Cloud HDFS

操作ガイド

製品ドキュメント

カタログ：

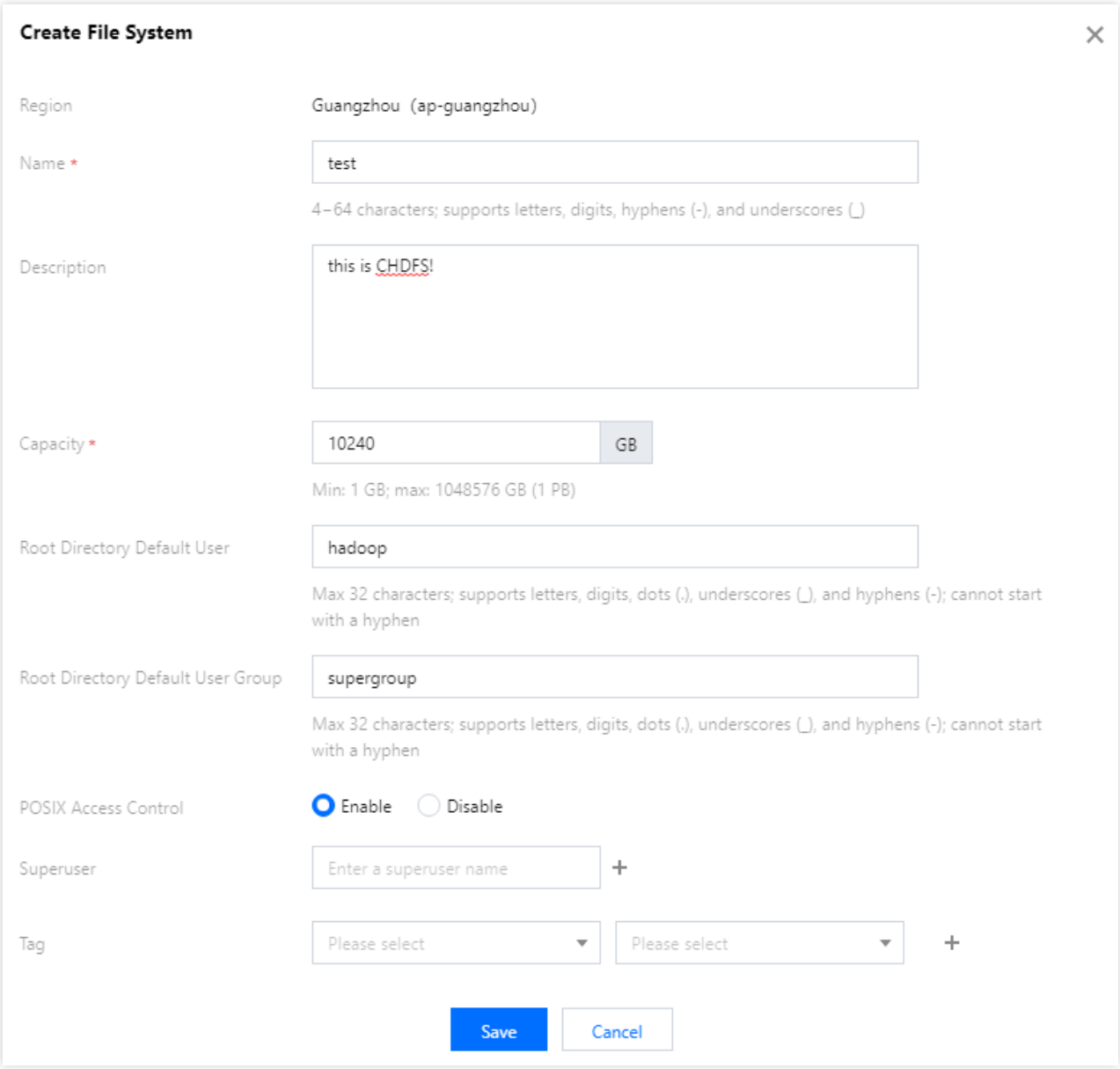
操作ガイド
CHDFSの作成

最終更新日：：2024-01-19 16:58:09

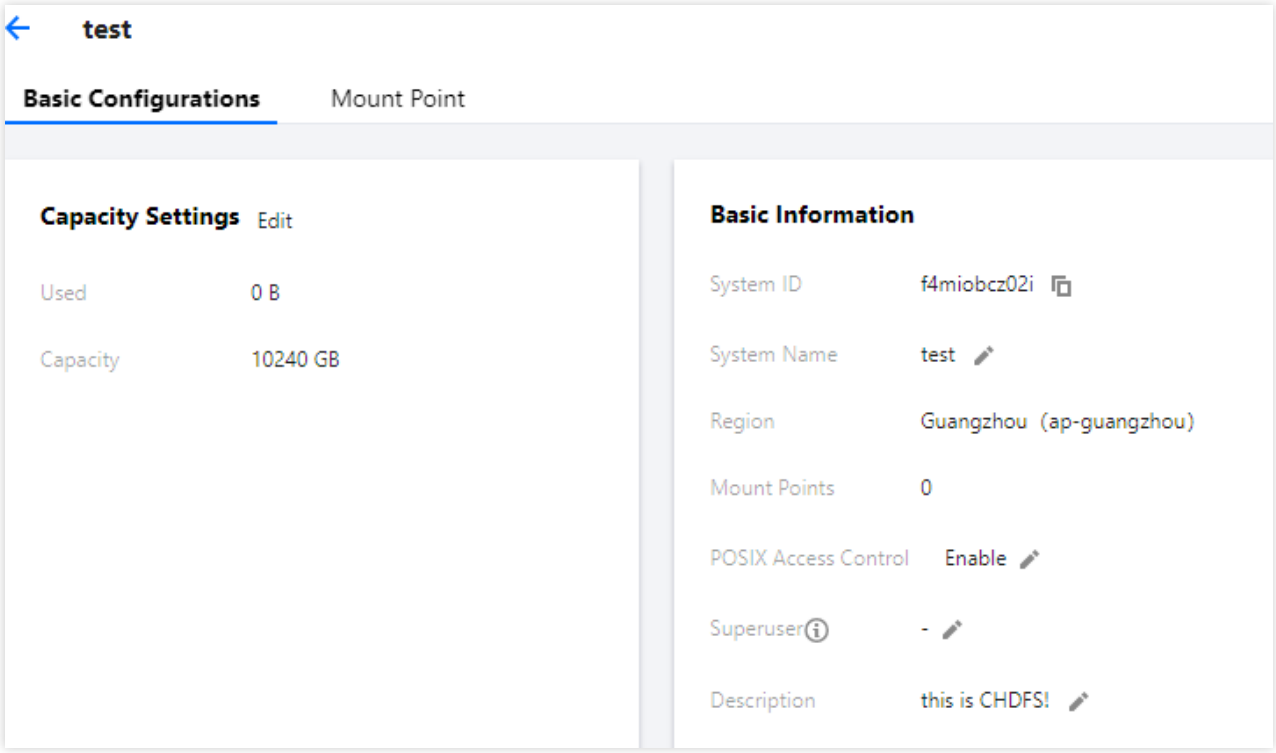
ここでは、CHDFSコンソールでファイルシステムを作成する方法について詳しくご説明します。

