Object Storage Service

ユーティリティ

ユーティリティ

ossftp

クイックインストール

概要

OSS FTP は、よく発生する FTP リクエストを受け取ったときに、ファイルやフォルダーに対する操作を OSS インスタンスにマップする特殊な FTP サーバーです。このユーティリティを使用すると、OSS インスタンスに格納されているファイルを、FTP プロトコルを使用して管理できます。

主な特徴

- **クロスプラットフォーム:** このユーティリティは、Windows、Linux、Mac の 32 ビットおよび 64 ビットの各オペレーティングシステムに対応しており、グラフィックとコマンドラインの両方のインターフェイスで使用できます。
- **インストール不要:** 展開後、そのまま実行できます。
- **設定不要:** 設定せずに実行できます。
- **透明性:** この FTP ユーティリティは Python で記述されているため、すべてのソースコードを確認できます。近いうちに GitHub でオープンソースとして入手可能にする予定です。

主な機能

- ファイルおよびフォルダーのアップロード、ダウンロード、削除などの操作
- 大きなファイルのマルチパートアップロード
- ほとんどの FTP コマンドを実行でき、日常のニーズに対応

注意

1. 現在、インストールとデプロイメントを簡素化するために、OSS FTP V1.0 は TLS 暗号化に対応していません。FTP プロトコルではプレーンテキスト転送が実装されます。**パスワードの漏洩を防ぐために、FTP サーバーとクライアントは同じマシンで実行し**、アクセスには

127.0.0.1 ポートを使用することをお勧めします。
2. このユーティリティでは、名前の変更および移動操作を実行できません。
3. インストールパッケージの extract-to パスには、漢字を使用しないでください。
4. FTP サーバーの管理コントロールページは、古い IE ブラウザーでは開けない可能性があります。
5. 対応している Python のバージョンは、Python 2.6 および Python 2.7 です。

ダウンロード

- Windows: ossftp-1.0.2-win.zip

Windows ではデフォルトで Python 2.7 がインストールされていません。そのため、インストールパッケージに含まれており、展開後、インストールや設定することなく使用できます。

- Linux/Mac: ossftp-1.0.2-linux-mac.zip

Linux および Mac システムではデフォルトで Python 2.7 または 2.6 がインストールされています。そのため、Linux および Mac 用のインストールパッケージには Python の実行可能プログラムは含まれていません。関連する依存ライブラリのみが含まれています。

実行

最初に、ダウンロードしたファイルを展開します。次に、環境に合わせて適切な実行モードを選択します。

- Windows: start.vbs をダブルクリックして実行します。
- Linux: ターミナルを起動し、実行します。

```
$ bash start.sh
```

- Mac: start.command をダブルクリックするか、ターミナルで実行します。

```
$ bash start.command
```

上記の操作によって、FTP サーバーが起動します。デフォルトでは、FTP サーバーは 127.0.0.1 でポート 2048 をリッスンします。
また、FTP サーバーのステータスを簡単に制御できるように、Web サーバーが起動します。Web サーバーは 127.0.0.1 でポート 8192 をリッスンします。
グラフィックインターフェイスを備えたシステムでは、コントロールページが自動的に開きます。

> ほとんどの場合、FTP サーバーを実行する前に設定する必要は一切ありません。設定を行う場合は、再起動しないと変更が反映されないことに注意してください。

FTP サーバーへの接続

FTP サーバーへの接続には FileZilla クライアントを使用することをお勧めします。ダウンロードしてインストールしたら、次の設定で FTP サーバーに接続します。

- ホスト: 127.0.0.1
- ログオンの種類: 通常
- ユーザ: access_key_id/bucket_name (スラッシュ (/) は、どちらか片方でなく両方が必要であることを意味します。たとえば、’ tSxyiUM3NKswPMEp/test-hz-jh-002’ とします。)
- パスワード: access_key_secret

高度な使用法

コンソールページでの FTP サーバーの管理

リスニングアドレスの変更
デフォルトのアドレス 127.0.0.1 はローカルアクセスに限定されているため、ネットワークを介して FTP サーバーにアクセスする場合はリスニングアドレスを変更する必要があります。リスニングアドレスは、イントラネット IP またはインターネット IP に変更できます。

リスニングポートの変更
FTP サーバーのリスニングポートを変更します。ポート番号が 1024 未満のポートは管理者による許可が必要であるため、1024 以上のポートを使用することをお勧めします。

ログレベルの変更
FTP サーバーのログレベルを設定します。FTP サーバーのログは、data/ossftp/ ディレクトリに出力されます。ログを表示するには、コンソールページの [Log] ボタンをクリックします。
デフォルトのログレベルは INFO で、ログに記録される情報は多くありません。詳細なログ情報が必要な場合は、レベルを DEBUG に変更します。ログに出力される情報を減らすには、レベルを WARNING または ERROR に設定します。

バケットエンドポイントの設定
FTP サーバーでは、デフォルトでバケットの場所情報を検索するため、その後のリクエストを該当するリージョン (oss-cn-hangzhou.aliyuncs.com、oss-cn-beijing.aliyuncs.com など) に送信することができます。最初にイントラネットを介して OSS インスタンスへのアクセスを試みます。
たとえば、’ test-bucket-a.oss-cn-hangzhou.aliyuncs.com’ というバケットエンドポイントを設定した場合、test-bucket-a にアクセスすると、’ oss-cn-hangzhou.aliyuncs.com’ という名前のドメインに接続されます。

注意 : 変更を反映するには、システムを再起動する必要があります。
上記の変更はすべて、実際には ftp ディレクトリの config.json ファイルに対する変更です。したがって、このファイルを直接変更することもできます。

FTP サーバーの直接起動 (Linux/Mac)直接 ossftp ディレクトリの ftpserver.py ファイルを起動すると、Web サーバーのオーバーヘッドを回避できます。

```
$ python ossftp/ftpserver.py &
```

設定の変更方法は、上記と同様です。

考えられる問題

FTP サーバーへの接続時にエラーが発生した場合

次の 2 つの原因が考えられます。

入力された access_key_id または access_key_secret に誤りがある。解決策: 正しい情報を入力して、もう一度試します。

使用されている access_key 情報が RAM サブアカウントの access_key であり、そのサブアカウントにバケットを一覧表示する権限がない。

解決策: サブアカウントを使用する場合は、コンソールページでバケットエンドポイントを指定して、特定のバケットにアクセスするときに使用するエンドポイントを FTP サーバーに伝えます。また、サブアカウントには必要な権限を設定する必要があります。
RAM を使用してアクセス制御を実装し、OSS にアクセスする方法については、「RAM」を参照してください。権限の詳細は、次のとおりです。

読み取り専用: OSS-FTP には[‘ListObjects’、’ GetObject’、’ HeadObject’] の各権限が必要です。**読み取り専用**権限を指定して RAM サブアカウントを作成する方法については、グラフィックチュートリアル「ファイル共有のために RAM を統合する方法」を参照してください。

RAM サブアカウントに**ファイルのアップロード**を許可する場合は、[‘PutObject’] 権限を割り当てます。

RAM サブアカウントに**ファイルの削除**を許可する場合は、[‘DeleteObject’] 権限を割り当てます。

Linux で FTP サーバーを実行している場合、FileZilla を使用してそのサーバーに接続すると、次のエラーが発生することがあります。

501 パスをデコードできません (サーバーファイルシステムのエンコーディングは ANSI_X3.4-1968 です)

これは通常、ローカルの中国語のコードでエラーが発生したために発生します。start.sh を実行するターミナルで次のコマンドを入力します。その後、プログラムを再起動します。

```
$ export LC_ALL=en_US.UTF-8; export LANG="en_US.UTF-8"; locale
```

Discuz でリモート添付ファイルを OSS インスタンスに格納する方法

序文

Web サイトのリモート添付ファイル機能は、アップロードされた添付ファイルを、FTP を介してリモートストレージサーバー (通常はリモート FTP サーバー) に直接格納する機能です。

現在は、Discuz フォーラム、PHPWind フォーラム、および WordPress Web サイトでリモート添付ファイル機能を利用できます。

このドキュメントでは、リモート添付ファイルを Discuz ベースのフォーラムから格納する方法を説明します。

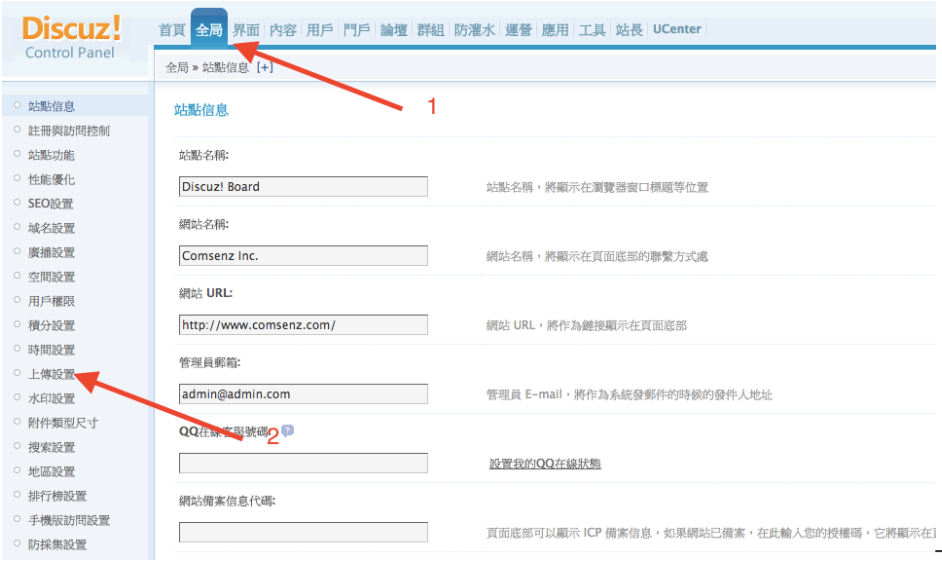
準備

OSS アカウントを申請し、**公開読み取り**バケットを作成します。匿名アクセスを許可する必要があるため、権限を公開読み取りに設定する必要があります。

手順

ここで使用する Discuz のバージョンは **Discuz! X3.1** です。詳しい設定プロセスは次のとおりです。

- Discuz の Web サイトにログオンし、管理インターフェイスに移動します。[**Global**] をクリックし、[**Upload Settings**] をクリックします。



—>

- [**Remote Attachments**] を選択し、機能を設定します。

  [Enable remote attachment] を [**Yes**] に設定します。

  [Enable SSL connection] を [**No**] に設定します。

  [FTP Server Address] を設定します。これは、OSS-FTP を実行するアドレスです。通常は “**127.0.0.1** “ です。

  [FTP service port No.] をデフォルトの “**2048** “ に設定します。

  [FTP Account] を **AccessKeyID/BucketName** という形式で指定します。” **/** “ は “**または** “ の意味ではありません。

  [FTP Password] を **AccessKeySecret** に設定します。

  [Passive Mode Connection] をデフォルトの [**Yes**] に設定します。

  [Remote Attachment Directory] を “**.** “ に設定します。これは、アップロード用のディレクトリをバケットのルートディレクトリの下に作成することを意味します。

