

Object Storage Service

ユーティリティ

ユーティリティ

ossutil

Ossutil では、コマンドラインを使用して OSS データを簡単に管理できます。現在のバージョンでは、一部のバケット管理機能およびマルチパート管理機能を使用できません。これらの機能は、今後のバージョンで提供されます。これらの機能が必要な場合は、osscli コマンドラインツールを代わりに使用してください。

ツールのダウンロード

現在のバージョン

- 現在のバージョン: 1.0.0

実行環境

- Windows/Linux/Mac
- サポートされているアーキテクチャ
- x86 (32bit, 64bit)

バイナリプログラムのダウンロード

- [Linux x86 32bit] ossutil32
- [Linux x86 64bit] ossutil64
- [Windows x86 32bit] ossutil32.zip
- [Windows x86 64bit] ossutil64.zip
- [mac x86 64bit] ossutilmac64

バイナリプログラムのインストールと使用

使用しているオペレーティングシステムのバイナリプログラムまたは対応する圧縮パッケージをダウンロードし、バイナリプログラムを実行します。(バイナリプログラムが実行ファイルでない場合、chmod 755 ossutilを実行して実行可能にします。) バイナリプログラムは次のように実行します。

Linux システムの場合: `./ossutil`

Window システムの場合、次の 2 つの方法のいずれかを使用します (次の例は 64 ビットオペレーティングシステムの場合です):

パッケージを解凍します。バッチファイルをダブルクリックし、`ossutil64.exe` と入力します。

パッケージを解凍します。`cmd` を実行し、バイナリプログラムがあるディレクトリに移動します。`ossutil64.exe` と入力します。

Mac システムの場合: `./ossutilmac64`

クイックスタート

ossutil の言語設定

`ossutil` のコマンドを実行するとき、`-L` オプションを使用して言語を設定できます。値には `CH` または `EN` を使用でき、それぞれ中国語と英語に対応します。値は大文字と小文字が区別されません。デフォルト値は `CH` (中国語) です。言語を `CH` (中国語) に設定する場合、システムが UTF-8 エンコードであることを確認する必要があります。そうでない場合、文字化けする場合があります。

例:

`./ossutil help ls` は、デフォルト言語で `ls` のヘルプを表示するときに使用します。

`./ossutil help ls -L ch` は、中国語で `ls` のヘルプを表示するときに使用します。

`./ossutil help ls -L en` は、英語で `ls` のヘルプを表示するときに使用します。

`./ossutil config -L ch` は、インタラクティブな設定コマンドの `ossutil config` を実行するときに使用します。プロンプトの言語は中国語です。

`./ossutil config -L en` は、インタラクティブな設定コマンドの `ossutil config` を実行するときに使用します。プロンプトの言語は英語です。

注意: デフォルトでは、`ossutil` のエラー出力は英語です。これは、前述のオプションの影響を受けません。

コマンドリストの取得

`./ossutil` または `./ossutil help`

```
$/ossutil
使用法: ossutil [command] [args...] [options...]
```

ossutil help を実行して、コマンドヘルプを表示

Commands:

mb cloud_url [options]

bucket の作成

ls [cloud_url] [options]

buckets または objects をリスト

rm cloud_url [options]

buckets または objects を削除

stat cloud_url [options]

buckets または objects の説明を表示

set-acl cloud_url [acl] [options]

buckets または objects の ACL を設定

set-meta cloud_url [meta] [options]

アップロードしたオブジェクトのメタデータを設定

cp src_url dest_url [options]

オブジェクトのアップロード、ダウンロード、またはコピー

restore cloud_url [options]

オブジェクトをフリーズ状態から読み取り可能状態に復元

create-symlink cloud_url target_url [options]

シンボリックリンクを作成

read-symlink cloud_url [options]

シンボリックリンクファイルの説明を取得

Additional Commands:

help [command]

コマンドのヘルプドキュメントを取得

config [options]

設定項目を保存する設定ファイルを作成

hash file_url [options]

ローカルファイルの crc64 または MD5 を計算

update [options]

ossutil を更新

\$/ossutil -L en

Usage: ossutil [command] [args...] [options...]

Please use 'ossutil help command' to show help of command

Commands:

mb cloud_url [options]

Make Bucket

ls [cloud_url] [options]

List Buckets or Objects

rm cloud_url [options]

Remove Bucket or Objects

stat cloud_url [options]

Display meta information of bucket or objects

set-acl cloud_url [acl] [options]

Set acl on bucket or objects

set-meta cloud_url [meta] [options]

set metadata on already uploaded objects

cp src_url dest_url [options]

Upload, Download or Copy Objects

restore cloud_url [options]

Restore Frozen State Object to Read Ready Status

```
create-symlink cloud_url target_url [options]
Create symlink of object
read-symlink cloud_url [options]
Display meta information of symlink object
```

Additional Commands:

```
help [command]
Get help about commands
config [options]
Create configuration file to store credentials
hash file_url [options]
Get crc64 or md5 of local file
update [options]
Update ossutil
```

コマンドのヘルプドキュメントの表示

`./ossutil help cmd` コマンドを実行する前に、`help` コマンドを実行してヘルプドキュメントを確認することを強くお勧めします。

```
./ossutil help config -L ch
SYNOPSIS
```

設定項目を保存する設定ファイルを作成

SYNTAX

```
ossutil config [-e endpoint] [-i id] [-k key] [-t token] [-L language] [--output-dir outdir] [-c file]
```

DETAIL DESCRIPTION

このコマンドは、設定ファイルを作成し、設定ファイルにカスタマイズされた設定項目を格納し、設定項目を使用して OSS にアクセスするときにアクセス情報を提供するために使用されます。（コマンドが設定項目を必要とするかどうかは、`--config-file` オプションがサポートされているかどうかによって異なります。詳細については、コマンドのヘルプを参照してください）。

設定ファイルを保存するパスを指定することができます。デフォルトパスは `/home/admin/.ossutilconfig` です。設定ファイル（たとえば、`a`）が存在する場合、`ossutil` は `a` を `a.bak` に格納し、ファイル `a` を再度作成し、設定項目を書き込みます。`a.bak` がすでに存在する場合は、ファイル `a` によって上書きされます。

注意：

（１）設定ファイルのパスがデフォルトパスでない場合は、`--config-file` オプションでパスを指定できます。`--config-file` オプションを指定しないと、コマンド実行時に `/home/admin/.ossutilconfig` がデフォルトで読み込まれます）。

（２）いくつかの設定項目は、コマンド実行時に `--endpoint` や `--access-key-id` などのオプションを使用して設定できます（オプションについての詳細は各コマンドのヘルプを参照してください）。コマンドを実行するときにオプションを指定し、設定ファイル内に設定もあった場合、優先順位は「指定オプション > 設定ファイル」です。

（３）コマンド実行時に `--endpoint`、`--access-key-id`、`--access-key-secret`、および `--sts-token` オプションを指定した場合、`ossutil` に設定ファイルの存在は必須ではありません。

使用法:

このコマンドは、１）対話モードまたは２）非対話モードで使用できます。対話モードは、より高いセキュリティを保証する

ので推奨されます。

1) ossutil config [-c file]

このモードは対話方式で情報を設定できます。Ossutil は対話で下記の情報をリクエストします。

(1) config file

設定ファイルのパスを指定します。空欄のまま Enter キーを押すと、ossutil は次のデフォルト設定ファイルを使用します。
/home/admin/.ossutilconfig.

設定ファイルを指定する場合は、コマンド実行時に設定ファイルのパスに --config-file オプションを使用します。--config-file オプションをサポートするコマンドの詳細については、各コマンドのヘルプを参照してください。

(2) language

初期設定の場合（設定ファイルは存在しません）、ossutil は言語を設定する必要があります。値は、CH（中国語）または EN（英語）です。Enter キーを押すと、ossutil は --language オプションの値に基づいて言語を設定します。--language オプションを設定しないと、ossutil はデフォルトで言語を CH に設定します。

設定ファイルが存在する場合、ossutil は、設定ファイル内の指定された言語オプションおよび言語情報に基づいて言語を構成します。

Ossutil は、操作中に設定ファイルから言語オプションを読み取ります。このオプションが存在しない、無効な場合、ossutil はデフォルトで言語を CH に設定します。

注：この設定項目は、config コマンドが正常に実行された後に有効になります。config コマンドを実行すると、表示された言語は設定の影響を受けません。

(3) endpoint, accessKeyID, accessKeySecret

Enter は、設定項目がスキップされたことを示します。注：endpoint は、oss.aliyuncs.com などの第 2 レベルドメイン（SLD）である必要があります。

上記のオプションは必須です。

(4) stsToken

一時トークンを使用して OSS にアクセスするには、このオプションを指定します。それ以外の場合は、Enter キーを押してこのオプションをスキップします。

(5) outputDir

このオプションは、出力ファイルが存在するディレクトリのパスを設定するために使用します。対話モードでは、このオプションの設定はサポートされていません。ただし、このオプションは設定ファイルで有効です。

outputDir オプションのデフォルトのディレクトリは、現在のディレクトリの ossutil_output です。Ossutil は、操作中にこのフォルダ内のすべての出力ファイルを生成します。現在、出力ファイルには、cp コマンドを実行してバッチ処理の例外が発生した場合の各ファイルの操作エラーを記録するレポートファイルが含まれています。

outputDir オプションとレポートファイルの詳細については、cp コマンドのヘルプを参照してください。

注意：outputDir オプションが存在しない場合、ossutil は出力ファイルを生成するときに自動的にディレクトリを作成します。outputDir オプションが指定するディレクトリが存在しない場合、エラーが報告されます。

次の Bucket-Endpoint および Bucket-Cname オプションは対話モードから削除されますが、設定ファイルでは有効です。

(6) Bucket-Endpoint

Bucket-Endpoint オプションは、指定した各バケットのエンドポイントを個別に設定するために使用します。このオプションは、設定ファイル内のデフォルトのエンドポイント構成の前にあります。

このバージョンでは、ossutil は対話モードで Bucket-Endpoint 設定を削除します。ただし、この設定項目は設定ファイルでも有効です。したがって、各バケットのエンドポイントを個別に指定する場合は、設定ファイルで設定を行うことができます。注：エンドポイントは SLD（oss.aliyuncs.com など）である必要があります。

Bucket-Endpoint オプションが指定されている場合、ossutil は、バケット上で操作を実行するときにオプション内のバケットに対応するエンドポイントを検索します。見つかった場合、エンドポイントは基本構成でエンドポイントを上書きします。ただし、コマンド実行時に --endpoint オプションを指定した場合、--endpoint オプションの優先順位が最も高くなります。

(7) Bucket-Cname

Bucket-Cname オプションは、指定されたバケットごとに CNAME ドメイン名（CDN ドメイン）を個別に設定するために使用されます。このオプションは、設定ファイルの Bucket-Endpoint オプションとエンドポイントの設定よりも前です。

このバージョンでは、ossutil は対話モードで Bucket-Cname 設定を削除しました。ただし、この設定項目は設定ファイルでも有効です。したがって、各バケットの CNAME ドメイン名を個別に指定する場合は、設定ファイルで設定を行うことができます。

Bucket-Cname オプションが指定されている場合、ossutil はバケット上で操作を実行するときにオプション内のバケットに対応する CNAME ドメイン名を検索します。CNAME ドメイン名が見つかった場合、バケットエンドポイントオプションと基本設定のエンドポイントを上書きします。ただし、コマンド実行時に --endpoint オプションを指定した場合、--endpoint

オプションの優先順位が最も高くなります。

優先度：--endpoint > Bucket-Cname > Bucket-Endpoint > endpoint > デフォルト endpoint

2) ossutil config options

コマンドの実行時に --language および --config-file オプション以外のオプションを指定すると、コマンドは非対話モードになります。すべての設定項目はオプション設定ファイルで指定します。

