパッケージを見つけて、解凍します。

例：/usr/local/service/tez/tez-0.8.5.tar.gz。

（2）CHDFSのjarパッケージを解凍したディレクトリ下に置き、圧縮したパッケージを再圧縮して出力します。

（3）tez.lib.urisで指定されたパス下に、新しい圧縮パッケージをアップロードします（以前にパスがある場合

は、直接置き換えてください）。
（4）CDHメインページでHIVEを見つけて、hiveserverおよびhivemetastoreを再起動します。

3. Spark

操作手順

（1）データマイグレーションの章に従って、HDFSの関連設定を行い、CHDFSのSDK jarパッケージをHDFSの対応するディレクトリに配置します。
（2）NodeManagerサービスを再起動します。

事例

例として、CHDFSによるSpark example word countテストの実施を取り上げます。

```
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
```

実行結果は次のとおりです。

```
20/07/17 18:35:05 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 3 ms
20/07/17 18:35:05 INFO executor.Executor: Finished task 0.0 in stage 1.0 (TID 1). 1870 bytes result sent to driver
20/07/17 18:35:05 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 31 ms on localhost (executor driver) (1
20/07/17 18:35:05 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
20/07/17 18:35:05 INFO scheduler.DAGScheduler: ResultStage 1 (collect at JavaWordCount.java:68) finished in 0.031 s
20/07/17 18:35:05 INFO scheduler.DAGScheduler: Job 0 finished: collect at JavaWordCount.java:68, took 0.548912 s
And: 4
day.: 1
God: 4
Let: 1
light:: 1
it: 1
light,: 1
evening: 1
was: 2
that: 1
light.: 1
be: 1
Day,: 1
said,: 1
darkness.: 1
he: 1
first: 1
there: 2
light: 2
Night.: 1
morning: 1
darkness: 1
were: 1
saw: 1
divided: 1
and: 4
good:: 1
from: 1
called: 2
the: 8
```

4. Sqoop

操作手順

（1）データマイグレーションの章に従って、HDFSの関連設定を行い、CHDFSのSDK jarパッケージをHDFSの対応するディレクトリに配置します。
（2）CHDFSのSDKjarパッケージもsqoopディレクトリ下に配置する必要があります
（例：/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/lib/sqoop/）。

（3）NodeManagerサービスを再起動します。

事例

例として、MYSQLテーブルのCHDFSへのエクスポートを取り上げます。リレーショナルデータベースとHDFSのインポート・エクスポート ドキュメントをご参照ください。

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

実行結果は次のとおりです。

```
20/07/17 18:48:33 INFO mapreduce.Job:  map 100% reduce 0%
20/07/17 18:48:33 INFO mapreduce.Job: Job job_1594976906551_0011 completed successfully
20/07/17 18:48:33 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=526689
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=295
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=3
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
                OFS: Number of bytes read=0
                OFS: Number of bytes written=104
                OFS: Number of read operations=0
                OFS: Number of large read operations=0
                OFS: Number of write operations=3
        Job Counters
                Launched map tasks=3
                Other local map tasks=3
                Total time spent by all maps in occupied slots (ms)=36308
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=9077
                Total vcore-milliseconds taken by all map tasks=9077
                Total megabyte-milliseconds taken by all map tasks=37179392
        Map-Reduce Framework
                Map input records=3
                Map output records=3
                Input split bytes=295
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=277
                CPU time spent (ms)=6430
                Physical memory (bytes) snapshot=1371717632
                Virtual memory (bytes) snapshot=18832379904
                Total committed heap usage (bytes)=6655311872
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=104
20/07/17 18:48:33 INFO mapreduce.ImportJobBase: Transferred 0 bytes in 13.0961 seconds (0 bytes/sec)
20/07/17 18:48:33 INFO mapreduce.ImportJobBase: Retrieved 3 records.
20/07/17 18:48:33 INFO fs.CHDFSHadoopFileSystemAdapter: actual-file-system-close usedTime: 13
20/07/17 18:48:33 INFO fs.CHDFSHadoopFileSystemAdapter: end-close time: 1594982913402, total-used-time: 13650
```

