安装 virtualenv Python虚拟环境

pip3 install virtualenv

下面报错 就需要更新 Pip 更新命令pip install --upgrade pip

**pip version 8.1.1,however version 9.0.1 is available**

**还要安装：https://www.linuxidc.com/Linux/2017-02/140417.htm**

**外面的文件夹有安装文档**

<http://www.jb51.net/article/130990.htm>

# centos 7.0 使用Nginx部署flask应用教程

转载  2017-12-19   作者：七度上弦   [[http://www.jb51.net/skin/2018/images/text-message.png](http://www.jb51.net/article/130990.htm#comments) 我要评论](http://www.jb51.net/article/130990.htm#comments)

这篇文章主要介绍了centos 7.0 使用Nginx部署flask应用教程,需要的朋友可以参考下

参考文章：<http://www.jb51.net/article/130989.htm>

修改说明：修改nginx配置部分，更换配置文件内容及路径；从ip访问切换到域名访问

**前言**

最近博主在开发一个小项目，本机的开发环境是python3.6，因为需要部署到服务器（python2.7）的关系，所以研究了一下，网上大部分的教程是部署在Ubuntu上的，博主使用的是centos7.0，大部分的部署其实都是类似的，这篇博文对完完全全的新手可能会有所帮助，对遇到的一些困惑也可以进行描述。

**环境准备**

python包安装工具pip : sudo apt-get install pip   
virtualenv：为了避免系统环境与应用环境之间的冲突，使用虚拟环境来安装应用的依赖，virtualenv可以为每个python应用创建独立的开发环境，安装方式：sudo pip install virtualenv

**虚拟环境使用**

flask项目地址：/usr/local/flasky，只需要在项目地址下运行指令：

[?](http://www.jb51.net/article/130990.htm)

|  |  |
| --- | --- |
| 1 | virtualenv venv |

约定俗成一般使用“venv”作为虚拟环境的名称，这样我们就创建了一个虚拟环境的文件夹venv如下图红框所示：

[这里写图片描述](http://files.jb51.net/file_images/article/201712/201712192116351.png)

接下来激活虚拟环境，只用指令

[?](http://www.jb51.net/article/130990.htm)

|  |  |
| --- | --- |
| 1 | source venv/bin/activate |

这里写图片描述

激活虚拟环境后，命令行抬头会有(venv)标识，仅需虚拟环境后，一切安装python的操作都会讲包和引用装在虚拟环境中，不会影响全局的python环境

退出虚拟环境的指令，直接敲：

deactivate

即可退出虚拟环境

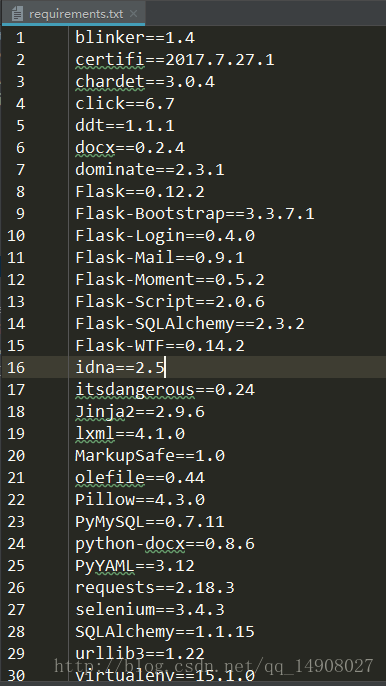
**配置flask应用环境**

flask应用开发的时候，会应用到非常多的模板，“狗书”中有介绍一种比较方便的方式，直接把开发环境的flask依赖等导出为一个txt文件，然后在centos的虚拟环境中使用相应指令根据这个txt文件安装会非常方便。

