



链滴

# Django 基于腾讯云对象存储 SDK 上传图片

作者: [zyk](#)

原文链接: <https://ld246.com/article/1566965564833>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

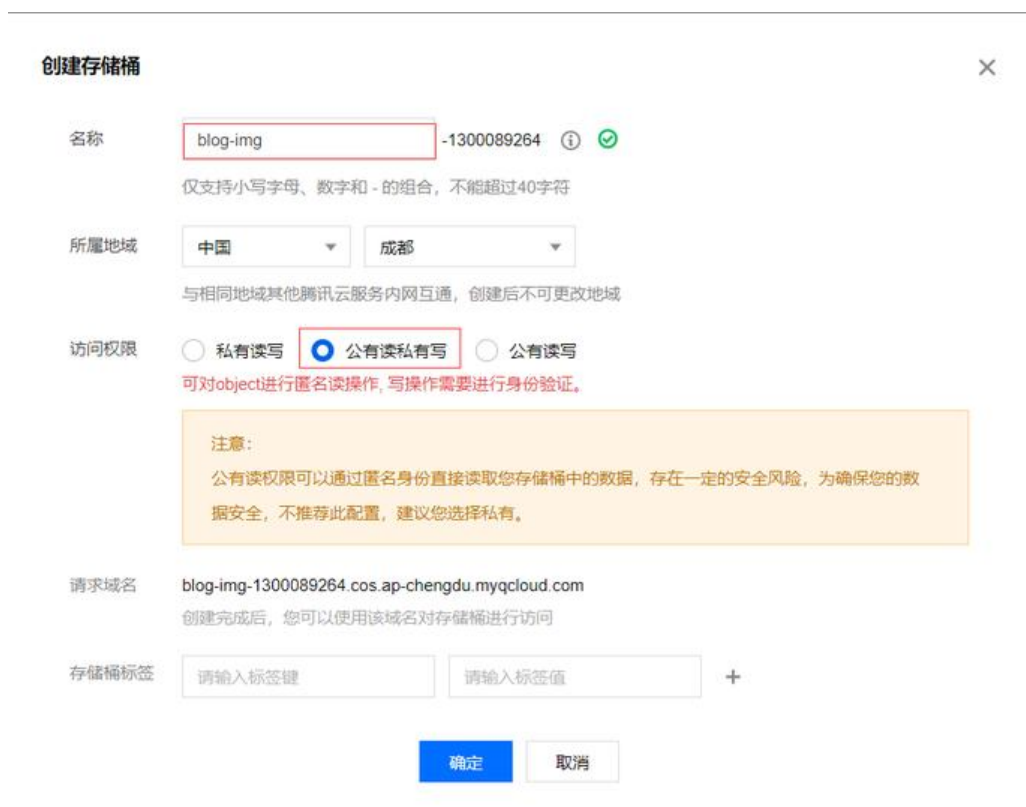
# 1. 前言

本篇文章将讲述如何利用腾讯云对象存储的python SDK上传图片，包含基础配置，原理概述和前后搭建。

文章内容总共分为四个部分，一为环境准备，如何搭建云对象存储基于python的SDK环境；二为编后台，如何调用SDK中的上传图片函数，并对图像进行进一步的处理；三为前端交互，如何利用ajax装formData并提交至后台；最后对结果进行测试。

## 2. 环境准备

- 首先进入腾讯云控制台，进入对象存储管理中心，点击存储桶->创建存储桶，填写名称和编辑访问权限。如果是搭建图床，权限可以设置为**共有读私有写**，因为图床一般需要给网站浏览者访问，但只有理员能上传图片。



