

# 下载与安装

## 相关资源

- SDK 压缩包快速下载地址: 见帮助中心

## 环境依赖

对象存储的 Python SDK 目前可以支持 Python 2.7 以及 Python 3.4 - python3.11 (python3.12 及以上版本暂不支持), 需要确保使用环境已经安装 Python (2.X 或者 3.X) 和 pip (Python2.X) 或者 pip3 (Python3.X)。

## SDK 安装

目前 SDK 安装仅支持下载之后离线安装的方式。

- Linux 下安装

下载 Python 版本 SDK 压缩包, 使用以下命令进行安装, 最好基于 root 用户进行安装, 不然部分脚本可能执行失败。

Python2.X 版本

```
unzip sdk_python2.X.zip
cd sdk_python2.X/sdk/

pip install -r requirements.txt
sh install_extension.sh
```

Python3.X 版本

```
unzip sdk_python3.X.zip
cd sdk_python3.X/sdk/

pip3 install -r requirements.txt
sh install_extension.sh
```

- windows 下安装

下载 Python 版本 SDK 压缩包, 使用以下命令进行安装。需要保证您的 windows 电脑上有解压工具, 可以解压下载下来的 .zip 文件。在文件解压成功后, 进入 windows 命令行, 执行以下命令:

Python2.X 版本

```
pip install -r requirements.txt
python install_extension_window.py
```

Python3.X 版本

```
pip install -r requirements.txt
python install_extension_window.py
```

在这里说明一下，window 下面的 python install\_extension\_window.py 命令是将压缩包下面的 service-2.s3.sdk-extras.json 文件拷贝到 python 的执行目录下，如果您执行的时候失败，需要您自己将 service-2.s3.sdk-extras.json 拷贝到 “python 安装目录\Lib\site-packages\botocore\data\s3\2006-03-01” 下面即可。

## 连接

在正式使用 SDK 接口之前，需要先连接 ZOS，使用以下的示例可以连接 ZOS(连接示例基于 Python2.7.5，同以下所有接口示例)。

```
#!/usr/bin/env python
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="" #access_key
secret_key="" #secret_key
url="" #ZOS 集群节点
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
```

## 全局错误码定义

请求可能会返回相关错误，具体错误码编号及信息请参考下表。同一个错误码可能对应不同的错误码描述，具体由接口来决定。

| 错误码 | 错误码描述                           |
|-----|---------------------------------|
| 100 | Continue                        |
| 200 | Success                         |
| 201 | Created                         |
| 202 | Accepted                        |
| 204 | NoContent                       |
| 206 | Partial content                 |
| 304 | NotModified                     |
| 400 | InvalidArgument                 |
| 400 | InvalidDigest                   |
| 400 | BadDigest                       |
| 400 | InvalidBucketName               |
| 400 | InvalidObjectName               |
| 400 | UnresolvableGrantByEmailAddress |
| 400 | InvalidPart                     |
| 400 | InvalidPartOrder                |
| 400 | RequestTimeout                  |

|     |                      |
|-----|----------------------|
| 400 | EntityTooLarge       |
| 403 | AccessDenied         |
| 403 | UserSuspended        |
| 403 | RequestTimeTooSkewed |
| 404 | NoSuchKey            |
| 404 | NoSuchBucket         |
| 404 | NoSuchUpload         |
| 405 | MethodNotAllowed     |
| 408 | RequestTimeout       |
| 409 | BucketAlreadyExists  |
| 409 | BucketNotEmpty       |
| 411 | MissingContentLength |
| 412 | PreconditionFailed   |
| 416 | InvalidRange         |
| 422 | UnprocessableEntity  |
| 500 | InternalServerError  |

# 1、Bucket 操作

## 1.1、Create Bucket

### 功能说明

Create Bucket 请求可以在指定账号下创建一个新的 Bucket。

### 方法原型

```
create_bucket(
    ACL="private"|"public-read"|"public-read-write"|"authenticated-read",
    Bucket="bucket-name",
    ObjectLockEnabledForBucket=True|False
)
```

### 参数说明

| 参数名称   | 参数描述   | 类型     | 是否必须 |
|--------|--|--------|------|
| ACL    | Bucket 的 ACL 配置，配置参考 <a href="#">1.9</a>                     | String | 否    |
| Bucket | Bucket 的名称，长度 3 到 63 位，必须以数字或者字母作为开头，必须以数字和字符以及特殊字符-和_和.作为结尾 | String | 是    |

|                            |   |         |   |
|----------------------------|---|---------|---|
| ObjectLockEnabledForBucket | 开启关闭新创建 Bucket 的 Object 锁定功能，具体锁定规则参考 1.25                            | Boolean | 否 |
| AZPolicy                   | 桶的数据冗余策略，可选值为 `single-az` 和 `multi-az`，分别表示单 AZ 存储和多 AZ 存储。默认为单 AZ 存储 | String  | 否 |
| StorageClass               | 桶的存储类型，可选值为 `STANDARD`、`STANDARD_IA` 和 `GLACIER`，分别表示标准、低频、归档。默认为标准存储 | String  | 否 |

## 返回结果及说明

```
{
  'ResponseMetadata': {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx0000000000000000000016-0060a4ca03-47ce2-default',
    'HTTPHeaders': {
      'date': 'Wed, 19 May 2021 08:19:15 GMT',
      'content-length': '0',
      'x-amz-request-id': 'tx0000000000000000000016-0060a4ca03-47ce2-default',
      'connection': 'Keep-Alive'
    }
  }
}
```

| 返回结果             | 描述          | 类型   |
|------------------|-------------|------|
| ResponseMetadata | http 请求返回数据 | Dict |

## 示例

```
#!/usr/bin/env python
#-*-coding:utf-8 -*-
from boto3.session import Session
from botocore.exceptions import ClientError
import sys

def is_client_error(code, e=None):
    if e is None:
        exc = sys.exc_info()[1]
    else:
        exc = e
    if isinstance(exc, ClientError) and exc.response["Error"]["Code"] == code:
        return ClientError
    return type("NeverEverRaisedException", (Exception,), {}) if e is None else False

def test_cb(bname, acl):
    access_key="your access key"
    secret_key="your secretkey"
```

```

url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
try:
    response = s3_client.create_bucket(Bucket=bname,ACL=acl)

except is_client_error("BucketAlreadyExists"):
    print('catch exception BucketAlreadyExists')
except is_client_error("BucketAlreadyOwnedByYou"):
    print('catch exception BucketAlreadyOwnedByYou')
except is_client_error("InvalidBucketName"):
    print("catch exception InvalidBucketName")
else:
    print(response)
if __name__ == '__main__':
    bucket = 'testing-bucket'
    ACL = "public-read-write"
    test_cb(bucket,ACL)

```

## 1.2、Delete Bucket

### 功能说明

Delete Bucket 请求可以在指定账号下删除 Bucket，删除之前要求 Bucket 为空。

### 方法原型

```
delete_bucket(Bucket='string')
```

### 参数说明

| 参数名称   | 参数描述            | 类型     | 是否必须 |
|--------|-----------------|--------|------|
| Bucket | 需要删除的 Bucket 名称 | String | 是    |

### 返回结果说明

```

{
  "ResponseMetadata": {
    "HTTPStatusCode": 204,
    "RetryAttempts": 0,
    "HostId": "",
    "RequestId": "tx000000000000000000006d-0060afde86-386a-default",
    "HTTPHeaders": {
      "date": "Thu, 27 May 2021 18:01:43 GMT",

```

```

        "x-amz-request-id": "tx00000000000000000006d-0060afde86-
386a-default"
    }
}
}

```

| 返回结果             | 描述          | 类型   |
|------------------|-------------|------|
| ResponseMetadata | http 请求返回数据 | Dict |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
from botocore.exceptions import ClientError
import sys

def is_client_error(code, e=None):
    if e is None:
        exc = sys.exc_info()[1]
    else:
        exc = e
    if isinstance(exc, ClientError) and exc.response["Error"]["Code"]
== code:
        return ClientError
    return type("NeverEverRaisedException", (Exception,), {}) if e is
None else False

def test_rb(bname):
    # #void
    access_key="your access key"
    secret_key="your secretkey"
    url="zos endpoint"
    session=Session(access_key,secret_key)
    s3_client=session.client("s3",endpoint_url=url)
    try:
        response = s3_client.delete_bucket(Bucket=bucket)

    except is_client_error("NoSuchBucket") as e:
        print('catch exception NoSuchBucket')
    except is_client_error("InvalidBucketName") as e:
        print("catch exception InvalidBucketName")
    except is_client_error("BucketNotEmpty") as e:
        print("catch exception BucketNotEmpty")
    else:
        print(response)

if __name__ == '__main__':
    bucket = 'test-bucket2'

    test_rb(bucket)

```

## 1.3、Head Bucket

### 功能说明

Head Bucket 请求可以判断某个 Bucket 是否存在或者是否有权限访问该 Bucket (其他用户名下 Bucket)。

### 方法原型

```
head_bucket(Bucket="bucket-name")
```

### 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

### 返回结果说明

```
{
  'ResponseMetadata': {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx0000000000000000000017-0060a4cd72-47ce2-default',
    'HTTPHeaders': {
      'content-length': '0',
      'connection': 'Keep-Alive',
      'x-amz-request-id': 'tx0000000000000000000017-0060a4cd72-47ce2-default',
      'x-rgw-object-count': '3',
      'date': 'Wed, 19 May 2021 08:33:54 GMT',
      'x-rgw-bytes-used': '20971986'
    }
  }
}
```

| 返回结果               | 描述   | 类型   |
|--------------------|--|------|
| ResponseMetadata   | http 请求返回数据  | Dict |
| HTTPStatusCode     | http 请求状态码   | Int  |
| x-rgw-object-count | 该 Bucket 内实际存在的文件数量, 非可见的文件数量, 和 Bucket 多版本相关, 参考 1.32 Put Bucket Versioning , 有些文件被删除的时候并没有 | Int  |

|                  |                          |     |
|------------------|--------------------------|-----|
|                  | 真正删除，主要用于数据恢复。           |     |
| x-rgw-bytes-used | 当前 Bucket 文件占据的实际空间，单位字节 | Int |

## 示例

```
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.head_bucket(Bucket="bucket-name")
```

## 1.4、List Bucket

### 功能说明

List Bucket 可以列出请求用户拥有的所有的 bucket 列表。

### 方法原型

```
list_buckets()
```

### 参数说明

无

### 返回结果说明

```
{"Owner": {"DisplayName": "liubingrun", "ID": "void"},
" Buckets": [
    {"CreationDate": datetime.datetime(2021, 5, 28, 8, 21, 52, 776000, tzinfo=tzutc()),
    "Name": "test-bucket"
    },
    {"CreationDate": datetime.datetime(2021, 5, 28, 8, 21, 56, 389000, tzinfo=tzutc()),
    "Name": "test-bucket1"
    },
    {"CreationDate": datetime.datetime(2021, 5, 28, 8, 21, 57, 765000, tzinfo=tzutc()),
    "Name": "test-bucket2"
    }
],
```



```

    "ResponseMetadata": {
      "HTTPStatusCode": 200,
      "RetryAttempts": 0,
      "HostId": "",
      "RequestId": "tx0000000000000000000000f-0060b0a885-20bdd-
default",
      "HTTPHeaders": {
        "transfer-encoding": "chunked",
        "date": "Fri, 28 May 2021 08:23:33 GMT",
        "x-amz-request-id": "tx0000000000000000000000f-0060b0a885-
20bdd-default",
        "content-type": "application/xml"
      }
    }
  }
}

```

| 返回结果             | 描述             | 类型   |
|------------------|----------------|------|
| ResponseMetadata | http 请求返回数据    | dict |
| Owner            | 列出的 bucket 的属主 | dict |
| Buckets          | bucket 的列表     | list |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client("s3", endpoint_url=url)
response = s3_client.list_buckets()
print("%s's bucket:" % response["Owner"]["DisplayName"])
for bucket in response["Buckets"]:
    print('bucket: %s create on %s' %
          (bucket["Name"], bucket["CreationDate"]))

```

## 1.5、List Bucket Object

### 功能说明

List Bucket Object 请求可以列出该 Bucket 下部分或者所有 Object，发起该请求需要拥有 Read 权限。

### 方法原型

```

client.list_objects(
    Bucket='string',

```

```
Delimiter='string',
Marker='string',
MaxKeys=123,
Prefix='string',
)
```

## 参数说明

| 参数名称      | 参数描述                          | 类型      | 是否必须 |
|-----------|-------------------------------|---------|------|
| Bucket    | Bucket 的名称                    | String  | 是    |
| Delimiter | 对文件名称进行分组的字符                  | String  | 否    |
| Marker    | 本次 list 操作的起始点                | String  | 否    |
| MaxKeys   | 一次返回 keys 的最大数目（默认值和上限为 1000） | Integer | 否    |
| Prefix    | 设置返回的 key 的前缀                 | String  | 否    |

## 返回结果说明

```
{
  "Name": "demo-bucket",
  "ResponseMetadata": {
    "HTTPStatusCode": 200,
    "RetryAttempts": 0,
    "HostId": "",
    "RequestId": "tx00000000000000000008c2-0060b455b8-20bdd-
default",
    "HTTPHeaders": {
      "transfer-encoding": "chunked",
      "date": "Mon, 31 May 2021 03:19:20 GMT",
      "x-amz-request-id": "tx00000000000000000008c2-0060b455b8-
20bdd-default",
      "content-type": "application/xml"
    }
  },
  "MaxKeys": 100,
  "Prefix": "",
  "Marker": "",
  "EncodingType": "url",
  "IsTruncated": False,
  "Contents": [
    {
      "LastModified": datetime.datetime(2021, 5, 31, 1, 36, 11,
396000, tzinfo=tzutc()),
      "ETag": "\"5e692532922c938080e691a0f895993d\"",
      "StorageClass": "STANDARD",
      "Key": "test/f3/f3_3",
      "Owner": {
        "DisplayName": "void",
        "ID": "void"
      }
    }
  ],
}
```

```

        "Size": 1377
    },
]
}

```

| 返回结果             | 描述                     | 类型     |
|------------------|------------------------|--------|
| ResponseMetadata | http 请求返回数据            | dict   |
| Name             | Bucket 的名称             | String |
| IsTruncated      | 返回结果是否截断               | Bool   |
| Contents         | 返回对象的列表                | List   |
| NextMarker       | 若结果被截断，则给出分页所需的 Marker | String |

## 示例

```

from boto3.session import Session

def test_lb(bname):
    access_key="your access key"
    secret_key="your secretkey"
    url="zos endpoint"
    session = Session(access_key, secret_key)
    s3_client = session.client("s3", endpoint_url=url)

    response = s3_client.list_objects(Bucket=bname, MaxKeys=100)
    # print(response)
    for obj in response["Contents"]:
        print('object: %s Size: %s Owner: %s' %
              (obj["Key"], obj["Size"], obj['Owner']['DisplayName']))

def test_lb_Delimiter(bname):
    access_key="your access key"
    secret_key="your secretkey"
    url="zos endpoint"
    session = Session(access_key, secret_key)
    s3_client = session.client("s3", endpoint_url=url)

    response = s3_client.list_objects(Bucket=bname, Prefix='test/',
    Delimiter='/', MaxKeys=100)

    # print(response)
    for prefix in response["CommonPrefixes"]:
        p = prefix['Prefix']
        print(p)
        response = s3_client.list_objects(Bucket=bname, Prefix=p)
        for obj in response["Contents"]:
            print('object: %s Size: %s Owner: %s' %
                  (obj["Key"], obj["Size"], obj['Owner']['DisplayName']))

if __name__ == '__main__':
    bucket = 'demo-bucket'

```

```
test_lb(bucket)
print('-----')
test_lb_Delimiter(bucket)
```

## 1.6、Put Bucket Policy

### 功能说明

Put Bucket Policy 请求用于为 ZOS S3 bucket 设置桶策略。

### 方法原型

```
put_bucket_policy(
    Bucket="bucket-name",
    Policy='{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "id-1",
                "Effect": "Allow",
                "Principal": {"*"},
                "Action": [ "s3:PutObject", "s3:PutObjectAcl"],
                "Resource": ["arn:aws:s3:::bucketName/*"]
            }
        ]
    }'
```

### 参数说明

| 参数名称   | 参数描述            | 类型     | 是否必须 |
|--------|-----------------|--------|------|
| Bucket | Bucket 的名称      | String | 是    |
| Policy | 设置在 bucket 上的策略 | String | 是    |

Bucket Policy 各字段描述如下：

| 字段        | 描述  | 类型     | 是否必须 |
|-----------|---|--------|------|
| Version   | 保持与 Amazon S3 一致，当前支持"2012-10-17"和"2006-03-01"  | string | 否    |
| Id        | 桶策略 ID，桶策略的唯一标识   | string | 否    |
| Statement | 桶策略描述，定义完整的权限控制。每条桶策略的 Statement 可由多条描述组成，每条描述是一个 dict，每条描述可包含以下字段：<br>Sid<br>Effect<br>Principal | list   | 是    |

|             | Action<br>Resource<br>Condition  |        |   |
|-------------|--|--------|---|
| Sid         | 本条桶策略描述的 ID  | string | 否                                       |
| Effect      | 桶策略的效果，即指定本条桶策略描述的权限是接受请求还是拒绝请求。<br>接受请求：配置为“Allow”，<br>拒绝请求：配置为“Deny”   | string | 是                                       |
| Principal   | 被授权人，即指定本条桶策略描述所作用的用户，支持通配符“*”，表示所有用户。当对某个 user 进行授权时，Principal 格式为“AWS”: “arn:aws:s3:::user/userId”   | dict   | 否                                       |
| Action      | 操作，即指定本条桶策略描述所作用的 ZOS 操作。以列表形式表示，可配置多条操作，以逗号间隔。支持通配符“*”，表示该资源能进行的所有操作。常用的 Action 有“s3:GetObject”，<br>“s3:GetObjectAcl”，“s3:PutObject”，<br>“s3:PutObjectAcl”等                           | list   | 否                                       |
| Condition   | 条件语句，指定本条桶策略所限制的条件。可以通过 Condition 对 ZOS 资源设置防盗链，形如：<br>“Condition”:<br>{“StringEquals”:{“aws:Referer”:[“www.ctyun.com”]}}, 此时如果 Effect 为“Allow”，则允许来自“www.ctyun.com”的请求；如果为“Deny”，则拒绝。 | dict   | 否                                       |
| Resource    | 指定策略作用的一组资源，支持用通配符“*”表示所有资源  | string | 可选，<br>Resource<br>与<br>NotResource 选其一 |
| NotResource | 指定策略不作用的一组资源，策略将作用于除此之外的其他资源，取值同 Resource  | string | 可选，<br>Resource<br>与<br>NotResource 选其一 |

## 返回结果说明

```
{
  'ResponseMetadata': {
    'RequestId': 'tx000000000000000000e2-0060adb2d8-105a-default',
    'HostId': '',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'x-amz-request-id': 'tx000000000000000000e2-0060adb2d8-105a-default',
      'content-length': '0',
      'date': 'Wed, 26 May 2021 02:30:48 GMT',
      'connection': 'Keep-Alive'
    },
  },
  'RetryAttempts': 0
}
```

```
}  
}
```

| 返回结果             | 描述          | 类型   |
|------------------|-------------|------|
| ResponseMetadata | http 请求返回数据 | Dict |
| HTTPStatusCode   | http 请求状态码  | Int  |

## 示例

```
#-*-coding:utf-8 -*-  
  
from boto3.session import Session  
import boto3  
access_key="your access key"  
secret_key="your secretkey"  
url="zos endpoint"  
session=Session(access_key,secret_key)  
s3_client=session.client("s3",endpoint_url=url)  
print  
s3_client.put_bucket_policy(  
    Bucket="bucket-name",  
    Policy='{  
        "Version": "2012-10-17",  
        "Statement": [  
            {  
                "Sid": "id-1",  
                "Effect": "Allow",  
                "Principal": {"*"},  
                "Action": [ "s3:PutObject","s3:PutObjectAcl"],  
                "Resource": ["arn:aws:s3:::acl3/*"]  
            }  
        ]  
    }'  
)
```

## 1.7、Get Bucket Policy

### 功能说明

Get Bucket Policy 请求为获取设置在一个 bucket 上的策略

### 方法原型

```
get_bucket_policy(Bucket="bucket-name")
```

## 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

## 返回结果说明

```
{
  'ResponseMetadata': {
    'RequestId': 'tx00000000000000000000e3-0060adb4f8-105a-default',
    'HostId': '',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'x-amz-request-id': 'tx00000000000000000000e3-0060adb4f8-105a-
default',
      'content-type': 'application/json',
      'content-length': '182',
      'date': 'Wed, 26 May 2021 02:39:52 GMT',
      'connection': 'Keep-Alive'
    },
    'RetryAttempts': 0
  },
  'Policy': '{
    "Version": "2012-10-17",
    "Id": "S3PolicyId1",
    "Statement": [
      {
        "Sid": "IPAllow",
        "Effect": "Allow",
        "Principal": "*",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::bucket1/*"
      }
    ]
  }'
```

| 返回结果             | 描述                 | 类型   |
|------------------|--------------------|------|
| ResponseMetadata | http 请求返回数据        | Dict |
| HTTPStatusCode   | http 请求状态码         | Int  |
| Policy           | 返回 bucket 的 policy | Dict |