  [Remote URL] を “**http://BucketName.Endpoint** “ に設定します。

ここでは、杭州リージョンのバケット test-hz-jh-002 をテストします。したがって、「http://test-hz-jh-

002.oss-cn-hangzhou.aliyuncs.com」と入力します。**ここで、BucketName はエンドポイントと一致している必要があります。**

> タイムアウト時間を 0 に設定します。これは、サービスのデフォルトの設定を使用することを意味します。
>
> 設定が完了したら、[Test Remote Attachment] をクリックします。テストが成功すると、次のような情報ボックスが表示されます。

- 検証

それでは、フォーラムに投稿をパブリッシュして機能をテストしてみましょう。 任意のボードで投稿を作成し、その投稿の添付ファイルとして画像をアップロードします。

画像を右クリックし、[Open image in new tab] を選択します。

ブラウザーで、画像の URL が http://test-hz-jh-002.oss-cn-hangzhou.aliyuncs.com/forum/201512/18/171012mzvkku2z3na2w2wa.png となっていることを確認できます。これは、画像が OSS の test-hz-jh-002 にアップロードされたことを示しています。

PHPWind でリモート添付ファイルを OSS インスタンスに格納する方法

序文

Web サイトのリモート添付ファイル機能は、アップロードされた添付ファイルを、FTP を介してリモートストレージサーバー (通常はリモート FTP サーバー) に直接格納する機能です。

現在は、Discuz フォーラム、PHPWind フォーラム、および WordPress Web サイトでリモート添付ファイル機能を利用できます。

このドキュメントでは、リモート添付ファイルを PHPWind ベースのフォーラムから格納する方法を説明します。

準備

OSS アカウントを申請し、**公開読み取り**バケットを作成します。匿名アクセスを許可する必要があるため、権限を公開読み取りに設定する必要があります。

手順

ここで使用する PHPWind は **PHPWind 8.7** です。設定プロセスは次のとおりです。

Web サイトにログオンします。

管理インターフェイスに移動し、[**Global**]、[**Upload Settings**]、[**Remote Attachments**] の順に選択します。

機能を設定します。

i. [**Enable FTP uploads**] を [**Yes**] に設定します。
ii. [**Website Attachment Address**] を “http://bucket-name.endpoint“ に設定します。ここでは、杭州リージョンのバケット test-hz-jh-002 をテストします。したがって、「http://test-hz-jh-002.oss-cn-hangzhou.aliyuncs.com」と入力します。**ここで、BucketName はエンドポイントと一致している必要があります。**
iii. FTP サーバーのアドレスを設定します。これは、OSS-FTP を実行するアドレスです。通常は **127.0.0.1** です。
iv. [FTP service port No.] をデフォルトの “**2048** “ に設定します。
v. [Remote attachment directory] を “**.** “ に設定します。これは、アップロード用のディレクトリをバケットのルートディレクトリの下に作成することを意味します。
vi. [FTP Account] を **AccessKeyID/BucketName** という形式で指定します。” **/** “ は “**または** “ の意味ではありません。
vii. [FTP Password] を **AccessKeySecret** に設定します。AccessKeyID と AccessKeySecret を取得するには、Alibaba Cloud コンソールにログオンし、Access Key 管理に移動します。
viii. FTP のタイムアウト時間を設定します。” 10” に設定した場合、10 秒以内にリクエストに対する応答が受信されないと、タイムアウト応答が送信されます。

検証

PHPWind では、テストボタンをクリックして直接機能をテストすることはできません。したがって、機能を検証するには、画像を含む投稿をパブリッシュする必要があります。

画像を右クリックし、[Open image in new tab] を選択します。新しいタブに画像が表示されます。

画像の URL は、この画像が OSS のバケット test-hz-jh-002 にアップロードされたことを示しています。

WordPress でリモート添付ファイルを OSS イ

ンスタンスに格納する方法

序文

Web サイトのリモート添付ファイル機能は、アップロードされた添付ファイルを、FTP を介してリモートストレージサーバー (通常はリモート FTP サーバー) に直接格納する機能です。

現在は、Discuz フォーラム、PHPWind フォーラム、および WordPress Web サイトでリモート添付ファイル機能を利用できます。

このドキュメントでは、リモート添付ファイルを WordPress ベースのフォーラムから格納する方法を説明します。

準備

OSS アカウントを申請し、**公開読み取り**バケットを作成します。匿名アクセスを許可する必要があるため、権限を公開読み取りに設定する必要があります。

手順

WordPress はこの機能にもともと対応していませんが、サードパーティのプラグインを使用してリモート添付ファイル機能を実装できます。ここで使用する WordPress は WordPress **4.3.1** で、プラグインは **Hacklog Remote Attachment** です。具体的な設定プロセスは次のとおりです。

1. WordPress の Web サイトにログオンし、[Install Plug-in] を選択します。キーワード “FTP” を検索し、**Hacklog Remote Attachment** のインストールを選択します。

   設定します。

   i. FTP サーバーのアドレスを設定します。これは、OSS-FTP を実行するアドレスです。通常は **127.0.0.1** です。
   ii. [FTP service port No.] をデフォルトの “**2048** “ に設定します。
   iii. [FTP Account] を **AccessKeyID/BucketName** という形式で指定します。” **/** “ は “**または** “ の意味ではありません。
   iv. [FTP Password] を **AccessKeySecret** に設定します。

   > AccessKeyID と AccessKeySecret を取得するには、Alibaba Cloud コンソールにログオンし、Access Key 管理に移動します。

   v. FTP タイムアウトをデフォルト値の 30 秒に設定します。
   vi. [Remote Basic URL] を http://BucketName.Endpoint/wp に設定します。ここでは、杭州リージョンのバケット test-hz-jh-002 をテストします。したがって、「http://test-hz-jh-002.oss-cn-hangzhou.aliyuncs.com/wp」と入力します。

vii. “FTP Remote Path” を設定します。「wp」と入力します。これは、すべての添付ファイルをバケットの wp ディレクトリに保存することを意味します。このフィールドは [Remote Basic URL] フィールドからの相対パスであることに注意してください。
viii. [HTTP Remote Path] を “.” に設定します。

検証します。
設定が完了したら、[Save] をクリックします。テストが自動的に始まります。テスト結果はページの一番上に表示されます。

新しい記事を投稿し、画像を挿入します。
これで、新しい記事を作成してリモート添付ファイル機能をテストできます。記事を作成したら、[Add Media] をクリックして添付ファイルをアップロードします。

添付ファイルがアップロードされたら、[Post] をクリックして記事を表示します。

画像を右クリックし、[Open image in new tab] をクリックして画像の URL を表示します。

画像の URL は、この画像が OSS に正常にアップロードされたことを示しています。

ファイル共有のために RAM を統合する方法

概要

このドキュメントでは、RAM サービスを統合してユーザーバケット内のファイルおよびフォルダーを共有する方法を説明します。バケットのオーナーはオブジェクトを編集できますが、それ以外のユーザーは読み取り専用権限しか持ちません。

プロセス: RAM の有効化 -> 読み取り専用権限付与ポリシーの作成 -> サブアカウントの作成 -> サブアカウントへの権限の付与 -> FTP ログオンの検証

アカウント ID の取得

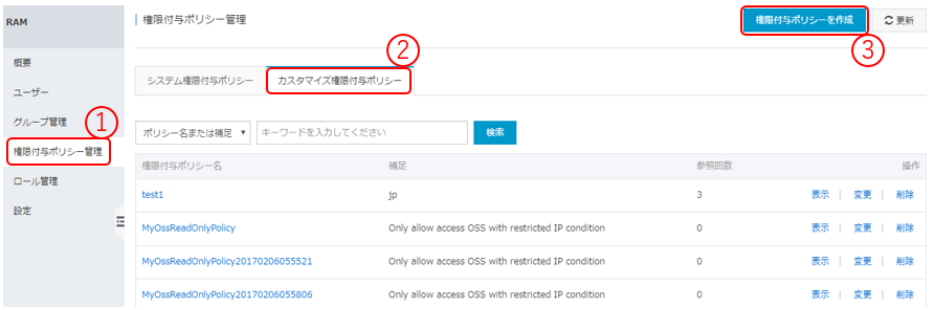
次の図のように、アカウント ID を取得します。

RAM の有効化

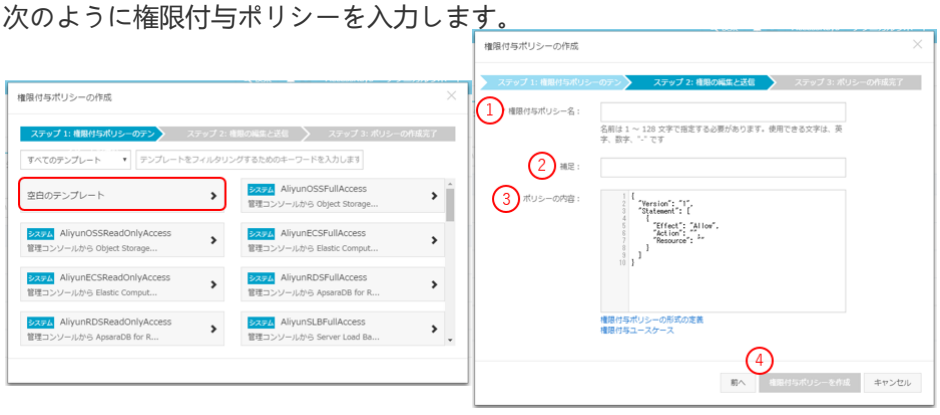
Resource Access Management (RAM) は、リソースアクセスを制御するための Alibaba Cloud のサービスです。ポリシーを作成することによって、共有読み取りアカウントを作成できます。ユーザーはこのアカウントを使用して FTP ツールにログオンし、ファイルを読むことができます。

権限付与ポリシーの作成

RAM を有効化したら、RAM コンソールに移動し、左側の [権限付与ポリシー管理] をクリックします。下の図に示した手順に従って、新しい権限付与ポリシーを作成します。



次のように権限付与ポリシーを入力します。



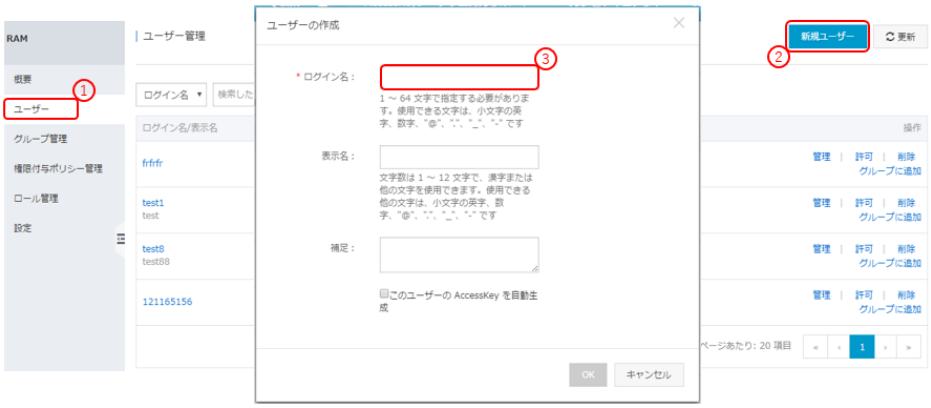
ポリシー名と、必要に応じて補足 (フィールド 1 および 2) を指定します。フィールド 3 の [ポリシーの内容] によってポリシーが決定されます。

```
{
"Version": "1",
"Statement": [
{
"Action": [
"oss:GetObject",
"oss:HeadObject"
],
"Resource": [
"acs:oss:*:**********:test-hz-john-001/*"
],
"Effect": "Allow"
},
{
"Action": [
"oss:ListObjects",
"oss:GetBucketAcl",
"oss:GetBucketLocation"
],
"Resource": [
"acs:oss:*:**********:test-hz-john-001"
],
"Effect": "Allow"
},
{
"Action": [
"oss:ListBuckets"
],
"Resource": [
"acs:oss:*:**********:*"
],
"Effect": "Allow"
}
]
}
```

