阿里云开放存储服务(OSS)

OSS Python SDK 文档

Release 0.3.1

2013.8.1

目录

1. 前言

这篇文档主要介绍：

1. 如何安装和验证 OSS Python SDK。
2. 如何使用 Python 来进行 OSS API 调用。
3. 如何使用 osscmd 来进行 OSS 操作。

这篇文档假设你已经熟悉 Python，了解并注册了阿里云的 OSS 服务，并获得了系统分配的 Access Key ID 和 Access Key Secret。文中的 ID 指的是 Access Key ID，KEY 指的是 Access Key Secret。如果你还没有开通或者还不了解 OSS，请登录 http://oss.aliyun.com 获取更多的帮助。

2. 快速安装

下面介绍一个完整的过程，示范如何使用 Python 在 Windows 平台和 Linux 平台上进行 OSS bucket 和 object 的基本操作。

2.1. 环境要求

Python SDK 需要一个可以运行 Python 的环境。Python 的版本要求要在 2.5 和 2.7 之间。SDK 适用于 Windows 平台和 Linux 平台，但由于 Python3.0 并不完全兼容 2.x 的版本，所以 SDK 暂时不支持 3.0 及以上的版本。

安装好 Python 后：

- Linux shell 环境下输入 python 并回车，来查看 Python 的版本。如下所示：

```
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 在 cmd 环境下输入 python 并回车，来查看 Python 的版本。如下所示：

```
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.
>>>
```

以上说明 python 安装成功。

- 异常情况，如 Windows 在 cmd 环境下输入 python 并回车后，提示“不是内部或外部命令”，请检查&配置“我的电脑”-“属性”-“高级”-“环境变量”-“Path”，增加 python 的安装路径。

  如图:

如果没有安装 Python，可以从 http://www.python.org 获取 Python 的安装包。

网站有详细的安装说明来指导用户如何安装和使用 Python。

2.2. 安装和验证 SDK

下面介绍在 Windows 平台和 Linux 平台上如何安装 Python SDK，以及如何验证 SDK 是否安装成功。

2.2.1. 安装 SDK

1. 打开浏览器，输入 oss.aliyun.com。
2. 在“产品帮助”-“开发者资源”这个位置找到 Python SDK 链接。
3. 点击链接并选择保存 SDK 安装包。
4. 下载后可以得到类似 OSS_Python_API_xxxxxxxx.tar.gz 的安装包。
5. 进入安装包所在的目录，将 tar.gz 的包进行解压缩。Linux 可以使用 tar -zxvf OSS_Python_API_xxxxxxxx.tar.gz 命令解压缩。Windows 可以使用 7-Zip 等解压缩工具。
6. 解压缩后得到的文件及目录如下

```
README
OSS_Python_SDK.pdf
setup.py
osscmd
oss/
    oss_api.py
    oss_util.py
    oss_xml_handler.py
    oss_sample.py
```

7. 两个平台的 SDK 安装
   - 假设 SDK 已经解压缩到 Windows 平台的 D 盘，安装的日志如下：

```
D:\>cd    OSS_Python_API_20130712
D:\ OSS_Python_API_20130712 >python setup.py install
running install
running build
running build_py
creating build
creating build\lib
creating build\lib\oss
copying oss\oss_api.py -> build\lib\oss
copying oss\oss_sample.py -> build\lib\oss
copying oss\oss_util.py -> build\lib\oss
copying oss\oss_xml_handler.py -> build\lib\oss
copying oss\pkg_info.py -> build\lib\oss
copying oss\__init__.py -> build\lib\oss
running install_lib
```

creating C:\Python27\Lib\site-packages\oss
copying build\lib\oss\oss_api.py -> C:\Python27\Lib\site-packages\oss
copying build\lib\oss\oss_sample.py -> C:\Python27\Lib\site-packages\oss
copying build\lib\oss\oss_util.py -> C:\Python27\Lib\site-packages\oss
copying build\lib\oss\oss_xml_handler.py -> C:\Python27\Lib\site-packages\oss
copying build\lib\oss\pkg_info.py -> C:\Python27\Lib\site-packages\oss
copying build\lib\oss__init__.py -> C:\Python27\Lib\site-packages\oss
byte-compiling C:\Python27\Lib\site-packages\oss\oss_api.py to oss_api.pyc
byte-compiling C:\Python27\Lib\site-packages\oss\oss_sample.py to oss_sample.pyc

byte-compiling C:\Python27\Lib\site-packages\oss\oss_util.py to oss_util.pyc
byte-compiling C:\Python27\Lib\site-packages\oss\oss_xml_handler.py to oss_xml_h
andler.pyc
byte-compiling C:\Python27\Lib\site-packages\oss\pkg_info.py to pkg_info.pyc
byte-compiling C:\Python27\Lib\site-packages\oss__init__.py to __init__.pyc
running install_egg_info
Writing C:\Python27\Lib\site-packages\oss-0.1.3-py2.7.egg-info

- 假设 SDK 已经解压缩到 Linux 平台的 oss 目录，安装的日志如下：

[oss@oss python]$ **sudo python setup.py install**
Password:
running install
running bdist_egg
running egg_info
writing oss.egg-info/PKG-INFO
writing top-level names to oss.egg-info/top_level.txt
writing dependency_links to oss.egg-info/dependency_links.txt
writing manifest file 'oss.egg-info/SOURCES.txt'
installing library code to build/bdist.linux-x86_64/egg
running install_lib
running build_py
creating build/bdist.linux-x86_64/egg
creating build/bdist.linux-x86_64/egg/oss
copying build/lib/oss/oss_util.py -> build/bdist.linux-x86_64/egg/oss
copying build/lib/oss/__init__.py -> build/bdist.linux-x86_64/egg/oss
copying build/lib/oss/pkg_info.py -> build/bdist.linux-x86_64/egg/oss
copying build/lib/oss/oss_xml_handler.py -> build/bdist.linux-x86_64/egg/oss
copying build/lib/oss/oss_sample.py -> build/bdist.linux-x86_64/egg/oss
copying build/lib/oss/oss_api.py -> build/bdist.linux-x86_64/egg/oss
byte-compiling build/bdist.linux-x86_64/egg/oss/oss_util.py to oss_util.pyc
byte-compiling build/bdist.linux-x86_64/egg/oss/__init__.py to __init__.pyc
byte-compiling build/bdist.linux-x86_64/egg/oss/pkg_info.py to pkg_info.pyc
byte-compiling build/bdist.linux-x86_64/egg/oss/oss_xml_handler.py to oss_xml_handler.pyc
byte-compiling build/bdist.linux-x86_64/egg/oss/oss_sample.py to oss_sample.pyc
byte-compiling build/bdist.linux-x86_64/egg/oss/oss_api.py to oss_api.pyc
creating build/bdist.linux-x86_64/egg/EGG-INFO