設定ファイルの形式：

```
[Credentials]
language = CH
endpoint = oss.aliyuncs.com
accessKeyID = your_key_id
accessKeySecret = your_key_secret
stsToken = your_sts_token
outputDir = your_output_dir
[Bucket-Endpoint]
bucket1 = endpoint1
bucket2 = endpoint2
...
[Bucket-Cname]
bucket1 = cname1
bucket2 = cname2
...
```

SAMPLE

```
ossutil config
ossutil config -e oss-cn-hangzhou.aliyuncs.com -c ~/.myconfig
```

OPTIONS

-c, --config-file

ossutil の設定ファイルのパスを指定します。Ossutil は、起動時に設定ファイルから設定を読み取り、設定コマンドを使用してファイルに設定を書き込みます。

-e, --endpoint

ossutil のエンドポイントを指定します（オプション値は、設定ファイルの対応する設定を上書きします）。設定値は SLD でなければなりません。

-i, --access-key-id

OSS へのアクセスに使用される AccessKeyID を指定します（オプション値は、設定ファイルの対応する設定を上書きします）。

-k, --access-key-secret

OSS にアクセスするために使用される AccessKeySecret を指定します（オプション値は、設定ファイル内の対応する設定を上書きします）。

-t, --sts-token

オプションです。OSS にアクセスするために使用する STSToken を指定します（オプション値は、設定ファイルの対応する設定を上書きします）。

--output-dir=ossutil_output

出力ファイルが格納されているディレクトリを指定します。出力ファイルには、cp コマンドを使用してファイルをバッチで

コピーする際にエラーが発生したときに生成されるレポートファイルが含まれます。(レポートファイルの詳細については、cp コマンドのヘルプを参照してください)。デフォルト値は、カレントディレクトリの ossutil_output サブディレクトリです。

-L CH, --language=CH

ossutil の言語を指定します。値は CH または EN にすることができ、デフォルト値は CH です。値が CH の場合は、システムが UTF-8 エンコードであることを確認してください。

ossutil の設定

コマンドを使用して OSS にアクセスする場合、最初に Access Key ペアを設定します。

Ossutil は、インタラクティブモードまたは非インタラクティブモードに設定できます。

設定コマンドのヘルプドキュメントを表示するには、ossutil help config を実行します。

非インタラクティブモードでの ossutil の設定

./ossutil config

\$/ossutil config -L ch

このコマンドは、設定ファイルを作成し、そこに設定情報を保管するために使用されます。

設定ファイルのパスを指定することができます。デフォルトパスは /home/admin/.ossutilconfig です。Enter キーを押すと、デフォルトのパスが使用されます。別のパスを指定する場合は、コマンド実行時に --config-file オプションでパスを指定します。

非インタラクティブモードでの ossutil の設定

./ossutil config -e oss.aliyuncs.com -i your_id -k your_key

サポートされているすべてのオプションの表示

-h オプションを使用すると、ossutil でサポートされているすべてのオプションを表示できます。

\$/ossutil -h

Usage of ossutil:

Options:

-s --short-format Used to display the short format. If this option is not specified, the long format is displayed by default.

--snapshot-path= Used to accelerate incremental uploading of files in batches in some scenarios. (File downloading and copying do not support this option currently.) This option is used when ossutil uses the cp command to upload files. Ossutil takes a snapshot of file uploads and stores it in the specified directory. It will read the snapshot from the specified directory for incremental upload when this option is used the next time. The specified snapshot directory must be a writable directory in the local file system. If the directory does not exist, ossutil creates a file to record the snapshot information. If the directory already exists, ossutil reads the snapshot information in the directory, performs incremental uploading accordingly, and updates the snapshot information. (Ossutil only uploads files that fail to be uploaded last time and have been locally modified.) NOTE: By using this option, the local lastModifiedTime of files that have been successfully uploaded is recorded and compared with that of files to be

uploaded next time to determine whether to skip uploading of same files. When using this option, ensure that the corresponding objects on the OSS are not modified during the two uploading periods. In other scenarios than this one, use the `--update` option to incrementally upload files in batches. In addition, ossutil does not actively delete the snapshot information under `snapshot-path`. To avoid too much snapshot information, clear `snapshot-path` when confirming that the snapshot information is useless.

`-j --jobs=` Specifies the number of concurrent tasks when multiple files are operated. The value ranges from 1 to 10000, and the default value is 5.

`-v --version` Used to display ossutil version (1.0.0.Beta2) and exit.

`--output-dir=` Specifies the directory in which output files are located. The output files include the report file generated when an error occurs for copying files in batches using the `cp` command. (For details about the report file, refer to the `cp` command help.) The default value is the `ossutil_output` sub-directory in the current directory.

`--parallel=` Specifies the number of concurrent tasks operated in a single file. The value ranges from 1 to 10000. The default value is determined by ossutil based on the operation type and file size.

`-L --language=` Specifies the language of ossutil. The value can be CH or EN, and the default value is CH.

`-t --sts-token=` Specifies the STSToken used to access the OSS (the option value will overwrite the corresponding settings in the configuration file). It is optional.

`-m --multipart` Indicates that the operation objects are the incomplete Multipart events in the bucket, rather than the default objects.

`-b --bucket` Used to operate a bucket, confirming that an operation is for the bucket.

`--delete` Used to delete an operation.

`-e --endpoint=` Specifies the basic endpoint configuration of ossutil (the option value will overwrite the corresponding settings in the configuration file). It must be a second-level domain.

`-k --access-key-secret=` Specifies the AccessKeySecret used to access the OSS (the option value will overwrite the corresponding settings in the configuration file).

`--bigfile-threshold=` Specifies the threshold for enabling the resumable data transfer for large files. The value ranges from 0 B to 9223372036854775807 B, and the default value is 100 MB.

`--retry-times=` Specifies the number of retries when an error occurs. The value ranges from 1 to 500, and the default value is 3.

`-a --all-type` Indicates that the operation objects are the objects and incomplete Multipart events in the bucket.

`-r --recursive` Indicates a recursive operation. If this option is specified, commands supporting this option will operate on all objects meeting the criteria in the bucket. Otherwise, the commands operate only on a single object specified in the URL.

`-f --force` Indicates a forcible operation without asking.

`-u --update` Indicates an update operation.

`-c --config-file=` Specifies the configuration file path of ossutil. Ossutil reads configuration from the configuration file during startup and writes configuration to the file using the `config` command.

`-i --access-key-id=` Specifies the AccessKeyID used to access the OSS (the option value will overwrite the corresponding settings in the configuration file).

`--acl=` Used to configure the ACL.

`-d --directory` Used to return files and sub-directories in the current directory, rather than recursively displaying all objects in all sub-directories.

`--checkpoint-dir=` Specifies the checkpoint directory path (the default value is `.ossutil_checkpoint`). If a resumable data transfer fails, ossutil automatically creates this directory and records the checkpoint information in the directory. If a resumable data transfer succeeds, ossutil deletes this directory. If this option is specified, ensure that the specified directory can be deleted.

`--type=` Specifies the calculation type. The value can be `crc64` or `md5`, and the default value is `crc64`.

`-h --help` Show usage message

ossutil のすべてのコマンドは、上記のオプションの一部をサポートしています。各コマンドでサポートされるオプションを調べるには、`ossutil help` コマンドを使用します。

バケット関連コマンド

Ossutil を使用すると、バケットを作成、削除、一覧表示することができます。また、バケットの ACL を設定できます。バケットに関連する他の管理機能は現在利用できません。これらの機能を使用する必要がある場合は、「osscli」を参照してください。

これらのコマンドを実行する前に、config コマンドを実行して Access Key ペアを設定してください。

バケットの作成

```
ossutil mb oss://bucket [--acl=acl] [--storage-class sc] [-c file]
```

ACL を指定しない場合、バケットにデフォルトで公開読み書き権限が設定されます。バケットが作成された後、ossutil は消費時間を表示して終了します。それ以外の場合、ossutil は、エラー情報を出力します。--storage-class オプションを使用すると、ストレージモードを指定できます。

ossutil help mb を実行すると、バケットの作成に関するヘルプ情報を表示できます。

```
./ossutil mb oss://test
0.220478(s) elapsed
```

バケットの削除

ossutil help rm を実行すると、バケットの削除に関するヘルプ情報を表示できます。

注意:

- バケットを削除するには、-b オプションを指定する必要があります。
- 削除したバケットを別のユーザが作成した場合、バケットは別のユーザの所有になります。
- 一度削除したバケットを復元することはできません。

バケットにデータが含まれていない場合

```
ossutil rm oss://bucket -b
```

```
./ossutil rm oss://test -b
Do you really mean to remove the Bucket: test(y or N)? y
0.220478(s) elapsed
```

バケットに、オブジェクト、マルチパート、または他のデータが含まれている場合、バケットを削除する前にすべてのデータを削除します。次のコマンドを実行すると、すべてのデータとバケットを削除できます。

```
ossutil rm oss://bucket -bar
```

ossutil help rm を実行すると、バケットの削除に関するヘルプ情報を表示できます。

バケットの一覧表示

```
./ossutil ls または ./ossutil ls oss://
```

-s オプションを使用すると、簡易形式で表示できます。詳細なヘルプ情報を表示するには、ossutil help ls を実行します。

```
$./ossutil ls
CreationTime Region StorageClass BucketName
2016-10-21 16:18:37 +0800 CST oss-cn-hangzhou Archive oss://go-sdk-test-bucket-xyz-for-object
2016-12-01 15:06:21 +0800 CST oss-cn-hangzhou Standard oss://ossutil-test
2016-07-18 17:54:49 +0800 CST oss-cn-hangzhou Standard oss://ossutilconfig
2016-07-20 10:36:24 +0800 CST oss-cn-hangzhou IA oss://ossutilupdate
2016-11-14 13:08:36 +0800 CST oss-cn-hangzhou IA oss://yyyyy
2016-08-25 09:06:10 +0800 CST oss-cn-hangzhou Archive oss://ztzt
2016-11-21 21:18:39 +0800 CST oss-cn-hangzhou Archive oss://ztztzt
Bucket Number is: 7
0.252174(s) elapsed
```

バケット内のファイルの一覧表示

Ossutil を使用すると、バケット内のオブジェクトと UploadID を一覧表示できます。デフォルトでは、オブジェクトが表示されます。-m オプションを使用すると、UploadID を表示できます。-a オプションを使用すると、オブジェクトと UploadID を同時に表示できます。

オブジェクトの一覧表示

```
./ossutil ls oss://bucket
```

```
$./ossutil ls oss://ossutil-test
LastModifiedTime Size(B) StorageClass ETAG ObjectName
2016-12-01 15:06:37 +0800 CST 10363812 Standard 61DE142E5AFF9A6748707D4A77BFBCFB oss://ossutil-test/a1
2016-12-01 15:06:42 +0800 CST 10363812 Standard 61DE142E5AFF9A6748707D4A77BFBCFB oss://ossutil-test/a2
2016-12-01 15:06:45 +0800 CST 10363812 Standard 61DE142E5AFF9A6748707D4A77BFBCFB oss://ossutil-test/a3
Object Number is: 3
0.007379(s) elapsed
```