5. Presto

操作手順

（1） データマイグレーション の章に従って、HDFSの関連設定を行い、CHDFSのSDK jarパッケージをHDFSの対応するディレクトリに配置します。

（2）CHDFSのSDK jarパッケージもprestoディレクトリ下に配置する必要があります（例：/usr/local/services/cos_presto/plugin/hive-hadoop2）。
（3）prestoはhadoop common下のgson-2...jarをロードしないため、gson-2...jarもprestoディレクトリ下に配置する必要があります（例： /usr/local/services/cos_presto/plugin/hive-hadoop2、CHDFSのみがgsonに依存します）。
（4）HiveServer、HiveMetaStoreおよびPrestoサービスを再起動します。

事例

例として、HIVEによってLocationをCHDFSとして作成したテーブルの照会を取り上げます。

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

> **説明：**
> chdfs_test_tableはlocationがofs schemeのテーブルです。

確認の結果は次のとおりです。

```
presto:inventory_search> select * from chdfs_test_table_0720 where bucket is not null limit 1;
   appid    |       bucket        |                          path                          |  size   |
------------+---------------------+--------------------------------------------------------+---------+
 125        | ssuupv80105841qq206 | 444600684/20190316/3/01a9ee49bd4045179c92e319ff03b810 | 3800424 |
(1 row)

Query 20200720 112833 00061 thb89, FINISHED, 7 nodes
```