上の例では、********** を自分のアカウント ID に、test-hz-john-001 を自分のバケット名にそれぞれ置き換えます。次に、すべての内容をコピーし、[ポリシーの内容] に貼り付けます。最後に、[権限付与ポリシーを作成] をクリックします。

アカウントの作成

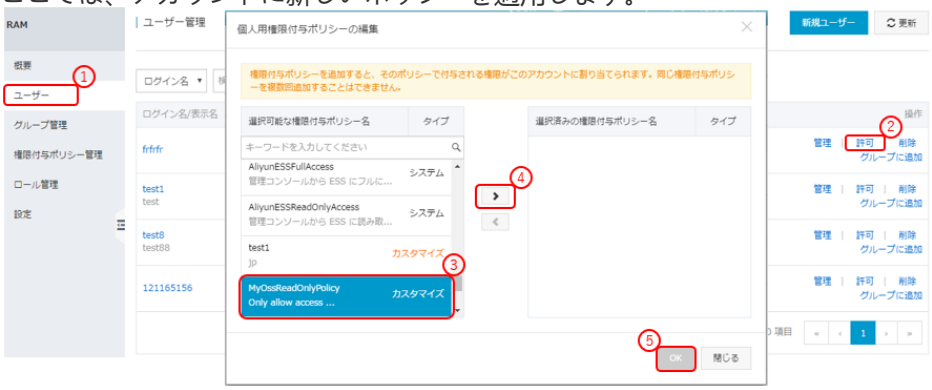
上記の権限付与ポリシーからは、読み取り専用ポリシーが作成されます。ここでは、アカウントを作成し、そのアカウントにこのポリシーを適用します。次の手順に従ってアカウントを作成します。



新しいアカウントの access_key を覚えておいてください。

アカウントへの権限付与

ここでは、アカウントに新しいポリシーを適用します。



サブアカウントでのログオン

サブアカウントの access_key と権限付与ポリシーのバケットを使用してログオンします。これで、ファイルとフォルダーをダウンロードすることができます。ただし、アップロード操作は失敗します。

ossfs

クイックインストール

概要

ossfs を使用すると、Linux システムで Alibaba Cloud OSS バケットをローカルファイルにマウントできます。バケットをマウントしたシステムでは、ローカルファイルシステムを使用して OSS オブジェクトに対する操作をすばやく実行して、データ共有を実現できます。

機能

ossfs は S3FS を基盤として構築されており、S3FS のすべての機能が組み込まれています。主な機能は次のとおりです。

- POSIX ファイルシステムの機能のほとんどに対応。具体的には、ファイルの読み書き、ディレクトリ、リンク操作、権限、UID/GID、拡張属性など。
- OSS マルチパート機能を使用した、大きなファイルのアップロード。
- データの完全性を確保するための MD5 チェックサム。

制限

ossfs によって提供される機能とパフォーマンスには、ローカルファイルシステムと比較して一定の制限があります。具体的には、次のとおりです。

- ランダムな書き込みおよび追加書き込み操作では、ファイル全体が上書きされます。
- ディレクトリの一覧表示などのメタデータ操作は、システムが OSS サーバーにリモートアクセスする必要があるため、パフォーマンスが低下します。
- ファイルおよびフォルダーの名前を変更する操作は、アトミックではありません。
- 複数のクライアントが 1 つの OSS バケットにアタッチされている場合は、各クライアントの動作を手動で調整する必要があります。たとえば、複数のクライアントが同じファイルに対して書き込みを行わないようにする必要があります。
- ハードリンクには対応していません。
- このシステムは、システム負荷が非常に大きくなる高並列の読み書きシナリオには適していません。

インストールと使用

インストールパッケージのダウンロード

Linux のバージョン	以下をダウンロード
Ubuntu 16.04 (x64)	ossfs_1.80.2_ubuntu16.04_amd64.deb
Ubuntu 14.04 (x64)	ossfs_1.80.2_ubuntu14.04_amd64.deb
CentOS 7.0 (x64)	ossfs_1.80.2_centos7.0_x86_64.rpm
CentOS 6.5 (x64)	ossfs_1.80.2_centos6.5_x86_64.rpm

ossfs のインストール

- Ubuntu でインストールするには、次のコマンドを実行します。

```
sudo apt-get update
sudo apt-get install gdebi-core
sudo gdebi your_ossfs_package
```

- CentOS 6.5 以上でインストールするには、次のコマンドを実行します。

```
sudo yum localinstall your_ossfs_package
```

- CentOS 5 でインストールするには、次のコマンドを実行します。

```
sudo yum localinstall your_ossfs_package --nogpgcheck
```

ossfs の使用

バケット名と AccessKeyId/Secret を設定し、/etc/passwd-ossfs ファイルに保存します。このファイルに対する権限を正しく設定する必要があることに注意してください。640 に設定することをお勧めします。

```
echo my-bucket:my-access-key-id:my-access-key-secret > /etc/passwd-ossfs
chmod 640 /etc/passwd-ossfs
```

指定されたディレクトリに OSS バケットをマウントします。

```
ossfs my-bucket my-mount-point -ourl=my-oss-endpoint
```

例

/tmp/ossfs ディレクトリに my-bucket バケットをマウントします。AccessKeyId は faint です。AccessKeySecret は 123 で、OSS エンドポイントは http://oss-cn-hangzhou.aliyuncs.com です。

```
echo my-bucket:faint:123 > /etc/passwd-ossfs
chmod 640 /etc/passwd-ossfs
mkdir /tmp/ossfs
ossfs my-bucket /tmp/ossfs -ourl=http://oss-cn-hangzhou.aliyuncs.com
```

バケットをマウント解除します。

```
fusermount -u /tmp/ossfs
```

詳細については、https://github.com/aliyun/ossfs#ossfs を参照してください。

制限

ossfs によって提供される機能とパフォーマンスには、ローカルファイルシステムと比較して一定の制限があります。具体的には、次のとおりです。

- ランダムな書き込みおよび追加書き込み操作では、ファイル全体が上書きされます。
- ディレクトリの一覧表示などのメタデータ操作は、システムが OSS サーバーにリモートアクセスする必要があるため、パフォーマンスが低下します。
- ファイルおよびフォルダーの名前を変更する操作は、アトミックではありません。
- 複数のクライアントが 1 つの OSS バケットにアタッチされている場合は、各クライアントの動作を手動で調整する必要があります。たとえば、複数のクライアントが同じファイルに対して書き込みを行わないようにする必要があります。
- ハードリンクには対応していません。
- このシステムは、システム負荷が非常に大きくなる高並列の読み書きシナリオには適していません。

リリースログ

https://github.com/aliyun/ossfs/blob/master/ChangeLog を参照してください。

よくある質問

- Q: ossfs はどのようなプログラムに適していますか。
    - ossfs では、OSS バケットがローカルでマウントされます。OSS に対応していないプログラムでデータを OSS に自動的に同期する場合、ossfs は優れた選択肢です。
- Q: ossfs にはどのような制限がありますか。
    - ネットワーク経由でデータをクラウドに同期する必要があるため、ossfs のパフォーマンスおよび機能はローカルのファイルシステムと異なることがあります。マウントした ossfs ディスクに対する I/O 操作が頻繁に行われるデータベースなどのアプリケーションを実行する場合は、この点を慎重に検討する必要があります。ossfs は、次の点でローカルのファイルシステムと異なります。
        - ランダムな書き込みおよび追加書き込み操作では、ファイル全体が上書きされます。
        - ディレクトリの一覧表示などのメタデータ操作は、システムが OSS サーバーにリモートアクセスする必要があるため、パフォーマンスが低下します。
        - ファイルおよびフォルダーの名前を変更する操作は、アトミックではありません。
        - 複数のクライアントが 1 つの OSS バケットにアタッチされている場合は、各ク

ライアントの動作を手動で調整する必要があります。たとえば、複数のクライアントが同じファイルに対して書き込みを行わないようにする必要があります。

- ハードリンクには対応していません。

- Q: ossfs を利用するためには Alibaba Cloud ホストを使用する必要がありますか。
  - ossfs は、Alibaba Cloud イントラネットと組み合わせて使用する必要はありません。外部のインターネットホストでも使用できます。
- Q: ossfs では同時に複数の OSS バケットをマウントできますか。
  - はい。passwd-ossfs ファイルに複数の OSS 設定情報エントリを書き込むだけです。異なる OSS アカウントのバケットをマウントできます。
- Q: バケットをマウントする際に、” ossfs: unable to access MOUNTPOINT /tmp/ossfs: Transport endpoint is not connected” というエラーが表示されるのはなぜですか。
  - まず、問題のディレクトリに対して umount コマンドを実行します。
  - ossfs でマウントする場合は、入力した URL パラメーターが正しいことと、バケット、アクセスキー ID、アクセスキーシークレットの組み合わせが正しいことを確認してください。
  - URL にはバケット名を含めないでください。たとえば、バケットのドメイン名が OSS コンソールに ossfs-test-1.oss-cn-hangzhou.aliyuncs.com と表示される場合、URL は http://oss-cn-hangzhou.aliyuncs.com に設定します。
- Q: ossfs で “ossfs: unable to access MOUNTPOINT /tmp/odat: No such file or directory” と表示されるのはなぜですか。
  - このエラーは、ディレクトリが未作成の場合に発生します。マウントする前にディレクトリを作成する必要があります。
- Q: ローカルでバケットをマウントし、そのディレクトリに対して ls コマンドを実行すると “operation not permitted” エラーが発生するのはなぜですか。
  - そのバケットで、不可視文字を含む OSS オブジェクトの名前がディレクトリ名に含まれていないかどうかを確認してください。ファイルシステムには、ファイルおよびディレクトリの名前に関して厳格な制限があります。ディレクトリ名がこの制限に違反している場合に、このエラーが発生します。別のツールを使用してそのようなオブジェクトの名前を変更してから、ls コマンドを実行すると、ディレクトリの内容が正しく表示されます。
- Q: ossfs をマウントする際、どのように権限を設定すればよいですか。
  - マウントしたフォルダーへのアクセスを他のユーザーに許可する場合は、ossfs の実行時に allow_other パラメーターを次のように指定します。
    - ossfs your_bucket your_mount_point -ourl=your_endpoint -o allow_other
  - 別のユーザーが所有するフォルダー (/tmp/ossfs) のマウントを許可する場合は、そのユーザーとしてフォルダーを作成してマウントし、OSSFS を使用する必要があります。
    - sudo -u user mkdir /tmp/ossfs
    - sudo -u user ossfs bucket-name /tmp/ossfs
- Q: デバイスの起動時に ossfs が自動的にマウントされるようにするには、どうすればよいですか。

