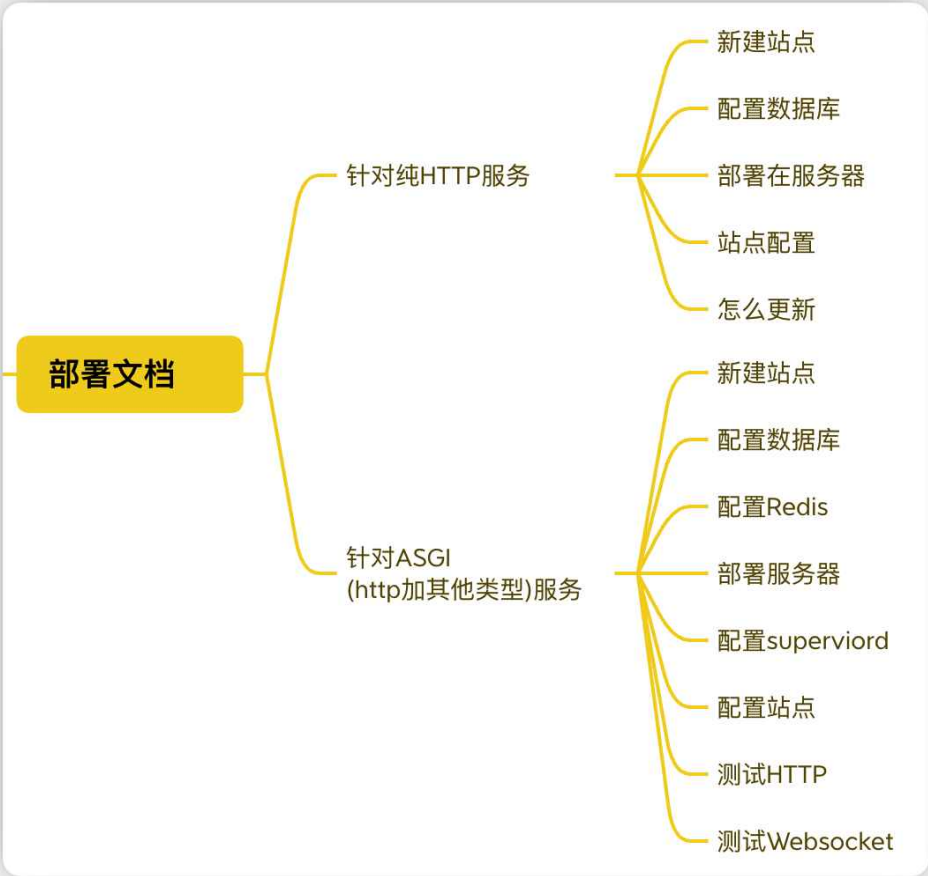


部署文档

📌 部署前必看ho!!!



前言

在项目完成后，我们需要部署至服务器以作为后端与前端进行最后的对接、测试，最后成为正式系统的一部分

部署在服务器中有两种，一种是普通的提供纯HTTP作为api接口承载的项目，比如存量系统，优惠券计算，银行支付系统，这一些都是以RESTful API的形式对外提供服务的，我们就可以用宝塔面板的Python项目管理器来帮助我们进行项目的部署。

Python项目管理器										
项目管理	添加项目									
版本管理										
日志										
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	操作	
time	/www/time_deploy/	6688	gunicorn	3.8.5	0%	47.2 MB	运行中 ▶	开启	取消映射	重启 配置 模块 删除
stock	/www/wwwroot/stock.laixinly...	577	uwsgi	3.8.5	0%	189.93 MB	运行中 ▶	开启	映射	重启 配置 模块 删除

key	/www/wwwroot/rmyydjango...	9982	uwsgi	3.8.5	0%	140.74 MB	运行中 ▶ 开启	映射 重启 配置 模块 删除
123	/www/wwwroot/stock.laixinl...	1999	uwsgi	3.8.5	0%	189.93 MB	运行中 ▶ 开启	映射 重启 配置 模块 删除
ceshi	/www/wwwroot/django-test.l...	995	uwsgi	3.8.5	0%	134.25 MB	运行中 ▶ 开启	映射 重启 配置 模块 删除
test	/www/wwwroot/django.laixinl...	8897	uwsgi	3.8.5	0%	144.62 MB	运行中 ▶ 开启	映射 重启 配置 模块 删除
weather	/www/wwwroot/爬虫/	123	gunicorn	3.8.5	0%	47.66 MB	运行中 ▶ 开启	取消映射 重启 配置 模块 删除
face_t...	/www/wwwroot/珠海横琴人脸...	543	gunicorn	3.8.5	0%	61.85 MB	运行中 ▶ 开启	映射 重启 配置 模块 删除

管理器默认使用pip安装项目根目录requirements.txt内的模块，如有其他模块需要安装请手动进入虚拟环境安装

进入虚拟环境方法：

在命令行输入 `source 项目路径/项目名_venv/bin/activate`

如： `source /data/python/project1_venv/bin/activate`

第二种则是比较复杂的一种，需要在提供http服务的同时需要提供一些异步的服务,比如人脸机后台，与设备的交互采用了websocket的方式，因此在部署时需要采用ASGI的方式来接管，在步骤上更多是在终端中进行，但是稳定性与第一种都是一致的。

