

目次

はじめに	1.1
目次	1.2
1. Python(boto3)で、オブジェクトストレージにアクセスしてみる	
2. Python(boto3)で、オブジェクトストレージのバケットを表 示してみる	1.3 1.4
3. Python(boto3)で、オブジェクトストレージのバケットを新規作 成してみる	1.5
4. Python(boto3)で、オブジェクトストレージのファイル転送して みる	1.6
5. Python(boto3)で、バケットに格納されているオブジェクトをリ スト表示してみる	1.7
6. Python(boto3)で、バケットに格納されているオブジェクトを削 除してみる	1.8
7. Python(boto3)で、バケットを削除してみる	1.9
8. Python(boto3)で、バケットのバージョニング設定をしてみる	
9. Python(boto3)で、オブジェクトのメタデータを表示してみ る	1.10 1.11
10. Python(boto3)で、オブジェクト公開のための署名付きURLを生 成してみる	1.12
11. Cloudianオブジェクトストレージ/アップロードの進捗状況を表 示してみる	1.13
12. Cloudianオブジェクトストレージ/アップロードの進捗状況を表 示してみる	1.14
おわりに	1.15

オブジェクトストレージ/CLLOUDIAN HyperStoreは Amazon S3 完全互換のオブジェクトストレージになりますので、AWS提供の各種SDKやコマンドラインツールを利用することができます。

<https://aws.amazon.com/jp/tools/>

本書では、AWS SDK for Python (boto3)を利用して CLOUDIAN HyperStore を操作するための環境設定方法と、簡単な使用例（バケットの作成/削除、ファイルのアップロード/ダウンロード、バージョンニング設定等）を説明してます。

※AWS SDK for Python の詳細につきましては、AWSのドキュメントページをご参照ください。

<https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>



目次

1. はじめに
2. AWS SDK for Python セットアップ
 - Amazon SDK for Python (boto3) のインストール
 - クレデンシャル情報の設定
3. AWS SDK For Python 利用方法
 - i. 作成済みバケットの表示 / LIST_BUCKETS()
 - 加工せずに表示
 - バケットに関する情報を絞り込んで表示(各バケット名と作成日時)
 - バケットに関する情報に絞り込んで表示し、作成日時を整形(各バケット名と作成日時)
 - バケットに関する情報を見やすく表示(各バケット名と作成日時)
 - ii. バケットの新規作成 / CREATE_BUCKET()
 - i. シンプルなバケット作成例(属性は全てデフォルト)
 - ii. 属性を指定したバケット作成例1
 - iii. 属性を指定したバケット作成例2

S3 TRANSFERS によるファイル転送処理 (ファイルのアップロード/ダウンロード) S3Transfer オブジェクトの作成 ファイルのアップロード / S3Transfer.upload_file() ファイルのダウンロード / S3Transfer.download_file() バケットに格納されているオブジェクトのリスト表示 / LIST_OBJECTS() バケットに格納されているオブジェクトの削除 / DELETE_OBJECT() バケットの削除 / DELETE_BUCKET() バケットに対するバージョン設定 バケットのバージョンを有効化 / put_bucket_versioning() バケットのバージョン状態の確認 / get_bucket_versioning() バージョニングが有効にされたバケットにファイルをアップロード バージョニングされたオブジェクトのリスト表示 / list_object_versions() バケットのバージョンを一時停止 / put_bucket_versioning() オブジェクトのメタデータ表示 / HEAD_OBJECT() 最新バージョンのオブジェクトのメタデータ 過去バージョンのオブジェクトのメタデータ 事前署名付き URL の生成 / GENERATE_PREIGNED_URL () デフォルトの有効期間(3,600 秒)で事前署名付き URL を生成 有効期限を 2 日間(7,200 秒)に設定して事前署名付き URL を生成 有効期限を 2 分(120 秒)に設定して事前署名付き URL を生成

1. Pythonサンプルプログラム

はじめに

- アップロードの進捗状況表示
- マルチスレッドによるファイルのアップロード

2. おわりに

Python(boto3)で、オブジェクトストレージにアクセスしてみる

Cloudianは、AWSのS3互換のAPIを持ったオブジェクトストレージです。

S3互換APIなので、AWS SDK を使用して操作可能なのですが、S3に接続する前提の記事が多いので、S3以外のオブジェクトストレージへ接続する方法を書こうと思います。

今回は、AWS SDK for Python(boto3)を使って、オブジェクトストレージのバケット一覧（バケットリスト）を取得できるところまで確認していきます。

1. AWS SDK for Python(boto3)のインストール

S3 のファイルを操作するためには、Pythonで AWS SDK for Python（以下boto3）をインポートする必要があります。boto3はpipコマンドで簡単にインストールできます

```
$ pip install boto3
```

2. クレデンシャル情報の設定

Python から boto3 を利用するために、クレデンシャル情報（アクセスキーとシークレットキー）の設定を行います。AWS CLIをインストールすれば、簡単設定できます

AWS CLIのインストール

pipコマンドでインストールします

```
$ pip install awscli
```

Configure設定

aws configureコマンドを実行します

- Cloudianのクレデンシャル情報（アクセスキーとシークレットキー）を入力

- Default region nameは、空エンター入力
- Default output formatは、 jsonと入力

```
$ aws configure

AWS Access Key ID: xxxxxxxxxxxxxxxxx
AWS Secret Access Key: yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
Default region name:
Default output format: json
```

これでクレデンシャル情報の設定完了です。設定した内容は、「~/aws/credentials」と「~/aws/config」で確認できます。

```
$ cat ~/.aws/credentials

[default]
aws_access_key_id = xxxxxxxxxxxxxxxxx
aws_secret_access_key = yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy

$ cat ~/.aws/config

[default]
output=json
```

boto3のimport

S3 のファイルを操作するためには、Pythonでboto3をimportする必要があります。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)
```

boto3.clientの引数として、以下のものを設定します。

- 第一引数には、AWSのサービス種別を設定します。ここには、's3'を設定します。※ boto3 自体は、AWS の S3 以外のサービスのためのメソッドも提供しています。

- 第二引数には、AWSのエンドポイントではなく、オブジェクトストレージのS3エンドポイントを指定するために、「endpoint_url='オブジェクトストレージのS3エンドポイント」を設定します。

```
client = boto3.client('s3', endpoint_url='https://xxx.yyy.com')
```

AWS のサービス種別として 's3' を、S3 エンドポイントの URL として endpoint_url='http://xxx.yyy.com' を設定しています。

endpoint_url の設定により、デフォルトの参照先である AWS の S3 エンドポイントを、オブジェクトストレージの S3 エンドポイントに上書きしています。

参考情報) Pythonプログラム内でクレデンシャル情報を設定したい場合 (AWS CLI/Configure設定で行わない場合) boto3.client/パラメータに、クレデンシャル情報(アクセスキーとシークレットキー)を渡して接続する方法もあります。

```
client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com',

    # Hard coded strings as credentials, not recommended
    aws_access_key_id='4dfba0a142971dbde38b',
    aws_secret_access_key='FdgZNJ7udLiVpbc9HE5M8sgAz5r'
)
```

注) AWS CLI/Configure設定なしですぐ試せますが、認証情報をプログラムに記述することは推奨されません (検証目的用途になります)

参考情報) S3オブジェクトの作成について boto3 を含む AWS SDK では S3 API に 1:1 で対応する「低レベルAPI」と、より高度な操作を行うことができるオブジェクト指向の「高レベルAPI」が提供されています。

- Client API (低レベルAPI) : S3Client(boto3.client) AWSのREST APIと1対1で対応した作りになっている。

- Resource API (高レベルAPI) : S3Resource(boto3.resource) AWSリソースをオブジェクト指向で取り扱えるようになっている。

今回のサンプルでは、低レベルAPIであるS3Client(boto3.client) オブジェクトを使用しています。

3. 動作確認

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケット一覧を取得
client.list_buckets()
```

上記コードを実行して、オブジェクトストレージ/Cloudianに設定したバケットの一覧を取得できます。

```
$ python test1.py

{'ResponseMetadata': {'RequestId': '9dad38b5-0e30-1dbc-a754...',
  'HostId': '',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Sun, 13 Dec 2020 21:57:43 GMT',
    'x-amz-request-id': '9dad38b5-0e30-1dbc-a754-06bdfcde1d5...',
    'content-type': 'application/xml; charset=UTF-8',
    'content-length': '519',
    'server': 'CloudianS3'},
  'RetryAttempts': 0},
 'Buckets': [{'Name': 'bucket1',
  'CreationDate': datetime.datetime(2020, 12, 1, 3, 2, 25),
  'Name': 'pythonbucket2',
  'CreationDate': datetime.datetime(2020, 12, 13, 19, 51,
    12),
  'Name': 'pythonbucket3',
  'CreationDate': datetime.datetime(2020, 12, 13, 21, 41,
    12),
  'Owner': {'DisplayName': '', 'ID': '27b8e84694ca0b529d5379...
```

4. まとめ

Pythonプログラムで作成した boto3のS3Clientオブジェクトには、オブジェクトストレージ/Cloudianに対して様々な操作(バケットの作成/削除やデータのアップロード/ダウンロード等々)を行うことができる多くのメソッドが用意されています。

S3Client(boto3.client)で使用可能なメソッドの詳細情報

<https://boto3.amazonaws.com/v1/documentation/api/latest/index.html>

今回は、このS3Clientオブジェクトを使用して、オブジェクトストレージであるCloudianをいろいろ操作していきたいと思います。

Python(boto3)で、オブジェクトストレージのバケットを表示してみる

Cloudianは S3完全互換なので、AWS SDKを使えて便利です。

前回¹は、AWS SDKのクレデンシャル設定をして、Python(boto3)で、とりあえずオブジェクトストレージにアクセスしてみました

今回は、Python(boto3)で、作成済みバケットをいろんなパターンで表示してみたいと思います。

作成済みバケットの表示 / list_buckets()

アクセスキーとシークレットキーで認証されたユーザーが所有する、全てのバケットの情報を返します。 戻り値は、Python の辞書(dict)型として返されます。

1. 加工せずに表示

以下の例では、list_buckets()を単純に呼び出して、全ての戻り値を表示しています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケット一覧を取得
response = client.list_buckets()
print(response)
```

```
{'ResponseMetadata': {'RequestId': '9dad38b7-0e30-1dbc-a754',
  'HostId': '',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Sun, 13 Dec 2020 21:58:07 GMT',
    'x-amz-request-id': '9dad38b7-0e30-1dbc-a754-06bdfcde1d5',
    'content-type': 'application/xml; charset=UTF-8',
    'content-length': '519',
    'server': 'CloudianS3'},
  'RetryAttempts': 0},
'Buckets': [{ 'Name': 'bucket1',
  'CreationDate': datetime.datetime(2020, 12, 1, 3, 2, 25),
  { 'Name': 'pythonbucket2',
  'CreationDate': datetime.datetime(2020, 12, 13, 19, 51,
  { 'Name': 'pythonbucket3',
  'CreationDate': datetime.datetime(2020, 12, 13, 21, 41,
  'Owner': { 'DisplayName': '', 'ID': '27b8e84694ca0b529d5379
```

`list_buckets()`からは非常に多くの情報が返されるため、以降の例では出力を絞り込んで表示させていきます。

2. バケットに関する情報を絞り込んで表示 (各バケット名と作成日時)

`list_buckets()`で返された辞書型の戻り値から、辞書キー「Buckets」の値を抽出し、さらに「Name」(バケット名)と「CreationDate」(作成日時)に絞り込んで表示させています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケット一覧を取得
for bucket in client.list_buckets()['Buckets']:
    print(bucket['Name'], bucket['CreationDate'])
```

```
bucket1 2020-12-01 03:02:25.876000+00:00
pythonbucket2 2020-12-13 19:51:20.267000+00:00
pythonbucket3 2020-12-13 21:41:08.495000+00:00
```

`list_buckets()`は辞書(dict)型で作成済みバケット に関する情報を返しますので、プログラムで必要となるデータのみを Python プログラムで抽出することができます。

string 型のバケット名と、datetime 型の作成日付のみを抽出します

3. バケットに関する情報に絞り込んで表示し、作成日時を整形（各バケット名と作成日時）

バケット名とその作成日付に絞り込み、表示される作成日付を整形しています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケット一覧を取得
for bucket in client.list_buckets()['Buckets']:
    print(bucket['Name'], bucket['CreationDate'].strftime("%s
```

```
bucket1 2020/12/01 03:02:25
pythonbucket2 2020/12/13 19:51:20
pythonbucket3 2020/12/13 21:41:08
```

datetime 型で戻される作成日付(CreationDate)を、Python 標準モジュールに含まれる `strftime()`を使って見やすいように整形して出力しています。

4. バケットに関する情報を見やすく表示（各バケット名と作成日時）

出力されるバケット作成日付に+9時間して、日本時間で表示しています。

```
import boto3

# 追加モジュール
import datetime

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケット一覧を取得
for bucket in client.list_buckets()['Buckets']:
    print('%s (%s)' % (bucket['Name'], (bucket['CreationDate'])))
```

```
bucket1 (2020/12/01 12:02:25)
pythonbucket2 (2020/12/14 04:51:20)
pythonbucket3 (2020/12/14 06:41:08)
```

5. まとめ

Python(boto3)で、作成済みバケットをいろんなパターンで表示してみました。

Python(boto3)で、オブジェクトストレージのバケットを新規作成してみる

Cloudianは S3完全互換なので、AWS SDKを使えて便利ですよ！ [前回](#)は、Python(boto3)で、作成済みバケットをいろんなパターンで表示してみました。

今回は、Python(boto3)で、オブジェクトストレージのバケットを新規作成してみたいと思います。

バケットの新規作成 / create_bucket()

オブジェクトストレージ/Cloudianにバケットを新規作成します。

1. シンプルなバケット作成例(属性は全てデフォルト)

以下の例では、create_bucket() の引数に作成するバケット名(Bucket='pythonbucket1')のみを設定し、属性値は全てデフォルトを使用してバケットを作成しています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケットの新規作成
client.create_bucket(Bucket='pythonbucket1')
```

```
{'ResponseMetadata': {'RequestId': '9dad4484-0e30-1dbc-a754-06bdfcde1d5f',
  'HostId': '3d5c05376530a2eb49e3e90576f83c5b',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Mon, 14 Dec 2020 01:51:29 GMT',
    'x-amz-request-id': '9dad4484-0e30-1dbc-a754-06bdfcde1d5f',
    'location': 'http://pythonbucket1.s3-region1.admin-tech.tokyo.amazonaws.com',
    'x-amz-id-2': '3d5c05376530a2eb49e3e90576f83c5b',
    'content-length': '0',
    'server': 'CloudianS3'},
  'RetryAttempts': 0},
  'Location': 'http://pythonbucket1.s3-region1.admin-tech.tokyo.amazonaws.com'}
```

`list_buckets()`からは非常に多くの情報が返されるため、以降の例では出力を絞り込んで表示させていきます。

2. 属性を指定したバケット作成例①

以下の例では、`create_bucket()` の引数に作成するバケット名 (`Bucket='pythonbucket1'`)とそのバケットの ACL (`ACL='public-read'`)を設定して、バケットを作成しています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケットの新規作成
client.create_bucket(
    ACL='public-read',
    Bucket='pythonbucket1'
)
```