操作手順

1. CHDFSコンソールにログインし、左側ナビゲーションバーの【ファイルシステム】をクリックして、広州など、必要とするリージョンを選択します。
2. 【新規作成】をクリックし、ファイルシステムの新規作成ページに名前と説明を入力します。
名前：ファイルシステム名、大文字と小文字、数字および-または\\ _の組み合わせのみをサポートし、長さは4〜64文字とします。
説明：ファイルシステムの説明情報。



3. 【保存】をクリックすると、CHDFSを新規作成できます。
4. CHDFSの【設定】をクリックすると、CHDFSの基本設定とマウント情報を確認できます。下図のとおりです。



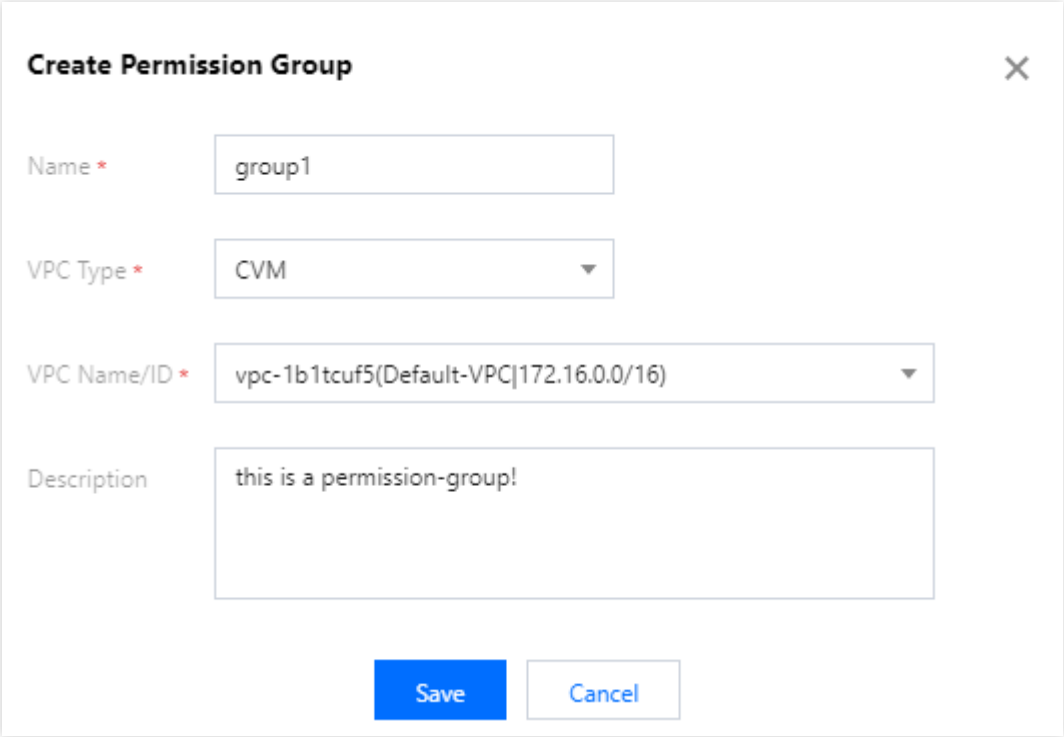
権限グループの作成

最終更新日：：2024-01-19 16:58:09

権限グループは、CHDFSの権限管理を実現します。CHDFSを使用する前に、権限グループを作成する必要があります。ここでは、権限グループを作成する方法について詳しくご説明します。

操作手順

1. CHDFSコンソールにログインし、左側ナビゲーションバーの【権限グループ】をクリックして、広州など、必要とするリージョンを選択します。
2. 【新規作成】をクリックし、ポップアップした権限グループの新規作成ウィンドウに名前と説明を入力します。
名前：ファイルシステム名、大文字と小文字、数字および-または_の組み合わせのみをサポートし、長さは4～64文字とします。
-説明：権限グループの説明情報。



3. 【保存】をクリックすると、権限グループが新規作成されます。

権限ルールの作成

最終更新日：：2024-01-19 16:58:09

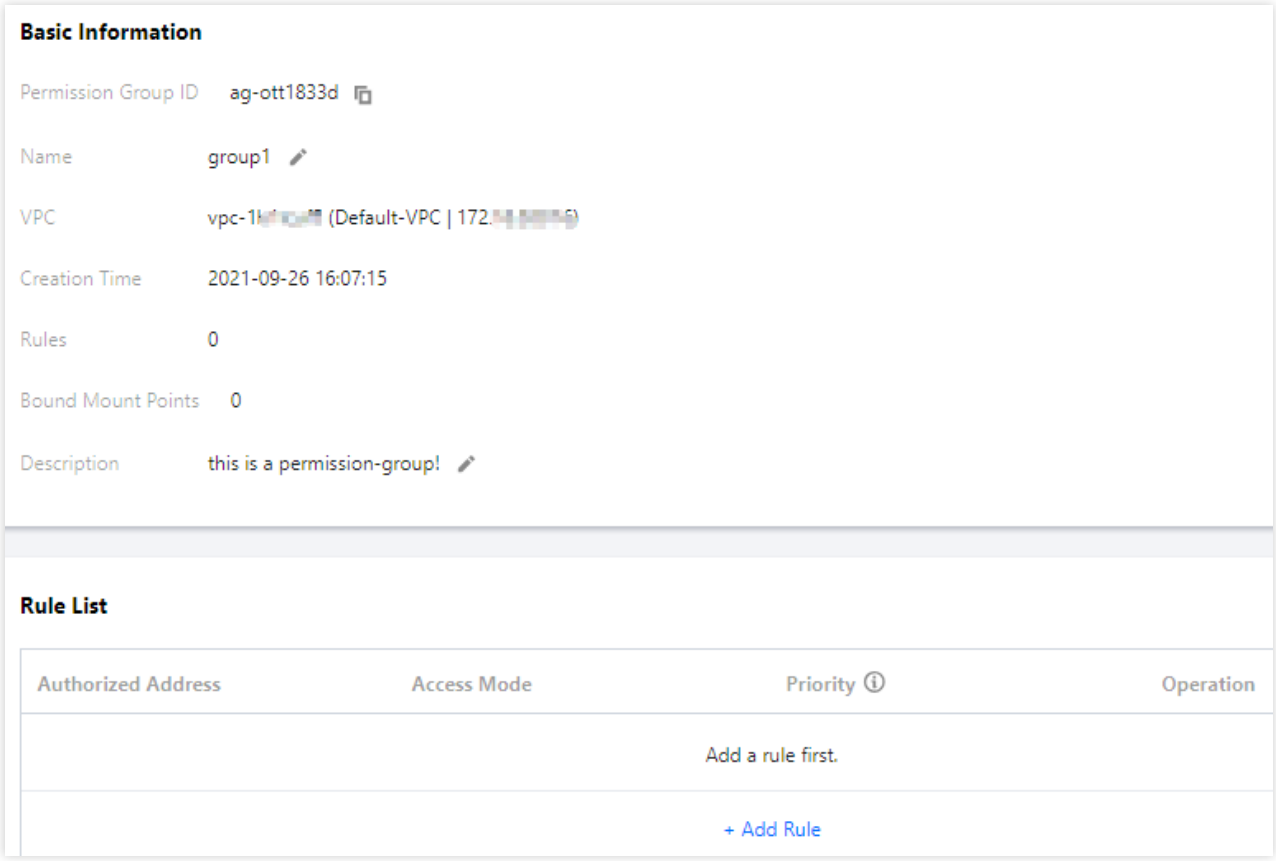
権限グループには、CHDFSの権限管理を実現するためのさまざまな権限ルールが含まれています。CHDFSを使用する前に、作成した権限グループに権限ルールを作成する必要があります。ここでは、権限ルールを作成する方法について詳しくご説明します。