部署前的准备

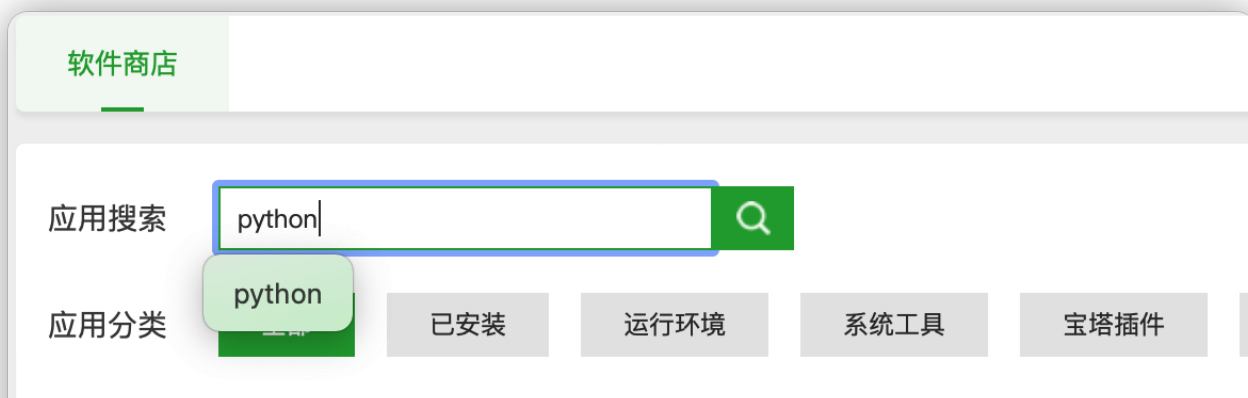
！ 在部署前，应该详细地在本地测试好功能与各项应该进行的稳定性测试。

- ☐ 一台安装了宝塔的服务器
- ☐ 服务器系统采用centos7.0
- ☐ 部署文件已经push到git仓库中

纯HTTP部署

安装Python项目管理插件

我们在左边的‘软件商店’里选择‘宝塔插件’，找到‘Python项目管理’这个插件进行安装



立即购买

Linux专业版优势

客服QQ1: 3007255432

客服QQ2: 3007255432

✓ 专业版插件

✓ 15天无理由退款

✓ 可更换IP

✓ 低至1.52元/月

最近使用入口

Python项目管理器

Docker管理器

Linux工具箱

宝塔面板

软件名称	开发商	说明
Python项目管理器 1.9	官方	快速部署Python项目，当前仅支持Centos7.x，可部署框架

设置Python版本

在Python项目管理器中选择版本管理，Python版本选择3.8.5（你本地对应的版本）

Python项目管理器

项目管理

版本管理

日志

Python版本:

3.8.5 [已安装]

• 安装python版本

3.8.5 [已安装]

3.9.0b5 [未安装]

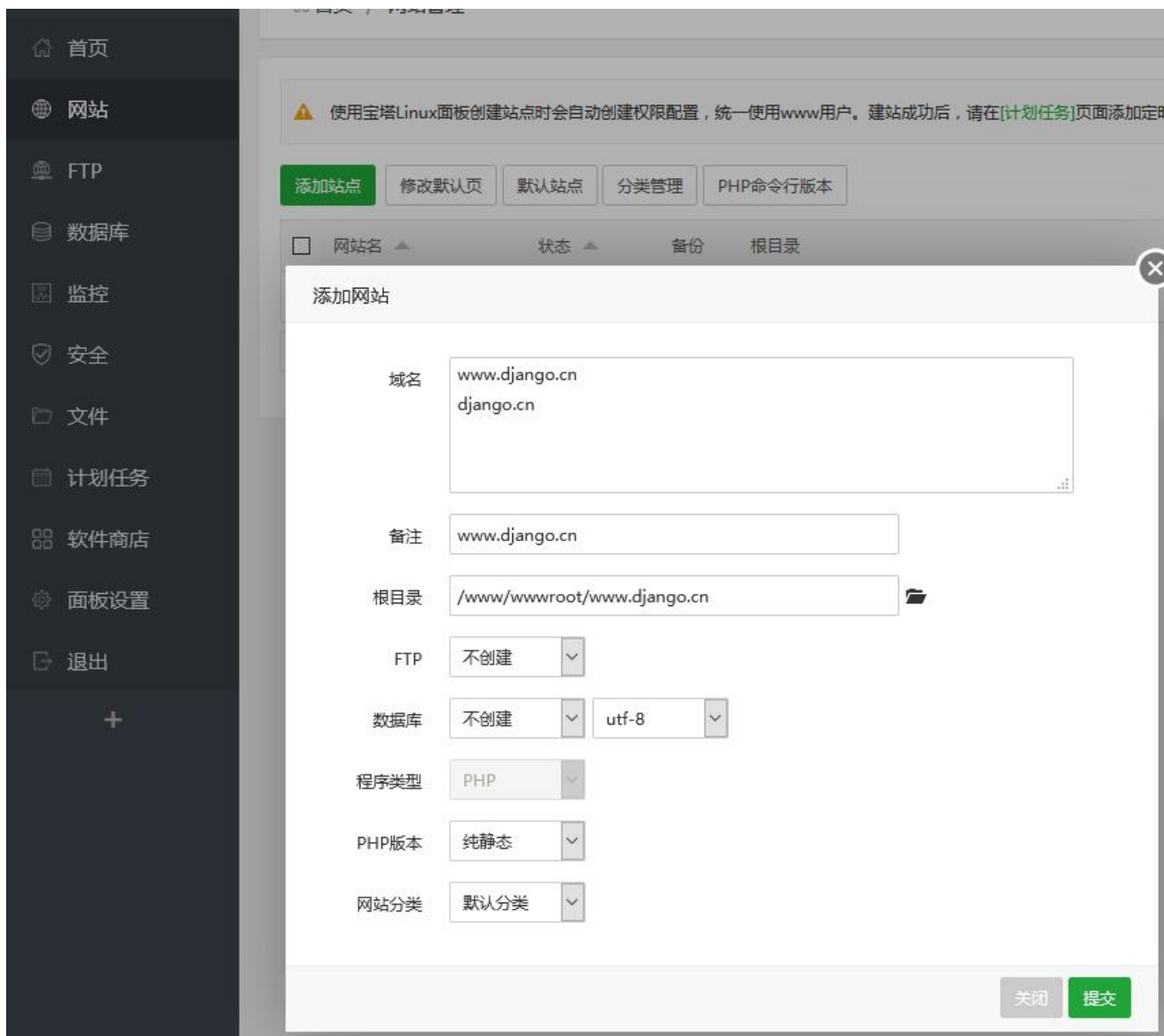
3.9-dev [未安装]

3.8.4 [未安装]

卸载版本

开始部署

添加项目站点



域名处填写自己的域名即可，没有域名填写服务器IP地址。

上传Django项目源码（本地上传，没有用到git）

！ 留意：在打包项目源码之前，先在本地环境使用下面的命令把环境依赖包导出到 requirements.txt 文件里，并把这个文件存放在项目目录下，这一步骤非常重要，请务必记得操作。

导出命令：

Bash

```
1 pip3 freeze > requirements.txt
```

这一步操作好之后，我们就通过下面的步骤把项目源码上传上到服务器上去。

⚠ 使用宝塔Linux面板创建站点时会自动创建权限配置，统一使用www用户。建站成功后，请在[计划任务]页面添加定时备份任务!