CHDFS Ranger権限システムソリューション

最終更新日：： 2024-01-19 16:58:09

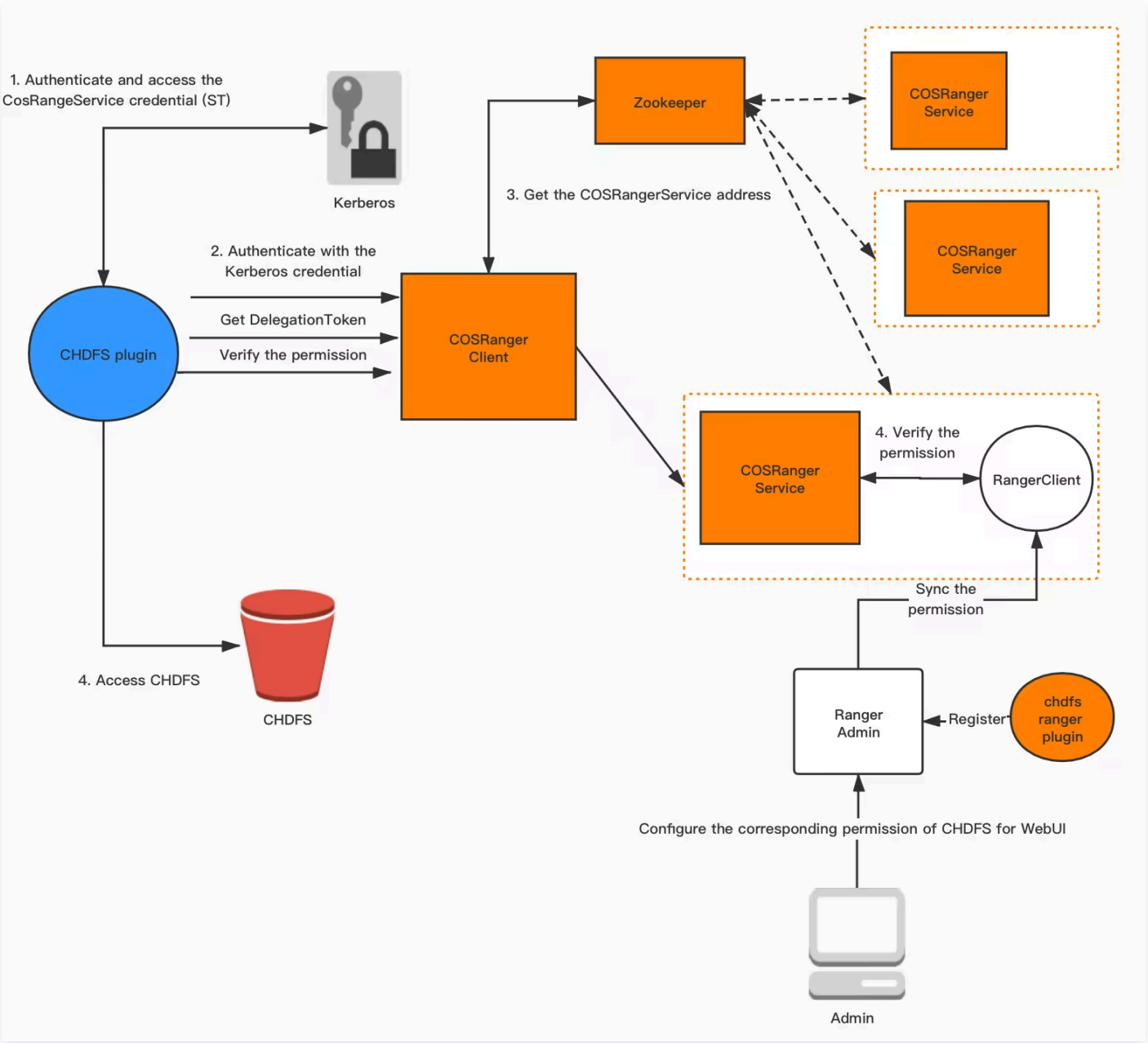
背景

ビッグデータユーザーがストレージ・コンピューティング分離を使用した後、Cloud HDFS(CHDFS)でデータをホストします。CHDFSは、HDFSに類する権限システム制御を提供します。Hadoop RangerはHDFS権限に基づいて、ユーザーグループの権限設定や特定のプレフィックスの権限設定など、よりきめ細かい権限制御を提供します。また同時にHadoop Rangerは、ワンストップの権限システムソリューションとして、ストレージ側の権限の管理と制御をサポートするだけでなく、YARNやHiveなどのコンポーネントの権限制御もサポートします。お客様の利便性を維持するため、お客様がRangerを使用してCHDFS権限を制御する際に便利なCHDFSのRangerアクセスソリューションを提供しています。

メリット

- Hadoop権限の習慣と互換性のある細粒度の権限制御。
- ユーザーは、ビッグデータコンポーネントとクラウドホスティングストレージを統合管理する権限を持っています。

ソリューションアーキテクチャ



Hadoop権限システムでは、認証はKerberosによって提供され、権限付与と認証はRangerが担当します。これをベースとして、以下のコンポーネントを提供し、CHDFSのRanger権限スキームをサポートします。

- CHDFS-Ranger-Plugin：Rangerサーバー用のサービス定義プラグインを提供します。Ranger側のCHDFSサービスの説明が提供され、このプラグインがデプロイされると、ユーザーはRangerの制御ページで対応するアクセス許可ポリシーを入力できるようになります。
- COSRangerService：このサービスは、Rangerのクライアントを統合し、Rangerサーバーから権限ポリシーを定期的に同期し、お客様の認証リクエストを受信した後、ローカルで権限チェックを行います。同時に、HadoopでのDelegationTokenの発行や更新などのインターフェースを提供します。すべてのインターフェースは、Hadoop IPCを介して定義されます。
- CosRangerClient：COSNプラグインはそれを動的にロードし、権限チェックのリクエストをCosRangerServiceに転送します。

デプロイ環境

- Hadoop環境。
- ZooKeeper、Ranger、Kerberosサービス（認証の必要がある場合は、デプロイします）。

> **説明：**
> 以上のサービスは成熟したオープンソースコンポーネントであるため、お客様はご自身でインストールすることができます。

コンポーネントのデプロイ

CHDFS-Ranger-Pluginのデプロイ

CHDFS-Ranger-Pluginは、Ranger Adminコンソールにおけるサービスの種類を拡張します。ユーザーはRangerコンソールで、CHDFSに関連する操作権限を設定することができます。

