

具体部署

因为我们的最终目的是部署flask，把模型做成一个标准restful API，并且使用python在本地笔记本上传一个照片，云主机推理，json返回结果。我们最重要的一步就是把代码放上服务器并且进行API的调用

云服务器与宝塔面板

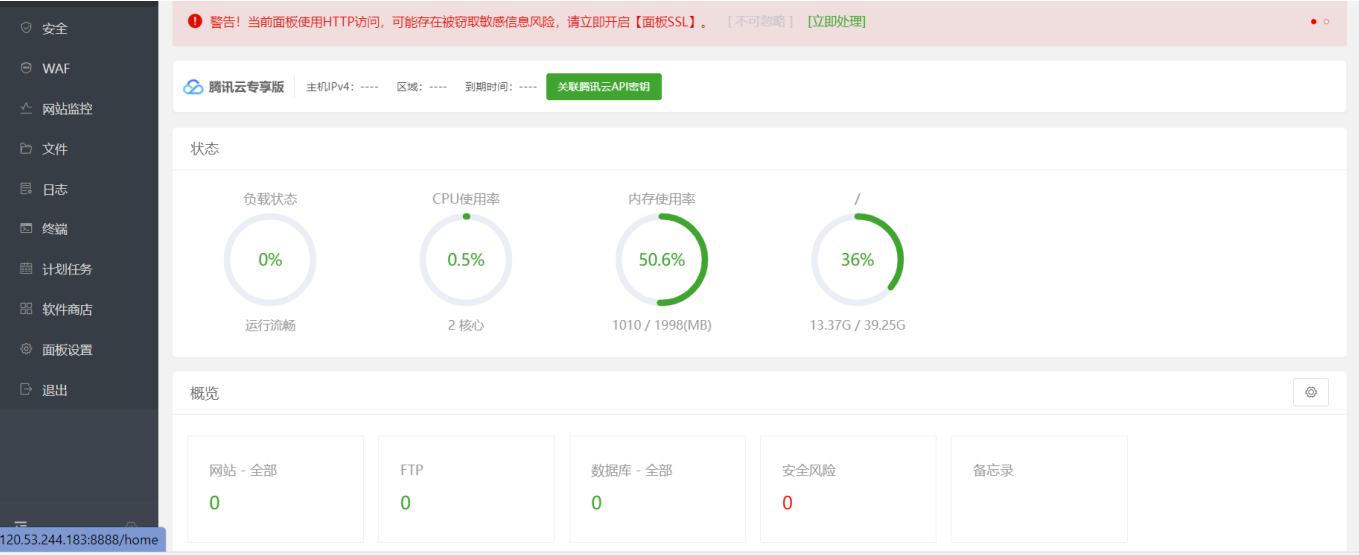
我们选择了腾讯云中的轻量应用服务器，并且进行了配置。

腾讯云的轻量应用服务器有一种比较简单的配置，如下图所示：



我们可以直接选择使用应用模板，选择宝塔面板。

宝塔面板有很多优点：面板的好处，就是通过一个交互界面就能完成服务器的维护工作，比如更新系统，添加网站，修改设置等等，以前需要记住各种命令，现在通过面板点点按钮就可以了，省时省力。下面就是宝塔的面板：



但是我们在选择服务器后，并不会直接生成宝塔面板，先在应用管理中获取指令后就可以在打开控制台后选择OrcaTerm，并从中获得宝塔面板的IP地址：

```
[root@VM-16-4-centos ~]# sudo /etc/init.d/bt default
=====
BT-Panel default info!
=====
外网面板地址: http://120.53.244.183:8888/tencentcloud
内网面板地址: http://10.0.16.4:8888/tencentcloud
username: [REDACTED]
password: [REDACTED]
Warning:
If you cannot access the panel,
release the following port (8888|888|80|443|20|21) in the security group
注意: 初始密码仅在首次登录面板前能正确获取, 其它时间请通过 bt 5 命令修改密码
=====
[root@VM-16-4-centos ~]#
```