前提条件

権限グループが作成されている必要があります。詳細については、権限グループの作成をご参照ください。

操作手順

1. CHDFSコンソールにログインし、左側ナビゲーションバーの【権限グループ】をクリックして、広州など、必要とするリージョンを選択します。
2. 操作が必要な権限グループを見つけ、【ルールの追加】をクリックし、権限グループ設定ページに進むと、権限グループの基本情報と権限ルールリストが確認できます。ルールリストに権限付与アドレス、アクセスモードおよび優先度を入力します。



権限付与アドレス：IPアドレスまたはネットワークセグメントをサポートします。このIPアドレスまたはネットワークセグメントに対し、10.10.1.2や10.10.1.2/20など、CHDFSにアクセスする権限が付与されることを意味します。
アクセスモード：読み取り/書き込みおよび読み取り専用をサポートします。CHDFSの読み取り/書き込みおよび読み取り専用のアクセス権限が付与されることを意味します。
優先度：1～100をサポートし、1が最高の優先度になります。CHDFSが複数の権限ルールと一致する場合、優先度の高いルールが優先度の低いルールを上書きします。
3. 【保存】をクリックすると、権限ルールが作成されます。

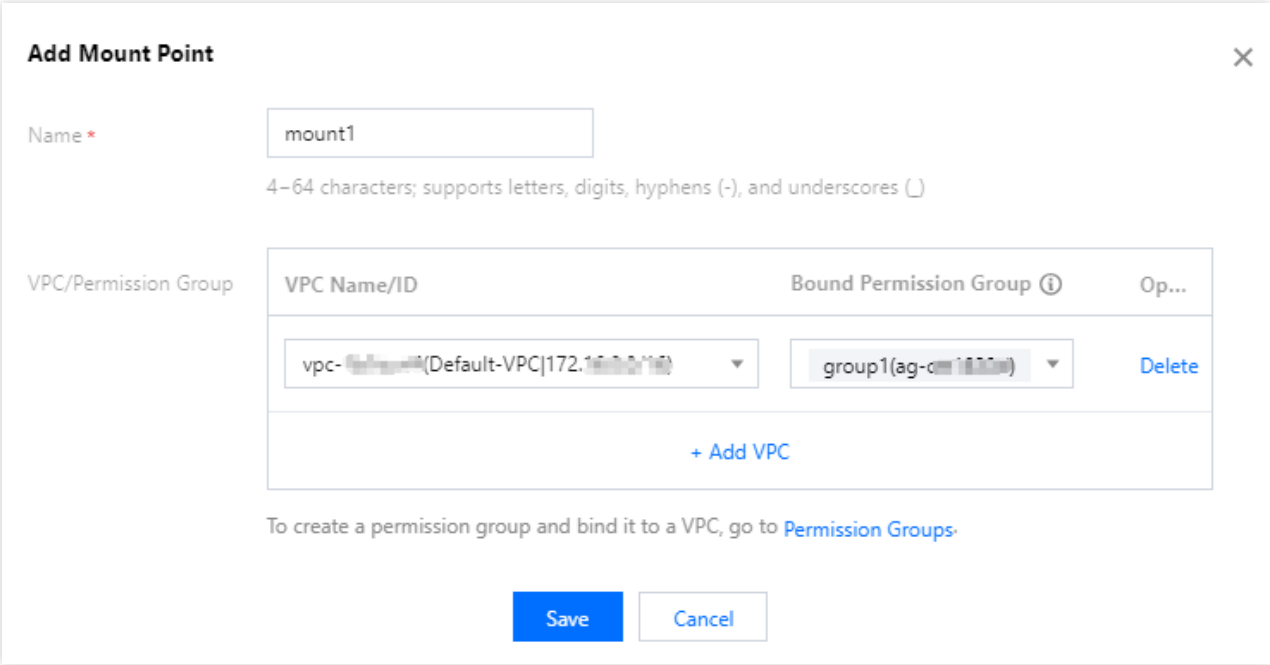
マウントポイントの作成

最終更新日：：2024-01-19 16:58:09

CVM、CPM 2.0またはコンテナは、マウントポイントを介してCHDFSのデータにアクセスします。マウントポイントは、CHDFSがVPCでアクセスするターゲットアドレスです。同時に、各マウントポイントは1つのドメイン名に対応します。ここでは、マウントポイントを作成する方法について詳しくご説明します。

操作手順

1. CHDFSコンソールにログインし、左側ナビゲーションバーの【ファイルシステム】をクリックして、広州など、必要とするリージョンを選択します。
2. 操作が必要なCHDFSを見つけ、その【設定】をクリックすると、基本設定とマウント情報が確認できます。
3. 【マウントポイント】>【マウントポイントの追加】を選択し、マウントポイントの追加ページで名前を入力し、VPCネットワークと権限グループを指定します。
名前：マウントポイント名、大文字と小文字、数字および-または_の組み合わせのみをサポートし、長さは4～64文字とします。
VPCネットワーク名/ID：VPCネットワークを選択します。
権限グループ：権限グループを選択します。新規作成していない場合は、まず権限グループの作成を行ってください。



