部署

- django中自带开发者服务器
  - runserver
    - 路由处理功能，动态资源处理
    - 如果是debug的话，静态资源处理功能
  - 功能健壮，性能是比较低的，仅适用于开发
- 部署不会使用单一服务器
  - Apache
  - Nginx
    - HTTP服务器
      - 处理静态资源
    - 反向代理
      - uWSGI HTTP服务器
      - gunicorn HTTP服务器
    - 邮件服务器
    - 流媒体服务器

Nginx简介

Nginx是一个高性能的HTTP和反向代理服务器，也是一个IMAP/POP3/SMTP服务器。

Nginx是一款轻量级的Web服务器/反向代理服务器以及电子邮件代理服务器，其特点是占用内存少，并发能力强，在同类型的网页服务器中表现优秀

Nginx是由伊戈尔.塞索耶夫开发的，于2004年10月4日公开源码，以类BSD许可证形式发布

Nginx因它的稳定性，丰富的功能，示例配置文件和低系统资源的消耗而闻名

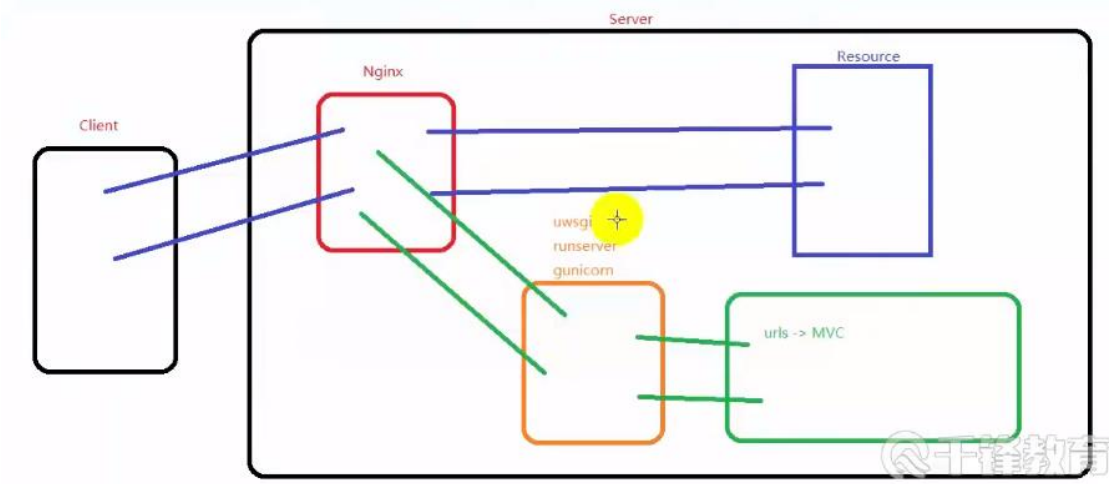
中国大陆使用Nginx的网站:
淘宝，京东，腾讯，百度，新浪，网易...

Nginx简介

官网
http://nginx.org/

中文资料
http://www.nginx.cn/doc/index.html
http://tengine.taobao.org/book/

为什么选择Nginx
作为We服务器：相比Apache，Nginx使用资源更少，支持更多的并发连接，体现更高的效率，使Nginx倍受欢迎，能够支持高达50000个并发连接数的响应
作为负载均衡服务器：Nginx既可以在内部直接支持Redis和PHP，也可以支持作为HTTP代理服务器对外进行服务，Nginx使用C编写，不论是系统资源开销还是CPU使用效率都处理的非常优秀
Nginx安装非常简单，配置文件非常简洁，Bug非常少：Nginx启动非常容易，并且几乎可以做到7 * 24小时不间断运行，即使运行数个月也不需要重新启动



Nginx控制

启动Nginx

```
nginx    [-c configpath]
```

信息查看

```
nginx    -v
nginx    -V
```

控制Nginx

```
nginx -s signal
        stop        快速关闭
        quit        优雅的关闭
        reload      重新加载配置
```