  - ステップ 1: バケット名、アクセスキー ID/シークレットなどの情報を /etc/passwd-ossfs に記述し、このファイルの権限を 640 に変更します。
    - echo your_bucket_name:your_access_key_id:your_access_key_secret >

/etc/passwd-ossfs
- chmod 640 /etc/passwd-ossfs

ステップ 2: 適切な設定を行います (設定方法はシステムのバージョンによって異なります)。

- ステップ 2A: fstab メソッドを使用して、自動的に ossfs をマウントします (Ubuntu 14.04 および CentOS 6.5 の場合)。
  - /etc/fstab に次のコマンドを追加します。
  - ossfs#your_bucket_name your_mount_point fuse _netdev,url=your_url,allow_other 0 0
  - 上のコマンドで、’ your_xxx’ はそれぞれ実際のバケット名などの情報に置き換えます。
  - /etc/fstab ファイルを保存します。mount -a コマンドを実行します。エラーが報告されなければ、設定は正常です。
  - これで、Ubuntu 14.04 で ossfs が自動的にマウントされます。CentOS 6.5 では、次のコマンドも実行します。
  - chkconfig netfs on
- ステップ 2B: 起動スクリプトを使用して ossfs をマウントします (CentOS 7.0 以上の場合)。
  - /etc/init.d/ ディレクトリにファイル ossfs を作成します。テンプレートファイルの内容を新しいファイルにコピーします。ここで、’ your_xxx’ は実際の情報に置き換えてください。
  - コマンド chmod a+x /etc/init.d/ossfs を実行します。
  - 上のコマンドにより、新しい ossfs スクリプトに実行権限が付与されます。これで、このスクリプトを実行できます。スクリプトの内容にエラーがなければ、指定されたディレクトリに OSS バケットがマウントされます。
  - コマンド chkconfig ossfs on を実行します。
  - このコマンドにより、ossfs 起動スクリプトは別のサービスとして設定されるため、デバイスの起動時に自動的に開始されます。
  - これで、ossfs は起動時に自動的にマウントされます。上記をまとめると、Ubuntu 14.04 または CentOS 6.5 を使用している場合はステップ 1 と 2A、CentOS 7.0 を使用している場合はステップ 1 と 2B を実行します。

Q: www ユーザーを使用して ossfs をマウントする必要があります。この場合は、どのようにして自動マウントを設定すればよいですか。

上の質問への回答を参照してください。まず、ステップ 1 を実行します。次に、/etc/init.d/ossfs ファイルのコマンドを次のように変更して、ステップ 2B を実行します。

sudo -u www ossfs your_bucket your_mountpoint -ourl=your_url

- sudo を使用した /etc/sudoers の編集が許可されるように、起動スクリプトを設定します。Defaults requiretty の行を #Defaults requiretty に変更します (この行をコメントアウトします)。

- Q: fusermount: failed to open current directory: Permission denied エラーを解決するには、どうすればよいですか。
    - これは、fuse のバグ です。このエラーにより、現在のユーザーは現在のディレクトリ (マウントされていないディレクトリ) に対して読み取り権限を持っていることが求められます。この問題を解決するには、cd コマンドを実行して読み取り権限を持っているディレクトリに移動し、再度 ossfs コマンドを実行します。

osscmd

クイックインストール

環境要件

Python SDK を使用するには、Python 対応の環境が必要です。 Python のバージョン: Version 2.5 ~ Version 2.7。 SDK は Windows および Linux で使用できますが、Python3.0 には SDK Version 2.x との完全な互換性がないため、SDK では Python3.0 以上がサポートされていません。

Python をインストールした後で以下を行います。

- Linux のシェルで「python」と入力して Enter キーを押し、Python のバージョンを確認します。次に例を示します。

```
Python 2.5.4 (r254:67916, Mar 10 2010, 22:43:17)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-46)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Windows のコマンドプロンプトで「python」と入力して Enter キーを押し、Python のバージョンを確認します。次に例を示します。

```
C:\Documents and Settings\Administrator>python
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

上のように表示された場合、Python は正常にインストールされています。

- 例外が発生する場合もあります。たとえば、Windows のコマンドプロンプトで「python」と入力

すると、内部コマンドまたは外部コマンドとして認識されないことを示すエラーが表示されることがあります。このような場合は、” 環境変数” - “Path” の設定を確認して Python のインストールパスを追加してください。

Python がインストールされていない場合は、Python の公式 Web サイトから入手できます。 この Web サイトでは Python のインストールと使用についての詳細な手順とガイダンスが提供されています。

インストールと使用方法

[こちら]をクリックしてファイルをダウンロードします。ダウンロードした Python SDK を osscmd のディレクトリに展開し、” python osscmd + 操作 ” を実行します。 たとえば、次のようにオブジェクトをバケットにアップロードします。

```
python  osscmd  put  myfile.txt  oss://mybucket
```

osscmd では、oss://bucket または oss://bucket/object を使用してバケットまたはオブジェクトを示します。oss:// はリソースを示す方法であって、それ以外の意味はありません。

詳細なコマンドリストが必要な場合は、次のように入力します: python osscmd

詳細なパラメーターリストの説明が必要な場合は、次のように入力します: python osscmd help

使用例

osscmd のインストールと設定

Linux または Windows で SDK インストーラをダウンロードした後、ダウンロードしたパケットを展開すると、osscmd の使用を開始できます。

使用方法を確認するには、python osscmd を直接起動します。 すべてのコマンドには 2 つの実行モードがあります。 たとえば、ユーザーが作成したバケットを照会するとします。 gs (“get service” の短縮形) コマンドを実行します。

- 方法 1: ID や Key を指定しません。この場合、ID と Key は osscmd によってデフォルトファイルから読み取られます。

```
$ python osscmd gs
can't get accessid/accesskey, setup use : config --id=accessid --key=accesskey
```

注意: このようなプロンプトが表示された場合は、ID と Key が正しく設定されていないことを示します。ステップ 2 の設定コマンドを参照してください。

ID と Key が正しく設定されており有効であれば、コマンドを実行します。

```
$ python osscmd gs
2013-07-19 08:11 test-oss-sample
Bucket Number is: 1
```

- 方法 2: コマンドに ID と Key を指定します。この場合、ID と Key は、osscmd によってコマンドラインから読み取られます。 ID と Key が有効であれば、コマンドを実行します。次のような結果が表示されます。

```
$ python osscmd gs --id=your_id --key=your_key
2013-07-19 08:11 test-oss-sample
Bucket Number is: 1
```

ユーザーの ID と Key をデフォルトファイルに設定するには、次のコマンドを実行します。 デフォルトの oss ホストは oss.aliyuncs.com です。

```
$python osscmd config --id=YOUR_ID --key=YOUR_KEY
```

“設定が保存されました” のようなプロンプトが表示された場合は、ID と Key が正常に保存されたことを示します。

基本操作

作成されたバケットの一覧表示

```
$python osscmd getallbucket
```

OSS ユーザーがバケットを 1 つも作成していない場合、出力は空になります。

バケットの作成

mybucketname という名前のバケットを作成します。

```
$python osscmd createbucket mybucketname
```

“mybucketname” という名前のバケットを作成できないことがあります。これは、OSS 内のバケットの名前はグローバルに一意である必要があり、このバケットが既に別のユーザーによって作成されている可能性があるからです。 その場合は、名前を変更する必要があります。たとえば、バケット名に特定の日付を追加することができます。

バケットが正常に作成されたかどうかを確認します。

```
$python osscmd getallbucket
```

失敗した場合は、返されたエラーメッセージを確認します。

オブジェクトの表示

バケットが正常に作成された後、バケット内のオブジェクトを確認します。

```
$python osscmd list oss://mybucketname/
```

バケット内にオブジェクトがないため、出力は空になります。

オブジェクトのアップロード

オブジェクトをバケットにアップロードします。 ローカルファイルの名前が local_existed_file であれば、MD5 の値は次のとおりです。

```
$ md5sum local_existed_file 7625e1adc3a4b129763d580ca0a78e44 local_existed_file
$ python osscmd put local_existed_file oss://mybucketname/test_object
```

オブジェクトの再確認

正常に作成された場合は、バケット内のオブジェクトをもう一度確認します。

```
$python osscmd list oss://mybucketname/
```

オブジェクトのダウンロード

バケットからローカルにオブジェクトをダウンロードし、ダウンロードしたファイルの md5 値と比較します。

```
$ python osscmd get oss://mybucketname/test_object download_file
$ md5sum download_file
7625e1adc3a4b129763d580ca0a78e44 download_file
```

オブジェクトの削除

```
$ python osscmd delete oss://mybucketname/test_object
```

バケットの削除注意: バケット内にオブジェクトがある場合、バケットは削除できません。

```
$ python osscmd deletebucket test-oss-aliyun-com
```

使用のライフサイクル

xml テキストファイルをライフサイクル用に設定

```
```

```
<LifecycleConfiguration>
<Rule>
<ID>1125</ID>
<Prefix>log_backup/</Prefix>
<Status>Enabled</Status>
<Expiration>
<Days>2</Days>
</Expiration>
</Rule>
</LifecycleConfiguration>
```

この例は、バケット内の、log_backup/ というプレフィックスのある、現在の時刻より 2 日以上前のオブジェクトを削除する操作を示しています。 詳細なルール設定については、「API リファレンス」を参照してください。

書き込みのライフサイクル

```
python osscmd putlifecycle oss://mybucket lifecycle.xml
0.150(s) elapsed
```

読み取りのライフサイクル

```
python osscmd getlifecycle oss://mybucket
<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration>
<Rule>
<ID>1125</ID>
<Prefix>log_backup/</Prefix>
<Status>Enabled</Status>
<Expiration>
<Days>2</Days>
</Expiration>
</Rule>
</LifecycleConfiguration>

0.027(s) elapsed
```

削除のライフサイクル

```
python osscmd deletelifecycle oss://mybucket
0.139(s) elapsed
```

読み取りのライフサイクル

```
python osscmd getlifecycle oss://mybucket
Error Headers:

[('content-length', '288'), ('server', 'AliyunOSS'), ('connection', 'close'), ('x-oss-request-id',
'54C74FEE5D7F6B24E5042630'), ('date', 'Tue, 27 Jan 2015 08:44:30 GMT'), ('content-type', 'application/xml')]
Error Body:
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
<BucketName>mybucket</BucketName>
<Code>NoSuchLifecycle</Code>
<Message>No Row found in Lifecycle Table.</Message>
<RequestId>54C74FEE5D7F6B24E5042630</RequestId>
<HostId>mybucket.oss-maque-hz-a.alibaba.net</HostId>
</Error>

Error Status:

404
getlifecycle Failed!
```