```
{'ResponseMetadata': {'RequestId': '9dad4487-0e30-1dbc-a754-06bdfcde1d5f',
  'HostId': '3d5c05376530a2eb49e3e90576f83c5b',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Mon, 14 Dec 2020 01:51:30 GMT',
    'x-amz-request-id': '9dad4487-0e30-1dbc-a754-06bdfcde1d5f',
    'location': 'http://pythonbucket1.s3-region1.admin-tech.tokyo.amazonaws.com',
    'x-amz-id-2': '3d5c05376530a2eb49e3e90576f83c5b',
    'content-length': '0',
    'server': 'CloudianS3'},
  'RetryAttempts': 0},
  'Location': 'http://pythonbucket1.s3-region1.admin-tech.tokyo.amazonaws.com'}
```

3. 属性を指定したバケット作成例②

以下の例では、`create_bucket()` の引数に作成するバケット名(`Bucket='bucket2'`)とバケットのACL(`ACL='private'`)、およびロケーションコンストレイント (`CreateBucketConfiguration={'LocationConstraint': 'region1'}`)を設定してバケットを作成しています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケットの新規作成
client.create_bucket(
    ACL='private',
    Bucket='pythonbucket1',
    CreateBucketConfiguration={
        'LocationConstraint': 'region1'
    }
)
```



```
{'ResponseMetadata': {'RequestId': '9dad448a-0e30-1dbc-a754-06bdfcde1d5f',  
  'HostId': '3d5c05376530a2eb49e3e90576f83c5b',  
  'HTTPStatusCode': 200,  
  'HTTPHeaders': {'date': 'Mon, 14 Dec 2020 01:51:31 GMT',  
    'x-amz-request-id': '9dad448a-0e30-1dbc-a754-06bdfcde1d5f',  
    'location': 'http://pythonbucket1.s3-region1.admin-tech.amazonaws.com/pythonbucket1.s3-region1.admin-tech.technical',  
    'x-amz-id-2': '3d5c05376530a2eb49e3e90576f83c5b',  
    'content-length': '0',  
    'server': 'CloudianS3'},  
  'RetryAttempts': 0},  
  'Location': 'http://pythonbucket1.s3-region1.admin-tech.technical'}
```

datetime 型で戻される作成日付(CreationDate)を、Python 標準モジュールに含まれる strftime()を使って見やすいように整形して出力しています。

4. まとめ

Python(boto3)で、いろんなパターンでバケットを新規作成してみました。

Python(boto3)で、オブジェクトストレージのファイル転送してみる

Cloudianは S3完全互換なので、AWS SDKを使えて便利です！ [前回](#)は、Python(boto3)で、バケットを新規作成してみました。

今回は、Python(boto3)で、オブジェクトストレージのファイル転送を試みたいと思います。

バケットの新規作成 / create_bucket()

オブジェクトストレージ/Cloudianにバケットを新規作成します。

S3 Transfers によるファイル転送処理 (ファイルのアップロード/ダウンロード)

boto3 には、より簡易に/より効率的にアップロード/ダウンロード操作を行うことができる S3 Transfers というモジュールが用意されています。

S3 Transfers は、以下のような機能を提供しています。

- 指定したファイルサイズを上回った際に、自動的にマルチパート転送に切り替わります。
- 並列処理でファイルのアップロード/ダウンロードを実行します。
- ネットワークの最大帯域幅に応じて、ファイルを転送します。
- 転送状況をモニタリングするために、転送の進捗状況をコールバックできます。
- ファイルのアップロード時に、リトライを実行します。

S3 Transfers のパラメータのデフォルト値には最適な値が設定されていますが、以下のパラメータ設定を変更することもできます。

- マルチパート閾値サイズ
- 並行ダウンロード処理の最大数
- 使用するネットワーク最大帯域幅
- ソケット・タイムアウト値
- リトライ回数

S3 Transfers のパラメータを変更するには、変更対象のパラメータの設定値を引数に指定して、TransferConfig オブジェクトを作成します(事前に "boto3.s3.transfer.TransferConfig" をインポートする必要があります)

```
from boto3.s3.transfer import S3Transfer
from boto3.s3.transfer import TransferConfig

config = TransferConfig(
    multipart_threshold = 8 * 1024 * 1024,
    max_concurrency = 10,
    multipart_chunksize = 8388608,
    num_download_attempts = 10,
    max_io_queue = 100
)

transfer = S3Transfer(client, config)
```

1. S3Transfer オブジェクトの作成

S3 Transfers の機能を利用するためには、まず S3Transfer オブジェクトを作成します。

事前に、`boto3.s3.transfer.S3Transfer` をインポートしておく必要があります。本書の例では、既に S3Transfer はインポート済みの状態になっています。

`S3Transfer()` の引数に、既に作成済みの `S3Client` オブジェクトを渡して S3Transfer オブジェクトを作成します。S3Client オブジェクトのみを引数として渡し S3 Transfers オブジェクトを作成した場合には、S3 Transfers のパラメータは全てデフォルト値が使用されます。

```
transfer = S3Transfer(client)
```

2. ファイルのアップロード / `S3Transfer.upload_file()`

S3 Transfers を使用して Cloudian にファイルをアップロードするには、`upload_file()` を使用します。

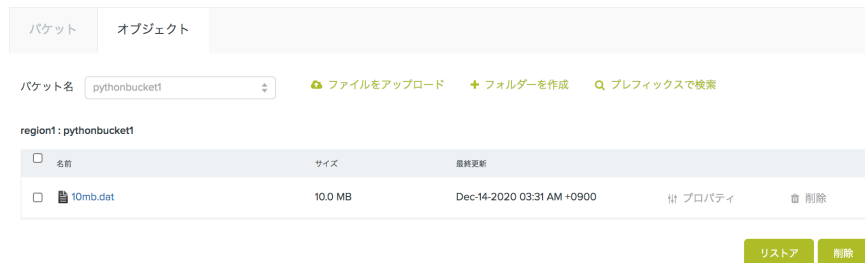
`extra_args` 引数を設定して `upload_file()` を呼び出すことにより、アップロード時にオブジェクト(ファイル)の ACL を設定したり、メタデータを付加したり、暗号化を行ったりできます。

シンプルなファイルのアップロード

以下の例では、ローカルにあるファイル「10mb.dat」を、Cloudian のバケット「pythonbucket1」にキー「10mb.dat」を設定してアップロードしています。

```
transfer.upload_file('fileup/10mb.dat', 'pythonbucket1', '10mb.dat')
```

下図は、upload_file()の使用例で Cloudian にアップロードしたファイルを Cloudant Management Console（以下CMC）のオブジェクト画面から参照したものです。

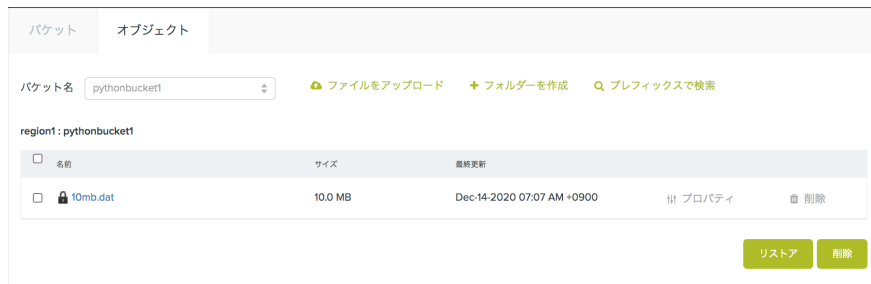


アップロード時に ACL、メタデータ、暗号化を指定

以下の例では、ローカルにあるファイル「10mb.dat」を、Cloudian のバケット「pythonbucket1」にキー「10mb.dat」を指定し、extra_args を設定してオブジェクトの ACL には「public-read」、3 つのメタデータを付加して「AES256」でサーバーサイド暗号化するように指定してアップロードしています。

```
transfer.upload_file(
    'fileup/10mb.dat', 'pythonbucket1', '10mb.dat',
    extra_args={
        'ACL': 'public-read',
        'Metadata': {
            'Purpose': 'boto3 demo',
            'Engineer': 'yamahiro',
            'Company': 'Networld'
        },
        'ServerSideEncryption': 'AES256'
    }
)
```

upload_file()の extra_args パラメータ「ServerSideEncryption」を使用し、AES256 で サーバーサイド暗号化を行うように設定したので、ファイル名の先頭に「🔒」マークが表示され、このファイルが暗号化されていることが分かります。



また、下図のようにこのファイルのプロパティを開いてみると、`extra_args` パラメータの ACL で設定したように、このファイルのアクセス権に「パブリック:読み出し可能('ACL': 'public-read')」にチェックが入っていることが分かります。



3. ファイルのダウンロード / `S3Transfer.download_file()`

S3 Transfers を使用して Cloudian からファイルをダウンロードするには、`download_file()`を使用します。`extra_args` 引数を設定して `download_file()`を呼び出すことにより、ダウンロード時にオブジェクトのバージョン ID を指定したりすることができます。

以下の例では、バケット「pythonbucket1」に保存されているキー「10mb.dat」の最新バージョンのオブジェクトをダウンロードしています。

バケット「pythonbucket1」のバージョニング機能が有効にされていた場合、この例ではバージョン ID を指定していないので“最新バージョンのオブジェクト”がダウンロードされます(バージョニング機能については、後述します)。

```
transfer.download_file('pythonbucket1', '10mb.dat', 'fileloc')
```

バージョニング機能が有効な場合

以下の例では、バージョニング機能が有効化されているバケット「pythonbucket1ver」に保存されているキー「10mb.dat」の、バージョン ID が「fe14c26e-1662-4f8f-a754-06bdfcde1d5e」のオブジェクトをダウンロードしています。

```
transfer.download_file(
    'pythonbucket1ver', '10mb.dat', 'filelocal/10mb-local.dat',
    extra_args={'VersionId': 'fe14c26e-1662-4f8f-a754-06bdfcde1d5e'}
)
```

S3 Transfers を使用したファイルのアップロード/ダウンロード時には、マルチパート閾値サイズ(multipart_threshold)が設定されており、そのデフォルト値は「8,388,608 bytes(約 8MB)」になっています。

アップロード/ダウンロード対象のファイルサイズが 8MB よりも大きい場合、S3 Transfers は自動的に複数のパーツにファイルを分割して同時並行で処理を実行します。

下図は大きなサイズのファイルを S3 Transfers でアップロードを行っているときの、CMC オブジェクト画面のスクリーンショットです。8MB よりも大きなファイルは、自動的にマルチパートアップロードが実行されていることが分かります。

バケット

オブジェクト

バケット名

📁 ファイルをアップロード

✚ フォルダを作成

🔍 プレフィックスで検索

region1: boto3

<input type="checkbox"/>	名前	サイズ	最終更新		プロパティ	削除
<input type="checkbox"/>	🔒 200mb_mpu.dat	200.0 MB	Dec-14-2020 08:27 AM +0900			

リストア

削除

マルチパートアップロード実行中

マルチパートアップロード開始日	アップロードオブジェクト名	アクション
Dec-14-2020 08:05 AM +0900	200mb_mpu.dat	🛑 中止

※補足: get_object() を使用したファイルのダウンロード

S3Transfers オブジェクトの `download_file()` を使用したファイルのダウンロード以外に、低レベル API に対応する `get_object()` を使用してファイルのダウンロードを行うこともできます。

```
with open('filelocal/10mb-local.dat', 'wb') as f:
    f.write(client.get_object(
        Bucket='pythonbucket1ver',
        Key='10mb.dat',
        VersionId='fe14c26e-1662-4f8f-a754-06bdfcc
    )['Body']).read()
)
```

ただしこの場合、上記サンプルコードのように、S3Transfers オブジェクトを使用した方法よりも若干、コードが複雑になります。同じ操作を実行するために複数の方法があることがありますので、適材適所でどの方法を使用するか決める必要があります。

4. まとめ

Python(boto3)で、ファイル転送（アップロード/ダウンロード）をしました。

Python(boto3)で、バケットに格納されているオブジェクトをリスト表示してみる

Cloudianは S3完全互換なので、AWS SDKを使えて便利ですよ！ [前回](#)は、Python(boto3)で、ファイル転送を行ってみました。

今回は、Python(boto3)で、オブジェクトストレージのバケットに格納されているオブジェクトをリスト表示してみたいと思います。

バケットに格納されているオブジェクトのリスト表示 / list_objects()

Cloudianにアップロードされたファイル(オブジェクト)の存在を、list_objects()を使って確認します。list_objects()は多くの情報を Python のdict型で返します。

以下の例では、返された戻り値の中からオブジェクトのキーとサイズ、最終変更日時のみを表示しています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケット名:pythonbucket1のオブジェクトを全てリスト表示
for obj in client.list_objects(Bucket='pythonbucket1')['Contents']:
    print(obj['Key'], obj['Size'],
          obj['LastModified'].strftime("%Y/%m/%d %H:%M:%S"))
```

```
10mb.dat 10485760 2020/12/14 03:12:38
```

まとめ

Python(boto3)で、オブジェクトのリスト表示してみました。

Python(boto3)で、バケットに格納されているオブジェクトを削除してみる

Cloudianは S3完全互換なので、AWS SDKを使えて便利ですね！ [前回](#)は、Python(boto3)で、オブジェクトのリスト表示してみました。

今回は、Python(boto3)で、オブジェクトストレージのバケットに格納されているオブジェクトを削除してみようと思います。

バケットに格納されているオブジェクトの削除 / delete_object()

Cloudianにアップロードされたファイル(オブジェクト)を、delete_object()を使って削します。

以下の例では、バケット「pythonbucket1」に保存されているキー「10mb.dat」の、最新バージョンのオブジェクトを削除しています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケット名:pythonbucket1 のオブジェクト:10mb.datファイル を削除
client.delete_object(Bucket='pythonbucket1', Key='10mb.dat')
```

```
{'ResponseMetadata': {'RequestId': '9dad3b68-0e30-1dbc-a754',
  'HostId': '',
  'HTTPStatusCode': 204,
  'HTTPHeaders': {'date': 'Sun, 13 Dec 2020 22:27:45 GMT',
    'x-amz-request-id': '9dad3b68-0e30-1dbc-a754-06bdfcde1d5',
    'server': 'CloudianS3'},
  'RetryAttempts': 0}}
```

まとめ

Python(boto3)で、オブジェクトを削除してみました。

Python(boto3)で、バケットを削除してみる

Cloudianは S3完全互換なので、AWS SDKを使えて便利ですよね！ [前回](#)は、Python(boto3)で、オブジェクトを削除してみました。

今回は、Python(boto3)で、オブジェクトストレージのバケットを削除してみようと思います。

バケットの削除 / delete_bucket()

Cloudianに作成されているバケットを削除します。

以下の例では、バケット「pythonbucket1」を削除しています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケット名:pythonbucket1 の削除
client.delete_bucket(
    Bucket='pythonbucket1'
)
```

```
{'ResponseMetadata': {'RequestId': '9dad3274-0e30-1dbc-a754-06bdfcde1d5f',
                        'HostId': '',
                        'HTTPStatusCode': 204,
                        'HTTPHeaders': {'date': 'Sun, 13 Dec 2020 20:12:33 GMT',
                                          'x-amz-request-id': '9dad3274-0e30-1dbc-a754-06bdfcde1d5f',
                                          'server': 'CloudianS3'},
                        'RetryAttempts': 0}}
```

※注意 バケットを削除するには、バケット内にオブジェクトが存在しない状態である必要があります。バケット内にオブジェクトが存在する状態で `delete_bucket()` を実行すると、以下のような `BucketNotEmpty` 例外が発生します。

```
client.delete_bucket(  
    Bucket='bucket1'  
)
```

```
ClientError                                Traceback (most recent call last)  
<ipython-input-3-0bb1849245e5> in <module>  
      1 client.delete_bucket(  
----> 2     Bucket='bucket1'  
      3 )  
  
~/pyenv/versions/anaconda3-2019.03/lib/python3.7/site-packages/boto3/compat.py:314: in _call_action  
    314         "%s() only accepts keyword arguments" % func_name, kwargs)  
    315         # The "self" in this scope is reference to the boto3 client.  
--> 316         return self._make_api_call(operation_name, kwargs)  
    317  
    318     _api_call.__name__ = str(py_operation_name)  
  
~/pyenv/versions/anaconda3-2019.03/lib/python3.7/site-packages/boto3/client.py:624: in _make_api_call  
    624         error_code = parsed_response.get("Error", {}).get("Code")  
    625         error_class = self.exceptions.from_code(error_code)  
--> 626         raise error_class(parsed_response, operation_name)  
    627     else:  
    628         return parsed_response  
  