通过系统管理

```
systemctl status nginx      查看nginx状态
systemctl start  nginx      启动nginx服务
systemctl stop   nginx      关闭nginx服务
systemctl enable nginx      设置开机自启
systemctl disable nginx     禁止开机自启
```

Nginx配置文件

Nginx配置文件包含指定指令控制的模块。

指令分为简单指令和块指令

一个简单指令由名称和参数组成，以空格分隔，并以分号结尾

一个块指令和简单指令具有相同的结构，但不是以分号结束，而是以一个大括号包围的一堆附加指令结束

如果一个大括号内可以有其他的指令，它就被称为一个上下文，比如（events，http，server，location）

指令

```
nginx     -t                不运行，仅测试配置文件
nginx     -c  configpath    从指定路径加载配置文件
nginx  -t -c  configpath    测试指定配置文件
```

Nginx配置文件结构

```
main                全局设置

events{             工作模式，连接配置
        ...
}
http{               http的配置
        ...
        upstream xxx{       负载均衡配置
                ...
        }
        server{             主机设置
                ...
                location xxx{       URL匹配
                        ...
                }
        }
}
```

main

```
user    nginx;    worker进程运行的用户和组

worker_processes  1;        指定Nginx开启的子进程数，多核CPU建议设置和CPU数量一样的进程数

error_log  xxx level;                用来定义全局错误日志文件，通常放在var中，level有 debug，info，notice，
                                                                                warn，error，crit

pid     xxx;                指定进程id的存储文件位置
```

events

指定工作模式和以及连接上限

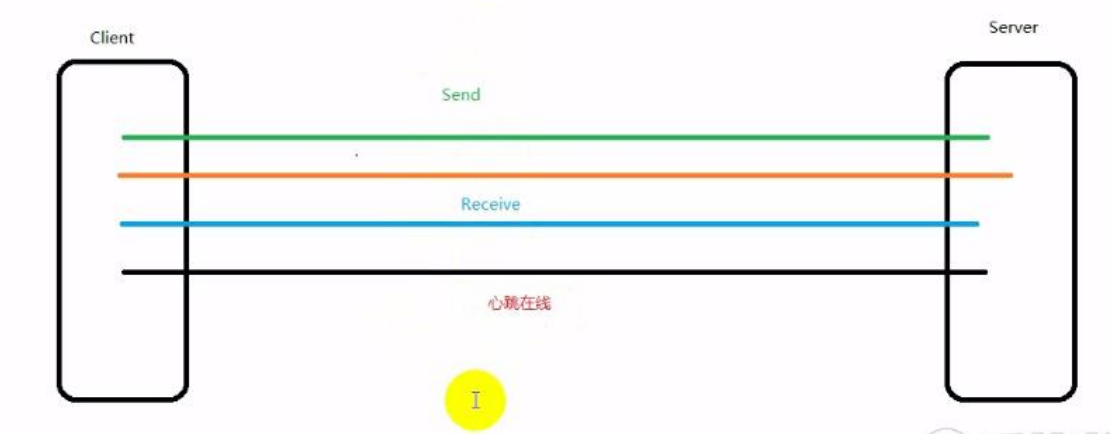
```
events{
        use epoll;
        worker_connections 1024;
}
```

use 指定nginx工作模式

```
        epoll       高效工作模式，linux
        kqueue      高效工作模式，bsd
        poll        标准模式
        select      标准模式
```

worker_connections 定义nginx每个进程的最大连接数

```
        正向代理            连接数 * 进程数
        反向代理            连接数 * 进程数 / 4
        linux系统限制最多能同时打开65535个文件，默认上限就是65535，可解除 ulimit -n 65535
```



