目录

[安装nginx 2](#_Toc413871016)

[启动nginx 2](#_Toc413871017)

[重新启动nginx: 3](#_Toc413871018)

[安装nginx +uwsgi+ Django 3](#_Toc413871019)

[五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(上) 4](#_Toc413871020)

[环境介绍 5](#_Toc413871021)

[安装uwsgi 5](#_Toc413871022)

[1.安装pip 5](#_Toc413871023)

[2.安装uwsgi 5](#_Toc413871024)

[测试uwsgi 5](#_Toc413871025)

[配置django 5](#_Toc413871026)

[连接django和uwsgi，实现简单的WEB服务器。 6](#_Toc413871027)

[五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(下) 6](#_Toc413871028)

[**环境介绍** 7](#_Toc413871029)

[配置uWSGI 7](#_Toc413871030)

[配置Nginx 8](#_Toc413871031)

[Nginx+uWSGI+Django的实现方式 10](#_Toc413871032)

[一些建议 10](#_Toc413871033)

[最后 11](#_Toc413871034)

# 安装nginx

**sudo apt-get install nginx**

Ubuntu安装之后的文件结构大致为：

* 所有的配置文件都在/etc/nginx下，并且每个虚拟主机已经安排在了**/etc/nginx/sites-available**下
* 程序文件在/usr/sbin/nginx
* 日志放在了/var/log/nginx中
* 并已经在/etc/init.d/下创建了启动脚本nginx
* 默认的虚拟主机的目录设置在了/var/www/nginx-default (有的版本 默认的虚拟主机的目录设置在了/var/www, 请参考/etc/nginx/sites-available里的配置)

# 启动nginx

**sudo /etc/init.d/nginx start**

然后就可以访问了，<http://localhost/> ， 一切正常！如果不能访问，先不要继续，看看是什么原因，解决之后再继续。 启动时候若显示端口80被占用： Starting nginx: [emerg]: bind() to 0.0.0.0:80 failed (98: Address already in use) 修改文件：/etc/nginx/sites-available/default,去掉 listen 前面的 # 号 , # 号在该文件里是注释的意思 , 并且把 listen 后面的 80 端口号改为自己的端口，访问是需要添加端口号。