**创建存储桶**

名称  -1300089264 ? ✓  
仅支持小写字母、数字和 - 的组合，不能超过40字符

所属地域 中国 成都  
与相同地域其他腾讯云服务内网互通，创建后不可更改地域

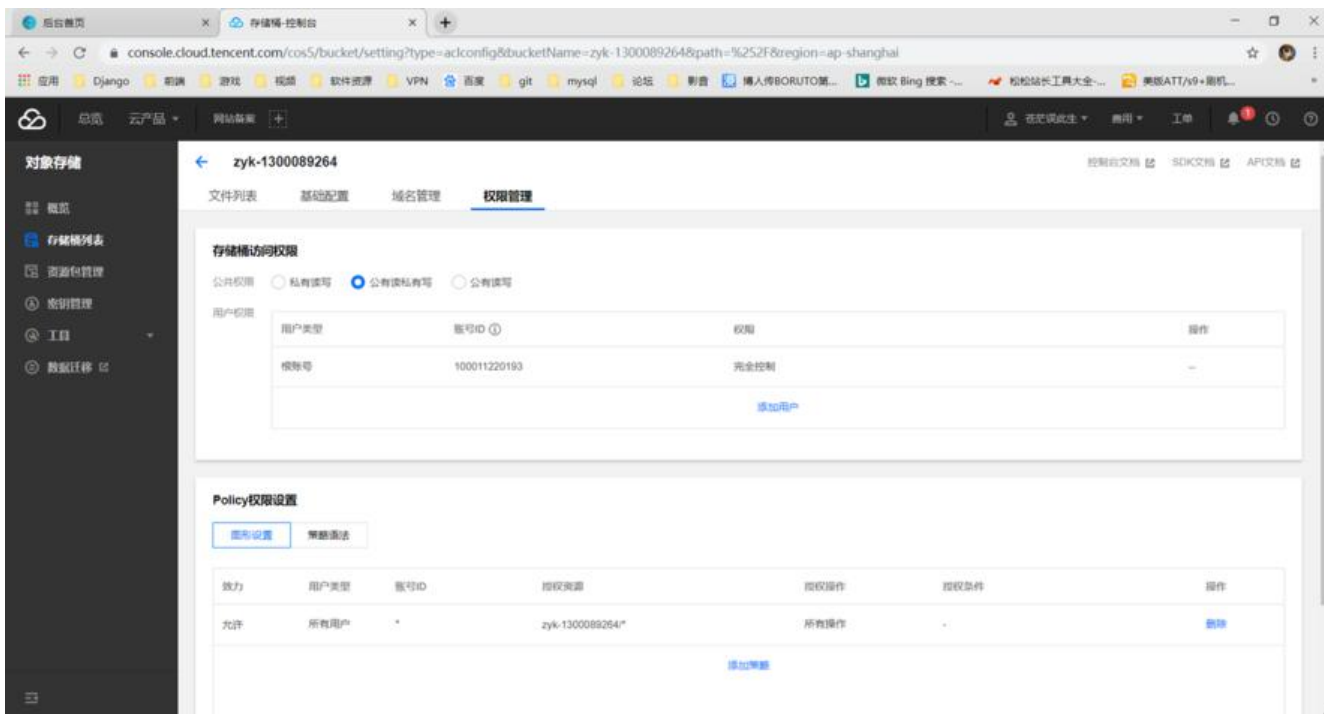
访问权限  私有读写  公有读私有写  公有读写  
可对Object进行匿名读操作，写操作需要进行身份验证。

**注意：**  
公有读权限可以通过匿名身份直接读取您存储桶中的数据，存在一定的安全风险，为确保您的数据安全，不推荐此配置，建议您选择私有。

请求域名   
创建完成后，您可以使用该域名对存储桶进行访问

存储桶标签   +

- 创建完成以后，点击已经创建的存储桶，然后进入权限管理。



● Policy权限设置中删除默认的策略，然后添加策略，设置用户为 **所有用户**，操作为**所有操作**，不置可能会导致之后上传图片报403错误，然后保存即可。



● 由于使用了腾讯云对象存储的Python SDK，所以我们需要利用pip安装所需的第三方库。在命令行输入 `pip install -U cos-python-sdk-v5` 安装SDK，到此环境配置就结束了。

## 3. 编写后台

### 初始化配置信息

- 我在Django项目中添加了一个App，名称为img\_cos，在该App中新建了一个config.py文件，在中写入了关于腾讯云存储的相关配置信息。

```
# appid 已在配置中移除,请在参数 Bucket 中带上 appid。Bucket 由 BucketName-APPID 组成
# 1. 设置用户配置, 包括 secretId, secretKey 以及 Region
# -*- coding=utf-8
import logging
import os
import sys
import uuid
from PIL import Image
from qcloud_cos import CosConfig, CosServiceError
from qcloud_cos import CosS3Client
from my_blog import settings

logging.basicConfig(level=logging.INFO, stream=sys.stdout)

secret_id = '*****' # 替换为用户的 secretId
secret_key = '*****' # 替换为用户的 secretKey
region = 'ap-shanghai' # 替换为用户的 Region
token = None # 使用临时密钥需要传入 Token, 默认为空, 可不填
scheme = 'https' # 指定使用 http/https 协议来访问 COS, 默认为 https, 可不填
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token,
cheme=scheme)
# 2. 获取客户端对象
client = CosS3Client(config)
host = 'https://zyk-1300089264.cos.ap-shanghai.myqcloud.com' # 存储桶访问地址
folder_path = '/article/' # 存储桶中保存图片文件夹名
```