```
copying oss.egg-info/PKG-INFO -> build/bdist.linux-x86_64/egg/EGG-INFO
copying oss.egg-info/SOURCES.txt -> build/bdist.linux-x86_64/egg/EGG-INFO
copying oss.egg-info/dependency_links.txt -> build/bdist.linux-x86_64/egg/EGG-INFO
copying oss.egg-info/top_level.txt -> build/bdist.linux-x86_64/egg/EGG-INFO
zip_safe flag not set; analyzing archive contents...
oss.oss_sample: module references __file__
creating 'dist/oss-0.1.3-py2.5.egg' and adding 'build/bdist.linux-x86_64/egg' to it
removing 'build/bdist.linux-x86_64/egg' (and everything under it)
Processing oss-0.1.3-py2.5.egg
removing '/usr/ali/lib/python2.5/site-packages/oss-0.1.3-py2.5.egg' (and everything under it)
creating /usr/ali/lib/python2.5/site-packages/oss-0.1.3-py2.5.egg
Extracting oss-0.1.3-py2.5.egg to /usr/ali/lib/python2.5/site-packages
oss 0.1.3 is already the active version in easy-install.pth
Installed /usr/ali/lib/python2.5/site-packages/oss-0.1.3-py2.5.egg
Processing dependencies for oss==0.1.3
Finished processing dependencies for oss==0.1.3
```

8. SDK 的卸载

不同的机器和平台，SDK 安装的目录不一样。卸载需要进入到 SDK 在安装的目录，将相关的安装目录删除即可。例如上一步所示的 /usr/ali/lib/python2.5/site-packages/oss-0.1.3-py2.5.egg 目录。

也可以通过下面的命令，从当前版本 Python 的 site-packages 目录中找到 oss 相关的安装目录删除来卸载。

```
>>> import sys
>>> print sys.path
['','/usr/ali/lib/python2.5/site-packages/httplib2-0.7.7-py2.5.egg',
'/usr/ali/lib/python2.5/site-packages/oss-0.1.3-py2.5.egg','/usr/ali/lib/python25.zip','/usr/ali/lib/python2.5',
'/usr/ali/lib/python2.5/plat-linux2','/usr/ali/lib/python2.5/lib-tk','/usr/ali/lib/python2.5/lib-dynload',
'/usr/ali/lib/python2.5/site-packages']
```

2.2.2. 验证 SDK

在 Linux shell 或者 Windows cmd 输入 python 并回车，进入 python 的环境后输入 import oss，没有安装成功则会出现如下错误信息：

```
>>> import oss
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named oss
```

安装成功后不会报异常，如下：

```
>>> import oss
>>>
```

查看 SDK 的版本号，无异常且能正常输出信息则安装成功。

```
>>> from oss.oss_api import *
>>> print OssAPI.Version
0.3.1
>>>
```

3. 使用说明

oss_api.py 中提供了一些操作 bucket 和 object 的方法。用户可以调用这些方法来操作 OSS，也可以直接使用 osscmd 来操作 OSS。下面将分别介绍两种使用方法。

3.1. 使用 oss_api.py

oss_api.py 中提供的接口大多直接返回 HTTP Response。关于 HTTP Response 的定义请参见 http://docs.python.org/2/library/httplib.html 中的说明。

下面示例中用 res 来表示一个 HTTP Response 的实例。res.status 表示 Python SDK 向 OSS 发送 HTTP 请求后，OSS HTTP Server 的响应状态码。具体各种状态码的定义见 OSS API 文档。res.getheaders()表示 OSS HTTP Server 响应的 Headers。res.read()表示 HTTP 响应的 Body，有些情况下 Body 是无内容的。

1. 创建 bucket

函数定义：

def create_bucket(self, bucket, acl='', headers=None):

函数说明：

有关 bucket 的概念请参见 http://oss.aliyun.com 网站的 OSS API 文档。创建的时候用户可以指定 bucket 的访问控制权限(Access Control List，acl 都表示访问控制权限)。目前 acl 只支持 private, public-read, public-read-write 三种，用户也可以不指定。不指定则默认 acl 是 private。

参数说明：

bucket：

类型：string

acl：

类型：string

目前为 private，public-read， public-read-write 中的一种。

headers：

类型：dict

可以用来设置 HTTP 请求中的 Headers，默认为空。

返回值说明：

HTTP Response

注意事项：

一般的情况下只需要指定 bucket 和 acl，headers 可以不用填。

使用示例：