ClientError: An error occurred (BucketNotEmpty) when calling the DeleteBucket operation: The bucket is not empty
```

まとめ

Python(boto3)で、バケットを削除してみました。

Python(boto3)で、バケットのバージョンニング設定を試みる

[Cloudian](#)は S3完全互換なので、AWS SDKを使えて便利です！ [前回](#)は、Python(boto3)で、バケットを削除してみました。

今回は、Python(boto3)で、オブジェクトストレージのバケットのバージョンニング設定を行ってみたいと思います。

バケットのバージョンニング設定

バージョンニング機能を有効にして、その動作を確認します。

バケット「pythonbucket2」は、既に作成されており存在しているものとして扱います。

1. バケットのバージョンニングを有効化 / put_bucket_versioning()

put_bucket_versioning()の引数に、バージョンニングの状態を変更するバケット名と変更する状態(ここでは'Enabled')を渡して実行し、バージョンニング機能を有効にします。

以下の例では、バケット「pythonbucket2」のバージョンニング機能の状態を有効('Status': 'Enabled')に変更しています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケット名:pythonbucket2 のバージョンニング機能の有効化
client.put_bucket_versioning(
    Bucket='pythonbucket2',
    VersioningConfiguration={'Status': 'Enabled'}
)
```

```
{'ResponseMetadata': {'RequestId': '9dad3e00-0e30-1dbc-a754',
  'HostId': '',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Sun, 13 Dec 2020 22:30:33 GMT',
    'x-amz-request-id': '9dad3e00-0e30-1dbc-a754-06bdfcde1d5',
    'content-length': '0',
    'server': 'CloudianS3'},
  'RetryAttempts': 0}}
```

注意 バケットのバージョニング機能は、有効化するとその後、無効化することはできません。

バージョニング機能を一時停止させたい場合には、後述するように VersioningConfiguration の Status に「Suspended」を渡して put_bucket_versioning()を実行します。

2. バケットのバージョニング状態の確認 / get_bucket_versioning()

get_bucket_versioning()の引数に、バージョニングの状態を確認するバケット名を渡して実行します。

以下の例では、バケット「pythonbucket2」のバージョニング状態を取得しています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケット名:pythonbucket2 のバージョニング状態を取得
client.get_bucket_versioning(Bucket='pythonbucket2')
```

```
{'ResponseMetadata': {'RequestId': '9dad3e02-0e30-1dbc-a754-06bdfcde1d5f',
  'HostId': '',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Sun, 13 Dec 2020 22:30:36 GMT',
    'x-amz-request-id': '9dad3e02-0e30-1dbc-a754-06bdfcde1d5f',
    'content-type': 'application/xml; charset=UTF-8',
    'content-length': '161',
    'server': 'CloudianS3'},
  'RetryAttempts': 0},
'Status': 'Enabled'}
```



3. バージョニングが有効にされたバケットにファイルをアップロード

バージョニング機能が有効化されたバケットに対して、前述の S3 Transfers の `upload_file()` を使用して、同じキーで複数回、ファイルをアップロードしてみます。

以下の例では、バージョニング機能が有効化されたバケット「pythonbucket2」に、キーに「10mb.dat」を設定してファイルを計 4 回アップロードしています。

```

import boto3

from boto3.s3.transfer import S3Transfer
from boto3.s3.transfer import TransferConfig

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

config = TransferConfig(
    multipart_threshold = 8 * 1024 * 1024,
    max_concurrency = 10,
    multipart_chunksize = 8388608,
    num_download_attempts = 10,
    max_io_queue = 100
)

# S3Transfer オブジェクトの作成
transfer = S3Transfer(client, config)

# 【1 回目】
transfer.upload_file('fileup/10mb.dat', 'pythonbucket2', '10mb.dat')

# 【2 回目】
transfer.upload_file('fileup/10mb.dat', 'pythonbucket2', '10mb.dat',
    extra_args={
        'ACL': 'public-read',
        'Metadata': {
            'Purpose': 'boto3 demo 1',
            'Engineer': 'yamahiro',
            'Company': 'nw'
        },
        'ServerSideEncryption': 'AES256'
    }
)

# 【3 回目】
transfer.upload_file('fileup/10mb.dat', 'pythonbucket2', '10mb.dat',
    extra_args={
        'ACL': 'public-read',
        'Metadata': {
            'Purpose': 'boto3 demo 2',

```

```
        'Engineer': 'miki',
        'Company': 'nw'
    },
    'ServerSideEncryption': 'AES256'
}

)

# 【4 回目】
transfer.upload_file('fileup/10mb.dat', 'pythonbucket2', '10mb.dat',
    extra_args={'ServerSideEncryption': 'AES256'})
```

※ アップロード時にユーザー定義のメタデータを付加 extra_args={Metadata: {'名前': '値',}}

※ アップロード時に暗号化(AES256)を指定
'ServerSideEncryption': 'AES256'

4. バージョニングされたオブジェクトのリスト表示 / list_object_versions()

バージョニングされた(今現在、最新バージョンではない)オブジェクト(ファイル)をリスト表示するには、list_object_versions()を実行します。

加工せずに表示

以下の例では、list_object_versions() の引数にバケット名 (Bucket='pythonbucket2')のみを設定し、全ての戻り値を表示しています

```
client.list_object_versions(Bucket='pythonbucket2')
```



```
{'ResponseMetadata': {'RequestId': '9dad3e3e-0e30-1dbc-a754-06bdfcde1d5e',
  'HostId': '',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Sun, 13 Dec 2020 22:35:45 GMT',
    'x-amz-request-id': '9dad3e3e-0e30-1dbc-a754-06bdfcde1d5e',
    'x-gmt-policyid': 'b46db4b3ebb2180b046ad1065c9702e1',
    'x-amz-bucket-region': 'region1',
    'content-type': 'application/xml; charset=UTF-8',
    'content-length': '1674',
    'server': 'CloudianS3'},
  'RetryAttempts': 0},
  'IsTruncated': False,
  'KeyMarker': '',
  'VersionIdMarker': '',
  'Versions': [{ 'ETag': '"669fdad9e309b552f1e9cf7b489c1f73-2"',
    'Size': 10485760,
    'StorageClass': 'STANDARD',
    'Key': '10mb.dat',
    'VersionId': 'fe14c26d-16ea-60bf-a754-06bdfcde1d5e',
    'IsLatest': True,
    'LastModified': datetime.datetime(2020, 12, 13, 22, 31, 0),
    'Owner': {'ID': '27b8e84694ca0b529d5379049564ebe1'}},
    { 'ETag': '"669fdad9e309b552f1e9cf7b489c1f73-2"',
    'Size': 10485760,
    'StorageClass': 'STANDARD',
    'Key': '10mb.dat',
    'VersionId': 'fe14c26d-18b5-d1ef-a754-06bdfcde1d5e',
    'IsLatest': False,
    'LastModified': datetime.datetime(2020, 12, 13, 22, 31, 0),
    'Owner': {'ID': '27b8e84694ca0b529d5379049564ebe1'}},
    { 'ETag': '"669fdad9e309b552f1e9cf7b489c1f73-2"',
    'Size': 10485760,
    'StorageClass': 'STANDARD',
    'Key': '10mb.dat',
    'VersionId': 'fe14c26d-1a6a-119f-a754-06bdfcde1d5e',
    'IsLatest': False,
    'LastModified': datetime.datetime(2020, 12, 13, 22, 31, 0),
    'Owner': {'ID': '27b8e84694ca0b529d5379049564ebe1'}},
    { 'ETag': '"669fdad9e309b552f1e9cf7b489c1f73-2"',
    'Size': 10485760,
    'StorageClass': 'STANDARD',
    'Key': '10mb.dat',
    'VersionId': 'fe14c26d-1e9b-382f-a754-06bdfcde1d5e',
    'IsLatest': False,
    'LastModified': datetime.datetime(2020, 12, 13, 22, 31, 0),
    'Owner': {'ID': '27b8e84694ca0b529d5379049564ebe1'}}],
  'Name': 'pythonbucket2',
  'Prefix': ''}
```

```
'MaxKeys': 1000,
'EncodingType': 'url'}
```

list_object_versions()からは非常に多くの情報が返されるため、以降の例では出力を絞り込んで表示させます。

バージョンングされたオブジェクトに関する情報に絞り込んで表示

以下の例では、list_object_versions() の引数にバケット名 (Bucket='pythonbucket2')のみを設定し、出力をキーとバージョンID、最終変更日時に絞って表示しています。

```
for version in client.list_object_versions(Bucket='pythonbu
    print(version['Key'],version['VersionId'], version['La
```

```
10mb.dat fe14c26d-16ea-60bf-a754-06bdfcde1d5e 2020/12/13 22
10mb.dat fe14c26d-18b5-d1ef-a754-06bdfcde1d5e 2020/12/13 22
10mb.dat fe14c26d-1a6a-119f-a754-06bdfcde1d5e 2020/12/13 22
10mb.dat fe14c26d-1e9b-382f-a754-06bdfcde1d5e 2020/12/13 22
```

バケット

オブジェクト

バケット名

pythonbucket2

📁

ファイルをアップロード

+

フォルダーを作成

🔍

プレフィックスで検索

バージョンを非表示

region1 : pythonbucket2

<input type="checkbox"/>	名前	サイズ	最終更新	
	<div><div>📄</div>10mb.dat</div>	--	--	
<input type="checkbox"/>	<div><div>🔒</div>fe14c27e-8097-f12f-a754-06bdfcde1d5e</div>	10.0 MB	Dec-14-2020 05:26 AM +0900	<div><div>📄</div>プロパティ</div> <div><div>🗑</div>削除</div>
<input type="checkbox"/>	<div><div>🔒</div>fe14c27e-836a-590f-a754-06bdfcde1d5e</div>	10.0 MB	Dec-14-2020 05:26 AM +0900	<div><div>📄</div>プロパティ</div> <div><div>🗑</div>削除</div>
<input type="checkbox"/>	<div><div>🔒</div>fe14c27e-85fd-11df-a754-06bdfcde1d5e</div>	10.0 MB	Dec-14-2020 05:26 AM +0900	<div><div>📄</div>プロパティ</div> <div><div>🗑</div>削除</div>
<input type="checkbox"/>	<div><div>📄</div>fe14c27e-8971-c63f-a754-06bdfcde1d5e</div>	10.0 MB	Dec-14-2020 05:26 AM +0900	<div><div>📄</div>プロパティ</div> <div><div>🗑</div>削除</div>

削除

[削除](#)

5. バケットのバージョンングを一時停止/put_bucket_versioning()

put_bucket_versioning()の引数に、バージョンングの状態を変更するバケット名と変更する状態(ここでは'Suspended')を渡して実行し、バージョンング機能を一時停止します。

以下の例では、バケット「pythonbucket1」のバージョニング機能を一時停止 (VersioningConfiguration={'Status': 'Suspended'})させています。

```
client.put_bucket_versioning( Bucket='pythonbucket2', Vers:
```

```
{'ResponseMetadata': {'RequestId': '9dad3e02-0e30-1dbc-a754',
  'HostId': '',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Sun, 13 Dec 2020 22:30:36 GMT',
    'x-amz-request-id': '9dad3e02-0e30-1dbc-a754-06bdfcde1d5',
    'content-type': 'application/xml; charset=UTF-8',
    'content-length': '161',
    'server': 'CloudianS3'},
  'RetryAttempts': 0},
  'Status': 'Enabled'}
```

上記の put_bucket_versioning()実行後、バケット「pythonbucket1」のバージョニング機能の状態を表示させ、バージョニング機能が一時停止していることを確認しています。

```
client.get_bucket_versioning(Bucket='pythonbucket2')
```

```
{'ResponseMetadata': {'RequestId': '9dad3e44-0e30-1dbc-a754',
  'HostId': '',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Sun, 13 Dec 2020 22:36:23 GMT',
    'x-amz-request-id': '9dad3e44-0e30-1dbc-a754-06bdfcde1d5',
    'content-type': 'application/xml; charset=UTF-8',
    'content-length': '163',
    'server': 'CloudianS3'},
  'RetryAttempts': 0},
  'Status': 'Suspended'}
```

pythonbucket2

region1

pol1

プロパティ

削除

個別アクセス権	バケットのCANNED ACL	ストレージポリシー	ライフサイクルポリシー	静的WEBサイトホスティング	クロスリージョンレプリケーション	VERSIONING	LOGGING	
このバケットのバージョニングは停止されています。								

まとめ

Python(boto3)で、バケットのバージョニング設定をしました。

Python(boto3)で、オブジェクトのメタデータを表示してみる

[Cloudian](#)は S3完全互換なので、AWS SDKを使えて便利ですよ！ [前回](#)は、Python(boto3)で、バケットのバージョニング設定をしました。

今回は、Python(boto3)で、オブジェクトストレージのオブジェクトのメタデータ表示を行っていきましょうと思います。

オブジェクトのメタデータ表示 / head_object()

アップロード時にオブジェクトに付与したメタデータを、head_object()で表示します。

1. 最新バージョンのオブジェクトのメタデータ

以下の例では、バケット「pythonbucket2」に保存されているキー「10mb.dat」の最新バージョンのオブジェクトに付与されているメタデータを表示させています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケット名: pythonbucket2fix の オブジェクトのメタデータ表示
ret = client.head_object(Bucket='pythonbucket2fix', Key='10mb.dat')

print(ret)
```

```
{'ResponseMetadata': {'RequestId': '9dad3fc3-0e30-1dbc-a754-06bdfcde1d5e',
  'HostId': '',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Sun, 13 Dec 2020 22:43:10 GMT',
    'x-amz-request-id': '9dad3fc3-0e30-1dbc-a754-06bdfcde1d5e',
    'last-modified': 'Sun, 13 Dec 2020 22:40:57 GMT',
    'etag': '"669fdad9e309b552f1e9cf7b489c1f73-2"',
    'content-type': 'binary/octet-stream',
    'x-amz-server-side-encryption': 'AES256',
    'x-amz-version-id': 'fe14c26b-bba0-6edf-a754-06bdfcde1d5e',
    'accept-ranges': 'bytes',
    'content-length': '10485760',
    'server': 'CloudianS3'},
  'RetryAttempts': 0},
  'AcceptRanges': 'bytes',
  'LastModified': datetime.datetime(2020, 12, 13, 22, 40, 57),
  'ContentLength': 10485760,
  'ETag': '"669fdad9e309b552f1e9cf7b489c1f73-2"',
  'VersionId': 'fe14c26b-bba0-6edf-a754-06bdfcde1d5e',
  'ContentType': 'binary/octet-stream',
  'ServerSideEncryption': 'AES256',
  'Metadata': {}}
```

格納されているバケットのバージョニング機能が有効にされており、かつバージョンID を指定しないで `head_object()` を実行した場合には、そのバケットに保存されている最新バージョンのオブジェクトに付与されているメタデータを返します。

2. 過去バージョンのオブジェクトのメタデータ

以下の例では、バージョニング機能が有効になっているバケット

「pythonbucket2fix」に保存されているキー「10mb.dat」の、バージョンID「fe14c26b-bed0-c73f-a754-06bdfcde1d5e」のオブジェクトに付与されているメタデータを表示させています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# バケット名: pythonbucket2fix の オブジェクトのメタデータ表示
ret = client.head_object(Bucket='pythonbucket2fix', Key='10

print(ret)
```

```
{'ResponseMetadata': {'RequestId': '9dad3fdf-0e30-1dbc-a754',
  'HostId': '',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'date': 'Sun, 13 Dec 2020 22:43:41 GMT',
    'x-amz-request-id': '9dad3fdf-0e30-1dbc-a754-06bdfcde1d5',
    'x-amz-meta-engineer': 'yamahiro',
    'x-amz-meta-company': 'nw',
    'x-amz-meta-purpose': 'boto3 demo 1',
    'last-modified': 'Sun, 13 Dec 2020 22:40:52 GMT',
    'etag': '"669fdad9e309b552f1e9cf7b489c1f73-2"',
    'content-type': 'binary/octet-stream',
    'x-amz-server-side-encryption': 'AES256',
    'x-amz-version-id': 'fe14c26b-bed0-c73f-a754-06bdfcde1d5',
    'accept-ranges': 'bytes',
    'content-length': '10485760',
    'server': 'CloudianS3'},
  'RetryAttempts': 0},
  'AcceptRanges': 'bytes',