コードアドレス

Github のranger-pluginディレクトリから取得できます。

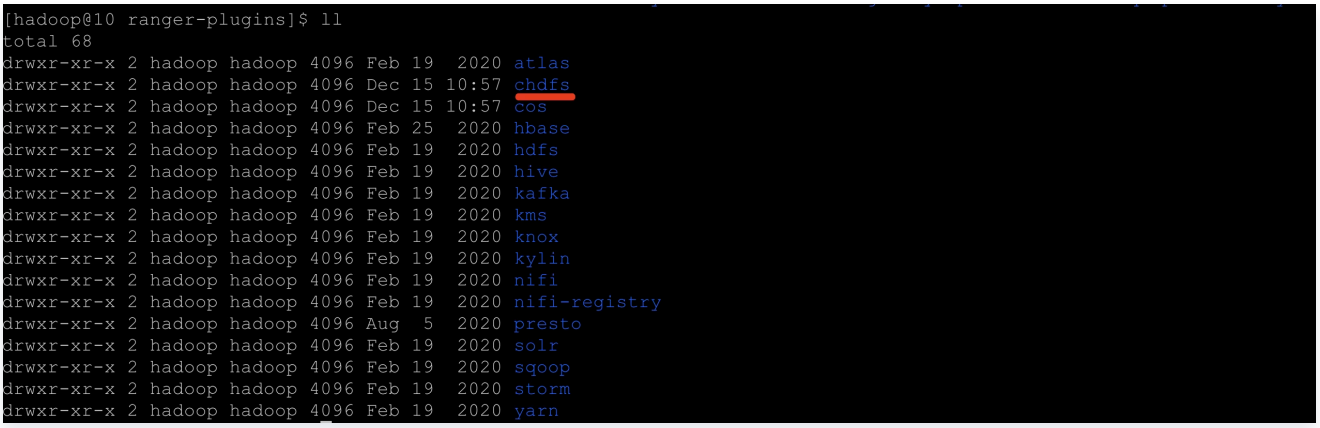
バージョン

V1.2およびそれ以降のバージョン。

デプロイ手順

1. Rangerのサービス定義ディレクトリの下にCOSディレクトリを新規作成します（注意：ディレクトリの権限には、少なくともxとrの権限が必要です）。
1. Tencent CloudのEMR環境の場合、パスはranger/ews/webapp/WEB-INF/classes/ranger-pluginsです。
2. セルフビルドのhadoop環境では、rangerディレクトリ下のfind hdfsなどの方法によって、rangerサービスに接続されているコンポーネントを探し、ranger-pluginsディレクトリの場所を見つけることができま