```
>>> from oss.oss_api import *
```

>>> oss = OssAPI("oss.aliyuncs.com", "ID", "KEY") #ID 和 KEY 需要填成用户的，可以在阿里云网站-“用户中心”-“我的帐户”-“安全认证”获取。

```
>>> res = oss.create_bucket("oss-test-sample","private")
>>> print "%s\n%s" % (res.status, res.read())
200
```

200 的状态码表示 bucket 创建成功。Body 为空，所以没有内容显示。

```
>>> res = oss.create_bucket("oss")
>>> print "%s\n%s" % (res.status, res.read())
409
<?xml version="1.0" encoding="UTF-8"?>
<Error>
    <BucketName>oss</BucketName>
    <Code>BucketAlreadyExists</Code>
    <Message>The requested bucket name is not available. The bucket namespace is shared by
    all users of the system. Please select a different name and try again.</Message>
    <RequestId>51FE0D97DFE05B372544222E</RequestId>
    <HostId>oss.oss.aliyuncs.com</HostId>
</Error>
```

409 的状态码表示创建的 bucket 已经存在了，没有创建成功。Body 中说明了创建不成功的原因。所以这个时候需要换一个 bucket 的名字。只有显示 200 状态码的时候才表示创建成功。

```
>>> res = oss.create_bucket("oss-test-sample", "no-acl")
>>> print "%s\n%s" % (res.status, res.read())
400
<?xml version="1.0" encoding="UTF-8"?>
  <Error>
    <Code>InvalidArgument</Code>
    <Message>no such bucket access control exists</Message>
    <ArgumentValue>no-acl</ArgumentValue>
    <ArgumentName>x-oss-acl</ArgumentName>
    <RequestId>51FE0DB67260AF325F440276</RequestId>
    <HostId>oss-test-sample.oss.aliyuncs.com</HostId>
</Error>
```

400 的状态码表示创建 bucket 传入的 acl 是非法的参数，在 Body 中说明了创建不成功的原因，是因为没有 no-acl 这个 acl 选项。

```
>>> res = oss.create_bucket("oss-test-sample", "private")
>>> print res.status, res.getheaders(), res.read()
200
```

200 的状态码表示带 acl 创建 bucket 成功。Body 没有内容，所以没有显示。

函数定义：

def put_bucket(self, bucket, acl='', headers=None):

函数说明：

等同 create_bucket 函数。

2. 删除 bucket

函数定义：

def delete_bucket(self, bucket, headers=None):

函数说明：

该函数将删除用户创建的 bucket。

参数说明：

bucket：

类型：string

headers：

类型：dict

可以用来设置 HTTP 请求中的 Headers，默认为空。

返回值说明：

HTTP Response

注意事项：

只能删除内容为空的 bucket。如果 bucket 存在 object 或者 multipart 相关的内容例如 Upload Id， 尚未 Complete 的 Parts， 则此 bucket 为内容非空，是不能被删除的。相关的 multipart 概念参见 OSS API 文档。

使用示例：

```
>>> from oss.oss_api import *
>>> oss = OssAPI("oss.aliyuncs.com", "ID", "KEY")   #ID 和 KEY 需要填成用户的，可以在阿里云网站-“用户中心”-“我的帐户”-“安全认证”获取。
>>> res = oss.delete_bucket("no-such-bucket-in-oss")
>>> print res.status, res.getheaders(), res.read()
404
<Error>
  <BucketName>no-such-bucket-in-oss</BucketName>
  <Code>NoSuchBucket</Code>
  <Message>The specified bucket does not exist.</Message>
  <RequestId>51FE1374EDD2656F5B61009F</RequestId>
  <HostId>no-such-bucket-in-oss.oss.aliyuncs.com</HostId>
</Error>
```

404 表示不存在。Body 中说明了是因为删除的 bucket 不存在。

```
>>> res = oss.delete_bucket("oss-test-sample")
>>> print "%s\n%s" % (res.status, res.read())
204
```

204 表示成功删除 bucket。Body 为空，没有内容显示。

```
>>> res = oss.delete_bucket("oss")
>>> print "%s\n%s" % (res.status, res.read())
```

403

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>AccessDenied</Code>
  <Message>AccessDenied</Message>
  <RequestId>51FE17517D42520E5E439A75</RequestId>
  <HostId>oss.oss.aliyuncs.com</HostId>
</Error>
```

403 表示禁止访问。Body 中进一步说明了用户没有权限删除 oss 这个 bucket。

3. 修改 bucket 的 acl

函数定义：

def put_bucket(self, bucket, acl='', headers=None):

函数说明：

等同 create_bucket 函数。

函数定义：

def create_bucket(self, bucket, acl='', headers=None):

函数说明：

这个函数既可以用来初次创建 bucket，也可以在 bucket 创建后修改 bucket 的 acl。

4. 获取 bucket 访问控制权限

函数定义：

def get_bucket_acl(self, bucket):