  'LastModified': datetime.datetime(2020, 12, 13, 22, 40, 52),
  'ContentLength': 10485760,
  'ETag': '"669fdad9e309b552f1e9cf7b489c1f73-2"',
  'VersionId': 'fe14c26b-bed0-c73f-a754-06bdfcde1d5e',
  'ContentType': 'binary/octet-stream',
  'ServerSideEncryption': 'AES256',
  'Metadata': {'Engineer': 'yamahiro',
    'Company': 'nw',
    'Purpose': 'boto3 demo 1'}}
```

補足: メタデータのみ選択して表示

以下の例は、バージョン ID を指定してオブジェクトのメタデータのみを表示させています。

```
print("~ メタデータのみ選択して表示 ~")
ret = client.head_object(Bucket='pythonbucket2fix',
                        Key='10mb.dat',
                        VersionId='fe14c26b-bba0-6edf-a754-06bdfcd6
                        )['Metadata']
print(ret)
ret = client.head_object(Bucket='pythonbucket2fix',
                        Key='10mb.dat',
                        VersionId='fe14c26b-bd5e-4b7f-a754-06bdfcd6
                        )['Metadata']
print(ret)
ret = client.head_object(Bucket='pythonbucket2fix',
                        Key='10mb.dat',
                        VersionId='fe14c26b-bed0-c73f-a754-06bdfcd6
                        )['Metadata']
print(ret)
ret = client.head_object(Bucket='pythonbucket2fix',
                        Key='10mb.dat',
                        VersionId='fe14c26b-c075-286f-a754-06bdfcd6
                        )['Metadata']
print(ret)
```

```
~ メタデータのみ選択して表示 ~
{}
{'Engineer': 'miki', 'Company': 'nw', 'Purpose': 'boto3 der
{'Engineer': 'yamahiro', 'Company': 'nw', 'Purpose': 'boto3
{}
```

補足: get_object と head_object の違い

以下のように"get_object"メソッドでも同様の情報を取得できますが、"head_object"はオブジェクト自体を戻さずにメタデータのみを読み取ることができます。

```
ret = client.get_object(Bucket='pythonbucket2fix',
                        Key='10mb.dat',
                        VersionId='fe14c26b-bed0-c73f-a754-06bdfcd6
                        )

print(ret)
```

```
{'ResponseMetadata': {'RequestId': '9dad3fed-0e30-1dbc-a754'}}
```

メタデータのみ読み取りたい場合は、“head_object”を使用するほうがお薦めです。

```
ret = client.get_object(Bucket='pythonbucket2fix',
                        Key='10mb.dat',
                        VersionId='fe14c26b-bed0-c73f-a754-06bdfcd6
                        )['Metadata']

print(ret)
```

```
{'Engineer': 'yamahiro', 'Company': 'nw', 'Purpose': 'boto3'}
```

まとめ

Python(boto3)で、バケットのバージョニング設定をしました。

Python(boto3)で、オブジェクト公開のための署名付きURLを生成してみる

CloudianはS3完全互換なので、AWS SDKを使えて便利ですよ！[前回](#)は、Python(boto3)で、オブジェクトのメタデータを表示してみました。

今回は、Python(boto3)で、オブジェクトをWEB公開するための、署名付きURLを生成してみようと思います。

事前署名付き URL の生成 / generate_presigned_url ()

Cloudian に格納されたオブジェクトの共有に使用できるパブリックURLを、generate_presigned_url()を呼び出して生成することができます。

1. デフォルトの有効期間(3,600 秒)で事前署名付き URL を生成

以下の例では、generate_presigned_url()のクライアントメソッドに「get_object」を設定し、バケット「pythonbucket3」に保存されているキー「HyperStoreInstallGuide_v-7.2.1.pdf」のオブジェクトを取得できる事前署名付きURLを生成しています。

有効期間を指定せずにURLを生成した場合、そのURLの有効期間はデフォルトで3,600秒(1時間)に設定されます。

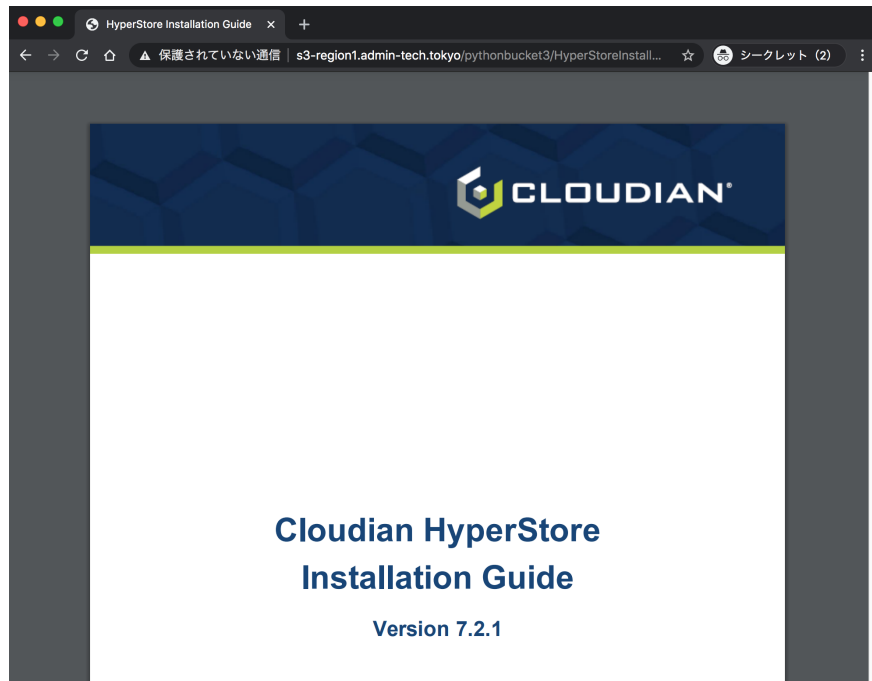
```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# オブジェクトWEB公開のための署名付きURLを生成
client.generate_presigned_url(
    'get_object',
    Params={
        'Bucket': 'pythonbucket3',
        'Key': 'HyperStoreInstallGuide_v-7.2.1.pdf'
    }
)
```

```
'http://s3-region1.admin-tech.tokyo/pythonbucket3/HyperStoreInsta...
```

この例で生成された URL を開くと、下図のように指定したオブジェクトを取得することができます (ブラウザで表示可能な PDF であるため、ブラウザで開いています)



2. 有効期限を 2 日間(7,200 秒)に設定して事前署名付き URL を生成

以下の例では、`generate_presigned_url()`の引数に 2 日間の有効期限 (`ExpiresIn=7200`)を設定し、事前署名付き URL を生成しています。

```
import boto3

client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# オブジェクトWEB公開のための署名付きURLを生成
client.generate_presigned_url(
    'get_object',
    Params={
        'Bucket': 'pythonbucket3',
        'Key': 'HyperStoreInstallGuide_v-7.2.1.pdf',
    },
    ExpiresIn=7200
)

'http://s3-region1.admin-tech.tokyo/pythonbucket3/HyperStoreInstallGuide_v-7.2.1.pdf'
```

3. 有効期限を 2 分(120 秒)に設定して事前署名付き URL を生成

以下の例では、generate_presigned_url()の引数に非常に短時間の有効期限 (この例の場合、「ExpiresIn=120」)を設定し、事前署名付き URL を生成してその挙動を確認しています。

有効期限 2 分の事前署名付き URL を生成

```
import boto3

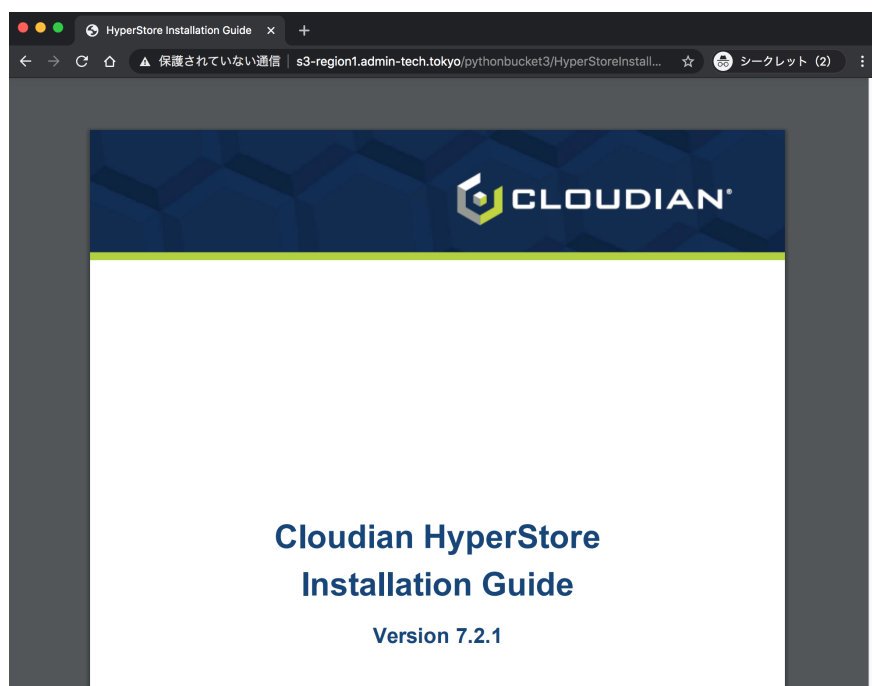
client = boto3.client(
    's3',
    endpoint_url='https://xxx.yyy.com'
)

# オブジェクトWEB公開のための署名付きURLを生成
client.generate_presigned_url(
    'get_object',
    Params={
        'Bucket': 'pythonbucket3',
        'Key': 'HyperStoreInstallGuide_v-7.2.1.pdf',
    },
    ExpiresIn=120
)

'http://s3-region1.admin-tech.tokyo/pythonbucket3/HyperStoreInstallGuide_v-7.2.1.pdf'
```

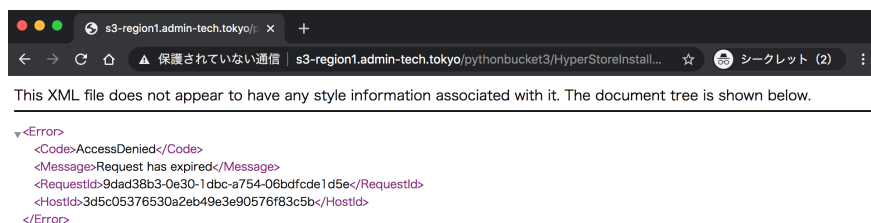
生成されたURLに、2分以内にアクセス

事前署名付きURLが指しているオブジェクトがWebブラウザで参照できるPDF形式であるため、WebブラウザでPDFが開かれています。



生成されたURLに、2分以上経過した後にアクセス

以下のようなエラーが返されて、オブジェクトへのアクセスが拒否されます。事前署名付きURLを生成した際に「ExpiresIn=120」を指定しているため、2分以上経過したアクセスは以下のように拒否されます。



まとめ

Python(boto3)で、オブジェクトWEB公開のための署名付きURLを生成してみました。

Clouddianオブジェクトストレージ/アップロードの進捗状況を表示してみる

アップロードの進捗状況表示

200MB のファイル「200mb.dat」を、オブジェクトストレージ/Clouddian上のバケット名「boto3」にアップロードする Python プログラムです。

アップロード先のバケット名「boto3」が存在しなければ、アップロード前にバケットを作成します。マルチパートアップロード時のオプションを変更するために、TransferConfig オブジェクトを作成して オプション値を設定しています。

S3Transfer の `upload_file()` では、8MB(`multipart_threshold = 8 1024 1024`)以上のファイルであればマルチパートでのアップロードを行います。マルチパートに分割されるチャンクのサイズは、8MB(`multipart_chunksize = 8388608`)になるように設定されています。

`upload_file()` ではアップロードされるファイルの ACL を「public-read」に設定し、いくつかのメタデータを付与し、AES256 のサーバーサイド暗号化を有効にしています。また、`upload_file()` のコールバック関数 `ProgressPercentage(upfile)` を設定しており、`ProgressPercentage`関数を実行して、アップロードの進捗状況を表示しています。

```

import os.path
import threading
import sys

import boto3, botocore
from boto3.s3.transfer import S3Transfer
from boto3.s3.transfer import TransferConfig

##### コールバック関数 #####
class ProgressPercentage(object):

    def __init__(self, filename):
        self._filename = filename
        self._size = float(os.path.getsize(filename))
        self._seen_so_far = 0
        self._lock = threading.Lock()

    def __call__(self, bytes_amount):
        # To simplify we'll assume this is hooked up
        # to a single filename.

        with self._lock:
            self._seen_so_far += bytes_amount
            percentage = (self._seen_so_far / self._size) * 100
            sys.stdout.write("%s    %s / %s  (%.2f%%)\n" % (self._filename,
                                                            self._seen_so_far,
                                                            self._size,
                                                            percentage))
            sys.stdout.flush()

##### メイン処理 #####

# boto3オブジェクト作成
client = boto3.client(
    's3',
    endpoint_url='http://s3-region1.admin-tech.tokyo',

    # 認証情報の直接記述は推奨されない (テスト目的)
    aws_access_key_id='アクセスキーを記述',
    aws_secret_access_key='シークレットキーを記述'
)

# 変数セット
bucket = 'boto3'
upfile = './fileup/200mb.dat'
fname, ext = os.path.splitext(os.path.basename(upfile))
print(fname, ext)
key = fname + '_mpu' + ext

```

```
try:
    # バケットの存在確認
    client.head_bucket(Bucket = bucket)

except botocore.exceptions.ClientError:
    # バケットが存在しなければ、新規作成
    client.create_bucket(
        ACL='private',
        Bucket=bucket,
        CreateBucketConfiguration={
            'LocationConstraint': 'region1'
        }
    )

# マルチパートでアップロード
config = TransferConfig(
    multipart_threshold = 8 * 1024 * 1024,
    max_concurrency = 10,
    multipart_chunksize = 8388608,
    num_download_attempts = 10,
    max_io_queue = 100
)

transfer = S3Transfer(client, config)

transfer.upload_file(
    upfile,
    bucket,
    key,
    extra_args={
        'ACL': 'public-read',
        'Metadata': {
            'Purpose': 'Upload test',
            'Engineer': 'Ryosuke Matsui',
            'Company': 'Cloudian K.K.'},
        'ServerSideEncryption': 'AES256'
    },
    callback=ProgressPercentage(upfile)
)
```