オブジェクトとマルチパートの一覧表示

```
./ossutil ls oss://bucket -a
```

```
$ ossutil ls oss://bucket1 -a
LastModifiedTime Size(B) StorageClass ETAG ObjectName
```

```
2015-06-05 14:06:29 +0000 CST 201933 Standard 7E2F4A7F1AC9D2F0996E8332D5EA5B41
oss://bucket1/dir1/obj11
2015-06-05 14:36:21 +0000 CST 201933 Standard 6185CA2E8EB8510A61B3A845EAFE4174 oss://bucket1/obj1
2016-04-08 14:50:47 +0000 CST 6476984 Standard 4F16FDAE7AC404CEC8B727FCC67779D6
oss://bucket1/sample.txt
Object Number is: 3
InitiatedTime UploadID ObjectName
2017-01-13 03:45:26 +0000 CST 15754AF7980C4DFB8193F190837520BB oss://bucket1/obj1
2017-01-13 03:43:13 +0000 CST 2A1F9B4A95E341BD9285CC42BB950EE0 oss://bucket1/obj1
2017-01-13 03:45:25 +0000 CST 3998971ACAF94AD9AC48EAC1988BE863 oss://bucket1/obj2
2017-01-20 11:16:21 +0800 CST A20157A7B2FEC4670626DAE0F4C0073C oss://bucket1/tobj
UploadId Number is: 4
0.191289(s) elapsed
```

-s オプションを使用すると、簡易形式で表示できます。

-d オプションを使用すると、レベル 1 ディレクトリの内容を表示できます。

```
$ ossutil ls oss://bucket1 -d
oss://bucket1/obj1
oss://bucket1/sample.txt
oss://bucket1/dir1/
Object and Directory Number is: 3
UploadID ObjectName
15754AF7980C4DFB8193F190837520BB oss://bucket1/obj1
2A1F9B4A95E341BD9285CC42BB950EE0 oss://bucket1/obj1
3998971ACAF94AD9AC48EAC1988BE863 oss://bucket1/obj2
A20157A7B2FEC4670626DAE0F4C0073C oss://bucket1/tobj
UploadId Number is: 4
0.119884(s) elapsed
```

バケットの ACL の設定

バケットを作成したとき、バケットのデフォルト ACL は公開読み書きに設定されます。set-acl コマンドを実行すると、バケットの ACL を変更できます。

バケットの ACL を設定するときは、-b オプションを指定する必要があります。

bucket1 にプライベート権限を付与するには、次のようにします。

```
./ossutil set-acl oss://bucket1 private -b
```

ヘルプ表示の set-acl コマンドを実行すると、ACL の設定に関する詳細情報を表示できます。

オブジェクト関連コマンド

Ossutil を使用すると、ファイルのアップロード、ダウンロード、およびコピー、オブジェクトの ACL およびメタの設定、オブジェクトのメタ情報の表示を行うことができます。

これらのコマンドを実行する前に、config コマンドを実行して Access Key ペアを設定してください。

ファイルのアップロード、ダウンロード、コピー

cp コマンドを実行する前に、ossutil help cp を使用してヘルプ情報を確認することを強くお勧めします。

cp コマンドを使用すると、ファイルをアップロード、ダウンロード、コピーすることができます。また、-r オプションを使用すると、フォルダーをコピーできます。Ossutil は、デフォルトで大きなファイルのマルチパートアップロードを実行し、再開可能なデータ転送をサポートしています (マルチパートアップロードが有効になる大きなファイルのしきい値は、--bigfile-threshold オプションを使用して設定できます)。

-f オプションを使用すると、デフォルトでファイルを強制的にアップグレードできます。ターゲット側に同じ名前のファイルが存在する場合、ファイルは直接上書きされます。

一括でのファイルのアップロード、ダウンロード、またはコピーの実行中に、あるファイルでエラーが発生した場合、ossutil は、デフォルトでレポートファイルにエラー情報を記録し、このファイルをスキップして他のファイルについて操作を実行します (バケットが存在しない場合、または accessKeyID や accessKeySecret が正しくないため権限認証が無効の場合、Ossutil は、他のファイルのコピーを続行します)。詳細については、ossutil help cp を参照してください。

Ossutil は、特定のシナリオでの増分アップロードポリシー --update および --snapshot-path をサポートしています。詳細については、ossutil help cp を参照してください。

ossutil 1.0.0.Beta1 以降、デフォルトで、ファイルのアップロード中に crc64 が有効になります。

(1) 1 つのファイルのアップロード:

```
./ossutil cp a oss://ossutil-test
Succeed: Total num: 1, size: 230. OK num: 1(upload 1 files).
0.699795(s) elapsed
```

(2) フォルダーのアップロード:

```
./ossutil cp -r dir oss://ossutil-test
Succeed: Total num: 35, size: 464,606. OK num: 35(upload 34 files, 1 directories).
0.896320(s) elapsed
```

ファイルのアップロード、ダウンロード、コピーのパフォーマンスチューニング

cp コマンドで、-jobs および -parallel オプションを使用して、同時操作数を制御できます。-jobs オプションは、複数のファイルをアップロード、ダウンロード、またはコピーするときにファイル間で有効になる同時操作数を制御します。-parallel オプションは、大きなファイルをマルチパートでアップロード、ダウンロード、またはコピーするときに大きなファイルで有効になる同時操作数を制御します。

Ossutil は、デフォルトではファイルサイズに基づいて並列操作数を計算します (このオプションは小さいファイルでは機能しません。また、マルチパートでアップロード、ダウンロード、またはコピーされる大きなファイルのしきい値は、--bigfile-threshold オプションで制御できます)。複数の大きなファイルを一括でア

アップロード、ダウンロード、またはコピーする場合、実際の同時操作数は、ジョブ数と並列操作数の積として計算されます。ossutil によって設定されるデフォルトの同時操作数がパフォーマンス要件を満たさない場合、これら 2 つのオプションを調整することで、パフォーマンスを向上または低減させることができます。

注意:

同時操作数が大きすぎる場合、スレッド間のリソースの切り替えや確保により、ossutil のアップロード、ダウンロード、コピーのパフォーマンスが低下する場合があります。そのため、実際のマシン状態に基づいて、これらの 2 つのオプションの値を調整します。負荷テストを実行する場合、最初は 2 つのオプションに小さい値を設定し、徐々に最適な値に調整します。

—jobs および —parallel オプションの値が大きすぎる場合、マシンのリソースが限られていると、データ転送速度の低下により EOF エラーが発生する場合があります。この場合、—jobs および —parallel オプションの値を適切に減らしてください。

オブジェクトの ACL の設定

Ossutil では、set-acl コマンドを使用してオブジェクトの ACL を設定します。-r オプションを使用すると、複数のオブジェクトの ACL を一括で設定できます。

詳細については、ossutil help set-acl を参照してください。

```
$/ossutil set-acl oss://dest/a private
0.074507(s) elapsed
```

オブジェクトの ACL を一括で設定するには、次のようにします。

```
$/ossutil set-acl oss://dest/a private -r
Do you really mean to recursively set acl on objects of oss://dest/a(y or N)? y
Succeed: Total 3 objects. Setted acl on 3 objects.
0.963934(s) elapsed
```

オブジェクトのメタの設定

Ossutil では、set-meta コマンドを使用してオブジェクトのメタ情報を設定します。-r オプションを使用すると、複数のオブジェクトのメタを一括で設定できます。

詳細については、ossutil help set-meta を参照してください。

```
./ossutil set-meta oss://dest/a x-oss-object-acl:private -u
```

オブジェクトの説明 (メタ) の表示

Ossutil では、stat コマンドを使用してオブジェクトの説明 (メタ) を表示します。

詳細については、ossutil help stat を参照してください。

```
./ossutil stat oss://dest/a
ACL : default
Accept-Ranges : bytes
Content-Length : 230
Content-Md5 : +5vbQC/MSQK0xXSiyKBZog==
Content-Type : application/octet-stream
Etag : FB9BDB402FCC4902B4C574A2C8A059A2
Last-Modified : 2017-01-13 15:14:22 +0800 CST
Owner : aliyun
X-Oss-Hash-Crc64ecma : 12488808046134286088
X-Oss-Object-Type : Normal
0.125417(s) elapsed
```

オブジェクトの無効化状態から読み取り可能な状態への復元

Ossutil では、restore コマンドを使用して、無効化状態から読み取り可能な状態へオブジェクトを復元します。-r オプションを使用すると、複数のオブジェクトを無効化状態から読み取り可能な状態へ一括で復元できます。

詳細については、ossutil help restore を参照してください。

```
./ossutil restore oss://utiltest/a
0.037729(s) elapsed
```

シンボリックリンクの作成

Ossutil では、create-symlink コマンドを使用してシンボリックリンクを作成します。

詳細については、ossutil help create-symlink を参照してください。

```
./ossutil create-symlink oss://utiltest/b a
0.037729(s) elapsed
```

シンボリックリンクファイルの説明の読み取り

Ossutil では、read-symlink コマンドを使用してシンボリックリンクファイルの説明を読み取ります。

詳細については、ossutil help read-symlink を参照してください。

```
./ossutil read-symlink oss://utiltest/b
Etag : D7257B62AA6A26D66686391037B7D61A
Last-Modified : 2017-04-26 15:34:27 +0800 CST
X-Oss-Symlink-Target : a
0.112494(s) elapsed
```

マルチパート関連コマンド

Ossutil を使用すると、UploadID を一覧表示したり、指定したオブジェクトのすべての UploadID を削除したりできます。マルチパート UploadID に関連する他の管理機能は現在利用できません。これらの機能を使用する必要がある場合は、「osscmd」を参照してください。

マルチパートの詳細については、「再開可能なアップロード」を参照してください。

注意: 大きなファイルをアップロードまたはコピーするとき、ossutil は、UploadPart コマンドを実行せずに、マルチパートアップロードと再開可能なデータ転送を自動的に実行します。

UploadID の一覧表示

-m オプションを使用すると、指定したオブジェクトの未完了の UploadID をすべて一覧表示できます。-a オプションを使用すると、オブジェクトと UploadID を一覧表示できます。

```
$ ossutil ls oss://bucket1/obj1 -m
InitiatedTime UploadID ObjectName
2017-01-13 03:45:26 +0000 CST 15754AF7980C4DFB8193F190837520BB oss://bucket1/obj1
2017-01-13 03:43:13 +0000 CST 2A1F9B4A95E341BD9285CC42BB950EE0 oss://bucket1/obj1
UploadId Number is: 2
0.070070(s) elapsed
```

指定したオブジェクトのすべての UploadID の削除

-m オプションを使用すると、指定したオブジェクトの未完了の UploadID をすべて削除できます。-r オプションを同時に指定すると、指定したオブジェクトをプレフィックスとして使用するすべてのオブジェクトの未完了の UploadID を削除できます。

bucket1 に以下のオブジェクトが含まれている場合を仮定します。

```
$ ossutil ls oss://bucket1 -a
LastModifiedTime Size(B) StorageClass ETAG ObjectName
2015-06-05 14:06:29 +0000 CST 201933 Standard 7E2F4A7F1AC9D2F0996E8332D5EA5B41
oss://bucket1/dir1/obj11
2015-06-05 14:36:21 +0000 CST 241561 Standard 6185CA2E8EB8510A61B3A845EAFE4174 oss://bucket1/obj1
2016-04-08 14:50:47 +0000 CST 6476984 Standard 4F16FDAE7AC404CEC8B727FCC67779D6
oss://bucket1/sample.txt
Object Number is: 3
InitiatedTime UploadID ObjectName
2017-01-13 03:45:26 +0000 CST 15754AF7980C4DFB8193F190837520BB oss://bucket1/obj1
2017-01-13 03:43:13 +0000 CST 2A1F9B4A95E341BD9285CC42BB950EE0 oss://bucket1/obj1
2017-01-13 03:45:25 +0000 CST 3998971ACAF94AD9AC48EAC1988BE863 oss://bucket1/obj2
2017-01-20 11:16:21 +0800 CST A20157A7B2FEC4670626DAE0F4C0073C oss://bucket1/tobj
UploadId Number is: 4
0.191289(s) elapsed
```

obj1 の 2 つの UploadID を削除します。

```
./ossutil rm -m oss://bucket1/obj1
Succeed: Total 2 uploadIds. Removed 2 uploadIds.
1.922915(s) elapsed
```

obj1 と obj2 の 3 つの UploadID を削除します。

```
./ossutil rm -m oss://bucket1/obj1
Succeed: Total 4 uploadIds. Removed 4 uploadIds.
1.922915(s) elapsed
```

obj1 に加え、obj1 と obj2 の 3 つの UploadID を同時に削除します。

```
./ossutil rm oss://dest1/a -a -r -f
Do you really mean to remove recursively objects and multipart uploadIds of oss://dest1/a(y or N)? y
Succeed: Total 1 objects, 3 uploadIds. Removed 1 objects, 3 uploadIds.
```