添加站点

修改默认页

默认站点

分类管理

PHP命令行版本

点击这里

<input type="checkbox"/>	网站名 ▲	状态 ▲	备份	根目录
<input type="checkbox"/>	www.django.cn	运行中 ▶	无备份	/www/wwwroot/www.django.cn

← 根目录 > www > wwwroot > www.django.cn >

上传

远程下载

新建 ▼

←

↻

📁

分享列表

收藏夹 ▼

📁 根目录(14G)

<input type="checkbox"/>	文件名 ▼	大小
<input type="checkbox"/>	 .htaccess PS: Apache用户配置文件(伪静态)	1 B
<input type="checkbox"/>	 .user.ini PS: PHP用户配置文件(防跨站)!	53 B
<input type="checkbox"/>	 404.html	479 B
<input type="checkbox"/>	 index.html	955 B

2、点击上传

1、删除这两个文件

上传文件到[/www/wwwroot/www.django.cn/] --- 支持断点续传

选择文件

选择目录

myblog.zip

5.20 MB

等待上传















解压项目



解压成功之后，就像下面那样。留意路径，记得源码一定要解压到根目录里。然后再检查一下，项目里有没有requirements.txt这个文件。



<input type="checkbox"/> 文件名 ▼	大小
<input type="checkbox"/>  DjangoUeditor	点击计算
<input type="checkbox"/>  blog	点击计算
<input type="checkbox"/>  media	点击计算
<input type="checkbox"/>  myblog	点击计算
<input type="checkbox"/>  static	点击计算
<input type="checkbox"/>  templates	点击计算
<input type="checkbox"/>  .htaccess PS: Apache用户配置文件(伪静态)	1 B
<input type="checkbox"/>  .user.ini PS: PHP用户配置文件(防跨站)	53 B
<input type="checkbox"/>  db.sqlite3	372.00 KB
<input type="checkbox"/>  manage.py	553 B
<input type="checkbox"/>  myblog.zip	5.20 MB
<input type="checkbox"/>  requirements.txt	44 B

! 这里为了演示，没有使用Git来管理代码，使用Git来管理需要在终端中首先绑定Git仓库

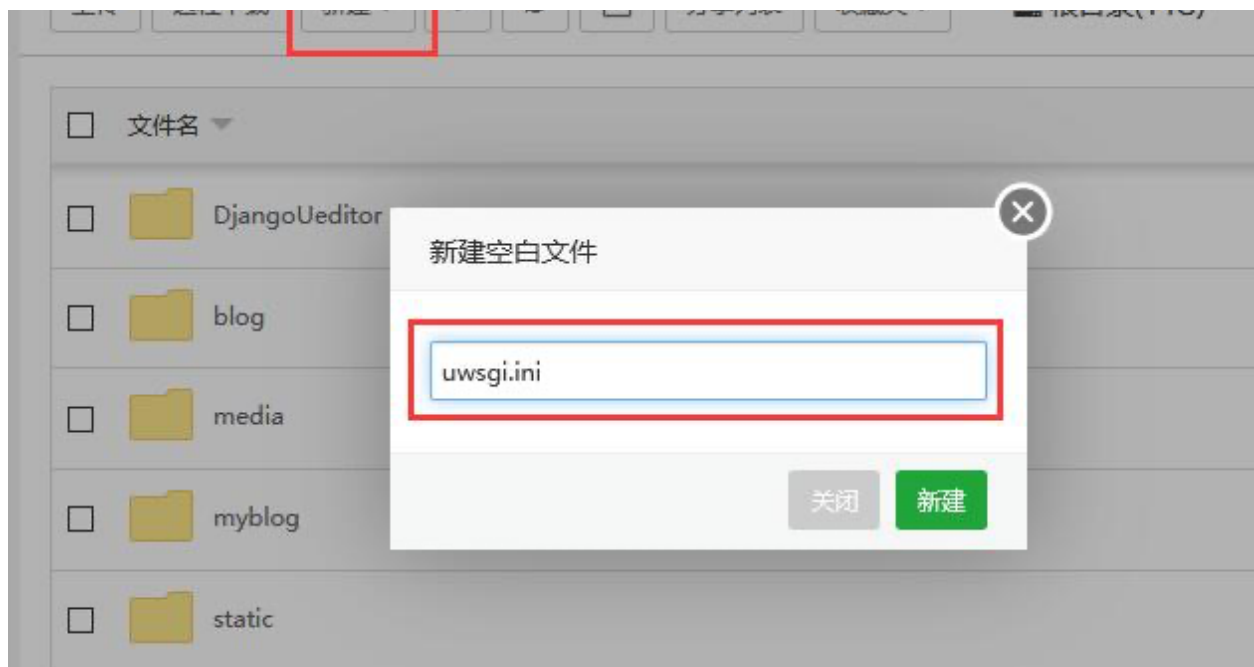
关联仓库到本地

Apache

```
1 git remote add origin https://gitee.com/hamster1963/django-test.git
2 git pull origin master --allow-unrelated-histories
```