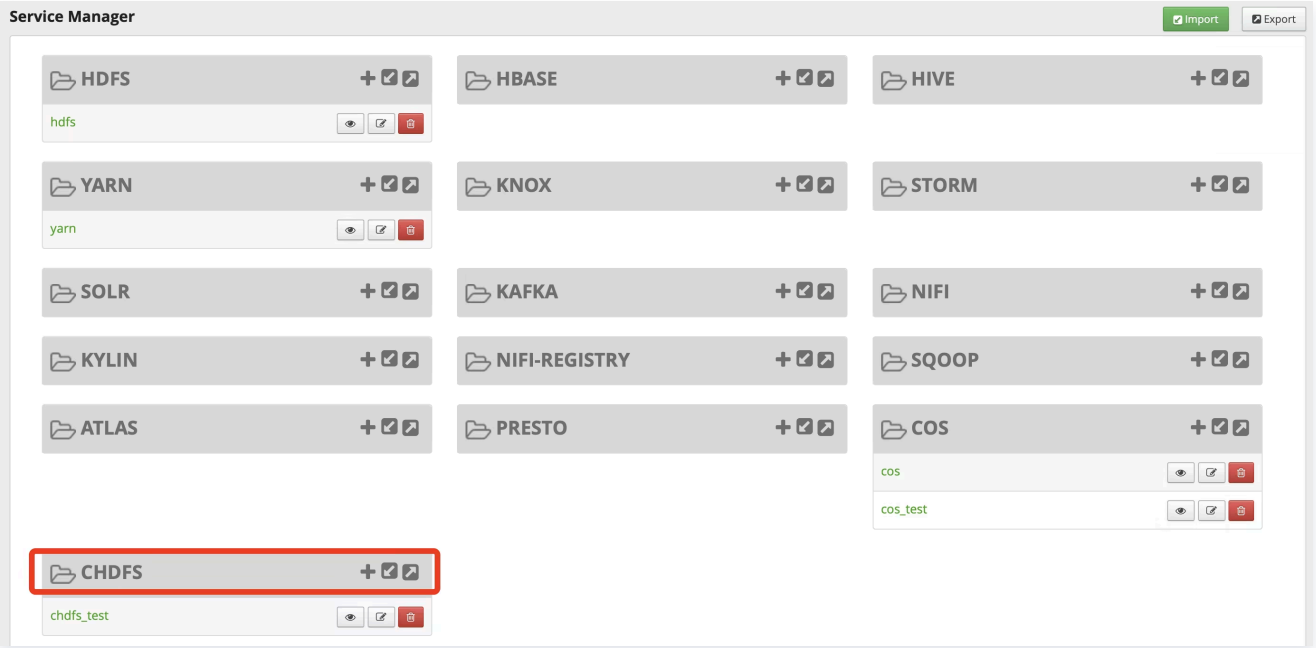
す。



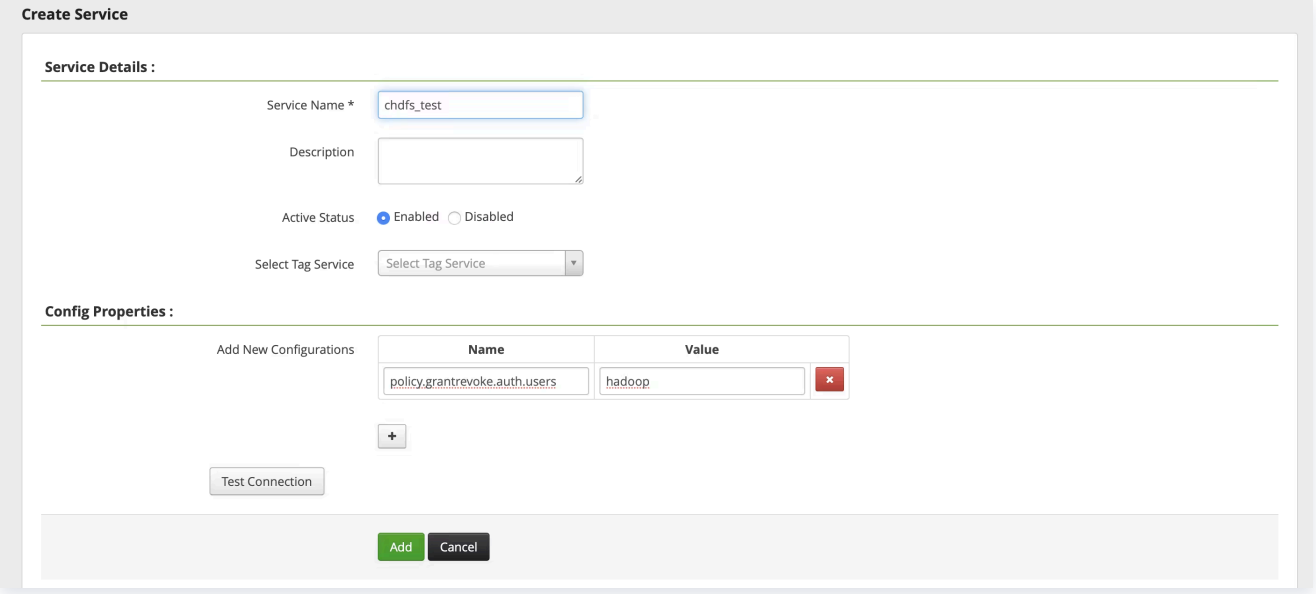
3. CHDFSディレクトリ下にcos-chdfs-ranger-plugin-xxx.jarを配置します（注意：jarパッケージには少なくともr個の権限があります）。
4. Rangerサービスを再起動します。
5. RangerにCHDFS Serviceを登録します。次のコマンドを参照できます。

```
##サービスを発行するには、Ranger管理者アカウントのパスワードとRangerサービスのアドレスを渡す必要があります。
##Tencent Cloud EMRクラスターの場合、管理者ユーザーはrootであり、パスワードはemrクラスターの構築時に設定されたrootパスワードです。rangerサービスのIPは、EMRのmasterノードのIPに置き換えられます。
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://${rangerServerAddr}/service/plugins/definitions
##定義されたサービスを削除する場合は、作成されたばかりのサービスが渡され、サービスIDが返されます
serviceId=102
curl -v -u${adminUser}:${adminPasswd} -X DELETE -H "Accept:application/json" -H "Content-Type:application/json" http://${rangerServerAddr}/service/plugins/definitions/${serviceId}
```