4. 【保存】をクリックすると、マウントポイントが新規作成されます。

CHDFSのマウント

最終更新日：：2024-01-19 16:58:09

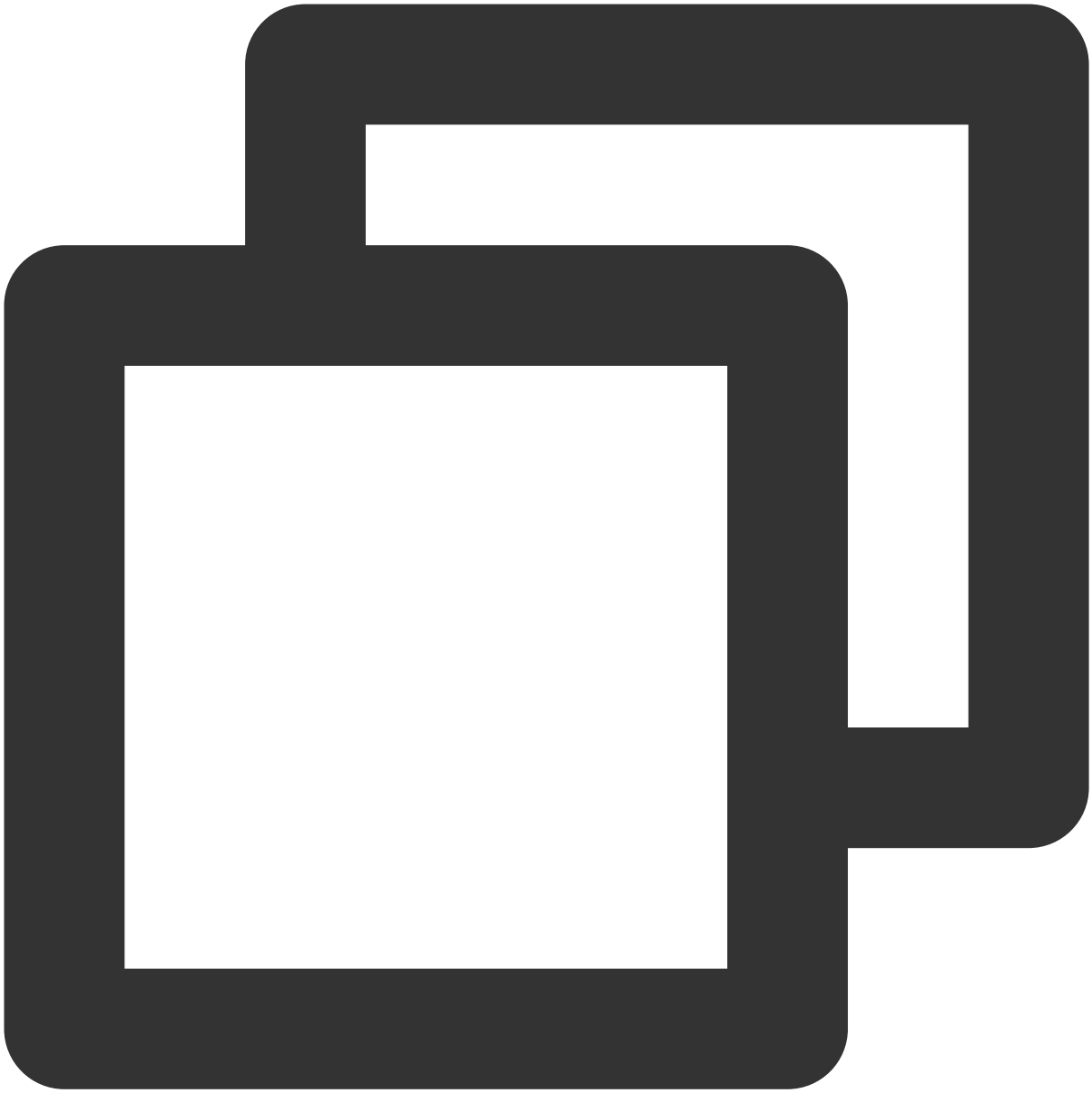
CHDFSとマウントポイントを作成すると、マウントポイントを介してCHDFSをマウントできるようになります。ここでは、CHDFSをマウントする方法について詳しくご説明します。

前提条件

マウントされたマシンまたはコンテナにJava1.8がインストールされていることを確認します。
マウントされたマシンまたはコンテナのVPCが、マウントポイントの指定されたVPCと同じであることを確認します。
マウントされたマシンまたはコンテナのVPC IPが、マウントポイントの指定された権限グループのうち1つの権限ルールの権限を付与されたアドレスと一致することを確認します。

操作手順

1. CHDFS-HadoopでJARパッケージをダウンロードします。
2. JARパッケージを対応するディレクトリ下に配置します。EMRクラスターの場合、すべてのノードの `/usr/local/service/hadoop/share/hadoop/common/lib/` ディレクトリ下に同期できます。
3. core-site.xmlファイルを編集し、次の基本設定を追加します。



```
<!--chdfsの実装クラス-->
<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
<property>
```

```
  <name>fs.ofs.tmp.cache.dir</name>
  <value>/data/chdfs_tmp_cache</value>
</property>
<!--appId-->
<property>
  <name>fs.ofs.user.appid</name>
  <value>1250000000</value>
</property>
```

4. core-site.xmlをすべてのhadoopノードに同期します。

説明：

EMRクラスターの場合、EMRコンソールのコンポーネント管理で上記の手順3と4において、HDFSの設定を変更することができます。

5. hadoop fsコマンドラインツールを使用して、`hadoop fs –ls ofs://${mountpoint}/`コマンドを実行します。ここで、mountpointはマウントアドレスです。ファイルリストが正常にリストアップされている場合は、CHDFSが正常にマウントされたことを意味します。

6. ユーザーは、hadoopの他の設定項目またはmrタスクを使用してCHDFSでデータタスクを実行することもできます。mrタスクの場合、`-Dfs.defaultFS=ofs://${mountpoint}/`によって、このタスクのデフォルトの入力・出力FSをCHDFSに変更することができます。

その他の設定項目

設定項目	説明	デフォルト値	入力必須かどうか
fs.ofs.tmp.cache.dir	一時データの保存	なし	はい
fs.ofs.map.block.size	chdfsファイルシステムのblockサイズ。単位はバイト、デフォルトは128MBです（mapのセグメンテーションにのみ影響を与え、chdfs最下層のストレージロックサイズとは関係ありません）	134217728	いいえ
fs.ofs.data.transfer.thread.count	chdfsデータ転送時の並列スレッドの数	32	いいえ
fs.ofs.block.max.memory.cache.mb	chdfsプラグインによって使用されるメモリbufferのサイズ。単位はMBです。（読み取りと書き込みのアクセラレーション効果あり）	16	いいえ
fs.ofs.block.max.file.cache.mb	chdfsプラグインによって使用されるディス	256	いい

	クbufferrのサイズ。単位はMBです。（書き込みにアクセラレーション効果あり）		え
fs.ofs.prev.read.block.count	読み取り時の先行読み取りchdfs block数（chdfsの最下層blockのサイズは通常4MB）	4	いいえ
fs.ofs.plugin.info.log	プラグインのデバッグログを印刷するかどうか。ログはinfoレベルごとに印刷されます。オプション値はtrue、falseです	false	いいえ

CAMを使用してアクセス権限を付与します

最終更新日：：2024-01-19 16:58:09

CHDFSのプリセットポリシー

CHDFSのプリセット権限ポリシーは次のとおりです。

ポリシー	説明
QcloudCHDFSReadOnlyAccess	CHDFSの読み取り専用アクセス権限
QcloudCHDFSFullAccess	CHDFS権限の管理