## 示例

```
##-*-coding:utf-8 -*-
from boto3.session import Session
```

```

import boto3
access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"

session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print
s3_client.get_bucket_policy(Bucket="bucket-name")

```

## 1.8、Delete Bucket Policy

### 功能说明

Delete Bucket Policy 请求为删除设置在某个 bucket 上的策略

### 方法原型

```
delete_bucket_policy(Bucket="bucket-name")
```

### 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

### 返回结果说明

```

{
  'ResponseMetadata': {
    'RequestId': 'tx000000000000000000e4-0060adb8ac-105a-default',
    'HostId': '',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'x-amz-request-id': 'tx000000000000000000e4-0060adb8ac-105a-
default',
      'content-length': '0',
      'date': 'Wed, 26 May 2021 02:55:40 GMT',
      'connection': 'Keep-Alive'
    },
    'RetryAttempts': 0
  }
}

```

| 返回结果 | 描述 | 类型 |
|------|----|----|
|------|----|----|



|                  |             |      |
|------------------|-------------|------|
| ResponseMetadata | http 请求返回数据 | Dict |
| HTTPStatusCode   | http 请求状态码  | Int  |

## 示例

```
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print
s3_client.delete_bucket_policy(Bucket="bucket-name")
```

## 1.9、Put Bucket ACL

### 功能说明

设置 Bucket 的 ACL，控制对 Bucket 的访问权限。该操作需要用户具有 WRITE\_ACP 权限。

有三种方式设置 ACL，三种方式不可同时使用，每次只能给一种参数赋值。其中，通过 ACL 参数方式进行操作，是设置预定义的固定的 ACL，不能针对特定用户进行授权，且该参数实现的效果，也可以借由另外两种方式实现，该参数使用请求头进行传递；AccessControlPolicy 参数方式和 Grant\* 参数方式则可以针对特定用户进行授权，AccessControlPolicy 方式通过请求体传递，而 Grant\* 方式通过请求头传递。三种方式都会覆盖原有 ACL 属性，包括桶所有者自身的权限，如需保留原有 ACL 属性，应将需要保留的原 ACL 添加到本次操作的授权中（ACL 参数方式会默认将桶所有者权限设为 FULL\_CONTROL，而另外两种方式则不会保留任何原 ACL 属性）。

### 方法原型

```
put_bucket_acl(
    Bucket='string',
    ACL='private'|'public-read'|'public-read-write'|'authenticated-
read',
    AccessControlPolicy={
        'Grants': [
            {
                'Grantee': {
                    'Type':
'CanonicalUser'|'AmazonCustomerByEmail'|'Group',
```

```

        'ID': 'string',
        'EmailAddress': 'string',
        'URI': 'string'
    },
    'Permission':
'FULL_CONTROL' | 'WRITE' | 'WRITE_ACP' | 'READ' | 'READ_ACP'
    },
],
'Owner': {
    'ID': 'string'
}
},
GrantFullControl='string',
GrantRead='string',
GrantReadACP='string',
GrantWrite='string',
GrantWriteACP='string',
)

```

## 参数说明

| 参数名称                | 参数描述   | 类型     | 是否必须                     |
|---------------------|--|--------|--------------------------|
| Bucket              | Bucket 的名称   | String | 是                        |
| ACL                 | 预定义的固定 ACL，<br>取值范围<br>'private'<br>'public-read'<br>'public-read-write'<br>'authenticated-read' | String | ACL 参数方式则必须，其他两种方式，则不能使用 |
| AccessControlPolicy | 包含多个授权列表和一个桶所有者参数  | dict   | 该方式下必须，其他两种方式下则不能使用      |
| Grants              | 授权列表   | list   | 该方式下必须                   |
| Grantee             | 被授权用户  | dict   | 该方式下必须                   |
| Type                | 被授权用户类型，<br>取值范围<br>'CanonicalUser'<br>'AmazonCustomerByEmail'<br>'Group'                        | String | 该方式下必须                   |

|                  |   |        |   |
|------------------|---|--------|---|
| ID(Grantee)      | 被授权用户 ID  | String | Type 为 'CanonicalUser', 则该字段必须            |
| EmailAddress     | 被授权用户邮箱   | String | 如果 Type 为 'AmazonCustomerByEmail', 则该字段必须 |
| URI              | 被授权组 URI<br>取值范围为<br>所有用户:<br>http://acs.amazonaws.com/groups/global/AllUsers<br>所有认证用户:<br>http://acs.amazonaws.com/groups/global/AuthenticatedUsers | String | 如果 Type 为 'Group', 则该字段必须                 |
| Permission       | 向被授权用户授予的权限,<br>取值范围<br>'FULL_CONTROL'<br>'WRITE'<br>'WRITE_ACP'<br>'READ'<br>'READ_ACP'  | String | 该方式下必须                                    |
| Owner            | Bucket 所有者  | dict   | 该方式下必须                                    |
| ID(Owner)        | Bucket 所有者 ID   | String | 该方式下必须                                    |
| GrantFullControl | 被授权用户可以对桶进行 read, write, read ACP, and write ACP 操作<br>以下 Grant* 参数, 格式都是 "id=xxxx"<br>或 "emailAddress=xxxx" 或者 "uri=xxxx" 以及他们的组合 (用逗号连接)            | String | 否   |
| GrantRead        | 被授权用户可以对桶进行读操作, 即 list object   | String | 否   |
| GrantWrite       | 被授权用户可以对桶进行写操作, 创建新的对象, 删除或覆盖写属于自己的对象   | String | 否   |
| GrantReadACP     | 被授权用户可以读取桶的 ACL   | String | 否   |
| GrantWriteACP    | 被授权用户可以修改桶的 ACL   | String | 否   |

## 返回结果说明

```

{
  'ResponseMetadata': {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx0000000000000000000001-0060b6f275-2cf25-default',
    'HTTPHeaders': {
      'date': 'Wed, 02 Jun 2021 02:52:37 GMT',
      'content-length': '0',
      'x-amz-request-id': 'tx0000000000000000000001-0060b6f275-2cf25-default',
      'content-type': 'application/xml',
      'connection': 'Keep-Alive'
    }
  }
}

```

| 返回结果             | 描述          | 类型   |
|------------------|-------------|------|
| ResponseMetadata | http 请求返回数据 | Dict |
| HTTPStatusCode   | http 请求状态码  | Int  |

## 示例

```

from boto3.session import Session
import boto3

access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url)

bucket_name = "bucket-1"

res = s3_client.put_bucket_acl(Bucket=bucket_name,
                              ACL='public-read-write')
print(res)

res = s3_client.put_bucket_acl(
    Bucket=bucket_name,
    AccessControlPolicy={
        'Grants': [
            {
                'Grantee': {
                    'Type': 'CanonicalUser',
                    'ID': 'test-1',
                },
                'Permission': 'FULL_CONTROL'
            },
            {
                'Grantee': {
                    'Type': 'CanonicalUser',

```

```

        'ID': 'test-2',
    },
    'Permission': 'READ'
},
{
    'Grantee': {
        'Type': 'AmazonCustomerByEmail',
        'EmailAddress': 'abc@abc.com',
    },
    'Permission': 'WRITE'
},
],
'Owner': {
    'ID': 'test-1'
}
}
)
print(res)

res = s3_client.put_bucket_acl(
    Bucket=bucket_name,
    GrantReadACP='id=test-4',
    GrantWriteACP='emailAddress=def@def.com')

print(res)

```

## 1.10、Get Bucket ACL

### 功能说明

Get Bucket ACL 接口用来获取 Bucket 的 ACL，即存储桶（Bucket）的访问权限控制列表。该操作需要 READ\_ACP 权限。该功能返回的结果与 Put Bucket ACL 参数一致，但是需要注意的是，如果以邮箱类型授权，返回结果中将会以对应被授权用户 ID 形式出现，即 Type 不会是 AmazonCustomerByEmail，而是 CanonicalUser。

### 方法原型

```
get_bucket_acl(Bucket='string')
```

### 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

### 返回结果说明

```

{
  'Owner': {
    'DisplayName': 'test-1',
    'ID': 'test-1'
  },
  'Grants': [
    {
      'Grantee': {
        'Type': 'Group',
        'URI': 'http://acs.amazonaws.com/groups/global/AllUsers'
      },
      'Permission': 'READ'
    },
    {
      'Grantee': {
        'EmailAddress': '',
        'Type': 'CanonicalUser',
        'DisplayName': 'test-1',
        'ID': 'test-1'
      },
      'Permission': 'FULL_CONTROL'
    }
  ],
  'ResponseMetadata': {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx0000000000000000000005-0060b6f276-2cf25-default',
    'HTTPHeaders': {
      'date': 'Wed, 02 Jun 2021 02:52:38 GMT',
      'content-length': '843',
      'x-amz-request-id': 'tx0000000000000000000005-0060b6f276-2cf25-default',
      'content-type': 'application/xml',
      'connection': 'Keep-Alive'
    }
  }
}

```

| 返回结果                 | 描述               | 类型     |
|----------------------|------------------|--------|
| Owner                | Bucket 所有者       | Dict   |
| DisplayName(Owner)   | Bucket 所有者的展示名   | String |
| ID(Owner)            | Bucket 所有者的用户 ID | String |
| Grants               | 授权列表             | list   |
| Grantee              | 被授权用户            | dict   |
| Type                 | 被授权用户类型          | string |
| URI                  | 被授权组 URI         | string |
| ID(Grantee)          | 被授权用户 ID         | string |
| DisplayName(Grantee) | 被授权用户展示名         | string |
| EmailAddress         | 被授权用户邮箱          | string |

|            |       |        |
|------------|-------|--------|
| Permission | 被授权权限 | string |
|------------|-------|--------|

## 示例

```
from boto3.session import Session
import boto3

access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url)

bucket_name = "bucket-1"
res = s3_client.get_bucket_acl(Bucket=bucket_name)

print(res)
```

## 1.11、Put Bucket Lifecycle Configuration

### 功能说明

Put Bucket Lifecycle Configuration 接口用来写入 Bucket 的生命周期规则。

### 方法原型

```
put_bucket_lifecycle_configuration(
    Bucket='string',
    LifecycleConfiguration={
        'Rules': [
            {
                'Expiration': {
                    'Date': datetime(2015, 1, 1),
                    'Days': 123,
                },
                'ID': 'string',
                'Prefix': 'string',
                'Filter': {
                    'Prefix': 'string',
                    'Tag': {
                        'Key': 'string',
                        'Value': 'string'
                    },
                },
                'And': {
                    'Prefix': 'string',
                    'Tags': [
                        {
                            'Key': 'string',
                            'Value': 'string'
                        }
                    ]
                }
            }
        ]
    }
)
```

```

    },
  ],
},
'Status': 'Enabled'|'Disabled',
'Transitions': [
  {
    'Date': datetime(2015, 1, 1),
    'Days': 123,
    'StorageClass':
'GLACIER'|'STANDARD_IA'|'STANDARD'
  },
],
'NoncurrentVersionTransitions': [
  {
    'NoncurrentDays': 123,
    'StorageClass':
'GLACIER'|'STANDARD_IA'|'STANDARD'
  },
],
'NoncurrentVersionExpiration': {
  'NoncurrentDays': 123
},
'AbortIncompleteMultipartUpload': {
  'DaysAfterInitiation': 123
}
},
]
}
)

```

## 参数说明

| 参数名称                   | 参数描述                                    | 类型       | 是否必须                |
|------------------------|---|----------|---------------------|
| ID                     | 标识唯一的规则                                 | string   | 否，建议指定，如不指定，自动生成 ID |
| Bucket                 | 指定存储桶名称                                 | string   | 是                   |
| LifecycleConfiguration | 生命周期规则的容器                               | dict     | 是                   |
| Rules                  | 生命周期规则                                  | list     | 是                   |
| Expiration             | 用日期或天数指定对象的过期时间                         | dict     | 否                   |
| Date                   | 标识对象的过期日期,日期为 ISO8601 格式，必须为 UTC 午夜 0 时 | datetime | Date 与 Days 二选一     |
| Days                   | 标识对象受规则约束的天数                            | integer  | Date 与 Days 二选一     |
| Filter                 | 过滤应用规则的对象                               | dict     | 否                   |
| Prefix                 | 标识应用规则的对象前缀                             | string   | 是                   |
| Tag                    | 应用规则到拥有指定标签的对象                          | dict     | 否                   |



|                                |  |         |                |
|--------------------------------|--|---------|----------------|
| Key                            | 标签的名称  | string  | 否              |
| Value                          | 标签的值   | string  | 否              |
| Status                         | 标识是否应用规则，可选值：Enabled, Disabled               | string  | 是              |
| Transtions                     | 标识对象何时转存到指定的 Storage Class                   | list    | 否              |
| StorageClass                   | 标识要转存储到哪种存储类别,如 STANDARD/STANDARD_IA/GLACIER | string  | 如设置转存储规则，该字段必选 |
| NoncurrentVersionTransitions   | 标识历史版本的转存储规则                                 | list    | 否              |
| NoncurrentDays                 | 标识对象的历史版本受规则约束的天数                            | integer | 否              |
| NoncurrentVersionExpiration    | 标识历史版本的过期规则                                  | dict    | 否              |
| AbortIncompleteMultipartUpload | 标识清除未完成的分段上传                                 | dict    | 否              |
| DaysAfterInitiation            | 标识一次分段上传最多持续天数                               | integer | 否              |

## 返回结果说明

```
{
  "ResponseMetadata": {
    "HTTPStatusCode": 200,
    "RetryAttempts": 0,
    "HostId": "",
    "RequestId": "tx0000000000000000000071-0060b9e5aa-aca5-default",
    "HTTPHeaders": {
      "date": "Fri, 04 Jun 2021 08:34:50 GMT",
      "content-length": "0",
      "x-amz-request-id": "tx0000000000000000000071-0060b9e5aa-aca5-default",
      "content-type": "application/xml"
    }
  }
}
```

## 示例

```
##-*-coding:utf-8 -*-
from boto3.session import Session
access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url, verify=False)
bucket_name = 'public'
```

```

res = s3_client.put_bucket_lifecycle_configuration(
    Bucket=bucket_name,
    LifecycleConfiguration={
        'Rules': [
            {
                'Expiration': {
                    'Days': 365
                },
                'ID': 'Lifecycle Test',
                'Status': 'Enabled',
                'Filter': {
                    'And': {
                        'Prefix': 'test',
                        'Tags': [
                            {
                                'Key': 'tag-1',
                                'Value': 'val-1'
                            }
                        ]
                    }
                }
            }
        ]
    }
)

```

## 1.12、Get Bucket Lifecycle Configuration

### 功能说明

Get Bucket Lifecycle Configuration 接口用来获取 Bucket 的生命周期规则。

### 方法原型

```

get_bucket_lifecycle_configuration(
    Bucket='string',
)

```

### 参数说明

| 参数名称   | 参数描述    | 类型     | 是否必须 |
|--------|---------|--------|------|
| Bucket | 指定存储桶名称 | String | 是    |

### 返回结果说明

```

{
  'Rules': [
    {
      'Expiration': {
        'Date': datetime(2015, 1, 1),
        'Days': 123,
        'ExpiredObjectDeleteMarker': True|False
      },
      'ID': 'string',
      'Prefix': 'string',
      'Filter': {
        'Prefix': 'string',
        'Tag': {
          'Key': 'string',
          'Value': 'string'
        },
        'And': {
          'Prefix': 'string',
          'Tags': [
            {
              'Key': 'string',
              'Value': 'string'
            }
          ],
        }
      },
      'Status': 'Enabled'|'Disabled',
      'Transitions': [
        {
          'Date': datetime(2015, 1, 1),
          'Days': 123,
          'StorageClass': 'GLACIER'|'STANDARD_IA'|'STANDARD'
        },
      ],
      'NoncurrentVersionTransitions': [
        {
          'NoncurrentDays': 123,
          'StorageClass': 'GLACIER'|'STANDARD_IA'|'STANDARD'
        },
      ],
      'NoncurrentVersionExpiration': {
        'NoncurrentDays': 123
      },
      'AbortIncompleteMultipartUpload': {
        'DaysAfterInitiation': 123
      }
    },
  ],
  "ResponseMetadata": {
    "HTTPStatusCode": 200,
    "RetryAttempts": 0,
    "HostId": "",
    "RequestId": "tx0000000000000000000073-0060b9ec59-aca5-default",
    "HTTPHeaders": {
      "date": "Fri, 04 Jun 2021 09:03:21 GMT",
    }
  }
}

```

```

        "content-length": "332",
        "x-amz-request-id": "tx00000000000000000073-0060b9ec59-
aca5-default",
        "content-type": "application/xml"
    }
}
}

```

| 返回结果                           | 描述   | 类型       |
|--------------------------------|--|----------|
| Rules                          | 生命周期规则   | list     |
| Expiration                     | 对象的过期规则  | dict     |
| Date                           | 对象的过期日期  | datetime |
| Days                           | 对象受规则约束的天数                                       | integer  |
| ID                             | 标识唯一的规则  | string   |
| Filter                         | 过滤应用规则的对象  | dict     |
| Prefix                         | 标识应用规则的对象前缀                                      | string   |
| Tag                            | 应用规则到拥有指定标签的对象                                   | dict     |
| Key                            | 标签的名称  | string   |
| Value                          | 标签的值   | string   |
| Status                         | 标识是否应用规则，可选值: Enabled, Disabled                  | string   |
| Transitions                    | 标识对象何时转存到指定的 Storage Class                       | list     |
| StorageClass                   | 标识要转存储到哪种存储类别,如 STANDARD/STANDARD_I<br>A/GLACIER | string   |
| NoncurrentVersionTransitions   | 标识历史版本的转存储规则                                     | list     |
| NoncurrentDays                 | 标识对象的历史版本受规则约束的天数                                | integer  |
| NoncurrentVersionExpiration    | 标识历史版本的过期规则                                      | dict     |
| AbortIncompleteMultipartUpload | 标识清除未完成的分段上传                                     | dict     |
| DaysAfterInitiation            | 标识一次分段上传最多持续天数                                   | integer  |
| ResponseMetadata               | http 请求返回数据                                      | Dict     |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url, verify=False)
bucket_name = 'public'

```

```
res = s3_client.get_bucket_lifecycle_configuration(  
    Bucket=bucket_name  
)  
print res
```

## 1.13、Delete Bucket Lifecycle

### 功能说明

Delete Bucket Lifecycle 接口用来删除 Bucket 的生命周期规则。

### 方法原型

```
delete_bucket_lifecycle(Bucket='string')
```

### 参数说明

| 参数名称   | 参数描述    | 类型     | 是否必须 |
|--------|---------|--------|------|
| Bucket | 指定存储桶名称 | String | 是    |

### 返回结果说明

```
{  
  "ResponseMetadata": {  
    "HTTPStatusCode": 204,  
    "RetryAttempts": 0,  
    "HostId": "",  
    "RequestId": "tx000000000000000000091-0060b9f573-aca5-default",  
    "HTTPHeaders": {  
      "date": "Fri, 04 Jun 2021 09:42:11 GMT",  
      "x-amz-request-id": "tx000000000000000000091-0060b9f573-aca5-default",  
      "content-type": "application/xml"  
    }  
  }  
}
```

| 返回结果             | 描述          | 类型   |
|------------------|-------------|------|
| ResponseMetadata | http 请求返回数据 | Dict |

### 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session

access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url, verify=False)
bucket_name = 'public'

res = s3_client.delete_bucket_lifecycle(
    Bucket=bucket_name
)
print res

```

## 1.14、Put Bucket Website

### 功能说明

调用 PutBucketWebsite 接口将存储空间（Bucket）设置成静态网站托管模式并设置跳转规则（RoutingRule）

### 方法原型

```

put_bucket_website(
    Bucket='string',
    WebsiteConfiguration={
        'ErrorDocument': {
            'Key': 'string'
        },
        'IndexDocument': {
            'Suffix': 'string'
        },
        'RedirectAllRequestsTo': {
            'HostName': 'string',
            'Protocol': 'http'|'https'
        },
        'RoutingRules': [
            {
                'Condition': {
                    'HttpErrorCodeReturnedEquals': 'string',
                    'KeyPrefixEquals': 'string'
                },
                'Redirect': {
                    'HostName': 'string',
                    'HttpRedirectCode': 'string',
                    'Protocol': 'http'|'https',
                    'ReplaceKeyPrefixWith': 'string',
                    'ReplaceKeyWith': 'string'
                }
            }
        ],
    ]
)