6. 次に示すように、サービスの作成に成功すると、RangerコンソールにCHDFSサービスが表示されます。

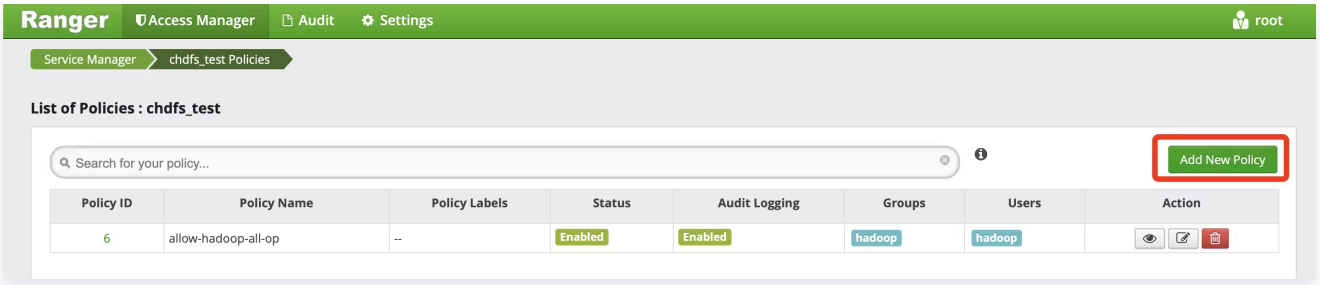


7. CHDFSサービス側の【+】をクリックして、新しいサービスインスタンスを定義します。サービスインスタンス名は、`chdfs` や `chdfs_test` などにカスタマイズできます。サービスの設定は次のとおりです。

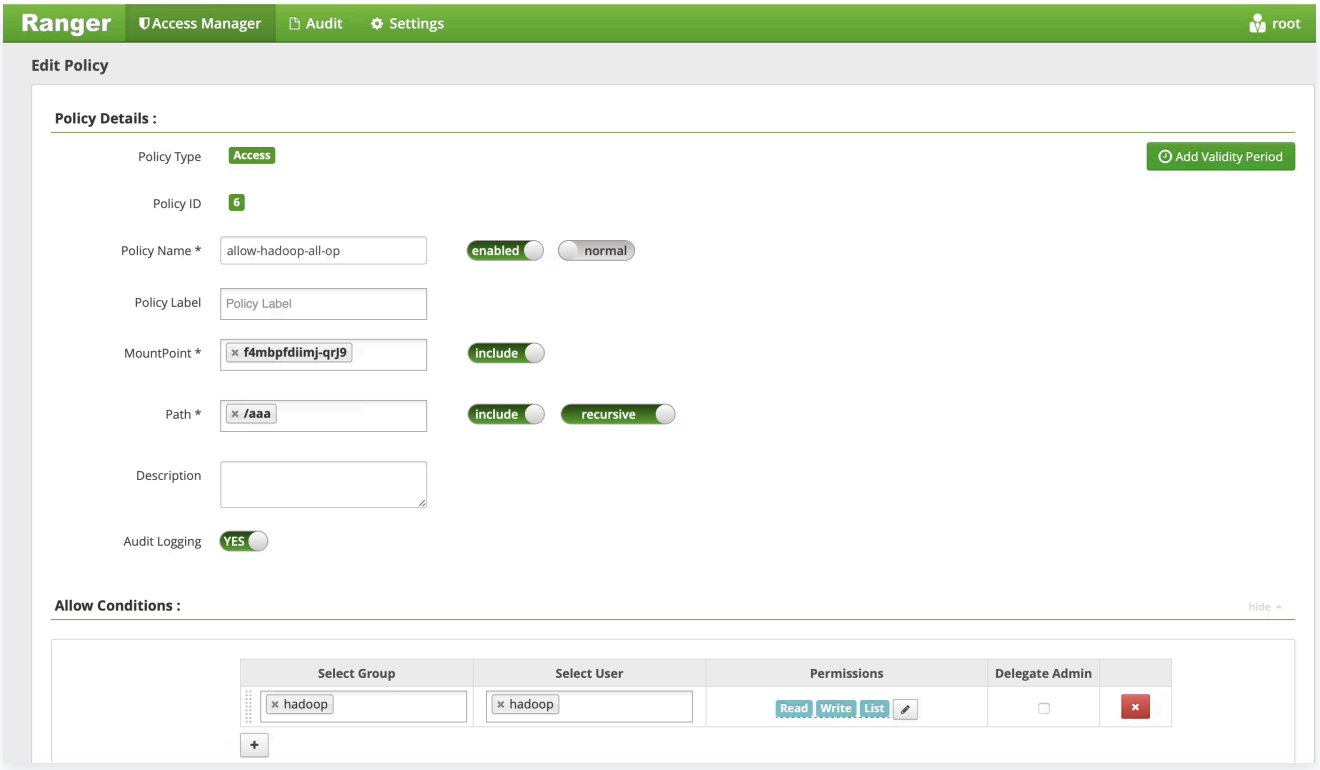


そのうちpolicy.grantrevoke.auth.usersは、COSRangerServiceサービスを後で起動するためのユーザー名を設定する必要があります。通常はhadoopに設定することをお勧めします。その後のCOSRangerServiceは、このユーザー名で起動できます。