添加uwsgi配置文件uwsgi.ini



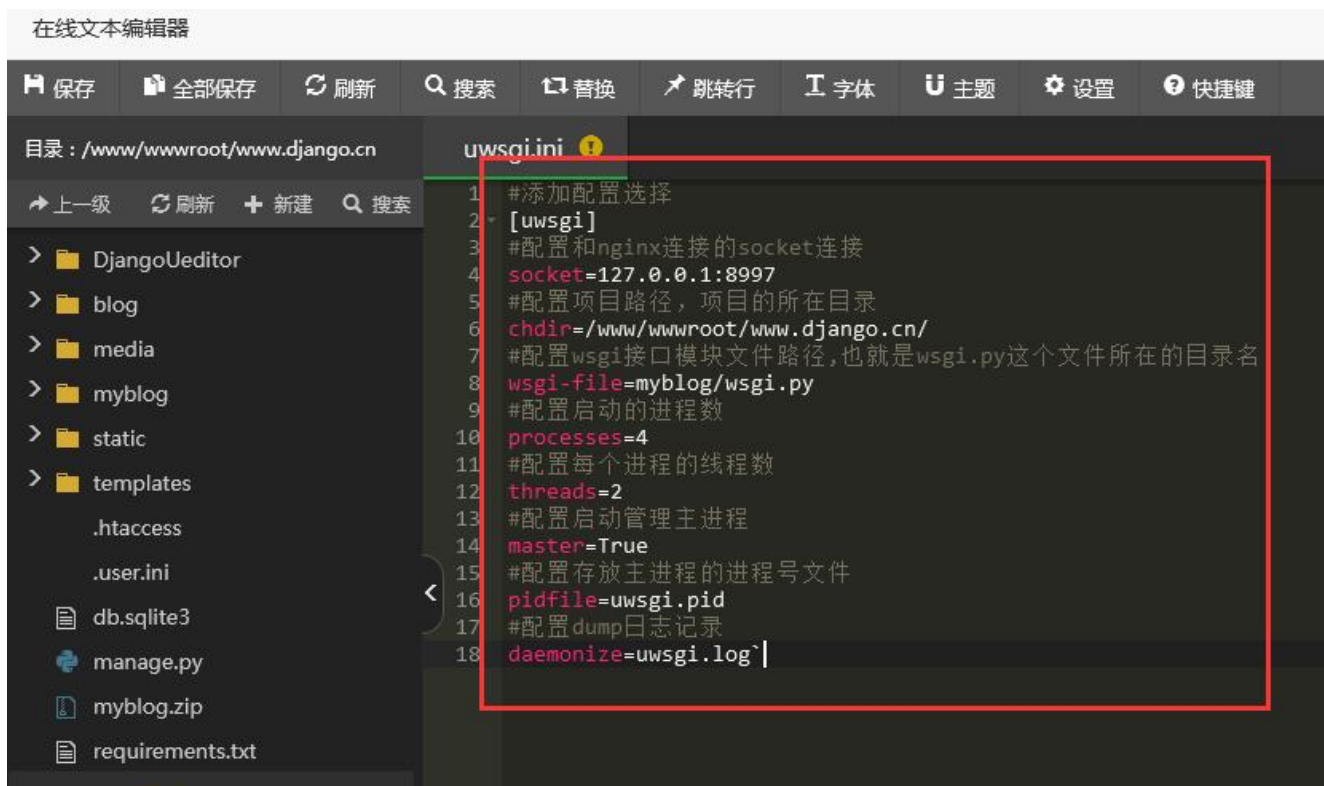


留意：新建一个空白文件，文件名为uwsgi.ini。新建成功之后输入如下代码，然后保存：

Plain Text

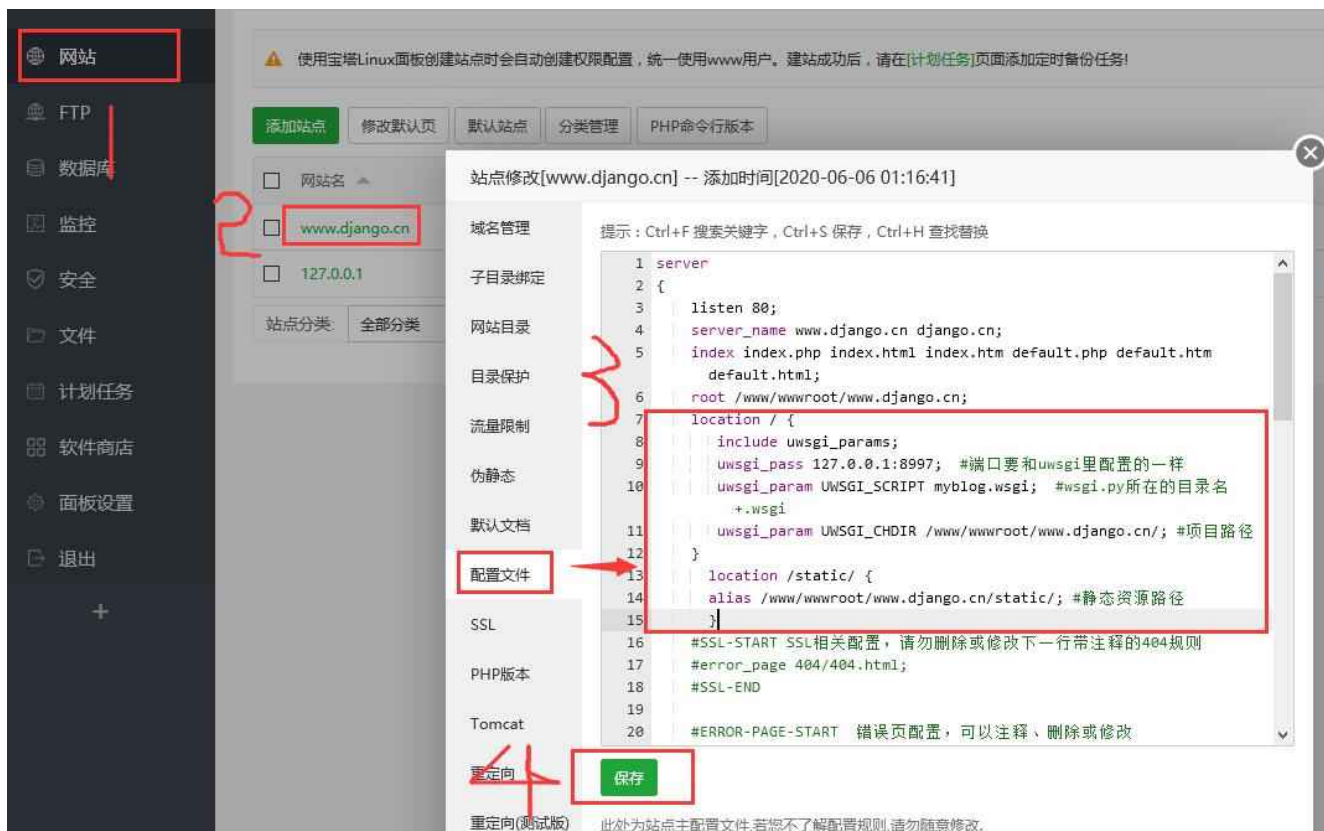
```
1 #添加配置选择
2 [uwsgi]
3 #配置和nginx连接的socket连接
4 socket=127.0.0.1:577
5 #配置项目路径，项目的所在目录
6 chdir=/www/wwwroot/stock.laixinly.top/
7 #配置wsgi接口模块文件路径,也就是wsgi.py这个文件所在的目录
8 wsgi-file=Stock_System/wsgi.py
9 #配置启动的进程数
10 processes=4
11 #配置每个进程的线程数
12 threads=2
13 #配置启动管理主进程
14 master=True
15 #配置存放主进程的进程号文件
16 pidfile=uwsgi.pid
17 #配置dump日志记录
18 daemonize=/www/wwwroot/stock.laixinly.top/logs/error.log
```


