链滴

flask 部署

作者：zhuhonglin
原文链接：https://ld246.com/article/1500907924431
来源网站：链滴

介绍

主要介绍 flask + nginx + uwsgi + supervisor 在 centos 上的部署。其中，flask 和 uwsgi 使用 virtualenv 创建独立环境。

准备

本次搭建 python2 和 python3 都可以完成，不涉及冲突的部分。

首先需要安装 virtualenv，使用 pip 安装即可

```
pip install virtualenv
```

virtualenv 可以为应用构建一套“隔离”的 python 运行环境。

安装好后，自己设定一个目录，作为虚拟环境的主目录，比如我的目录/root/document/python/myapp。在这个目录里创建虚拟环境，并且激活当前 virtualenv

```
virtualenv venv
source ./venv/bin/activate
```

如果需要退出，直接输入 deactivate 就可以退出虚拟环境。

这一步结束，你会发现在 myapp 这个目录中出现了一个 venv 的目录。

之后在虚拟环境中安装flask 和 uwsgi

```
pip install flask
pip install uwsgi
```

安装完成之后，venv 下的 bin 目录中就会出现名为 uwsgi 和 flask 的文件

nginx 的安装就不介绍了，没遇到什么问题。

最后是 supervisor 的安装，因为是centos，所以使用了 yum

```
yum install python-setuptools
easy_install supervisor
```

这一步如果出现问题的话，多半是 yum 和 easy_install 的源出现了问题，更改一下就好。

supervisor 是一个进程管理工具，并对进程进行监控，出现异常还能控制进程重新启动。在这个例子，我们借助 supervisor 确保 flask 后台能够一直平稳运行。

部署

使用一个最简单的 flask 例子

在 myapp 目录下创建 app.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
```

```
# app.py
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'hello python'

if __name__ == '__main__':
    app.run()
```

这就是一个基础的 flask 框架搭建的后台， 使用python app.py 就可以运行起来，并且根据 官网的述，在 app.run() 中指定参数 host 为 0.0.0.0 就可以实现公网访问。也就是你可以直接在浏览器输入服务器公网ip:端口 的方式来访问。

但是，flask 自带的这个 server 只是适用于开发，而不是用于生产环境，所以在部署的时候，我们使用 uWSGI （后文简写 uwsgi）。uwsgi是一个 web 服务器，他实现了各种协议，用于和 web 应用 或者其他服务器进行信息交换。

我们接下来的目标是使用 uwsgi 替代 flask 自带的 server，从而更好的应用到生产环境中。

首先，在 myapp 目录下 创建 config.ini 文件

```
[uwsgi]
socket = 127.0.0.1:8081
chdir = /root/document/python/myapp/
wsgi-file = app.py
;flask 需要增加 callable
callable = app
```

socket 指明了 通讯的地址，uwsgi 将会绑定本地 8081 端口。 chdir 这个是指出项目的路径，在 Django 中是必须的，这里不写也没事。 wsgi-file 是指 入口文件，callable 需要指定 WSGI 函数，总之们使用 flask 的时候，一切都已经解决，app = Flask(__name__) ，app 已经的到了这个函数，所以定app。

配置好以后我们先尝试使用 uwsgi 启动一下我们的 flask 应用（在 myapp 目录下）：

```
uwsgi --ini config.ini
```

之后我们可以通过浏览器直接访问 服务器公网ip:8081。

到了这里，就会发现一个问题，假如我退出了控制台，那么这个进程就会退出，那就不能继续访问了或者那一天服务器重启了，又或者遇到了一点故障，导致这个进程结束了，这个时候我们肯定希望能有一种方式，可以帮助我们维护这个进程，保证进程能够一直运行或者重启，从而保证用户可以无障的访问我们的网站。这个时候就需要借助 supervisor 进程管理工具了。

在之前安装好了 supervisor 之后，我们先生成一个配置文件：

```
echo_supervisord_conf > /root/supervisord.conf
```

该命令后面的路径可以自己指定。自动生成 supervisor 的默认配置文件

打开这个配置文件，直接在最下面添加：

```
[program:myapp]
```

```
command = /root/document/python/myapp/venv/bin/uwsgi --ini /root/document/python/myapp/config.ini
stopsignal = QUIT
autostart = true
autorestart = true
stdout_logfile = /var/log/uwsgi/supervisor_myapp.log
stderr_logfile = /var/log/uwsgi/supervisor_myapp_err.log
```

program 后面写明 项目名，就是我们的 文件夹名。 command 指的是 supervisor 会执行的命令，于我们使用 virtualenv，所以 uwsgi 命令需要指定路径。

注意最后两个是日志输出文件，自己指定一个目录和文件名即可，必须保证路径存在。

之后执行：

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

启动 supervisor。

查看 supervisor 和 uwsgi 是否在运行：

```
ps aux | grep supervisord
ps aux | grep uwsgi
```

其他一些 supervisor 的使用方式或者指令可以查看官方文档

现在所有的配置基本完成了，在这个时候已经可以顺利的访问后台了，并且相当稳定，不过由于我们务器上不仅仅只是跑 uwsgi 的服务，或许还有别的服务，所以使用 nginx 负责转发服务（抛开 nginx 的其他特性不说）。

nginx 配置文件：

```
upstream uwsgitest {
    server 127.0.0.1:8081;
}
server{
      listen          your port;
    server_name     your domain;

    location / {
        include            uwsgi_params;
        uwsgi_pass         uwsgitest;

        proxy_redirect      off;
        proxy_set_header    Host $host;
        proxy_set_header    X-Real-IP $remote_addr;
        proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header    X-Forwarded-Host $server_name;
    }
    error_page  404              /404.html;
    location = /404.html {
         root   /usr/share/nginx/html;
    }
}
```

明确目的：我们需要使用 nginx 转发服务， 和 uwsgi 进行交换。

因此，保证配置文件中 upstream 中的 server 内容和 uwsgi 中的一致。include 指明了 双方传递的数，uwsgi_pass 指明传递路径。

重启 nginx 服务后，就可以通过你自己指定的方式访问 flask 的后台服务了。