

Flask项目部署Pythonanywhere中

• 一、部署前的准备

对于某些配置，生产环境下需要使用不同的值。为了让配置更加灵活，我们把需要在生成环境下使用的配置改为优先从环境变量中读取，如果没有读取到，则使用默认值：

```
# app.py 修改配置信息

app.config['SECRET_KEY'] = os.getenv('SECRET_KEY', 'dev')
app.config['SQLALCHEMY_DATABASE_URI'] = prefix +
os.path.join(os.path.dirname(app.root_path), os.getenv('DATABASE_FILE', 'data.db'))
```

注意 像密钥这种敏感信息，保存到环境变量中要比直接写在代码中更加安全。

wsgi.py: 手动设置环境变量并导入程序实例

```
# 项目下创建wsgi.py文件
import os

from dotenv import load_dotenv

dotenv_path = os.path.join(os.path.dirname(__file__), '.env')
if os.path.exists(dotenv_path):
    load_dotenv(dotenv_path)

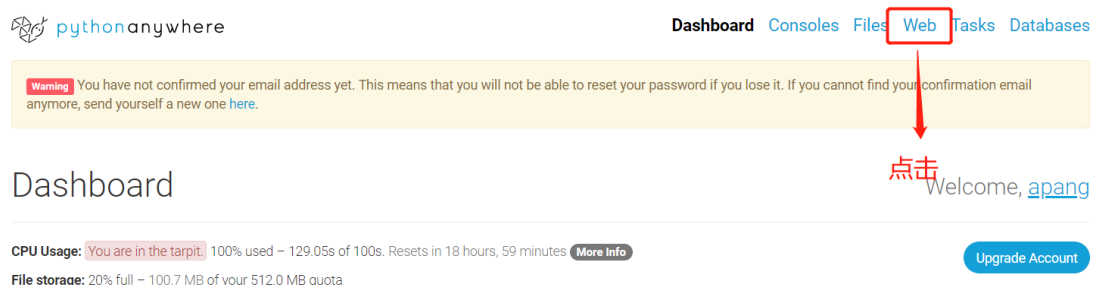
# app 属于应用名字 修改成你自己的应用名
from app import app
```

最后提交到git仓库中

```
$ git add .
$ git commit -m "项目部署环境准备工作"
$ git push
```

• 二、使用 PythonAnywhere 部署程序

注册账号



首页展示这样的

导航栏包含几个常用的链接，可以打开其他面板：

- Consoles（控制台）：可以打开 Bash、Python Shell、MySQL 等常用的控制台
- Files（文件）：创建、删除、编辑、上传文件，你可以在这里直接修改代码
- Web：管理 Web 程序
- Tasks（任务）：创建计划任务
- Databases（数据库）：设置数据库，免费账户可以使用 MySQL



Warning You have not confirmed your email anymore, send yourself a new one [here](#).

apang.pythonanywhere.com

+ Add a new web app



点击

点击之后创建

Your web app's domain name

Your account doesn't support custom domain names, so your PythonAnywhere web app will live at `greyli.pythonanywhere.com`.


Want to change that? [Upgrade now!](#)

Otherwise, just click "Next" to continue.

Cancel

« Back

Next »



Select a Python Web framework

...or select "Manual configuration" if you want detailed control.

- » Django
 - » web2py
 - » Flask
 - » Bottle
 - » **Manual configuration** (including virtualenvs)
- 

What other frameworks should we have here? Send us some feedback using the link at the top of the page!

[Cancel](#)[« Back](#)[Next »](#)



Select a Python version

- » Python 2.7
- » Python 3.4
- » Python 3.5
- » Python 3.6

Cancel

« Back

Next »

选择符合自己的python版本

Manual Configuration

Manual configuration involves editing your own WSGI configuration file in `/var/www/`. Usually this imports a WSGI-compatible application which you've stored elsewhere

When you click "Next", we will create a WSGI file for you, including a simple "Hello World" app which you can use to get started, as well as some comments on how to use other frameworks.

You will also be able to specify a *virtualenv* to use for your app.