！ 这里面，我们最需要留意的是项目路径和wsgi.py所在的目录。



修改网站配置

左侧网站，然后点击网站名，在弹出的窗口里找到'配置文件'，然后配置文件里输入如下代码：

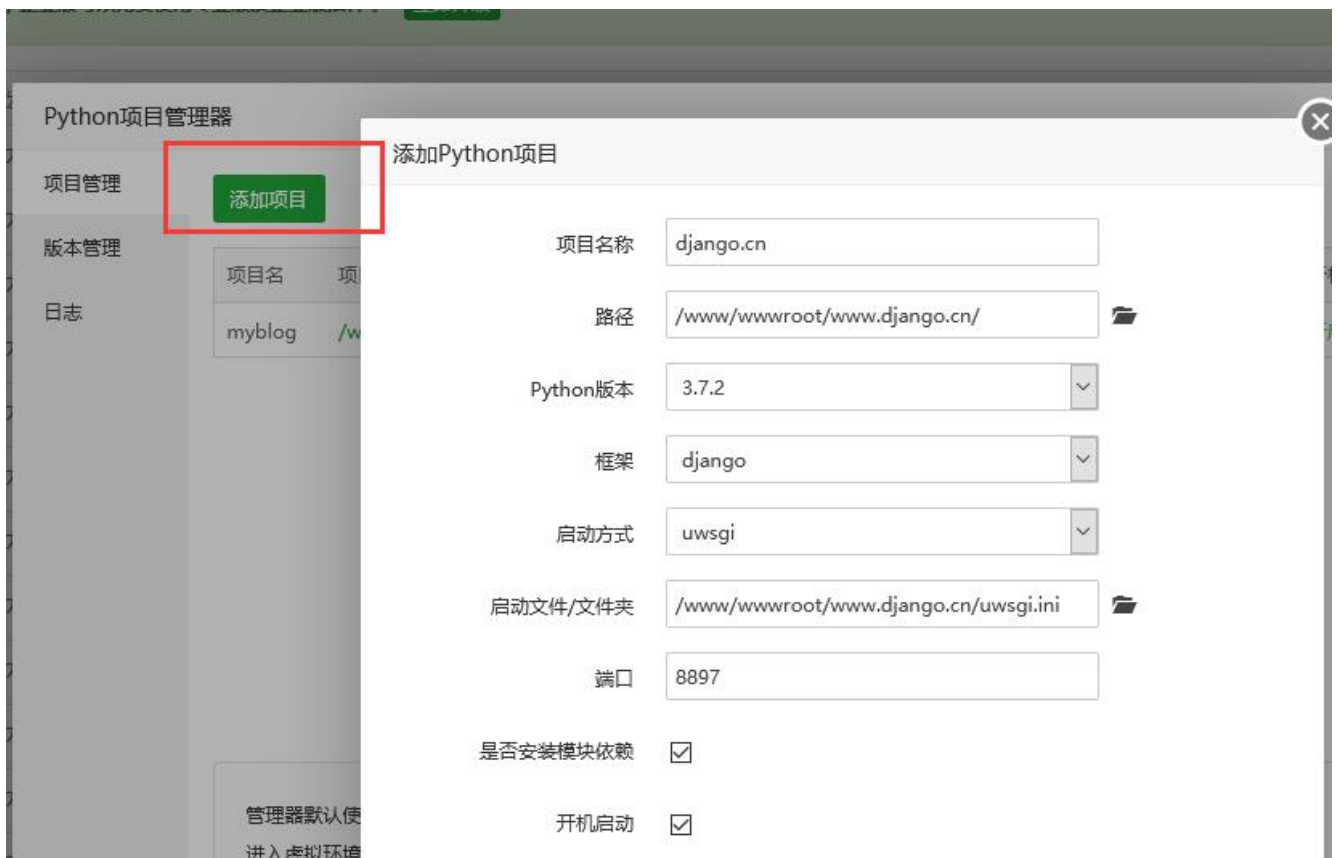


Nginx

```
1     location / {
2         include uwsgi_params;
3         uwsgi_pass 127.0.0.1:8997; #端口要和uwsgi里配置的一样
4         uwsgi_param UWSGI_SCRIPT myblog.wsgi; #wsgi.py所在的目录名+.wsgi
5         uwsgi_param UWSGI_CHDIR /www/wwwroot/www.django.cn/; #项目路径
6     }
7     location /static/ {
8         alias /www/wwwroot/www.django.cn/static/; #静态资源路径
9     }
```

Python项目管理插件里添加项目

在左右的软件商店里找到Python项目管理插件，然后点击设置，添加项目。里面的各种选项很简单，按实际情况填写就行。



！ 值得说的就是那个端口，端口要和uwsgi.ini里面的那个端口一致。如果有多个项目的话，不同的项目要填写不同的端口。端口随便填写，只要不与其它常用软件端口冲突就好。