8. 次に示すように、新しく発行されたCHDFSサービスインスタンスをクリックして、policyを追加します。



9. ジャンプインターフェースで、以下のパラメータを設定します。具体的な手順は次のとおりです。



- **MountPoint**: マウントポイントの名前。形式はf4mxxxxxx-yyyyなどのスタイルで、CHDFSコンソールにログインして確認することができます。
- **path**: CHDFSパス。CHDFSパスは、必ず `/` で始まる必要があることにご注意ください。
  - include：設定された権限がpath自体またはpath以外の他のパスに適用されることを示します。
  - recursive：権限がpathだけでなく、path下のサブメンバー（つまり、再帰的なサブメンバー）にも適用できることを示します。通常、pathがディレクトリに設定されている場合に用いられます。
- **user/group**：ユーザー名とユーザーグループ。ここでは「OR」関係です。ユーザー名またはユーザーグループがそれらのうちいずれかを満たす場合、対応する操作権限を有することができます。
- **Permissions**:
  - Read：読み取り操作。オブジェクトのダウンロード、オブジェクトメタデータの照会など、Cloud Object StorageでのGET、HEADといった操作に対応します。

- Write：書き込み操作。オブジェクトのアップロードなど、COS内のPUTといった変更操作に対応します。
- Delete：削除操作。COS内のObjectの削除に対応します。HadoopのRename操作の場合、元のパスに対する削除操作権限と、新しいパスに対する書き込み操作権限が必要です。
- List：トラバース権限。COS内のList Objectに対応します。

COS-Ranger-Serviceのデプロイ

COS-Ranger-Serviceは、権限システム全体のコアであり、rangerクライアントの統合、ranger client認証リクエストの受信、token発行・更新のリクエストや一時キー発行リクエストを担当します。同時に、センシティブ情報（Tencent Cloudのキー情報）が配置されているエリアでもあり、通常は踏み台サーバーにデプロイされ、クラスター管理者のみが操作や設定の確認などを行うことができます。
COS-Ranger-Serviceは、1つの本番データベースと1つ以上のスタンバイによるHAデプロイをサポートし、DelegationTokenのステータスはHDFSに永続化されます。ZKを介してロックを取得することにより、Leaderの身分を決定します。Leaderの身分を取得するサービスは、COS Ranger Clientがルーティング・アドレッシングを実行できるように、アドレスをZKに書き込みます。

コードアドレス

Github のcos-ranger-serverディレクトリ下から取得できます。

バージョン

V1.2およびそれ以降のバージョン。

デプロイ手順

1. COS Ranger Serviceサービスコードをクラスター内の複数のマシンにコピーします。本番環境は、少なくとも2台のマシン（1台は本番データベースでもう1台はスタンバイ）にすることをお勧めします。センシティブ情報が含まれるため、踏み台サーバーまたは厳格に制御されているマシンにすることをお勧めします。
2. cos-ranger.xmlファイルの関連設定を変更します。そのうち変更が必要な設定項目を、以下に示します。設定項目の説明については、ファイル内のコメントをご参照ください。
   - qcloud.object.storage.rpc.address
   - qcloud.object.storage.status.port
   - qcloud.object.storage.enable.chdfs.ranger
   - qcloud.object.storage.zk.address
3. ranger-chdfs-security.xmlファイルの関連設定を変更します。そのうち変更が必要な設定項目を、以下に示します。設定項目の説明については、ファイル内のコメントをご参照ください。
   - ranger.plugin.chdfs.policy.cache.dir

- ranger.plugin.chdfs.policy.rest.url
- ranger.plugin.chdfs.service.name

4. start_rpc_server.shのhadoop_conf_pathとjava.library.pathの設定を変更します。これら2つの設定はそれぞれ、hadoop設定ファイルが配置されているディレクトリ（core-site.xml、hdfs-site.xmlなど）とhadoop native libパスを指します。
5. 次のコマンドを実行して、サービスを起動します。