```

200mb .dat
./fileup/200mb.dat 262144 / 209715200.0 (0.12%)
./fileup/200mb.dat 524288 / 209715200.0 (0.25%)
./fileup/200mb.dat 786432 / 209715200.0 (0.38%)
./fileup/200mb.dat 1048576 / 209715200.0 (0.50%)
./fileup/200mb.dat 1310720 / 209715200.0 (0.62%)
./fileup/200mb.dat 1572864 / 209715200.0 (0.75%)
./fileup/200mb.dat 1835008 / 209715200.0 (0.88%)
./fileup/200mb.dat 2097152 / 209715200.0 (1.00%)
./fileup/200mb.dat 2359296 / 209715200.0 (1.12%)
./fileup/200mb.dat 2621440 / 209715200.0 (1.25%)
./fileup/200mb.dat 2883584 / 209715200.0 (1.38%)
./fileup/200mb.dat 3145728 / 209715200.0 (1.50%)
./fileup/200mb.dat 3407872 / 209715200.0 (1.62%)
./fileup/200mb.dat 3670016 / 209715200.0 (1.75%)
./fileup/200mb.dat 3932160 / 209715200.0 (1.88%)
./fileup/200mb.dat 4194304 / 209715200.0 (2.00%)
./fileup/200mb.dat 4456448 / 209715200.0 (2.12%)
./fileup/200mb.dat 4718592 / 209715200.0 (2.25%)
./fileup/200mb.dat 4980736 / 209715200.0 (2.38%)
./fileup/200mb.dat 5242880 / 209715200.0 (2.50%)
./fileup/200mb.dat 5505024 / 209715200.0 (2.62%)
./fileup/200mb.dat 5767168 / 209715200.0 (2.75%)
./fileup/200mb.dat 6029312 / 209715200.0 (2.88%)
./fileup/200mb.dat 6291456 / 209715200.0 (3.00%)
./fileup/200mb.dat 6553600 / 209715200.0 (3.12%)
./fileup/200mb.dat 6815744 / 209715200.0 (3.25%)
./fileup/200mb.dat 7077888 / 209715200.0 (3.38%)
./fileup/200mb.dat 7340032 / 209715200.0 (3.50%)
./fileup/200mb.dat 7602176 / 209715200.0 (3.62%)
./fileup/200mb.dat 7864320 / 209715200.0 (3.75%)
./fileup/200mb.dat 8126464 / 209715200.0 (3.88%)
./fileup/200mb.dat 8388608 / 209715200.0 (4.00%)
./fileup/200mb.dat 8650752 / 209715200.0 (4.12%)
./fileup/200mb.dat 8912896 / 209715200.0 (4.25%)
./fileup/200mb.dat 9175040 / 209715200.0 (4.38%)
./fileup/200mb.dat 9437184 / 209715200.0 (4.50%)
./fileup/200mb.dat 9699328 / 209715200.0 (4.62%)
./fileup/200mb.dat 9961472 / 209715200.0 (4.75%)
./fileup/200mb.dat 10223616 / 209715200.0 (4.88%)
./fileup/200mb.dat 10485760 / 209715200.0 (5.00%)
./fileup/200mb.dat 10747904 / 209715200.0 (5.12%)
./fileup/200mb.dat 11010048 / 209715200.0 (5.25%)
./fileup/200mb.dat 11272192 / 209715200.0 (5.38%)
./fileup/200mb.dat 11534336 / 209715200.0 (5.50%)
./fileup/200mb.dat 11796480 / 209715200.0 (5.62%)
./fileup/200mb.dat 12058624 / 209715200.0 (5.75%)
./fileup/200mb.dat 12320768 / 209715200.0 (5.88%)

```

./fileup/200mb.dat	12582912	/	209715200.0	(6.00%)
./fileup/200mb.dat	12845056	/	209715200.0	(6.12%)
./fileup/200mb.dat	13107200	/	209715200.0	(6.25%)
./fileup/200mb.dat	13369344	/	209715200.0	(6.38%)
./fileup/200mb.dat	13631488	/	209715200.0	(6.50%)
./fileup/200mb.dat	13893632	/	209715200.0	(6.62%)
./fileup/200mb.dat	14155776	/	209715200.0	(6.75%)
./fileup/200mb.dat	14417920	/	209715200.0	(6.88%)
./fileup/200mb.dat	14680064	/	209715200.0	(7.00%)
./fileup/200mb.dat	14942208	/	209715200.0	(7.12%)
./fileup/200mb.dat	15204352	/	209715200.0	(7.25%)
./fileup/200mb.dat	15466496	/	209715200.0	(7.38%)
./fileup/200mb.dat	15728640	/	209715200.0	(7.50%)
./fileup/200mb.dat	15990784	/	209715200.0	(7.62%)
./fileup/200mb.dat	16252928	/	209715200.0	(7.75%)
./fileup/200mb.dat	16515072	/	209715200.0	(7.88%)
./fileup/200mb.dat	16777216	/	209715200.0	(8.00%)
./fileup/200mb.dat	17039360	/	209715200.0	(8.12%)
./fileup/200mb.dat	17301504	/	209715200.0	(8.25%)
./fileup/200mb.dat	17563648	/	209715200.0	(8.38%)
./fileup/200mb.dat	17825792	/	209715200.0	(8.50%)
./fileup/200mb.dat	18087936	/	209715200.0	(8.62%)
./fileup/200mb.dat	18350080	/	209715200.0	(8.75%)
./fileup/200mb.dat	18612224	/	209715200.0	(8.88%)
./fileup/200mb.dat	18874368	/	209715200.0	(9.00%)
./fileup/200mb.dat	19136512	/	209715200.0	(9.12%)
./fileup/200mb.dat	19398656	/	209715200.0	(9.25%)
./fileup/200mb.dat	19660800	/	209715200.0	(9.38%)
./fileup/200mb.dat	19922944	/	209715200.0	(9.50%)
./fileup/200mb.dat	20185088	/	209715200.0	(9.62%)
./fileup/200mb.dat	20447232	/	209715200.0	(9.75%)
./fileup/200mb.dat	20709376	/	209715200.0	(9.88%)
./fileup/200mb.dat	20971520	/	209715200.0	(10.00%)
./fileup/200mb.dat	21233664	/	209715200.0	(10.12%)
./fileup/200mb.dat	21495808	/	209715200.0	(10.25%)
./fileup/200mb.dat	21757952	/	209715200.0	(10.38%)
./fileup/200mb.dat	22020096	/	209715200.0	(10.50%)
./fileup/200mb.dat	22282240	/	209715200.0	(10.62%)
./fileup/200mb.dat	22544384	/	209715200.0	(10.75%)
./fileup/200mb.dat	22806528	/	209715200.0	(10.88%)
./fileup/200mb.dat	23068672	/	209715200.0	(11.00%)
./fileup/200mb.dat	23330816	/	209715200.0	(11.12%)
./fileup/200mb.dat	23592960	/	209715200.0	(11.25%)
./fileup/200mb.dat	23855104	/	209715200.0	(11.38%)
./fileup/200mb.dat	24117248	/	209715200.0	(11.50%)
./fileup/200mb.dat	24379392	/	209715200.0	(11.62%)
./fileup/200mb.dat	24641536	/	209715200.0	(11.75%)
./fileup/200mb.dat	24903680	/	209715200.0	(11.88%)

./fileup/200mb.dat	25165824	/	209715200.0	(12.00%)
./fileup/200mb.dat	25427968	/	209715200.0	(12.12%)
./fileup/200mb.dat	25690112	/	209715200.0	(12.25%)
./fileup/200mb.dat	25952256	/	209715200.0	(12.38%)
./fileup/200mb.dat	26214400	/	209715200.0	(12.50%)
./fileup/200mb.dat	26476544	/	209715200.0	(12.62%)
./fileup/200mb.dat	26738688	/	209715200.0	(12.75%)
./fileup/200mb.dat	27000832	/	209715200.0	(12.88%)
./fileup/200mb.dat	27262976	/	209715200.0	(13.00%)
./fileup/200mb.dat	27525120	/	209715200.0	(13.12%)
./fileup/200mb.dat	27787264	/	209715200.0	(13.25%)
./fileup/200mb.dat	28049408	/	209715200.0	(13.38%)
./fileup/200mb.dat	28311552	/	209715200.0	(13.50%)
./fileup/200mb.dat	28573696	/	209715200.0	(13.63%)
./fileup/200mb.dat	28835840	/	209715200.0	(13.75%)
./fileup/200mb.dat	29097984	/	209715200.0	(13.88%)
./fileup/200mb.dat	29360128	/	209715200.0	(14.00%)
./fileup/200mb.dat	29622272	/	209715200.0	(14.12%)
./fileup/200mb.dat	29884416	/	209715200.0	(14.25%)
./fileup/200mb.dat	30146560	/	209715200.0	(14.37%)
./fileup/200mb.dat	30408704	/	209715200.0	(14.50%)
./fileup/200mb.dat	30670848	/	209715200.0	(14.62%)
./fileup/200mb.dat	30932992	/	209715200.0	(14.75%)
./fileup/200mb.dat	31195136	/	209715200.0	(14.88%)
./fileup/200mb.dat	31457280	/	209715200.0	(15.00%)
./fileup/200mb.dat	31719424	/	209715200.0	(15.12%)
./fileup/200mb.dat	31981568	/	209715200.0	(15.25%)
./fileup/200mb.dat	32243712	/	209715200.0	(15.38%)
./fileup/200mb.dat	32505856	/	209715200.0	(15.50%)
./fileup/200mb.dat	32768000	/	209715200.0	(15.62%)
./fileup/200mb.dat	33030144	/	209715200.0	(15.75%)
./fileup/200mb.dat	33292288	/	209715200.0	(15.88%)
./fileup/200mb.dat	33554432	/	209715200.0	(16.00%)
./fileup/200mb.dat	33816576	/	209715200.0	(16.12%)
./fileup/200mb.dat	34078720	/	209715200.0	(16.25%)
./fileup/200mb.dat	34340864	/	209715200.0	(16.38%)
./fileup/200mb.dat	34603008	/	209715200.0	(16.50%)
./fileup/200mb.dat	34865152	/	209715200.0	(16.62%)
./fileup/200mb.dat	35127296	/	209715200.0	(16.75%)
./fileup/200mb.dat	35389440	/	209715200.0	(16.88%)
./fileup/200mb.dat	35651584	/	209715200.0	(17.00%)
./fileup/200mb.dat	35913728	/	209715200.0	(17.12%)
./fileup/200mb.dat	36175872	/	209715200.0	(17.25%)
./fileup/200mb.dat	36438016	/	209715200.0	(17.38%)
./fileup/200mb.dat	36700160	/	209715200.0	(17.50%)
./fileup/200mb.dat	36962304	/	209715200.0	(17.62%)
./fileup/200mb.dat	37224448	/	209715200.0	(17.75%)
./fileup/200mb.dat	37486592	/	209715200.0	(17.88%)

./fileup/200mb.dat	37748736	/	209715200.0	(18.00%)
./fileup/200mb.dat	38010880	/	209715200.0	(18.12%)
./fileup/200mb.dat	38273024	/	209715200.0	(18.25%)
./fileup/200mb.dat	38535168	/	209715200.0	(18.38%)
./fileup/200mb.dat	38797312	/	209715200.0	(18.50%)
./fileup/200mb.dat	39059456	/	209715200.0	(18.62%)
./fileup/200mb.dat	39321600	/	209715200.0	(18.75%)
./fileup/200mb.dat	39583744	/	209715200.0	(18.88%)
./fileup/200mb.dat	39845888	/	209715200.0	(19.00%)
./fileup/200mb.dat	40108032	/	209715200.0	(19.12%)
./fileup/200mb.dat	40370176	/	209715200.0	(19.25%)
./fileup/200mb.dat	40632320	/	209715200.0	(19.38%)
./fileup/200mb.dat	40894464	/	209715200.0	(19.50%)
./fileup/200mb.dat	41156608	/	209715200.0	(19.62%)
./fileup/200mb.dat	41418752	/	209715200.0	(19.75%)
./fileup/200mb.dat	41680896	/	209715200.0	(19.88%)
./fileup/200mb.dat	41943040	/	209715200.0	(20.00%)
./fileup/200mb.dat	42205184	/	209715200.0	(20.12%)
./fileup/200mb.dat	42467328	/	209715200.0	(20.25%)
./fileup/200mb.dat	42729472	/	209715200.0	(20.38%)
./fileup/200mb.dat	42991616	/	209715200.0	(20.50%)
./fileup/200mb.dat	43253760	/	209715200.0	(20.62%)
./fileup/200mb.dat	43515904	/	209715200.0	(20.75%)
./fileup/200mb.dat	43778048	/	209715200.0	(20.88%)
./fileup/200mb.dat	44040192	/	209715200.0	(21.00%)
./fileup/200mb.dat	44302336	/	209715200.0	(21.12%)
./fileup/200mb.dat	44564480	/	209715200.0	(21.25%)
./fileup/200mb.dat	44826624	/	209715200.0	(21.38%)
./fileup/200mb.dat	45088768	/	209715200.0	(21.50%)
./fileup/200mb.dat	45350912	/	209715200.0	(21.62%)
./fileup/200mb.dat	45613056	/	209715200.0	(21.75%)
./fileup/200mb.dat	45875200	/	209715200.0	(21.88%)
./fileup/200mb.dat	46137344	/	209715200.0	(22.00%)
./fileup/200mb.dat	46399488	/	209715200.0	(22.12%)
./fileup/200mb.dat	46661632	/	209715200.0	(22.25%)
./fileup/200mb.dat	46923776	/	209715200.0	(22.38%)
./fileup/200mb.dat	47185920	/	209715200.0	(22.50%)
./fileup/200mb.dat	47448064	/	209715200.0	(22.62%)
./fileup/200mb.dat	47710208	/	209715200.0	(22.75%)
./fileup/200mb.dat	47972352	/	209715200.0	(22.88%)
./fileup/200mb.dat	48234496	/	209715200.0	(23.00%)
./fileup/200mb.dat	48496640	/	209715200.0	(23.12%)
./fileup/200mb.dat	48758784	/	209715200.0	(23.25%)
./fileup/200mb.dat	49020928	/	209715200.0	(23.38%)
./fileup/200mb.dat	49283072	/	209715200.0	(23.50%)
./fileup/200mb.dat	49545216	/	209715200.0	(23.62%)
./fileup/200mb.dat	49807360	/	209715200.0	(23.75%)
./fileup/200mb.dat	50069504	/	209715200.0	(23.88%)

./fileup/200mb.dat	50331648	/	209715200.0	(24.00%)
./fileup/200mb.dat	50593792	/	209715200.0	(24.12%)
./fileup/200mb.dat	50855936	/	209715200.0	(24.25%)
./fileup/200mb.dat	51118080	/	209715200.0	(24.38%)
./fileup/200mb.dat	51380224	/	209715200.0	(24.50%)
./fileup/200mb.dat	51642368	/	209715200.0	(24.62%)
./fileup/200mb.dat	51904512	/	209715200.0	(24.75%)
./fileup/200mb.dat	52166656	/	209715200.0	(24.88%)
./fileup/200mb.dat	52428800	/	209715200.0	(25.00%)
./fileup/200mb.dat	52690944	/	209715200.0	(25.12%)
./fileup/200mb.dat	52953088	/	209715200.0	(25.25%)
./fileup/200mb.dat	53215232	/	209715200.0	(25.37%)
./fileup/200mb.dat	53477376	/	209715200.0	(25.50%)
./fileup/200mb.dat	53739520	/	209715200.0	(25.62%)
./fileup/200mb.dat	54001664	/	209715200.0	(25.75%)
./fileup/200mb.dat	54263808	/	209715200.0	(25.87%)
./fileup/200mb.dat	54525952	/	209715200.0	(26.00%)
./fileup/200mb.dat	54788096	/	209715200.0	(26.12%)
./fileup/200mb.dat	55050240	/	209715200.0	(26.25%)
./fileup/200mb.dat	55312384	/	209715200.0	(26.38%)
./fileup/200mb.dat	55574528	/	209715200.0	(26.50%)
./fileup/200mb.dat	55836672	/	209715200.0	(26.62%)
./fileup/200mb.dat	56098816	/	209715200.0	(26.75%)
./fileup/200mb.dat	56360960	/	209715200.0	(26.88%)
./fileup/200mb.dat	56623104	/	209715200.0	(27.00%)
./fileup/200mb.dat	56885248	/	209715200.0	(27.12%)
./fileup/200mb.dat	57147392	/	209715200.0	(27.25%)
./fileup/200mb.dat	57409536	/	209715200.0	(27.38%)
./fileup/200mb.dat	57671680	/	209715200.0	(27.50%)
./fileup/200mb.dat	57933824	/	209715200.0	(27.62%)
./fileup/200mb.dat	58195968	/	209715200.0	(27.75%)
./fileup/200mb.dat	58458112	/	209715200.0	(27.88%)
./fileup/200mb.dat	58720256	/	209715200.0	(28.00%)
./fileup/200mb.dat	58982400	/	209715200.0	(28.12%)
./fileup/200mb.dat	59244544	/	209715200.0	(28.25%)
./fileup/200mb.dat	59506688	/	209715200.0	(28.38%)
./fileup/200mb.dat	59768832	/	209715200.0	(28.50%)
./fileup/200mb.dat	60030976	/	209715200.0	(28.62%)
./fileup/200mb.dat	60293120	/	209715200.0	(28.75%)
./fileup/200mb.dat	60555264	/	209715200.0	(28.88%)
./fileup/200mb.dat	60817408	/	209715200.0	(29.00%)
./fileup/200mb.dat	61079552	/	209715200.0	(29.12%)
./fileup/200mb.dat	61341696	/	209715200.0	(29.25%)
./fileup/200mb.dat	61603840	/	209715200.0	(29.38%)
./fileup/200mb.dat	61865984	/	209715200.0	(29.50%)
./fileup/200mb.dat	62128128	/	209715200.0	(29.62%)
./fileup/200mb.dat	62390272	/	209715200.0	(29.75%)
./fileup/200mb.dat	62652416	/	209715200.0	(29.88%)

./fileup/200mb.dat	62914560	/	209715200.0	(30.00%)
./fileup/200mb.dat	63176704	/	209715200.0	(30.12%)
./fileup/200mb.dat	63438848	/	209715200.0	(30.25%)
./fileup/200mb.dat	63700992	/	209715200.0	(30.38%)
./fileup/200mb.dat	63963136	/	209715200.0	(30.50%)
./fileup/200mb.dat	64225280	/	209715200.0	(30.63%)
./fileup/200mb.dat	64487424	/	209715200.0	(30.75%)
./fileup/200mb.dat	64749568	/	209715200.0	(30.88%)
./fileup/200mb.dat	65011712	/	209715200.0	(31.00%)
./fileup/200mb.dat	65273856	/	209715200.0	(31.13%)
./fileup/200mb.dat	65536000	/	209715200.0	(31.25%)
./fileup/200mb.dat	65798144	/	209715200.0	(31.37%)
./fileup/200mb.dat	66060288	/	209715200.0	(31.50%)
./fileup/200mb.dat	66322432	/	209715200.0	(31.62%)
./fileup/200mb.dat	66584576	/	209715200.0	(31.75%)
./fileup/200mb.dat	66846720	/	209715200.0	(31.87%)
./fileup/200mb.dat	67108864	/	209715200.0	(32.00%)
./fileup/200mb.dat	67371008	/	209715200.0	(32.12%)
./fileup/200mb.dat	67633152	/	209715200.0	(32.25%)
./fileup/200mb.dat	67895296	/	209715200.0	(32.38%)
./fileup/200mb.dat	68157440	/	209715200.0	(32.50%)
./fileup/200mb.dat	68419584	/	209715200.0	(32.62%)
./fileup/200mb.dat	68681728	/	209715200.0	(32.75%)
./fileup/200mb.dat	68943872	/	209715200.0	(32.88%)
./fileup/200mb.dat	69206016	/	209715200.0	(33.00%)
./fileup/200mb.dat	69468160	/	209715200.0	(33.12%)
./fileup/200mb.dat	69730304	/	209715200.0	(33.25%)
./fileup/200mb.dat	69992448	/	209715200.0	(33.38%)
./fileup/200mb.dat	70254592	/	209715200.0	(33.50%)
./fileup/200mb.dat	70516736	/	209715200.0	(33.62%)
./fileup/200mb.dat	70778880	/	209715200.0	(33.75%)
./fileup/200mb.dat	71041024	/	209715200.0	(33.88%)
./fileup/200mb.dat	71303168	/	209715200.0	(34.00%)
./fileup/200mb.dat	71565312	/	209715200.0	(34.12%)
./fileup/200mb.dat	71827456	/	209715200.0	(34.25%)
./fileup/200mb.dat	72089600	/	209715200.0	(34.38%)
./fileup/200mb.dat	72351744	/	209715200.0	(34.50%)