```

```
}  
)
```

## 参数说明

| 参数名称                         | 参数描述  | 类型     | 是否必须   |
|------------------------------|---|--------|--|
| Bucket                       | bucket 名称   | string | 是  |
| WebsiteConfiguration         | 请求的配置信息的容器  | dict   | 是  |
| ErrorDocument                | 错误文档配置  | dict   | 否  |
| Key                          | 指定通用错误文档的对象键，当发生错误且未命中重定向规则中的错误码重定向时，将返回该对象键的内容   | string | 是  |
| IndexDocument                | 索引文档配置  | dict   | 是  |
| Suffix                       | 指定索引文档的对象键后缀。例如指定为 <code>index.html</code> ，那么当访问到存储桶的根目录时，会自动返回 <code>index.html</code> 的内容，或者当访问到 <code>article/</code> 目录时，会自动返回 <code>article/index.html</code> 的内容 | string | 是  |
| RedirectAllRequestsTo        | 重定向所有请求配置，该规则与其他规则互斥，也就是说使用了重定向规则就不能配置其他规则。   | dict   | 否  |
| HostName                     | 要重定向的主机名  | string | 是  |
| Protocol                     | 重定向时使用的协议，默认使用原请求的协议  | string | 否  |
| RoutingRules                 | 重定向规则配置   | dict   | 否  |
| Condition                    | 重定向规则的条件配置  | dict   | 否  |
| HttpErrorCodeReturned Equals | 指定重定向规则的错误码匹配条件，只支持配置 4XX 返回码，例如 403 或 404  | string | 当 condition 配置后，HttpErrorCodeReturned Equals 和 KeyPrefixEquals 两者只能配置一个。 |
| KeyPrefixEquals              | 指定重定向规则的对象键前缀匹配条件   | string | 当 condition 配置后，HttpErrorCodeReturned Equals 和 KeyPrefixEq               |

|                      |   |        |  |
|----------------------|---|--------|--|
|                      |   |        | uals 两者只能配置一个。                             |
| Redirect             | 重定向格则容器，可以配置规则重定向其他主机、页面或其他协议，当发生错误时，也可以配置错误码。                                | dict   | RoutingRules 中的必要配置。                       |
| HostName             | 重定向机器名  | string | 否  |
| HttpRedirectCode     | http 返回码规则，   | string | 否  |
| Protocol             | 重定向请求要用的协议，默认使用原请求所是应用的协议。  | string | 否  |
| ReplaceKeyPrefixWith | 指定重定向规则的具体重定向目标的对象键，替换方式为替换原始请求中所匹配到的前缀部分，仅可在 Condition 为 KeyPrefixEquals 时设置 | string | ReplaceKeyWith 与 ReplaceKeyPrefixWith 必选其一 |
| ReplaceKeyWith       | 指定重定向规则的具体重定向目标的对象键，替换方式为替换整个原始请求的对象键   | string | ReplaceKeyWith 与 ReplaceKeyPrefixWith 必选其一 |

## 返回结果说明

```
{
  'ResponseMetadata': {
    '...': '...',
  },
}
```

| 返回结果             | 描述                                  | 类型   |
|------------------|-------------------------------------|------|
| ResponseMetadata | http 请求返回数据，包括请求返回状态码及 RequestID 等等 | Dict |

## 示例

```
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
response = s3_client.put_bucket_website(
    Bucket='examplebucket',
    ContentMD5='',
    WebsiteConfiguration={
```



```
    'ErrorDocument': {
      'Key': 'error.html',
    },
    'IndexDocument': {
      'Suffix': 'index.html',
    },
  },
)
print(response)
```

## 1.15、Get Bucket Website

### 功能说明

GET Bucket website 请求用于查询与存储桶关联的静态网站配置信息。

### 方法原型

```
client.get_bucket_website(Bucket='string')
```

### 参数说明

| 参数名称   | 参数描述  | 类型     | 是否必须 |
|--------|-------|--------|------|
| Bucket | 存储桶名称 | string | 是    |

### 返回结果说明

```
{
  'RedirectAllRequestsTo': {
    'HostName': 'string',
    'Protocol': 'http'|'https'
  },
  'IndexDocument': {
    'Suffix': 'string'
  },
  'ErrorDocument': {
    'Key': 'string'
  },
  'RoutingRules': [
    {
      'Condition': {
        'HttpErrorCodeReturnedEquals': 'string',
```

```

        'KeyPrefixEquals': 'string'
    },
    'Redirect': {
        'HostName': 'string',
        'HttpRedirectCode': 'string',
        'Protocol': 'http|https',
        'ReplaceKeyPrefixWith': 'string',
        'ReplaceKeyWith': 'string'
    }
},
]]}

```

| 返回结果                        | 描述  | 类型     |
|-----------------------------|---|--------|
| ErrorDocument               | 错误文档配置容器  | dict   |
| Key                         | 指定通用错误文档的对象键，当发生错误且未命中重定向规则中的错误码重定向时，将返回该对象键的内容   | string |
| IndexDocument               | 索引文档配置容器  | dict   |
| Suffix                      | 指定索引文档的对象键后缀。例如指定为 <code>index.html</code> ，那么当访问到存储桶的根目录时，会自动返回 <code>index.html</code> 的内容，或者当访问到 <code>article/</code> 目录时，会自动返回 <code>article/index.html</code> 的内容 | string |
| RedirectAllRequestsTo       | 重定向所有请求配置容器，该规则与其他规则互斥，也就是说使用了重定向规则就不能配置其他规则。   | dict   |
| HostName                    | 要重定向的主机名  | string |
| Protocol                    | 重定向时使用的协议，默认使用原请求的协议  | string |
| RoutingRules                | 重定向规则配置容器   | dict   |
| Condition                   | 重定向规则的条件配置  | dict   |
| HttpErrorCodeReturnedEquals | 指定重定向规则的错误码匹配条件，只支持配置 4XX 返回码，例如 403 或 404  | string |
| KeyPrefixEquals             | 指定重定向规则的对象键前缀匹配条件   | string |
| Redirect                    | 重定向规则容器，可以配置规则重定向其他主机、页面或其他协议，当发生错误时，也可以配置错误码。  | dict   |
| HostName                    | 重定向机器名  | string |
| HttpRedirectCode            | http 返回码规则，   | string |
| Protocol                    | 重定向请求要用的协议，默认使用原请求所是应用的协议。  | string |
| ReplaceKeyPrefixWith        | 指定重定向规则的具体重定向目标的对象键，替换方式为替换原始请求中所匹配到的前缀部分，仅可在 Condition 为 KeyPrefixEquals 时设置   | string |
| ReplaceKeyWith              | 指定重定向规则的具体重定向目标的对象键，替换方式为替换整个原始请求的对象键   | string |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session

```

```
import boto3
access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
response = s3_client.get_bucket_website(
    Bucket='examplebucket',
)
print(response)
```

## 1.16、Delete Bucket Website

### 功能说明

DELETE Bucket website 请求用于删除存储桶中的静态网站配置。

### 方法原型

```
client.delete_bucket_website(Bucket='string')
```

### 参数说明

| 参数名称   | 参数描述            | 类型     | 是否必须 |
|--------|-----------------|--------|------|
| Bucket | 要删除静态网站配置的存储桶名称 | string | 是    |

### 返回结果说明

```
{
  'ResponseMetadata': {
    '...': '...',
  },
}
```

| 返回结果             | 描述                                  | 类型   |
|------------------|-------------------------------------|------|
| ResponseMetadata | http 请求返回数据，包括请求返回状态码及 RequestID 等等 | Dict |

### 示例

```
#!/usr/bin/env python
#-*-coding:utf-8 -*-
from boto3.session import Session
```

```

import boto3
access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
response = s3_client.delete_bucket_website(
    Bucket='examplebucket',
)
print(response)

```

## 1.17、Put Bucket Tagging

### 功能说明

为指定的 Bucket 设置标签。一个 Bucket 最多设置 50 个标签。该操作需要 s3:PutBucketTagging 权限，桶的所有者默认拥有该权限。该操作会覆盖原有标签。

### 方法原型

```

put_bucket_tagging(
    Bucket='string',
    Tagging={
        'TagSet': [
            {
                'Key': 'string',
                'Value': 'string'
            },
        ]
    })

```

### 参数说明

| 参数名称    | 参数描述   | 类型     | 是否必须 |
|---------|--|--------|------|
| Bucket  | Bucket 的名称   | String | 是    |
| Tagging | 标签集容器，内部为标签列表，最多 50 个标签，每个标签都是键值对，key 最大 128 字节，value 最大 256 字节，value 可以为空，key 和 value 均为 utf-8 编码 | dict   | 是    |

### 返回结果说明

```

{
    'ResponseMetadata': {
        'HTTPStatusCode': 200,

```

```

'RetryAttempts': 0,
'HostId': '',
'RequestId': 'tx000000000000000000002-0060aeeb68-28109-default',
'HTTPHeaders': {
  'date': 'Thu, 27 May 2021 00:44:24 GMT',
  'content-length': '0',
  'x-amz-request-id': 'tx000000000000000000002-0060aeeb68-28109-
default',
  'content-type': 'application/xml',
  'connection': 'Keep-Alive'
}
}
}

```

## 示例

```

#-*-coding:utf-8 -*-

from boto3.session import Session
import boto3

access_key="your access key"
secret_key="your secretkey"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url)

bucket_name = "bucket-1"

response = s3_client.put_bucket_tagging(
    Bucket=bucket_name,
    Tagging={
        'TagSet': [
            {
                'Key': 'key-1',
                'Value': 'val-1'
            }
        ]
    }
)

print(response)

```

## 1.18、Get Bucket Tagging

### 功能说明

获取指定 BUCKET 的标签。该操作需要 s3:GetBucketTagging 权限，桶的拥有者默认具有该权限。

### 方法原型

```
get_bucket_tagging(Bucket='string')
```

## 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

## 返回结果说明

```
{
  u'TagSet': [
    {
      u'Value': 'val-1',
      u'Key': 'key-1'
    }
  ],
  'ResponseMetadata': {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx00000000000000000003-0060af0c4b-28109-default',
    'HTTPHeaders': {
      'date': 'Thu, 27 May 2021 03:04:43 GMT',
      'content-length': '169',
      'x-amz-request-id': 'tx00000000000000000003-0060af0c4b-28109-default',
      'content-type': 'application/xml',
      'connection': 'Keep-Alive'
    }
  }
}
```

| 返回结果   | 描述       | 类型     |
|--------|----------|--------|
| TagSet | 标签集      | list   |
| Key    | 标签 key   | string |
| Value  | 标签 value | string |

## 示例

```
## -*- coding: utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
```

```
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url)
bucket_name = "bucket-1"
response = s3_client.get_bucket_tagging(Bucket=bucket_name)

print(response)
```

## 1.19、Delete Bucket Tagging

### 功能说明

删除 Bucket 上的标签。该操作需要 s3:PutBucketTagging 权限，桶的拥有者默认具有该权限。

### 方法原型

```
delete_bucket_tagging(Bucket='string')
```

### 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

### 返回结果说明

```
{
  'ResponseMetadata': {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx00000000000000000004-0060af10ee-28109-default',
    'HTTPHeaders': {
      'date': 'Thu, 27 May 2021 03:24:30 GMT',
      'content-length': '0',
      'x-amz-request-id': 'tx00000000000000000004-0060af10ee-28109-
default',
      'content-type': 'application/xml',
      'connection': 'Keep-Alive'
    }
  }
}
```

### 示例

```

#-*-coding:utf-8 -*-

from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url)
bucket_name = "bucket-1"
response = s3_client.delete_bucket_tagging(Bucket=bucket_name)

print(response)

```

## 1.20、Put Bucket Encryption

### 功能说明

put bucket encryption 请求可以启用存储桶默认加密功能

### 方法原型

```

put_bucket_encryption(
    Bucket="bucket-name",
    ServerSideEncryptionConfiguration={
        'Rules': [
            {
                'ApplyServerSideEncryptionByDefault': {
                    'SSEAlgorithm': 'AES256'|'aws:kms',
                    'KMSEMasterKeyID': 'string'
                },
            },
        ]
    },
)

```

### 参数说明

| 参数名称                               | 参数描述   | 类型     | 是否必须 |
|------------------------------------|--|--------|------|
| Bucket                             | Bucket 的名称   | String | 是    |
| ServerSideEncryptionConfiguration  | 指定默认服务端加密配置  | Dict   | 是    |
| Rules                              | 一个特殊的服务端加密配置规则信息   | list   | 是    |
| ApplyServerSideEncryptionByDefault | 指定默认的服务端加密会应用于新对象上传至存储桶时。若是在上传对象时请求中未指定任何加密信息，则存储桶默认加密将会应用 | Dict   | 否    |



|                |   |        |   |
|----------------|---|--------|---|
| SSEAlgorithm   | 服务端加密算法   | string | 是 |
| KMSMasterKeyID | 若加密算法选用的是 aws:kms, 则此项必填, 按照 cmkuuid:keyspec:userid 模式配置, 其中 cmkuuid 是 CMKID, keyspec 是指定生成的数据密钥长度, userid 是用户 id; 若是 AES256 算法, 则此项可不填, 若填, 则字符长度需为 32 | string | 否 |

## 返回结果说明

```
{
  'ResponseMetadata': {
    'RequestId': 'tx00000000000000000001-005fc8d74e-5e67-default',
    'HostId': '',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'date': 'Thu, 03 Dec 2020 12:17:18 GMT',
      'content-length': '0',
      'connection': 'Keep-Alive',
      'x-amz-request-id': 'tx00000000000000000001-005fc8d74e-5e67-
default'
    },
    'RetryAttempts': 0
  }
}
```

| 返回结果             | 描述                                   | 类型   |
|------------------|--------------------------------------|------|
| ResponseMetadata | http 请求返回数据, 包括请求返回状态码及 RequestID 等等 | Dict |

## 示例

```
#!/usr/bin/env python
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.put_bucket_encryption(
    Bucket='bucket1',
    ServerSideEncryptionConfiguration={
        'Rules': [
            {
                'ApplyServerSideEncryptionByDefault': {
                    'SSEAlgorithm': 'aws:kms',
                    'KMSMasterKeyID': '6b1f657c-816b-4534-a41a-
903e7a60e703:AES_256:e3d16fba6ae84e33a1d386dd880696c0'
                }
            }
        ],
    },
)
```

```
) } ]
```

## 1.21、Get Bucket Encryption

### 功能说明

get bucket encryption 请求可以返回存储桶默认加密配置。若是存储桶不存在默认加密配置，则返回 NoSuchEncryptionSetError 错误。

### 方法原型

```
get_bucket_encryption(Bucket="bucket-name")
```

### 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

### 返回结果说明

| 返回结果                               | 描述   | 类型     |
|------------------------------------|--|--------|
| ServerSideEncryptionConfiguration  | 默认的服务端加密配置   | dict   |
| Rules                              | 一个特殊的服务端加密配置规则信息   | list   |
| ApplyServerSideEncryptionByDefault | 指定默认的服务端加密会应用于新对象上传至存储桶时。若是在上传对象时请求中未指定任何加密信息，则存储桶默认加密将会应用 | dict   |
| SSEAlgorithm                       | 服务端加密算法  | string |
| ResponseMetadata                   | http 请求返回数据，包括请求返回状态码及 RequestID 等等                        | Dict   |

### 示例

```
#!/usr/bin/env python
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
```

```
print s3_client.get_bucket_encryption(Bucket='bucket1')
```

## 1.22、Delete Bucket Encrytion

### 功能说明

delete bucket encryption 请求删除存储桶默认加密配置

### 方法原型

```
delete_bucket_encryption(Bucket="bucket-name")
```

### 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

### 返回结果说明

```
{
  'ResponseMetadata': {
    'RequestId': 'tx00000000000000000001-005fc8d74e-5e67-default',
    'HostId': '',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'date': 'Thu, 03 Dec 2020 12:17:18 GMT',
      'content-length': '0',
      'connection': 'Keep-Alive',
      'x-amz-request-id': 'tx00000000000000000001-005fc8d74e-5e67-
default'
    },
    'RetryAttempts': 0
  }
}
```

| 返回结果             | 描述                                   | 类型   |
|------------------|--------------------------------------|------|
| ResponseMetadata | http 请求返回数据, 包括请求返回状态码及 RequestID 等等 | Dict |

## 示例

```
#!/usr/bin/env python
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.delete_bucket_encryption(Bucket='bucket1')
```

## 1.23、Put Bucket Object Lock Configuration

### 功能说明

put bucket object lock 请求在指定的存储桶上增加对象锁定配置。默认规则将会应用到每一个新放入桶中的对象。

### 方法原型

```
put_object_lock_configuration(
    Bucket="bucket-name",
    ObjectLockConfiguration={
        'ObjectLockEnabled': 'Enabled',
        'Rule': {
            'DefaultRetention': {
                'Mode': 'GOVERNANCE' | 'COMPLIANCE',
                'Days': 123,
                'Years': 123
            }
        }
    }
)
```

### 参数说明

| 参数名称                    | 参数描述                                       | 类型     | 是否必须 |
|-------------------------|--|--------|------|
| Bucket                  | Bucket 的名称                                 | String | 是    |
| ObjectLockConfiguration | 应用到指定存储桶的对象锁定配置                            | dict   | 否    |
| ObjectLockEnabled       | 表示指定桶的对象锁定功能是否生效。当对存储桶设置了对象锁定配置，即为生效       | string | 否    |
| Rule                    | 指定对象的对象锁定规则，配置需要指定模式和时间。年或日只能指定一个，不能在配置中既指 | dict   | 否    |

|                  | 定年又指定日  |         |   |
|------------------|---|---------|---|
| DefaultRetention | 对象锁定规则中指定的默认模式和时间。存储桶配置同时需要模式和时间，年和日不能同时指定，只能指定一个 | dict    | 否 |
| Mode             | 存储桶默认的对象锁定保留期限模式                                  | string  | 否 |
| Days             | 保留期限日期，单位天。与年数设置只能二选其一                            | integer | 否 |
| Years            | 保留期限日期，单位年。与天数设置只能二选其一                            | integer | 否 |

## 返回结果说明

```
{
  'ResponseMetadata': {
    'RequestId': 'tx0000000000000000000001-005fc8d74e-5e67-default',
    'HostId': '',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'date': 'Thu, 03 Dec 2020 12:17:18 GMT',
      'content-length': '0',
      'connection': 'Keep-Alive',
      'x-amz-request-id': 'tx0000000000000000000001-005fc8d74e-5e67-
default'
    },
    'RetryAttempts': 0
  }
}
```

| 返回结果             | 描述                                  | 类型   |
|------------------|-------------------------------------|------|
| ResponseMetadata | http 请求返回数据，包括请求返回状态码及 RequestID 等等 | Dict |

## 示例

```
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.put_bucket_encryption(
    Bucket='bucket1',
    ObjectLockConfiguration={
        'ObjectLockEnabled': 'Enabled',
        'Rule': {
            'DefaultRetention': {
                'Mode': 'GOVERNANCE',
                'Days': 1,
            }
        }
    }
)
```

```
    }  
  }  
)
```

## 1.24、Get Bucket Object Lock Configuration

### 功能说明

get bucket object lock 请求获取存储桶的对象锁定配置。默认的对象锁定功能将会应用到每一个新放入到存储桶中的对象。

### 方法原型

```
get_object_lock_configuration(Bucket="bucket-name")
```

### 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

### 返回结果说明

```
{  
  'ObjectLockConfiguration': {  
    'ObjectLockEnabled': 'Enabled',  
    'Rule': {  
      'DefaultRetention': {  
        'Mode': 'GOVERNANCE',  
        'Days': 1  
      }  
    }  
  }  
  'ResponseMetadata': {  
    'RequestId': 'tx00000000000000000001-005fc8d74e-5e67-default',  
    'HostId': '',  
    'HTTPStatusCode': 200,  
    'HTTPHeaders': {  
      'date': 'Thu, 03 Dec 2020 12:17:18 GMT',  
      'content-length': '0',  
      'connection': 'Keep-Alive',  
      'x-amz-request-id': 'tx00000000000000000001-005fc8d74e-5e67-  
default'  
    },  
  },  
}
```

```

    'RetryAttempts': 0
  }
}

```

| 返回结果                    | 描述  | 类型      |
|-------------------------|---|---------|
| ObjectLockConfiguration | 应用到指定存储桶的对象锁定配置                                   | dict    |
| ObjectLockEnabled       | 表示指定桶的对象锁定功能是否生效。当对存储桶设置了对象锁定配置，即为生效              | string  |
| Rule                    | 指定对象的对象锁定规则，配置需要指定模式和时间。年或日只能指定一个，不能在配置中既指定年又指定日  | dict    |
| DefaultRetention        | 对象锁定规则中指定的默认模式和时间。存储桶配置同时需要模式和时间，年和日不能同时指定，只能指定一个 | dict    |
| Mode                    | 存储桶默认的对象锁定保留期限模式                                  | string  |
| Days                    | 保留期限日期，天  | integer |
| Years                   | 保留期限日期，年  | integer |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.get_object_lock_configuration(Bucket='bucket1')

```

## 1.25、Put Bucket Logging

### 功能说明

put bucket logging 请求设置日志转存参数。所有的日志将会保留到和源存储桶属于同一拥有者的目标存储桶中。桶的拥有者可以设置桶的日志状态。桶的拥有者对所有的日志具有 FULL\_CONTROL 权限，可以通过 Grantee 授权其他用户，其中 Permissions 参数指定了用户对日志的访问权限。

### 方法原型

```

put_bucket_logging(
    Bucket="bucket-name",
    BucketLoggingStatus={
        'LoggingEnabled': {
            'TargetBucket': 'string',
            'TargetGrants': [
                {

```

```

        'Grantee': {
            'DisplayName': 'string',
            'EmailAddress': 'string',
            'ID': 'string',
            'Type':
'CanonicalUser'|'AmazonCustomerByEmail'|'Group',
            'URI': 'string'
        },
        'Permission': 'FULL_CONTROL'|'READ'|'WRITE'
    },
    ],
    'TargetPrefix': 'string'
}
)