Anti-leech 設定

空のリファラーのアクセスを許可する

```
$osscmd putreferer oss://test --allow_empty_referer=true
0.004(s) elapsed
```

設定済みのリファラーを取得する

```
$osscmd getreferer oss://test
<?xml version="1.0" encoding="UTF-8"?>
<RefererConfiguration>
<AllowEmptyReferer>true</AllowEmptyReferer>
<RefererList />
</RefererConfiguration>
```

空のリファラーを許可せず、 test リファラーのリクエストのみを許可する

```
$osscmd putreferer oss://test --allow_empty_referer=false --referer='www.test.com'
0.092(s) elapsed
```

設定済みのリファラーを取得する

```
$osscmd getreferer oss://test
<?xml version="1.0" encoding="UTF-8"?>
<RefererConfiguration>
<AllowEmptyReferer>false</AllowEmptyReferer>
<RefererList>
<Referer>www.test.com</Referer>
</RefererList>
</RefererConfiguration>
```

空のリファラーを許可せず、 test および test1 リファラーのリクエストのみを許可する

```
$osscmd putreferer oss://test --allow_empty_referer=false --referer='www.test.com,www.test1.com'
```

設定済みのリファラーを取得する

```
$osscmd getreferer oss://test
<?xml version="1.0" encoding="UTF-8"?>
<RefererConfiguration>
<AllowEmptyReferer>false</AllowEmptyReferer>
<RefererList>
<Referer>www.test.com</Referer>
<Referer>www.test1.com</Referer>
</RefererList>
</RefererConfiguration>
```

バケットコマンド

config

コマンドの説明:

config --id=[accessid] --key=[accesskey] --host=[host] --sts_token=[sts_token]

osscmd のデフォルトのホスト、ID、および Key を設定します。 デフォルトのホストは oss.aliyuncs.com です。oss-internal.aliyuncs.com にアクセスするには、—host=oss-internal.aliyuncs.com を追加します。sts_token パラメーターは必須ではありません。sts_token が指定されると、STS メソッドで認証が実行されます。

例:

- python osscmd config --id=your_id --key=your_key
- python osscmd config --id=your_id --key=your_key --host=oss-internal.aliyuncs.com

getallbucket(gs)

コマンドの説明:

getallbucket(gs)

ユーザーによって作成されたバケットを表示します。 gs は、get service の短縮形です。 gs では、getallbucket と同じ結果が得られます。

例:

- python osscmd getallbucket

- python osscmd gs

createbucket(cb,mb,pb)

コマンドの説明:

createbucket(cb,mb,pb) oss://bucket --acl=[acl]

バケットの作成コマンドです。cb は create bucket の短縮形、mb は make bucket の短縮形、pb は put bucket の短縮形であり、oss://bucket はバケットを示します。 —acl パラメーターを含めることができますが、必須ではありません。 いくつか、同じ結果を得ることのできるコマンドがあります。

例:

- python osscmd createbucket oss://mybucket
- python osscmd cb oss://myfirstbucket --acl=public-read
- python osscmd mb oss://mysecondbucket --acl=private
- python osscmd pb oss://mythirdbucket

deletebucket(db)

コマンドの説明:

deletebucket(db) oss://bucket

バケットの削除コマンドです。db は delete bucket の短縮形です。 Deletebucket では、db と同じ結果が得られます。

例:

- python osscmd deletebucket oss://mybucket
- python osscmd db oss://myfirstbucket

deletewholebucket

注意:このコマンドは大きなリスクを伴います。すべてのデータが消去され、消去されたデータは復元できないからです。 このコマンドは慎重に使用してください。**コマンドの説明:**

deletewholebucket oss://bucket

バケット、その中のオブジェクト、およびマルチパートコンテンツを削除します。

例:

- python osscmd deletewholebucket oss://mybucket

getacl

コマンドの説明:

getacl oss://bucket

バケットに対するアクセスおよび制御権限を取得します。

例:

- python osscmd getacl oss://mybucket

setacl

コマンドの説明:

setacl oss://bucket --acl=[acl]

バケットに対するアクセスおよび制御権限を変更します。acl には、private、public-read、public-read-write のうち、いずれか 1 つのみを指定できます。

例:

- python osscmd setacl oss://mybucket --acl=private

putlifecycle

コマンドの説明:

putlifecycle oss://mybucket lifecycle.xml

ライフサイクルルールを設定します。ライフサイクルの設定ファイルは lifecycle.xml です。詳細なルール設定については、「API リファレンス」を参照してください。

例:

- python osscmd putlifecycle oss://mybucket lifecycle.xml

lifecycle.xml には、ライフサイクルの設定ルールが記述されています。 例:

```
<LifecycleConfiguration>
<Rule>
<ID>1125</ID>
<Prefix>log_backup/</Prefix>
<Status>Enabled</Status>
<Expiration>
<Days>2</Days>
</Expiration>
</Rule>
</LifecycleConfiguration>
```

getlifecycle

コマンドの説明:

osscmd getlifecycle oss://bucket

バケットのライフサイクルルールを取得します。

例:

- python osscmd getlifecycle oss://mybucket

deletelifecycle

コマンドの説明:

osscmd deletelifecycle oss://bucket

バケット内にあるライフサイクルルールをすべて削除します。

例:

- python osscmd deletelifecycle oss://mybucket

putreferer

コマンドの説明:

osscmd putreferer oss://bucket --allow_empty_referer=[true|false] --referer=[referer]

Anti-leech ルールを設定します。 allow_empty_referer パラメーターは必須であり、null の指定を許可するかどうかを設定するために使用します。 referer パラメーターは、アクセスが許可されているホワイトリストを設定するために使用します。たとえば、”www.test1.com,www.test2.com” のように指定します (“,” は区切り文字)。 詳細なルール設定については、プロダクトのドキュメントを参照してください。

例:

- python osscmd putreferer oss://mybucket --allow_empty_referer=true --referer="www.test1.com,www.test2.com"

getreferer

コマンドの説明:

osscmd getreferer oss://bucket

バケットの Anti-leech ルールを取得します。

例:

- python osscmd getreferer oss://mybucket

putlogging

コマンドの説明:

osscmd putlogging oss://source_bucket oss://target_bucket/[prefix]

source_bucket はログ対象のバケットを示し、target_bucket はログの保存先を示します。 ソースバケット内で生成されるログファイル用に、プレフィックスを設定できます。これにより、カテゴリ別のクエリが簡単になります。

例:

- python osscmd putlogging oss://mybucket oss://myloggingbucket/mb

getlogging

コマンドの説明:

osscmd getlogging oss://bucket

バケットのログルールを取得し、xml ファイルを返します。

例:

- python osscmd getlogging oss://mybucket

オブジェクトコマンド

ls(list)

コマンドの説明:

ls(list) oss://bucket/[prefix] [marker] [delimiter] [maxkeys]

バケット内のオブジェクトを一覧表示します。

例:

- python osscmd ls oss://mybucket/folder1/folder2
- python osscmd ls oss://mybucket/folder1/folder2 maker1
- python osscmd ls oss://mybucket/folder1/folder2 maker1 /
- python osscmd ls oss://mybucket/
- python osscmd list oss://mybucket/ "" "" 100

コマンドの説明:

ls(list) oss://bucket/[prefix] --marker=xxx --delimiter=xxx --maxkeys=xxx

バケット内のオブジェクトを一覧表示します。

例:

- python osscmd ls oss://mybucket/folder1/folder2 --delimiter=/
- python osscmd ls oss://mybucket/folder1/folder2 --maker=a
- python osscmd ls oss://mybucket/folder1/folder2 --maxkeys=10

mkdir

コマンドの説明:

mkdir oss://bucket/dirname

“/ “ で終わる、サイズが 0 のオブジェクトを作成します。

例:

- python osscmd mkdir oss://mybucket/folder

listallobject

コマンドの説明:

listallobject oss://bucket/[prefix]

バケット内のすべてのオブジェクトを表示します。プレフィックスを指定することもできます。

例:

- python osscmd listallobject oss://mybucket
- python osscmd listallobject oss://mybucket/testfolder/

deleteallobject

コマンドの説明:

deleteallobject oss://bucket/[prefix]

バケット内のすべてのオブジェクトを削除します。プレフィックスを指定することもできます。

例:

- python osscmd deleteallobject oss://mybucket
- python osscmd deleteallobject oss://mybucket/testfolder/

downloadallobject

コマンドの説明:

downloadallobject oss://bucket/[prefix] localdir --replace=false --thread_num=5

ディレクトリ構造を変更せずに、バケット内のオブジェクトをローカルディレクトリにダウンロードします。 プレフィックスを指定してダウンロードすることもできます。 —replace=false を指定すると、ダウンロード時に同じ名前のローカルファイルが既に存在していても置き換わりません。 —replace=true を指定すると、同じ名前のローカルファイルが置き換えられます。 ダウンロード時のスレッド化を設定するには、thread_num を使用します。

例:

- python osscmd downloadallobject oss://mybucket /tmp/folder
- python osscmd downloadallobject oss://mybucket /tmp/folder --replace=false
- python osscmd downloadallobject oss://mybucket /tmp/folder --replace=true --thread_num=5

downloadtodir

コマンドの説明:

downloadallobject oss://bucket/[prefix] localdir --replace=false

ディレクトリ構造を変更せずに、バケット内のオブジェクトをローカルディレクトリにダウンロードします。プレフィックスを指定してダウンロードすることもできます。—replace=false を指定すると、ダウンロード時に同じ名前のローカルファイルが既に存在していても置き換わりません。—replace=true を指定すると、同じ名前のローカルファイルが置き換えられます。このコマンドでは、downloadallobject と同じ結果が得られます。

例:

- python osscmd downloadtodir oss://mybucket /tmp/folder
- python osscmd downloadtodir oss://mybucket /tmp/folder --replace=false
- python osscmd downloadtodir oss://mybucket /tmp/folder --replace=true

uploadfromdir

コマンドの説明:

uploadfromdir localdir oss://bucket/[prefix] --check_point=check_point_file --replace=false --check_md5=false --thread_num=5

ローカルファイルをバケットにアップロードします。 たとえば、localdir には /tmp/ のように指定します。

3 つのファイル a/b、a/c、および a がある場合、これらを OSS にアップロードすると oss://bucket/a/b、oss://bucket/a/c、および oss://bucket/a となります。 prefix として mytest と指定した場合、OSS にアップロードしたファイルは oss://bucket/mytest/a/b、oss://bucket/mytest/a/c、および oss://bucket/mytest/a となります。

--check_point=check_point_file は、指定されたファイルです。 ファイルを指定すると、アップロードされ

たローカルファイルが check_point_file にタイムスタンプとして格納されます。uploadfromdir コマンドは、アップロード対象ファイルのタイムスタンプと check_point_file に記録されたタイムスタンプを比較します。 タイムスタンプが違っていた場合は、そのファイルが再度アップロードされます。それ以外の場合、そのファイルはスキップされます。 デフォルトでは、check_point_file は存在しません。 --replace=false を指定すると、ダウンロード時に同じ名前のローカルファイルが既に存在していても置き換わりません。—replace=true を指定すると、同じ名前のローカルファイルが置き換えられます。--check_md5=false は、ファイルのアップロード中に Content-MD5 リクエストヘッダーの検証が行われないことを示します。true は、Content-MD5 リクエストヘッダーの検証が行われることを示します。