CHDFS権限付与の操作

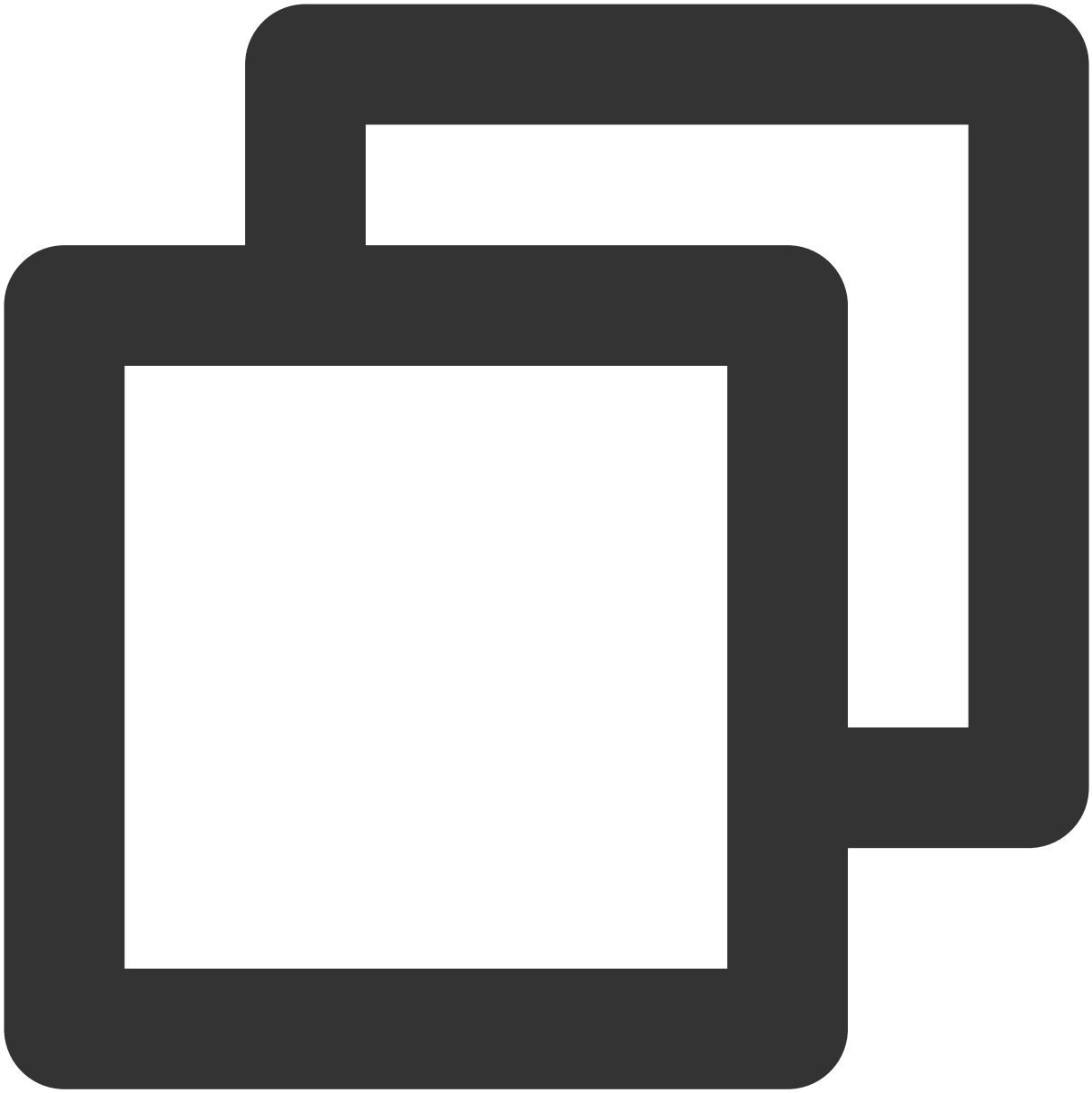
Action	Resouce	説明
chdfs:CreateFileSystem	qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/*	CHDFSの作成
chdfs:DeleteFileSystem	qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id}	CHDFSの削除
chdfs:ModifyFileSystem	qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id}	CHDFSの属性変更
chdfs:DescribeFileSystem	qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id}	CHDFSの詳細情報の確認
chdfs:DescribeFileSystems	qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id}	CHDFSリストの確認
chdfs:CreateMountPoint	qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id}	マウントポイントの作成
chdfs:DeleteMountPoint	qcs::chdfs:${region-id}:uin/${account-uin}:mountpoint/${mount-point-id}	マウントポイントの削除
chdfs:ModifyMountPoint	qcs::chdfs:${region-id}:uin/${account-uin}:mountpoint/${mount-point-id}	マウントポイントの属性変更
chdfs:DescribeMountPoint	qcs::chdfs:${region-id}:uin/${account-uin}:mountpoint/${mount-point-id}	マウントポイントの詳細情報の確認

chdfs:DescribeMountPoints	qcs::chdfs:${region-id}:uin/${account-uin}:mountpoint/${mount-point-id}	マウントポイントリストの確認
chdfs:AssociateAccessGroups	qcs::chdfs:${region-id}:uin/${account-uin}:mountpoint/${mount-point-id}	権限グループリストのバインド
chdfs:DisassociateAccessGroups	qcs::chdfs:${region-id}:uin/${account-uin}:mountpoint/${mount-point-id}	権限グループリストのバインド解除
chdfs:CreateAccessGroup	qcs::chdfs:${region-id}:uin/${account-uin}:vpc/${vpc-id}またはqcs::chdfs:${region-id}:uin/${account-uin}:unVpcId/${unVpcId}	権限グループの作成
chdfs:DeleteAccessGroup	qcs::chdfs:${region-id}:uin/${account-uin}:accessgroup/${access-group-id}	権限グループの削除
chdfs:ModifyAccessGroup	qcs::chdfs:${region-id}:uin/${account-uin}:accessgroup/${access-group-id}	権限グループの属性変更
chdfs:DescribeAccessGroup	qcs::chdfs:${region-id}:uin/${account-uin}:accessgroup/${access-group-id}	権限グループの詳細情報の確認
chdfs:DescribeAccessGroups	qcs::chdfs:${region-id}:uin/${account-uin}:accessgroup/${access-group-id}	権限グループリストの確認
chdfs:CreateAccessRules	qcs::chdfs:${region-id}:uin/${account-uin}:accessgroup/${access-group-id}	権限ルールの一括作成
chdfs:DeleteAccessRules	qcs::chdfs:${region-id}:uin/${account-uin}:accessrule/${access-rule-id}	権限ルールの一括削除
chdfs:ModifyAccessRules	qcs::chdfs:${region-id}:uin/${account-uin}:accessrule/${access-rule-id}	権限ルールの属性の一括変更
chdfs:DescribeAccessRules	qcs::chdfs:${region-id}:uin/${account-uin}:accessgroup/${access-group-id}	権限ルールリストの確認
chdfs:CreateLifeCycleRules	qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id}	ライフサイクルルールの一括作成
chdfs:DeleteLifeCycleRules	qcs::chdfs:${region-id}:uin/${account-uin}:lifecyclerule/${life-cycle-rule-id}	ライフサイクルルールの一括削除
chdfs:ModifyLifeCycleRules	qcs::chdfs:${region-id}:uin/${account-uin}:lifecyclerule/${life-cycle-rule-id}	ライフサイクルルール属性の一括変更
chdfs:DescribeLifeCycleRules	qcs::chdfs:${region-id}:uin/${account-	ライフサイクル

	uin}:filesystem/${file-system-id}	ルールリストの確認
chdfs:CreateRestoreTasks	qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id}	リストアタスクの一括作成
chdfs:DescribeRestoreTasks	qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id}	リストアタスクリストの確認
chdfs:ModifyResourceTags	qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id}	リソースタグリストの変更
chdfs:DescribeResourceTags	qcs::chdfs:${region-id}:uin/${account-uin}:filesystem/${file-system-id}	リソースタグリストの確認