[Cancel](#)[« Back](#)[Next »](#)

创建完成

• 三、初始化程序运行环境

可以通过GitHub远程连接克隆项目到当前

```
$ pip3 install --user pipenv # 安装 Pipenv
$ git clone https://github.com/apang-ai/flask1901 # 注意替换 Git 仓库地址
$ cd flask1901/day1/watchlist # 切换到项目目录中去
```

下面我们在项目根目录创建 `.env` 文件，并写入生产环境下需要设置的两个环境变量。其中，密钥（`SECRET_KEY`）的值是随机字符串，我们可以使用 `uuid` 模块来生成：

```
$ python3
>>> import uuid
>>> uuid.uuid4().hex
'3d6f45a5fc12445dbac2f59c3b6c7cb1'
```

复制生成的随机字符串备用，接着创建 `.env` 文件：

```
nano .env
```

写入设置密钥和数据库名称的环境变量：

```
SECRET_KEY=3d6f45a5fc12445dbac2f59c3b6c7cb1 # 刚才自己生成的随机字符串
DATABASE_FILE=data-prod.db
```

最后安装依赖并执行初始化操作：


```
$ pipenv install # 创建虚拟环境并安装依赖
$ pipenv shell # 激活虚拟环境
$ flask initdb # 初始化数据库
$ flask admin # 创建管理员账户
```

在浏览器的新标签页打开 Web 面板

• 四、代码，设置并启动

Code:

What your site is running.

Source code:	/home/apang/flask1901/day1/watchlist
Working directory:	/home/apang/flask1901/day1/watchlist
WSGI configuration file:	/var/www/apang-pythonanywhere_com_wsgi.py
Python version:	3.8 

都写到项目的文件夹名字

点击源码 (Source code) 和工作目录 (Working directory) 后的路径并填入项目根目录，目录规则为“/home/用户名/项目文件夹名”。

点击 WSGI 配置文件 (WSGI configuration file) 后的链接打开编辑页面，删掉这个文件内的所有内容，填入下面的代码：

```
import sys

path = '/home/apang/flask1901/day1/watchlist' # 路径规则为 /home/你的用户名/项目文件夹名
if path not in sys.path:
    sys.path.append(path)

from wsgi import app as application
```

• 五、虚拟环境

为了让程序正确运行，我们需要在 Virtualenv 部分填入虚拟环境文件夹的路径：

Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

Enter path to a virtualenv, if desired

使用 Pipenv 时，你可以在项目根目录下使用下面的命令获取当前项目对应的虚拟环境文件夹路径（返回前面打开的命令行会话输入下面的命令）：

```
pipenv --venv
```

复制输出的路径，点击 Virtualenv 部分的红色字体链接，填入并保存。


• 六、静态文件

静态文件可以交给 PythonAnywhere 设置的服务器来处理，这样会更高效。

如图所示：

Static files:

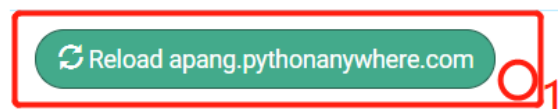
Files that aren't dynamically generated by your code, like CSS, JavaScript or uploaded files, can be served much faster straight off the disk if you specify them here. You need to **Reload your web app** to activate any changes you make to the mappings below.

URL	Directory	Delete
/static/	/home/apang/flask1901/day1/watchlist/static	
Enter URL	Enter path	

• 七、启动

Configuration for [apang.pythonanywhere.com](#)

Reload:



2
按照顺序就可以访问了

• 八、更新部署后的程序

当你需要更新程序时，流程和部署类似。在本地完成更新，确保程序通过测试后，将代码推送到 GitHub 上的远程仓库。登录到 PythonAnywhere，打开一个命令行会话（Bash），切换到项目目录，使用 git pull 命令从远程仓库拉取更新：

```
$ cd watchlist
$ git pull
```

然后你可以执行一些必要的操作，比如安装新的依赖等等。最后在 Web 面板点击绿色的重载（Reload）按钮即可完成更新。