开始

所谓运维，就是将系统部署到服务器上，并且连接公网，在这个过程种要求系统能够有效的接收外界请求并能够正常工作。这个过程通常没有唯一的方案与工具，每个技术栈都八仙过海、各显神通。在网上大家能查到的资料也是五花八门，运维这个分支也是涉及工具链和知识面最广的，我费了好大力气将其搜集并整理出一套最适合目前django生产部署的一个方案。

这个方案兼顾易用性、可移植性以及高性能。

Django运维部署框架

整体部署架构

操作系统: Linux

优势：生态系统丰富，程序支持度高，如docker在linux的性能就比在windows好

数据库: Mysql

优势：开源，性能强劲的关系型数据库

前端web服务器: Nginx

后端app服务器: uWSGI

概念扫盲

web服务器与web应用服务器的区别

The difference between web server and web application server

很多教程在涉及到运维这个部分的时候，上来就web服务器，应用服务器的开始部署，对于这块知识比较薄弱的同学就会感觉到懵逼，因为根本不懂应用服务器和web服务器的区别。就像我们看双胞胎，这两个名字经常一起出现，而且带着相同的名称，总是将我们混淆

Web服务器的基本功能就是提供Web信息浏览服务。它只需支持HTTP协议、HTML文档格式及URL。

web服务器专注http请求的处理与返回，通用性，如nginx和apache可以处理所有http请求然后返回静态文件内容

应用服务器是通过很多协议来为应用程序提供(serves)商业逻辑,具备解释服务端代码能力

应用服务器专注程序框架和动态内容的处理，专用性，如tomcat只能处理java程序而不能处理python程序

现在大部分应用服务器具备了

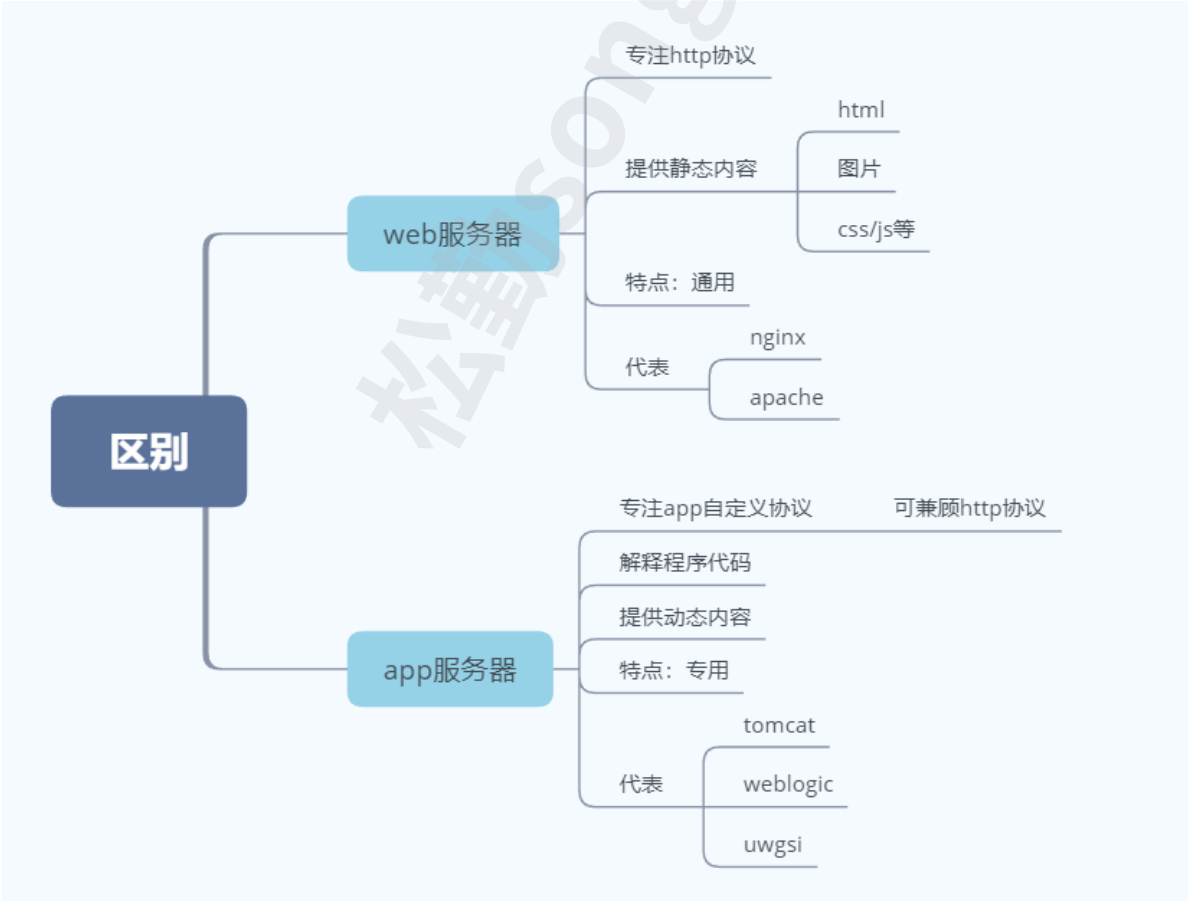
简单归纳一下两者的区别：

web服务器专注提供静态文件内容

app服务器专注提供动态内容

按照这个规则我们可以区分哪些是web服务器，哪些是app服务器。

软件名词	是否Web服务器	是否app服务器
IIS	是	是
Nginx	是	
Apache	是	
Tomcat	是	是
Jetty	是	是
WebSphere	是	是
WebLogic	是	是
uWSGI		是



##

先从宏观上了解下整个项目架构

Django配置项

settings.py

```
ALLOWED_HOSTS = ['*']  #host白名单，不填默认只能本机通过127.0.0.1访问服务器
DEBUG = False          #生产环境设置False
```

部署环境准备

安装python3

linux环境一键安装python

```
wget https://gitee.com/shonekey666/LinuxEnv/raw/master/CentOS/CentOS7-installpython3.sh && sh CentOS7-installpython3.sh
```

如果提示没有wget命令，可以尝试下面的

```
curl -O  https://gitee.com/shonekey666/LinuxEnv/raw/master/CentOS/CentOS7-installpython3-4.sh && sh CentOS7-installpython3-4.sh
```

创建虚拟环境

用于创建和管理虚拟环境的模块称为 `venv`。 `venv` 通常会安装你可用的最新版本的 Python。如果您的系统上有多个版本的 Python，您可以通过运行 `python3` 或您想要的任何版本来选择特定的Python版本。

要创建虚拟环境，请确定要放置它的目录，并将 `venv` 模块作为脚本运行目录路径:

```
python3 -m venv tutorial-env
```

激活虚拟环境

创建虚拟环境后，您可以激活它。