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。

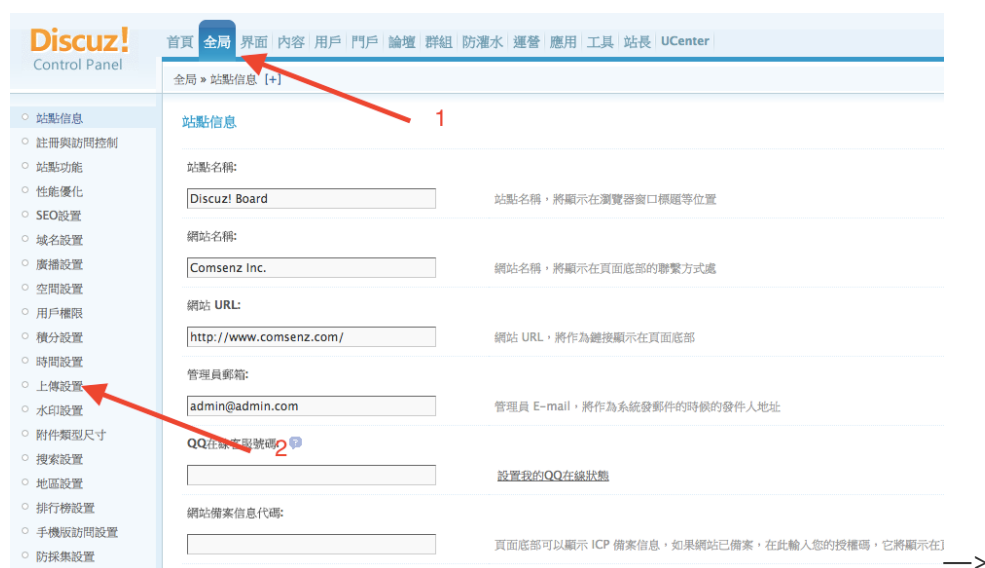
準備

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。

ブラウザで、画像の URL が **http://test-hz-jh-002.oss-cn-hangzhou.aliyuncs.com/forum/201512/18/171012mzvku2z3na2w2wa.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

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する必要があります。
- 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

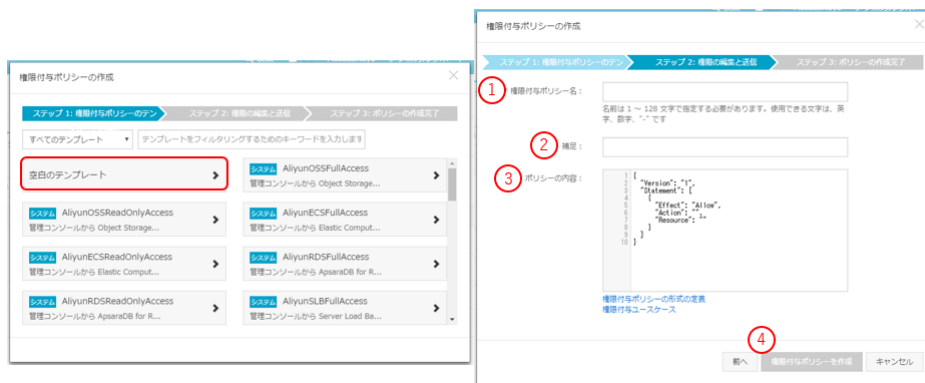
Resource Access Management (RAM) は、リソースアクセスを制御するための Alibaba Cloud のサービスです。ポリシーを作成することによって、共有読み取りアカウントを作成できます。ユーザーはこのアカウントを使用して FTP ツールにログオンし、ファイルを読むことができます。

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。

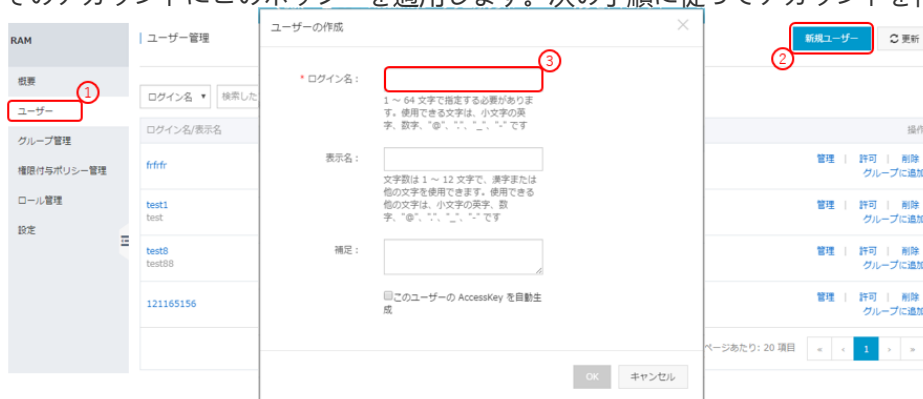
```
{
  "Version": "1",
  "Statement": [
    {
      "Action": [
        "oss:GetObject",
        "oss:HeadObject"
      ],
      "Resource": [
        "acs:oss:*:*:*:*:*:*:*:*:test-hz-john-001/*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "oss:ListObjects",
        "oss:GetBucketAcl",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

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インします。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_1.80.3_ubuntu16.04_amd64.deb

Ubuntu 14.04 (x64)	ossfs_1.80.3_ubuntu14.04_amd64.deb
CentOS 7.0 (x64)	ossfs_1.80.3_centos7.0_x86_64.rpm
CentOS 6.5 (x64)	ossfs_1.80.3_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、こちらを参照してください。

リリースログ

こちら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

概要

osscmdは、Python 2.xを利用するコマンドラインツールであり、バケットの管理やファイル管理を提供します。osscmdの代わりにossutilを使用することをお勧めします。

アプリケーションのシナリオ

- APIの開発とデバッグ。たとえば、特定の形式の要求を送信し、複数の部分のアップロードを段階的に実行します。
- コンソールが利用できない場合のバケット設定（例：logging / website / lifecycle）。

制限事項

- osscmdはPython 2.5 / 2.6 / 2.7をサポートしていますが、Python 3.xはサポートしていません。
- osscmdは、Python SDK V0.xに基づいて開発されました。このバージョンはすでにメンテナンスされていません。現在、Python SDK V2.xxはサポートされています。
- osscmdは、新規機能が追加される予定はありません。デバッグだけをサポートします。

ossutilの代わりにossutilを使用することをお勧めします。ossutilには次の利点があります。

- Windows / Linux / Macをサポートします。
- 簡単なインストールと優れたパフォーマンスを備えたGo SDKに基づいて実装されています。
- 簡単なコマンドと豊富なヘルプ情報を提供しています。

環境要件

Python SDKには、Python対応の環境が必要です。Pythonバージョン：バージョン2.5からバージョン2.7。SDKはWindowsとLinuxに適用されますが、Python3.0はSDKバージョン2.xと完全に互換性がないため、SDKはPython3.0以降をサポートしていません。

Pythonのインストール後：

- Linuxシェルに **python** を入力し、Enterキーを押してPythonのバージョンを表示します。

```
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cmdにInput **python** を入力し、Enterキーを押してPythonのバージョンを表示します。

```
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.
>>>
```

上記は、Pythonが正常にインストールされたことを示しています。

例外：Windowsのcmdに **python** を入力してEnterキーを押すと、**内部または外部のコマンドではありません**というプロンプトが表示されます。そのような場合は、**環境変数** > **Path** の設定を確認し、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 official website**からインストーラを入手できます。このウェブサイトでは、Pythonをインストールして使用するための手順とガイダンスがあります。

インストールと使用法

Python SDKから、ファイルのダウンロード方法が表示されます。ダウンロードしたPython SDKを osscmd のディレクトリに解凍し、python osscmd + operationを実行します。例えば、オブジェクトをバケットアップロード：

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

osscmdであることに注意してください。oss://bucket やoss://bucket/objectはバケットやオブジェクトを示すために使われます。oss://は意味のないリソースを示す単なる方法です。

詳細なコマンドリストが必要な場合は、python osscmdと入力してください。

詳細なパラメータリストの指示が必要な場合は、python osscmd helpと入力してください。

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ッケージを展開すると、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
```

注意: md5sum コマンドは Windows ではなく Linux 上で使用されます。

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
```

注意: md5sum コマンドは Windows ではなく Linux 上で使用されます。

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

```
$osscli 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

```
$osscli 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

```
$osscli 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

```
$osscli 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>
```

ロギングの使用

ロギングを設定する

```
$osscli putlogging oss://mybucket oss://myloggingbucket/mb
```

ロギングを取得する

```
$osscli getlogging oss://mybucket
```

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 Reference](#)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
```

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 --marker=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
```

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。

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。	<code>ossprobe --download -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName</code>
指定したファイルをダウンロードします。	<code>ossprobe --download -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -o ObjectName</code>
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。	<code>ossprobe --download -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -o ObjectName -t tmp/example.txt</code>
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。	<code>ossprobe --download -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -a Addr</code>

注意：

- 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。	<code>ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName</code>
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。	<code>ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -o ObjectName -m append</code>
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。	<code>ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -o ObjectName -m multipart</code>
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。	<code>ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -o ObjectName -m multipart -s src</code>
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。	<code>ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -m multipart -s src -o example.txt</code>
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。	<code>ossprobe --upload -i AccessKeyId -k AccessKeySecret -p EndPoint -b BucketName -a Addr</code>

注意: 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。

```
D:\tw108174\workspace\1111\src>ossprobe --download -i xxxxxxxx -k xxxxxxxx -p xxxxxxxx -b xxxxxxxxxxxx
```

- 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。

```
[admin@ss1c30376-42eqq /home/admin/tianwei/gofile]
$./ossprobe --upload -i xxxxxxxxxxxx -k xxxxxxxxxxxx -p xxxxxxxxxxxx -b xxxxxxxxxxxx
```

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。

(注意: 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 traceroute、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

移行ツール

概要

OssImport 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たデータを OSS に移行できます。以下の特徴があります：

- ローカルドライブ、Qiniu、Baidu BOS、AWS S3、Azure Blob、Youpai Cloud、Tencent Cloud COS、Kingsoft KS3、HTTP、OSSなどの豊富なデータソースをサポートし、必要に応じて拡張できます。

- 再開可能なデータ転送をサポートします。
- トラフィック制御をサポートします。
- 指定された時刻または指定された接頭辞の後にオブジェクトを移行することをサポートします。
- パラレルデータのアップロードとダウンロードをサポートします。
- スタンドアロンモードと分散モードをサポートします。スタンドアロンモードは展開と使用が容易で、分散モードは大規模なデータ移行に適しています。

アーキテクチャー

OssImport は、下の図に示すように、**master-worker** 分散アーキテクチャに基づいています。