```

## 参数说明

| 参数名称                | 参数描述  | 类型      | 是否必须 |
|---------------------|---|---------|------|
| Bucket              | Bucket 的名称  | String  | 是    |
| BucketLoggingStatus | 日志状态信息。若此参数置为空，则表示关闭日志转存功能  | dict    | 是    |
| LoggingEnabled      | 描述日志存储位置和日志对象前缀   | dict    | 否    |
| TargetBucket        | 日志存储位置。可以将日志存储到任意用户拥有的桶中，包含源存储桶。用户可以配置多个源桶的日志均投放到同一个目标存储桶中，在这种情况下，用户可以使用 TargetPrefix 区分日志来自哪个源存储桶。 | string  | 是    |
| TargetGrants        | 授权信息  | list    | 否    |
| Grantee             | 授权许可  | dict    | 否    |
| DisplayName         | 展示名字  | string  | 否    |
| EmailAddress        | 邮件地址  | integer | 否    |
| ID                  | 授权用户 ID   | string  | 否    |
| Type                | 授权类型  | string  | 是    |
| URI                 | 授权组 URI   | string  | 否    |
| Permission          | 日志访问许可  | string  | 否    |
| TargetPrefix        | 日志对象前缀。若多个源存储桶的日志均写到同一个目标存储桶中，则可以通过目标前缀来区分日志来自哪一个源存储桶   | string  | 是    |

## 返回结果说明

```

{
  'ResponseMetadata': {
    'RequestId': 'tx00000000000000000001-005fc8d74e-5e67-default',

```



```

    'HostId': '',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'date': 'Thu, 03 Dec 2020 12:17:18 GMT',
      'content-length': '0',
      'connection': 'Keep-Alive',
      'x-amz-request-id': 'tx0000000000000000000001-005fc8d74e-5e67-
default'
    },
    'RetryAttempts': 0
  }
}

```

| 返回结果             | 描述                                   | 类型   |
|------------------|--------------------------------------|------|
| ResponseMetadata | http 请求返回数据, 包括请求返回状态码及 RequestID 等等 | Dict |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.put_bucket_logging(
    Bucket='bucket1',
    BucketLoggingStatus={
        'LoggingEnabled': {
            'TargetBucket': 'bucket6',
            'TargetGrants': [
                {
                    'Grantee': {
                        'DisplayName': 'Second User',
                        'ID': 's3',
                        'Type': 'CanonicalUser',
                    },
                    'Permission': 'READ'
                },
            ],
            'TargetPrefix': 'log/',
        },
    },
)

```

## 1.26、Get Bucket Logging

### 功能说明

get bucket logging 请求获取存储桶的日志转存配置

## 方法原型

```
get_bucket_logging(Bucket="bucket-name")
```

## 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

## 返回结果说明

```
{
  'LoggingEnabled': {
    'TargetPrefix': 'string',
    'TargetBucket': 'string',
    'TargetGrants': [
      {
        'Grantee': {
          'DisplayName': 'string',
          'EmailAddress': 'string',
          'ID': 'string',
          'Type': 'CanonicalUser'|'AmazonCustomerByEmail'|'Group',
          'URI': 'string'
        },
        'Permission': 'FULL_CONTROL'|'READ'|'WRITE'
      }
    ]
  },
  'ResponseMetadata': {
    'RequestId': 'tx00000000000000000001-005fc8d74e-5e67-default',
    'HostId': '',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'date': 'Thu, 03 Dec 2020 12:17:18 GMT',
      'content-length': '0',
      'connection': 'Keep-Alive',
      'x-amz-request-id': 'tx00000000000000000001-005fc8d74e-5e67-
default'
    },
    'RetryAttempts': 0
  }
}
```

| 返回结果           | 描述  | 类型      |
|----------------|---|---------|
| LoggingEnabled | 描述日志存储位置和日志对象前缀   | dict    |
| TargetBucket   | 日志存储位置。可以将日志存放到任意用户拥有的桶中，包含源存储桶。用户可以配置多个源桶的日志均投放到同一个目标存储桶中，在这种情况下，用户可以使用 TargetPrefix 区分日志来自哪个源存储桶。 | string  |
| TargetGrants   | 授权信息  | list    |
| Grantee        | 授权许可  | dict    |
| DisplayName    | 展示名字  | string  |
| EmailAddress   | 邮件地址  | integer |
| ID             | 授权用户 ID   | string  |
| Type           | 授权类型  | string  |
| URI            | 授权组 URI   | string  |
| Permission     | 日志访问许可  | string  |
| TargetPrefix   | 日志对象前缀。若多个源存储桶的日志均写到同一个目标存储桶中，则可以通过目标前缀来区分日志来自哪一个源存储桶   | string  |

## 示例

```
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.get_bucket_logging(Bucket='bucket1')
```

## 1.27、Put Bucket CORS

### 功能说明

Put Bucket CORS 接口用来请求设置 Bucket 的跨域资源共享权限。

### 方法原型

```
put_bucket_cors(
    Bucket='string',
    CORSConfiguration={
        'CORSRules': [
            {
                'ID': 'string',
                'AllowedHeaders': [
                    'string',
```

```

    ],
    'AllowedMethods': [
      'string',
    ],
    'AllowedOrigins': [
      'string',
    ],
    'ExposeHeaders': [
      'string',
    ],
    'MaxAgeSeconds': 123
  },
]
}
)

```

## 参数说明

| 参数名称              | 参数描述   | 类型      | 是否必须 |
|-------------------|--|---------|------|
| Bucket            | 要设置跨域的 bucket 名称   | string  | 是    |
| CORSConfiguration | 描述存储桶中对象的跨源访问配置。   | dict    | 是    |
| CORSRules         | 为指定 bucket 配置的所有跨域规则的集合，允许配置 100 条规则   | list    | 是    |
| ID                | 跨域规则 ID, 最大长度 255  | string  | 否    |
| AllowedHeaders    | 允许浏览器发送 CORS 请求时携带的自定义 HTTP 请求头部，不区分英文大小写，单条 CORSRule 可以配置多个 AllowedHeader。                  | string  | 否    |
| AllowedMethods    | 允许该源执行的 HTTP 方法列表，包括 GET, PUT, HEAD, POST, and DELETE。单挑规则可以配置多个方法。                          | string  | 是    |
| AllowedOrigins    | 允许能够访问该 bucket 的一个或多个源, 支持 * 通配符，表示所有域名都允许访问，不推荐。一条 CORSRule 可以配置多个 allowedorigins           | string  | 是    |
| ExposeHeaders     | 允许浏览器获取的 CORS 请求响应中的头部，不区分英文大小写, 单条 CORSRule 可以配置多个 ExposeHeader。                            | string  | 否    |
| MaxAgeSeconds     | 跨域资源共享配置的有效时间，单位为秒，对应 CORS 请求响应中的 Access-Control-Max-Age 头部，单条 CORSRule 只能配置一个 MaxAgeSeconds | integer | 否    |

## 返回结果说明

```

{
  'ResponseMetadata': {
    '...': '...',
  },
}

```

| 返回结果             | 描述                                  | 类型   |
|------------------|-------------------------------------|------|
| ResponseMetadata | http 请求返回数据，包括请求返回状态码及 RequestID 等等 | Dict |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
response = s3_client.put_bucket_cors(
    # Bucket='my-new-bucket0',
    Bucket='bucket-zq',
    CORSConfiguration={
        'CORSRules': [
            {
                'AllowedHeaders': [
                    '*'
                ],
                'AllowedMethods': [
                    'PUT',
                    'POST',
                    'DELETE',
                    'HEAD',
                    'GET',
                ],
                'AllowedOrigins': [
                    'http://127.0.0.1:5500'
                ],
                'ExposeHeaders': [
                    'ETag',
                ],
                'MaxAgeSeconds': 30,
            }
        ]
    },
)
print(response)

```

## 1.28、Get Bucket CORS

### 功能说明

Get Bucket CORS 接口用来请求获取 Bucket 的跨域资源共享权限配置。

### 方法原型

```
get_bucket_cors(Bucket='string')
```

### 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

## 返回结果说明

```
{
  'CORSRules': [
    {
      'ID': 'string',
      'AllowedHeaders': [
        'string',
      ],
      'AllowedMethods': [
        'string',
      ],
      'AllowedOrigins': [
        'string',
      ],
      'ExposeHeaders': [
        'string',
      ],
      'MaxAgeSeconds': 123
    },
  ]
}
```

| 返回结果           | 描述   | 类型      |
|----------------|--|---------|
| CORSRules      | 为指定 bucket 配置的所有跨域规则的集合，允许配置 100 条规则。  | list    |
| ID             | 跨域规则的 ID,最大长度 255  | string  |
| AllowedHeaders | 允许浏览器发送 CORS 请求时携带的自定义 HTTP 请求头部，不区分英文大小写，单条 CORSRule 可以配置多个 AllowedHeader。                  | string  |
| AllowedMethods | 允许该源执行的 HTTP 方法列表，包括 GET , PUT , HEAD , POST , and DELETE                                    | string  |
| AllowedOrigins | 允许能够访问该 bucket 的一个或多个源   | string  |
| ExposeHeaders  | 允许浏览器获取的 CORS 请求响应中的头部，不区分英文大小写,单条 CORSRule 可以配置多个 ExposeHeader。                             | string  |
| MaxAgeSeconds  | 跨域资源共享配置的有效时间，单位为秒，对应 CORS 请求响应中的 Access-Control-Max-Age 头部，单条 CORSRule 只能配置一个 MaxAgeSeconds | integer |

## 示例

```
#!/usr/bin/env python
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.get_bucket_cors(Bucket='bucket1')
```

## 1.29、Delete Bucket CORS

### 功能说明

Delete Bucket CORS 接口用来删除 Bucket 的跨域资源共享权限配置。

### 方法原型

```
delete_bucket_cors(Bucket='string')
```

### 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

### 返回结果说明

```
{
  'ResponseMetadata': {
    '...': '...',
  },
}
```

| 返回结果             | 描述                                  | 类型   |
|------------------|-------------------------------------|------|
| ResponseMetadata | http 请求返回数据，包括请求返回状态码及 RequestID 等等 | Dict |

### 示例

```
#!/usr/bin/env python
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.delete_bucket_cors(Bucket='bucket1')
```

## 1.30、Put Bucket Versioning

### 功能说明

Put Bucket Versioning 接口实现启用或者暂停 Bucket 的版本控制功能。

## 方法原型

```
put_bucket_versioning(Bucket='bucket-name',  
VersioningConfiguration={'Status':'Enabled'|'Suspended'})
```

## 参数说明

| 参数名称                    | 参数描述                                       | 类型     | 是否必须 |
|-------------------------|--|--------|------|
| Bucket                  | Bucket 的名称                                 | String | 是    |
| VersioningConfiguration | 是否开启 Bucket 的版本控制功能，枚举值：Suspended，Enabled。 | Dict   | 是    |

## 返回结果说明

```
{  
  'ResponseMetadata': {  
    'HTTPStatusCode': 200,  
    'RetryAttempts': 0,  
    'HostId': '',  
    'RequestId': 'tx00000000000000000018-0060a4d268-47ce2-default',  
    'HTTPHeaders': {  
      'date': 'Wed, 19 May 2021 08:55:04 GMT',  
      'content-length': '0',  
      'x-amz-request-id': 'tx00000000000000000018-0060a4d268-47ce2-default',  
      'content-type': 'application/xml',  
      'connection': 'Keep-Alive'  
    }  
  }  
}
```

| 返回结果             | 描述          | 类型   |
|------------------|-------------|------|
| ResponseMetadata | http 请求返回数据 | Dict |
| HTTPStatusCode   | http 请求状态码  | Int  |

## 示例

```
#-*-coding:utf-8 -*-  
  
from boto3.session import Session  
import boto3
```



```
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client('s3',endpoint_url=url)
#设置 bucket 的版本控制配置

print s3_client.put_bucket_versioning(Bucket='bucket-name',
VersioningConfiguration={'Status':'Enabled'})
```

## 1.31、Get Bucket Versioning

### 功能说明

Get Bucket Versioning 接口实现获得 Bucket 的版本控制配置。

### 方法原型

```
get_bucket_versioning(Bucket='bucket-name')
```

### 参数说明

| 参数名称   | 参数描述       | 类型     | 是否必须 |
|--------|------------|--------|------|
| Bucket | Bucket 的名称 | String | 是    |

### 返回结果说明

```
{
  u 'Status': 'Enabled', u 'MFADelete': 'Disabled', 'ResponseMetadata':
  {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx000000000000000000019-0060a4d332-47ce2-default',
    'HTTPHeaders': {
      'date': 'Wed, 19 May 2021 08:58:26 GMT',
      'content-length': '192',
      'x-amz-request-id': 'tx000000000000000000019-0060a4d332-
47ce2-default',
      'content-type': 'application/xml',
      'connection': 'Keep-Alive'
    }
  }
}
```

| 返回结果             | 返回结果描述                            | 类型     |
|------------------|-----------------------------------|--------|
| Status           | Bucket 的版本控制配置，假如没有设置过该配置，该字段不会返回 | String |
| ResponseMetadata | http 请求返回数据                       | Dict   |
| HTTPStatusCode   | http 请求状态码                        | Int    |

## 示例

```
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client('s3',endpoint_url=url)
#获取 Bucket 的版本控制功能配置
print s3_client.get_bucket_versioning(Bucket='bucket-name')
```

## 2、Object 操作

### 2.1、Get Object

#### 功能说明

Get Object 请求可以将一个文件（Object）下载至本地。该操作需要对目标 Object 具有读权限或目标 Object 对所有人都开放了读权限（公有读）。

#### 方法原型

```
get_object(
    Bucket='string',
    IfMatch='string',
    IfModifiedSince=datetime(2015, 1, 1),
    IfNoneMatch='string',
    IfUnmodifiedSince=datetime(2015, 1, 1),
    Key='string',
    Range='string',
    VersionId='string'
)
```

#### 参数说明

| 参数名称              | 参数描述                                       | 类型     | 是否必须 |
|-------------------|--|--------|------|
| Bucket            | Bucket 的名称                                 | String | 是    |
| IfMatch           | 当对象的 Etag 和 IfMatch 中的值一致时返回对象，否则返回 412 错误 | String | 否    |
| IfModifiedSince   | 只有指定时间之后有修改记录的对象才会返回对象，否则返回 304 错误         | String | 否    |
| IfNoneMatch       | 只有对象的 Etag 不满足指定字符串时才会返回对象，否则返回 304 错误     | String | 否    |
| IfUnmodifiedSince | 只有指定时间之后没有修改记录的对象才会返回，否则返回 412 错误          | String | 否    |
| Key               | Object 的名称                                 | String | 是    |
| Range             | 指定下载对象的字节范围，例如 bytes=0-9 表示下载开头 10 个 byte  | String | 否    |
| VersionId         | 对象的某个特定版本的版本号，没有该版本则返回 404 错误              | String | 否    |

## 返回结果说明

```
{
  "Body": <botocore.response.StreamingBody object at 0x7fba661401d0>,
  "AcceptRanges": "bytes",
  "ContentType": "application/octet-stream",
  "ResponseMetadata": {
    "HTTPStatusCode": 206,
    "RetryAttempts": 0,
    "HostId": "",
    "RequestId": "tx00000000000000000000008d6-0060b49d76-20bdd-
default",
    "HTTPHeaders": {
      "content-length": "21",
      "x-rgw-object-type": "Normal",
      "accept-ranges": "bytes",
      "x-amz-meta-s3cmd-attrs":
"atime:1621340201/ctime:1621340201/gid:1001/gname:test_user/md5:aa516f
e673e70c11ad659cb708094fc2/mode:33204/mtime:1621340201/uid:1001/uname:
test_user",
      "last-modified": "Tue, 18 May 2021 12:17:14 GMT",
      "content-range": "bytes 20-40/20971520",
      "etag": "\"7fa986abc03d636c14dea69df48a7090-2\"",
      "x-amz-request-id": "tx00000000000000000000008d6-0060b49d76-
20bdd-default",
      "date": "Mon, 31 May 2021 08:25:26 GMT",
      "content-type": "application/octet-stream"
    }
  },
  "LastModified": datetime.datetime(2021, 5, 18, 12, 17, 14,
tzinfo=tzutc()),
  "ContentRange": "bytes 20-40/20971520",
  "ETag": "\"7fa986abc03d636c14dea69df48a7090-2\"",
  "ContentLength": 21,
  "Metadata": {
```

```

"s3cmd-attrs":
  "atime:1621340201/ctime:1621340201/gid:1001/gname:test_user/md5:aa516fe673e70c11ad659cb708094fc2/mode:33204/mtime:1621340201/uid:1001/uname:test_user"
  }
}

```

| 参数名称             | 参数描述             | 类型             |
|------------------|------------------|----------------|
| Body             | 返回对象的内容          | Streaming Body |
| ResponseMetadata | http 请求返回数据      | Dict           |
| LastModified     | 对象的创建时间          | Datetime       |
| ContentRange     | 指明对象在返回的响应中包含的部分 | String         |
| ETag             | 对象的 ETag         | String         |
| ContentLength    | 响应体的长度           | Integer        |
| Metadata         | 和对象一起存储的元数据信息    | Dict           |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
import datetime
bucket = 'demo-bucket'
key = 'object5'
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client("s3", endpoint_url=url)
date=datetime.datetime(2021, 5, 1, 12, 17, 14)
response = s3_client.get_object(
    Bucket=bucket, Key=key, IfModifiedSince=date
)
print(response)

```

## 2.2、Head Object

### 功能说明

Head Object 请求可以获取对应 Object 的元数据，Head 的权限与 Get 的权限一致。

### 方法原型

```
head_object(
```



```

        'x-amz-version-id': 'u.zsHpk1QMTlgMvqOjQQbjWqmvqjjom',
        'content-type': 'text/plain'
    }

    }, u 'LastModified': datetime.datetime(2021, 5, 19, 3, 2, 52, tzinfo
= tzutc()), u 'ContentLength': 10, u 'VersionId':
'u.zsHpk1QMTlgMvqOjQQbjWqmvqjjom', u 'ETag':
'"2d950f68bda376b90126224bcbd03d5e"', u 'StorageClass': 'STANDARD', u
'Metadata': {
    's3cmd-attrs':
'atime:1621393265/ctime:1621393370/gid:0/gname:root/md5:2d950f68bda376
b90126224bcbd03d5e/mode:33188/mtime:1621393370/uid:0/uname:root'
}
}
}

```

| 返回结果             | 描述           | 类型     |
|------------------|--------------|--------|
| ResponseMetadata | http 请求返回数据  | Dict   |
| HTTPStatusCode   | http 请求状态码   | Int    |
| content-length   | 文件大小，单位字节    | Int    |
| etag             | 文件最新版本的 etag | String |
| StorageClass     | 文件的存储级别      | String |

### 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client('s3',endpoint_url=url)
print (s3_client.head_object(Bucket="bucket-name", Key="Object-name",
IfUnmodifiedSince="2022-01-01", VersionId
="aABzBeMXoSdIZCQ.igDhMK00rnteqml"))

```

## 2.3、Put Object

### 功能说明

Put Object 请求可以将一个文件（Object）上传至指定 Bucket。对象权限默认为私有权限。

### 方法原型

```

# 方法原型

```

```

put_object(
    ACL='private'|'public-read'|'public-read-write'|'authenticated-read',
    Body=b'bytes'|file,
    Bucket='string',
    ContentMD5='string',
    Key='string',
    Metadata={
        'string': 'string'
    },
    Append=True|False,
    AppendPosition=integer)

```

## 参数说明

| 参数名称           | 参数描述                   | 类型      | 是否必须 |
|----------------|------------------------|---------|------|
| ACL            | Object 的访问控制;传空默认为私有权限 | String  | 否    |
| Body           | 对象的数据                  | Bytes   | 否    |
| Bucket         | 存入 bucket 的名称          | String  | 是    |
| ContentMD5     | 对象数据的 MD5              | String  | 否    |
| Key            | 对象的 Key                | String  | 是    |
| Metadata       | Object 的元数据            | Dict    | 是    |
| Append         | 是否以追加模式上传 Object       | bool    | 否    |
| AppendPosition | 追加模式下, 指定追加的位置         | Integer | 否    |

## 返回结果说明

### 普通上传模式

```

{
  "ETag": "\"21fd59a7339f2e5f73c97039254fd784\"",
  "ResponseMetadata": {
    "HTTPStatusCode": 200,
    "RetryAttempts": 0,
    "HostId": "",
    "RequestId": "tx00000000000000000092e-0060b4cd10-20bdd-
default",
    "HTTPHeaders": {
      "date": "Mon, 31 May 2021 11:48:32 GMT",
      "content-length": "0",
      "etag": "\"21fd59a7339f2e5f73c97039254fd784\"",
      "accept-ranges": "bytes",
      "x-amz-request-id": "tx00000000000000000092e-0060b4cd10-
20bdd-default"
    }
  }
}

```

| 返回结果 | 描述 | 类型 |
|------|----|----|
|------|----|----|

|                  |               |        |
|------------------|---------------|--------|
| ETag             | Object 的 ETag | String |
| ResponseMetadata | http 请求返回数据   | Dict   |

## 追加上传模式

```
{
  "ETag": "\"a3b89c7dff296d1940664896e5ed09f6\"",
  "ResponseMetadata": {
    "HTTPStatusCode": 200,
    "RetryAttempts": 0,
    "HostId": "",
    "RequestId": "tx0000000000000000000010b-0060cc6cab-47cd6-default",
    "HTTPHeaders": {
      "content-length": "0",
      "accept-ranges": "bytes",
      "etag": "\"a3b89c7dff296d1940664896e5ed09f6\"",
      "x-amz-request-id": "tx0000000000000000000010b-0060cc6cab-47cd6-default",
      "date": "Fri, 18 Jun 2021 09:51:39 GMT",
      "x-rgw-next-append-position": "110"
    }
  },
  "AppendPosition": 110
}
```