（安装完后如出现403错误，那可能是nginx配置文件里的网站路径不正确）

|  |
| --- |
| [http://wiki.ubuntu.org.cn/images/thumb/f/f0/Nginxdefault.png/500px-Nginxdefault.png](http://wiki.ubuntu.org.cn/File:Nginxdefault.png)  [http://wiki.ubuntu.org.cn/skins/common/images/magnify-clip.png](http://wiki.ubuntu.org.cn/File:Nginxdefault.png)  nginx默认页面 重新启动nginx: **/etc/init.d/nginx stop**  **/etc/init.d/nginx start**  **/etc/init.d/nginx force-reload** 安装nginx +uwsgi+ Django 这次采用 uwsgi 1.2+django1.4+nginx1.1.19+ubuntu12.04组合进行安装  首先注意 要安装比较新的软件 如果你是ubuntu11.04和11.10 因为内核不同 所以安装软件版本不同  1.首先 安装nginx： **sudo apt-get install nginx**  2.然后： 进行uwsgi的安装**之前** 要安装其他几个：  **sudo apt-get install mysql-server libxml2 libxml2-dev python-dev libpcre3 libpcre3-dev python-MySQLdb**  3.进行uwsgi的安装： 可能由于软件版本不够新吧 ppa里头安装后出现502错误 所以 用软件包安装：   1. **wget** [**http://projects.unbit.it/downloads/uwsgi-1.2.tar.gz**](http://projects.unbit.it/downloads/uwsgi-1.2.tar.gz) 2. **tar zxvf uwsgi-1.2.tar.gz** 3. **cd uwsgi-1.2** 4. **make** 5. **cp uwsgi /usr/bin**   注：如果在MAKE 过程中出现如：（Python.h：没有那个文件或目录）  解决方法：**sudo apt-get install python-dev**  4.安装django 之前我用ubuntu源安装.由于admin管理的静态问题.所以我还是改用下载安装比较好. google下django 然后进入下载 <https://www.djangoproject.com/download/> 用chrome下载默认/home/helios/下载 然后cd 然后cd 下载 接着   1. **tar** xzvf Django-1.4.tar.gz 2. cd Django-1.4 3. sudo python setup.py install   这是最新版本.copy官网的安装方法. 接着  sudo cp django/bin/django-admin.py /usr/bin/  接下来  sudo mv /usr/bin/django-admin.py /usr/bin/django-admin  这是为了和源安装在操作上保持一致 但是不知道有没有问题.暂时没有发现问题.有问题大家可以上来补教程. 可以了. 安装好了. 五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(上) By [Django中国社区](http://django-china.cn/user/1/) at 2013-05-12 15:05  Django的部署可以有很多方式，采用nginx+uwsgi的方式是其中比较常见的一种方式。在这种方式中，我们的通常做法是，将nginx作为服务器最前端，它将接收WEB的所有请求，统一管理请求。nginx把所有静态请求自己来处理 （这是NGINX的强项）。然后，NGINX将所有非静态请求通过uwsgi传递给Django，由Django来进行处理，从而完成一次WEB请求。  可见，uwsgi的作用就类似一个桥接器。起到桥梁的作用。  NOTE：不使用nginx，只使用uwsgi+django也是可以实现WEB服务的。uwsgi也可以直接处理WEB请求。  为了完成上述的方式部署，我将分成两篇文章来分别进行阐述。   * 第一步先解决uwsgi与django的桥接。解决在没有nginx的情况下，如何使用uwsgi+DJANGO来实现一个简单的WEB服务器。 * 第二步解决uwsgi与Nginx的桥接。通过nginx与uwsgi的桥接，打通nginx与django的连通，从而比较完美的实现django的部署。   本文将分成五步来详细阐述uwsgi+django的部署方式。nginx+uwsgi+django的部署将在下一篇 文章中阐述。  本文大纲：   * 环境介绍 * 安装uwsgi * 测试uwsgi * 配置django * 连接django和uwsgi，实现简单的WEB服务器。  环境介绍  1. Ubuntu 12.04.1 LTS 2. django 1.5.12  安装uwsgi1.安装pip 可以参考这篇文章：<http://www.jsxubar.info/install-pip.html> 2.安装uwsgi $ export LDFLAGS="-Xlinker --no-as-needed"  $ pip install uwsgi 测试uwsgi 在你的机器上写一个test.py  # test.py  def application(env, start\_response):  start\_response('200 OK', [('Content-Type','text/html')])  return "Hello World"  然后执行shell命令：  **uwsgi --http :8001 --wsgi-file test.py**  访问网页：  <http://127.0.0.1:8001/>  看在网页上是否有Hello World 配置django NOTE：  请保证你的django项目是正常使用的。可以使用  **python manage.py runserver 0.0.0.0:8002**  来测试一下你的django项目是否能正常跑起来。  请保证你的django程序已经关闭。  **编写django\_wsgi.py文件**，将其放在**与文件manage.py同一个目录下**。  注意： 编写文件时需要注意语句os.environ.setdefault。比如，如果你的项目为mysite，则你的语句应该是 os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "mysite.settings")  #!/usr/bin/env python  # coding: utf-8  import os  import sys  # 将系统的编码设置为UTF8  reload(sys)  sys.setdefaultencoding('utf8')  os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "mysite.settings")  from django.core.handlers.wsgi import WSGIHandler  application = WSGIHandler() 连接django和uwsgi，实现简单的WEB服务器。 我们假设你的Django项目的地址是/home/work/src/sites/testdjango1/testdjango/mysite，  然后，就可以执行以下命令：  uwsgi --http :8000 --chdir /home/work/src/sites/testdjango1/testdjango/mysite --module django\_wsgi  这样，你就可以在浏览器中访问你的Django程序了。所有的请求都是经过uwsgi传递给Django程序的。 