```

Master ----- Job ----- Console
|
|
| TaskTracker
|_____
|Task | Task | Task
|||
Worker Worker Worker

```

Job : ユーザーが送信したデータ移行ジョブ。ユーザーの場合、1つのジョブは1つの構成ファイル **job.cfg** に対応します。

Task : ジョブは、データサイズとファイル数によって複数のタスクに分けられます。各タスクは、ファイルの一部を移行します。ジョブをタスクに分割するための最小単位はファイルです。1つのファイルを複数のタスクに分割することはできません。

OssImport ツールモジュールは、次の表のとおりです。

ロール	説明
Master	マスタは、データサイズとファイル数によってジョブを複数のタスクに分割する役割を担います。sys.properties でデータサイズとファイル数を設定することができます。 ジョブを複数のタスクに分割する詳細なプロセスは、次のとおりです。 - マスターノードは、マイグレーションするファイルの完全リストをローカル/他のクラウドストレージデバイス。 - マスタは、データサイズとファイル数によってタスクにフルファイルリストを分割し、各タスクはファイルの一部の移行または検証を行います。
Worker	- ワーカーはファイルマイグレーションとタスクのデータ検証を担当します。特定のファイルをデータソースから取り出し、ファイルを指定されたディレクトリにOSSにアップロードします。job.cfg または local.job.cfg で、移行する

	データソースとOSS構成を指定できます。 - ワーカーデータ移行では、トラフィックの制限と並行タスクの数の指定がサポートされています。 sys.properties で設定を構成できます。
TaskTracker	TaskTracker は Tracker と略されています。タスクの分散とタスクの状態の追跡を行います。
Console	コンソールは、ユーザーと対話し、コマンド表示結果を受け取る役割を担います。 deploy、start、stop などのシステム管理コマンド、および submit、retry、clean などのジョブ管理コマンドをサポートしています。

展開のモード

OssImport には、**スタンドアロンモード**と**分散モード**の2つの展開モードがあります。スタンドアロンモードは、データが **30 TB 未満**の小規模なデータ移行には十分です。大規模なデータ移行には分散モードを推奨します。

スタンドアロン

master、worker、tracker、および console は、同じマシン上で実行されます。システムには 1 つの worker しかありません。スタンドアロンモードの展開と実行をカプセル化して最適化し、スタンドアロンの展開と実行は非常に簡単です。スタンドアロンモードでは、master、worker、tasktracker、および console モジュールは **ossimport2.jar** にパッケージ化されています。

スタンドアロンモードのファイル構造は次のとおりです。

```
ossimport
├── bin
│   └── ossimport2.jar # The JAR including master, worker, tracker, and console modules
├── conf
│   ├── local_job.cfg # Standalone job configuration file
│   └── sys.properties # Configuration file for the system running parameters
├── console.bat # Windows command line, which can execute distributed call-in tasks
├── console.sh # Linux command line, which can execute distributed call-in tasks
├── import.bat # The configuration file for one-click import and execution in Windows is the data migration job
               configured in conf/local_job.cfg, including start, migration, validation, and retry
├── import.sh # The configuration file of one-click import and execution in Linux is the data migration job
              configured in conf/local_job.cfg, including start, migration, validation, and retry
├── logs # Log directory
└── README.md # Description documentation. We recommend that you carefully read the documentation before
using this feature
```

注意：

- **import.bat** または **import.sh** ファイルは、ワンクリックでインポートするスクリプトで、**local.job.cfg** への変更が完了した直後に実行できます。

- **console.bat** または **console.sh** はコマンドラインツールであり、コマンドの分散実行に使用できます。
- スクリプトやコマンドを `ossimport`ディレクトリ、つまり`*.bat/*.sh`と同じレベルのディレクトリに実行します。

分散

分散モードでは、データ移行のために複数のワーカーノードを開始できます。タスクはワーカーノードに均等に割り当てられ、1つのワーカーノードは複数のタスクを実行できます。1台のマシンでは、1つのワーカーノードしか開始できません。マスターは `workers` で設定された最初のワーカーノードと同時に起動され、`tasktracker` と `console` もマシン上で実行されます。

分散モードのファイル構造は次のとおりです。

```
ossimport
├── bin
│   ├── console.jar # The JAR package of the console module
│   ├── master.jar # The JAR package of the master module
│   ├── tracker.jar # The JAR package of the tracker module
│   └── worker.jar # The JAR package of the worker module
├── conf
│   ├── job.cfg # The template of the job configuration file
│   ├── sys.properties # Configuration file for the system running parameters
│   └── workers # Worker list
├── console.sh # The command line tool. Currently it only supports Linux
├── logs # Log directory
└── README.md # Description documentation. We recommend that you carefully read the documentation before using the feature
```

注意：分散コマンドラインツール **console.sh** は現在 Linux のみをサポートしており、Windows はサポートしていません。

設定ファイル

スタンドアロンモードでは、**sys.properties** と **local_job.cfg** の2つの設定ファイルがあります。分散モードでは、**sys.properties**、**local_job.cfg**、**workers** の3つの設定ファイルがあります。具体的には、**local_job.cfg** と **job.cfg** は同一ですが、名前は異なります。**workers** ファイルは分散環境専用です。

sys.properties

システム実行パラメータ。

フィールド	意味	説明
<code>workingDir</code>	作業ディレクトリ	- ツールキットが抽出された後のディレクトリです。

		- スタンドアロンモードでこのオプションを変更しないでください。 - 各マシンの作業ディレクトリは、分散モードで同じでなければなりません。
workerUser	ワーカーマシンの SSH ユーザー名	- privateKeyFile を構成している場合、privateKeyFile が優先順位で使用されます。 - privateKeyFile が構成されていない場合、workerUser/workerPassword の組み合わせが使用されます。 - スタンドアロンモードでこのオプションを変更しないでください。
workerPassword	ワーカーマシンの SSH ユーザー・パスワード	このオプションはスタンドアロン・モードでは変更しないでください。
privateKeyFile	公開鍵のファイルパス	- SSHチャンネルを確立している場合は、公開鍵ファイルのパスを指定できます。それ以外の場合は、空のままにします。 - privateKeyFile を設定している場合は、privateKeyFile が優先されます。 - privateKeyFile が設定されていない場合、workerUser/workerPassword が使用されます。 - スタンドアロンモードでこのオプションを変更しないでください。
sshPort	SSH ポート	デフォルト値は22です。通常は変更する必要はありません。スタンドアロンモードでこのオプションを変更しないでください。
workerTaskThreadNum	ワーカーがタスクを実行する最大スレッド数	- このパラメータは、マシンのメモリとネットワークに関連しています。推奨値は 60 です。 - 物理マシンの場合は、値を 150 に増やすことができます。ネットワーク帯域幅がすでに上限に達している場合は、さらに値を大きくしないでください。 - ネットワークが不十分な場合

		は、適切な値（たとえば 30）を減らします。この方法で、要求からの多数の要求のタイムアウトを回避できます。
workerMaxThroughput (KB/s)	ワーカーノード上のデータ移行トラフィックの上限	この値は、トラフィックを制限します。デフォルト値 0 は、トラフィック制限が課せられていないことを示します。
dispatcherThreadNum	タスク配布用のスレッド数とトラッカーのステータス確認	基本的にはデフォルト値を使用します。特別な要件がない場合は、デフォルト値を変更する必要はありません。
workerAbortWhenUncaughtException	不明なエラーが発生した場合にスキップするか、中止するかどうか	不明なエラーはデフォルトでスキップされます。
workerRecordMd5	メタデータ x-oss-meta-md5 を使用して、OSS 内の移行されたファイルの MD5 値を記録するかどうか。デフォルト設定は NO	主に MD5 を使用したファイルデータ検証に使用されます。

job.cfg

データ移行ジョブの構成。 **local.job.cfg** と **job.cfg** オプションは、名前以外は同じです。

フィールド	意味	説明
jobName	ジョブ名、文字列	- ジョブの一意的識別子。命名規則は[a-zA-Z0-9_-]{4,128}です。それは、異なる名前の複数のジョブの提出をサポートします。 - 別のジョブと同じ名前のジョブをサブミットすると、ジョブがすでに存在することを知らせるメッセージが表示されます。元のジョブを名前でクリーンアップする前に、同じ名前のジョブをサブミットすることはできません。
jobType	ジョブの種類、文字列	2つのタイプがあります： import と audit。デフォルト値は import です。 - import：データ移行を実行し、移行されたデータの整合性を検証します。 - audit：データの一貫性のみを検証します。
isIncremental	増分マイグレーション・モード	- デフォルト値：False。

	を boolean 値として使用可能にするかどうか	- 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再スキャンされ、OSSに同期されます。
incrementalModeInterval	インクリメンタル・モードでの同期間隔、整数値。単位：秒	isIncremental = trueの場合に有効です。設定可能な最小間隔は 900 秒です。3,600 秒より小さい値に設定することは推奨されません。これは、多数の要求が無駄になり、システムのオーバーヘッドが増加するためです。
importSince	この時間値よりも後でデータをマイグレーションします。これは整数値です。単位：秒	- この時刻の値は UNIX のタイムスタンプ。つまり 1970 年 1 月 1 日の UTC00 : 00 以降の秒数です。
srcType	同期ソース・タイプ、文字列。大文字と小文字を区別する	現在、このパラメータは、local、oss、qiniu、bos、ks3、s3、youpai、HTTP、cos、azure などの 10 種類をサポートしています。 - local：ローカルファイルから OSS にデータを移行します。このオプションの srcPrefix を入力するだけで済みます。srcAccessKey、srcSecretKey、srcDomain、および srcBucket を入力する必要はありません。 - oss：あるバケットから別のバケットにデータを移行します。 - qiniu：QiniuクラウドストレージからOSSにデータを移行します。 - bos：BaiduクラウドストレージからOSSにデータを移行します。 - ks3：KingsoftのクラウドストレージからOSSにデータを移行します。 - s3：AWS S3からOSSにデータを移行します。 - youpai：Youpai CloudからOSSにデータを移行します。 - HTTP：提供されたHTTPリン

		クリストを使用してOSSにデータを移行します。 - cos : TencentクラウドストレージCOSからOSSにデータを移行します。 - azure : Azure BLOBからOSSにデータを移行します。
srcAccessKey	ソース AccessKey、文字列	srcType が oss、qiniu、baidu、ks3 または s3 に設定されている場合は、データソースの AccessKey を入力します。 - またはローカルと HTTP のタイプの場合、このオプションは空のままにすることができます。 - Youpai タイプと Azure タイプの場合は、AccountName を入力します。
srcSecretKey	ソース SecretKey、文字列	srcType が oss、qiniu、baidu、ks3 または s3 に設定されている場合は、データソースの SecretKey を入力します。 ローカルおよび HTTP タイプの場合、このオプションは空のままにできます。 - youpai : オペレータのパスワードを入力します。 - azure : AccountKey。
srcDomain	ソースエンドポイント	srcType がローカルまたは HTTP に設定されている場合、この設定項目は不要です。 - oss : コンソールから取得したドメイン名。これはバケット接頭辞のない第2レベルのドメイン名です。ドメイン名リストに完全なリストがあります。 - qiniu : 該当のドメイン名バケット BaiduのBOSのドメイン名は、http://bj.bcebos.comまたはhttp://gz.bcebos.com のようなドメイン名になります。 - Kingsoft ks3 : http://kss.ksyun.comも

		しくはhttp://ks3-cn-beijing.ksyun.comもしくはhttp://ks3-us-west-のようなドメイン名になります。様々な地域のS3ドメイン名とAWS S3ドメイン名は、S3エンドポイントで見つけることができます。 - youpai : Youpai Cloudのドメイン名。例えば光回線のhttp://v0.api.upyun.com、または通信回線のhttp://v1.api.upyun.com、または中国聯通または中国ネットコム回線のhttp://v2.api.upyun.comまたは中国携帯電話または中国 Railcom の http://v3.api.upyun.comがあります。 - cos : バケットなど中国南部などテンセントクラウドのリージョン : 南中国 : gz、北中国 : tj、東中国 : shがあります。 - azuru : core.chinacloudapi.cnなどの EndpointSuffixがあります。
srcBucket	送信元バケットまたはコンテナの名前	srcType がローカルまたはHTTP に設定されている場合、この設定項目は不要です。 azure : Azure BLOBのコンテナ名を入力し、他のバケット名を入力します。
srcPrefix	ソースプレフィックス、文字列。デフォルト値は空です。	srcTypeが local に設定されている場合は、ローカルディレクトリを完全に入力し、/、c:/example/または/data/example/となります。 srcTypeは、OSS、qiniu、BOS、KS3、youpaiまたはS3がある場合は、値は、オブジェクトのプレフィックスがなく、すべてのオブジェクトはsrcPrefixが空の状態と同期されます。
destAccessKey	宛先AccessKey、文字列	OSS AccessKeyIDを表示するには、コンソールにログインします。
destSecretKey	宛先SecretKey、文字列	OSS AccessKeySecretを表示するには、コンソールにログイン

		ンします。
destDomain	宛先エンドポイント、文字列	コンソールから取得します。これはバケット接頭辞のない第2レベルのドメイン名です。完全なリストはドメイン名リストにあります。
destBucket	宛先バケット、文字列	OSSバケット名。 / で終わる必要はありません。
destPrefix	宛先プレフィックス、文字列。 デフォルト値は空です。	- 宛先プレフィックス。デフォルト値は空です。 オブジェクトが宛先バケット置かれている場合 - OSS上の特定のディレクトリにデータを同期させる場合は、data/in/oss/のようなプレフィックスになります。 - OSSはオブジェクトプレフィックスとして / をサポートしていないので、 / で始まる destPrefixを設定しないでください。 - パス srcPrefix + relativePath のローカルファイルは、OSSの destDomain / destBucket / destPrefix + relativePath に移行されます。 - パス srcDomain / srcBucket / srcPrefix + relativePath 内のクラウド上のオブジェクトは、OSS上の destDomain / destBucket / destPrefix + relativePath に移行されます。
taskObjectCountLimit	タスク内のファイルの最大数 (整数)。デフォルト値は 10,000 です。	この構成オプションは、実行されたジョブの並行性に影響します。一般的には、ファイルの総数/ワーカーの総数/マイグレーションスレッドの数 (workerTaskThreadNum) に設定され、最大数は 50,000 に設定されます。ファイルの総数が不明な場合は、デフォルト値を使用してください。
taskObjectSizeLimit	タスク内の最大データサイズ、整数。単位：バイト。デフォルト値は 1 GB です。	この構成オプションは、実行されたジョブの並行性に影響します。一般的に、設定は合計データサイズ/ワーカーの総数/マイグレーションスレッドの数 (workerTaskThreadNum) に設定されます。合計データ

		・サイズが不明な場合は、デフォルト値を使用してください。
isSkipExistFile	データ移行中に既存のオブジェクトをスキップするかどうか、ブール値。	trueに設定すると、オブジェクトはサイズとLastModifiedTimeに従ってスキップされます。falseに設定すると、既存のオブジェクトが上書きされます。デフォルト値はfalseです。jobTypeが監査に設定されている場合、このオプションは無効です。
scanThreadCount	パラレルファイルスキャンのスレッド数（整数）。デフォルト値は1です。	この構成オプションは、ファイルのスキャン効率に関連しています。特別な要件がない場合は、構成を変更しないでください。
maxMultiThreadScanDepth	パラレルスキャンのディレクトリの最大許容深度（整数）。デフォルト値は1です。	デフォルト値1は、最上位ディレクトリのパラレルスキャンを示します。 特別な要件がない場合は、この構成を変更しないでください。値が大きすぎると、ジョブが正常に実行されないことがあります。
appId	Tencent COSのappId、整数。	srcTypeが cos に設定されている場合に有効です。
httpListFilePath	HTTPリストファイルの絶対パス、文字列	- srcTypeがHTTPに設定されている場合に有効です。ソースがHTTPリンクアドレスの場合、 c:/example/http.list のように、HTTPリンクアドレスをコンテンツとして持つファイルの絶対パスを指定する必要があります。 ファイルは、アップロード後のOSS上の接頭辞と相対パスを表すスペースで区切られた2つのカラムに分けられます。例えば、 c:/example/http.list のように、 http: /mingdi-hz.oss-cn-hangzhou.aliyuncs.com/aa/bb.jpg http://mingdi-hz.oss-cn-hangzhou.aliyuncs.com/cc/dd.jpg のオブジェクト名それらがOSSに移行された後の2行は、それぞれ destPrefix + bb.jpg および destPrefix + cc/dd.jpg になります。

Workers

ワーカーは分散モード専用で、すべての IP アドレスは次のような行です。