./fileup/200mb.dat	72613888	/	209715200.0	(34.62%)
./fileup/200mb.dat	72876032	/	209715200.0	(34.75%)
./fileup/200mb.dat	73138176	/	209715200.0	(34.88%)
./fileup/200mb.dat	73400320	/	209715200.0	(35.00%)
./fileup/200mb.dat	73662464	/	209715200.0	(35.12%)
./fileup/200mb.dat	73924608	/	209715200.0	(35.25%)
./fileup/200mb.dat	74186752	/	209715200.0	(35.38%)
./fileup/200mb.dat	74448896	/	209715200.0	(35.50%)
./fileup/200mb.dat	74711040	/	209715200.0	(35.62%)
./fileup/200mb.dat	74973184	/	209715200.0	(35.75%)
./fileup/200mb.dat	75235328	/	209715200.0	(35.88%)

./fileup/200mb.dat	75497472	/	209715200.0	(36.00%)
./fileup/200mb.dat	75759616	/	209715200.0	(36.12%)
./fileup/200mb.dat	76021760	/	209715200.0	(36.25%)
./fileup/200mb.dat	76283904	/	209715200.0	(36.38%)
./fileup/200mb.dat	76546048	/	209715200.0	(36.50%)
./fileup/200mb.dat	76808192	/	209715200.0	(36.62%)
./fileup/200mb.dat	77070336	/	209715200.0	(36.75%)
./fileup/200mb.dat	77332480	/	209715200.0	(36.88%)
./fileup/200mb.dat	77594624	/	209715200.0	(37.00%)
./fileup/200mb.dat	77856768	/	209715200.0	(37.12%)
./fileup/200mb.dat	78118912	/	209715200.0	(37.25%)
./fileup/200mb.dat	78381056	/	209715200.0	(37.38%)
./fileup/200mb.dat	78643200	/	209715200.0	(37.50%)
./fileup/200mb.dat	78905344	/	209715200.0	(37.62%)
./fileup/200mb.dat	79167488	/	209715200.0	(37.75%)
./fileup/200mb.dat	79429632	/	209715200.0	(37.88%)
./fileup/200mb.dat	79691776	/	209715200.0	(38.00%)
./fileup/200mb.dat	79953920	/	209715200.0	(38.12%)
./fileup/200mb.dat	80216064	/	209715200.0	(38.25%)
./fileup/200mb.dat	80478208	/	209715200.0	(38.38%)
./fileup/200mb.dat	80740352	/	209715200.0	(38.50%)
./fileup/200mb.dat	81002496	/	209715200.0	(38.62%)
./fileup/200mb.dat	81264640	/	209715200.0	(38.75%)
./fileup/200mb.dat	81526784	/	209715200.0	(38.88%)
./fileup/200mb.dat	81788928	/	209715200.0	(39.00%)
./fileup/200mb.dat	82051072	/	209715200.0	(39.12%)
./fileup/200mb.dat	82313216	/	209715200.0	(39.25%)
./fileup/200mb.dat	82575360	/	209715200.0	(39.38%)
./fileup/200mb.dat	82837504	/	209715200.0	(39.50%)
./fileup/200mb.dat	83099648	/	209715200.0	(39.62%)
./fileup/200mb.dat	83361792	/	209715200.0	(39.75%)
./fileup/200mb.dat	83623936	/	209715200.0	(39.88%)
./fileup/200mb.dat	83886080	/	209715200.0	(40.00%)
./fileup/200mb.dat	84148224	/	209715200.0	(40.12%)
./fileup/200mb.dat	84410368	/	209715200.0	(40.25%)
./fileup/200mb.dat	84672512	/	209715200.0	(40.38%)
./fileup/200mb.dat	84934656	/	209715200.0	(40.50%)
./fileup/200mb.dat	85196800	/	209715200.0	(40.62%)
./fileup/200mb.dat	85458944	/	209715200.0	(40.75%)
./fileup/200mb.dat	85721088	/	209715200.0	(40.88%)
./fileup/200mb.dat	85983232	/	209715200.0	(41.00%)
./fileup/200mb.dat	86245376	/	209715200.0	(41.12%)
./fileup/200mb.dat	86507520	/	209715200.0	(41.25%)
./fileup/200mb.dat	86769664	/	209715200.0	(41.38%)
./fileup/200mb.dat	87031808	/	209715200.0	(41.50%)
./fileup/200mb.dat	87293952	/	209715200.0	(41.62%)
./fileup/200mb.dat	87556096	/	209715200.0	(41.75%)
./fileup/200mb.dat	87818240	/	209715200.0	(41.88%)

./fileup/200mb.dat	88080384	/	209715200.0	(42.00%)
./fileup/200mb.dat	88342528	/	209715200.0	(42.12%)
./fileup/200mb.dat	88604672	/	209715200.0	(42.25%)
./fileup/200mb.dat	88866816	/	209715200.0	(42.38%)
./fileup/200mb.dat	89128960	/	209715200.0	(42.50%)
./fileup/200mb.dat	89391104	/	209715200.0	(42.62%)
./fileup/200mb.dat	89653248	/	209715200.0	(42.75%)
./fileup/200mb.dat	89915392	/	209715200.0	(42.88%)
./fileup/200mb.dat	90177536	/	209715200.0	(43.00%)
./fileup/200mb.dat	90439680	/	209715200.0	(43.12%)
./fileup/200mb.dat	90701824	/	209715200.0	(43.25%)
./fileup/200mb.dat	90963968	/	209715200.0	(43.38%)
./fileup/200mb.dat	91226112	/	209715200.0	(43.50%)
./fileup/200mb.dat	91488256	/	209715200.0	(43.62%)
./fileup/200mb.dat	91750400	/	209715200.0	(43.75%)
./fileup/200mb.dat	92012544	/	209715200.0	(43.88%)
./fileup/200mb.dat	92274688	/	209715200.0	(44.00%)
./fileup/200mb.dat	92536832	/	209715200.0	(44.12%)
./fileup/200mb.dat	92798976	/	209715200.0	(44.25%)
./fileup/200mb.dat	93061120	/	209715200.0	(44.38%)
./fileup/200mb.dat	93323264	/	209715200.0	(44.50%)
./fileup/200mb.dat	93585408	/	209715200.0	(44.62%)
./fileup/200mb.dat	93847552	/	209715200.0	(44.75%)
./fileup/200mb.dat	94109696	/	209715200.0	(44.88%)
./fileup/200mb.dat	94371840	/	209715200.0	(45.00%)
./fileup/200mb.dat	94633984	/	209715200.0	(45.12%)
./fileup/200mb.dat	94896128	/	209715200.0	(45.25%)
./fileup/200mb.dat	95158272	/	209715200.0	(45.38%)
./fileup/200mb.dat	95420416	/	209715200.0	(45.50%)
./fileup/200mb.dat	95682560	/	209715200.0	(45.62%)
./fileup/200mb.dat	95944704	/	209715200.0	(45.75%)
./fileup/200mb.dat	96206848	/	209715200.0	(45.88%)
./fileup/200mb.dat	96468992	/	209715200.0	(46.00%)
./fileup/200mb.dat	96731136	/	209715200.0	(46.12%)
./fileup/200mb.dat	96993280	/	209715200.0	(46.25%)
./fileup/200mb.dat	97255424	/	209715200.0	(46.38%)
./fileup/200mb.dat	97517568	/	209715200.0	(46.50%)
./fileup/200mb.dat	97779712	/	209715200.0	(46.62%)
./fileup/200mb.dat	98041856	/	209715200.0	(46.75%)
./fileup/200mb.dat	98304000	/	209715200.0	(46.88%)
./fileup/200mb.dat	98566144	/	209715200.0	(47.00%)
./fileup/200mb.dat	98828288	/	209715200.0	(47.12%)
./fileup/200mb.dat	99090432	/	209715200.0	(47.25%)
./fileup/200mb.dat	99352576	/	209715200.0	(47.38%)
./fileup/200mb.dat	99614720	/	209715200.0	(47.50%)
./fileup/200mb.dat	99876864	/	209715200.0	(47.62%)
./fileup/200mb.dat	100139008	/	209715200.0	(47.75%)
./fileup/200mb.dat	100401152	/	209715200.0	(47.88%)

./fileup/200mb.dat	100663296	/	209715200.0	(48.00%)
./fileup/200mb.dat	100925440	/	209715200.0	(48.12%)
./fileup/200mb.dat	101187584	/	209715200.0	(48.25%)
./fileup/200mb.dat	101449728	/	209715200.0	(48.38%)
./fileup/200mb.dat	101711872	/	209715200.0	(48.50%)
./fileup/200mb.dat	101974016	/	209715200.0	(48.62%)
./fileup/200mb.dat	102236160	/	209715200.0	(48.75%)
./fileup/200mb.dat	102498304	/	209715200.0	(48.88%)
./fileup/200mb.dat	102760448	/	209715200.0	(49.00%)
./fileup/200mb.dat	103022592	/	209715200.0	(49.12%)
./fileup/200mb.dat	103284736	/	209715200.0	(49.25%)
./fileup/200mb.dat	103546880	/	209715200.0	(49.38%)
./fileup/200mb.dat	103809024	/	209715200.0	(49.50%)
./fileup/200mb.dat	104071168	/	209715200.0	(49.62%)
./fileup/200mb.dat	104333312	/	209715200.0	(49.75%)
./fileup/200mb.dat	104595456	/	209715200.0	(49.88%)
./fileup/200mb.dat	104857600	/	209715200.0	(50.00%)
./fileup/200mb.dat	105119744	/	209715200.0	(50.12%)
./fileup/200mb.dat	105381888	/	209715200.0	(50.25%)
./fileup/200mb.dat	105644032	/	209715200.0	(50.38%)
./fileup/200mb.dat	105906176	/	209715200.0	(50.50%)
./fileup/200mb.dat	106168320	/	209715200.0	(50.62%)
./fileup/200mb.dat	106430464	/	209715200.0	(50.75%)
./fileup/200mb.dat	106692608	/	209715200.0	(50.88%)
./fileup/200mb.dat	106954752	/	209715200.0	(51.00%)
./fileup/200mb.dat	107216896	/	209715200.0	(51.12%)
./fileup/200mb.dat	107479040	/	209715200.0	(51.25%)
./fileup/200mb.dat	107741184	/	209715200.0	(51.38%)
./fileup/200mb.dat	108003328	/	209715200.0	(51.50%)
./fileup/200mb.dat	108265472	/	209715200.0	(51.62%)
./fileup/200mb.dat	108527616	/	209715200.0	(51.75%)
./fileup/200mb.dat	108789760	/	209715200.0	(51.88%)
./fileup/200mb.dat	109051904	/	209715200.0	(52.00%)
./fileup/200mb.dat	109314048	/	209715200.0	(52.12%)
./fileup/200mb.dat	109576192	/	209715200.0	(52.25%)
./fileup/200mb.dat	109838336	/	209715200.0	(52.38%)
./fileup/200mb.dat	110100480	/	209715200.0	(52.50%)
./fileup/200mb.dat	110362624	/	209715200.0	(52.62%)
./fileup/200mb.dat	110624768	/	209715200.0	(52.75%)
./fileup/200mb.dat	110886912	/	209715200.0	(52.88%)
./fileup/200mb.dat	111149056	/	209715200.0	(53.00%)
./fileup/200mb.dat	111411200	/	209715200.0	(53.12%)
./fileup/200mb.dat	111673344	/	209715200.0	(53.25%)
./fileup/200mb.dat	111935488	/	209715200.0	(53.37%)
./fileup/200mb.dat	112197632	/	209715200.0	(53.50%)
./fileup/200mb.dat	112459776	/	209715200.0	(53.62%)
./fileup/200mb.dat	112721920	/	209715200.0	(53.75%)
./fileup/200mb.dat	112984064	/	209715200.0	(53.87%)

./fileup/200mb.dat	113246208	/	209715200.0	(54.00%)
./fileup/200mb.dat	113508352	/	209715200.0	(54.12%)
./fileup/200mb.dat	113770496	/	209715200.0	(54.25%)
./fileup/200mb.dat	114032640	/	209715200.0	(54.37%)
./fileup/200mb.dat	114294784	/	209715200.0	(54.50%)
./fileup/200mb.dat	114556928	/	209715200.0	(54.62%)
./fileup/200mb.dat	114819072	/	209715200.0	(54.75%)
./fileup/200mb.dat	115081216	/	209715200.0	(54.87%)
./fileup/200mb.dat	115343360	/	209715200.0	(55.00%)
./fileup/200mb.dat	115605504	/	209715200.0	(55.12%)
./fileup/200mb.dat	115867648	/	209715200.0	(55.25%)
./fileup/200mb.dat	116129792	/	209715200.0	(55.38%)
./fileup/200mb.dat	116391936	/	209715200.0	(55.50%)
./fileup/200mb.dat	116654080	/	209715200.0	(55.62%)
./fileup/200mb.dat	116916224	/	209715200.0	(55.75%)
./fileup/200mb.dat	117178368	/	209715200.0	(55.88%)
./fileup/200mb.dat	117440512	/	209715200.0	(56.00%)
./fileup/200mb.dat	117702656	/	209715200.0	(56.12%)
./fileup/200mb.dat	117964800	/	209715200.0	(56.25%)
./fileup/200mb.dat	118226944	/	209715200.0	(56.38%)
./fileup/200mb.dat	118489088	/	209715200.0	(56.50%)
./fileup/200mb.dat	118751232	/	209715200.0	(56.62%)
./fileup/200mb.dat	119013376	/	209715200.0	(56.75%)
./fileup/200mb.dat	119275520	/	209715200.0	(56.88%)
./fileup/200mb.dat	119537664	/	209715200.0	(57.00%)
./fileup/200mb.dat	119799808	/	209715200.0	(57.12%)
./fileup/200mb.dat	120061952	/	209715200.0	(57.25%)
./fileup/200mb.dat	120324096	/	209715200.0	(57.38%)
./fileup/200mb.dat	120586240	/	209715200.0	(57.50%)
./fileup/200mb.dat	120848384	/	209715200.0	(57.63%)
./fileup/200mb.dat	121110528	/	209715200.0	(57.75%)
./fileup/200mb.dat	121372672	/	209715200.0	(57.88%)
./fileup/200mb.dat	121634816	/	209715200.0	(58.00%)
./fileup/200mb.dat	121896960	/	209715200.0	(58.13%)
./fileup/200mb.dat	122159104	/	209715200.0	(58.25%)
./fileup/200mb.dat	122421248	/	209715200.0	(58.38%)
./fileup/200mb.dat	122683392	/	209715200.0	(58.50%)
./fileup/200mb.dat	122945536	/	209715200.0	(58.63%)
./fileup/200mb.dat	123207680	/	209715200.0	(58.75%)
./fileup/200mb.dat	123469824	/	209715200.0	(58.88%)
./fileup/200mb.dat	123731968	/	209715200.0	(59.00%)
./fileup/200mb.dat	123994112	/	209715200.0	(59.13%)
./fileup/200mb.dat	124256256	/	209715200.0	(59.25%)
./fileup/200mb.dat	124518400	/	209715200.0	(59.38%)
./fileup/200mb.dat	124780544	/	209715200.0	(59.50%)
./fileup/200mb.dat	125042688	/	209715200.0	(59.62%)
./fileup/200mb.dat	125304832	/	209715200.0	(59.75%)
./fileup/200mb.dat	125566976	/	209715200.0	(59.88%)

./fileup/200mb.dat	125829120	/	209715200.0	(60.00%)
./fileup/200mb.dat	126091264	/	209715200.0	(60.12%)
./fileup/200mb.dat	126353408	/	209715200.0	(60.25%)
./fileup/200mb.dat	126615552	/	209715200.0	(60.38%)
./fileup/200mb.dat	126877696	/	209715200.0	(60.50%)
./fileup/200mb.dat	127139840	/	209715200.0	(60.62%)
./fileup/200mb.dat	127401984	/	209715200.0	(60.75%)
./fileup/200mb.dat	127664128	/	209715200.0	(60.88%)
./fileup/200mb.dat	127926272	/	209715200.0	(61.00%)
./fileup/200mb.dat	128188416	/	209715200.0	(61.12%)
./fileup/200mb.dat	128450560	/	209715200.0	(61.25%)
./fileup/200mb.dat	128712704	/	209715200.0	(61.38%)
./fileup/200mb.dat	128974848	/	209715200.0	(61.50%)
./fileup/200mb.dat	129236992	/	209715200.0	(61.62%)
./fileup/200mb.dat	129499136	/	209715200.0	(61.75%)
./fileup/200mb.dat	129761280	/	209715200.0	(61.88%)
./fileup/200mb.dat	130023424	/	209715200.0	(62.00%)
./fileup/200mb.dat	130285568	/	209715200.0	(62.12%)
./fileup/200mb.dat	130547712	/	209715200.0	(62.25%)
./fileup/200mb.dat	130809856	/	209715200.0	(62.38%)
./fileup/200mb.dat	131072000	/	209715200.0	(62.50%)
./fileup/200mb.dat	131334144	/	209715200.0	(62.62%)
./fileup/200mb.dat	131596288	/	209715200.0	(62.75%)
./fileup/200mb.dat	131858432	/	209715200.0	(62.88%)
./fileup/200mb.dat	132120576	/	209715200.0	(63.00%)
./fileup/200mb.dat	132382720	/	209715200.0	(63.12%)
./fileup/200mb.dat	132644864	/	209715200.0	(63.25%)
./fileup/200mb.dat	132907008	/	209715200.0	(63.38%)
./fileup/200mb.dat	133169152	/	209715200.0	(63.50%)
./fileup/200mb.dat	133431296	/	209715200.0	(63.62%)
./fileup/200mb.dat	133693440	/	209715200.0	(63.75%)
./fileup/200mb.dat	133955584	/	209715200.0	(63.88%)
./fileup/200mb.dat	134217728	/	209715200.0	(64.00%)
./fileup/200mb.dat	134479872	/	209715200.0	(64.12%)
./fileup/200mb.dat	134742016	/	209715200.0	(64.25%)
./fileup/200mb.dat	135004160	/	209715200.0	(64.38%)
./fileup/200mb.dat	135266304	/	209715200.0	(64.50%)
./fileup/200mb.dat	135528448	/	209715200.0	(64.62%)
./fileup/200mb.dat	135790592	/	209715200.0	(64.75%)
./fileup/200mb.dat	136052736	/	209715200.0	(64.88%)
./fileup/200mb.dat	136314880	/	209715200.0	(65.00%)
./fileup/200mb.dat	136577024	/	209715200.0	(65.12%)
./fileup/200mb.dat	136839168	/	209715200.0	(65.25%)
./fileup/200mb.dat	137101312	/	209715200.0	(65.38%)
./fileup/200mb.dat	137363456	/	209715200.0	(65.50%)
./fileup/200mb.dat	137625600	/	209715200.0	(65.62%)
./fileup/200mb.dat	137887744	/	209715200.0	(65.75%)
./fileup/200mb.dat	138149888	/	209715200.0	(65.88%)

./fileup/200mb.dat	138412032	/	209715200.0	(66.00%)
./fileup/200mb.dat	138674176	/	209715200.0	(66.12%)
./fileup/200mb.dat	138936320	/	209715200.0	(66.25%)
./fileup/200mb.dat	139198464	/	209715200.0	(66.38%)
./fileup/200mb.dat	139460608	/	209715200.0	(66.50%)
./fileup/200mb.dat	139722752	/	209715200.0	(66.62%)
./fileup/200mb.dat	139984896	/	209715200.0	(66.75%)
./fileup/200mb.dat	140247040	/	209715200.0	(66.88%)
./fileup/200mb.dat	140509184	/	209715200.0	(67.00%)
./fileup/200mb.dat	140771328	/	209715200.0	(67.12%)
./fileup/200mb.dat	141033472	/	209715200.0	(67.25%)
./fileup/200mb.dat	141295616	/	209715200.0	(67.38%)
./fileup/200mb.dat	141557760	/	209715200.0	(67.50%)
./fileup/200mb.dat	141819904	/	209715200.0	(67.62%)
./fileup/200mb.dat	142082048	/	209715200.0	(67.75%)
./fileup/200mb.dat	142344192	/	209715200.0	(67.88%)
./fileup/200mb.dat	142606336	/	209715200.0	(68.00%)
./fileup/200mb.dat	142868480	/	209715200.0	(68.12%)
./fileup/200mb.dat	143130624	/	209715200.0	(68.25%)
./fileup/200mb.dat	143392768	/	209715200.0	(68.38%)