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

6. 起動に失敗した場合は、log下のerrorログをチェックして、エラー情報があるかどうかを確認します。
7. cos-ranger-serviceは、HTTPポートステータスの表示をサポートします（ポート名はqcloud.object.storage.status.port、デフォルト値は9998）。ユーザーは次のコマンドを使用して、ステータス情報（leader、認証数の統計など）を取得できます。

```
# 以下の10.xx.xx.xxxをranger serviceがデプロイされているマシンのIPに置き
換えてください
# port 9998はqcloud.object.storage.status.port設定値に設定されます
curl -v http://10.xx.xx.xxx:9998/status
```

COS-Ranger-Clientのデプロイ

COS-Ranger-Clientは、hadoop chdfsプラグインによって動的にロードされ、COS-Ranger-Serviceにアクセスするための関連するリクエストを代行します。例えば、token、認証操作などです。

コードアドレス

Github のcos-ranger-clientディレクトリ下から取得できます。

バージョン

V1.0およびそれ以降のバージョン。

デプロイ方法

1. cos-ranger-client jarパッケージをCHDFSプラグインと同じディレクトリ下にコピーします（ご自身のhadoopバージョンと一致するjarパッケージを選択してコピーしてください）。
2. core-site.xmlに次の設定項目を追加します。

```
```
<configuration>
            <!--*****設定必須********-->
            <!-- zkのアドレス、クライアントはzkからranger-serviceのサービス
アドレスを照会して取得します -->
            <property>
                <name>qcloud.object.storage.zk.address</name>
                <value>10.0.0.8:2121</value>
            </property>

            <!--***オプション設定****-->
            <!-- cos ranger service端末で使用されるkerberos認証情報を設定し
ます。cos ranger service端末の設定をご参照ください。必ず一致を維持する必要があ
ります。認証要件がない場合、設定は必要ありません -->
            <property>

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

          <!--***オプション設定****-->
          <!-- zkに記録されたranger serverのipアドレスパス。ここではデフォル
ト値が使用され、設定は必ずcos-ranger-serviceの設定と一致する必要があります -->
           <property>

<name>qcloud.object.storage.zk.leader.ip.path</name>

<value>/ranger_qcloud_object_storage_leader_ip</value>
           </property>
</configuration>
```
```

CHDFSのデプロイ

バージョン

V2.1およびそれ以降のバージョン。

デプロイ方法

CHDFSプラグインのデプロイ方法については、CHDFSのマウント ドキュメントを参照し、rangerを有効にして以下の方法を追加して行います。

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

検証

1. hadoop cmdを使用して、chdfsへのアクセスに関連する操作を実行します。以下に例を示します。

```
# マウントポイント、パスなどをご自分の実際の情報に置き換えます。
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.txt
hadoop fs -rm ofs://f4mxxxxyyyy-zzzz.chdfs.ap-
guangzhou.myqcloud.com/exampleobject.txt
```

2. MR Jobを使用して検証します。検証の前に、Yarn、Hiveなどの関連サービスを再起動する必要があります。

よくあるご質問

Kerberosをインストールする必要がありますか。

内部でのみ使用されるクラスターなど、所在するクラスターでユーザーが信頼できる場合は、Kerberosをインストールして認証要件を満たすことができます。ユーザーが認証操作のみを行う場合、権限を有さないお客様による誤操作を避けるため、Kerberosをインストールせずにrangerのみを使用して認証を行うことができます。Kerberosをインストールすると、パフォーマンスが低下します。ユーザーは、ご自分の総合的なセキュリティ要

件とパフォーマンス要件を考慮してください。認証が必要な場合は、Kerberosを有効にした後、COS Ranger ServiceとCOS Ranger Clientの関連する項目を設定する必要があります。

Rangerが有効になっていても、Policyが何も設定されていない場合、またはいずれのPolicyにもマッチしていない場合、どのように操作すればいいですか。

いずれのpolicyにもマッチしない場合、この操作はデフォルトで拒否されます。

rangerを有効にした後、CHDFS端末はPOSIX認証を実行しますか。

Ranger認証はクライアント環境で実行されます。rangerによって認証されたリクエストはCHDFSサーバーに送信され、サーバーはデフォルトでPOSIX認証を実行します。そのため、権限がRanger端末で制御されている場合は、CHDFSコンソールでPOSIX権限をオフにしてください。