CHDFS権限付与ポリシーの例

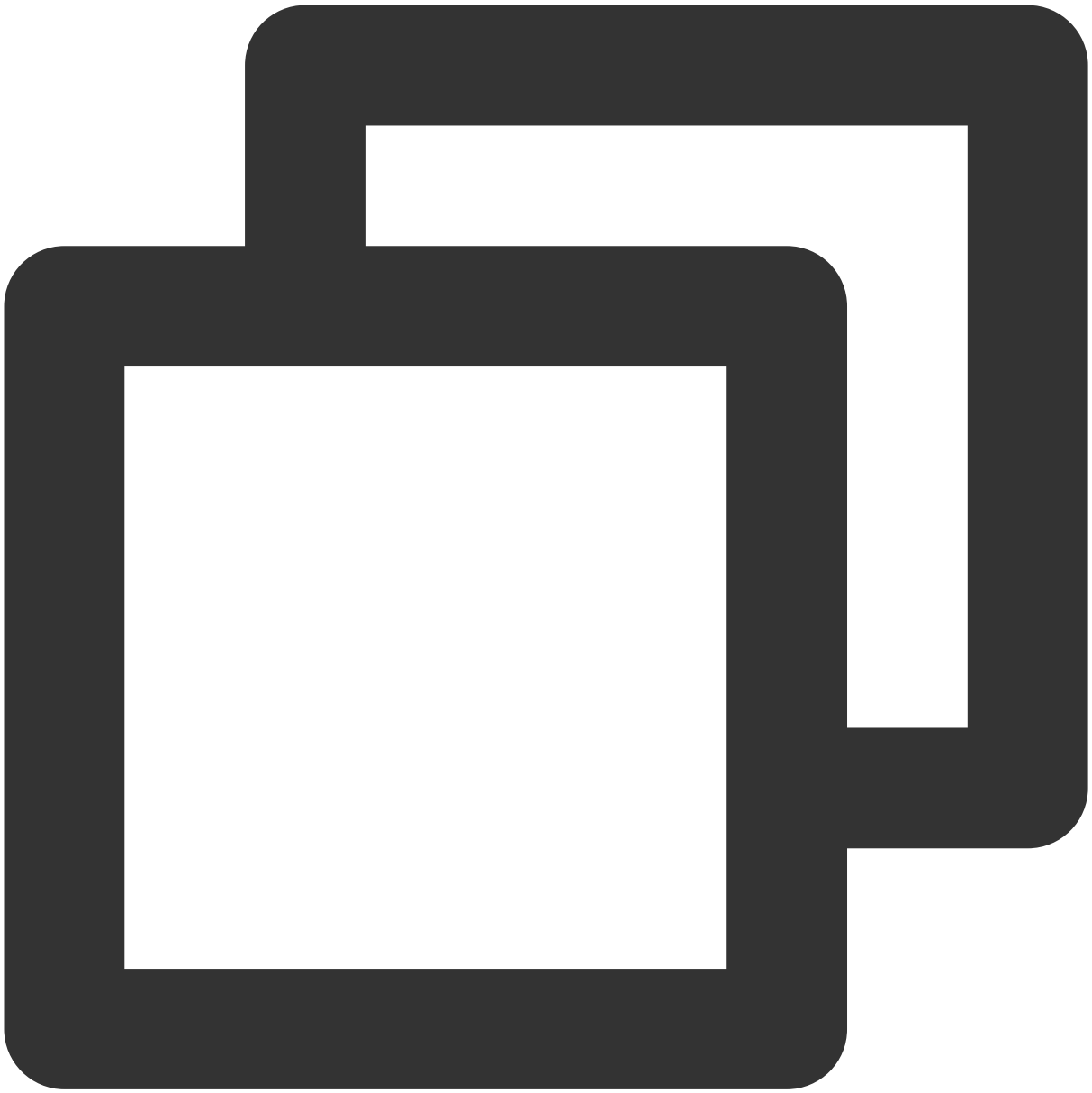
サブアカウントにCHDFS制御システムの読み取り専用アクセス権限を付与するためのポリシーの例は、次のとおりです。



```
{
    "version": "2.0",
    "statement": [{
        "effect": "allow",
        "action": [
            "name/chdfs:Describe*"
        ],
        "resource": [
                "*"
        ]
    }]
```

```
}
```

サブアカウントにCHDFSを確認する権限を付与するためのポリシーの例は、次のとおりです。



```
{
    "version": "2.0",
    "statement": [{
        "effect": "allow",
        "action": [
             "name/chdfs:DescribeFileSystem"
          ],
        "resource": [
```

```
            "qcs::chdfs::uin/ownerUin:filesystem/fileSystemId"
        ]
    }]
}
```

Javaコードを介してCHDFSにアクセスします

最終更新日：：2024-01-19 16:58:09

操作シナリオ

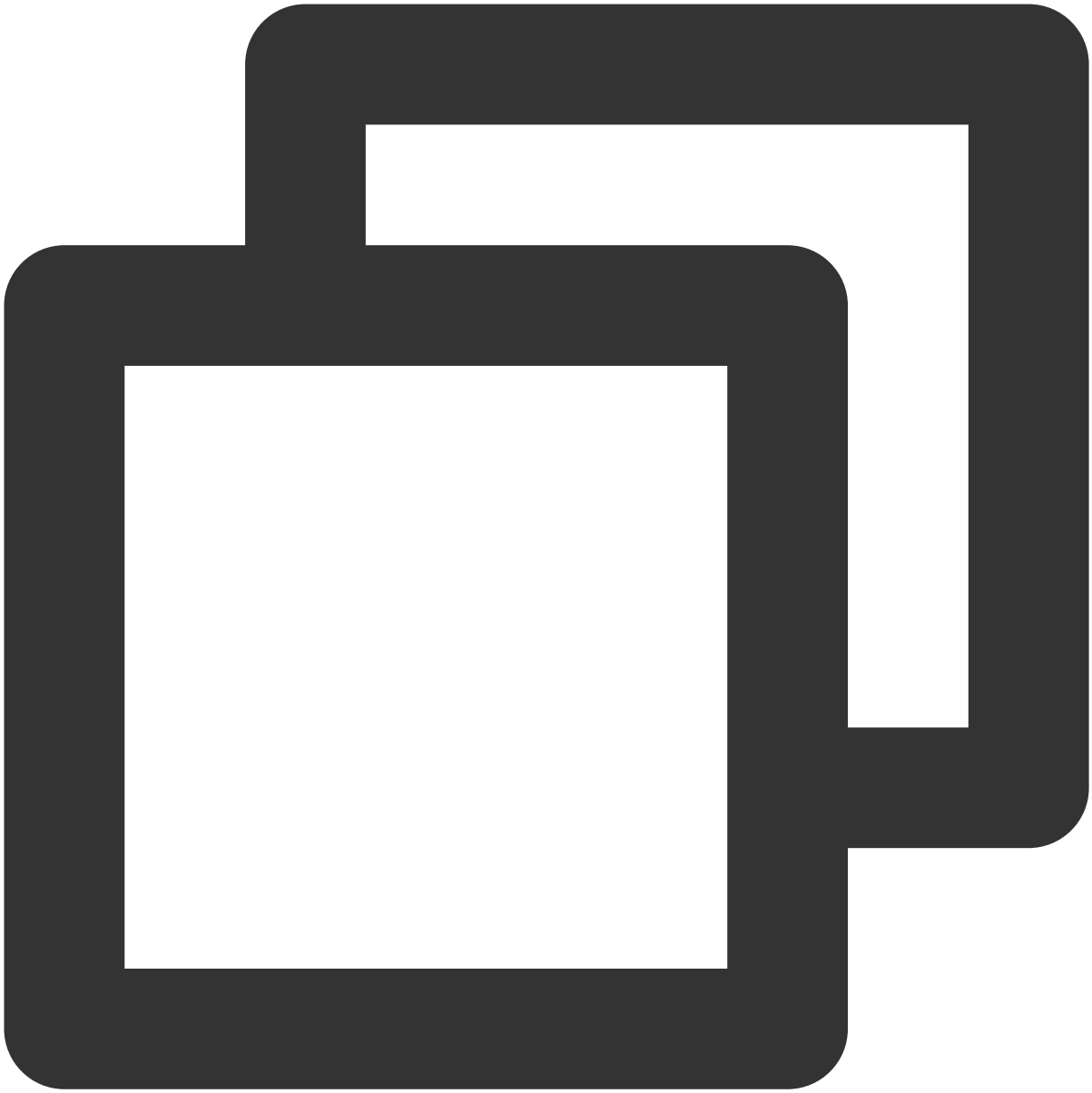
Cloud HDFS(CHDFS)のJARパッケージをデプロイすると、コマンドラインやビッグデータコンポーネントなどを使用してCHDFSを操作するだけでなく、Javaコードを介してCHDFSにアクセスすることもできます。ここでは、Javaコードを介してCHDFSにアクセスする方法についてご説明します。