在Windows上，运行:

```
tutorial-env\Scripts\activate.bat
```

在Unix或MacOS上，运行:

```
source tutorial-env/bin/activate
```

退出虚拟环境

创建虚拟环境后，您可以激活它。

在Windows上，运行:

```
tutorial-env\Scripts\deactivate.bat
```

在Unix或MacOS上，运行:

```
deactivate
```

Mysql安装

CentOs7 安装 Mysql5.7

1、下载mysql源安装包

```
wget http://dev.mysql.com/get/mysql57-community-release-el7-8.noarch.rpm
```

2、安装mysql源

```
yum localinstall mysql57-community-release-el7-8.noarch.rpm
```

3、检查mysql源是否安装成功

```
yum repolist enabled | grep "mysql.*-community.*"
```

返回

```
[root@VM_18_105_centos ~]# yum repolist enabled | grep "mysql.*-community.*"
mysql-connectors-community/x86_64    MySQL Connectors Community               63
mysql-tools-community/x86_64         MySQL Tools Community                    69
mysql57-community/x86_64             MySQL 5.7 Community Server              287
```

看到上图所示表示安装成功。

也可以修改 `vim /etc/yum.repos.d/mysql-community.repo` 源，改变默认安装的mysql版本。比如要安装5.6版本，将5.7源的enabled=1改成enabled=0。然后再将5.6源的enabled=0改成enabled=1即可。改完之后的效果如下所示:

```
..........
# Enable to use MySQL 5.5
[mysql55-community]
name=MySQL 5.5 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.5-community/el/7/$basearch/
enabled=0 # 这里 0表示不选
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

# Enable to use MySQL 5.6
[mysql56-community]
```

```
name=MySQL 5.6 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.6-community/el/7/$basearch/
enabled=0 # 这里 0表示不选
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

[mysql57-community]
name=MySQL 5.7 Community Server
baseurl=http://repo.mysql.com/yum/mysql-5.7-community/el/7/$basearch/
enabled=1 # 这里 1 表示 选中
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql
.........
```