http

最核心的模块，主要负责http服务器相关配置，包含server，upstream子模块

include mime.types;设置文件的mime类型

include xxxconfig; 包含其它配置文件，分开规划解耦

default_type xxx; 设置默认类型为二进制流，文件类型未知时就会使用默认

log_format 设置日志格式

sendfile 设置高效文件传输模式

keepalive_timeout 设置客户端连接活跃超时

gzip gzip压缩

开启 uwsgi：

```
You should consider upgrading via the 'pip install --upgrade pip' command.
(GP1) rock@Rock:~/GP1/Day10/GPAXF$ uwsgi --ini /home/rock/GP1/Day10/GPAXF/uwsgi.ini
[uWSGI] getting INI configuration from /home/rock/GP1/Day10/GPAXF/uwsgi.ini
(GP1) rock@Rock:~/GP1/Day10/GPAXF$
```

开启 gunicorn：

You can now run the app with the following command:

```
$ gunicorn --workers=2 test:app
```

Django

Gunicorn will look for a WSGI callable named `application` if not specified. So for a typical Django project, invoking Gunicorn would look like:

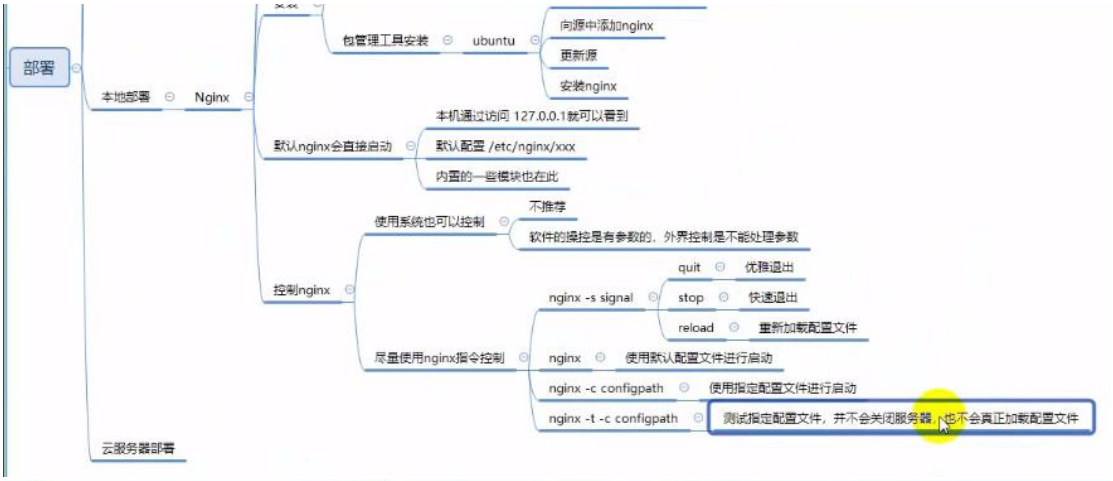
```
$ gunicorn myproject.wsgi
```

部署云服务器

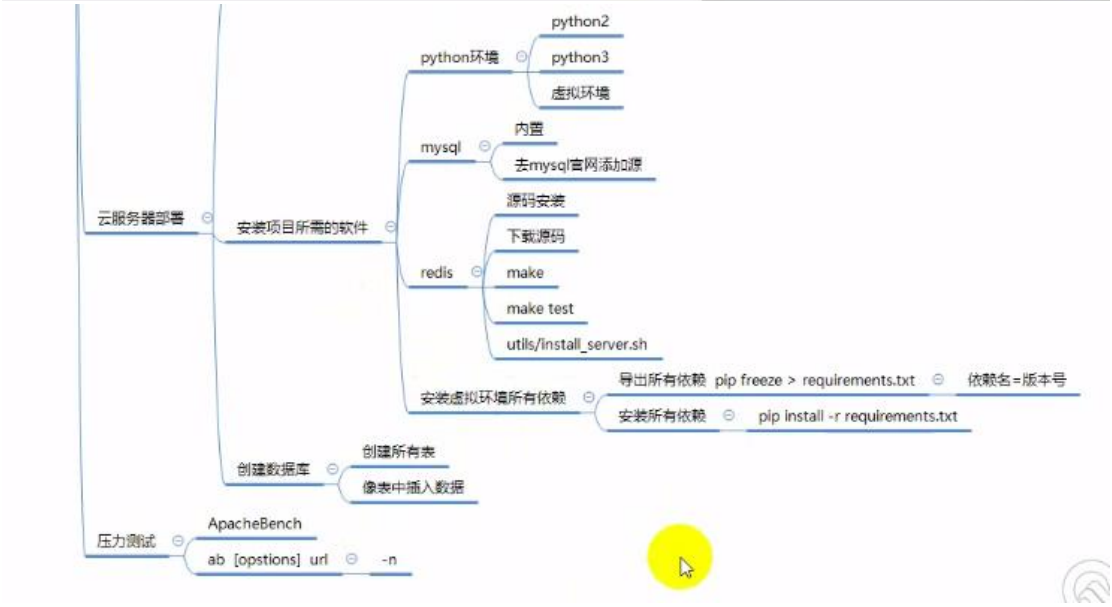
- 从零开始做的
- 安装云服务器系统
  - Ubuntu 16.04
- 安装一套开发环境
  - Python
    - 2.x
    - 3.x
  - pip
    - 注意版本兼容
  - virtualenv
    - 版本不兼容
    - workon_home
    - source xxx
- virtualenv
  - 版本不兼容
  - workon_home
  - source xxx
- mysql
  - apt 直接安装
- redis
  - 源码安装
  - make & make test
  - utils/install_server.sh
- nginx
  - 添加钥匙
  - 添加源
  - update， install
- 准备进行部署
  - 安装项目所需依赖
    - pip install -r xxx.txt
  - 修改配置文件到指定路径
