Cloud Storage Python SDK develop guide

目录

1. 准备

1.1 环境

使用盛大云存储的 Python SDK 进行开发需要具备 python 2.6 以上版本

1.2 使用到的 Python module

os, sys, httplib, urllib, socket, logging, time, base64, hmac, sha, md5, xml.sax, string
由于需要将存储服务返回的 RFC3339 日期格式转化成 datetime 数据类型，因此需要使用 rfc3339 这个 module
Linux 安装:

pip install -e git://github.com/tonyg/python-rfc3339.git#egg=rfc3339

Windows 安装:

1) 下载 http://github.com/tonyg/python-rfc3339/tarball/master
2) 解压缩后，执行 python setup.py install 安装

1.3 下载 sndacspylib

将 sndacspylib 下载到需要安装的目录，解压缩

1.4 安装 sndacspylib

1) 修改目录 sndacspylib/config/下，cs.properties 文件中的 AccessKey, SecretKey 的值为盛大为云存储用户提供的相应 key 值；CheckHash 可接受的配置为 True 和 False，True 表示上传和下载数据时会本地校验数据的 MD5 值，False 表示不作本地校验；SecureComm 可接受的配置为 True 和 False，True 表示使用 HTTPS 连接，端口使用 443，False 表示使用 HTTP 连接，端口使用 80
2) 在 sndacspylib 目录下运行命令，python setup.py install

2. 如何编写代码

2.1 创建连接

```
conn=cs_util.CS.SNDAAuthConnection(Config.CSProperties['AccessKey
'], Config.CSProperties['SecretKey'],
(Config.CSProperties['SecureComm'] == False))
```

2.2 创建云存储服务对象

```
cloud_storage=cs_util.SNDA_CS( ConnectionObject = conn )
```

2.3 获取 Bucket 列表

```
cloud_storage.ListOfBuckets
```

其中的每个对象都有对应的 name 和 creation_date 字段值可以通过以下方法取得

```
cloud_storage.ListOfBuckets[name].creation_date
cloud_storage.ListOfBuckets[name].location
```

2.4 添加新 Bucket

第一个参数用以指定 bucket 的名字，第二个参数为 bucket 的位置信息，目前可选的 idc 位置有 huabei-1 和 huadong-1，默认为 huabei-1

```
cloud_storage.add_bucket("your-universe-bucket-name", location)
```

2.5 删除 Bucket

```
cloud_storage.delete_bucket("your-universe-bucket-name")
```

2.6 创建云存储数据对象

```
temp_object=cs_util.SNDA_Object(cloud_storage.CONN,
"your-universe-bucket-name", "temp_key")
```

2.7 上传数据

第一个参数指定需要上传的文件的路径，第二个参数指定需要指定的数据的 meta 信息，默认可以不传入此参数

```
temp_object.put_object_from_file("C:\\folder\\your_test_file",hea
ders)
```

headers 为用户需要自定义的 HTTP Header 信息，例如，用户可以自定义 Content-Type 为值 XXX

2.8 获取数据 Meta 信息

```
response = temp_object.get_object_info()
print response.http_response.msg
```

2.9 下载数据

```
temp_object.get_object_to_file("C:\\folder\\your_test_file.get")
```

2.10 删除数据

```
temp_object.delete_object()
```

2.11 生成 Bucket Policy 对象以及 xml 格式字符串

```
from sndacspylib.snda_cs_model import *
effect = Effects.Allow
actions = Actions.AllActions
resources = "*"
conditions = \
    {ConditionTypes.Bool:{AvailableKeys.SecureTransport:True},\
    ConditionTypes.IpAddress:{AvailableKeys.SourceIp:"102.168.0.0
   /24"}}
statement = PolicyStatement(Sid=None,Effect=effect,Principal=None,
Action=actions,Resource=resources,Condition=conditions)
Statement.sid_regenerate()
policy = \
   BucketPolicy(Id="your_id",Version=None,Statement=[statement])
import json
policy_xml = json.dumps(policy.toDict())
```

2.12 设置 Bucket Policy

```
bucket = cs_util.SNDA_Bucket(cloud_storage.CONN, \
   "your-bucket-name")
bucket.set_policy(policy_xml)
```

2.13 获取 Bucket Policy

```
policy_xml = bucket.get_policy()
```

2.14 删除 Bucket Policy

```
bucket.delete_policy()
```

2.15 初始化 Multiupload

```
temp_object = cs_util.SNDA_Object(cloud_storage.CONN,
"your-universe-bucket-name", "temp_key")
temp_object._initiate_multipart_upload_()
```

2.16 取消 Multiupload

```
temp_object._abort_multipart_upload_(temp_object.init_result.uplo
ad_id)
```