## 示例

### 普通上传模式

```
#!/usr/bin/env python
#-*-coding:utf-8 -*-
from boto3.session import Session
import hashlib
import base64
import datetime

def test_po(bname, key, file, md5):
    # #void
    access_key="your access key"
    secret_key="your secret key"
    url="zos endpoint"
    session = Session(access_key, secret_key)
    s3_client = session.client("s3", endpoint_url=url)

    response = s3_client.put_object(ACL='private', Bucket=bname,
    Metadata=dict(m1='m1'), Body=file, Key=key, ContentMD5=md5,
    ServerSideEncryption='AES256', Tagging='tag=tag-value')
    print(response)

if __name__ == '__main__':
    bucket = 'test-bucket'
    with open('/root/SDK/test/test_file', 'r') as file:
        byte = file.read()
        md5 = hashlib.md5(byte).digest()
        md5 = base64.b64encode(md5)
        test_po(bucket, 'test_put', byte, md5)
```



追加上传模式

```
#!/usr/bin/env python
#-*-coding:utf-8 -*-
from boto3.session import Session
import hashlib
import base64

def test_po(bname, key, file, md5):
    # #void
    access_key="your access key"
    secret_key="your secret key"
    url="zos endpoint"
    session = Session(access_key, secret_key)
    s3_client = session.client("s3", endpoint_url=url)

    response = s3_client.put_object(ACL='private', Bucket=bname,
    Metadata=dict(m1='m1'), Body=file, Key=key, ContentMD5=md5,
    Append=True, AppendPosition=0)
    pos = response['AppendPosition']
    print('next pos:' + str(pos))
    response = s3_client.put_object(ACL='private', Bucket=bname,
    Metadata=dict(m1='m1'), Body=file, Key=key, ContentMD5=md5,
    Append=True, AppendPosition=pos)
    print(response)

if __name__ == '__main__':
    bucket = 'test_append'
    with open('/root/SDK/test/1k', 'r') as file:
        byte = file.read()
        md5 = hashlib.md5(byte).digest()
        md5 = base64.b64encode(md5)
        test_po(bucket, '1k', byte, md5)
```

## 2.4、Delete Object

### 功能说明

Delete Object 请求可以将一个文件（Object）删除。

### 方法原型

```
delete_object(
    Bucket='string',
    Key='string',
    VersionId='string'
)
```

### 参数说明

| 参数名称      | 参数描述                      | 类型     | 是否必须 |
|-----------|---------------------------|--------|------|
| Bucket    | 指定 Object 所在的 Bucket      | String | 是    |
| Key       | 指定要删除的 Key                | String | 是    |
| VersionId | 指定要删除的 Object 的 VersionId | String | 否    |

## 返回结果说明

```
{
  "ResponseMetadata": {
    "HTTPStatusCode": 204,
    "RetryAttempts": 0,
    "HostId": "",
    "RequestId": "tx00000000000000000000b60-0060b4e390-20bdd-default",
    "HTTPHeaders": {
      "date": "Mon, 31 May 2021 13:24:32 GMT",
      "x-amz-request-id": "tx00000000000000000000b60-0060b4e390-20bdd-default"
    }
  }
}
```

| 返回结果             | 描述          | 类型   |
|------------------|-------------|------|
| ResponseMetadata | http 请求返回数据 | Dict |

## 示例

```
##-*-coding:utf-8 -*-
from boto3.session import Session
bucket = 'test-bucket'
key = 'test_put'
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client("s3", endpoint_url=url)
s3_client1 = session.client("s3", endpoint_url=url)

response = s3_client.delete_object(Bucket=bucket, Key=key)
print(response)
```

## 2.5、Delete Multiple Object

### 功能说明

Delete Multiple Object 请求实现批量删除文件，最大支持单次删除 1000 个文件。对于返回结果，COS 提供 Verbose 和 Quiet 两种结果模式。

Verbose 模式将返回每个 Object 的删除结果；Quiet 模式只返回报错的 Object 信息。

## 方法原型

```
delete_objects(  
    Bucket='string',  
    Delete={  
        'Objects': [  
            {  
                'Key': 'string',  
                'VersionId': 'string'  
            },  
        ],  
        'Quiet': True|False  
    },  
)
```

## 参数说明

| 参数名称   | 参数描述   | 类型     | 是否必须 |
|--------|--|--------|------|
| Bucket | 指定 Object 所在的 Bucket   | String | 是    |
| Delete | 其中 Delete[Objects] 指定要删除的对象数组， Delete[Quiet] 为 bool 变量，指定 Verbose 模式的关和开 | Dict   | 是    |

## 返回结果说明

### Verbose 模式开

```
{  
  "Deleted": [  
    {  
      "Key": "test_put1"  
    },  
    {  
      "Key": "test_put2"  
    }  
  ],  
  "ResponseMetadata": {  
    "HTTPStatusCode": 200,  
    "RetryAttempts": 0,  
    "HostId": "",  
    "RequestId": "tx00000000000000000000b65-0060b4e603-20bdd-default",  
    "HTTPHeaders": {  
      "transfer-encoding": "chunked",  
      "date": "Mon, 31 May 2021 13:34:59 GMT",  
      "x-amz-request-id": "tx00000000000000000000b65-0060b4e603-20bdd-default",  
    }  
  }  
}
```

```

    "content-type": "application/xml"
  }
}

```

Verbose 模式关

```

{
  "ResponseMetadata": {
    "HTTPStatusCode": 200,
    "RetryAttempts": 0,
    "HostId": "",
    "RequestId": "tx00000000000000000000b66-0060b4e60c-20bdd-
default",
    "HTTPHeaders": {
      "transfer-encoding": "chunked",
      "date": "Mon, 31 May 2021 13:35:08 GMT",
      "x-amz-request-id": "tx00000000000000000000b66-0060b4e60c-
20bdd-default",
      "content-type": "application/xml"
    }
  }
}

```

| 返回结果             | 描述          | 类型   |
|------------------|-------------|------|
| ResponseMetadata | http 请求返回数据 | Dict |
| Deleted          | 删除成功的对象列表   | List |

示例

```

#-*-coding:utf-8 -*-

from boto3.session import Session
bucket = 'test-bucket'
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"

session = Session(access_key, secret_key)
s3_client = session.client("s3", endpoint_url=url)

delete = {'Objects': [{'Key': 'test_put1'},
                      {'Key': 'test_put2'}], 'Quiet': False}
response = s3_client.delete_objects(Bucket=bucket, Delete=delete)
print(response)

```

## 2.6、Put Object ACL

功能说明

设置 Object 的 ACL，控制对 Object 的访问权限。该操作需要用户具有 WRITE\_ACP 权限。

有三种方式设置 ACL，三种方式不可同时使用，每次只能给一种参数赋值。其中，通过 ACL 参数方式进行操作，是设置预定义的固定的 ACL，不能针对特定用户进行授权，且该参数实现的效果，也可以借由另外两种方式实现，该参数使用请求头进行传递；AccessControlPolicy 参数方式和 Grant\* 参数方式则可以针对特定用户进行授权，AccessControlPolicy 方式通过请求体传递，而 Grant\* 方式通过请求头传递。三种方式都会覆盖原有 ACL 属性，包括对象所有者自身的权限，如需保留原有 ACL 属性，应将需要保留的原 ACL 添加到本次操作的授权中（ACL 参数方式会默认将对象所有者权限设为 FULL\_CONTROL，而另外两种方式则不会保留任何原 ACL 属性）。

## 方法原型

```
put_object_acl(  
    Bucket='string',  
    Key='string',  
    VersionId='string',  
    ACL='private'|'public-read'|'public-read-write'|'authenticated-  
read',  
    AccessControlPolicy={  
        'Grants': [  
            {  
                'Grantee': {  
                    'Type':  
'CanonicalUser'|'AmazonCustomerByEmail'|'Group',  
                    'ID': 'string',  
                    'EmailAddress': 'string',  
                    'URI': 'string'  
                },  
                'Permission':  
'FULL_CONTROL'|'WRITE'|'WRITE_ACP'|'READ'|'READ_ACP'  
            },  
        ],  
        'Owner': {  
            'ID': 'string'  
        }  
    },  
    GrantFullControl='string',  
    GrantRead='string',  
    GrantReadACP='string',  
    GrantWrite='string',  
    GrantWriteACP='string',  
)
```

## 参数说明

| 参数名称                | 参数描述  | 类型     | 是否必须  |
|---------------------|---|--------|---|
| Bucket              | Bucket 的名称  | String | 是   |
| Key                 | 对象名   | String | 是   |
| VersionId           | 多版本场景下，指定对象的特定版本  | String | 否   |
| ACL                 | 预定义的固定 ACL，<br>取值范围<br>'private'<br>'public-read'<br>'public-read-write'<br>'authenticated-read'  | String | ACL 参数方式<br>则必须，其他<br>两种方式，则<br>不能使用                    |
| AccessControlPolicy | 包含多个授权列表和一个所有者参数  | dict   | 该方式下必<br>须，其他两种<br>方式下则不能<br>使用                         |
| Grants              | 授权列表  | list   | 该方式下必须  |
| Grantee             | 被授权用户   | dict   | 该方式下必须  |
| Type                | 被授权用户类型，<br>取值范围<br>'CanonicalUser'<br>'AmazonCustomerByEmail'<br>'Group'   | String | 该方式下必须  |
| ID(Grantee)         | 被授权用户 ID  | String | Type 为<br>'CanonicalUs<br>er'，则该字<br>段必须                |
| EmailAddress        | 被授权用户邮箱   | String | 如果 Type 为<br>'AmazonCust<br>omerByEmail'<br>，则该字段必<br>须 |
| URI                 | 被授权组 URI<br>取值范围为<br>所有用户：<br><a href="http://acs.amazonaws.com/groups/global/AllUsers">http://acs.amazonaws.com/groups/global/AllUsers</a><br>所有认证用户：<br><a href="http://acs.amazonaws.com/groups/global/AuthenticatedUsers">http://acs.amazonaws.com/groups/global/AuthenticatedUsers</a> | String | 如果 Type 为<br>'Group'，则<br>该字段必须                         |
| Permission          | 向被授权用户授予的权限，<br>取值范围<br>'FULL_CONTROL'<br>'WRITE'<br>'WRITE_ACP'<br>'READ'<br>'READ_ACP'  | String | 该方式下必须  |
| Owner               | 对象所有者   | dict   | 该方式下必须  |

|                  |   |        |        |
|------------------|---|--------|--------|
| ID(Owner)        | 对象所有者 ID  | String | 该方式下必须 |
| GrantFullControl | 被授权用户可以对桶进行 read, write, read ACP, and write ACP 操作<br>以下 Grant*参数, 格式都是"id=xxxx"<br>或"emailAddress=xxxx"或者"uri=xxxx"以及他们的组合<br>(用逗号连接) | String | 否      |
| GrantRead        | 被授权用户可以对对象进行读操作   | String | 否      |
| GrantWrite       | 被授权用户可以对对象进行写操作, 删除或覆盖写该对象  | String | 否      |
| GrantReadACP     | 被授权用户可以读取对象的 ACL  | String | 否      |
| GrantWriteACP    | 被授权用户可以修改对象的 ACL  | String | 否      |

## 返回结果说明

```
{
  'ResponseMetadata': {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx0000000000000000000b-0060b870b1-2f63c-default',
    'HTTPHeaders': {
      'date': 'Thu, 03 Jun 2021 06:03:29 GMT',
      'content-length': '0',
      'x-amz-request-id': 'tx0000000000000000000b-0060b870b1-2f63c-default',
      'content-type': 'application/xml',
      'connection': 'Keep-Alive'
    }
  }
}
```

| 返回结果             | 描述          | 类型   |
|------------------|-------------|------|
| ResponseMetadata | http 请求返回数据 | Dict |
| HTTPStatusCode   | http 请求状态码  | Int  |

## 示例

```
from boto3.session import Session
import boto3

access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session = Session(access_key, secret_key)
```

```

s3_client = session.client('s3', endpoint_url=url)

bucket_name = "bucket-1"
object_name = "test"

res = s3_client.put_object_acl(Bucket=bucket_name, Key=object_name,
ACL='public-read-write')
print(res)
res = s3_client.put_object_acl(
    Bucket=bucket_name,
    Key=object_name,
    AccessControlPolicy={
        'Grants': [
            {
                'Grantee': {
                    'Type': 'CanonicalUser',
                    'ID': 'test-1',
                },
                'Permission': 'FULL_CONTROL'
            },
            {
                'Grantee': {
                    'Type': 'CanonicalUser',
                    'ID': 'test-2',
                },
                'Permission': 'READ'
            },
            {
                'Grantee': {
                    'Type': 'AmazonCustomerByEmail',
                    'EmailAddress': 'abc@abc.com',
                },
                'Permission': 'WRITE'
            },
        ],
        'Owner': {
            'ID': 'test-1'
        }
    }
)
print(res)

res = s3_client.put_object_acl(
    Bucket=bucket_name,
    Key=object_name,
    GrantReadACP='id=test-4',
    GrantWriteACP='emailAddress=def@def.com')

print(res)

```

## 2.7、Get Object ACL

功能说明



获取指定 Object 的 ACL。该操作需要 READ\_ACP 权限。该功能返回的结果与 Put Object ACL 参数一致，但是需要注意的是，如果以邮箱类型授权，返回结果中将会以对应被授权用户 ID 形式出现，即 Type 不会是 AmazonCustomerByEmail，而是 CanonicalUser。

## 方法原型

```
get_object_acl(Bucket='string', Key='string', VersionId='string')
```

## 参数说明

| 参数名称      | 参数描述             | 类型     | 是否必须 |
|-----------|------------------|--------|------|
| Bucket    | Bucket 的名称       | String | 是    |
| Key       | 对象名              | String | 是    |
| VersionId | 多版本场景下，指定对象的特定版本 | String | 否    |

## 返回结果说明

```
{
  'Owner': {
    'DisplayName': 'test-1',
    'ID': 'test-1'
  },
  'Grants': [
    {
      'Grantee': {
        'Type': 'Group',
        'URI': 'http://acs.amazonaws.com/groups/global/AllUsers'
      },
      'Permission': 'READ'
    },
    {
      'Grantee': {
        'EmailAddress': '',
        'Type': 'CanonicalUser',
        'DisplayName': 'test-1',
        'ID': 'test-1'
      },
      'Permission': 'FULL_CONTROL'
    }
  ],
  'ResponseMetadata': {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx0000000000000000000010-0060b870b1-2f63c-default',
    'HTTPHeaders': {
```

```

    'date': 'Thu, 03 Jun 2021 06:03:29 GMT',
    'content-length': '843',
    'x-amz-request-id': 'tx00000000000000000010-0060b870b1-
2f63c-default',
    'content-type': 'application/xml',
    'connection': 'Keep-Alive'
  }
}
}

```

| 返回结果                 | 描述          | 类型     |
|----------------------|-------------|--------|
| Owner                | 对象所有者       | Dict   |
| DisplayName(Owner)   | 对象所有者的展示名   | String |
| ID(Owner)            | 对象所有者的用户 ID | String |
| Grants               | 授权列表        | list   |
| Grantee              | 被授权用户       | dict   |
| Type                 | 被授权用户类型     | string |
| URI                  | 被授权组 URI    | string |
| ID(Grantee)          | 被授权用户 ID    | string |
| DisplayName(Grantee) | 被授权用户展示名    | string |
| EmailAddress         | 被授权用户邮箱     | string |
| Permission           | 被授权权限       | string |

## 示例

```

from boto3.session import Session

import boto3

access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url)

bucket_name = "bucket-1"
object_name = "test"
res = s3_client.get_object_acl(Bucket=bucket_name, Key=object_name)

print(res)

```

## 2.8、Put Object Tagging

### 功能说明

将提供的标签集设置为存储桶中已存在的对象。标签是一个键值对。请注意，标签的最大数量限制为每个对象 10 个标签。要使用此操作，您必须具有

执行 s3:PutObjectTagging 操作的权限。默认情况下, Bucket 拥有者拥有此权限, 并且可以将此权限授予其他人。要放置任何其他版本的标签, 请使用 versionId 查询参数。您还需要 s3:PutObjectVersionTagging 操作的权限。

## 方法原型

```
put_object_tagging(  
    Bucket='string',  
    Key='string',  
    VersionId='string',  
    Tagging={  
        'TagSet': [  
            {  
                'Key': 'string',  
                'Value': 'string'  
            },  
        ]  
    }  
)
```

## 参数说明

| 参数名称      | 参数描述   | 类型     | 是否必须 |
|-----------|--|--------|------|
| Bucket    | Bucket 的名称   | String | 是    |
| Key       | Object 的名称   | String | 是    |
| VersionId | 多版本场景下, 指定对象的特定版本。不指定时, 默认为最新版本的 Id, 如存在 null 版本号, 并需对其操作, 则需指定 VersionId='null'。VersionId="" 等效于指定最新版本的 Id。 | String | 否    |
| Tagging   | 标签集容器, 内部为标签列表, 最多 50 个标签, 每个标签都是键值对, key 最大 128 字节, value 最大 256 字节, value 可以为空, key 和 value 均为 utf-8 编码    | dict   | 是    |

## 返回结果说明

```
{  
    'ResponseMetadata': {  
        'RequestId': 'tx000000000000003dfc24-0060b9d6a3-8d768-cn_east2',  
        'HostId': '',  
        'HTTPStatusCode': 200,  
        'HTTPHeaders': {  
            'server': 'ct-zos',  
            'date': 'Fri, 04 Jun 2021 07:30:43 GMT',  
            'content-type': 'application/xml',  
            'content-length': '0',  
            'connection': 'keep-alive',
```

```

        'x-amz-request-id': 'tx000000000000003dfc24-0060b9d6a3-
8d768-cn_east2'
    },
    'RetryAttempts': 0
}
}
}

```

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3

access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url)

bucket_name = "bucket-002"
object_key = "object-002"
version_Id = "yGuRzcUi-2RxqngoTLzdt1jLyKR-p1w"

response = s3_client.put_object_tagging(
    Bucket=bucket_name,
    Key=object_key,
    VersionId=version_Id,
    Tagging={
        'TagSet': [
            {
                'Key': 'key-1',
                'Value': 'val-1'
            }
        ]
    }
)

print(response)

```

## 2.9、Get Object Tagging

### 功能说明

返回对象的标签集。要使用此操作，您必须具有执行 `s3:GetObjectTagging` 操作的权限。默认情况下，操作返回有关对象当前版本的信息。对于多版本的存储桶，您的存储桶中可以有一个对象的多个版本。此时，要检索任何其他版本的标签，请使用 `versionId` 查询参数。同时，您还需要 `s3:GetObjectVersionTagging` 操作的权限。

默认情况下，存储桶拥有者具有此权限，并且可以将此权限授予其他人。

## 方法原型

```
get_object_tagging(  
    Bucket='string',  
    Key='string',  
    VersionId='string'  
)
```

## 参数说明

| 参数名称      | 参数描述  | 类型     | 是否必须 |
|-----------|---|--------|------|
| Bucket    | Bucket 的名称  | String | 是    |
| Key       | Object 的名称  | String | 是    |
| VersionId | 多版本场景下，指定对象的特定版本。不指定时，默认为最新版本<br>的 Id，如存在 null 版本号，并需对其操作，则需指定<br>VersionId='null'。VersionId=""等效于指定最新版本的 Id。 | String | 否    |

## 返回结果说明

```
{  
    'ResponseMetadata': {  
        'RequestId': 'tx00000000000000000000003e02a2-0060b9d762-8d768-cn_east2',  
        'HostId': '',  
        'HTTPStatusCode': 200,  
        'HTTPHeaders': {  
            'server': 'ct-zos',  
            'date': 'Fri, 04 Jun 2021 07:33:54 GMT',  
            'content-type': 'application/xml',  
            'content-length': '169',  
            'connection': 'keep-alive',  
            'x-amz-request-id': 'tx00000000000000000000003e02a2-0060b9d762-  
8d768-cn_east2'  
        },  
        'RetryAttempts': 0  
    },  
    'TagSet': [{  
        'Key': 'key-1',  
        'Value': 'val-1'  
    }]  
}
```

## 示例

```
#-*-coding:utf-8 -*-  
from boto3.session import Session  
import boto3
```

```

access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url)

bucket_name = "bucket-002"
object_key = "object-002"
version_Id = "yGuRzcUi-2RxqngoTLzdt1jLyKR-p1w"

response = s3_client.get_object_tagging(
    Bucket=bucket_name,
    Key=object_key,
    VersionId=version_Id
)

print(response)

```

## 2.10、Delete Object Tagging

### 功能说明

从指定的对象中删除整个标记集。要使用此操作，您必须具有执行 `s3:DeleteObjectTagging` 操作的权限。要删除特定对象版本的标签，请在请求中添加 `versionId` 查询参数。您将需要 `s3:DeleteObjectVersionTagging` 操作的权限。

### 方法原型