以上配置完成之后，基本已经部署完毕。可以使用apipost来测试我们部署的接口

如果发现项目没有成功运行，尝试进入虚拟环境后运行python3 manage.py或查看日志，可以更快定位到问题。

留意：

使用Python项目管理插件新建项目成功之后，会自动在项目源码目录里创建一个虚拟环境，虚拟环境目录一般都是以项目名_venv形式命名的。

进入虚拟环境方法：

在命令行输入 source 项目路径/项目名_venv/bin/activate 如：

Bash

```
1 source /www/wwwroot/myblog/myblog_venv/bin/activate
```

项目管理器默认使用pip安装项目根目录requirements.txt内的模块，这也是之前我强调把环境依赖包文件放到项目目录下的原因，如有其他模块需要安装请手动进入虚拟环境安装。

问题整理：

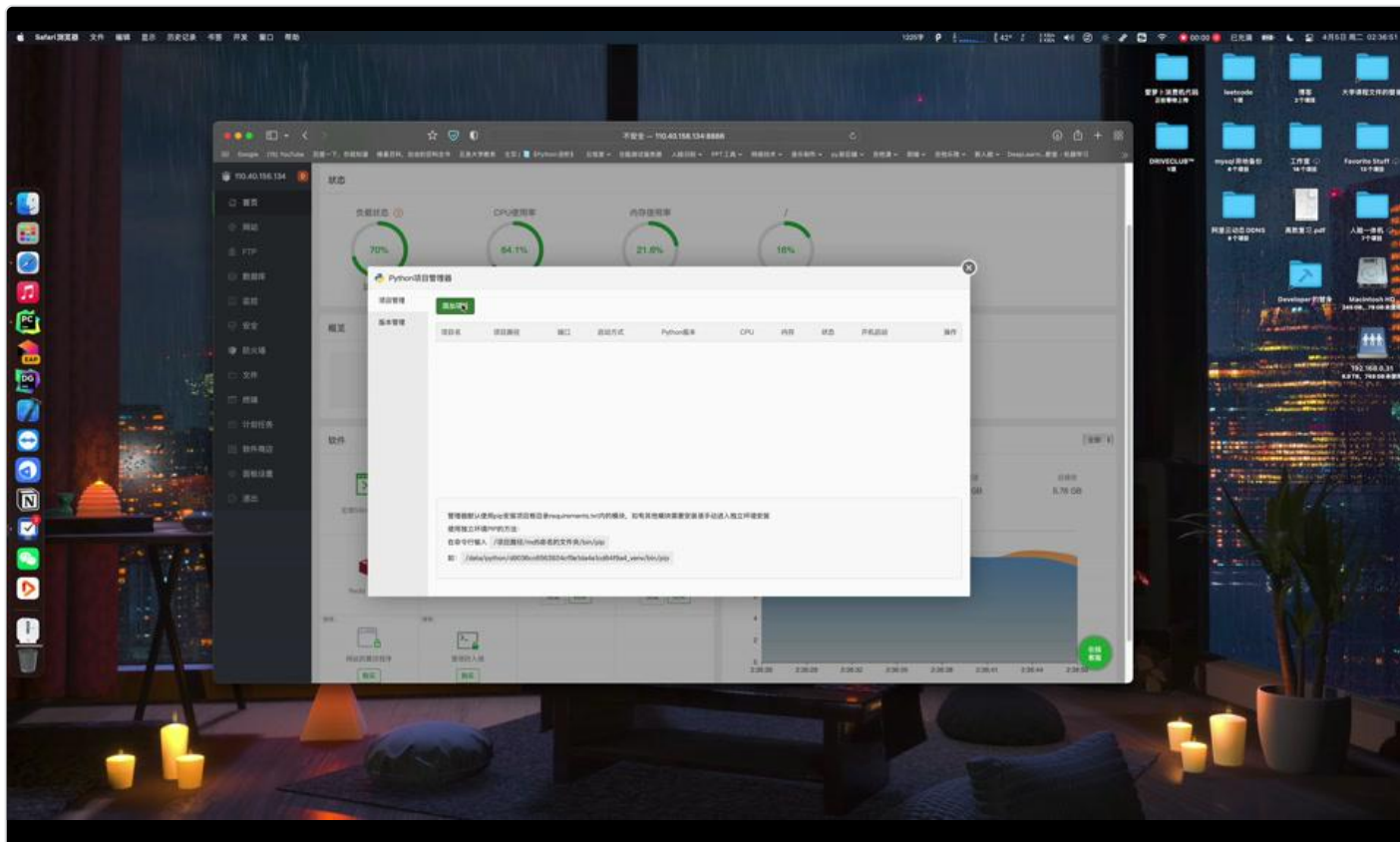
！ 访问显示出现 Internal Server Error 之类的错误的话，可能是程序不能正常运行的原因。请进入虚拟环境下，使用python manage.py runserver 命令运行项目，看项目能不能正常运行，不能正常运行就按错误提示进行解决就好。程序能正常运行使用项目管理器重启一下项目就能正常访问。

！ 如果出现：Django运行提示:SQLite 3.8.3 or later is required (found 3.7.17) 这样的错误，请按这篇文章操作。<https://www.django.cn/forum/forum-21090.html>

！ 如果在第十二步建立项目时提示出错，请把“是否安装模块依赖”这个选项去掉，等项目建立完成之后再进入虚拟环境手动安装依赖模块。（错误提示类似帖子后面留言处。）

某次项目的完整部署过程

(已经上传好文件，从配置开始进行)



iShot2022-04-05_02.36.49.mp4

ASGI部署

这是第二种部署方式,由于第一种uwsgi托管的方式无法满足异步的需求，所以需要用到Django3终于加入的ASGI服务器来进行托管(finally!).