打开外网的网站进行注册登陆后就可以使用宝塔面板了。

面板打开之后, 先进行配置。在软件商店里找到Python项目管理器下载, 之后的API上云需要进行应用。

项目部署

代码修改

在代码部署之前, 先需要对代码进行修改。

这是修改过后的代码块:

```
import json
from flask_restful import reqparse, Resource, Api
from werkzeug.datastructures import FileStorage
from flask import Flask
from ultralytics import YOLO
import cv2
from gevent import pywsgi

# from wsgiref.simple_server import make_server

app = Flask(__name__)

model = YOLO("/www/wwwroot/ultralytics-main/best.pt")
# model = YOLO('best.pt')

def pred(im):
    results = model.predict(source=im, save=False, save_txt=False)
```

```

cls = results[0].boxes.cls.numpy().tolist()
conf = results[0].boxes.conf.numpy().tolist()
xywh = results[0].boxes.xywh.numpy().tolist()

r = {'r': []}
names = results[0].names

for i, j, k in zip(cls, conf, xywh):
    r['r'].append({'cls': names[i], 'conf': float(j), 'xywh': [float(kk) for kk
in k]})

return r

class UploadImg(Resource):
    def __init__(self):
        self.parser = reqparse.RequestParser()
        self.parser.add_argument('imgFile', required=True, type=FileStorage,
location='files', help="imgFile is wrong.")

    def post(self):
        img_file = self.parser.parse_args().get('imgFile')
        img_file.save(img_file.filename)
        rr = pred(img_file.filename)
        return json.dumps(rr), 201

api = Api(app)
api.add_resource(UploadImg, '/uploadimg')

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=False)

```

这个代码首先将

```
app = Flask(__name__)
```

放在了定义函数的最上面，这样的话是定义了一个 Flask 应用程序对象app，并且在全局声明，它在文件的顶层被创建。这样做可以确保在整个应用程序中都可以访问到 app 对象。

如果放在if语句里面，则确保只有当当前脚本作为主程序运行时才会创建 Flask 应用程序对象，并且由外部访问时会造成错误无法访问到app。这种情况下在外部进行访问会造成错误如下：

```
[2024-02-19 21:41:36 +0800][23829][INFO] Starting gunicorn 21.2.0
[2024-02-19 21:41:36 +0800][23829][INFO] Listening at: http://0.0.0.0:808 (23829)
[2024-02-19 21:41:36 +0800][23829][INFO] Using worker: gthread
[2024-02-19 21:41:36 +0800][23830][INFO] Booting worker with pid: 23830
[2024-02-19 21:41:41 +0800][23830][INFO] Worker exiting (pid: 23830)
[2024-02-19 21:41:42 +0800][23829][ERROR] Worker (pid:23830) exited with code 4
[2024-02-19 21:41:42 +0800][23829][ERROR] Shutting down: Master
[2024-02-19 21:41:42 +0800][23829][ERROR] Reason: App failed to load.
```

```
Failed to find attribute 'app' in 'app'.
Failed to find attribute 'app' in 'app'.
Failed to find attribute 'app' in 'app'.
```

同时要注意需要将运行的Flask文件名称设置为 app.py，方便我们后面进行项目创建时gunicorn进行寻找。gunicorn一般的默认运行命令为：

```
gunicorn -w 4 -b 0.0.0.0:80 app:app
```

- -w 是指定了启动的 worker 进程数量
- -b 是指定了需要监听的主机和端口号
- app:app 这个参数指定了 Gunicorn 所要加载的 Flask 应用程序对象。第一个 app 表示要加载的模块或者包的名称，这里是 app.py 文件。第二个 app 是 Flask 应用程序对象的名称，这里是 app。post 否则则会

```
Failed to find attribute 'app' in '111'.
Failed to find attribute 'app' in '111'.
Failed to find attribute 'app' in '111'.
Failed to find attribute 'app' in '111'.
Failed to find attribute 'app' in '111'.
```

报错：

配置完成后，我们之后进行了测试，首先先简单介绍一下http的请求种类与常见的网页报错：

- HTTP请求的方法：
 - GET:从服务器上获取资源
 - POST: 在服务器上新创建一个资源
 - PUT: 在服务器上更新资源。(客户端提供所有改变后的数据)
 - PATCH: 在服务器上更新资源。(客户端只提供需要改变的属性)
 - DELETE: 从服务器上删除资源
- 常用状态码：
 - 200 服务器成功响应客户端的请求
 - 400 用户发出的请求有错误，服务器没有进行新建或修改数据的操作
 - 404 用户发送的请求的url不存在
 - 406 用户请求不被服务器接受(比如服务器期望客户端发送某个字段，但是没有发送)
 - 500 服务器内部错误，比如出现了bug

但是我们在部署完毕用postman进行post的时候，他的回应却是404，说明我们找不到这个url，可能是我们网络配置的问题，在排查了接口，模块，代码错误之后，发现到了错误点：

```
api = Api(app)
api.add_resource(UploadImg, '/uploadimg')
```