```

delete_object_tagging(
    Bucket='string',
    Key='string',
    VersionId='string'
)

```

### 参数说明

| 参数名称      | 参数描述  | 类型     | 是否必须 |
|-----------|---|--------|------|
| Bucket    | Bucket 的名称  | String | 是    |
| Key       | Object 的名称  | String | 是    |
| VersionId | 多版本场景下，指定对象的特定版本。不指定时，默认为最新版本的 Id，如存在 null 版本号，并需对其操作，则需指定 <code>VersionId='null'</code> 。 <code>VersionId=""</code> 等效于指定最新版本的 Id。 | String | 否    |

## 返回结果说明

```
{
  'ResponseMetadata': {
    'RequestId': 'tx00000000000000410f7f-0060b9db25-843b5-cn_east2',
    'HostId': '',
    'HTTPStatusCode': 204,
    'HTTPHeaders': {
      'server': 'ct-zos',
      'date': 'Fri, 04 Jun 2021 07:49:57 GMT',
      'connection': 'keep-alive',
      'x-amz-request-id': 'tx00000000000000410f7f-0060b9db25-843b5-cn_east2'
    },
    'RetryAttempts': 0
  }
}
```

## 示例

```
##-*-coding:utf-8 -*-
from boto3.session import Session
import boto3

access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url)

bucket_name = "bucket-002"
object_key = "object-002"
version_Id = "yGuRzcUi-2RxqngoTLzdt1jLyKR-p1w"

response = s3_client.delete_object_tagging(
    Bucket=bucket_name,
    Key=object_key,
    VersionId=version_Id
)
print(response)
```

## 2.11、Put Object Legal Hold

### 功能说明

put object legal hold 请求在指定对象上使用依法保留配置

### 方法原型

```

put_object_legal_hold(
    Bucket="bucket-name",
    Key='string',
    LegalHold={
        'Status': 'ON'|'OFF'
    },
    VersionId='string'
)

```

## 参数说明

| 参数名称      | 参数描述             | 类型     | 是否必须 |
|-----------|------------------|--------|------|
| Bucket    | Bucket 的名称       | String | 是    |
| Key       | 进行依法保留设置的对象名称    | string | 是    |
| LegalHold | 指定对象依法保留配置       | dict   | 否    |
| Status    | 表示指定对象是否设置依法保留配置 | string | 否    |
| VersionId | 配置依法保留的对象的版本 ID  | string | 否    |

## 返回结果说明

```

{
  'ResponseMetadata': {
    'RequestId': 'tx00000000000000000001-005fc8d74e-5e67-default',
    'HostId': '',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'date': 'Thu, 03 Dec 2020 12:17:18 GMT',
      'content-length': '0',
      'connection': 'Keep-Alive',
      'x-amz-request-id': 'tx00000000000000000001-005fc8d74e-5e67-
default'
    },
    'RetryAttempts': 0
  }
}

```

| 返回结果             | 描述                                  | 类型   |
|------------------|-------------------------------------|------|
| ResponseMetadata | http 请求返回数据，包括请求返回状态码及 RequestID 等等 | Dict |

## 示例

```

#-*-coding:utf-8 -*-

```



```

from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.put_object_legal_hold(
    Bucket='bucket1',
    Key='obj',
    LegalHold={
        'Status': 'ON'
    },
    VersionId='5KGFjmi87mTwpTE1HQITIKHcU1W0y8z',
)

```

## 2.12、Get Object Legal Hold

### 功能说明

get object legal hold 请求获取指定对象的当前依法保留状态

### 方法原型

```

get_object_legal_hold(
    Bucket="bucket-name",
    Key='string',
    VersionId='string'
)

```

### 参数说明

| 参数名称      | 参数描述            | 类型     | 是否必须 |
|-----------|-----------------|--------|------|
| Bucket    | Bucket 的名称      | String | 是    |
| Key       | 进行依法保留设置的对象名称   | String | 是    |
| VersionId | 配置依法保留的对象的版本 ID | String | 否    |

### 返回结果说明

```

{
  'LegalHold': {
    'Status': 'ON'
  }
  'ResponseMetadata': {

```

```

'RequestId': 'tx00000000000000000001-005fc8d74e-5e67-default',
'HostId': '',
'HTTPStatusCode': 200,
'HTTPHeaders': {
  'date': 'Thu, 03 Dec 2020 12:17:18 GMT',
  'content-length': '0',
  'connection': 'Keep-Alive',
  'x-amz-request-id': 'tx00000000000000000001-005fc8d74e-5e67-
default'
},
'RetryAttempts': 0
}
}

```

| 返回结果             | 描述                                   | 类型     |
|------------------|--------------------------------------|--------|
| ResponseMetadata | http 请求返回数据, 包括请求返回状态码及 RequestID 等等 | Dict   |
| Status           | 表示指定对象是否设置依法保留配置                     | string |
| VersionId        | 配置依法保留的对象的版本 ID                      | string |

## 示例

```

#-*-coding:utf-8 -*-

from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.get_object_legal_hold(
    Bucket='bucket1',
    Key='obj',
    VersionId='5KGFjmi87mTwpTE1HQITIKHcUlW0y8z',
)

```

## 2.13、Put Object Retention

### 功能说明

put object retention 请求设置对象保留期限配置。

### 方法原型

```
put_object_retention(
```

```

Bucket="bucket-name",
  Key='string',
  Retention={
    'Mode': 'GOVERNANCE'|'COMPLIANCE',
    'RetainUntilDate': datetime(2015, 1, 1)
  },
  VersionId='string',
  BypassGovernanceRetention=True|False
)

```

## 参数说明

| 参数名称                      | 参数描述             | 类型       | 是否必须 |
|---------------------------|------------------|----------|------|
| Bucket                    | Bucket 的名称       | String   | 是    |
| Key                       | 进行依法保留设置的对象名称    | string   | 是    |
| Retention                 | 对象保留期限配置元素       | dict     | 否    |
| Mode                      | 表示指定对象的保留期限模式    | string   | 否    |
| RetainUntilDate           | 对象锁定保留期限过期日期     | datetime | 否    |
| VersionId                 | 配置依法保留的对象的版本 ID  | string   | 否    |
| BypassGovernanceRetention | 表示是否这个操作应该绕过监管模式 | Boolean  | 否    |

## 返回结果说明

```

{
  'ResponseMetadata': {
    'RequestId': 'tx00000000000000000001-005fc8d74e-5e67-default',
    'HostId': '',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'date': 'Thu, 03 Dec 2020 12:17:18 GMT',
      'content-length': '0',
      'connection': 'Keep-Alive',
      'x-amz-request-id': 'tx00000000000000000001-005fc8d74e-5e67-
default'
    },
    'RetryAttempts': 0
  }
}

```

| 返回结果             | 描述                                  | 类型   |
|------------------|-------------------------------------|------|
| ResponseMetadata | http 请求返回数据，包括请求返回状态码及 RequestID 等等 | Dict |

## 示例

```
#-*-coding:utf-8 -*-

from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.put_object_retention(
    Bucket='bucket1',
    Key='obj',
    Retention={
        'Mode': 'GOVERNANCE',
        'RetainUntilDate': datetime(2020, 11, 25)
    },
    VersionId='5KGFjmi87mTwpTE1HQITIKHcUlW0y8z',
    BypassGovernanceRetention=True
)
```

## 2.14、Get Object Retention

### 功能说明

get object retention 获取对象的保留期限设置

### 方法原型

```
get_object_retention(
    Bucket="bucket-name",
    Key='string',
    VersionId='string'
)
```

### 参数说明

| 参数名称   | 参数描述          | 类型     |
|--------|---------------|--------|
| Bucket | Bucket 的名称    | String |
| Key    | 进行依法保留设置的对象名称 | string |

|           |                 |        |
|-----------|-----------------|--------|
| VersionId | 配置依法保留的对象的版本 ID | string |
|-----------|-----------------|--------|

## 返回结果说明

```
{
  'Retention': {
    'Mode': 'GOVERNANCE',
    'RetainUntilDate': datetime.datetime(2020, 12, 11, 0, 0,
tzinfo=tzutc())
  }
  'ResponseMetadata': {
    'RequestId': 'tx00000000000000000001-005fc8d74e-5e67-default',
    'HostId': '',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'date': 'Thu, 03 Dec 2020 12:17:18 GMT',
      'content-length': '0',
      'connection': 'Keep-Alive',
      'x-amz-request-id': 'tx00000000000000000001-005fc8d74e-5e67-
default'
    },
    'RetryAttempts': 0
  }
}
```

| 返回结果             | 描述                                  | 类型       |
|------------------|-------------------------------------|----------|
| ResponseMetadata | http 请求返回数据，包括请求返回状态码及 RequestID 等等 | Dict     |
| Retention        | 对象保留期限配置元素                          | dict     |
| Mode             | 表示指定对象的保留期限模式                       | string   |
| RetainUntilDate  | 对象锁定保留期限过期日期                        | datetime |

## 示例

```
##-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.get_object_retention(
    Bucket='bucket1',
    Key='obj',
    VersionId='5KGFjmi87mTwpTE1HQITIKHcUlW0y8z',
)
```

## 2.15、Generate Presigned Url

### 功能说明

Generate Presigned Url 请求生成一个临时的预签名的 Url，没有权限访问集群的用户可以通过该 Url 访问集群，包括上传、下载文件等等。

### 方法原型

```
generate_presigned_url(  
    ClientMethod='put_object',  
    Params={  
        'Bucket': 'bucket-s6',  
        'Key': 'test1.py',  
    },  
    ExpiresIn=3600,  
    HttpMethod='PUT',  
)
```

### 参数说明

| 参数名称         | 参数描述                                  | 类型     | 是否必须 |
|--------------|---------------------------------------|--------|------|
| ClientMethod | 客户端使用的 S3 方法                          | String | 是    |
| Params       | 客户端方法使用的参数                            | String | 是    |
| ExpiresIn    | 生成的 Url 的有效时间，超过该时间使用该 Url 操作集群无效，单位秒 | Dict   | 否    |
| HttpMethod   | Http 请求使用的方法                          | String | 否    |

### 返回结果说明

```
http://192.168.218.130:7480/bucket-s6/test1.py?AWSAccessKeyId=/\*  
your access  
key&Expires=1622796987&Signature=EW8brpCcL5E36UfqDXCbF%2FBuHTk%3D
```

| 返回结果 | 描述           | 类型   |
|------|--------------|------|
| Url  | 集群生成的临时的 Url | Dict |

### 示例

```
#-*-coding:utf-8 -*-  
from boto3.session import Session
```

```

import boto3
import requests
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client('s3',endpoint_url=url)
#print s3_client.upload_part(Bucket='bucket-
version',Body=open("boto.py", 'rb').read(), Key='file-
name',PartNumber=1, UploadId='2~CmIMX303hyo2PJ9Kbegklv2pu7R93MA')
url= s3_client.generate_presigned_url(
    ClientMethod='put_object',
    Params={
        'Bucket': 'bucket-s6',
        'Key': 'test1.py',
    },
    ExpiresIn=3600,
    HttpMethod='PUT',
)
print url
with open('/root/test1.py', 'rb') as file:
    response = requests.put(url, data=file)

    response.raise_for_status()

```

## 2.16、Generate Presigned Post

### 功能说明

Generate Presigned Post 请求生成一个临时的预签名的 Url，没有权限访问集群的用户可以通过该 Url 上传文件到集群(通过 http 的 post 方式)。预签名上传的对象，权限默认为私有。

### 方法原型

```

generate_presigned_url(
    Bucket='bucket-name',
    key = 'key-name',
    ExpiresIn=3600,
)

```

### 参数说明

| 参数名称 | 参数描述 | 类型 | 是否必须 |
|------|------|----|------|
|------|------|----|------|

|           |   |        |   |
|-----------|---|--------|---|
| Bucket    | 文件要上传的 Bucket 名称                        | String | 是 |
| Key       | 文件在集群中保存的名字                             | String | 是 |
| ExpiresIn | 生成的 Url 的有效时间, 超过该时间使用该 Url 操作集群无效, 单位秒 | Dict   | 否 |

## 返回结果说明

```
{
  'url': u 'http://192.168.218.130:7480/bucket-s6',
  'fields': {
    'policy': u
'eyJjb25kaXRpb25zIjogW3siYnVja2V0IjogImJ1Y2tldC1zNiJ9LCB7ImtleSI6ICJib3RvLnB5In1dLCAiZXhwaXJhdGlvbiI6IClYMDIxLTA2LTA0VDA5OjIzOjI2WiJ9',
    'AWSAccessKeyId': u 'your access key',
    'key': 'boto.py',
    'signature': u '2Tny1QLVyPxuqWJfviH/mWQCvxg='
  }
}
```

| 返回结果   | 描述  | 类型     |
|--------|---|--------|
| Url    | 集群生成的临时的 Url                                      | String |
| fields | 服务端返回的数据, 包含签名、policy 等数据, 需要在 post 文件上传的时候发送到服务端 | Dict   |

## 示例

```
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
import requests
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client('s3',endpoint_url=url)
#print s3_client.upload_part(Bucket='bucket-
version',Body=open("boto.py", 'rb').read(), Key='file-
name',PartNumber=1, UploadId='2~CmIMX303hyo2PJ9Kbegklv2pu7R93MA')
response = s3_client.generate_presigned_post(
    Bucket='bucket-s6',
    Key = 'boto.py',
    ExpiresIn=3600
)
print response
with open('/root/boto.py', 'rb') as f:
    file = {'file': ('/root/boto.py', f)}
    http_response = requests.post(response['url'],
data=response['fields'],files=file)
    print http_response
```



## 2.17、Put Object Copy

### 功能说明

Put Object Copy 请求实现将一个文件从源路径复制到目标路径。

### 方法原型

```
copy_object( ACL='private'|'public-read'|'public-read-  
write'|'authenticated-read',  
    Bucket='string',  
    CopySource='string' or {'Bucket': 'string', 'Key': 'string',  
'VersionId': 'string'},  
    Key='string',  
    Metadata={  
        'string': 'string'  
    },  
    MetadataDirective='COPY'|'REPLACE',  
    TaggingDirective='COPY'|'REPLACE',  
    Tagging='string'  
)
```

### 参数说明

| 参数名称              | 参数描述  | 类型              | 是否必须 |
|-------------------|---|-----------------|------|
| ACL               | object 的 ACL  | String          | 否    |
| Bucket            | 复制的目的 Bucket  | String          | 是    |
| CopySource        | 复制的源 String 格式: {bucket}/{key} or {bucket}/{key}?versionId={versionId}, Dict 格式 {'Bucket': 'bucket', 'Key': 'key', 'VersionId': 'id'} | String 或 Dict   | 是    |
| Key               | 复制的目的 Key   | String          | 是    |
| Metadata          | 复制的目的 Metadata  | Dict            | 否    |
| MetadataDirective | Metadata 复制选项, 'COPY' 复制源的 Metadata 或 'REPLACE'使用新指定的 Metadata 覆盖   | Client          | 否    |
| TaggingDirective  | Tagging 复制选项 'COPY' 复制源的 Tag 或 'REPLACE'使用新指定的 Tag 覆盖   | Transfer Config | 否    |
| Tagging           | 指定目的 Object 的 Tag   | String          | 否    |

### 返回结果说明

```
{  
  "CopyObjectResult": {
```

```

    "LastModified": datetime.datetime(2021, 5, 31, 12, 56, 59, 770000,
tzinfo=tzutc()),
    "ETag": "21fd59a7339f2e5f73c97039254fd784"
},
"ResponseMetadata": {
    "HTTPStatusCode": 200,
    "RetryAttempts": 0,
    "HostId": "",
    "RequestId": "tx000000000000000000b2c-0060b4dd1b-20bdd-default",
    "HTTPHeaders": {
        "transfer-encoding": "chunked",
        "date": "Mon, 31 May 2021 12:56:59 GMT",
        "x-amz-request-id": "tx000000000000000000b2c-0060b4dd1b-20bdd-
default",
        "content-type": "application/xml"
    }
}
}

```

| 返回结果             | 描述                                  | 类型   |
|------------------|-------------------------------------|------|
| CopyObjectResult | 复制的结果，包含复制的 Object 的 ETag 和创建时间     | Dict |
| ResponseMetadata | http 请求返回数据，包括请求返回状态码及 RequestID 等等 | Dict |

## 示例

```

#-*-coding:utf-8 -*-

from boto3.session import Session
bucket = 'test-bucket'
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client("s3", endpoint_url=url)
response = s3_client.copy_object(ACL='public-read',
CopySource={'Bucket':'test-bucket','Key':'test_put'}, Bucket='test-
bucket1', Key='test_put2')

print(response)

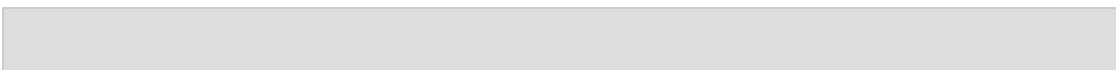
```

## 2.18、Copy

### 功能说明

Copy 请求实现将一个文件从源路径复制到目标路径，支持拷贝大于 5GB 的文件。

### 方法原型



```
copy(CopySource, Bucket, Key, ExtraArgs=None, Callback=None,
SourceClient=None, Config=None)
```

## 参数说明

| 参数名称         | 参数描述  | 类型             | 是否必须 |
|--------------|---|----------------|------|
| CopySource   | 复制的源 bucket、object 和可选的 VersionId 格式: {'Bucket': 'bucket', 'Key': 'key', 'VersionId': 'id'} | Dict           | 是    |
| Bucket       | 复制的目的 Bucket  | String         | 是    |
| Key          | 复制的目的 Key   | String         | 是    |
| ExtraArgs    | 传递给 client 的额外参数  | Dict           | 否    |
| Callback     | 回调函数, 周期性调用, 接受参数为传输的字节数  | Function       | 否    |
| SourceClient | 源 client, 若不提供, 默认使用当前 client   | Client         | 否    |
| Config       | 传输配置, 详见 boto3.s3.transfer.TransferConfig   | TransferConfig | 否    |

## 返回结果说明

无

## 示例

```
#-*-coding:utf-8 -*-
from boto3.session import Session

total_bytes = 0
def show_progress(nbytes):
    global total_bytes
    total_bytes += nbytes
    print('copied bytes: %s' % total_bytes)

bucket = 'test-bucket'
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client("s3", endpoint_url=url)
response = s3_client.copy(CopySource={'Bucket': 'test-bucket', 'Key': '1G'}, Bucket = 'test-bucket1', Key = 'test_copy',
Callback=show_progress)
```

## 2.19、Restore Object

### 功能说明

用于解冻归档对象数据。

## 方法原型

```
restore_object(  
    Bucket='string',  
    Key='string',  
    VersionId='string',  
    RestoreRequest={  
        'Days': 123,  
        'GlacierJobParameters': {  
            'Tier': 'Standard'  
        }  
    }  
)
```

## 参数说明

| 参数名称           | 参数描述                       | 类型      | 是否必须 |
|----------------|----------------------------|---------|------|
| Bucket         | Bucket 的名称                 | string  | 是    |
| Key            | 要执行解冻操作的对象的名称              | string  | 是    |
| VersionId      | 对象的版本号                     | string  | 否    |
| RestoreRequest | 解冻操作的相关参数                  | Dict    | 是    |
| Days           | 解冻之后副本的保留时间，目前支持范围为 1-31 天 | integer | 是    |
| Tier           | 解冻模式，目前只支持 Standard        | String  | 是    |

## 返回结果说明

```
{  
    'ResponseMetadata': {  
        'HTTPStatusCode': 202,  
        'RetryAttempts': 0,  
        'HostId': '',  
        'RequestId': 'tx0000000000000000000004-00621c88d4-12211-default',  
        'HTTPHeaders': {  
            'date': 'Mon, 28 Feb 2022 08:33:24 GMT',  
            'content-length': '0',  
            'x-amz-request-id': 'tx0000000000000000000004-00621c88d4-12211-default',  
            'connection': 'Keep-Alive'  
        }  
    }  
}
```

| 返回结果             | 描述          | 类型   |
|------------------|-------------|------|
| ResponseMetadata | http 请求返回数据 | Dict |

## 示例

```
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
import sys
bucket=sys.argv[1]
obj=sys.argv[2]
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client("s3",endpoint_url=url)
print s3_client.restore_object(
    Bucket=bucket,
    Key=obj,
    RestoreRequest={
        'Days': 4,
        'GlacierJobParameters': {
            'Tier': 'Standard'
        }
    }
)
```

## 3、分块上传操作

### 3.1、Create Multipart Upload

#### 功能说明

Create Multipart Upload 请求实现初始化分片上传，成功执行此请求以后会返回 Upload ID 用于后续的 Upload Part 请求。对象权限默认为私有权限。

#### 方法原型

```
create_multipart_upload(
    ACL='private'|'public-read'|'public-read-write'|'authenticated-read',
    Bucket='string',
    Key='string',
    StorageClass='string',
```

```
    Tagging='string',  
  )
```

## 参数说明

| 参数名称         | 参数描述  | 类型     | 是否必须 |
|--------------|---|--------|------|
| ACL          | 要上传文件的 ACL 规则   | String | 否    |
| Bucket       | 文件要上传的 Bucket 名称  | String | 是    |
| Key          | 要上传的文件名称  | String | 是    |
| StorageClass | 文件的存储级别   | String | 否    |
| Tagging      | 文件的标签，字符串要类似于这种 Key1=Value1，而且只能有一个=符号，=之前为 key，之后为 value，所有即使有多个=也会解析为 value | String | 否    |

## 返回结果说明