> 注意: check_point_file には、アップロードされたすべてのファイルに関するログが記録されます。 アップロードするファイルが多すぎる場合、check_point_file のサイズは変更することができます。

例:

- python osscmd uploadfromdir /mytemp/folder oss://mybucket
- python osscmd uploadfromdir /mytemp/folder oss://mybucket --check_point_file=/tmp/mytemp_record.txt
- python osscmd uploadfromdir C:\Documents and Settings\User\My Documents\Downloads oss://mybucket --check_point_file=C:\cp.txt

put

コマンドの説明:

put localfile oss://bucket/object --content-type=[content_type] --headers="key1:value1#key2:value2" --check_md5=false

ローカルファイルをバケットにアップロードする際には、オブジェクトの content-type を指定することも、カスタマイズされたヘッダーを指定することもできます。 --check_md5=false は、ファイルのアップロード中に Content-MD5 リクエストヘッダーの検証が行われないことを示します。true は、Content-MD5 リクエストヘッダーの検証が行われることを示します。

例:

- python osscmd put myfile.txt oss://mybucket
- python osscmd put myfile.txt oss://mybucket/myobject.txt
- python osscmd put myfile.txt oss://mybucket/test.txt --content-type=plain/text --headers="x-oss-meta-des:test#x-oss-meta-location:CN"
- python osscmd put myfile.txt oss://mybucket/test.txt --content-type=plain/text

upload

コマンドの説明:

upload localfile oss://bucket/object --content-type=[content_type] --check_md5=false

ローカルファイルをオブジェクトグループとしてアップロードします。 推奨されていません。 --check_md5=false は、ファイルのアップロード中に Content-MD5 リクエストヘッダーの検証が行われないことを示します。true は、Content-MD5 リクエストヘッダーの検証が行われることを示します。

例:

- python osscmd upload myfile.txt oss://mybucket/test.txt --content-type=plain/text

get

コマンドの説明:

get oss://bucket/object localfile

オブジェクトをローカルにダウンロードします。

例:

- python osscmd get oss://mybucket/myobject /tmp/localfile

multiget(multi_get)

コマンドの説明:

multiget(multi_get) oss://bucket/object localfile --thread_num=5

オブジェクトをマルチスレッドでローカルにダウンロードします。 スレッド数は設定できます。

例:

- python osscmd multiget oss://mybucket/myobject /tmp/localfile
- python osscmd multi_get oss://mybucket/myobject /tmp/localfile

cat

コマンドの説明:

cat oss://bucket/object

オブジェクトのコンテンツを読み取り、直接出力します。 オブジェクトのコンテンツのサイズが大きい場合は使用しないでください。

例:

- python osscmd cat oss://mybucket/myobject

meta

コマンドの説明:

meta oss://bucket/object

オブジェクトのメタ情報を読み取り、出力します。 メタ情報には、content-type、ファイルの長さ、カスタムメタなどが含まれています。

例:

- python osscmd meta oss://mybucket/myobject

copy

コマンドの説明:

copy oss://source_bucket/source_object oss://target_bucket/target_object --headers="key1:value1#key2:value2"

ソースバケット内のソースオブジェクトをコピー先バケット内のコピー先オブジェクトにコピーします。

例:

- python osscmd copy oss://bucket1/object1 oss://bucket2/object2

rm(delete,del)

コマンドの説明:

rm(delete,del) oss://bucket/object

オブジェクトを削除します。

例:

- python osscmd rm oss://mybucket/myobject
- python osscmd delete oss://mybucket/myobject
- python osscmd del oss://mybucket/myobject

signurl(sign)

コマンドの説明:

signurl(sign) oss://bucket/object --timeout=[timeout_seconds]

署名を含む URL を生成し、タイムアウト値を指定します。 このコマンドを適用できるのは、プライベートバケット内の指定されたオブジェクトに他のユーザーがアクセスできる場合です。

例:

- python osscmd sign oss://mybucket/myobject

- python osscmd signurl oss://mybucket/myobject

マルチパートコマンド

init

コマンドの説明:

init oss://bucket/object

アップロード ID を開始および生成します。 アップロード ID は、multiupload コマンドと組み合わせて使用できます。

例:

- python osscmd init oss://mybucket/myobject

listpart

コマンドの説明:

listpart oss://bucket/object --upload_id=xxx

指定されたオブジェクトについて、アップロード ID に対応するアップロード済みパートを表示します。 関連する概念については、OSS API リファレンス参照してください。 アップロード ID を指定する必要があります。

例:

- python osscmd listpart oss://mybucket/myobject --upload_id=
  75835E389EA648C0B93571B6A46023F3

listparts

コマンドの説明:

listparts oss://bucket

バケット内の未完了のマルチパートアップロード ID とオブジェクトを表示します。 バケットを削除する必要があり、バケットが空でないというメッセージが表示された場合は、このコマンドを使用して、マルチパートコンテンツが存在するかどうかを確認できます。

例:

- python osscmd listparts oss://mybucket

getallpartsize

コマンドの説明:

getallpartsize oss://bucket

バケットに含まれる既存のアップロード ID に対応するパートの合計サイズを表示します。

例:

- python osscmd getallpartsize oss://mybucket

cancel

コマンドの説明:

cancel oss://bucket/object --upload_id=xxx

アップロード ID に対応するマルチパートアップロードイベントを終了します。

例:

- python osscmd cancel oss://mybucket/myobject --upload_id=
  D9D278DB6F8845E9AFE797DD235DC576

multiupload(multi_upload,mp)

コマンドの説明:

multiupload(multi_upload,mp) localfile oss://bucket/object --check_md5=false --thread_num=10

ローカルファイルをマルチパートで OSS にアップロードします。

例:

- python osscmd multiupload /tmp/localfile.txt oss://mybucket/object
- python osscmd multiup_load /tmp/localfile.txt oss://mybucket/object
- python osscmd mp /tmp/localfile.txt oss://mybucket/object

コマンドの説明:

multiupload(multi_upload,mp) localfile oss://bucket/object --upload_id=xxx --thread_num=10 --max_part_num=1000 --check_md5=false

ローカルファイルをマルチパートで OSS にアップロードします。ローカルファイルのパート数は、max_part_num で定義します。 このコマンドではまず、アップロード ID の対応するパートの ETag がローカルファイルの MD5 値に一致するかどうかの判断が行われます。一致した場合、アップロードはスキップされます。 そのため、使用前に生成されたアップロード ID はパラメーターとして組み込まれます。 アップロードが失敗しても、multiupload コマンドを繰り返してアップロードを再開できます。 --

check_md5=false は、ファイルのアップロード中に Content-MD5 リクエストヘッダーの検証が行われないことを示します。true は、Content-MD5 リクエストヘッダーの検証が行われることを示します。

例:

- python osscmd multiupload /tmp/localfile.txt oss://mybucket/object --upload_id=D9D278DB6F8845E9AFE797DD235DC576
- python osscmd multiup_load /tmp/localfile.txt oss://mybucket/object --thread_num=5
- python osscmd mp /tmp/localfile.txt oss://mybucket/object --max_part_num=100

copylargefile

コマンドの説明:

copylargefile oss://source_bucket/source_object oss://target_bucket/target_object --part_size=10*1024*1024 --upload_id=xxx

1 G を超える大きなファイルをコピーする場合、オブジェクトをマルチパートで目的の場所にコピーすることができます (ソースバケットとコピー先バケットは同じリージョン内にある必要があります)。 upload_id はオプションのパラメーターです。マルチパートコピーイベントの転送を再開する必要がある場合は、upload_id を示すことができます。 part_size は、パートサイズを定義するために使用します。1 つのパートの最小サイズは 100 KB で、最大 10,000 個のパートがサポートされます。 part_size に設定されている値が OSS の制限値と競合する場合は、パートサイズがアプリケーションによって自動的に調整されます。

例:

- python osscmd copylargefile oss://source_bucket/source_object oss://target_bucket/target_object --part_size=10*1024*1024

uploadpartfromfile (upff)

コマンドの説明:

uploadpartfromfile (upff) localfile oss://bucket/object --upload_id=xxx --part_number=xxx

このコマンドは主に、テスト目的で使用されます。実際の使用はお勧めしません。

uploadpartfromstring(upfs)

コマンドの説明:

uploadpartfromstring(upfs) oss://bucket/object --upload_id=xxx --part_number=xxx --data=xxx

このコマンドは主に、テスト目的で使用されます。実際の使用はお勧めしません。

ossprobe

概要

ossprobe は OSS のアクセス検出ツールです。アップロードやダウンロードプロセス中に、ネットワークエラーや基本パラメーターの誤った設定が原因で発生する問題のトラブルシューティングに使用されます。データをアップロードまたはダウンロードする間にエラーが発生した場合、ossprobe が考えられる原因を表示するため、エラーの特定に時間がかかりません。

バージョン

バージョン: 1.0.0

主な機能

- ネットワーク環境が正常かどうかをチェック
- パラメーターが正しいかどうかをチェック
- アップロードおよびダウンロードの速度をテスト

プラットフォーム

- Linux
- Windows
- Mac

ソフトウェアのダウンロード

- windows64 ossprobe
- linux64 ossprobe
- mac ossprobe

ダウンロードの問題の検出

使用方法

```
    ossprobe --download  [-i AccessKeyId] [-k AccessKeySecret] [-p EndPoint] [-b BucketName] [-o ObjectName] [-t LocalPath]
[-f Url] [-a Address]
```

```
-f --from ObjectのUrl
-i --id AccessKeyId
-k --key AccessKeySecret
-p --endpoint EndPoint
-b --bucket BucketName
-o --object ObjectName
-t --to ダウンロードしたコンテンツのパスを保存します。デフォルトでは、現在のディレクトリの一時ファイルのパスです
。
-a --addr 検出用のネットワークアドレス。デフォルトのアドレスは www.aliyun.com です。プライベートクラウドを使用
している場合は、プライベートクラウド内のアクセス可能なアドレスを選択してください。

ヒント：-f パラメーターが指定されている場合、ダウンロードに URL を使用できます。-f パラメーターが指定されていな
い場合は、AccessKeyID、AccessKeySecret、EndPoint、BucketName パラメーターを設定する必要があります。
```

例

URL ベースのダウンロードが正常かどうかをチェックするには (URL の取得方法、次のコマンドを実行します。

方法	コマンド
指定した URL からダウンロードします。	ossprobe --download -f Url
指定した URL からダウンロードして、コンテンツを指定したファイルに保存します。	ossprobe --download -f Url -t tmp/example.txt
指定した URL からダウンロードして、指定したアドレスのネットワークの状態を検出します。	ossprobe --download -f Url -a Addr

指定したパラメーター (AccessKeyID、AccessKeySecret、EndPoint、BucketName) を使用して正常にダウンロードできるかどうかをチェックするには次のコマンドを実行します。