函数说明：

该函数用来获取用户的 bucket acl。

参数说明：

bucket：

类型：string

返回值说明：

HTTP Response

注意事项：

acl 存在于 HTTP Response 的 Body 中，需要解析 xml 才能得到想要的结果。

使用示例：

```
>>> from oss.oss_api import *
>>>oss = OssAPI("oss.aliyuncs.com", "ID", "KEY")   #ID 和 KEY 需要填成用户的，可以在阿里云网站-“用户中心”-“我的帐户”-“安全认证”获取。
>>> res = oss.create_bucket("oss-test-sample")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.get_bucket_acl("oss-test-sample")
>>> print "%s\n%s" % (res.status, res.read())
200
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy>
  <Owner>
```

```
    <ID>oss-tester</ID>
    <DisplayName>oss-tester</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>private</Grant>
  </AccessControlList>
</AccessControlPolicy>
```

200 表示获取 bucket 的 acl 成功。Body 中显示了 bucket 的 acl 为 private。

```
>>> res = oss.get_bucket_acl("oss")
>>> print "%s\n%s" % (res.status, res.read())
403
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>AccessDenied</Code>
  <Message>AccessDenied</Message>
  <RequestId>51FE778E51BA961427499E42</RequestId>
  <HostId>oss.oss.aliyuncs.com</HostId>
</Error>
```

403 表示用户没有权限去获取 oss 这个 bucket 的权限。

5. 显示创建的 bucket

函数定义：

def list_all_my_buckets(self, headers=None):

函数说明：

该函数用来获取用户所创建的 bucket。

参数说明：

headers：

类型：dict

返回值说明：

HTTP Response

注意事项：

所有的 bucket 名称等信息都存在于 HTTP Response 的 Body 中，需要解析 xml 才能得到想要的结果。

使用示例：

```
>>> from oss.oss_api import *
>>>oss = OssAPI("oss.aliyuncs.com", "ID", "KEY")   #ID 和 KEY 需要填成用户的，可以在阿里云网站-“用户中心”-“我的帐户”-“安全认证”获取。
>>> res = oss.list_all_my_buckets()
>>> print "%s\n%s" % (res.status, res.read())
200
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult>
  <Owner>
    <ID>oss-tester</ID>
    <DisplayName>oss-tester</DisplayName>
```

```
    </Owner>
    <Buckets>
      <Bucket>
        <Name>oss-test-sample</Name>
        <CreationDate>2013-08-04T15:36:11.000Z</CreationDate>
      </Bucket>
    </Buckets>
</ListAllMyBucketsResult>
```

200 表示获取所有的 bucket 的列表成功。Body 中显示了 bucket 的创建时间，owner 等信息。

函数定义：

def get_service(self, bucket):

函数说明：

等同 list_all_my_buckets 函数。

6. 上传 object

函数定义：

def put_object_from_file(self, bucket, object, filename,
content_type='', headers=None,
params=None):

函数说明：

该函数将本地的一个文件的内容上传到 bucket 下，名字由 object 来指定。该 object 的 content-type 可以传入指定，如果没有指定，函数内部会调用 mimetypes 模块，根据本地文件名的后缀来获取 content-type。

参数说明：

bucket：

类型：string

object：

类型：string

filename：

类型：string

本地文件的文件名字

content_type：

类型：string

默认为空，当 content_type 为空时，函数内部会调用 mimetypes 模块，根据本地文件名的后缀来获取 content-type。当无法获取时会使用 application/octet-stream 作为 object 的 content_type。

headers:

类型：dict

headers 中的参数会作为 HTTP 请求中的 header 传送给 OSS。

params:

类型：dict

params 中的参数会作为 HTTP 请求中的 query string 传送给 OSS。

返回值说明：

HTTP Response

注意事项：

在上传文件的时候为单进程，且本地文件的大小不能超过 5GB。

使用示例：

```
>>> from oss.oss_api import *
>>>oss = OssAPI("oss.aliyuncs.com", "ID", "KEY")   #ID 和 KEY 需要填成用户的，可以在阿里云网站-“用户中心”-“我的帐户”-“安全认证”获取。
>>> res = oss.create_bucket("oss-test-sample")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.put_object_from_file("oss-test-sample", "object_name", "test.txt")
>>> print "%s\n%s" % (res.status, res.read())
200
```

200 表示成功将本地的 test.txt 文件上传到 bucket 中了。

```
>>> res = oss.put_object_from_file("oss-test-sample", "object_name", "/tmp/test.txt")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/ali/lib/python2.5/site-packages/oss-0.1.3-py2.5.egg/oss/oss_api.py", line 614, in put_object_from_file
    fp = open(filename, 'rb')
IOError: [Errno 2] No such file or directory: '/tmp/test.txt'
```

上传文件不存在，SDK 直接抛出异常。

函数定义：

def put_object_from_string(self, bucket, object, input_content,
content_type=DefaultContentType,
headers=None, params=None):

函数说明：

该函数将字符串中的内容上传到 bucket 下，名字由 object 来指定。

该 object 的 content-type 可以传入指定，如果没有指定，默认为 application/octet-stream。

参数说明：

bucket：

类型：string

object：

类型：string

input_content：

类型：string

需要上传的字符串。

content_type：

类型：string

需要用户指定。如果用户不指定，默认为 application/octet-stream。

headers:

类型：dict

headers 中的参数会作为 HTTP 请求中的 header 传送给 OSS。

params:

类型：dict

params 中的参数会作为 HTTP 请求中的 query string 传送给 OSS。

返回值说明：

HTTP Response

注意事项：

需要指定 content_type。

使用示例：

```
>>> from oss.oss_api import *
>>>oss = OssAPI("oss.aliyuncs.com", "ID", "KEY")   #ID 和 KEY 需要填成用户的，可以在阿里云网站-“用户中心”-“我的帐户”-“安全认证”获取。
>>> res = oss.create_bucket("oss-test-sample")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.put_object_from_string("oss-test-sample", "object_name2", "hello, oss", "plain/text")
>>> print "%s\n%s" % (res.status, res.read())
200
```