这两条语句代码位置的问题：

- 如果这两句代码放在if里面：
 - 当 Python 解释器执行到这两行代码时，它们会立即执行，即使是在导入模块时也会执行。
 - 这意味着 Flask 的路由和资源会在应用启动之前就已经被设置好了。
 - 这种方式适用于在模块被导入时就需要设置 Flask 应用的情况，比如在测试或者部署时，你希望 Flask 应用在模块导入时就开始运行。
- 如果这两句代码放在if外面：
 - 当将这两行代码放在 if __name__ == '__main__': 内部时，它们只有在当前脚本被直接运行时才会执行。
 - 意味着 Flask 的路由和资源会在应用被直接运行时设置
 - 这种方式适用于当你希望在特定条件下运行 Flask 应用，比如当作为主程序直接运行时，而不是在模块被导入时运行。

将他放在if外面之后就可以检查到该url了。

项目创建

我们在配置的时候下载了python项目管理器，我们将会用这个软件进行项目的配置与部署。

首先选择版本管理，下载python，我们的版本是3.9.7，python的版本最好在3.8以上，否则有些库会与环境不匹配而报错。

然后选择flask框架，在启动方式上注意要选择gunicorn。

Gunicorn 是一个 Python 的 WSGI HTTP 服务器。它所在的位置通常是在反向代理（如 Nginx）或者 负载均衡（如 AWS ELB）和一个 web 应用（比如 Django 或者 Flask）之间。它是一个移植自Ruby的 Unicorn项目的pre-fork worker模型，即支持eventlet也支持greenlet。

而不能选择python，如果选择python的话就像在电脑本地运行的一样，也不是运行在云服务器上，并且在运行参数上选择所指定的端口。然后就可以确定生成项目了。

项目生成之后：

| 项目名 | 项目路径 | 端口 | 启动方式 | Python版本 | CPU | 内存 | 状态 | 开机启动 | 守护进程 | 操作 |
|------|------------------|------|----------|----------|-----|-----------|-------|------|------|-----------------------------|
| 5000 | /www/wwwroot/... | 5000 | gunicorn | 3.9.7 | 0% | 805.16 MB | 运行中 ▶ | 开启 | 未设置 | 日志 映射 重启 配置 模块 删除 |
| yuu | /www/wwwroot/... | 888 | gunicorn | 3.9.7 | 0% | 39.77 MB | 运行中 ▶ | 开启 | 未设置 | 日志 映射 重启 配置 模块 删除 |

点击模块，在模块里寻找以下需求包：

```
ultralytics
flask_restful
opencv-python-headless #先卸载opencv-python
urllib3 #一定要选择1.26.15版本，下载失败的话，把版本号删掉再添加一次
```

然后点击重新运行，可以查看日志看端口是否已经被监听。

python发送post请求

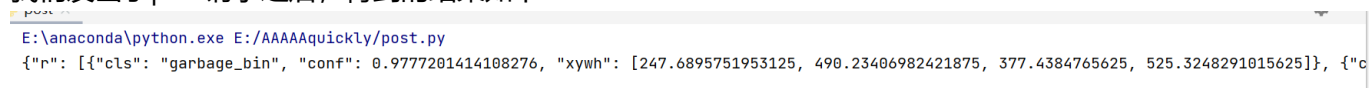
下面只需要写一个python代码进行api的调用就行，代码如下：

```
import requests
url = 'http://120.53.244.183:5000/uploadimg'
files = {'imgFile': open('1.jpg', 'rb')}
response = requests.post(url, files=files)
print(response.json())
```

url就是我们云服务器的外网网址，5000就是定义端口，uploadimg就是我们所定义的功能路径，我们定义了文件发送的字典files，'imgFile' 是表单中文件字段的名称，'1.jpg' 是要上传的文件名，通过二进制读取模式打开文件，以便将其作为文件对象传输到服务器端。然后获取回应，并将其打印成json格式

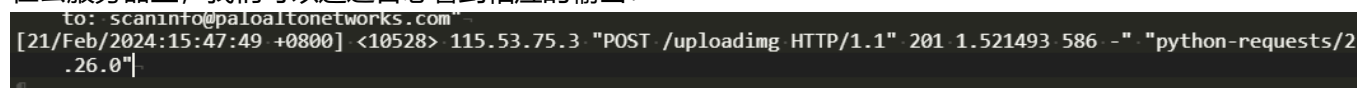
成果展示

我们发出了post请求之后，得到的结果如下：



```
E:\anaconda\python.exe E:/AAAAAquickly/post.py
{"r": [{"cls": "garbage_bin", "conf": 0.9777201414108276, "xywh": [247.6895751953125, 490.23406982421875, 377.4384765625, 525.3248291015625]}, {"c
```

在云服务器上，我们可以通过日志看到相应的输出：



```
to: scaninfo@paloaltonetworks.com"
[21/Feb/2024:15:47:49 +0800] <10528> 115.53.75.3 "POST /uploadimg HTTP/1.1" 201 1.521493 586 -" python-requests/2.26.0"
```

图片中可以看到输出，显示了201，表示成功。