4、安装MySQL

```
yum install mysql-community-server
```

5、启动MySQL服务

```
systemctl start mysqld
```

查看MySQL的启动状态

```
[root@VM_18_105_centos ~]# systemctl status mysqld
● mysqld.service - MySQL Server
   Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; vendor
preset: disabled)
   Active: active (running) since 四 2018-08-23 15:27:28 CST; 1h 26min ago
     Docs: man:mysqld(8)
           http://dev.mysql.com/doc/refman/en/using-systemd.html
  Process: 21453 ExecStart=/usr/sbin/mysqld --daemonize --pid-
file=/var/run/mysqld/mysqld.pid $MYSQLD_OPTS (code=exited, status=0/SUCCESS)
  Process: 21432 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited,
status=0/SUCCESS)
 Main PID: 21457 (mysqld)
   Memory: 202.1M
   CGroup: /system.slice/mysqld.service
           └─21457 /usr/sbin/mysqld --daemonize --pid-
file=/var/run/mysqld/mysqld.pid
```

6、设置开机启动

```
 systemctl enable mysqld
 systemctl daemon-reload
```

7、获取root登陆密码

mysql安装完成之后，在/var/log/mysqld.log文件中给root生成了一个默认密码。通过下面的方式找到root默认密码，然后登录mysql进行修改：

```
[root@VM_18_105_centos ~]#  grep 'temporary password' /var/log/mysqld.log
2018-08-23T06:10:44.014590Z 1 [Note] A temporary password is generated for
root@localhost: thI/5wEl_chk
```

ps:如果没有返回，找不到root密码，解决方案：

```
# 1删除原来安装过的mysql残留的数据（这一步非常重要，问题就出在这）
rm -rf /var/lib/mysql
# 2重启mysqld服务
systemctl restart mysqld
# 3再去找临时密码
grep 'temporary password' /var/log/mysqld.log
```

原因有可能是之前安装过一次，没有安装好。

8、登陆

```
[root@VM_18_105_centos ~]# mysql -uroot -p
---- 输入密码：thI/5wEl_chk
# 修改密码
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY '123456Aa!';
```

MySql 默认密码级别一定要有大小写字母和特殊符号，所以比较麻烦。

9、修改密码策略

在/etc/my.cnf文件添加validate_password_policy配置，指定密码策略

```
# 0（LOW）：验证 Length
# 1（MEDIUM）：验证 Length; numeric, lowercase/uppercase, and special characters
# 2（STRONG）：验证 Length; numeric, lowercase/uppercase, and special characters;
dictionary file
validate_password_policy=0
```

当然如果不需要密码策略，可以禁用：
在/etc/my.cnf文件添加

```
validate_password = off
```

重启生效：

```
systemctl restart mysqld
```

Mysql的root用户，只能本地访问，这里在创建一个远程可以访问的 用户。

```
 GRANT ALL PRIVILEGES ON *.* TO 'its'@'%' IDENTIFIED BY '123456' WITH GRANT
OPTION;
```

10、忽略大小写

登陆mysql查看

```
mysql> show variables like "%case%";
+------------------------+-------+
| Variable_name          | Value |
+------------------------+-------+
| lower_case_file_system | OFF   |
| lower_case_table_names | 0     |  ##0区分 1 不区分
+------------------------+-------+
2 rows in set (0.00 sec)
```

修改配置文件 /etc/my.cnf 添加:

```
# 0：区分大小写，1：不区分大小写
lower_case_table_names =1
```

重启后生效:

```
systemctl restart mysqld
```

11、用户权限

把数据库迁移到新的服务器上,执行存储过程时出现了如下问题:

```
execute command denied to user ‘用户名’@’%’ for routine ‘函数名称’
```

后来一查原来是权限问题,只要用下面的语句改一下相应用户的权限就可以了:

```
GRANT ALL PRIVILEGES ON *.* TO ‘用户名’@’%’ ;
FLUSH PRIVILEGES;
```

相应的撤消权限命令:

```
REVOKE ALL PRIVILEGES ON *.* FROM  ‘用户名’@’%’ ;
FLUSH PRIVILEGES;
```

12、跑脚本的时候:

1. `ERROR 1067 (42000): Invalid default value for 'FAILD_TIME'` (对TIMESTAMP 类型的子段如果不设置缺省值或没有标志not null时候在创建表时会报这个错误)
   这是因为`sql_mode`中的`NO_ZERO_DATE`导制的，在`strict mode`中不允许'0000-00-00'作为合法日期

使用下面的命令查看sql_mode

```
mysql>show variables like 'sql_mode';

+---------------+-------------------------------------------------------------------
--------------------------------------------------------------------------------+
| Variable_name | Value
                                                                                 |
+---------------+-------------------------------------------------------------------
--------------------------------------------------------------------------------+
| sql_mode      |
ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DI
VISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION |
+---------------+-------------------------------------------------------------------
--------------------------------------------------------------------------------+
```

将上面的`NO_ZERO_DATE`改为下面的`ALLOW_INVALID_DATES`

```
mysql> set
sql_mode='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,ALLOW_INVALID_D
ATES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION';
```

上面的设置是临时设置，在重新登陆后，该设置又恢复为NO_ZERO_DATE

安装Django和相关模块

手动：

pip install django==3.1

pip install guardian

pip install mysqlclient

pip install jira

自动：

导出项目需要安装的模块：pip freeze > requirements.txt

安装需要的模块：pip install -r requirements.txt

python manage.py runserver #检查django相关的模块是否安装好

###

代码包上传

方式1.打包上传

方式2.git同步

Django托管服务器uWSGI

什么是uWSGI

uWSGI　　Web Server Gateway Interface

WSGI是Web服务器网关接口。属于web服务器和应用程序之间的通信协议

uuwsgi

uwsgi协议是uWSGI服务器使用的本地协议。它是一个二进制协议，可以携带任何类型的数据。属性线路协议。

uuWSGI

uWSGI是一个全功能的HTTP服务器，实现了WSGI协议、uwsgi协议、http协议等。它要做的就是把HTTP协议转化成语言支持的网络协议。比如把HTTP协议转化成WSGI协议，让Python可以直接使用。

为什么选择uWSGI

同等条件下，uWSGI的性能表现最强

https://blog.kgriffs.com/2012/12/18/uwsgi-vs-gunicorn-vs-node-benchmarks.html

如何使用uWSGI

安装 pip install uwsgi

配置 见附录

使用

```
启动: uwsgi  xxx.ini    #   ini是配置文件，保存启动项参数
重启: uwsgi -reload   xxx.pid    # pid是进程文件
停止: uwsgi  --stop     xxx.pid
```


BugFix

uwsgi问题

uwsgi启动提示 The app module <module 'sqtp' (namespace)> has multiple filesystem locations错误，导致服务启动不了

```
*** Operational MODE: single process ***
Traceback (most recent call last):
  File "./autotpsite/wsgi.py", line 16, in <module>
    application = get_wsgi_application()
  File "/root/software/djenv/lib/python3.6/site-packages/django/core/wsgi.py",
line 12, in get_wsgi_application
    django.setup(set_prefix=False)
  File "/root/software/djenv/lib/python3.6/site-packages/django/__init__.py",
line 24, in setup
    apps.populate(settings.INSTALLED_APPS)
  File "/root/software/djenv/lib/python3.6/site-
packages/django/apps/registry.py", line 91, in populate
```

```
    app_config = AppConfig.create(entry)
  File "/root/software/djenv/lib/python3.6/site-packages/django/apps/config.py",
line 255, in create
    return app_config_class(app_name, app_module)
  File "/root/software/djenv/lib/python3.6/site-packages/django/apps/config.py",
line 49, in __init__
    self.path = self._path_from_module(app_module)
  File "/root/software/djenv/lib/python3.6/site-packages/django/apps/config.py",
line 91, in _path_from_module
    "with a 'path' class attribute." % (module, paths))
django.core.exceptions.ImproperlyConfigured: The app module <module 'sqtp'
(namespace)> has multiple filesystem locations (['./sqtp',
'/root/software/autotpsite007/sqtp']); you must configure this app with an
AppConfig subclass with a 'path' class attribute.
```

解决方法：

根据错误提示增加path配置

```
# sqtp/apps.py
class SqtpConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'sqtp'
    path = './autotpsite/sqtp'  # 增加这行代码
```

