1. Django生产环境部署主要指将Django部署到实际的云平台或者 Web服务器中，而不是Django自带的开发服务器。

2. WSGI (Web Server Gateway Interface), 他描述了Web服务器如何和Web应用（Web application）交流来处理一个HTTP请求的规范。Web服务器指的是Apache，而Web application指的是Django, PHP, JSP等网页开发框架所开发的网站，应用。

3.mod\_wsgi模块就是Python的Apache模块，用于将所有Python 的WSGI应用，例如Django, Flask部署到Apache服务器上。

4. 查看linux系统中所安装的apache版本，apachectl -v

5. WSGIScriptAlias / /path/to/mysite.com/mysite/wsgi.py

这句话表明WSGI将处理/开始的路径，并且将由/path/to/mysite.com/mysite/wsgi.py文件负责。

在wsgi.py文件中，wsgi.py的作用是创建一个WSGI应用对象。从Django1.4版本开始，一旦使用startproject，Django将自动创建一个wsgi.py文件。

6. 如果我们在settings.py中的installed\_apps中加入[django.contrib.staticfiles](https://docs.djangoproject.com/en/1.9/ref/contrib/staticfiles/" \l "module-django.contrib.staticfiles" \o "django.contrib.staticfiles: An app for handling static files.)，那么表明这台服务器不仅将host apps也将host apps中的所有静态文件。因为除了这种方式以外，我们还可以设置成apps由一台服务器host,而其中的静态文件由另一台服务器host.

7. 将Django部署到Apache服务器时，我们需要Apache服务器装载mod\_wsgi.so模块。整体的过程为，我们先从网上下载mod\_wsgi.so模块的源码，然后进行配置（配置是指针对什么python版本等），然后编译，然后安装。

Step1：mod\_wsgi.so模块的源码我们可以从<https://github.com/GrahamDumpleton/mod_wsgi/releases>获取。然后将其解压。

Step2：第二部进行编译前的配置。在解压好的文件夹中执行. /configure, 如果我们不加任何参数，那么configure命令将默认寻找系统中默认使用的python，另外我们也可以手动指定（这多用于系统中含有多个Python版本的情况）。

Step3：编译。Make, make的时候，系统根据我们在第二步中的配置，编译生成mod\_wsgi.so.

Step4：安装。安装有两种方式，第一种是自动安装执行make install. 自动安装其实就是将第三步中生成的so文件通过系统自动检测，放置到检测到的apache的模块目录下。第二种是手动安装，第二步生成的.so文件将会放置在解压文件夹的lib目录下，我们手动的拷贝到apache的模块目录下（一般为/etc/httpd/modules/中）

Step5: 修改apache的配置文件。让apache在运行的时候去加载这个模块。添加配置LoadModule wsgi\_module modules/mod\_wsgi.so. 然后重启apache.

Step6: make clean, make distclean进行编译后的清理工作。

### 8.Django在部署到Apache的时候，一个重要的点是需要提供一个WSGI APPLICATION CALLEABLED OBJECT. 这个object可以在settings.py中以WSGI\_APPLICATION指定，如果没有执行，默认将会是django.core.wsgi.get\_wsgi\_application()的返回值。所指定的文件（一般是wsgi.py）中，含有一个WSGI APPLICATION CALLEABLED OBJECT

### 9. Python-Style指定路径的方式是dotted path 例如mysite.settings也就是mysite/settings.py

### 10.Django在创建项目的时候会自动生成一个wsgi.py文件，里面便提供一个最简单的WSGI APPLICATION CALLEABLED OBJECT.本质是提供一个这样的object, 所以我们还可以对这个object进行改造，形成中间件的效果，还可以继承这个类，写出满足自己要求的WSGI APPLICATION类来扩展。

### 11.Apache的主文件部署是httpd.conf, 在其中如果我们开启了一些额外的配置，那么这些配置都将引入httpd.conf中作为配置。例如在 httpd.conf中

### #

### # Load config files from the config directory "/etc/httpd/conf.d".

### #

### Include conf.d/\*.conf

那么我们在/etc/httpd/conf.d/下面新建配置文件的时候，这些新建的配置文件都将被生效。

12.Apache的虚拟主机 (virtual host) 技术。Apache的虚拟主机技术用于，在一台物理机器上面同时host多次网站。我们可以有两种方式做到，第一种是在物理机器上面有多张网卡，我们指定每张网卡负责一个IP的网站，这叫Ip-Based virtual host. 第二种更常用的是hostname-based virtual host，就是基于手动指定的域名来定义。

每一段virtual host的基本配置有：

<VirtualHost \*:80> 🡪表示监听什么ip地址上的什么端口

WSGIScriptAlias / /srv/www/megatron/public\_html/sam/sam/wsgi.py 🡪如果部署为Django的时候，指明wsgi object在哪

ServerAdmin [hui\_h.jin@nokia.com](mailto:hui_h.jin@nokia.com) 🡪系统管理员的邮箱

ServerName 135.121.4.88 🡪域名

Alias /static /srv/www/megatron/public\_html/sam/regression\_f/static ->如果是访问/static的时候，将访问本机上的/srv/www/megatron/public\_html/sam/regression\_f/static目录

ServerAlias megatron

DocumentRoot /srv/www/megatron/public\_html/sam/sam

<Directory /srv/www/megatron/public\_html/sam/sam>

<Files wsgi.py>

Options FollowSymLinks

Order deny,allow

Allow from all

</Files>

</Directory>

ErrorLog /srv/www/megatron/logs/error.log ->错误日志所在地

CustomLog /srv/www/megatron/logs/access.log combined ->自定义日志所在地

</VirtualHost>

12.配置完apache以后可以使用/usr/sbin/httpd -S来查看配置形况, apache会检查配置中是否存在语法错误

13.