当然这种部署方式会比第一种难一点，但是，顶住！



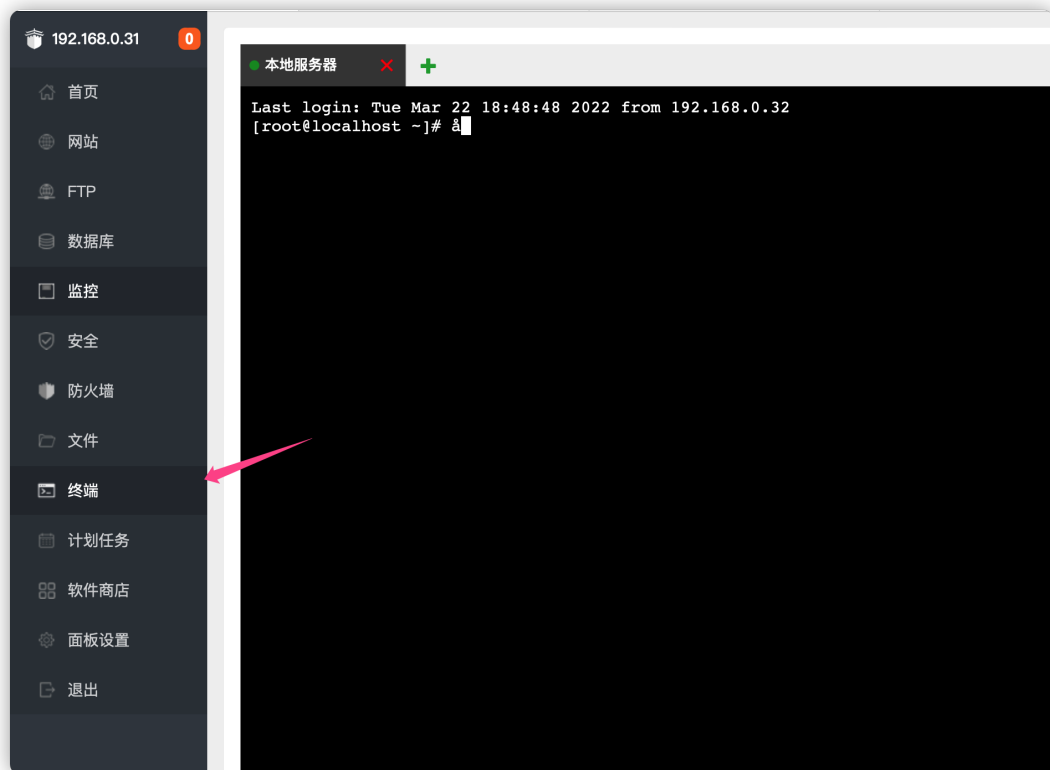
用到的工具

- ☐ 一台服务器
- ☐ 安装好Python3
- ☐ 可以ssh连接

SSH连接到服务器

可以用宝塔的终端，也可以用Putty

推荐还是宝塔的终端



安装Python

参考开发文档里的服务器部署 我这里就偷懒一下了.. (逃

介绍一下Daphne

采用daphne来托管我们的接口，[Daphne](#) 是一个纯Python编写的应用于UNIX环境的由Django项目维护的ASGI服务器。它扮演着ASGI参考服务器的角色。

安装 Daphne

可以通过 pip 来安装 Daphne

Bash

```
1 python3 -m pip install daphne
```

在 Daphne 中运行 Django

一旦 Daphne 安装完毕，你就可以使用 `daphne` 命令了，它将用来启动 Daphne 服务进程。在最简单的情形下，Daphne 加上包含一个 ASGI 应用模块的位置和应用的名称（以冒号分隔）。

Bash

```
1 daphne myproject.asgi:application
```

它将开启一个进程，监听 127.0.0.1:8000。这需要你的项目位于 Python path 上。为了确保这点，你应该在与 `manage.py` 文件相同的路径中运行这个命令。

如果需要更改运行端口，使用以下命令：

Bash

```
1 daphne myproject.asgi:application -b 0.0.0.0 -p 8000
```

！ 说明：-b 监听地址 -p 监听端口

本地实际项目运行

咱们的项目开发好了，但是直接使用daphne运行，会遇到以下错误：

比如1：

Bash

```
1 django.core.exceptions.ImproperlyConfigured: Requested setting LOGGING_CONFIG, but settings are not configured. You must either define the environment variable DJANGO_SETTINGS_MODULE or call settings.configure() before accessing settings.
```

比如2：

Bash

```
1 django.core.exceptions.AppRegistryNotReady: Apps aren't loaded yet.
```

解决方法

修改asgi.py，增加django.setup()

Bash

```
1  import osimport django
2  os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'websocket_demo.settings')
3  django.setup()
4
5  from channels.auth import AuthMiddlewareStack
6  from django.core.asgi import get_asgi_application
7  # Import other Channels classes and consumers here.
8  from channels.routing import ProtocolTypeRouter, URLRouter
9  # from apps.websocket_app.urls import websocket_urlpatterns
10 from websocket_demo.urls import websocket_urlpatterns
11
12 # application = get_asgi_application()
13 application = ProtocolTypeRouter({
14     # Explicitly set 'http' key using Django's ASGI application.
15     "http": get_asgi_application(),
16     'websocket': AuthMiddlewareStack(
17         URLRouter(
18             websocket_urlpatterns
19         )
20     ),
21 })
```

注意：django.setup()要置顶，不能在底部，否则使用daphne启动会报上面的错误。

运行项目

注意：要在manage.py同级目录下执行此命令

Bash

```
1  daphne websocket_demo.asgi:application -b 0.0.0.0 -p 8000
```

开始正式部署

nginx+daphne+supervise

官方文档：<https://channels.readthedocs.io/en/stable/deploying.html#configuring-the-asgi-application>

根据官方文档，推荐使用nginx+daphne+supervise

supervise

Bash

```
1 yum install -y supervisor
```

生成配置文件

Bash

```
1 echo_supervisord_conf > /etc/supervisord.conf
```

修改配置文件/etc/supervisord.conf，最后一行增加

Bash

```
1 [include]
2 files = supervisord.d/*.ini
```

表示配置文件读取supervisord.d目录下所有后缀为.ini的文件。

创建配置目录，并创建配置文件

Bash

```
1 mkdir /etc/supervisord.d/vi /etc/supervisord.d/asgi.ini
```

内容如下：

Bash

```
1 [fcgi-program:asgi]
2 # TCP socket used by Nginx backend upstream
3 socket=tcp://localhost:8000
4
5 # Directory where your site's project files are located
6 directory=/tmp/internal_tools
7
8 # Each process needs to have a separate socket file, so we use process_num
9 # Make sure to update "mysite.asgi" to match your project name
10 command=/virtualenvs/venv1/bin/daphne -u /run/daphne/daphne%
    (process_num)d.sock --fd 0 --access-log - --proxy-headers
    internal_tools.asgi:application
11
12 # Number of processes to startup, roughly the number of CPUs you have
13 numprocs=4
14
15 # Give each process a unique name so they can be told apart
16 process_name=asgi%(process_num)d
17
18 # Automatically start and recover processes
19 autostart=true
20 autorestart=true
21
22 # Choose where you want your log to go
23 stdout_logfile=/var/log/asgi.log
24 redirect_stderr=true
```