- **注意：**secret id, secret key和 region都需要替换为自己存储桶对应的信息。secret\_id和secret\_key可以到对象存储控制台的密钥管理获取，这里采用的是**项目管理**中的secret\_id和secret\_key。



# 上传图片

- 配置完毕之后，开始编写上传图片函数，在这里调用的SDK中的断点续传图片，支持断点续传，安全性更高。
- 上传原理为：将上传的图片暂时保存至本地服务器，调用SDK中的上传图片方法（捕捉异常），若上传成功，则删除本地服务器上的图片，并返回图片在对象存储中的访问地址。
- 由于图片名称可能会重复，因此采用UUID生成图片名称，保证名称唯一性。

为了节约空间，调用Pillow（没有安装Pillow库的小伙伴，可以先用 `pip` 命令安装Pillow）中 `thumbnail` 方法生成缩略图。具体代码如下。

```
# appid 已在配置中移除,请在参数 Bucket 中带上 appid。Bucket 由 BucketName-APPID 组成
# 1. 设置用户配置, 包括 secretId, secretKey 以及 Region
# -*- coding=utf-8
import logging
import os
import sys
import uuid
from PIL import Image
from qcloud_cos import CosConfig, CosServiceError
from qcloud_cos import CosS3Client
from my_blog import settings
logging.basicConfig(level=logging.INFO, stream=sys.stdout)

secret_id = '*****' # 替换为用户的 secretId
secret_key = '*****' # 替换为用户的 secretKey
region = 'ap-shanghai' # 替换为用户的 Region
token = None # 使用临时密钥需要传入 Token, 默认为空, 可不填
scheme = 'https' # 指定使用 http/https 协议来访问 COS, 默认为 https, 可不填
config = CosConfig(Region=region, SecretId=secret_id, SecretKey=secret_key, Token=token,
cheme=scheme)
# 2. 获取客户端对象
client = CosS3Client(config)
host = 'https://zyk-1300089264.cos.ap-shanghai.myqcloud.com' # 存储桶访问地址
folder_path = '/article/' # 存储桶中保存图片文件夹名

# 上传图片
def upload_file_senior(file):
    file_name = create_file_name(file) # 根据UUID生成文件名
    root_path = os.path.join(settings.MEDIA_ROOT, 'vditor') # 生成文件上传目录
    if not os.path.exists(root_path): # 判断文件上传目录是否存在
        os.makedirs(root_path) # 不存在, 则创建
    file_path = os.path.join(root_path, file_name)
    with open(file_path, 'wb') as f: # 文件流将图片写入本地
        for c in file.chunks():
            f.write(c)
    file_path = compress_img(file_path, 0.75) # 压缩图片, 生成缩略图
    try: # 捕捉异常
        response = client.upload_file( # 根据文件大小自动选择简单上传或分块上传, 分块上传具备
点续传功能。
            Bucket='zyk-1300089264', # 存储桶名称
            LocalFilePath=file_path, # 本地图片路径
            Key=folder_path + file_name, # 上传路径
```

```

        PartSize=1,
        MAXThread=10,
        EnableMD5=False
    )
    if response['ETag'] != "":
        os.remove(file_path) # 上传成功, 删除本地图片
        return host + folder_path + file_name
except CosServiceError as e:
    print(e.get_digest_msg())
    return None

```

# 利用UUID生成文件名, 防止重名

```

def create_file_name(file):
    type_name = file.name[file.name.index('.')] # 获取文件后缀
    file_name = '{}{}'.format(uuid.uuid4(), type_name) # 生成文件名
    return file_name

```

# 压缩图片 (file\_path为图片路径, rate为压缩率, 压缩率范围为0~1)

```

def compress_img(file_path, rate):
    image = Image.open(file_path) # 获得图像
    width = int(image.width * rate) # 宽
    height = int(image.height * rate) # 高
    image.thumbnail((width, height), Image.ANTIALIAS) # 生成缩略图
    image.save(file_path) # 保存
    return file_path