200 表示成功将字符串传到 bucket 中了。

7. 显示上传的 object

函数定义：

def get_bucket(self, bucket, prefix='', marker='', delimiter='', maxkeys='', headers=None):

函数说明：

该函数将显示 bucket 下已经上传了哪些 object。

参数说明：

bucket：

类型：string

prefix：

类型：string

限定返回的 object 必须以 prefix 作为前缀。注意使用 prefix 查询时，返回的 object 中仍会包含 prefix。

marker：

类型：string

设定返回的 object 从 marker 之后按字母排序的第一个开始返回，返回的 object 中不会包含 marker。

delimiter：

类型：string

用于对 object 进行分组的字符，通常为单个字符。

maxkeys：

类型：string

用于限定一次查询操作返回的 object 最大个数，默认为 1000 个，并且 1000 也是一次所能返回的最大个数。

headers:

类型：dict

headers 中的参数会作为 HTTP 请求中的 header 传送给 OSS。

返回值说明：

HTTP Response

注意事项：

需要根据不用的需求来指定 prefix，marker，delimiter，maxkeys 等参数。并且需要从 Body 的 xml 文件中来解析相关内容。

使用示例：

```
>>> from oss.oss_api import *
>>>oss = OssAPI("oss.aliyuncs.com", "ID", "KEY")  #ID 和 KEY 需要填成用户的，可以在阿里云网站-“用户中心”-“我的帐户”-“安全认证”获取。
>>> res = oss.create_bucket("oss-test-sample")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.put_object_from_string("oss-test-sample", "1", "hello, oss", "plain/text")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.put_object_from_string("oss-test-sample", "2", "hello, oss", "plain/text")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.put_object_from_string("oss-test-sample", "3", "hello, oss", "plain/text")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.put_object_from_string("oss-test-sample", "4", "hello, oss", "plain/text")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.put_object_from_string("oss-test-sample", "5", "hello, oss", "plain/text")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.get_bucket("oss-test-sample")
>>> print "%s\n%s" % (res.status, res.read())
200
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
  <Name>oss-test-sample</Name>
  <Prefix></Prefix>
  <Marker></Marker>
  <MaxKeys>100</MaxKeys>
  <Delimiter></Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>1</Key>
    <LastModified>2013-08-05T06:23:23.000Z</LastModified>
```

```
    <ETag>"B353A6AC7A6EE89236F6D56F81694BBB"</ETag>
    <Type>Normal</Type>
    <Size>10</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>oss-tester</ID>
      <DisplayName>oss-tester</DisplayName>
    </Owner>
  </Contents>
  <Contents>
    <Key>2</Key>
    <LastModified>2013-08-05T06:23:44.000Z</LastModified>
    <ETag>"B353A6AC7A6EE89236F6D56F81694BBB"</ETag>
    <Type>Normal</Type>
    <Size>10</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>oss-tester</ID>
      <DisplayName>oss-tester</DisplayName>
    </Owner>
  </Contents>
  <Contents>
    <Key>3</Key>
    <LastModified>2013-08-05T06:23:50.000Z</LastModified>
    <ETag>"B353A6AC7A6EE89236F6D56F81694BBB"</ETag>
    <Type>Normal</Type>
    <Size>10</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>oss-tester</ID>
      <DisplayName>oss-tester</DisplayName>
    </Owner>
  </Contents>
  <Contents>
    <Key>4</Key>
    <LastModified>2013-08-05T06:27:35.000Z</LastModified>
    <ETag>"B353A6AC7A6EE89236F6D56F81694BBB"</ETag>
    <Type>Normal</Type>
    <Size>10</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>oss-tester</ID>
      <DisplayName>oss-tester</DisplayName>
    </Owner>
  </Contents>
```

```
  <Contents>
    <Key>5</Key>
    <LastModified>2013-08-05T06:27:38.000Z</LastModified>
    <ETag>"B353A6AC7A6EE89236F6D56F81694BBB"</ETag>
    <Type>Normal</Type>
    <Size>10</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>oss-tester</ID>
      <DisplayName>oss-tester</DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

200 表示查询成功了。Body 中显示了上传的 5 个 object。具体 xml 中的内容的含义请参考 OSS API 中的说明。

```
>>> res = oss.get_bucket("oss-test-sample", "1")
>>> print "%s\n%s" % (res.status, res.read())
200
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
  <Name>oss-test-sample</Name>
  <Prefix>1</Prefix>
  <Marker></Marker>
  <MaxKeys>100</MaxKeys>
  <Delimiter></Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>1</Key>
    <LastModified>2013-08-05T06:23:23.000Z</LastModified>
    <ETag>"B353A6AC7A6EE89236F6D56F81694BBB"</ETag>
    <Type>Normal</Type>
    <Size>10</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>oss-tester</ID>
      <DisplayName>oss-tester</DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