  - 从静态文件开始部署
  - 动态资源
    - 处理好数据
    - 创建库，创建表
    - 插入数据
- 坑点
  - 邮件发送
    - 25端口是非安全端口，阿里不允许使用
    - 使用安全SSL端口 465

阿帕奇测试工具：

```
[root@localhost ~]# ab
ab: wrong number of arguments
Usage: ab [options] [http[s]://]hostname[:port]/path
Options are:
    -n requests     Number of requests to perform
    -c concurrency  Number of multiple requests to make at a time
    -t timelimit    Seconds to max. to spend on benchmarking
```




events
工作模式
工作进程处理的最大连接数
http
upstream
service
location
路由匹配
配置过程
使用Nginx处理静态资源
单独使用uwsgi处理动态资源
安装使用uwsgi
key=value
修改工程位置和wsgi位置即可
对接Nginx
只需将配置中的HTTP模式修改为SOCKET模式
云服务器部署
从安装系统开始的
Aliyun中可以选择纯净的系统使用
也可以去镜像市场选择一些搭建好环境的
安装项目所需的软件
python环境
python2
python3
虚拟环境
mysql
内置
去mysql官网添加源
redis


云服务器部署
安装项目所需的软件
python环境
python2
python3
虚拟环境
mysql
内置
去mysql官网添加源
redis
源码安装
下载源码
make
make test
utils/install_server.sh
安装虚拟环境所有依赖
导出所有依赖 pip freeze > requirements.txt
依赖名=版本号
安装所有依赖
pip install -r requirements.txt
创建数据库
创建所有表
像表中插入数据
压力测试
ApacheBench
ab [opstions] url
-n

Django项目部署

django 服务器
- runserver
- wsgi

uwsgi : web服务器,多线程处理的不错
1. pip install uwsgi
2. 工程目录下创建uwsgi.ini 配置文件
3. 书写配置信息
4. 使用uwsgi服务器
   - 启动　　uwsgi --ini uwsgi.ini
   - 停止　　uwsgi --stop uwsgi.pid

nginx配置

```
location /static{
        alias xxx/static/;
}
location / {
        include uwsgi_params;
        uwsgi_pass localhost:8000;
}
```