方法	コマンド
ランダムなファイルをダウンロードします。	ossprobe --download -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName
指定したファイルをダウンロードします。	ossprobe --download -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -o ObjectName
指定したファイルをダウンロードして、コンテンツを指定したローカルファイルに保存します。	ossprobe --download -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -o ObjectName -t tmp/example.txt
ランダムなファイルをダウンロードして、指定したアドレスのネットワークの状態を検出します。	ossprobe --download -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -a Addr

ヒント：

- ダウンロードしたファイルはバイナリ形式の実行プログラムなので、Linux システムで「chmod +x ossprobe」を実行し、ossprobe を実行できる権限を追加する必要があります。
- デフォルトでは、-t パラメーターが現在のディレクトリの一時ファイルへのパスを示します (ファ

イル名の形式は、ossfilestore20160315060101 です)。
- -t パラメーターがディレクトリを示す場合、一時ファイルはそのディレクトリに生成され、データが保存されます (ファイル名の形式は、ossfilestore20160315060101 です)。
- ファイルを URL からダウンロードした場合、そのファイルの名前は、URL のスラッシュ “/ “ の後の末尾の文字列になります。たとえば、URL が http://aliyun.com/a.jpg の場合、ファイルは a.jpg という名前で保存されます。

アップロードの問題の検出

使用方法

```
ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint  -b BucketName [-m normal|append|multipart] [-s UploadFilePath] [-o ObjectName] [-a Addr]

-i --id AccessKeyID
-k --key AccessKeySecret
-p --endpoint EndPoint
-b --bucket BucketName
-s --src アップロードするファイルのパス。デフォルトでは、ローカルの一時ファイルのパスです。
-m --mode ファイルのアップロードモード。デフォルトは通常のアップロードです。
-o --object アップロードされるオブジェクトの名前。-s が null ではない場合、オブジェクトの名前はデフォルトでアップロードされるファイルの名 前になります。-s が null の場合、オブジェクトの名前はデフォルトで tem で始まる一時ファイルの名前になります。
-a --addr 検出用のネットワークアドレス。デフォルトのアドレスは、Alibaba Cloud Web サイトのアドレスです。プライベートクラウドを使用している場合は、プライベートクラウド内のアクセス可能なアドレスを選択してください。
```

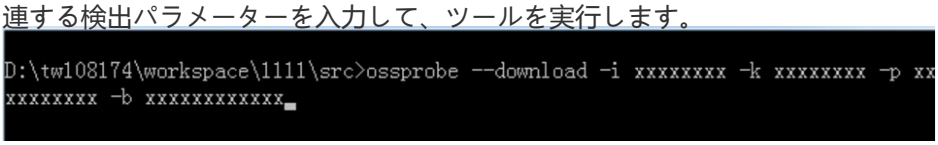
例

方法	コマンド
一時ファイルを生成し、通常モードでアップロードします。	ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName
一時ファイルを生成し、追加モードでアップロードします。	ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -o ObjectName -m append
一時ファイルを生成し、マルチパートモードでアップロードします。	ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -o ObjectName -m multipart
マルチパートモードで指定したコンテンツをアップロードします。	ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -o ObjectName -m multipart -s src
マルチパートモードで指定したコンテンツをアップロードし、オブジェクト名を指定します。	ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -m multipart -s src -o example.txt
一時ファイルを生成し、通常モードでアップロードして、指定したアドレスのネットワークの状態を検出します。	ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -a Addr

ヒント: ランダムに生成されるファイルの名前は、ossuploadtmp で始まります。

プラットフォームによる違い

- Windows
  Win + R キーを押して、[ファイル名を指定して実行] ダイアログボックスを表示し、「cmd」と入力し、Enter キーを押します。コマンドラインインターフェイス (CLI) でツールのパスを入力し、関連する検出パラメーターを入力して、ツールを実行します。



- Linux および Mac
  ターミナルを開きます。表示されたインターフェイスでツールのパスを入力し、関連するパラメーターを入力して、ツールを実行します。

レポートデータの表示

コマンドを実行した後に、logOssProbe20060102150405.txt という名前のレポートが生成されます (logOssProbe に続く数字は、レポートが生成された日付を示します)。コマンドラインモードに考えられるエラーの原因が表示されますが、 エラーメッセージから原因を特定できない場合は、レポートの参照が可能です。問題が解決しない場合は、検出レポートにアタッチされているチケットを起票することができます。

コンソールの表示

コンソールでは、以下の主要な情報が表示されます。

- 実行した後に、× とマークされたステップは失敗し、× とマークされていないステップは成功しています。
- アップロードまたはダウンロード操作が成功したかどうかを示す結果。アップロードまたはダウンロードが成功した場合、コンソールにファイルサイズとアップロード/ダウンロードの時間が表示されます。
- [Suggested Change] 列には、エラーの原因と変更提案が表示されます。
- OSS エラーコードに詳しい場合は、OSS で返されたエラーメッセージを基にトラブルシューティングを行うことができます。
- [Log Info] 列には、ログの名前とアドレスが表示されるため、ログを見つけることができます。

(ヒント: エラーが検出されても、変更提案が表示されないことがあります。この場合、OSSエラーコードを参照して、返されたエラーコードを基にトラブルシューティングを行ってください。)

ログファイル

コンソール表示と異なり、ログファイルにはネットワーク検出の詳細が含まれています。指定したネットワークまたは指定した EndPoint のネットワークの検出には ping、EndPoint アクセスの経路の検出には tracert、DNS の検出には nslookup を使用します。

リファレンス

- OSSエラーコード
- パケットおよびオブジェクト
- URLの取得方法

Official migration tool

Linux での OSS へのデータの移行

ossimport2 同期ツールを使用すると、ローカルまたはサードパーティのクラウドに格納されているファイルを OSS に同期することができます。

主な機能:

- ローカルファイル、HTTP リンクファイルのほか、OSS、Qiniu Resource (Cloud) Storage、Baidu Object Storage、Kingsoft Standard Storage、Upyun Storage、Amazon S3 に格納されているファイルの、指定された OSS バケットへの同期
- インベントリデータの同期 (指定された時刻以降に変更されたファイルだけを同期できます)
- 増分データの自動同期
- 再開可能なデータ転送
- アップロード/ダウンロードトラフィック制御
- 並列 LIST および並列データアップロード/ダウンロード

短時間で大量のデータ (20 TB 超) を OSS に移行する場合のために、[ossimport2] 同期ツールに加えて、弊社の技術スタッフは複数マシンでの並行同期ソリューションも用意しています。サポートチームにお問い合わせください。

実行環境

[ossimport2] 同期ツールを実行するには、Java JDK 1.7 以上の環境が必要です。Oracle JDK が推奨されます。

クリックして表示

注意:プログラムを実行する前に ulimit -n コマンドを実行して、プロセスで開くことのできるファイルの数を確認します。この数が 10,240 よりも小さい場合は、変更してください。

ツールのデプロイ

まず、ローカルサーバーに同期作業ディレクトリを作成し、そのディレクトリに [ossimport2] ツールキットをダウンロードします。

たとえば、作業ディレクトリとして /root/ms を作成し、このディレクトリにツールキットをダウンロードします。

```
export work_dir=/root/ms
wget http://oss.aliyuncs.com/import-service-package/ossimport4linux.zip
unzip ./ossimport4linux.zip -d "$work_dir"
```

ツールの設定

作業ディレクトリ ($work_dir) にある設定ファイル /conf/sys.properties を次のように編集します。

```
vim $work_dir/conf/sys.properties
workingDir=/root/ms
slaveUserName=
slavePassword=
privateKeyFile=
slaveTaskThreadNum=60
slaveMaxThroughput(KB/s)=100000000
slaveAbortWhenUncatchedException=false
dispatcherThreadNum=5
```

デフォルトの設定値を使用できます。必要に応じて設定フィールドの値を編集することもできます。

フィールド	説明
workingDir	現在の作業ディレクトリ、つまりツールキットが展開されるディレクトリを示します。
slaveTaskThreadNum	同時に同期を実行するワーカースレッドの数を示します。
slaveMaxThroughput(KB/s)	最大移行スループットを示します。
slaveAbortWhenUncatchedException	不明なエラーが発生した場合に、スキップするか中止するかを示します。デフォルトでは、不明なエラーによって中止されることはありません。
dispatcherThreadNum	ディスパッチジョブでの並列スレッドの数を示します。通常はデフォルト値で問題ありません。

サービスの実行

ossimport2 では、次のコマンドを利用できます。

- ジョブの送信: 'java -jar $work_dir/bin/ossimport2.jar -c $work_dir/conf/sys.properties submit $jobConfigPath'
- ジョブのキャンセル: 'java -jar $work_dir/bin/ossimport2.jar -c $work_dir/conf/sys.properties clean $jobName'
- ステータスの表示: 'java -jar $work_dir/bin/ossimport2.jar -c $work_dir/conf/sys.properties stat detail'
- ジョブの再試行: 'java -jar $work_dir/bin/ossimport2.jar -c $work_dir/conf/sys.properties retry $jobName'

サービスの実行と使用のプロセスは次のとおりです。

1. 次のコマンドを実行してサービスを起動します。

```
cd $work_dir
nohup java -jar $work_dir/bin/ossimport2.jar -c $work_dir/conf/sys.properties start >
$work_dir/logs/ossimport2.log 2>&1 &
```

注意: サービスを起動したディレクトリのログディレクトリに、関連するログファイルが自動的に生成されます。サービスは、作業ディレクトリ ($work_dir) で起動することをお勧めします。

1. サンプルのジョブ説明ファイル local_job.cfg を編集します。

フィールドの説明:

フィールド	説明
jobName	カスタムジョブ名によってジョブが一意に識別されます。複数のジョブをそれぞれ異なる名前で送信できます。
jobType	このフィールドには、import (データ同期を実行することを示します) または audit (同期されたソースデータとターゲットデータのグローバルな一貫性を確認するだけであることを示します) を設定できます。
isIncremental=false	このフィールドは、自動増分モードを有効にするかどうかを示します。true に設定すると、incrementalModeInterval (単位: 秒) で指定された間隔で増分データがスキャンされ、OSS に同期されます。
incrementalModeInterval=86400	このフィールドは、増分モードでの同期間隔を示します。
importSince	このフィールドは、時刻を示します。この時刻よりも後のデータだけが同期されます。時刻は、UNIX タイムスタンプ (秒数) で指定します。デフ