此次查询为找前缀为 1 的 object，200 表示查询成功了。Body 中显示了上传的 1 个 object。

```
>>> res = oss.get_bucket("oss-test-sample", "", "", "", "2")
>>> print "%s\n%s" % (res.status, res.read())
200
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
  <Name>oss-test-sample</Name>
  <Prefix></Prefix>
  <Marker></Marker>
  <MaxKeys>2</MaxKeys>
  <Delimiter></Delimiter>
  <IsTruncated>true</IsTruncated>
  <NextMarker>2</NextMarker>
  <Contents>
    <Key>1</Key>
    <LastModified>2013-08-05T06:23:23.000Z</LastModified>
    <ETag>"B353A6AC7A6EE89236F6D56F81694BBB"</ETag>
    <Type>Normal</Type>
    <Size>10</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>oss-tester</ID>
      <DisplayName>oss-tester</DisplayName>
    </Owner>
  </Contents>
  <Contents>
    <Key>2</Key>
    <LastModified>2013-08-05T06:23:44.000Z</LastModified>
    <ETag>"B353A6AC7A6EE89236F6D56F81694BBB"</ETag>
    <Type>Normal</Type>
    <Size>10</Size>
    <StorageClass>Standard</StorageClass>
    <Owner>
      <ID>oss-tester</ID>
      <DisplayName>oss-tester</DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

此次查询最多返回 2 个结果，200 表示查询成功了。Body 中显示了上传的 2 个 object。Body 中的其他内容请参考 OSS API 文档。

函数定义：

def list_bucket(self, bucket, prefix='', marker='', delimiter='', maxkeys='', headers=None):

函数说明：

等同 get_bucket 函数。

8. 删除 object

函数定义：

def delete_object(self, bucket, object, headers=None):

函数说明：

该函数用来删除指定 bucket 的 object。

参数说明：

bucket：

类型：string

object：

类型：string

headers：

类型：dict

可以用来设置 HTTP 请求中的 Headers，默认为空。

返回值说明：

HTTP Response

注意事项：

一般的情况下只需要指定 bucket 和 object，headers 可以不用填。

删除成功返回 204，不是 200。删除一个已经成功删除的 object 依然会返回 204，而不是 404。

使用示例：

```
>>> from oss.oss_api import *
>>> oss = OssAPI("oss.aliyuncs.com", "ID", "KEY")   #ID 和 KEY 需要填成用户的，可以在阿里云网站-“用户中心”-“我的帐户”-“安全认证”获取。
>>> res = oss.create_bucket("oss-test-sample")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.put_object_from_string("oss-test-sample", "1", "hello, oss", "plain/text")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.delete_object("oss-test-sample", "1")
>>> print "%s\n%s" % (res.status, res.read())
204
```

204 表示删除成功。Body 为空，所以没有内容显示。

```
>>> res = oss.delete_object("oss-test-sample", "1")
>>> print "%s\n%s" % (res.status, res.read())
204
```

重复删除仍然返回 204。

9. 下载 object

函数定义：

def get_object_to_file(self, bucket, object, filename, headers=None):

函数说明：

该函数用来将指定 bucket 的 object 的内容下载到本地文件，文件名由 filename 来指定。

参数说明：

bucket：

类型：string

object：

类型：string

filename：

类型：string

headers：

类型：dict

可以用来设置 HTTP 请求中的 Headers，默认为空。

返回值说明：

HTTP Response

注意事项：

一般的情况下只需要指定 bucket 和 object，filename。headers 可以不用填。下载的时候如果没有下载成功，本地文件是不会被删除。需要用户自己清除。如果指定同一个本地文件，文件会被覆盖。

使用示例：

```
>>> from oss.oss_api import *
>>> oss = OssAPI("oss.aliyuncs.com", "ID", "KEY")   #ID 和 KEY 需要填成用户的，可以在阿里云网站-“用户中心”-“我的帐户”-“安全认证”获取。
>>> res = oss.create_bucket("oss-test-sample")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.put_object_from_string("oss-test-sample", "1", "hello, oss", "plain/text")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> res = oss.get_object_to_file("oss-test-sample", "1", "/tmp/test.1")
>>> print "%s\n%s" % (res.status, res.read())
200
>>> f = open("/tmp/test.1")
>>> print f.read()
hello, oss
>>> f.close()
```

200 表示下载成功，打开文件后读取内容，和上传之前的内容一致。