```
{  
  u 'Bucket': 'bucket-s8', u 'UploadId':  
'2~YG1lhM4GVEFF9dIsy4lHdUu3TSwPGH4', u 'Key': 'file1',  
'ResponseMetadata': {  
  'HTTPStatusCode': 200,  
  'RetryAttempts': 0,  
  
  'HostId': '',  
  'RequestId': 'tx0000000000000000002a-0060a505ca-47ce2-default',  
  'HTTPHeaders': {  
    'date': 'Wed, 19 May 2021 12:34:18 GMT',  
    'content-length': '245',  
    'x-amz-request-id': 'tx0000000000000000002a-0060a505ca-  
47ce2-default',  
    'content-type': 'application/xml',  
    'connection': 'Keep-Alive'  
  }  
}  
}
```

| 返回结果             | 描述                | 类型     |
|------------------|-------------------|--------|
| Bucket           | 文件保存的 Bucket 名称   | String |
| UploadId         | 分段上传 ID           | String |
| Key              | 文件在 Bucket 中保存的名称 | String |
| ResponseMetadata | http 请求返回数据       | Dict   |
| HTTPStatusCode   | http 请求状态码        | Int    |

## 示例

```
#-*-coding:utf-8 -*-
```

```

from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client('s3',endpoint_url=url)
print s3_client.create_multipart_upload(Bucket='bucket-
name',Key='file-name', Tagging='tag-key=tag-value')

```

## 3.2、Upload Part

### 功能说明

Upload Part 请求实现在初始化以后的分块上传，支持的块的数量为 1 到 10000，除了最后一块，其他每块大小都必须大于或等于 5M。在每次请求 Upload Part 时，需要携带 partNumber 和 uploadID，partNumber 为块的编号，支持乱序上传。partNumber 的取值需要从 1 开始。

### 方法原型

```

upload_part(
    Body=b'bytes'|file,
    Bucket='string',
    Key='string',
    PartNumber=1,
    UploadId='string',
)

```

### 参数说明

| 参数名称       | 参数描述   | 类型     | 是否必须 |
|------------|--|--------|------|
| Body       | 要上传的文件数据                                     | String | 是    |
| Bucket     | 文件要上传的 Bucket 名称                             | String | 是    |
| Key        | 要上传的文件名称                                     | String | 是    |
| PartNumber | 当前分段上传的编号                                    | String | 是    |
| UploadId   | 分段上传的 ID，在 <a href="#">3.1</a> 中返回的 UploadId | String | 是    |

### 返回结果说明

```

{
  u 'ETag': '"59d0d19fc45ca69230d858f60a5557f8"', 'ResponseMetadata':
  {

```

```

'HTTPStatusCode': 200,
'RetryAttempts': 0,
'HostId': '',
'RequestId': 'tx0000000000000000000003-0060a5c6d6-5dc75-default',
'HTTPHeaders': {
  'content-length': '0',
  'accept-ranges': 'bytes',
  'connection': 'Keep-Alive',
  'etag': '"59d0d19fc45ca69230d858f60a5557f8"',
  'x-amz-request-id': 'tx0000000000000000000003-0060a5c6d6-5dc75-default',
  'date': 'Thu, 20 May 2021 02:17:58 GMT'
}
}
}
}

```

| 返回结果             | 描述           | 类型     |
|------------------|--------------|--------|
| Etag             | 当前分段数据的 Etag | String |
| ResponseMetadata | http 请求返回数据  | Dict   |
| HTTPStatusCode   | http 请求状态码   | Int    |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client('s3',endpoint_url=url)
print s3_client.upload_part(Bucket='bucket-name',Body=open("file-name",
'rb').read(), Key='key-name',PartNumber=2,
UploadId='2~ziyYVSXVMaM_5IVXMJ4safWl11OU5yA')

```

## 3.3、Complete Multipart Upload

### 功能说明

Complete Multipart Upload 用来实现完成整个分块上传。当您已经使用 Upload Parts 上传所有块以后，你可以用该 API 完成上传。在使用该 API 时，您必须在 Body 中给出每一个块的 PartNumber 和 ETag，用来校验块的准确性。

### 方法原型

```
complete_multipart_upload(
```



```

Bucket='string',
Key='string',
MultipartUpload={
  'Parts': [
    {
      'ETag': 'string',
      'PartNumber': 1
    },
  ]
},
UploadId='string',
)

```

## 参数说明

| 参数名称            | 参数描述   | 类型     | 是否必须 |
|-----------------|--|--------|------|
| Bucket          | 文件要上传的 Bucket 名称                             | String | 是    |
| Key             | 要上传的文件名称                                     | String | 是    |
| MultipartUpload | 所有成功上传的分段的分段信息，包括分段对应 Etag 和分段号              | Dict   | 是    |
| Etag            | 分段对应的 Etag                                   | String | 是    |
| PartNumber      | 分段对应的分段编号                                    | String | 是    |
| UploadId        | 分段上传的 ID，在 <a href="#">3.1</a> 中返回的 UploadId | String | 是    |

## 返回结果说明

```

{
  u 'ETag': 'a7d414b9133d6483d9a1c4e04e856e3b-2', u 'Bucket': 'bucket-s8', u 'Location': 'http://192.168.218.130:7480/bucket-s8/5M',
  'ResponseMetadata': {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx0000000000000000000064-0060a62bc5-5dc75-default',
    'HTTPHeaders': {
      'date': 'Thu, 20 May 2021 09:28:37 GMT',
      'content-length': '296',
      'x-amz-request-id': 'tx0000000000000000000064-0060a62bc5-5dc75-default',
      'content-type': 'application/xml',
      'connection': 'Keep-Alive'
    }
  }, u 'Key': '5M'
}

```

| 返回结果 | 描述 | 类型 |
|------|----|----|
|------|----|----|

|                  |                 |        |
|------------------|-----------------|--------|
| Etag             | 整体文件的 Etag      | String |
| Bucket           | 保存文件的 Bucket 名称 | String |
| Location         | 文件具体位置          | String |
| ResponseMetadata | http 请求返回数据     | Dict   |
| HTTPStatusCode   | http 请求状态码      | Int    |

## 示例

```
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client('s3',endpoint_url=url)
print s3_client.complete_multipart_upload(
    Bucket='bucket-s8',
    Key='5M',
    MultipartUpload={
        'Parts': [
            {
                'ETag': '5f363e0e58a95f06cbe9bbc662c5dfb6',
                'PartNumber': 1
            },
            {
                'ETag': '34ae08b4a695b10f23d94913cb881c6c',
                'PartNumber': 2
            }
        ]
    },
    UploadId='2~3BRpGICgM_5Lym31H53HhY1-XD6vjeH',
)
```

## 3.4、List Parts

### 功能说明

List Parts 用来查询特定分段上传中的已上传的分段的信息。

### 方法原型

```
list_parts(
    Bucket='string',
    Key='string',
    MaxParts=123,
    PartNumberMarker=123,
    UploadId='string',
)
```

## 参数说明

| 参数名称             | 参数描述   | 类型     | 是否必须 |
|------------------|--|--------|------|
| Bucket           | 文件要上传的 Bucket 名称                             | String | 是    |
| Key              | 上传文件在集群中保存的文件名称                              | String | 是    |
| MaxParts         | 最多返回的分段数目                                    | Int    | 否    |
| PartNumberMarker | list 的分段编号的起始编号，只有分段编号大于这个数字的分段信息才会返回        | Int    | 否    |
| UploadId         | 分段上传的 ID，在 <a href="#">3.1</a> 中返回的 UploadId | String | 是    |

## 返回结果说明

```
{
  'ResponseMetadata': {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx00000000000000000000b-0060a5cd69-5dc75-default',
    'HTTPHeaders': {
      'transfer-encoding': 'chunked',
      'date': 'Thu, 20 May 2021 02:46:01 GMT',
      'connection': 'Keep-Alive',
      'x-amz-request-id': 'tx00000000000000000000b-0060a5cd69-5dc75-default',
      'content-type': 'application/xml'
    }
  },
  u 'Bucket': 'bucket-s8',
  u 'NextPartNumberMarker': 3,
  u 'Parts': [{
    u 'LastModified': datetime.datetime(2021, 5, 20, 2, 12, 22, 643000, tzinfo = tzutc()),
    u 'PartNumber': 1,
    u 'ETag': '"eb260e9ae827821beceeed4104f0ad89"',
    u 'Size': 6
  }, {
    u 'LastModified': datetime.datetime(2021, 5, 20, 2, 29, 2, 299000, tzinfo = tzutc()),
    u 'PartNumber': 2,
    u 'ETag': '"59d0d19fc45ca69230d858f60a5557f8"',
    u 'Size': 7
  }],
  u 'UploadId': '2~ziyYVSXVMaM_5IVXMJ4safWl110U5yA',
  u 'StorageClass': 'STANDARD',
  u 'Key': 'file1',
  u 'Owner': {
```

```

    u 'DisplayName': '',
    u 'ID': ''
  },
  u 'MaxParts': 3,
  u 'IsTruncated': False,
  u 'PartNumberMarker': 0
}

```

| 返回结果                 | 描述   | 类型       |
|----------------------|--|----------|
| ResponseMetadata     | http 请求返回数据  | Dict     |
| HTTPStatusCode       | http 请求状态码   | Int      |
| Bucket               | 文件上传的 Bucket 名称  | String   |
| NextPartNumberMarker | 下一次 list 的时候的分段起始编号, 主要用于截断返回时(也就是已上传的分段数目大于当前返回的分段数目), 作为下一次 list 的分段起始编号 | Int      |
| Parts                | 已经上传的分段信息  | Dict     |
| LastModified         | 该分段上次被修改的时间  | Datetime |
| PartNumber           | 分段编号   | Int      |
| Etag                 | 该分段数据对应 Etag   | String   |
| Size                 | 该分段数据大小, 单位字节  | Int      |
| UploadId             | 分段上传的 ID, 在 3.1 中返回  | String   |
| StorageClass         | 分段上传的文件对应的存储级别   | String   |
| Key                  | 分段上传的文件在集群中保存的名字   | String   |
| Owner                | 分段上传的文件所属用户  | Dict     |
| MaxParts             | 当前 list 最多返回的分段数目  | Int      |
| IsTruncated          | 是否截断   | Boolean  |
| PartNumberMarker     | 当前 list 的分段起始编号  | Int      |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session

import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client('s3',endpoint_url=url)
print s3_client.list_parts(Bucket='bucket-s8', Key='file1',MaxParts=2,
PartNumberMarker=0, UploadId='2~ziyYVSXVMaM_5IVXMJ4safWl11OU5yA')

```

## 3.5、Abort Multipart Upload

## 功能说明

Abort Multipart Upload 用来实现舍弃一个分块上传并删除已上传的块。当您调用 Abort Multipart Upload 时，如果有正在使用这个 Upload Parts 上传块请求，则 Upload Parts 会返回失败。

## 方法原型

```
abort_multipart_upload(  
    Bucket='string',  
    Key='string',  
    UploadId='string',  
)
```

## 参数说明

| 参数名称     | 参数描述   | 类型     | 是否必须 |
|----------|--|--------|------|
| Bucket   | 文件要上传的 Bucket 名称                             | String | 是    |
| Key      | 上传文件在集群中保存的文件名称                              | String | 是    |
| UploadId | 分段上传的 ID，在 <a href="#">3.1</a> 中返回的 UploadId | String | 是    |

## 返回结果说明

```
{  
  'ResponseMetadata': {  
    'HTTPStatusCode': 204,  
    'RetryAttempts': 0,  
    'HostId': '',  
    'RequestId': 'tx0000000000000000000014-0060a5d778-5dc75-default',  
    'HTTPHeaders': {  
      'date': 'Thu, 20 May 2021 03:28:56 GMT',  
      'connection': 'Keep-Alive',  
      'x-amz-request-id': 'tx0000000000000000000014-0060a5d778-5dc75-default'  
    }  
  }  
}
```

| 返回结果             | 描述          | 类型   |
|------------------|-------------|------|
| ResponseMetadata | http 请求返回数据 | Dict |

## 示例

```
#!/usr/bin/env python  
#-*-coding:utf-8 -*-  
from boto3.session import Session  
import boto3
```

```

access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client('s3',endpoint_url=url)
print s3_client.abort_multipart_upload(Bucket='bucket-name', Key='key-
name',UploadId='2~ziyYVSVXVMaM_5IVXMJ4safWl110U5yA')

```

## 3.6、List Multipart Uploads

### 功能说明

List Multipart Uploads 用来查询正在进行中的分段上传，也就是已经 Created 但是还没有 Aaborted 或者 Completed 的分段上传数据，单次最多列出 1000 个正在进行中的分段上传。

### 方法原型

```

list_multipart_uploads(
    Bucket='string',
    KeyMarker='string',
    MaxUploads=123,
    Prefix='string',
)

```

### 参数说明

| 参数名称       | 参数描述  | 类型     | 是否必须 |
|------------|---|--------|------|
| Bucket     | 文件要上传的 Bucket 名称                              | String | 是    |
| KeyMarker  | 只有 Key 大于 KeyMarker 的分段上传数据才会返回               | String | 否    |
| MaxUploads | 单次最多返回的分段上传数据，大小是 1-1000，超过 1000 的数据会被视为 1000 | Int    | 否    |
| Prefix     | Key 的前缀，只有以 Prefix 为开头的 Key 才会被返回             | String | 否    |

### 返回结果说明

```

{
  u 'Uploads': [{
    u 'Initiator': {
      u 'DisplayName': 'testuser',
      u 'ID': 'testuser'
    },
    u 'Initiated': datetime.datetime(2021, 6, 23, 8, 20, 45, 149000,
tzinfo = tzutc()),

```

```

    u 'UploadId': '2~AB_w22rcwgDnKcpGeVvae8WPJwqyw3_',
    u 'StorageClass': 'STANDARD',
    u 'Key': '30M',
    u 'Owner': {
      u 'DisplayName': 'testuser',
      u 'ID': 'testuser'
    }
  }, {
    u 'Initiator': {
      u 'DisplayName': 'testuser',
      u 'ID': 'testuser'
    },
    u 'Initiated': datetime.datetime(2021, 6, 23, 8, 21, 1, 499000,
tzinfo = tzutc()),
    u 'UploadId': '2~BRlrFShYE_5m_MU30nxAEmUpn1hPjba',
    u 'StorageClass': 'STANDARD',
    u 'Key': '40M',
    u 'Owner': {
      u 'DisplayName': 'testuser',
      u 'ID': 'testuser'
    }
  }], 'ResponseMetadata': {
  'HTTPStatusCode': 200,
  'RetryAttempts': 0,
  'HostId': '',
  'RequestId': 'tx000000000000000000000038-0060d2ef92-193d76-default',
  'HTTPHeaders': {
    'transfer-encoding': 'chunked',
    'date': 'Wed, 23 Jun 2021 08:23:46 GMT',
    'connection': 'Keep-Alive',
    'x-amz-request-id': 'tx000000000000000000000038-0060d2ef92-193d76-default',
    'content-type': 'application/xml'
  }
}, u 'MaxUploads': 1000, u 'NextKeyMarker': '40M', u 'Bucket':
'bucket1', u 'IsTruncated': False, u 'NextUploadIdMarker':
'2~BRlrFShYE_5m_MU30nxAEmUpn1hPjba'
}

```

| 返回结果             | 描述              | 类型       |
|------------------|-----------------|----------|
| ResponseMetadata | http 请求返回数据     | Dict     |
| Uploads          | 分段上传数据          | Dict     |
| Initiator        | 该分段上传的创建者       | Dict     |
| Initiated        | 该分段上传创建的时间      | datetime |
| UploadId         | 该分段上传的 UploadId | String   |
| StorageClass     | 存储级别            | String   |
| Key              | 分段上传的 Key       | String   |
| Owner            | 分段上传的所有者        | Dict     |

## 示例

```
#-*-coding:utf-8 -*-
from boto3.session import Session
import boto3
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session=Session(access_key,secret_key)
s3_client=session.client('s3',endpoint_url=url)
print s3_client.list_multipart_uploads(
    Bucket='bucket-name',
)
```

## 4、图片处理

### 4.1、Post 请求处理图片

#### 功能说明

该功能是对存储桶中的图片进行处理，并将处理后的图片持久化到指定的存储桶中，支持处理图片格式 JPG、PNG、WEBP、BMP、TIFF。

#### 方法原型

```
process_object(
    ProcessSource='string',
    Bucket='string',
    Key='string',
    ZosProces='string'
)
```

#### 参数说明

| 参数名称          | 参数描述                                      | 类型     | 是否必须 |
|---------------|---|--------|------|
| ProcessSource | 待处理的图片路径，格式{Bucket}/{Object}[?Versionid=] | string | 是    |
| Bucket        | 指定处理后的图片要存放的存储桶                           | string | 是    |
| Key           | 处理后的图片要保存的名称                              | string | 是    |
| ZosProces     | 图片处理的方式，包含缩放，裁剪，旋转，水印，格式转换，获取信息功能         | string | 是    |

ZosProcess 的定义方法为：

- 图片缩放 (e. g. image/resize, w\_300, h\_200, m\_fixed)

| 参数名称 | 参数用途      | 取值       | 是否必须           |
|------|-----------|----------|----------------|
| w    | 指定目标缩放图宽度 | [1,4096] | 使用按百分比缩放可不指定宽高 |



|       |                     |   |                      |
|-------|---------------------|---|----------------------|
| h     | 指定目标缩放图高度           | [1,4096]  | 使用按百分比缩放可不指定宽高       |
| m     | 指定缩放模式              | lfit (默认值): 等比缩放, 目标缩放图为指定 w 和 h 矩形框内的最大图形。<br>mfit: 等比缩放, 目标缩放图为延伸出指定 w 和 h 矩形框外的最小图形。<br>fill: 将原图等比缩放为延伸出指定 w 与 h 的矩形框外的最小图片, 之后将超出的部分进行居中裁剪。<br>pad: 将原图等比缩放为指定 w 和 h 矩形框内最大的图形, 然后使用 color 指定的颜色将矩形框内剩余部分进行填充。<br>fixed: 固定宽高, 强制缩放。 | 否                    |
| color | 缩放模式为 pad 时, 指定填充颜色 | RGB 颜色值, 默认 FFFFFFF(白色)   | 否 (仅当 m 为 pad 模式时使用) |
| p     | 按百分比进行缩放            | [1,1000]<br>小于 100 缩小, 大于 100 放大  | 否                    |
| limit | 指定目标缩放图大于原图时是否缩放    | 1(默认): 目标缩放图大于原图时返回原图<br>0: 按指定参数缩放   | 否                    |

● 图片裁剪 (e. g. image/crop, w\_100, h\_100, x\_10, y\_10, g\_se)

| 参数名称 | 参数用途   | 取值   | 是否必须 |
|------|--|--|------|
| w    | 指定裁剪宽度。  | [0,图片宽度]<br>默认为最大值。  | 否    |
| h    | 指定裁剪高度。  | [0,图片高度]<br>默认为最大值。  | 否    |
| x    | 指定裁剪起点横坐标 (默认左上角为原点)。                              | [0,图片边界]   | 否    |
| y    | 指定裁剪起点纵坐标 (默认左上角为原点)。                              | [0,图片边界]   | 否    |
| g    | 设置裁剪的原点位置。原点按照九宫格的形式分布, 一共有九个位置可以设置, 为每个九宫格的左上角顶点。 | nw: 左上(默认)<br>north: 中上<br>ne: 右上<br>west: 左中<br>center: 中部<br>east: 右中<br>sw: 左下<br>south: 中下<br>se: 右下 | 否    |

● 图片旋转 (e. g. image/rotate, 45)

| 参数名称    | 参数用途         | 取值                        | 是否必须 |
|---------|--------------|---------------------------|------|
| [value] | 图片按顺时针旋转的角度。 | [0,360]<br>默认值: 0, 表示不旋转。 | 是    |

● 水印

- 图片水印 (e. g. image/watermark, image\_aHVkaWUuanBnP3gtem9zLXByb2N1c3M9aW1hZ2UvcmlhbnR1LHBfMzAvcm90YXR1LDE4MA==, g\_north, t\_40)

- 文字水印(e. g. image/watermark, text\_Q2hpbmF0ZWx1Y29t, type\_heiti, color\_FF0000, size\_40, g\_se, t\_80)

| 参数名称   | 参数用途                         | 取值  | 是否必须 |
|--------|------------------------------|---|------|
| t      | 图片水印或文字水印的透明度                | [0, 100]  | 否    |
| x      | 文字水印距离图片边界的水平距离              | [0, 4096]<br>默认值: 10  | 否    |
| y      | 文字水印距离图片边界的垂直距离              | [0, 4096]<br>默认值: 10  | 否    |
| text   | 指定文字水印内容                     | Base64 编码后的字符串, 编码结果字符串中'/'要替换为'_'  | 否    |
| color  | 指定文字水印的颜色                    | RGB 颜色值。<br>默认: FFFFFFF (白色)  | 否    |
| size   | 指定文字水印的字体大小                  | 默认值: 40   | 否    |
| type   | 指定文字水印的字体类型                  | 如 Airal, Helvetica, 支持中文字体包括 yahei(微软雅黑), heiti(黑体), kaishu (楷书), youyuan(幼圆)   | 否    |
| rotate | 指定文字水印顺时针旋转角度                | [0, 360]<br>默认值: 0  | 否    |
| image  | 指定图片水印名称, 水印图片需要和原图存放在相同存储空间 | 水印图片可以进行预处理 (e.g. 水印图片缩放为 30%并旋转 180 度, hudi e.jpg?x-zos-process=image/resize,p_30/rotate,180), 需要转换成 base64 编码, 编码结果字符串中'/'要替换为'_' | 否    |
| g      | 指定水印在图片中的位置                  | nw: 左上(默认)<br>north: 中上<br>ne: 右上<br>west: 左中<br>center: 中部<br>east: 右中<br>sw: 左下<br>south: 中下<br>se: 右下                            | 否    |

- 格式转化(e. g. image/format, png)

| 参数名称    | 参数用途       | 取值                    | 是否必须 |
|---------|------------|-----------------------|------|
| [value] | 将原图转换成指定格式 | Jpg、png、webp、bmp、tiff | 是    |

- 获取图片信息(e. g. image/info)

### 返回结果说明