[?](http://www.jb51.net/article/130990.htm)

|  |  |
| --- | --- |
| 1 | pip freeze >requerements.txt |

 该指令导出一个文件名为requirements.txt的文本，里面的内容如下图所示



然后centos中使用以下指令

pip install -r requerments.txt

这样就会在虚拟环境中安装一个与开发环境一致的副本了。  
注意一定要在虚拟环境中使用批量安装指令，不然就安装到全局去了，后果还是比较蛋疼的。

**安装uWSGI**

flask的实际生产运行环境选择不多，比较成熟的是【Gunicorn】和【uWSGI】，这里推荐一本书董伟明的《Python Web开发实战》里面有讲这两种部署。

以下内容来自百度百科：

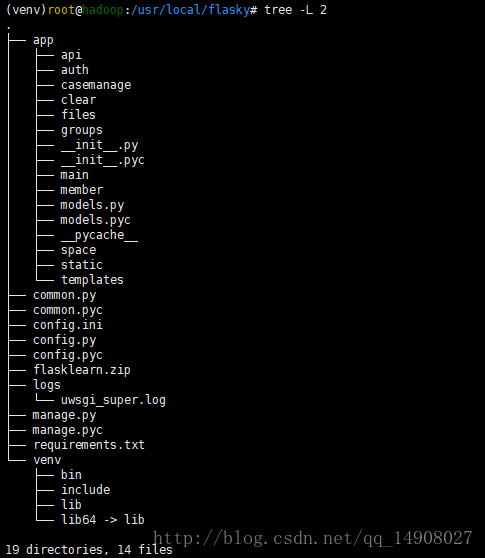
uWSGI是一个Web服务器，它实现了WSGI协议、uwsgi、http等协议。Nginx中HttpUwsgiModule的作用是与uWSGI服务器进行交换。WSGI是一种Web服务器网关接口。它是一个Web服务器（如nginx，uWSGI等服务器）与web应用（如用Flask框架写的程序）通信的一种规范。  
安装指令如下，注意一定要保证已经进入了虚拟环境并激活：

pip install uwsgi

不需要使用sudo ，因为virtualvenv没有权限要求。

**上传项目文件**

博主使用的linux连接工具是xshell，在之前的一篇文章<http://www.jb51.net/article/130991.htm>中有讲如何上传文件到linux主机，这里不再详细说明，展示一下大概的项目结构以及启动文件manage.py



tree命令可以以树形图的形式展示文件结构，可以赋予参数设置树的深度，这里博主展示了2层   
安装命令：

sudo apt-get install tree

manage.py代码如下

[?](http://www.jb51.net/article/130990.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #coding=utf-8  #!/usr/local/flasky/venv python  '''  Created on 2017/11/9  @author: Savitar丶  Project:  '''  import os  from app import create\_app  from flask\_script import Manager, Shell  app = create\_app(os.getenv('FLASK\_CONFIG') or 'default')  manager = Manager(app)    def make\_shell\_context():   return dict(app=app)  manager.add\_command("shell", Shell(make\_context=make\_shell\_context))  @manager.command  def deploy():   """Run deployment tasks."""   pass  if \_\_name\_\_ == '\_\_main\_\_':   manager.run()   #app.run() |

**命令行运行flask项目**

python manage.py runserver

以上命令一般都是在开发环境进行调试的时候使用，或者注释掉“manager.run()”，使用“app.run()”，后者可以直接在pycharm中执行项目，这里不再详细说明

**配置uWSGI**

我们直接在flask的根目录下面新建一个文件“config.ini”,使用的方式是配置启动。文件内容如下

[?](http://www.jb51.net/article/130990.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | [uwsgi]    # uwsgi 启动时所使用的地址与端口    socket = 127.0.0.1:5000    #虚拟环境目录  home = /usr/local/flasky/venv    #指向网站根目录  chdir = /usr/local/flasky    #python启动程序文件  wsgi-file = manage.py    #python程序内用于启动的application变量名  callable = app    #处理器数  processes = 4    #线程数  threads = 2    #状态监测地址  stats = 127.0.0.1:9191    #设置uwsgi包解析的内部缓存区大小。默认4k    buffer-size = 32768 |