CentOs7+Python3.8 安装库问题

2 CentOs7+Python3.8安装库提示：ModuleNotFoundError: No module named '_ctypes'

```
    Complete output (11 lines):
    Traceback (most recent call last):
      File "<string>", line 1, in <module>
      File "/usr/local/lib/python3.8/site-packages/setuptools/__init__.py", line
18, in <module>
        from setuptools.dist import Distribution
      File "/usr/local/lib/python3.8/site-packages/setuptools/dist.py", line 34,
in <module>
        from setuptools import windows_support
      File "/usr/local/lib/python3.8/site-
packages/setuptools/windows_support.py", line 2, in <module>
        import ctypes
      File "/usr/local/lib/python3.8/ctypes/__init__.py", line 7, in <module>
        from _ctypes import Union, Structure, Array
    ModuleNotFoundError: No module named '_ctypes'
```

解决方法：

一键安装脚本已经更新，直接安装应该不存在这个问题

卸载Python3

```
whereis python3 |xargs rm -frv
```

重新安装python3

```
curl -O  https://gitee.com/shonekey666/LinuxEnv/raw/master/CentOS/CentOS7-installpython3-4.sh && sh CentOS7-installpython3-4.sh
```

附录

Django部署清单

https://docs.djangoproject.com/zh-hans/2.2/howto/deployment/checklist/

centos7防火墙工具firewall-cmd使用介绍

1、firewalld的基本使用

启动： systemctl start firewalld

查看状态： systemctl status firewalld

停止： systemctl disable firewalld

禁用： systemctl stop firewalld

2. 服务管理工具介绍

systemctl是CentOS7的服务管理工具中主要的工具，它融合之前service和chkconfig的功能于一体。

启动一个服务：systemctl start firewalld.service
关闭一个服务：systemctl stop firewalld.service
重启一个服务：systemctl restart firewalld.service
显示一个服务的状态：systemctl status firewalld.service
在开机时启用一个服务：systemctl enable firewalld.service
在开机时禁用一个服务：systemctl disable firewalld.service
查看服务是否开机启动：systemctl is-enabled firewalld.service
查看已启动的服务列表：systemctl list-unit-files|grep enabled
查看启动失败的服务列表：systemctl --failed

3.配置firewalld-cmd

查看版本： firewall-cmd --version

查看帮助： firewall-cmd --help

显示状态： firewall-cmd --state

查看所有打开的端口： firewall-cmd --zone=public --list-ports

更新防火墙规则： firewall-cmd --reload

查看区域信息: firewall-cmd --get-active-zones

查看指定接口所属区域： firewall-cmd --get-zone-of-interface=eth0

拒绝所有包：firewall-cmd --panic-on

取消拒绝状态： firewall-cmd --panic-off

查看是否拒绝： firewall-cmd --query-panic

那怎么开启一个端口呢

添加

firewall-cmd --zone=public --add-port=80/tcp --permanent （--permanent永久生效，没有此参数重启后失效）

重新载入

firewall-cmd --reload

查看-指定端口是否开启

firewall-cmd --zone=public --query-port=80/tcp

查看已开启的端口

firewall-cmd --list-port

删除

firewall-cmd --zone=public --remove-port=80/tcp --permanent

Git安装与配置

https://www.runoob.com/git/git-install-setup.html

uWSGI文档

英文版：https://uwsgi-docs.readthedocs.io/en/latest/Download.html

中文版：https://uwsgi-docs-zh.readthedocs.io/zh_CN/latest/Download.html

uWSGI配置

```
[uwsgi]
chdir = /root/software/autotpsite/
//项目根目录
module = autotpsite.wsgi:application
//  指定wsgi模块下的application对象
http = 0.0.0.0:8081
#http-socket = 0.0.0.0:8081
//对本机8081端口提供服务
master = true
//主进程

# 以上4个是核心配置项
```

```
#vhost = true           //多站模式
#no-site = true         //多站模式时不设置入口模块和文件
#workers = 2            //子进程数
#reload-mercy = 10
#vacuum = true          //退出、重启时清理文件
#max-requests = 1000
#limit-as = 512
#buffer-size = 30000
pidfile = /root/software/autotpsite/uwsgi8081.pid
//pid文件，用于下脚本启动、停止该进程
daemonize = /root/software/autotpsite/uwsgi_server.log
// 日志文件
disable-logging = true
//不记录正常信息，只记录错误信息
```

卸载python3

```
whereis python3 |xargs rm -frv
```