```
{
  'ETag': '"4cc020485083c1ee00ab44d6ebe34c0b"',
  'ResponseMetadata': {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx000000000000000000024-0060bca54e-d355-default',
    'HTTPHeaders': {
      'date': 'Sun, 06 Jun 2021 10:37:02 GMT',
      'content-length': '0',
      'etag': '"4cc020485083c1ee00ab44d6ebe34c0b"',
    }
  }
}
```

```

        'accept-ranges': 'bytes',
        'x-amz-request-id': 'tx0000000000000000000024-0060bca54e-d355-
default'
    }
}
}
}

```

| 参数名称             | 参数描述        | 类型     |
|------------------|-------------|--------|
| ResponseMetadata | http 请求返回数据 | Dict   |
| ETag             | 对象的 ETag    | String |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"

session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url, verify=False)

dst_bucket_name = 'dst'
dst_key_name = 'hudie_mini.jpg'
source_name = 'src/hudie.jpg'
process_method = 'image/resize,w_200'

print s3_client.process_object(
    Bucket=dst_bucket_name,
    ProcessSource=source_name,
    Key=dst_key_name,
    ZosProcess=process_method
)
print res

```

## 4.2、Get 请求获取图片

### 功能说明

图片处理是在 `get_object` 基础进行的扩展，用于对存储桶中的图片文件在线处理，支持处理图片格式 JPG、PNG、WEBP、BMP、TIFF。

### 方法原型

```

get_object(
    Bucket='string',
    Key='string',
    ZosProces='string'
)

```

)

## 参数说明

| 参数名称      | 参数描述  | 类型     | 是否必须 |
|-----------|---|--------|------|
| Bucket    | 存储桶的名称  | String | 是    |
| Key       | 存储对象的名称   | String | 是    |
| ZosProces | 图片处理的方式, 包含缩放, 裁剪, 旋转, 水印, 格式转换, 获取信息功能, 定义方法同 5.1 参数说明 | String | 是    |

## 返回结果说明

```
{
  'Body': <botocore.response.StreamingBody object at 0x7f85b8466d90>,
  'AcceptRanges': 'bytes',
  'ContentType': 'image/jpeg',
  'ResponseMetadata': {
    'HTTPStatusCode': 200,
    'RetryAttempts': 0,
    'HostId': '',
    'RequestId': 'tx0000000000000000000017-0060bc9edc-d355-default',
    'HTTPHeaders': {
      'content-length': '7997',
      'x-rgw-object-type': 'Normal',
      'x-amz-storage-class': 'STANDARD',
      'accept-ranges': 'bytes',
      'x-amz-meta-s3cmd-attrs':
'atime:1621495036/ctime:1621424206/gid:0/gname:root/md5:97a6d0c27011dd
c89b42456da8b1be8a/mode:33188/mtime:1621424206/uid:0/username:root',
      'last-modified': 'Tue, 25 May 2021 15:55:46 GMT',
      'etag': '"37a704f5a75b786730d7280cf727b72b"',
      'x-amz-request-id': 'tx0000000000000000000017-0060bc9edc-d355-
default',
      'date': 'Sun, 06 Jun 2021 10:09:32 GMT',
      'content-type': 'image/jpeg'
    }
  },
  'LastModified': datetime.datetime(2021, 5, 25, 15, 55, 46, tzinfo =
tzutc()),
  'ContentLength': 7997,
  'ETag': '"37a704f5a75b786730d7280cf727b72b"',
  'StorageClass': 'STANDARD',
  'Metadata': {
    's3cmd-attrs':
'atime:1621495036/ctime:1621424206/gid:0/gname:root/md5:97a6d0c27011dd
c89b42456da8b1be8a/mode:33188/mtime:1621424206/uid:0/username:root'
  }
}
```

```
}
```

| 参数名称             | 参数描述             | 类型            |
|------------------|------------------|---------------|
| Body             | 返回对象的内容          | StreamingBody |
| ContentType      | 返回对象的类型          | string        |
| ResponseMetadata | http 请求返回数据      | Dict          |
| LastModified     | 对象的创建时间          | Datetime      |
| ContentRange     | 指明对象在返回的响应中包含的部分 | String        |
| ETag             | 对象的 ETag         | String        |
| ContentLength    | 响应体的长度           | Integer       |
| Metadata         | 和对象一起存储的元数据信息    | Dict          |

## 示例

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from boto3.session import Session
access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
session = Session(access_key, secret_key)
s3_client = session.client('s3', endpoint_url=url, verify=False)
bucket_name = 'public'
key_name = '1000.jpg'

process_method = 'image/resize,w_300,h_200,m_fixed'

res = s3_client.get_object(
    Bucket=bucket_name,
    Key=key_name,
    ZosProcess= process_method
)
print res
```

## 5、Topic 操作

### 5.1、Put Bucket Topic Configuration

#### 功能说明

创建可以向其发布通知的主题。此操作是幂等的，因此如果请求者已经拥有一个该指定名称的主题，则返回该主题的 ARN，而不创建新主题。

#### 方法原型

```

create_topic(
    Name='string',
    Attributes={
        'string': 'string'
    }
)

```

## 参数说明

| 参数名称       | 参数描述   | 类型     | 是否必须 |
|------------|--|--------|------|
| Name       | Topic 的名称  | String | 是    |
| Attributes | Topic 的属性，根据不同的通知后端（Kafka、RabbitMQ、HTTP/S），具有不同的键值 | Dict   | 是    |

## 返回结果说明

```

{
    'TopicArn': 'arn:aws:sns:zg_cn::Kafka_topic',
    'ResponseMetadata': {
        'RequestId': '77809110-426b-48d8-9e15-32fde7bb27e7.157067.1102100',
        'HTTPStatusCode': 200,
        'HTTPHeaders': {
            'server': 'ct-zos',
            'date': 'Sun, 06 Jun 2021 02:53:37 GMT',
            'content-type': 'application/xml',
            'content-length': '294',
            'connection': 'keep-alive',
            'x-amz-request-id': 'tx000000000000000010d114-0060bc38b1-2658b-cn_east2'
        },
        'RetryAttempts': 0
    }
}

```

| 返回结果             | 描述                                  | 类型     |
|------------------|-------------------------------------|--------|
| ResponseMetadata | http 请求返回数据，包括请求返回状态码及 RequestID 等等 | Dict   |
| TopicArn         | 主题 ARN                              | String |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
from botocore.client import Config
import boto3

```

```

access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
sns_client = boto3.client('sns',
    endpoint_url=url,
    aws_access_key_id=access_key,
    aws_secret_access_key=secret_key,
    use_ssl=True,
    verify=False,
    region_name='default',
    config=Config(signature_version='s3'),
)

topic_name = "Kafka_topic"
kafka_endpoint = "11.50.132.160:9092"

attributes = {}
attributes['push-endpoint'] = 'kafka://' + kafka_endpoint
attributes['kafka-ack-level'] = 'broker'

response = sns_client.create_topic(
    Name=topic_name,
    Attributes=attributes
)

print(response)

```

## 5.2、Get Bucket Topic Configuration

### 功能说明

返回主题的所有属性。

### 方法原型

```
get_topic(TopicArn='string')
```

### 参数说明

| 参数名称     | 参数描述   | 类型     | 是否必须 |
|----------|--|--------|------|
| TopicArn | Topic 的统一标识，参考 ARN 的格式，将会有以下格式：<br>arn:aws:sns:<zone-group>:<tenant>:<topic> | String | 是    |

### 返回结果说明

```

{
  'GetTopicResult': {
    'Topic': {
      'User': 'zousheng',
      'Name': 'Kafka_topic',
      'EndPoint': {
        'EndpointAddress': 'kafka://11.50.132.160:9092',
        'EndpointArgs': 'Attributes.entry.1.key=push-
endpoint&Attributes.entry.1.value=kafka%3A%2F%2F11.50.132.160%3A9092&A
ttributes.entry.2.key=kafka-ack-
level&Attributes.entry.2.value=broker&Version=2010-03-31&kafka-ack-
level=broker&push-endpoint=kafka://11.50.132.160:9092',
        'EndpointTopic': 'Kafka_topic'
      },
      'TopicArn': 'arn:aws:sns:zg_cn::Kafka_topic'
    }
  },
  'ResponseMetadata': {
    'RequestId': '77809110-426b-48d8-9e15-
32fde7bb27e7.157067.1110405',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'server': 'ct-zos',
      'date': 'Sun, 06 Jun 2021 03:11:00 GMT',
      'content-type': 'application/xml',
      'content-length': '719',
      'connection': 'keep-alive',

      'x-amz-request-id': 'tx00000000000000010f185-0060bc3cc4-
2658b-cn_east2'
    },
    'RetryAttempts': 0
  }
}

```

| 返回结果             | 描述                                  | 类型     |
|------------------|-------------------------------------|--------|
| ResponseMetadata | http 请求返回数据，包括请求返回状态码及 RequestID 等等 | Dict   |
| GetTopicResult   | 请求的获取结果                             | Dict   |
| Topic            | Topic 信息                            | Dict   |
| TopicArn         | 主题 ARN                              | String |

## 示例

```

#-*-coding:utf-8 -*-
from boto3.session import Session
from botocore import exceptions
from botocore.client import Config
import boto3

access_key="your access key"
secret_key="your secret key"

```



```

url="zos endpoint"
sns_client = boto3.client('sns',
    endpoint_url=url,
    aws_access_key_id=access_key,
    aws_secret_access_key=secret_key,
    use_ssl=True,
    verify=False,
    region_name='default',
    config=Config(signature_version='s3'),
)

topic_arn = "arn:aws:sns:zg_cn::Kafka_topic"

try:
    response = sns_client.get_topic(TopicArn=topic_arn)
except sns_client.exceptions.ClientError:
    response = "There is not Topic"

print(response)

```

## 5.3、Delete Bucket Topic Configuration

### 功能说明

删除主题。此操作是幂等（Idempotence）操作，因此删除不存在的主题不会导致错误。

### 方法原型

```
delete_topic(TopicArn='string')
```

### 参数说明

| 参数名称     | 参数描述   | 类型     | 是否必须 |
|----------|--|--------|------|
| TopicArn | Topic 的统一标识，参考 ARN 的格式，将会有以下格式：<br>arn:aws:sns:<zone-group>:<tenant>:<topic> | String | 是    |

### 返回结果说明

```

{
  'ResponseMetadata': {
    'RequestId': '77809110-426b-48d8-9e15-32fde7bb27e7.157067.1142347',

```

```
'HTTPStatusCode': 200,
'HTTPHeaders': {
  'server': 'ct-zos',
  'date': 'Sun, 06 Jun 2021 04:14:38 GMT',
  'content-type': 'application/xml',
  'content-length': '204',
  'connection': 'keep-alive',
  'x-amz-request-id': 'tx0000000000000000116e4b-0060bc4bae-
2658b-cn_east2'
},
'RetryAttempts': 0
}
}
```

## 示例

```
#!/usr/bin/env python
#-*-coding:utf-8 -*-
from boto3.session import Session
from botocore.client import Config
import boto3

access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
sns_client = boto3.client('sns',
    endpoint_url=url,
    aws_access_key_id=access_key,
    aws_secret_access_key=secret_key,
    use_ssl=True,
    verify=False,
    region_name='default',
    config=Config(signature_version='s3'),
)

topic_arn = "arn:aws:sns:zg_cn::Kafka_topic"

response = sns_client.delete_topic(TopicArn=topic_arn)
print(response)
```

## 5.4、List Bucket Topic Configuration

### 功能说明

返回所有主题的所有属性。当不含主题时，返回空 Topics 数组元素。

### 方法原型

```
list_topics()
```

## 参数说明

- 无

## 返回结果说明

```
{
  'Topics': [{
    'User': 'zousheng',
    'Name': 'Kafka_topic',
    'EndPoint': {
      'EndpointAddress': 'kafka://11.50.132.160:9092',
      'EndpointArgs': 'Attributes.entry.1.key=push-
endpoint&Attributes.entry.1.value=kafka%3A%2F%2F11.50.132.160%3A9092&A
ttributes.entry.2.key=kafka-ack-
level&Attributes.entry.2.value=broker&Version=2010-03-31&kafka-ack-
level=broker&push-endpoint=kafka://11.50.132.160:9092',
      'EndpointTopic': 'Kafka_topic'
    },
    'TopicArn': 'arn:aws:sns:zg_cn::Kafka_topic'
  ]},
  'ResponseMetadata': {
    'RequestId': '77809110-426b-48d8-9e15-
32fde7bb27e7.157067.1269231',
    'HTTPStatusCode': 200,
    'HTTPHeaders': {
      'server': 'ct-zos',
      'date': 'Sun, 06 Jun 2021 09:04:45 GMT',
      'content-type': 'application/xml',
      'content-length': '796',
      'connection': 'keep-alive',
      'x-amz-request-id': 'tx0000000000000000135def-0060bc8fad-
2658b-cn_east2'
    },
    'RetryAttempts': 0
  }
}
```

## 示例

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from boto3.session import Session
from botocore.client import Config
import boto3

access_key="your access key"
secret_key="your secret key"
url="zos endpoint"
sns_client = boto3.client('sns',
```

```
endpoint_url=url,  
aws_access_key_id=access_key,  
aws_secret_access_key=secret_key,  
use_ssl=True,  
verify=False,  
region_name='default',  
config=Config(signature_version='s3'),  
)  
  
topic_arn = "arn:aws:sns:zg_cn::Kafka_topic"  
  
response = sns_client.list_topics()  
print(response)
```

## 6、客户端加密

客户端加密，在上传对象时提供加密和解密功能。

- 加密：sdk 上传对象时，通过用户提供的密钥，sdk 将数据进行加密后再上传到服务端。服务端保存的对象即为加密后的对象。
- 解密：sdk 下载对象时，通过用户提供的密钥，解密服务端返回的加密对象，并保存到本地。

加密和解密功能都是 sdk 提供的接口自动完成。使用者唯一需要注意的是妥善保存密钥信息，如丢失密钥，则数据不再能够成功解密。

当前 python sdk 仅支持 AES.GCM 加密算法。

### 6.1、安装依赖包

如您需要使用客户端加密功能，在您完成天翼云 python sdk 的安装之后，需要额外安装一个依赖包。包名为：pycryptodome，版本不低于 3.20.0。

执行以下命令

```
pip install pycryptodome==3.20.0
```

### 6.2、生成加密密钥

功能说明

该接口在本地随机成功一个加密密钥，使用者需要妥善保管加密密钥，如丢失，则数据将没有办法进行解密。

### 方法原型

```
get_crypte_key()
```

### 参数说明

无

### 返回结果说明

该接口返回一个 32 byte 大小的密钥，格式为 byte。

### 示例

```
from s3transfer import zoscript
my_key = zoscript.get_crypte_key()
```

## 6.3、保存加密密钥

### 功能说明

为了避免您生成的 key 丢失，提供接口将您生成的 key 保存到本地。保存到本地的 key 已通过 base64 进行编码。

### 方法原型

```
save_key(key, file_path)
```

### 参数说明

| 参数名称 | 参数描述     | 类型     | 是否必须 |
|------|----------|--------|------|
| key  | 您生成的 key | Byte[] | 是    |

|           |               |        |   |
|-----------|---------------|--------|---|
| file_path | 需要将 key 保存的路径 | String | 是 |
|-----------|---------------|--------|---|

### 返回结果说明

无

### 示例

```
from s3transfer import zoscrypt

#生成 key
my_key = zoscrypt.get_crypte_key()

#保存 key
zoscrypt.save_key(my_key, "D:\\data_test\\my_key")
```

## 6.4、加载加密密钥

### 功能说明

加载用户提供的路径下的密钥信息到内存中。

### 方法原型

```
load_key(file_path)
```

### 参数说明

| 参数名称      | 参数描述          | 类型     | 是否必须 |
|-----------|---------------|--------|------|
| file_path | 需要加载的 key 的路径 | String | 是    |

### 返回结果说明

返回指定路径下的密钥信息，类型为 byte 数组。

### 示例

```
from s3transfer import zoscrypt
load_key = zoscrypt.load_key("D:\\data_test\\my_key")
```

## 6.5、加密上传示例

### 功能说明

提供加密上传接口。该接口需要额外传入 `s3_client`, `user_key`, `file_path` 三个参数。其他参数和 2.3 章节的 `Put Object` 接口一致。

### 方法原型

```
put_object_encrypt(s3_client, user_key, file_path, bucket,
                  key, **kwargs):
```

### 参数说明

| 参数名称                   | 参数描述   | 类型   | 是否必须 |
|------------------------|--|--|------|
| <code>s3_client</code> | 初始化成功客户端接口。用来执行上传对象操作。   | <code>boto3.session.Session.SessionClient</code> | 是    |
| <code>user_key</code>  | 用户自己保管的密钥信息，需要包装为 <code>base64</code> 格式。  | <code>String</code>                              | 是    |
| <code>file_path</code> | 需要上传的文件路径  | <code>String</code>                              | 是    |
| <code>bucket</code>    | 需要上传的对象所属的桶  | <code>String</code>                              | 是    |
| <code>key</code>       | 需要上传的对象的 <code>key</code>  | <code>String</code>                              | 是    |
| <code>**kwargs</code>  | 上传对象的其他参数信息，如 <code>ACL</code> , <code>Metadata</code> 等。参数使用方式和 <code>Put Object</code> 一致。 | <code>Dict</code>                                | 否    |

### 返回结果说明

和 `Put Object` 一致。

### 示例

```

#-*-coding:utf-8 -*-
import base64
from s3transfer import zoscrypt
from boto3.session import Session

ak = "your ak"
sk = "your sk"
host = "your host address"

# 初始化 session and client
session=Session(ak,sk)
s3_client=session.client("s3",endpoint_url=host)

#加载本地的密钥信息，在这之前，您需要保证已经成功生成密钥
#并保存到了相应的路径下面。可参考 6.3 章节
load_key = zoscrypt.load_key("D:\\data_test\\my_key")

#需要上传的对象的文件路径，bucket_name, key_name
file_path = "D:\\data_test\\my_data.txt"
bucket_name = "bucket-crypt"
key_name = "test_crypt"

#调用接口上传对象，ACL, metadata 等额外信息可设置到最后的参数中
response = zoscrypt.put_object_encrypt(s3_client,
                                       base64.b64encode(load_key),
                                       file_path,
                                       bucket_name,
                                       key_name,
                                       ACL = "public-read-write")

print(response)

```

## 6.5、加密下载示例

### 功能说明

提供加密上传接口。该接口需要额外传入 `s3_client`, `user_key`, `file_path` 三个参数。其他参数和 2.1 章节的 `Get Object` 接口一致。

### 方法原型

```

get_object_decrypt(s3_client, user_key, file_path, bucket, key,
                  **kwargs)

```

### 参数说明



| 参数名称      | 参数描述  | 类型                           | 是否必须 |
|-----------|---|------------------------------|------|
| s3_client | 初始化成功客户端接口。用来执行获取对象操作。                                    | boto3.session.Session.client | 是    |
| user_key  | 用户自己保管的密钥信息，需要包装为 base64 格式。                              | String                       | 是    |
| file_path | 获取的文件保存的路径  | String                       | 是    |
| bucket    | 需要获取的对象所属的桶   | String                       | 是    |
| key       | 需要获取的对象的 key  | String                       | 是    |
| **kwargs  | 获取对象的其他参数信息，如 IfMatch, VersionId 等。参数使用方式和 Get Object 一致。 | Dict                         | 否    |

## 返回结果说明

和 Put Object 一致。

## 示例

```

#-*-coding:utf-8 -*-

import base64
from s3transfer import zoscrypt
from boto3.session import Session

ak = "your ak"
sk = "your sk"
host = "your host address"

# 初始化 session and client
session=Session(ak,sk)
s3_client=session.client("s3",endpoint_url=host)

#加载本地的密钥信息，在这之前，您需要保证已经成功生成密钥
#并保存到了相应的路径下面。可参考 6.3 章节
load_key = zoscrypt.load_key("D:\\data_test\\my_key")

#需要获取的对象解析后保存的文件路径，bucket_name, key_name
down_path = "D:\\data_test\\my_down.txt"
bucket_name = "bucket-crypt"
key_name = "test_crypt"

#调用接口下载对象，IfMatch, VersionId 等额外信息可设置到最后的参数中
response = zoscrypt.get_object_decrypt(s3_client,
                                       base64.b64encode(load_key),
                                       down_path,
                                       bucket_name,
                                       key_name)

```

```
print(response)
```