```
>>> res = oss.delete_object("oss-test-sample”, “1”)
>>> print "%s\n%s" % (res.status, res.read())
204
>>> res = oss.get_object_to_file("oss-test-sample", "1", "/tmp/test.1")
>>> print "%s\n%s" % (res.status, res.read())
404
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The specified key does not exist.</Message>
  <Key>1</Key>
  <RequestId>51FF64A3A519E044305451C2</RequestId>
```

<HostId>oss-test-sample.oss.aliyuncs.com</HostId>

</Error>

404 表示下载的 object 不存在。由于前面删除了 object，所以下载的 object 不存在。

3.2. 使用 osscmd

这部分对 osscmd 的命令进行一些说明，并示范如何用 osscmd 创建 bucket，上传 object，列出 object，删除 object。

osscmd 中用 oss://bucket 或者 oss://bucket/object 表示这是一个 bucket 或者 object。oss://只是一种资源的表示方式，没有其他意义。

3.2.1. osscmd 命令说明

3.2.1.1.**getallbucket(gs)**

命令说明：

getallbucket(gs)

用来显示用户创建的 bucket。gs 是 get service 的简写。(gs)表示和 getallbucket 是同样的效果。

使用示范：

- python osscmd getallbucket
- python osscmd gs

3.2.1.2.**createbucket**(cb,mb,pb)

命令说明：

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

创建 bucket 的命令，cb 是 create bucket 的简写，mb 是 make bucket 的简写，pb 是 put bucket 的简写，oss://bucket 表示 bucket。--acl 参数可以传入，也可以不传入。这几个命令都是同样的效果。

使用示范：

- 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

3.2.1.3.**deletebucket**(db)

命令说明：

deletebucket(db) oss://bucket

删除 bucket 的命令，db 是 delete bucket 的简写。deletebucket 和 db 是同样的效果。

使用示范：

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

3.2.1.4.deletewholebucket

注意：

该命令十分危险，将会删除所有的数据，并且不可恢复。请慎重使用。

命令说明：

deletewholebucket oss://bucket

删除 bucket 及其内部 object 以及 multipart 相关的内容。

使用示范：

- python osscmd deletewholebucket oss://mybucket

3.2.1.5.getacl

命令说明：

getacl oss://bucket

获取 bucket 的访问控制权限

使用示范：

- python osscmd getacl oss://mybucket

3.2.1.6.setacl

命令说明：

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

修改 bucket 的访问控制权限。acl 只允许为 private，public-read，public-read-write 三个当中的一个。

使用示范：

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

3.2.1.7.ls(list)

命令说明：

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

列出 bucket 中的 object。

使用示范：

- 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

命令说明：

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

列出 bucket 中的 object。

使用示范：

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

3.2.1.8.mkdir

命令说明：

mkdir oss://bucket/dirname

创建一个以“/”结尾的 object，并且 size 为 0。

使用示范：

- python osscmd mkdir oss://mybucket/folder

3.2.1.9.listallobject

命令说明：

listallobject oss://bucket/[prefix]

显示 bucket 下所有的 object，可以指定 prefix 来显示。

使用示范：

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

3.2.1.10. deleteallobject

命令说明：

deleteallobject oss://bucket/[prefix]

删除 bucket 下所有的 object，可以指定特定的 prefix 来删除。

使用示范：

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

3.2.1.11. downloadallobject

命令说明：

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

将 bucket 下的 object 下载到本地目录，并且保持目录结构。可以指定 prefix 下载。--replace=false 表示如果下载时，本地已经存在同名文件，不会覆盖。true 则会覆盖。

使用示范：

- 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

3.2.1.12. downloadtodir

命令说明：

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

将 bucket 下的 object 下载到本地目录，并且保持目录结构。可以指定 prefix 下载。--replace=false 表示如果下载时，本地已经存在同名文件，不会覆盖。true 则会覆盖。同 downloadallobject 效果一样。

使用示范：

- 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

3.2.1.13. uploadfromdir

命令说明：

uploadfromdir localdir oss://bucket/[prefix]
--check_point=check_point_file

将本地目录里的文件上传到 bucket 中。例如 localdir 为/tmp/
里面有 a/b，a/c，a 三个文件，则上传到 OSS 中为 oss://bucket/a/b，oss://bucket/a/c，oss://bucket/a。如果指定了 prefix 为 mytest，则上传到 OSS 中为 oss://bucket/mytest/a/b，oss://bucket/mytest/a/c，oss://bucket/mytest/a。

--check_point=check_point_file 是指定文件。指定文件后，osscmd 会将已经上传的本地文件以时间戳的方式放到 check_point_file 中，uploadfromdir 命令会将正在上传的文件的时间戳和 check_point_file 记录的时间戳进行比较。如果有变化则会重新上传，否则跳过。默认情况下是没有

check_point_file 的。

注意：由于 check_point_file 文件中记录的是上传的所有文件的。所以当上传文件特别多的时候，check_point_file 会特别巨大。

使用示范：

- 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

3.2.1.14. put

命令说明：

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

上传一个本地的文件到 bucket 中，可以指定 object 的 content-type，或则指定自定义的 headers。

使用示范：

- 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

3.2.1.15. upload

命令说明：

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

将本地文件以 object group 的形式上传。**不推荐使用。**

使用示范：

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

3.2.1.16. get

命令说明：

get oss://bucket/object localfile

将 object 下载到本地文件。

使用示范：

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

3.2.1.17. multiget(multi_get)

命令说明：

multiget(multi_get) oss://bucket/object localfile

将 object 以多线程的方式下载到本地文件。

使用示范：

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

3.2.1.18. cat

命令说明：

cat oss://bucket/object

读取 object 的内容，直接打印出来。在 object 内容比较大的时候请不要使用。

使用示范：

- python osscmd cat oss://mybucket/myobject

3.2.1.19. meta

命令说明：

meta oss://bucket/object

读取 object 的 meta 信息，打印出来。meta 信息包括 content-type，文件长度，自定义 meta 等内容。

使用示范：

- python osscmd meta oss://mybucket/myobject