前提条件

CHDFSに関連するJARパッケージがデプロイされていることを確認します。詳細については、CHDFSのマウントをご参照ください。
Javaプログラムを実行しているマシンが、マウントポイント権限グループによりアクセスを許可されているVPC内にあることを確認します。

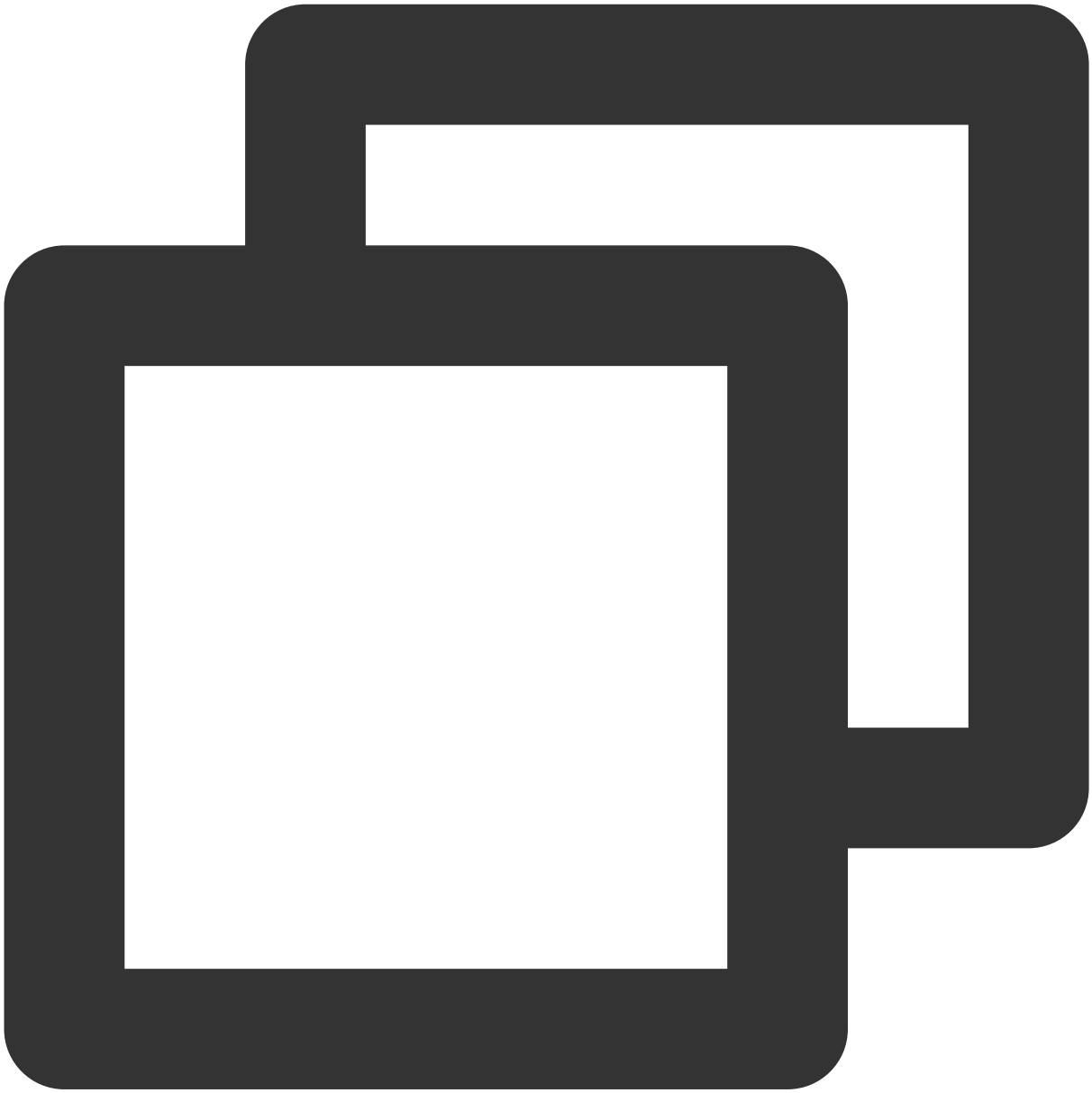
操作手順

1. mavenプロジェクトを新規作成し、mavenのpom.xmlに次の依存項目を追加します（実際のhadoop環境に応じてhadoop-commonパッケージのバージョンを設定してください）。



```
<dependencies>
     <dependency>
         <groupId>org.apache.hadoop</groupId>
         <artifactId>hadoop-common</artifactId>
         <version>2.8.5</version>
         <scope>provided</scope>
     </dependency>
</dependencies>
```