五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(下) By [Django中国社区](http://django-china.cn/user/1/) at 2013-05-18 04:38  在上一篇文章[《五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(上)》](http://django-china.cn/topic/101/)中，阐述了如何只使用uWSGI来部署Django程序。  当然，单单只有uWSGI是不够的，在实际的部署环境中，Nginx是必不可少的工具。  在本篇文章中，我将一直延用“N步法”的风格来阐述如何将uWSGI与Nginx做连接来部署Django程序。并在最后，会较为完整的阐述本社区的部署方法。  **本文大纲：**   * 环境介绍 * 配置uWSGI * 配置Nginx * Nginx+uWSGI+Django的实现方式 * 一些建议   **环境介绍**   1. Ubuntu 12.04.1 LTS 2. django 1.4.2 3. nginx/1.2.6 4. uWSGI 1.4.4   关于uWSGI的安装可参见上一篇文章 上一篇文章[《五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(上)》](http://django-china.cn/topic/101/)  我们假定你已经安装好Nginx了。 配置uWSGI 在上一篇文章[《五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(上)》](http://django-china.cn/topic/101/)中,我们是直接使用命令行来启动uWSGI，在实际部署环境中，我们常用的是配置文件的方式，而非命令行的方式。我的一般做法是用命令行来测试是否uWSGI安装成功，然后用配置文件来真正部署。  另外，为了实现Nginx与uWSGI的连接，两者之间将采用soket来通讯方式。  在本节中，我们将使用uWSGI配置文件的方式来改进uWSGI的启动方式。  假定你的程序目录是 /home/work/src/sites/testdjango1/testdjango/mysite  我们将要让Nginx采用8077端口与uWSGI通讯，请确保此端口没有被其它程序采用。  注意，请确定你在上一节[《五步教你实现使用Nginx+uWSGI+Django方法部署Django程序(上)》](http://django-china.cn/topic/101/)中的django\_wsgi.py文件已经存在了。  新建一个XML文件：  **djangochina\_socket.xml**，  将它放在 /home/work/src/sites/testdjango1/testdjango/mysite 目录下：(**与manage.py同一个目录**)  <uwsgi>  <socket>:8077</socket>  <chdir>/home/work/src/sites/testdjango1/testdjango/mysite</chdir>  <module>django\_wsgi</module>  <processes>4</processes> <!--进程数-->配置时要删除，否则编码出错  <daemonize>uwsgi.log</daemonize>  </uwsgi>  在上面的配置中，我们使用 uwsgi.log 来记录日志，开启4个进程来处理请求。  这样，我们就配置好uWSGI了。 配置Nginx 我们假设你将会把Nginx程序日志放到你的目录  /home/work/var/test/logs/下，**请确保该目录存在**。  我们假设你的Django的  **static目录是**：  /home/work/src/sites/testdjango1/testdjango /collectedstatic/  **media目录是：**  /home/work/src/sites/testdjango1/testdjango/public/media/  请确保这些目 录存在。  我们假设你的域名是 [www.you.com](http://www.you.com) （在**调试时**你可以设置成你的机器IP）  我们假设你的域名端口是 80（在调试时你可以设置一些特殊端口如 8070）  基于上面的假设，我们为conf/nginx.conf（高级版本中已无该文件夹了，修改见后面）添加以下配置  server {  listen 80;  server\_name www.you.com;  access\_log /home/work/var/test/logs/access.log;  error\_log /home/work/var/test/logs/error.log;  #charset koi8-r;  #access\_log logs/host.access.log main;  location / {  include uwsgi\_params;  uwsgi\_pass 127.0.0.1:8077;  }  #error\_page 404 /404.html;  # redirect server error pages to the static page /50x.html  #  error\_page 500 502 503 504 /50x.html;  location = /50x.html {  root html;  }  location /static/ {  alias /home/work/src/sites/testdjango1/testdjango/collectedstatic/;  index index.html index.htm;  }  location /media/ {  alias /home/work/src/sites/testdjango1/testdjango/public/media/;  }  }  在上面的设置后，可以让Nginx来处理静态文件(/static/ 和 /media/ ）。非静态文件请求Nginx会发给 socket 8077，然后让uWSGI来进行处理。  **在高级版本中，conf文件夹已经被更改为conf.d空文件夹，**安装后在/etc/nginx 下有配置文件nginx.conf，但我尝试将楼主的代码直接写入发现会出现错误，后来查看了一些资料，**在/etc/nginx/conf.d 下建立任意名称.conf文件，将上述server代码全部写入并根据自己的情况修改，我只修改了服务器地址和静态文件地址，保存后重启Nginx生效。** Nginx+uWSGI+Django的实现方式 在完成上面配置后，需要按以下步骤来做：   1. 重启Nginx服务器，以使Nginx的配置生效。 2. **nginx -s reload**   重启后检查Nginx日志是否有异常。   1. 启动uWSGI服务器 2. cd /home/work/src/sites/testdjango1/testdjango/mysite 3. **uwsgi -x djangochina\_socket.xml**   检查日志 uwsgi.log 是否有异常发现。   1. 访问服务   基于上面的假设你的域名是[www.you.com](http://www.you.com) 因此，我们访问 [www.you.com](http://www.you.com)，如果发现程序与 单独使用Django启动的程序一模一样时，就说明成功啦！   1. 关闭服务的方法   将uWSGi进程杀死即可。 一些建议  1. uWSG配置文件的进程数，可以根据实际情况分配。不要开得太大，否则机器可能会内存耗用太高。一般来说，对于一个小社区来说，4个进程已经足够了。 2. 一般情况下，可以编写一下 stop.sh 脚本 来关闭uWSGI。  最后 最后面，请大家要支持[Django中国社区](http://django-china.cn/)哦，单靠一两个人是不行的，一起推广一下，让Django社区更有力量哈！更有人气哈！  推广链接： <http://django-china.cn/> |
|  |