```
192.168.1.6
192.168.1.7
192.168.1.8
```

注意：

- 前の設定では、マスタ、ワーカー、タスクトラッカーは 192.168.1.6 で起動され、コンソールもマシン上で実行する必要があります。
- 複数のワーカーモードのユーザー名、ログオンモード、作業ディレクトリが同じであることを確認します。

実行環境

Java 1.7 以降

ダウンロード

スタンドアロン展開は Linux と Windows をサポートします。

スタンドアロン展開用のツールをダウンロードします：ossimport-2.2.4.zip

ツールをローカルディレクトリにダウンロードし、ツールまたは unzip コマンドを使用してファイルを解凍します。解凍後のファイル構造は次のとおりです。

```
ossimport
├── bin
│   └── ossimport2.jar # The JAR including master, worker, tracker and console modules
├── conf
│   ├── local_job.cfg # The job configuration file
│   └── sys.properties # Configuration file of the system running parameters
├── console.bat # Windows command line, which can execute distributed call-in tasks
├── console.sh # Linux command line, which can execute distributed call-in tasks
├── import.bat # The configuration file for one-click import and execution in Windows is the data migration job
               configured in conf/local_job.cfg, including start, migration, validation and retry
├── import.sh # The configuration file for one-click import and execution in Linux is the data migration job
               configured in conf/local_job.cfg, including start, migration, validation and retry
├── logs # Log directory
└── README.md # Description documentation. We strongly recommend you carefully read the documentation
               before using the feature
```

設定

スタンドアローン版には `conf/sys.properties` と `conf/local_job.cfg` という 2 つの設定ファイルがあります。設定項目の詳細については、紹介の章を参照してください。

`conf/sys.properties` で `workingDir`、`workerUserName`、`workerPassword`、`privateKeyFile` の設定項目を変更しないでください。`conf/local_job.cfg` の名前と場所を変更しないでください。`jobName` 設定項目を変更しないでください。他の項目を適切に設定します。

注意：ジョブを送信する前に、**`sys.properties`** および **`local_job.cfg`** のパラメータを確認してください。ジョブのパラメーターは、ジョブの実行後に変更することはできません。

実行

スタンドアロンモードでは、データ移行ジョブにワンクリックインポートとステップバイステップ実行という 2 つの実行モードがあります。

ワンクリックでインポートすると、すべてのステップがカプセル化され、スクリプトのプロンプトに従ってデータの移行を完了できます。**新しいユーザーにワンクリックでインポートすることを強くおすすめします。**

ステップバイステップの実行には、開始サービスの実行、ジョブのサブミット、および失敗したタスクの再試行が含まれます。

スタンドアロンモードの設定ファイルは **`conf/local_job.cfg`** です。データ移行前の実際のニーズに基づいてジョブパラメータを変更します。デフォルトのジョブ名 **`local_test`** は変更しないでください。

ワンクリックでインポート

ワンクリックインポートを実行するには、Windows の **`cmd.exe`** で `import.bat` を実行し、Linux では `bash import.sh` を実行します。

以前にこのジョブを実行した場合は、最後のブレークポイントからジョブを続行するか、新しい同期ジョブを実行するかを尋ねられます。新しいデータマイグレーションジョブを開始した場合、または同期ソースの終了/送信先を変更した場合は、同期ジョブを再実行してください。

Windows でジョブが開始されると、進行中の同期ジョブとログを示す新しい `cmd` ウィンドウが表示されます。古いウィンドウのジョブステータスは 10 秒ごとに更新されます。データ移行プロセス中にこれら 2 つのウィンドウを閉じないでください。Linux では、上記のプロセスはバックグラウンドで実行されます。

ジョブが完了すると、タスクが失敗した場合、再試行するかどうかを尋ねられます。**`y`** を入力して

再試行するか、**n** を入力してこの手順をスキップして終了してください。

アップロードが失敗した理由を調べるには、ファイル `master/jobs/local_test/failed_tasks//audit.log` を実行し、障害の原因を確認します。

ステップバイステップ実行

同じ名前のジョブをクリアします。以前に同じ名前のジョブを実行していて、ジョブを再度実行したい場合は、最初に同じ名前のジョブをクリアしてください。ジョブを実行したことがない、または失敗したジョブを再試行する場合は、`clear` コマンドを実行しないでください。Windows では、**cmd.exe** で `console.bat clean` を実行してください。Linux では、`bash console.sh clean` を実行してください。

データ移行ジョブを送信します。OssImport は、同じ名前のジョブの送信をサポートしていません。同じ名前のジョブがある場合は、最初に同じ名前のジョブをクリアします。サブミットされたジョブの構成ファイルは `conf/local_job.cfg` で、デフォルトのジョブ名は **local_test** です。ジョブを投入するには、Windows の **cmd.exe** で `console.bat submit` を実行し、Linux では `bash console.sh submit` を実行してください。

サービスを開始します。Windows では **cmd.exe** で `console.bat start` を実行し、Linux では `bash console.sh start` を実行します。

ジョブのステータスを表示します。Windows では **cmd.exe** で `console.bat stat` を実行し、Linux では `bash console.sh stat` を実行します。

失敗したタスクを再試行します。ネットワークの問題やその他の原因により、タスクが失敗することがあります。失敗したタスクだけが再試行されます。Windows では **cmd.exe** で `console.bat retry` を実行し、Linux では `bash console.sh retry` を実行します。

サービスを停止します。Windows では **%JAVA_HOME%/bin/java.exe** ウィンドウを閉じ、Linux では `bash console.sh stop` を実行してください。

注意：特別な要件がない場合は、データの移行にワンクリックインポートを使用することをお勧めします。

一般的な失敗の原因

アップロードプロセス中に、ソースディレクトリのファイルが変更されました。この原因は

log/audit.log の **SIZE_NOT_MATCH**エラーで表示されます。この場合、古いファイルは正常にアップロードされましたが、変更は OSS に同期されていません。

アップロード処理中にソースファイルが削除され、ダウンロードに失敗しました。

ソースファイル名が OSS の命名規則に準拠していない (ファイル名が / で始まらないか、または空である) ことができず、アップロードに失敗しました。

データソースファイルをダウンロードできませんでした。

プログラムが予期せず終了し、ジョブの状態が **Abort** です。これが発生した場合、SB Cloud サポートセンター にお問い合わせください。

ジョブのステータスとログ

ジョブがサブミットされると、マスターはジョブをタスクに分割し、ワーカーはタスクを実行し、トラッカーはタスクのステータスを収集します。ジョブが完了すると、ossimport ディレクトリーには以下が含まれます。

```
ossimport
├── bin
│   └── ossimport2.jar # The standalone version JAR
├── conf
│   ├── local_job.cfg # The job configuration file
│   └── sys.properties # Configuration file of the system running parameters
├── console.sh # The command line tool
├── import.sh # One-click import script
├── logs
│   ├── import.log # Archive logs
│   ├── job_stat.log # Job status record
│   ├── ossimport2.log # Running log of the standalone version
│   └── submit.log # Job submission record
├── master
│   ├── jobqueue # Store jobs that have not been fully split
│   ├── jobs # Store the job running status
│   ├── local_test # Job name
│   ├── checkpoints # The checkpoint record of the master splitting the job to tasks
│   │   ├── 0
│   │   └── 034DC9DD2860B0CFE884242BC6FF92E7.cpt
│   ├── dispatched # Tasks that have been assigned to the workers but haven't been fully run
│   │   └── localhost
│   ├── failed_tasks # Tasks that failed to run
│   ├── pending_tasks # Tasks that have not been assigned
│   ├── succeed_tasks # Tasks that run successfully
│   └── A41506C07BF1DF2A3EDB4CE31756B93F_1499744514501@localhost
├── audit.log # The task running log. You can view the error causes in the log
├── DONE # Mark of successful tasks
└── error.list # The task error list. You can view the error file list
```