2. 以下を参照して、Hadoopを操作するためのコードを変更します。設定項目とその説明については、CHDFSのマウントをご参照ください。
以下に、一般的なファイルシステム操作インターフェースの一部を示します。その他のインターフェースについては、Hadoop FileSystemインターフェースドキュメントをご参照ください。



```
package com.qcloud.chdfs.demo;

import org.apache.commons.io.IOUtils;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
```

```
import org.apache.hadoop.fs.FileChecksum;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

import java.io.IOException;
import java.net.URI;
import java.nio.ByteBuffer;

public class Demo {
        private static FileSystem initFS() throws IOException {
            Configuration conf = new Configuration();
            // CHDFSの設定項目については、https://cloud.tencent.com/document/product/1
            // 以下の設定は、入力必須項目です

        conf.set("fs.ofs.impl", "com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter"
            conf.set("fs.AbstractFileSystem.ofs.impl", "com.qcloud.chdfs.fs.CHDFSD
            conf.set("fs.ofs.tmp.cache.dir", "/data/chdfs_tmp_cache");
            conf.set("fs.ofs.user.appid", "1250000000");
            // その他のオプションの設定項目については、https://cloud.tencent.com/document

        String chdfsUrl = "ofs://f4maaabbb-ccdd.chdfs.ap-guangzhou.myqcloud.com/";
            return FileSystem.get(URI.create(chdfsUrl), conf);
        }

    private static void mkdir(FileSystem fs, Path filePath) throws IOException {
            fs.mkdirs(filePath);
        }

    private static void createFile(FileSystem fs, Path filePath) throws IOExceptio
            // ファイルを作成します（存在する場合は上書きします）
            // if the parent dir does not exist, fs will create it!
            FSDataOutputStream out = fs.create(filePath, true);
            try {
                    // ファイルに書き込みます
                    String content = "test write file";
                    out.write(content.getBytes());
            } finally {
                    IOUtils.closeQuietly(out);
            }
        }

    private static void readFile(FileSystem fs, Path filePath) throws IOException
            FSDataInputStream in = fs.open(filePath);
            try {
                    byte[] buf = new byte[4096];
                    int readLen = -1;
```

```
                do {
                        readLen = in.read(buf);
                } while (readLen >= 0);
        } finally {
                IOUtils.closeQuietly(in);
        }
    }


    private static void queryFileOrDirStatus(FileSystem fs, Path path) throws
        FileStatus fileStatus = fs.getFileStatus(path);
        if (fileStatus.isDirectory()) {
                System.out.printf("path %s is dir\\n", path);
                return;
        }

    long fileLen = fileStatus.getLen();
        long accessTime = fileStatus.getAccessTime();
        long modifyTime = fileStatus.getModificationTime();
        String owner = fileStatus.getOwner();
        String group = fileStatus.getGroup();


        System.out.printf("path %s is file, fileLen: %d, accessTime: %d, modif
                path, fileLen, accessTime, modifyTime, owner, group);
    }


    // デフォルトのチェックタイプはCOMPOSITE-CRC32Cです
    private static void getFileCheckSum(FileSystem fs, Path path) throws IOExc
        FileChecksum checksum = fs.getFileChecksum(path);
        System.out.printf("path %s, checkSumType: %s, checkSumCrcVal: %d\\n",
                path, checksum.getAlgorithmName(), ByteBuffer.wrap(checksum.ge
    }


    private static void copyFileFromLocal(FileSystem fs, Path chdfsPath, Path
        fs.copyFromLocalFile(localPath, chdfsPath);
    }


    private static void copyFileToLocal(FileSystem fs, Path chdfsPath, Path lo
        fs.copyToLocalFile(chdfsPath, localPath);
    }


    private static void renamePath(FileSystem fs, Path oldPath, Path newPath)
```

```
        fs.rename(oldPath, newPath);
    }


    private static void listDirPath(FileSystem fs, Path dirPath) throws IOExce
        FileStatus[] dirMemberArray = fs.listStatus(dirPath);


        for (FileStatus dirMember : dirMemberArray) {
                System.out.printf("dirMember path %s, fileLen: %d\\n", dirMemb
        }
    }


    // 再帰的削除フラグは、ディレクトリを削除するために用いられます
    // 再帰がfalseで、dirが空でない場合、操作は失敗します
    private static void deleteFileOrDir(FileSystem fs, Path path, boolean recu
        fs.delete(path, recursive);
    }


    private static void closeFileSystem(FileSystem fs) throws IOException {
        fs.close();
    }


    public static void main(String[] args) throws IOException {
        // ファイルの初期化
        FileSystem fs = initFS();


        // ファイルの作成
        Path chdfsFilePath = new Path("/folder/exampleobject.txt");
        createFile(fs, chdfsFilePath);


        // ファイルの読み取り
        readFile(fs, chdfsFilePath);


        // ファイルまたはディレクトリの照会
        queryFileOrDirStatus(fs, chdfsFilePath);


        // ファイルのチェックサムの取得
        getFileCheckSum(fs, chdfsFilePath);
```

```
            // ローカルからファイルをコピーする
            Path localFilePath = new Path("file:///home/hadoop/ofs_demo/data/examp
            copyFileFromLocal(fs, chdfsFilePath, localFilePath);


            // ファイルをローカルで取得する
            Path localDownFilePath = new Path("file:///home/hadoop/ofs_demo/data/e
            copyFileToLocal(fs, chdfsFilePath, localDownFilePath);


            // リネーム
            Path newPath = new Path("/doc/example.txt");
            renamePath(fs, chdfsFilePath, newPath);


            // ファイルの削除
            deleteFileOrDir(fs, newPath, false);


            // ディレクトリの作成
            Path dirPath = new Path("/folder");
            mkdir(fs, dirPath);


            // ディレクトリにファイルを作成する
            Path subFilePath = new Path("/folder/exampleobject.txt");
            createFile(fs, subFilePath);


            // ディレクトリのリストアップ
            listDirPath(fs, dirPath);


            // ディレクトリの削除
            deleteFileOrDir(fs, dirPath, true);


            // ファイルシステムを閉じる
            closeFileSystem(fs);
        }
}
```

3. コンパイルと実行。

説明：

実行する前に、classpathが正しく設定されていることを確認してください。classpathには、Hadoop commonパッケージとCHDFSパッケージのパスが含まれる必要があります。
EMR環境の場合、CHDFSのマウントを手順に従って操作した場合、Hadoop commonパッケージは通常、`/usr/local/service/hadoop/share/hadoop/common/` ディレクトリ下にあり、CHDFSパッケージは通常、`/usr/local/service/hadoop/share/hadoop/common/lib/` ディレクトリ下にあります。

ファイルシステムの削除

最終更新日：：2024-01-19 16:58:09

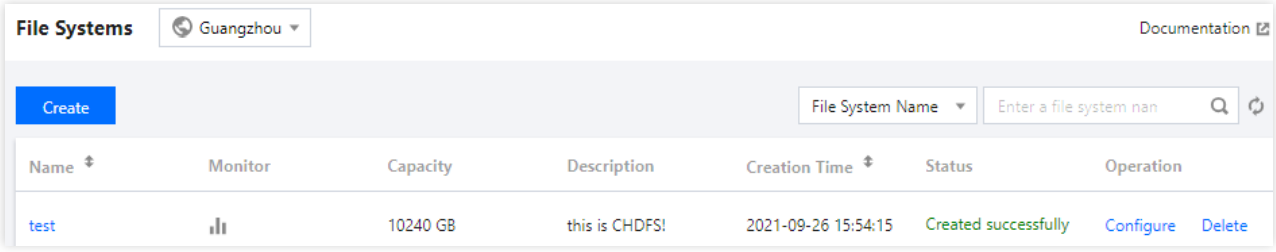
ここでは、CHDFSコンソールでファイルシステムを削除する方法について詳しくご説明します。

注意：

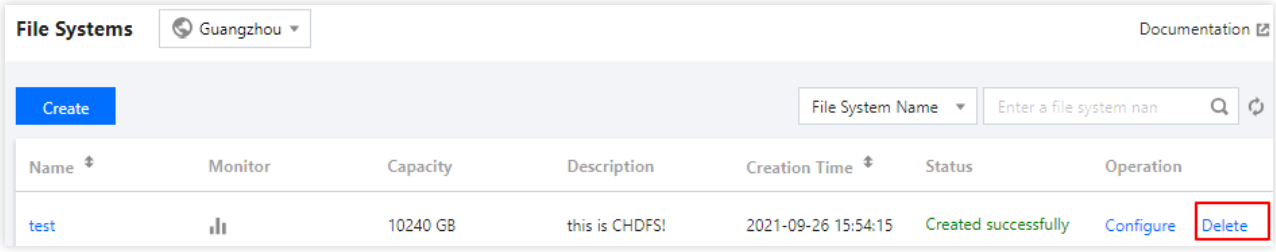
ファイルシステムを削除すると、システム内のすべての設定がクリアされ、復元できなくなりますので、慎重に操作してください。

操作手順

1. CHDFSコンソールにログインします。
2. 左側ナビゲーションバーで、【ファイルシステム】をクリックします。
3. ファイルシステムインターフェースで、広州など、ファイルシステムが配置されているリージョンを選択します。



4. ファイルシステムリストで削除する必要のあるファイルシステムを見つけ、ファイルシステム右側の【削除】をクリックします。



5. ポップアップしたウィンドウで、【削除】をクリックすると、ファイルシステムが削除できます。