./fileup/200mb.dat	143654912	/	209715200.0	(68.50%)
./fileup/200mb.dat	143917056	/	209715200.0	(68.62%)
./fileup/200mb.dat	144179200	/	209715200.0	(68.75%)
./fileup/200mb.dat	144441344	/	209715200.0	(68.88%)
./fileup/200mb.dat	144703488	/	209715200.0	(69.00%)
./fileup/200mb.dat	144965632	/	209715200.0	(69.12%)
./fileup/200mb.dat	145227776	/	209715200.0	(69.25%)
./fileup/200mb.dat	145489920	/	209715200.0	(69.38%)
./fileup/200mb.dat	145752064	/	209715200.0	(69.50%)
./fileup/200mb.dat	146014208	/	209715200.0	(69.62%)
./fileup/200mb.dat	146276352	/	209715200.0	(69.75%)
./fileup/200mb.dat	146538496	/	209715200.0	(69.88%)
./fileup/200mb.dat	146800640	/	209715200.0	(70.00%)
./fileup/200mb.dat	147062784	/	209715200.0	(70.12%)
./fileup/200mb.dat	147324928	/	209715200.0	(70.25%)
./fileup/200mb.dat	147587072	/	209715200.0	(70.38%)
./fileup/200mb.dat	147849216	/	209715200.0	(70.50%)
./fileup/200mb.dat	148111360	/	209715200.0	(70.62%)
./fileup/200mb.dat	148373504	/	209715200.0	(70.75%)
./fileup/200mb.dat	148635648	/	209715200.0	(70.88%)
./fileup/200mb.dat	148897792	/	209715200.0	(71.00%)
./fileup/200mb.dat	149159936	/	209715200.0	(71.12%)
./fileup/200mb.dat	149422080	/	209715200.0	(71.25%)
./fileup/200mb.dat	149684224	/	209715200.0	(71.38%)
./fileup/200mb.dat	149946368	/	209715200.0	(71.50%)
./fileup/200mb.dat	150208512	/	209715200.0	(71.62%)
./fileup/200mb.dat	150470656	/	209715200.0	(71.75%)
./fileup/200mb.dat	150732800	/	209715200.0	(71.88%)

./fileup/200mb.dat	150994944	/	209715200.0	(72.00%)
./fileup/200mb.dat	151257088	/	209715200.0	(72.12%)
./fileup/200mb.dat	151519232	/	209715200.0	(72.25%)
./fileup/200mb.dat	151781376	/	209715200.0	(72.38%)
./fileup/200mb.dat	152043520	/	209715200.0	(72.50%)
./fileup/200mb.dat	152305664	/	209715200.0	(72.62%)
./fileup/200mb.dat	152567808	/	209715200.0	(72.75%)
./fileup/200mb.dat	152829952	/	209715200.0	(72.88%)
./fileup/200mb.dat	153092096	/	209715200.0	(73.00%)
./fileup/200mb.dat	153354240	/	209715200.0	(73.12%)
./fileup/200mb.dat	153616384	/	209715200.0	(73.25%)
./fileup/200mb.dat	153878528	/	209715200.0	(73.38%)
./fileup/200mb.dat	154140672	/	209715200.0	(73.50%)
./fileup/200mb.dat	154402816	/	209715200.0	(73.62%)
./fileup/200mb.dat	154664960	/	209715200.0	(73.75%)
./fileup/200mb.dat	154927104	/	209715200.0	(73.88%)
./fileup/200mb.dat	155189248	/	209715200.0	(74.00%)
./fileup/200mb.dat	155451392	/	209715200.0	(74.12%)
./fileup/200mb.dat	155713536	/	209715200.0	(74.25%)
./fileup/200mb.dat	155975680	/	209715200.0	(74.38%)
./fileup/200mb.dat	156237824	/	209715200.0	(74.50%)
./fileup/200mb.dat	156499968	/	209715200.0	(74.62%)
./fileup/200mb.dat	156762112	/	209715200.0	(74.75%)
./fileup/200mb.dat	157024256	/	209715200.0	(74.88%)
./fileup/200mb.dat	157286400	/	209715200.0	(75.00%)
./fileup/200mb.dat	157548544	/	209715200.0	(75.12%)
./fileup/200mb.dat	157810688	/	209715200.0	(75.25%)
./fileup/200mb.dat	158072832	/	209715200.0	(75.38%)
./fileup/200mb.dat	158334976	/	209715200.0	(75.50%)
./fileup/200mb.dat	158597120	/	209715200.0	(75.62%)
./fileup/200mb.dat	158859264	/	209715200.0	(75.75%)
./fileup/200mb.dat	159121408	/	209715200.0	(75.88%)
./fileup/200mb.dat	159383552	/	209715200.0	(76.00%)
./fileup/200mb.dat	159645696	/	209715200.0	(76.12%)
./fileup/200mb.dat	159907840	/	209715200.0	(76.25%)
./fileup/200mb.dat	160169984	/	209715200.0	(76.38%)
./fileup/200mb.dat	160432128	/	209715200.0	(76.50%)
./fileup/200mb.dat	160694272	/	209715200.0	(76.62%)
./fileup/200mb.dat	160956416	/	209715200.0	(76.75%)
./fileup/200mb.dat	161218560	/	209715200.0	(76.88%)
./fileup/200mb.dat	161480704	/	209715200.0	(77.00%)
./fileup/200mb.dat	161742848	/	209715200.0	(77.12%)
./fileup/200mb.dat	162004992	/	209715200.0	(77.25%)
./fileup/200mb.dat	162267136	/	209715200.0	(77.38%)
./fileup/200mb.dat	162529280	/	209715200.0	(77.50%)
./fileup/200mb.dat	162791424	/	209715200.0	(77.62%)
./fileup/200mb.dat	163053568	/	209715200.0	(77.75%)
./fileup/200mb.dat	163315712	/	209715200.0	(77.88%)

./fileup/200mb.dat	163577856	/	209715200.0	(78.00%)
./fileup/200mb.dat	163840000	/	209715200.0	(78.12%)
./fileup/200mb.dat	164102144	/	209715200.0	(78.25%)
./fileup/200mb.dat	164364288	/	209715200.0	(78.38%)
./fileup/200mb.dat	164626432	/	209715200.0	(78.50%)
./fileup/200mb.dat	164888576	/	209715200.0	(78.62%)
./fileup/200mb.dat	165150720	/	209715200.0	(78.75%)
./fileup/200mb.dat	165412864	/	209715200.0	(78.88%)
./fileup/200mb.dat	165675008	/	209715200.0	(79.00%)
./fileup/200mb.dat	165937152	/	209715200.0	(79.12%)
./fileup/200mb.dat	166199296	/	209715200.0	(79.25%)
./fileup/200mb.dat	166461440	/	209715200.0	(79.38%)
./fileup/200mb.dat	166723584	/	209715200.0	(79.50%)
./fileup/200mb.dat	166985728	/	209715200.0	(79.62%)
./fileup/200mb.dat	167247872	/	209715200.0	(79.75%)
./fileup/200mb.dat	167510016	/	209715200.0	(79.88%)
./fileup/200mb.dat	167772160	/	209715200.0	(80.00%)
./fileup/200mb.dat	168034304	/	209715200.0	(80.12%)
./fileup/200mb.dat	168296448	/	209715200.0	(80.25%)
./fileup/200mb.dat	168558592	/	209715200.0	(80.38%)
./fileup/200mb.dat	168820736	/	209715200.0	(80.50%)
./fileup/200mb.dat	169082880	/	209715200.0	(80.62%)
./fileup/200mb.dat	169345024	/	209715200.0	(80.75%)
./fileup/200mb.dat	169607168	/	209715200.0	(80.88%)
./fileup/200mb.dat	169869312	/	209715200.0	(81.00%)
./fileup/200mb.dat	170131456	/	209715200.0	(81.12%)
./fileup/200mb.dat	170393600	/	209715200.0	(81.25%)
./fileup/200mb.dat	170655744	/	209715200.0	(81.38%)
./fileup/200mb.dat	170917888	/	209715200.0	(81.50%)
./fileup/200mb.dat	171180032	/	209715200.0	(81.62%)
./fileup/200mb.dat	171442176	/	209715200.0	(81.75%)
./fileup/200mb.dat	171704320	/	209715200.0	(81.88%)
./fileup/200mb.dat	171966464	/	209715200.0	(82.00%)
./fileup/200mb.dat	172228608	/	209715200.0	(82.12%)
./fileup/200mb.dat	172490752	/	209715200.0	(82.25%)
./fileup/200mb.dat	172752896	/	209715200.0	(82.38%)
./fileup/200mb.dat	173015040	/	209715200.0	(82.50%)
./fileup/200mb.dat	173277184	/	209715200.0	(82.62%)
./fileup/200mb.dat	173539328	/	209715200.0	(82.75%)
./fileup/200mb.dat	173801472	/	209715200.0	(82.88%)
./fileup/200mb.dat	174063616	/	209715200.0	(83.00%)
./fileup/200mb.dat	174325760	/	209715200.0	(83.12%)
./fileup/200mb.dat	174587904	/	209715200.0	(83.25%)
./fileup/200mb.dat	174850048	/	209715200.0	(83.38%)
./fileup/200mb.dat	175112192	/	209715200.0	(83.50%)
./fileup/200mb.dat	175374336	/	209715200.0	(83.62%)
./fileup/200mb.dat	175636480	/	209715200.0	(83.75%)
./fileup/200mb.dat	175898624	/	209715200.0	(83.88%)

./fileup/200mb.dat	176160768	/	209715200.0	(84.00%)
./fileup/200mb.dat	176422912	/	209715200.0	(84.12%)
./fileup/200mb.dat	176685056	/	209715200.0	(84.25%)
./fileup/200mb.dat	176947200	/	209715200.0	(84.38%)
./fileup/200mb.dat	177209344	/	209715200.0	(84.50%)
./fileup/200mb.dat	177471488	/	209715200.0	(84.62%)
./fileup/200mb.dat	177733632	/	209715200.0	(84.75%)
./fileup/200mb.dat	177995776	/	209715200.0	(84.88%)
./fileup/200mb.dat	178257920	/	209715200.0	(85.00%)
./fileup/200mb.dat	178520064	/	209715200.0	(85.12%)
./fileup/200mb.dat	178782208	/	209715200.0	(85.25%)
./fileup/200mb.dat	179044352	/	209715200.0	(85.38%)
./fileup/200mb.dat	179306496	/	209715200.0	(85.50%)
./fileup/200mb.dat	179568640	/	209715200.0	(85.62%)
./fileup/200mb.dat	179830784	/	209715200.0	(85.75%)
./fileup/200mb.dat	180092928	/	209715200.0	(85.88%)
./fileup/200mb.dat	180355072	/	209715200.0	(86.00%)
./fileup/200mb.dat	180617216	/	209715200.0	(86.12%)
./fileup/200mb.dat	180879360	/	209715200.0	(86.25%)
./fileup/200mb.dat	181141504	/	209715200.0	(86.38%)
./fileup/200mb.dat	181403648	/	209715200.0	(86.50%)
./fileup/200mb.dat	181665792	/	209715200.0	(86.62%)
./fileup/200mb.dat	181927936	/	209715200.0	(86.75%)
./fileup/200mb.dat	182190080	/	209715200.0	(86.88%)
./fileup/200mb.dat	182452224	/	209715200.0	(87.00%)
./fileup/200mb.dat	182714368	/	209715200.0	(87.12%)
./fileup/200mb.dat	182976512	/	209715200.0	(87.25%)
./fileup/200mb.dat	183238656	/	209715200.0	(87.38%)
./fileup/200mb.dat	183500800	/	209715200.0	(87.50%)
./fileup/200mb.dat	183762944	/	209715200.0	(87.62%)
./fileup/200mb.dat	184025088	/	209715200.0	(87.75%)
./fileup/200mb.dat	184287232	/	209715200.0	(87.88%)
./fileup/200mb.dat	184549376	/	209715200.0	(88.00%)
./fileup/200mb.dat	184811520	/	209715200.0	(88.12%)
./fileup/200mb.dat	185073664	/	209715200.0	(88.25%)
./fileup/200mb.dat	185335808	/	209715200.0	(88.38%)
./fileup/200mb.dat	185597952	/	209715200.0	(88.50%)
./fileup/200mb.dat	185860096	/	209715200.0	(88.62%)
./fileup/200mb.dat	186122240	/	209715200.0	(88.75%)
./fileup/200mb.dat	186384384	/	209715200.0	(88.88%)
./fileup/200mb.dat	186646528	/	209715200.0	(89.00%)
./fileup/200mb.dat	186908672	/	209715200.0	(89.12%)
./fileup/200mb.dat	187170816	/	209715200.0	(89.25%)
./fileup/200mb.dat	187432960	/	209715200.0	(89.38%)
./fileup/200mb.dat	187695104	/	209715200.0	(89.50%)
./fileup/200mb.dat	187957248	/	209715200.0	(89.62%)
./fileup/200mb.dat	188219392	/	209715200.0	(89.75%)
./fileup/200mb.dat	188481536	/	209715200.0	(89.88%)


./fileup/200mb.dat	188743680	/	209715200.0	(90.00%)
./fileup/200mb.dat	189005824	/	209715200.0	(90.12%)
./fileup/200mb.dat	189267968	/	209715200.0	(90.25%)
./fileup/200mb.dat	189530112	/	209715200.0	(90.38%)
./fileup/200mb.dat	189792256	/	209715200.0	(90.50%)
./fileup/200mb.dat	190054400	/	209715200.0	(90.62%)
./fileup/200mb.dat	190316544	/	209715200.0	(90.75%)
./fileup/200mb.dat	190578688	/	209715200.0	(90.88%)
./fileup/200mb.dat	190840832	/	209715200.0	(91.00%)
./fileup/200mb.dat	191102976	/	209715200.0	(91.12%)
./fileup/200mb.dat	191365120	/	209715200.0	(91.25%)
./fileup/200mb.dat	191627264	/	209715200.0	(91.38%)
./fileup/200mb.dat	191889408	/	209715200.0	(91.50%)
./fileup/200mb.dat	192151552	/	209715200.0	(91.62%)
./fileup/200mb.dat	192413696	/	209715200.0	(91.75%)
./fileup/200mb.dat	192675840	/	209715200.0	(91.88%)
./fileup/200mb.dat	192937984	/	209715200.0	(92.00%)
./fileup/200mb.dat	193200128	/	209715200.0	(92.12%)
./fileup/200mb.dat	193462272	/	209715200.0	(92.25%)
./fileup/200mb.dat	193724416	/	209715200.0	(92.38%)
./fileup/200mb.dat	193986560	/	209715200.0	(92.50%)
./fileup/200mb.dat	194248704	/	209715200.0	(92.62%)
./fileup/200mb.dat	194510848	/	209715200.0	(92.75%)
./fileup/200mb.dat	194772992	/	209715200.0	(92.88%)
./fileup/200mb.dat	195035136	/	209715200.0	(93.00%)
./fileup/200mb.dat	195297280	/	209715200.0	(93.12%)
./fileup/200mb.dat	195559424	/	209715200.0	(93.25%)
./fileup/200mb.dat	195821568	/	209715200.0	(93.38%)
./fileup/200mb.dat	196083712	/	209715200.0	(93.50%)
./fileup/200mb.dat	196345856	/	209715200.0	(93.62%)
./fileup/200mb.dat	196608000	/	209715200.0	(93.75%)
./fileup/200mb.dat	196870144	/	209715200.0	(93.88%)
./fileup/200mb.dat	197132288	/	209715200.0	(94.00%)
./fileup/200mb.dat	197394432	/	209715200.0	(94.12%)
./fileup/200mb.dat	197656576	/	209715200.0	(94.25%)
./fileup/200mb.dat	197918720	/	209715200.0	(94.38%)
./fileup/200mb.dat	198180864	/	209715200.0	(94.50%)
./fileup/200mb.dat	198443008	/	209715200.0	(94.62%)
./fileup/200mb.dat	198705152	/	209715200.0	(94.75%)
./fileup/200mb.dat	198967296	/	209715200.0	(94.88%)
./fileup/200mb.dat	199229440	/	209715200.0	(95.00%)
./fileup/200mb.dat	199491584	/	209715200.0	(95.12%)
./fileup/200mb.dat	199753728	/	209715200.0	(95.25%)
./fileup/200mb.dat	200015872	/	209715200.0	(95.38%)
./fileup/200mb.dat	200278016	/	209715200.0	(95.50%)
./fileup/200mb.dat	200540160	/	209715200.0	(95.62%)
./fileup/200mb.dat	200802304	/	209715200.0	(95.75%)
./fileup/200mb.dat	201064448	/	209715200.0	(95.88%)

./fileup/200mb.dat	201326592	/	209715200.0	(96.00%)
./fileup/200mb.dat	201588736	/	209715200.0	(96.12%)
./fileup/200mb.dat	201850880	/	209715200.0	(96.25%)
./fileup/200mb.dat	202113024	/	209715200.0	(96.38%)
./fileup/200mb.dat	202375168	/	209715200.0	(96.50%)
./fileup/200mb.dat	202637312	/	209715200.0	(96.62%)
./fileup/200mb.dat	202899456	/	209715200.0	(96.75%)
./fileup/200mb.dat	203161600	/	209715200.0	(96.88%)
./fileup/200mb.dat	203423744	/	209715200.0	(97.00%)
./fileup/200mb.dat	203685888	/	209715200.0	(97.12%)
./fileup/200mb.dat	203948032	/	209715200.0	(97.25%)
./fileup/200mb.dat	204210176	/	209715200.0	(97.38%)
./fileup/200mb.dat	204472320	/	209715200.0	(97.50%)
./fileup/200mb.dat	204734464	/	209715200.0	(97.62%)
./fileup/200mb.dat	204996608	/	209715200.0	(97.75%)
./fileup/200mb.dat	205258752	/	209715200.0	(97.88%)
./fileup/200mb.dat	205520896	/	209715200.0	(98.00%)
./fileup/200mb.dat	205783040	/	209715200.0	(98.12%)
./fileup/200mb.dat	206045184	/	209715200.0	(98.25%)
./fileup/200mb.dat	206307328	/	209715200.0	(98.38%)
./fileup/200mb.dat	206569472	/	209715200.0	(98.50%)
./fileup/200mb.dat	206831616	/	209715200.0	(98.62%)
./fileup/200mb.dat	207093760	/	209715200.0	(98.75%)
./fileup/200mb.dat	207355904	/	209715200.0	(98.88%)
./fileup/200mb.dat	207618048	/	209715200.0	(99.00%)
./fileup/200mb.dat	207880192	/	209715200.0	(99.12%)
./fileup/200mb.dat	208142336	/	209715200.0	(99.25%)
./fileup/200mb.dat	208404480	/	209715200.0	(99.38%)
./fileup/200mb.dat	208666624	/	209715200.0	(99.50%)
./fileup/200mb.dat	208928768	/	209715200.0	(99.62%)
./fileup/200mb.dat	209190912	/	209715200.0	(99.75%)
./fileup/200mb.dat	209453056	/	209715200.0	(99.88%)
./fileup/200mb.dat	209715200	/	209715200.0	(100.00%)

CMC 上では、マルチパートアップロード中のファイルが存在する場合には、下図のように「マルチパートアップロード実行中」が表示されます。

マルチパートアップロード実行中

マルチパートアップロード開始日	アップロードオブジェクト名	アクション
Dec-14-2020 08:05 AM +0900	200mb_mpu.dat	中止

アップロード時に AES256 のサーバーサイド暗号化を有効にしたので、アップロードしたファイル名の先頭に  が表示されます。

はじめに

バケット

オブジェクト

バケット名


boto3

📁 ファイルをアップロード

+ フォルダーを作成

🔍 プレフィックスで検索

region1 : boto3

<input type="checkbox"/>	名前	サイズ	最終更新	
<input type="checkbox"/>	 200mb_mpu.dat	200.0 MB	Dec-14-2020 08:27 AM +0900	<div>🔗 プロパティ</div> <div>🗑 削除</div>

リストア

削除

Cloudianオブジェクトストレージ/アップロードの進捗状況を表示してみる

マルチスレッドによるファイルのアップロード

指定したディレクトリ(このサンプルでは「./log」)にある全てのファイルを、Cloudian上のバケット「logs」の中の実行日のフォルダ(YYYY-MM-DD)にアップロードする Python プログラムです。

アップロードするファイル毎にスレッドを作成して、マルチスレッドで Cloudian にファイルをアップロードします。アップロード時にファイルに対して、ACL に「private」を設定し、メタデータとしてアップロードした日付を付与し、AES256 のサーバーサイド暗号化を有効にしています。