```

| └─ STATUS # The task status marker file. The content is Failed or Completed
| └─ TASK # The task description information
└─ worker # Status of the task being run by the worker. After running, tasks are managed by the master
    └─ jobs
        └─ local_test
            └─ tasks

```

注意：

- ジョブ実行情報については、**log/ossimport2.log** を参照してください。
- タスクの失敗原因については、
master/jobs/\${JobName}/failed_tasks/\${TaskName}/audit.log を参照してください。
- 失敗したタスクファイルについては、
master/jobs/\${JobName}/failed_tasks/\${TaskName}/error.list を参照してください。

FAQs

FAQs を参照してください。

ダウンロード

現在、分散配置は Linux のみをサポートしており、Windows はサポートしていません。

分散配置のためのツールをダウンロードしてください：**ossimport-2.2.4.tar.gz**

ツールをローカルディレクトリにダウンロードし、コマンド `tar -zxvf ossimport-2.2.4.tar.gz -C $HOME/ossimport` を実行してファイルを解凍します。解凍後のファイル構造は次のとおりです。

```

ossimport
├─ bin
| └─ console.jar # The JAR package of the console module
| └─ master.jar # The JAR package of the master module
| └─ tracker.jar # The JAR package of the tracker module
| └─ worker.jar # The JAR package of the worker module
├─ conf
| └─ job.cfg # The template of the job configuration file
| └─ sys.properties # Configuration file of the system running parameters
| └─ workers # Worker list
├─ console.sh # The command line tool. Currently it only supports Linux
├─ logs # Log directory
└─ README.md # Description documentation. Read it carefully before use

```

注意：

- **OSS_IMPORT_HOME** : OssImport のルートディレクトリ。unzip コマンドのデフォルトディレクトリは **\$HOME/ossimport** です。コマンド `export OSS_IMPORT_HOME=` を実行するか

- 、システム構成ファイル `$HOME/.bashrc` を変更してディレクトリを設定します。
- `OSS_IMPORT_WORK_DIR` : `OssImport` 作業ディレクトリ。 `conf/sys.properties` の設定項目 `workingDir` でディレクトリを指定することができます。推奨値は `$HOME/ossimport/workdir` です。
- `OSS_IMPORT_HOME` または `OSS_IMPORT_WORK_DIR` の絶対パスを使用します。たとえば `/home//ossimport` または `/home//ossimport/workdir` と入力します。

設定

分散版には `conf/sys.properties`、`conf/job.cfg`、`conf/workers` という 3 つの設定ファイルがあります。構成項目の説明については、紹介の章を参照してください。

`conf/job.cfg` : 分散モードのジョブの設定ファイルテンプレート。データ移行の前に実際のパラメータに従って値を変更します。

`conf/sys.properties` : システム実行パラメータの設定ファイル (作業ディレクトリやワーカー実行パラメータなど) 。

`conf/workers` : ワーカーリスト。

注意 :

- ジョブを送信する前に、 `sys.properties` および `job.cfg` のパラメータを確認します。ジョブのパラメーターは、ジョブの実行後に変更することはできません。
- サービスを開始する前にワーカーリスト `ワーカー` を決定します。サービスの開始後、ワーカーは追加または削除することはできません。

実行

コマンドの実行

分散配置では、ジョブ実行の一般的な手順は次のとおりです。

ジョブ構成ファイルを変更します。

サービスを展開します。

Linuxで `bash console.sh deploy`を実行してください。

注意:設定ファイル **conf/job.cfg** および **conf/workers** が展開前に変更されていることを確認してください。

同じ名前のジョブをクリアします。

前に同じ名前のジョブを実行していて、ジョブを再度実行する場合は、最初に同じ名前のジョブをクリアします。ジョブを実行したことがない、または失敗したジョブを再試行する場合は、**clear**コマンドを実行しないでください。Linuxで **bash console.sh clean job_name** を実行してください。

データ移行ジョブを送信します。

OssImportは、同じ名前のジョブの送信をサポートしていません。同じ名前のジョブがある場合は、まず**clean** コマンドを使用して、同じ名前のジョブを先にクリーニングします。ジョブをサブミットするには、ジョブ構成ファイルを指定する必要があります。ジョブの設定ファイルテンプレートは、**conf/job.cfg**にあります。テンプレートに基づいて設定を変更することをお勧めします。Linuxで**bash console.sh submit [job_cfg_file]** `を実行し、設定ファイル **job_cfg_file**でジョブを送信してください。**job_cfg_file** はオプションのパラメータです。指定しない場合、パラメータは **\$OSS_IMPORT_HOME/conf/job.cfg** です。**\$OSS_IMPORT_HOME** は、デフォルトでは、**console.sh** ファイルが置かれているディレクトリです。

マイグレーションサービスを開始します。

Linuxで**bash console.sh start**を実行してください。

ジョブの状態を表示します。

Linuxで **bash console.sh stat**を実行してください。

失敗したタスクを再試行します。

ネットワークの問題やその他の原因により、タスクが実行に失敗することがあります。失敗したタスクだけが再試行されます。Linuxで**bash console.sh retry [job_name]**を実行してください。**job_name** はオプションのパラメーターで、**job_name**という名前のジョブのタスクを再試行することを指定します。**job_name** パラメーターが指定されていない場合、すべてのジョブの失敗したタスクが再試行されます。

マイグレーションジョブを停止します。

Linuxで **bash console.sh stop**を実行してください。

注意 :

- bash console.shパラメータにエラーがある場合、console.shは自動的にコマンドフォーマットを通知します。
- 設定ファイルのディレクトリとサブミットされたジョブの絶対パスを使用することをお勧めします。
- ジョブの設定 (job.cfg の設定項目) は、送信後に変更することはできません。

ジョブの失敗の一般的な原因

アップロードプロセス中に、ソースディレクトリのファイルが変更されました。この原因は **log/audit.log** の **SIZE_NOT_MATCH**エラーで表示されます。この場合、古いファイルは正常にアップロードされましたが、変更は OSS に同期されていません。

アップロード処理中にソースファイルが削除され、ダウンロードに失敗しました。

ソースファイル名が OSS の命名規則に合致していない (ファイル名が / で始まらないか空である) ため、OSS へのアップロードに失敗します。

データソースファイルのダウンロードに失敗しました。

プログラムが予期せず終了し、ジョブの状態が **Abort** です。これになった場合は、SB Cloud サポートセンターにお問い合わせください。

ジョブのステータスとログ

ジョブがサブミットされると、マスターはジョブをタスクに分割し、ワーカーはタスクを実行し、トラッカーはタスクのステータスを収集します。ジョブが完了すると、workdir ディレクトリには次の情報が格納されます。

```
workdir
├── bin
│   ├── console.jar # The JAR package of the console module
│   ├── master.jar # The JAR package of the master module
│   ├── tracker.jar # The JAR package of the tracker module
│   └── worker.jar # The JAR package of the worker module
├── conf
│   ├── job.cfg # The template of the job configuration file
│   ├── sys.properties # Configuration file of the system running parameters
│   └── workers # Worker list
├── logs
│   ├── import.log # Archive logs
│   ├── master.log # Master logs
│   ├── tracker.log # Tracker logs
│   └── worker.log # Worker logs
└── master
```