```

- 编写视图函数, 在 **img\_cos/views.py**中加入图片上传函数 (在这里利用了

django restframework库, 未安装的小伙伴可以通过[pip install django restframework](#)命令安装此)。先从请求中获取图片, 然后调用之前封装好的SDK上传图片方法, 根据上传成功与否, 封装json数据返回至前端, 上传成功则将图片访问URL同时返回至前端, 代码如下。

```

from rest_framework.decorators import api_view
from rest_framework.response import Response
from img_cos.config import upload_file, upload_file_senior

"""
上传图片至腾讯云COS
"""

@api_view(['POST']) # 只允许POST请求
def upload_to_cos(request):
    file = request.FILES.get('smfile') # 获取上传图片
    url = upload_file_senior(file) # 调用上传图片SDK, 返回图片访问URL
    if url is None: # 上传失败
        return Response({"msg": "上传失败", "code": 0})
    return Response({"msg": "上传成功", "code": 1, "url": url}) # 上传成功

```

- 将视图函数注册到 **img\_cos/urls.py**中, 如下。

```

from django.urls import path
from img_cos import views

urlpatterns = [
    path('upload_to_cos', views.upload_to_cos, name='upload_to_cos'), # 上传图片至腾讯云COS
]

```

## 4. 前端交互

- 获取上传图片文件对象，利用append()方法将其封装成FormData，调用ajax上传至后台。在这里要注意ajax的编写，需要添加**不处理**上传数据的参数。
- 具体思路：选择要上传的图片，然后设置一个上传图片按钮，为该按钮添加点击事件，事件中调用上传图片函数，用户点击此按钮即可实现图片上传。
- 在我的博客项目中，文章表单字段较多，而文章数据表的图片字段只保存图片访问URL。因此，根据ajax上传图片返回的数据，若上传成功，将返回的图片访问URL写入一个隐藏的 <input> 标签，然后提交表单。具体实现代码如下：

```
//上传文章贴图
function uploadArticleImg() {
  if ($('#img').val() === "" || $('#img').val() == null) { // 判断文件是否为空
    layer.msg("请选择要上传的图片");
    return;
  }
  let formData = new FormData();
  formData.append('smfile', $('#img')[0].files[0]); //将文件对象写入formData
  $.ajax({
    type: 'POST',
    url: '{% url 'upload_to_cos' %}',
    headers: {'X-CSRFToken': getCsrfToken()},
    data: formData,
    dataType: 'JSON',
    contentType: false, //不处理数据
    processData: false,
    cache: false,
    success: function (data) {
      if (data.code === 1) {
        layer.msg(data.msg);
        $("#link_url").val(data.url); // 将图片访问URL写入隐藏的input标签中
      } else {
        layer.msg(data.msg);
      }
    },
    error: function (data) {
      console.log(data);
    }
  });
}
```

## 5. 结果测试

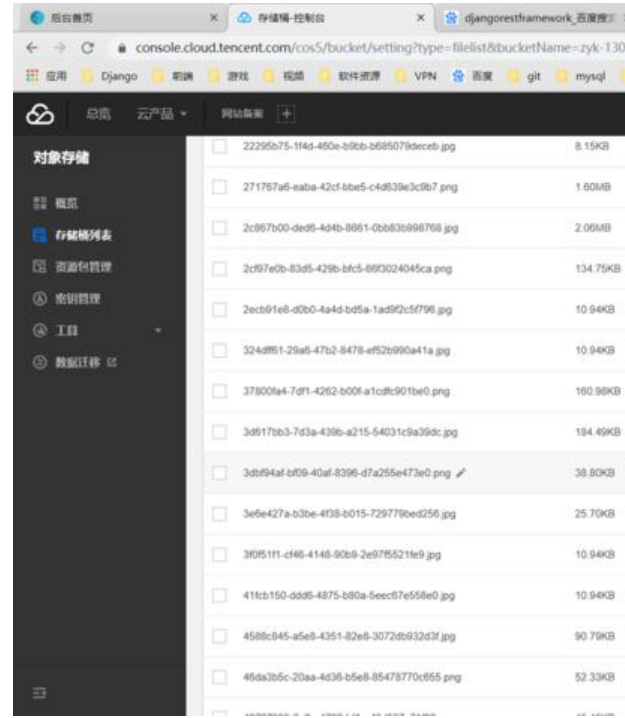
- 运行Django项目，测试图片上传功能，选择要上传的图片，点击上传图片按钮，观察结果。

选择文件 下载.jpg



上传至图床

- 进入腾讯云控制台，查看存储桶中是否生成刚刚上传的图片。



- **注意：**如果想指定网站访问对象存储中的图片，需要在存储桶中的**基础配置->跨域访问CORS设置**加自己网站的域名。