2.17 上传 Multiupload parts

```
part1 = temp_object._upload_part_from_file_(upload_id, "1",
"folder/file")
part2 = temp_object._upload_part_from_data_(upload_id, "2", "This is
part 2.")
```

2.18 完成 Multiupload

```
complete_content = CompleteMultipartUpload([part1, part2])
from xml.dom.minidom import Document
import sndacspylib.snda_cs.cs_genutilities as Util
document = Document()
Util.object_convert_to_xml(document, complete_content)
temp_object._complete_multipart_upload_(upload_id,
document.toxml())
```

2.19 获取 Multiupload 对象列表

```
temp_object._list_parts_(upload_id)
```

对象列表存放在 temp_object.list_part_result 中

3. 存储服务对象

3.1 SNDA_CS

该类型封装了对盛大云存储 Bucket 数据类型的相关操作，其中包含：

1) 获取 bucket 列表: get_list_of_buckets()
2) 添加 bucket: add_bucket(bucket_name)
3) 删除 bucket: delete_bucket(bucket_name)

3.2 SNDA_Bucket

该类型封装了对盛大云存储 Bucket 下数据进行同步以及 Bucket Policy 的相关操作，其中包含：

1) 获取 bucket 下文件列表: get_list_of_keys_in_bucket()
2) 同步本地目录到 bucket: upload_dir(dir_path)
3) 同步 bucket 到本地目录: download_dir(dir_path)
4) 设置 policy: set_policy(policy)
5) 获取 policy: get_policy()
6) 删除 policy: delete_policy()

3.3 SNDA_Object

该类型封装了对盛大云存储 Object 数据类型的相关操作，其中包含：

1) 上传文件: put_object_from_file(file_name, headers)
2) 下载文件: get_object_to_file(file_name)
3) 获取文件信息: get_object_info()
4) 删除文件: delete_object()

4. 代码示例

Python IDLE 中执行下述代码：

```
>>>import sndacspylib.snda_cs.cs_util
>>>from sndacspylib.snda_cs_config import Config
>>>conn=sndacspylib.snda_cs.cs_util.CS.SNDAAuthConnection(Config.
CSProperties['AccessKey'], Config.CSProperties['SecretKey'],
(Config.CSProperties['SecureComm']==False))
>>>cloud_storage=sndacspylib.snda_cs.cs_util.SNDA_CS(ConnectionOb
ject = conn)
>>>cloud_storage.get_list_of_buckets()
>>>for entry in cloud_storage.ListOfBuckets:
…   print '%s\t\t\t%s' % (entry.name, entry.creation_date)
…
>>>
```

完整 python 代码示例：

```
from sndacspylib.snda_cs_exception import *
from sndacspylib.snda_cs_config import Config
import sndacspylib.snda_cs_genutilities as Util
import sndacspylib.snda_cs.cs_util as CSUtil

import uuid

UNIVERSE_BUCKET_NAME = uuid.uuid4().hex

def init():
    conn =
CSUtil.CS.SNDAAuthConnection(Config.CSProperties['AccessKey'],
                                    Config.CSProperties['SecretKey'],
( Config.CSProperties['SecureComm'] == False))
    cloud_storage = CSUtil.SNDA_CS( ConnectionObject  = conn )
```

```
    return cloud_storage

def pre_create_bucket(cloud_storage):

    cloud_storage.add_bucket(UNIVERSE_BUCKET_NAME)

def final_delete_bucket(cloud_storage):

    cloud_storage.delete_bucket(UNIVERSE_BUCKET_NAME)

def test_bucket(cloud_storage):

    # list buckets
    cloud_storage.get_list_of_buckets();
    for entry in cloud_storage.ListOfBuckets:
        print '%s\t\t\t%s' % (entry.name, entry.creation_date)

    # add bucket
    cloud_storage.add_bucket("your-universe-bucket-name")

    # delete bucket
    cloud_storage.delete_bucket("your-universe-bucket-name")

def test_object(cloud_storage):

    pre_create_bucket(cloud_storage)

    temp_object =
CSUtil.SNDA_Object(cloud_storage.CONN ,UNIVERSE_BUCKET_NAME,
"temp_key")
    # upload object
    temp_object.put_object_from_file("C:\Documents and
Settings\user\workspace\python\README.md")

    # get object meta
    response = temp_object.get_object_info()
    print response.http_response.msg

    # download object
    temp_object.get_object_to_file("C:\Documents and
Settings\user\workspace\python\README.md.get")
```

```
    # delete object
    temp_object.delete_object()

    final_delete_bucket(cloud_storage)

if __name__ == '__main__':

    cloud_storage = init()
    test_bucket(cloud_storage)
    test_object(cloud_storage)
```