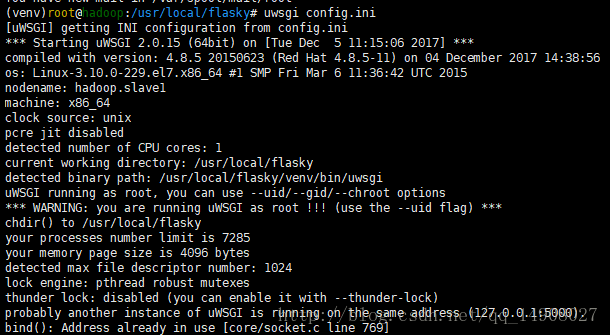
配置文件的执行方式，命令行输入指令：

uwsgi config.ini

或者不写配置文件（不建议这样做），直接命令行输入

uwsgi –socket 127.0.0.1:5000 –wsgi-file manage.py –callable app –process 4 –threads 2

这里的callable=app 这个app是manage.py程序文件内的一个变量，这个变量的类型是flask的application类



这里是因为博主已经配置好了项目自启动，已经有一个配置文件在运行了，大家按照正常流程配置就好。ctrl+c关闭程序，在实际项目中我们的服务器上可能会有多个项目在运行，我们需要应用随同服务器启动并作为后台服务运行才是实际项目需要，所以我们需要安装另一个工具来引导执行uwsgi

**安装supervisor**

supervisor可以同时启动多个应用，最重要的是当某个应用down掉的时候，他可以自动重启该应用，保证可用性。

sudo apt-get install supervisor

supervisor的全局配置文件在

[这里写图片描述](http://files.jb51.net/file_images/article/201712/201712192116366.png)

打开该默认配置文件，最下面一行我们看到，该默认配置文件会从 /etc/supervisord/目录下面加载所有的配置文件

这里写图片描述

我们不需要修改默认的配置文件，只需要在/etc/supervisord/目录下新建一个配置文件（命名为flask\_supervisor.conf）   
这里写图片描述

该文件内容如下：

[?](http://www.jb51.net/article/130990.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | [program:flasky]  # 启动命令入口  command=/usr/local/flasky/venv/bin/uwsgi /usr/local/flasky/config.ini    # 命令程序所在目录  directory=/usr/local/flasky  #运行命令的用户名  user=root    autostart=true  autorestart=true  #日志地址  stdout\_logfile=/usr/local/flasky/logs/uwsgi\_super.log |

这里command这一行代码看起来很长，其实就是之前我们的 “uwsgi config.ini”指令，这里使用的是绝对路径，保证命令和文件的准确性，大家也可以copy这一行代码出去执行，结果是ok的。autostart和autorestart参数保证了我们的应用可以一直保持启动的状态，即使是down掉了也能重启服务。

启动服务

sudo service supervisor start

终止服务

sudo service supervisor stop

**安装Nginx**

Nginx是轻量级，性能强，占用资源少，能很好的处理高并发的反向代理软件。

**正向代理和反向代理**

正向代理，作为一个媒介将互联网上获取的资源返回给相关联的客户端。代理和客户端在一个局域网，对于服务是透明的。反向代理，根据客户端的请求，从后端的服务器上获取资源，然后再讲这些资源返回给客户端。代理和服务器再一个局域网，对客户端是透明的。Nginx是反向代理的最佳选择。

**反向代理的作用**

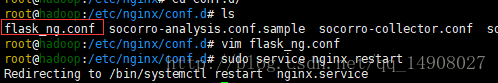
提高动态语言的IO处理能力   
加密和SSL加速   
安全。   
负载均衡   
缓存静态内容   
支持压缩。