```

| |─ jobqueue # Store jobs that have not been fully split
| |─ jobs # Store the job running status
| |─ xxtoss # Job name
| |─ checkpoints # The checkpoint record that the master splits the job to tasks
| |   |─ 0
| |   |─ ED09636A6EA24A292460866AFDD7A89A.cpt
| |─ dispatched # Tasks that have been assigned to the workers but haven't been fully run
| |   |─ 192.168.1.6
| |─ failed_tasks # Tasks that failed to run
| |   |─ A41506C07BF1DF2A3EDB4CE31756B93F_1499348973217@192.168.1.6
| |   |─ audit.log # The task running log. You can view the error causes in the log
| |   |─ DONE # Mark of successful tasks. If the task fails, the mark is empty
| |   |─ error.list # The task error list. You can view the error file list
| |   |─ STATUS # The task status mark file. The content is Failed or Completed, indicating that the task failed or
succeeded
| |   |─ TASK # The task description information
| |─ pending_tasks # Tasks that have not been assigned
| |─ succeed_tasks # Tasks that run successfully
| |   |─ A41506C07BF1DF2A3EDB4CE31756B93F_1499668462358@192.168.1.6
| |   |─ audit.log # The task running log. You can view the error causes in the log
| |   |─ DONE # Mark of successful tasks
| |   |─ error.list # Task error list. If the task is successful, the list is empty
| |   |─ STATUS # The task status mark file. The content is Failed or Completed, indicating that the task failed or
succeeded
| |   |─ TASK # The task description information
| |─ worker # Status of the task being run by the worker. After running, tasks are managed by the master
| |   |─ jobs
| |   |─ local_test2
| |   |─ tasks
| |   |─ local_test_4
| |   |─ tasks

```

注意：

- ジョブの実行状況については、**logs/tracker.log** を参照してください。ログを実行しているワーカーの場合は、**logs/worker.log** を表示します。マスター実行ログについては、**logs/master.log** を参照してください。
- タスクの失敗原因については、**master/jobs/{JobName}/failed_tasks/{TaskName}/audit.log** を参照してください。
- 失敗したタスクファイルについては、**master/jobs/{JobName}/failed_tasks/{TaskName}/error.list** を参照してください。

FAQ

FAQ を参照してください。

この記事では主に OssImport の一般的なアプリケーションと典型的な要件の実装を紹介します。

スタンドアロンモードと分散モード

OssImport には、スタンドアロンモードと分散モードの2つの展開モードがあります。データサイズが 30 TB 未満の小規模なデータ移行では、スタンドアロンモードが向いています。分散モードは、大きなデータサイズの移行に推奨されます。

時刻別のトラフィック制限

ワーカーはトラフィック制限機能を提供します。設定ファイル `sys.properties` 内の `workerMaxThroughput` (KB/s) 項目を変更することにより、トラフィック制限を実装できます。この設定項目は即時に有効になりません。項目を変更した後にサービスを再起動する必要があります。分散配置モードでは、各作業者の `sys.properties` と `$OSS_IMPORT_WORK_DIR/conf` を変更し、サービスを再起動する必要があります。

`sys.properties` から `crontab` までの時間変更を実装してから、サービスを再起動して時間固有のトラフィック制限を実装することができます。

ワーカーの追加

ジョブを送信する前にワーカーリストを決定します。現在、OSS では、ワーカーを動的に追加することはできません。

移行なしのデータ検証

OssImport は、移行せずにデータ検証をサポートします。設定項目は、ジョブ設定ファイル `job.cfg` または `local_job.cfg` の `import` の代わりに `jobType = audit` となります。他の構成項目は、データ移行と同じです。

第三者のストレージサービスから OSS にシームレスに切替え

以下の手順に従って、他のストレージサービスから OSS にシームレスに切り替えることができます。

完全マイグレーション。現時点では、ビジネスはサードパーティのストレージサービス上で実行されています。データ移行 T1 の開始時刻をマークします。時刻は **Unix タイムスタンプ** 形式、つまり 1970 年 1 月 1 日 00:00 UTC からの秒数で指定する必要があります。date +%s コマンドで値を取得できます。

OSS イメージの back-to-source 機能を使用します。データの移行が完了したら、OSS コンソールの **バケット** の **Back-To-Source 設定** 機能を設定します。Back-To-Source アドレスは第三者ストレージです。

OSS への読み書きを切り替えます。この時点で、T1 より前のデータは OSS から読み取られ、T1 より後のデータは、Back-To-Source のイメージを使用して第三者サービスから読み取られ、新しいデータは完全に OSS に書き込まれます。

インクリメンタルマイグレーション。増分マイグレーションジョブの設定項目は、設定ファイル (**job.cfg** または **local.job.cfg**) の **importSince = T1** です。インクリメンタルマイグレーションは **T2** で完了します。

サードパーティストレージを削除します。 **T2** 以降は、すべてのビジネスの読み書きが OSS 上で行われ、サードパーティのストレージは履歴データのコピーとなります。それを保持するか、削除するかを決めることができます。OssImport はデータの移行と検証を担当し、データを削除しません。

注意：ステップ 4 でのインクリメンタルマイグレーションは、データマイグレーションのインクリメンタルモードを指していません。

データ移行のインクリメンタルモード

データ移行のインクリメンタルモードは、データ移行ジョブの開始後に最初にフルマイグレーションを実行してから、設定した間隔で増分マイグレーション操作を自動的に実行するプロセスを指します。最初のデータ移行ジョブは完全マイグレーションです。ジョブは、送信された直後に開始されます。以降のデータ移行ジョブは、設定された間隔ごとに 1 回開始されます。インクリメンタルモードには 2 つの設定項目があります。

job.cfg 内の **isIncremental** がインクリメンタルマイグレーションモードを有効にしているかどうかを確認します。値が **true** の場合、増分モードが有効になっていることを示します。値が **false** の場合は、インクリメンタルモードが無効になっていることを示します。デフォルト設定は **false** です。

job.cfg の **incrementalModeInterval** 設定はインクリメンタルモードの同期間隔を秒単位で指定します。この設定は、**isIncremental** が **true** に設定されている場合に使用されます。構成可能な最小値は **900秒** です。 **3,600秒** より小さい値に設定することはお勧めしません。これは、多数の要求を無駄にし、システムオーバーヘッドを増加させるためです。

アプリケーションのシナリオ：

- データバックアップ
- データ同期

オブジェクト移行にフィルタ条件の指定

指定されたフィルタリング条件を満たすオブジェクトのみが移行されます。OssImport では、接頭辞と最終変更時刻の指定がサポートされています。

- **job.cfg** の **srcPrefix** 設定は、移行されるオブジェクトのプレフィックスを指定します。デフォルトでは空です。**srcType** が **local** の場合は、フルパスでローカルディレクトリに入力し、入力値を / で

終わらせる必要があります (例: `c:/example/` または `/data/example/`) `srcType` が `oss`、`qiniu`、`bos`、`ks3`、`youpai`、または `s3` の場合、`data/to/oss/` のようなバケット名を除いて、同期するオブジェクトの接頭辞を入力する必要があります。

注意：すべてのオブジェクトの `srcPrefix` は空に設定する必要があります。

- `job.cfg` の `importSince` オプションは、移行オブジェクトの最終更新時刻を指定します。これは整数で秒単位で表されます。オブジェクトの Last Modified Time が `importSince` 以前かそれ以前であれば、それは移行されます。オブジェクトの最終変更時刻が `importSince` の後にある場合、そのオブジェクトは移行されません。`importSince` の設定は **Unixのタイムスタンプ** 形式、つまり 1970 年 1 月 1 日 00:00 UTC からの秒数です。`date +%s` コマンドで値を取得できます。デフォルト値は 0 で、すべてのデータを移行することを示します。データ移行のインクリメンタルモードは、最初の完全マイグレーションにのみ有効です。非インクリメンタルモードは、移行ジョブ全体に対して有効です。

ローカルデータを OSS に移行

ローカルデータを OSS に移行するツール：

30 TB 未満のローカルデータファイルを移行する場合、またはストレージサービスをローカルファイルシステムにマウントする場合は、`OssUtil` を使用することをお勧めします。このツールは使いやすく簡単です。`OssUtil` は **オブジェクトレベル** でインクリメンタルアップロードをサポートし、`-u/--update` と `--snapshot-path` オプションを通して機能を実装します。詳細な説明は、`ossutil help cp` コマンドを実行して詳細を参照してください。

大規模データの移行には、`OssImport` の分散バージョンを推奨します。

注意：ローカルデータのインクリメンタルマイグレーション中に、ファイルシステムの一部の操作は、Windows の `cp` や `mv` などのオブジェクトの最終変更時刻を変更しません。これらの操作からのデータ変更は検出されず、OSS に同期されません。

OSS 間のデータ移行

`OssImport` の利用タイミング：

異なるリージョンでの OSS の間でのデータ移行のための **Cross-Region Replication** 機能を追加するには、コンソールでできます。

Cross-Region Replication がリージョンをサポートしていなかったり、ポリシーやセキュリティ上で制限がある場合、データをバックアップするには、`OssImport` を使用することができます。

同じリージョン内の別のアカウントとバケットの間でのデータ移行が可能です。

OSS 内で直接データを移行する場合、つまり **内部** で ECS または OSS ドメイン名を使用する場合、Alibaba Cloud イン트라ネットを推奨します。

OSS 内の直接データ移行の料金：

内部 のドメイン名を使用する場合は、トラフィック費用は発生しませんが、リクエストと保存料を支払う必要があります。

内部 でドメイン名を使用しなかった場合、トラフィック費用が発生する可能性があります。

推奨されない使用例：

Cross-Region Replication 機能が有効になっているリージョン間のデータ移行。

インクリメンタルモードでOSS間でオブジェクトの変更を同期すると、OssImport はオブジェクトの変更 (put/append/multipart) の同期のみをサポートし、読み取りと削除の同期操作はサポートしません。データの同期は、特定の **SLA** によってタイムリーに行われることは保証されません。このオプションを選択するときは注意してください。コールバックのアップロードを使用することをお勧めします。

ECS とトラフィック

クラウド (非ローカル) から OSS にデータを移行し、帯域幅リソースが不足している場合は、移行用の従量課金 ECS インスタンスを購入することをお勧めします。ECS 設定：

- 請求方法として従量課金を選択します。
- OSS に対応するリージョン選択します。
- 帯域幅のピークとして 100 MB を選択します。

移行ジョブの設定では、targetDomainを **内部** を含むイントラネットドメイン名に設定します。ソースの終わりが OSS でもある場合は、srcDomainを **内部** を含むイントラネットドメイン名に設定してください。これにより、OSS ソースドメイン名からダウンロードする費用が節約され、OSS アクセスの料金が課金されません。

HTTP データを OSS に移行

HTTP データ移行ジョブ用に構成するパラメータ：

`job.cfg` で、srcTypeをsrcType = httpに設定します。大文字と小文字が区別されます。

`job.cfg` の `httpListFilePath` では、**絶対パス** を使用して `c:/example/http.list` や `/root/example/http.list` などの HTTP アドレスリストファイルを指定します。完全な HTTP リンクは、**127.0.0.1/aa/bb.jpg** です。異なる分割メソッドは、アップロード後に OSS 上で異なるオブジェクトパスにつながる可能性があります。

```
http://127.0.0.1/aa/ bb.jpg    # The first line
http://127.0.0.1/ aa/bb.jpg # The second line
```

最初の行が OSS にインポートされた後のオブジェクト名は **destPrefix + bb.jpg** であり、2 行目のオブジェクト名は **destPrefix + aa/bb.jpg** です。 `httpPrefixColumn` パラメーターは、ドメイン名の列を指定します。前述の **127.0.0.1/aa/** または **127.0.0.1/** など、最初の列がデフォルトで適用されます。

- `relativePathColumn` は、前述の **bb.jpg** や `* aa/bb.jpg` など、OSS 内のオブジェクト名を指定します。オブジェクトに複数の列がある場合は、次のようになります。

```
http://127.0.0.1/aa/ bb/cc dd/ee ff.jpg
```

設定は以下のようにする必要があります： `httpPrefixColumn = 1`、`relativePathColumn = 4`

- `destAccessKey`、`destSecretKey`、`destDomain`、`destBucket` の設定項目は、`job.cfg` のなかでも特に重要です。

HTTP データ移行タスクのパラメータの分割：

`taskObjectCountLimit`：各タスクのオブジェクトの最大数。デフォルト値は 10,000 です。

`taskObjectSizeLimit`：各タスクの最大データサイズ。このパラメータは、**HTTP データ移行** では無効です。マスターが分割作業を行っているときに、すべての HTTP オブジェクトがソースから取得されたオブジェクトのサイズである場合、各オブジェクトは 1 つの HTTP 要求オーバーヘッドを持ちます。タスクの同時実行と移行の効率性を損なう可能性があります。

Domain name：`httpListFilePath` で指定されたオブジェクトの最初の列。同じドメイン名を持つ連続ジョブは `taskObjectCountLimit` パラメータに従って分割され、異なるドメイン名を持つ連続ジョブは異なるタスクに分割され、接続の再利用が改善されます。例えば：

```
http://mingdi-hz.oss-cn-hangzhou.aliyuncs.com/ import/test1.txt
http://mingdi-hz.oss-cn-hangzhou.aliyuncs.com/ import/test2.txt
http://mingdi-bj.oss-cn-beijing.aliyuncs.com/ import/test3.txt
http://mingdi-bj.oss-cn-beijing.aliyuncs.com/ import/test4.txt
```

`taskObjectCountLimit` の値が 2 より大きい場合、ジョブは 2 つのタスクに分割されます。

```
http://mingdi-hz.oss-cn-hangzhou.aliyuncs.com/ import/test1.txt
```

```
http://mingdi-bj.oss-cn-beijing.aliyuncs.com/ import/test3.txt
http://mingdi-hz.oss-cn-hangzhou.aliyuncs.com/ import/test2.txt
http://mingdi-bj.oss-cn-beijing.aliyuncs.com/ import/test4.txt
```

上記は 4 つのタスクに分割されます。これが `httpListFilePath` で指定された HTTP アドレスリストオブジェクトが最初にドメイン名でソートされる理由です。

ネットワークトラフィックとパラメータ設定

次のパラメータの設定は、ネットワークトラフィックに関連しています。

- `sys.properties` の `'workerTaskThreadNum'` パラメータは、`worker` が同時に実行するジョブの数を示します。ネットワークの品質が悪い場合、または並行性が高い場合は、タイムアウトエラーが多数発生する可能性があります。この時点で、並行性を減らし、設定項目を変更してサービスを再起動することをお勧めします。
- `sys.properties` の `workerMaxThroughput ( KB/s )` パラメータは、`worker` のトラフィック上限を示します。送信元側のトラフィック制御やネットワーク制限などのトラフィックを制限する場合、このパラメータの値は、マシンに許可されている最大のネットワークトラフィックより小さく、ビジネス要件に基づいて評価する必要があります。
- `job.cfg` の `taskObjectCountLimit` パラメータは、各 `task` のオブジェクトの最大数を示します。デフォルト値は 10,000 です。このパラメータは、タスクの数に影響します。タスクの数が少なすぎると、並行タスクの効率が低下する可能性があります。
- `job.cfg` の `taskObjectSizeLimit` は、各 `task` の最大データサイズを示します。デフォルト値は 1 GB です。このパラメータは、タスクの数に影響します。タスクの数が少なすぎると、並行タスクの効率が低下する可能性があります。

注意：

- 移行を開始する前に、構成ファイルのパラメータを決定することをお勧めします。
- `sys.propertie` のパラメータの変更は、移行サーバーの再起動後に有効になります。
- `job.cfg` ジョブがサブミットされると、ジョブの構成パラメーターを変更することはできません。