! 注意：红色部分，请根据实际情况修改。

启动supervisord

Bash

```
1 supervisord -c /etc/supervisord.conf
```

查看asgi运行状态

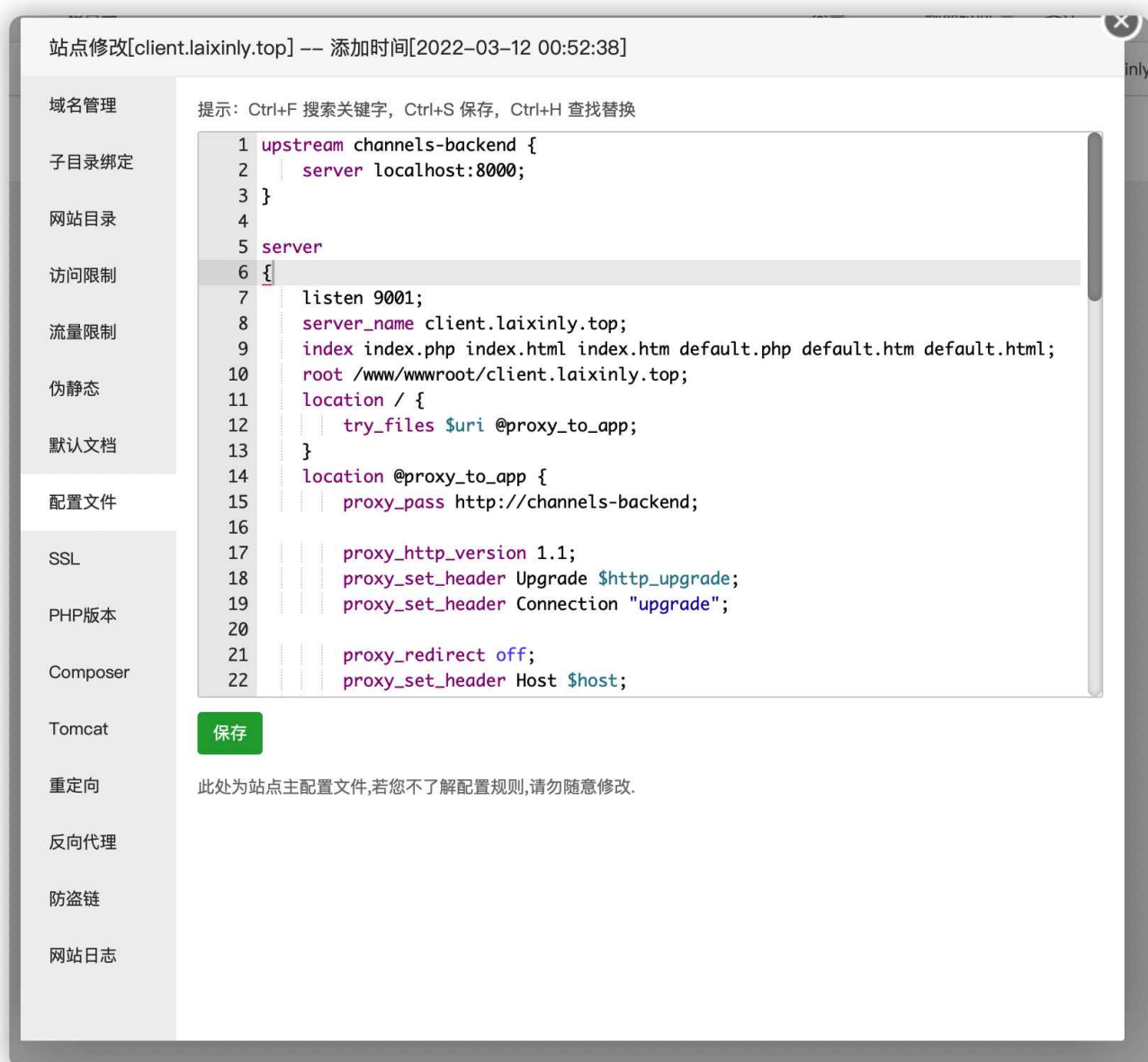
Bash

```
1 # supervisorctl
2 asgi:asgi0          RUNNING    pid 17567, uptime 0:00:04
3 asgi:asgi1          RUNNING    pid 17566, uptime 0:00:04
4 asgi:asgi2          RUNNING    pid 17569, uptime 0:00:04
5 asgi:asgi3          RUNNING    pid 17568, uptime 0:00:04
```

可以看到，有4个进程。如果状态不是RUNNING，请查看ini配置文件，是否正常。

修改网站配置

将配置添加入需要部署的站点的配置文件中



Perl

```
1 upstream channels-backend {
2     server localhost:8000;
3 }
4 server {
5     listen 8093;
6     location / {
7         try_files $uri @proxy_to_app;
8     }
9     location @proxy_to_app {
10        proxy_pass http://channels-backend;
11
12        proxy_http_version 1.1;
13        proxy_set_header Upgrade $http_upgrade;
14        proxy_set_header Connection "upgrade";
15
16        proxy_redirect off;
17        proxy_set_header Host $host;
18        proxy_set_header X-Real-IP $remote_addr;
19        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
20        proxy_set_header X-Forwarded-Host $server_name;
21    }
22 }
23
```

! 注意红色部分，upstream 是asgi的监听端口。在server里面的8093是对外端口，也可以改成别的，根据实际情况而定。

重启站点



点一下，再点一下，and！我们最难的部署就完成啦！！

其实，daphne不光可以处理asgi，它也可以处理wsgi，没有必要部署uswgi来处理wsgi了。

总之：nginx+daphne+supervise就可以处理django的所有功能了

测试

用apipost和Postman分别进行http接口与websocket服务的测试