**Nginx安装指令：**

sudo apt-get install nginx

**配置Nginx**

我们找到nginx的配置文件，不要修改默认的nginx.conf（路径 /etc/nginx/nginx.conf）文件，只需要在同样的文件夹下面新建一个文件夹(conf.d)然后在conf.d下面新建配置文件（flask\_ng.conf）即可，如下图



flask\_ng.conf文件内容如下

[?](http://www.jb51.net/article/130990.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | server {    listen 80;    server\_name www.cloud-test.com; #公网地址      location / {    include  uwsgi\_params;    uwsgi\_pass 127.0.0.1:5000; # 指向uwsgi 所应用的内部地址,所有请求将转发给uwsgi 处理    uwsgi\_param UWSGI\_PYHOME /usr/local/flasky/venv; # 指向虚拟环境目录    uwsgi\_param UWSGI\_CHDIR /usr/local/flasky; # 指向网站根目录    uwsgi\_param UWSGI\_SCRIPT manage:app; # 指定启动程序    }   } |

**启动Nginx**

sudo service nginx restart

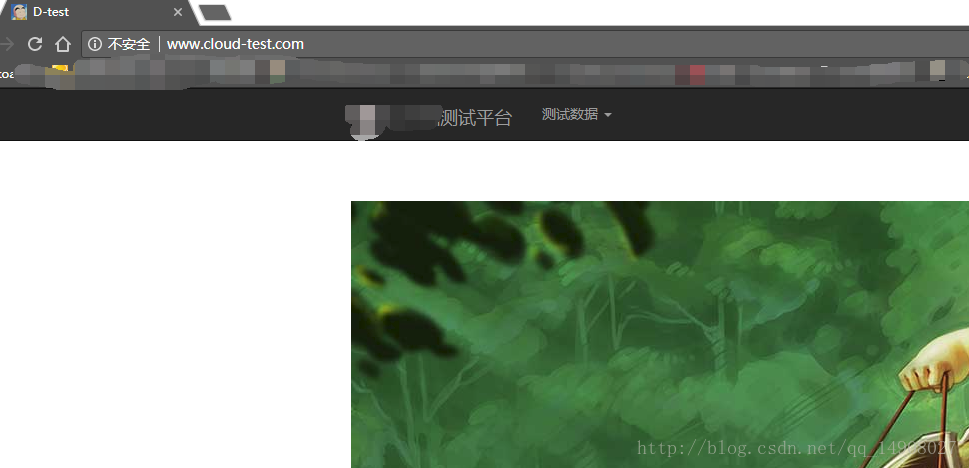
然后浏览器直接访问服务器http://www.cloud-test.com地址，结果如下



小case，解决办法，直接修改本机hosts，绑定就好了，如下图所示



然后再去访问[http://www.cloud-test.com](http://www.cloud-test.com/)

[](http://files.jb51.net/file_images/article/201712/2017121921163712.png)

然后也可以找自己的同事一起访问，看看并发请求，网站的运行情况，想来能解决在开发环境使用了host 0.0.0.0 也只能处理一个用户的请求的问题

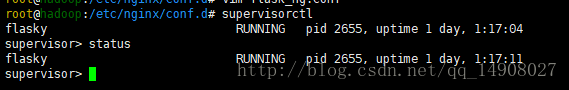
查看应用的运行情况

supervisorctl 是 supervisord 的命令行客户端工具，使用的配置和 supervisord 一样，这里就不再说了。下面，主要介绍 supervisorctl 操作的常用命令：   
输入命令 supervisorctl 进入 supervisorctl 的 shell 交互界面（还是纯命令行），可以看到应用的运行情况   
就可以在下面输入命令了。：

help # 查看帮助   
status # 查看程序状态   
stop program\_name # 关闭 指定的程序   
start program\_name # 启动 指定的程序   
restart program\_name # 重启 指定的程序   
tail -f program\_name # 查看 该程序的日志   
update # 重启配置文件修改过的程序（修改了配置，通过这个命令加载新的配置)

也可以直接通过 shell 命令操作：

supervisorctl status   
supervisorctl update



s