	ォルト値は 0 です。
srcType	このフィールドは、同期のソースタイプを示します。現在指定できる値は、oss、qiniu、baidu、ks3、youpai、local です。
srcAccessKey	srcType を oss、qiniu、baidu、ks3、または youpai に設定した場合、このフィールドにデータソースのアクセスキーを設定する必要があります。
srcSecretKey	srcType を oss、qiniu、baidu、ks3、または youpai に設定した場合、このフィールドにデータソースのシークレットキーを設定する必要があります。
srcDomain	このフィールドは、ソースのエンドポイントを示します。
srcBucket	このフィールドは、ソースバケットの名前を示します。
srcPrefix	このフィールドは、ソースプレフィックスを示します。デフォルトでは空白です。srcType を local に設定した場合、同期するローカルディレクトリのパスを指定します。ディレクトリのパスは完全 (末尾が '/') である必要があります。srcType を oss、qiniu、baidu、ks3、または youpai に設定した場合、同期するオブジェクトの名前プレフィックスを指定します。すべてのファイルを同期するには、プレフィックスを空白にします。
destAccessKey	同期先 OSS のアクセスキーを指定します。
destSecretKey	同期先 OSS のシークレットキーを指定します。
destDomain	同期先 OSS のエンドポイントを指定します。
destBucket	同期先 OSS のバケットを指定します。
destPrefix	同期先に使用するファイル名プレフィックスを指定します。デフォルトは空白です。
taskObjectCountLimit	このフィールドは、1 つの子ジョブあたりの最大ファイル数を示します。この値はタスクの並列実行に影響します。通常は、ファイルの総数か、指定したダウンロードスレッド数に設定します。ファイルの総数が不明な場合は、デフォルト値に設定してください。
taskObjectSizeLimit	このフィールドは、1 つの子ジョブあたりのデータダウンロードの上限 (バイト数) を示します。
scanThreadCount	このフィールドは、ファイルの並列スキャンスレッドの数を示します。この値は、ファイルスキャンの効率に影響します。
maxMultiThreadScanDepth	このフィールドは、並列スキャンするディレクトリの最大の深さを示します。デフォルト値で問題ありません。

注意:(1) 自動増分モードを設定すると、ジョブが定期的かつ永続的に実行されて、最新データがスキャンされます。(2) srcType を youpai に設定すると、Upyun Storage の API 制限により、ファイルの LIST 操作に対してチェックポイントを実行することができません。ファイルの LIST 操作が完了する前にプロセスが停止されると、ファイルの一覧表示を最初からやり直します。

1. ジョブを送信します。

```
java -jar $work_dir/bin/ossimport2.jar -c $work_dir/conf/sys.properties  submit $work_dir/local_job.cfg
```

注意:

- 送信しようとしているジョブと同じ名前のジョブが実行中の場合、そのジョブは送信できません。
- 同期ジョブを一時停止する必要がある場合は、ossimport2 プロセスを停止します。続行する必要がある場合は、ossimport2 プロセスを再開します。同期ジョブは一時停止したところから再開されます。
- すべてのファイルを再同期するには、ossimport2 プロセスを停止してから、次のコマンドを再度呼び出して現在のジョブをクリアします。

たとえば、現在のジョブの名前が local_test の場合は (この名前は local_job.cfg ファイルで指定します)、次のコマンドを実行します。

```
ps axu | grep "ossimport2.jar.* start" | grep -v grep | awk '{print "kill -9 "$2}' | bash
java -jar $work_dir/bin/ossimport2.jar -c $work_dir/conf/sys.properties clean local_test
```

1. ジョブの実行ステータスを確認します。

```
[root@iZ23gztmc8fZ local]# java -jar $work_dir/bin/ossimport2.jar -c $work_dir/conf/sys.properties  stat
detail
--------------job stats begin---------------
-----------------job stat begin-------------------
JobName:local_test
JobState:Running
PendingTasks:0
RunningTasks:1
SucceedTasks:0
FailedTasks:0
ScanFinished:true
RunningTasks Progress:
FD813E8B93F55E67A843DBCFA3FAF5B6_1449307162636:26378979/26378979 1/1
-----------------job stat end-------------------
--------------job stats end---------------
```

これにより、現在のジョブ全体の進行状況と、現在のタスクの進行状況が表示されます。上記の例の場合は、次のようになります。 “26378979/26378979” は、アップロードするデータの総量 (26,378,979 バイト)/既にアップロードされたデータの量 (26,378,979 バイト) を示しています。 “1/1” は、アップロードするファイルの総数 (1 ファイル)/既にアップロードされたファイルの数 (1 ファイル) を示しています。

移行ツールでは、送信した 1 つのジョブが並列実行のために複数のタスクに分割されます。すべてのタスクが完了すると、ジョブが完了したと見なされます。ジョブが完了すると、JobState が “Succeed” または “Failed” と表示され、そのジョブが成功したか失敗したかが示されます。ジョブが失敗した場合は、次のコマンドを使用して各タスクの失敗の原因を確認します。

次のコマンドでは、$jobName を実際のジョブの名前に置き換えます (jobName は local_job.cfg ファイルで指定します)。

```
cat $work_dir/master/jobs/$jobName/failed_tasks/*/audit.log
```

失敗したジョブを既にツールで再試行したので、この失敗はソースまたはターゲットのデータが一時的に使用できないためである可能性があります。次のコマンドを使用して、失敗したタスクを再試行します。

```
java -jar $work_dir/bin/ossimport2.jar  -c $work_dir/conf/sys.properties retry $jobName
```

1. ジョブが失敗する一般的な原因

- ジョブの設定が不適切である。たとえば、アクセスキー/ID が間違っていることや、権限が不十分なことがあります。そうした場合は、すべてのタスクが失敗します。具体的な原因を確認するには、$work_dir/logs/ossimport2.log ファイルを調べます。
- ソースファイル名のエンコーディング方法が、システムのデフォルトのファイル名エンコーディング方法と異なっている。たとえば、Windows のデフォルトのファイル名エンコーディング方法は GBK であり、Linux では UTF-8 です。これが原因の失敗は、NFS データソースで多く見られます。
- ソースディレクトリ内のファイルがアップロード中に変更された。これが原因で失敗した場合は、audit.log に SIZE_NOT_MATCH エラーと示されます。この場合、変更前のファイルは正常にアップロードされていますが、OSS に変更が同期されていません。
- ソースファイルがアップロード中に削除された。この場合、ファイルをダウンロードできません。
- ソースファイルの名前が OSS の命名規則に準拠していない (たとえば、ファイル名を ‘/ ‘ で始めることや空白にすることはできません)。この場合、ファイルを OSS にアップロードできません。
- データソースでエラーが発生し、ソースデータのダウンロードが失敗した。

  プロセスの停止前に CLEAN 操作が実行された。これにより、プログラムの実行エラーが発生することがあります。

  プログラムが予期せず終了され、ジョブのステータスが Aborted になった。この場合は、サポートチームにお問い合わせください。

1. 推奨事項

データをクラウド (非ローカル) から OSS に移行する必要があるが、帯域幅リソースが十分でない場合は、

従量課金モードで ECS インスタンスを購入することをお勧めします (購入アドレス: https://ecs-buy.aliyun.com/#/postpay)。

ECS 設定:

i. 支払い方法として [Pay-As-You-Go] を選択します。
i. OSS インスタンスが存在するリージョンを選択します。
i. ピーク帯域幅として 100M を選択します (インバウンドトラフィックおよびイントラネットトラフィックには課金されません)。
i. 必要に応じて Linux または Windows システムを選択します。
i. 他の項目については、最小設定を選択します。

料金はリージョンによって多少異なります。杭州では、設定料金が 0.277 元/時、パブリックトラフィック費用が 0.8 元/GB です (パブリック帯域幅は、トラフィック課金モードで課金されます。アウトバウンドトラフィックのみに課金され、インバウンドトラフィックとイントラネットトラフィックには課金されません。料金は、実際の使用量に応じて時間単位で引き落とされます。たとえば、インターネットのアウトバウンドトラフィックを 1 時間で 2.5 GB 使用すると、請求額は 2.5 GB x 0.8 元/時 = 2.0 元になります)。

移行サービスを設定する場合は、”internal” を含むイントラネットドメイン名を targetDomain に設定します。ソース側も OSS の場合は、”internal” を含むイントラネットドメイン名を srcDomain に設定します。このように設定すると、ソース OSS からダウンロードする際に発生するトラフィック料金を節約でき、OSS アクセス時間に対する支払いだけで済みます。

移行が終わったら、ECS インスタンスをリリースします (閉じないでください)。

Windows での OSS へのデータの移行

概要

ossimport2 ツールを使用すると、ローカルまたは他のクラウドストレージシステムに格納されているファイルを OSS に移行することができます。

主な機能:

- ローカルファイル、HTTP リンクファイルのほか、Qiniu Resource (Cloud) Storage、Baidu Object Storage、Kingsoft Standard Storage、Upyun Storage、Amazon S3 に格納されているファイルの移行
- 再開可能なデータ転送
- アップロードトラフィック制御
- 指定された時刻以降に変更されたデータファイルのみの同期
- 並列 LIST および並列データアップロード/ダウンロード
- 増分データの自動同期

オフラインの ossimport2 ツールに加えて、移行サービスも提供しています。移行サービスを使用すると、上記のデータソースを移行することや、OSS への移行を複数のマシンで並列実行することができます。大量のデータを短時間で OSS に移行する場合は、サポートチームにお問い合わせください。

実行環境

お使いのマシンで Java JDK 7 以降の環境が実行されていることを確認します。JDK 7 よりも前の Java 環境の場合は、こちらをクリックして最新の Java 環境にアップグレードしてください。

ツールのデプロイ

こちらをクリックして、ツールキットをダウンロードし、展開します。

ツールの設定

作業ディレクトリにある設定ファイル conf/sys.properties と conf/local_job.cfg を編集します。ファイル内の指示に従ってツールを設定します。

サービスの実行

“One-key Import.bat” プログラムを実行します。

以前にこのプログラムを実行したことがある場合は、以前のジョブを一時停止したところから続行するか、新しい同期ジョブを実行するかをたずねるメッセージが表示されます。同期元または同期先の設定が変更されている場合は、新しい同期ジョブを実行します。

ジョブが開始されると、新しいコマンドウィンドウが開き、同期ジョブの進行状況とログが表示されます。既に開いているウィンドウ内のジョブのステータスは、10 秒ごとに更新されます。同期プロセス中、この 2 つのウィンドウは閉じないでください。

ジョブが完了してもタスクが失敗した場合は、再試行するかどうかをたずねるメッセージが表示されます。再試行する場合は「y」、プログラムを終了する場合は「n」を入力します。

アップロードできなかったファイルと失敗の原因は、
master/jobs/local_test/failed_tasks/taskid/audit.log で確認できます。

ジョブが失敗する一般的な原因

- ジョブの設定が不適切である。たとえば、アクセスキー/ID が間違っていることや、権限が不十分なことがあります。そうした場合は、すべてのタスクが失敗します

。具体的な原因を確認するには、logs/ossimport2.log ファイルを調べます。
- ソースファイル名のエンコーディング方法が、システムのデフォルトのファイル名エンコーディング方法と異なっている。たとえば、Windows のデフォルトのファイル名エンコーディング方法は GBK であり、Linux では UTF-8 です。
- ソースディレクトリ内のファイルがアップロード中に変更された。これが原因で失敗した場合は、audit.log に SIZE_NOT_MATCH エラーと示されます。この場合、変更前のファイルは正常にアップロードされていますが、OSS に変更が同期されていません。
- ソースファイルがアップロード中に削除された。この場合、ファイルをダウンロードできません。
- ソースファイルの名前が OSS の命名規則に準拠していない (たとえば、ファイル名を ‘/‘ で始めることや空白にすることはできません)。この場合、ファイルを OSS にアップロードできません。
- データソースでエラーが発生し、ソースデータのダウンロードが失敗した。
- プログラムが予期せず終了され、ジョブのステータスが Aborted になった。この場合は、サポートチームにお問い合わせください。