```

import os
import glob
import threading
import sys
from datetime import datetime

import boto3, botocore
from boto3.s3.transfer import S3Transfer
from boto3.s3.transfer import TransferConfig

##### マルチスレッド関数 #####
def multithreads_upload(upfile):

    print("Thread %s is uploading file: %s" % (threading.cu

    transfer.upload_file(
        upfile,
        bucket,
        today + '/' + os.path.basename(upfile),

        extra_args={
            'ACL': 'private',
            'Metadata': {'Stored': today},
            'ServerSideEncryption': 'AES256'
        }
    )

    print("Thread %s done uploading file: %s" % (threading.

##### メイン処理 #####
# 変数セット
dir = './fileup/'

bucket = 'logs'
region = 'region1'
today = datetime.now().strftime('%Y-%m-%d')

# boto3オブジェクト作成
client = boto3.client(
    's3',
    endpoint_url='http://s3-region1.admin-tech.tokyo',

    # 認証情報の直接記述は推奨されない（テスト目的）
    aws_access_key_id='アクセスキーを記述',

```

```

        aws_secret_access_key='シークレットキーを記述'
    )

    # 転送オブジェクト作成
    transfer = S3Transfer(client)

    try:
        # バケットの存在確認
        client.head_bucket(Bucket = bucket)

    except botocore.exceptions.ClientError:
        # バケットが存在しなければ、新規作成
        client.create_bucket(
            ACL='private',
            Bucket=bucket,
            CreateBucketConfiguration={
                'LocationConstraint': 'region1'
            }
        )

    # マルチスレッドによるファイルのアップロード
    upfiles = [dir + file for file in os.listdir(dir) if os.path.isfile(os.path.join(dir, file))]
    threads = []

    for upfile in upfiles:
        t = threading.Thread(target=multithreads_upload, args=(client, bucket, upfile))
        t.start()
        threads.append(t)

    for t in threads:
        t.join()

```

プログラム実行の出力ログから、アップロードするファイルサイズにより各スレッドの開始と終了に差があることが分かります。このサンプルでは各スレッドの終了を `threading.join()` を使って、全てのファイルのアップロードが終了するのを待機します。

下図はこのサンプルプログラムによりアップロードされたファイルを、CMC から確認した画面です

はじめに

バケット

オブジェクト

バケット名

logs

📁 ファイルをアップロード

+ フォルダーを作成

🔍 プレフィックスで検索

region1 : logs => 2020-12-14

<input type="checkbox"/>	名前	サイズ	最終更新		
<input type="checkbox"/>	🔒 .DS_Store	6.0 KB	Dec-14-2020 09:07 AM +0900	👤 プロパティ	🗑 削除
<input type="checkbox"/>	🔒 10mb.dat	10.0 MB	Dec-14-2020 09:07 AM +0900	👤 プロパティ	🗑 削除

リストア

削除

マルチバートアップロード実行中