3.2.1.20. copy

命令说明：

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

将源 bucket 的源 object 复制到目的 bucket 中的目的 object。

使用示范：

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

3.2.1.21. rm(delete,del)

命令说明：

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

删除 object。

使用示范：

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

3.2.1.22. signurl(sign)

命令说明：

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

生成一个包含签名的 URL，并指定超时的时间。适用于 bucket 为私有时将特定的 object 提供给他人访问。

使用示范：

- python osscmd sign oss://mybucket/myobject
- python osscmd signurl oss://mybucket/myobject

3.2.1.23. init

命令说明：

init oss://bucket/object

初始化生成一个 Upload ID。这个 Upload ID 可以配合后面的 multiupload 命令来使用。

使用示范：

- python osscmd init oss://mybucket/myobject

3.2.1.24. listpart

命令说明：

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

显示指定 object 的 Upload ID 下已经上传的 Parts。相关概念见 OSS API 文档。必须要指定 Upload ID。

使用示范：

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

3.2.1.25. listparts

命令说明：

listparts oss://bucket

显示 bucket 中未完成的 multipart Upload ID 和 object。一般在删除 bucket 提示 bucket 非空的情况下可以用这个命令查看是否有 multipart 相关的内容。

使用示范：

- python osscmd listparts oss://mybucket

3.2.1.26. getallpartsize

命令说明：

getallpartsize oss://bucket

显示 bucket 中还存在的 Upload ID 已经上传的 Parts 的总大小。

使用示范：

- python osscmd getallpartsize oss://mybucket

3.2.1.27. cancel

命令说明：

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

终止 Upload ID 对应的 Multipart Upload 事件。

使用示范：

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

3.2.1.28. multiupload(multi_upload,mp)

命令说明：

multiupload(multi_upload,mp) localfile oss://bucket/object

将本地文件以 multipart 的方式上传到 OSS。

使用示范：

- 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

命令说明：

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

将本地文件以 multipart 的方式上传到 OSS。本地文件划分的块数由 max_part_num 来指定。这个命令在实现的时候，会先去判断 Upload ID 对应的 Parts 的 ETag 是否和本地文件的 MD5 值是否相等，相等则跳过上传。所以如果在使用之前生成一个 Upload ID，作为参数传进来。即使上传没有成功，重复执行相同的 multiupload 命令可以达到一个断点续传的效果。

使用示范：

- 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

3.2.1.29. uploadpartfromfile (upff)

命令说明：

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

主要用于测试，不推荐使用。

3.2.1.30. uploadpartfromstring(upfs)

命令说明：

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

主要用于测试，不推荐使用。

3.2.1.31. config

命令说明：

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

配置 osscmd 使用的默认 host，ID 和 KEY。默认的 host 为 oss.aliyuncs.com 如果需要访问 oss-internal.aliyuncs.com 可以加上 --host=oss-internal.aliyuncs.com。

使用示范：

- python osscmd config --id=your_id --key=your_key
- python osscmd config --id=your_id --key=your_key

--host=oss-internal.aliyuncs.com

3.2.2. osscmd 使用示例

1. 在 Linux 或者 Windows 上下载 SDK 安装包后，解压缩后就可以使用 osscmd 了。
   使用时直接调用 python osscmd 即可获取相应的说明。每种命令有两种执行模式。以查询用户所创建的 bucket 为例子。执行的是 gs 命令，get service 的简写。
   - 方法 1：不指定 ID 和 KEY，osscmd 从默认文件中读取 ID 和 KEY。
     $ **python osscmd gs**
     can't get accessid/accesskey, setup use : config --id=accessid --key=accesskey
     注意：如果出现这样的提示表明没有配置好 ID 和 KEY，见步骤 2 中提示的配置命令。

     如果配置好 ID 和 KEY，并且 ID 和 KEY 有效，执行
     $ python osscmd gs 会得类似如下的结果
     2013-07-19 08:11 test-oss-sample
     Bucket Number is: 1
   - 方法 2：直接在命令中指定 ID 和 KEY，osscmd 从命令行中读取 ID 和 KEY。
     $ **python osscmd gs --id=your_id --key=your_key**
     如果 ID 和 KEY 有效，执行后会得类似如下的结果
     2013-07-19 08:11 test-oss-sample
     Bucket Number is: 1

2. 配置用户的 ID 和 KEY 到默认的文件中。
   请运行如下命令用来配置访问 OSS 所需要的 ID 和 KEY。默认的 OSS HOST 为 oss.aliyuncs.com。
   $python osscmd config --id=YOUR_ID --key=YOUR_KEY

   如果出现类似“Your configuration is saved into ”的提示表明 ID 和 KEY 已经保存成功了。
   注意：如果需要访问 oss-internal.aliyun-inc.com 的用户，在配置的时候可以使用--host=oss-internal.aliyuncs.com 来修改默认的 OSS HOST。
   $python osscmd config --id=YOUR_ID --key=YOUR_KEY --host=oss-internal.aliyun-inc.com

3. 列出创建的 bucket。
   $python osscmd getallbucket
   如果是刚刚使用 OSS 的用户因为没有创建 bucket，输出是空

4. 创建 bucket。Bucket 名字为 **mybucketname。**
   $python osscmd createbucket mybucketname
   创建“mybucketname”的 bucket，有可能不成功。因为 OSS 中的 bucket 名字是全局唯一的，并且有人已经创建了这个 bucket。这个时候需要换一个名字。

例如在 bucket 名字中加入特定的日期。

5. 可以再次查看是否创建成功。
   $python osscmd getallbucket
   如果没有成功请检查 osscmd 返回的错误信息。

6. 成功创建 bucket 后，查看 bucket 中有哪些 object。
   $python osscmd list oss://mybucketname/
   由于 bucket 中还没有 object，输出是空的。

7. 向 bucket 中上传一个 object。假如本地文件名叫 local_existed_file，其 MD5 值如下所示。
   $ md5sum local_existed_file 7625e1adc3a4b129763d580ca0a78e44 local_existed_file
   $ python osscmd put local_existed_file oss://mybucketname/test_object

8. 如果创建成功，再次查看 bucket 中有哪些 object。
   $python osscmd list oss://mybucketname/

9. 从 bucket 中下载 object 到本地文件，并比对下载的文件的 md5 值
   $ python osscmd get oss://mybucketname/test_object download_file
   $ md5sum download_file
   7625e1adc3a4b129763d580ca0a78e44 download_file

10. 删除 bucket 中的 object
    $ python osscmd delete oss://mybucketname/test_object

11. 删除 bucket, 注意，如果 bucket 中还有 object 的话则这个 bucket 不能被删除
    $ python osscmd deletebucket test-oss-aliyun-com