反向代理

proxy_pass URL;	反向代理转发地址，默认不转发header，需要转发header则设置 proxy_set_header HOST $host;
proxy_method POST;	转发的方法名
proxy_hide_header Cache-Control;	指定头部不被转发
proxy_pass_header Cache-Control;	设置哪些头部转发
proxy_pass_request_header on;	设置转发http请求头
proxy_pass_request_body on;	设置转发请求体

```
server {
    listen       80;
    server_name  localhost;

    root   /home/rock/GP1/Day10/GPAXF;

    location /static {
        alias /home/rock/GP1/Day10/GPAXF/static;
    }

    #location / {
    #    include /etc/nginx/uwsgi_params;
    #    uwsgi_pass 127.0.0.1:8888;
    #}

    location / {
        proxy_pass http://127.0.0.1:9000;
    }
}
```

upstream

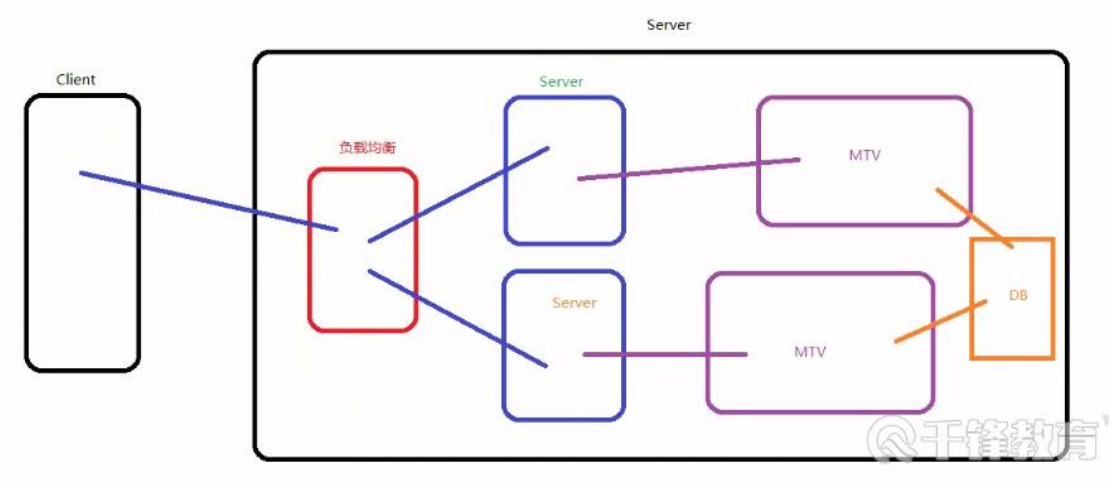
负载均衡模块，通过一个简单的调度算法来实现客户ip到后端服务器的负载平衡

写法 upstream myproject{

```
        ip_hash;
        server 127.0.0.1:8000;
        server 127.0.0.1:8001 down;
        server 127.0.0.1:8002 weight=3;
        server 127.0.0.1:8003 backup;
        fair;
    }
```

负载均衡算法

weight	负载权重
down	当前server不参与负载均衡
backup	其它机器全down掉或满载使用此服务
ip_hash	按每个请求的hash结果分配
fair	按后端响应时间来分（第三方的）



```
    #gzip  on;

    upstream my_server{
        server 10.0.122.88:8000 weight=1;
        server 10.0.122.64:8000 weight=1;
    }

    server {
        listen       80;
        server_name  localhost;

        root   /home/rock/GP1/Day10/GPAXF;

        location /static {
            alias /home/rock/GP1/Day10/GPAXF/static;
        }

        #location / {
        #    include /etc/nginx/uwsgi_params;
        #    uwsgi_pass 127.0.0.1:8888;
        #}

        location / {
        #    proxy_pass http://127.0.0.1:9000;
            proxy_pass http://my_server;
        }
    }

}
```

负载均衡：