# 课程介绍

* 1. nginx服务器简介;（了解）
* 2. 安装Nginx;（了解）
* 3. Nginx配置文件；（掌握）
* 4. Nginx虚拟主机及反向代理；（掌握）

1. Nginx简介

Web服务器的技术,主要是为了解决三个方面的问题：

1. 如何更快地处理和响应客户端请求；

2. 保持服务器的稳定.如,有大量并发请求时,如何提供不间断的服务；

3. 实施步骤简单,部署迅速；

目前,市场上比较流行的Web服务器有:**Apache**, Microsoft IIS (Microsoft Internet Information Services), Sun, **Nginx**, Tomcat, Lighttpd等；

* 1. 什么是Nginx?

**Nginx服务器(engine-X)**，是一种web服务器。它是一种开源的高性能HTTP和反向代理服务器。 它也可提供IMAP/POP3/SMTP代理服务等功能。

Nginx服务器由俄国的Igor Sysoev为Rambler.ru站点开发。自Nginx发布来，因其稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。

Nginx服务器所占市场份额不断提升, 2017年,已占到21.4%。

**在国内谁在使用Nginx服务器?**

国内使用Nginx服务器的网站有：新浪、网易、腾讯、CSDN、酷六、水木社区、豆瓣、六房间、小米等.

* 1. Nginx服务器的架构

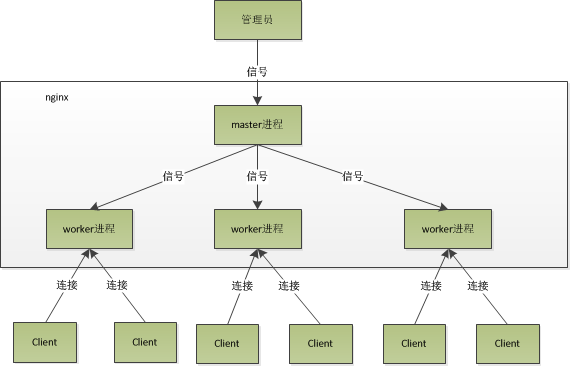
**Nginx是以多进程的方式来工作的。**也支持多线程的方式，只是主流的方式还是多进程的方式，也是Nginx的默认方式。

Nginx启动后，它在Unix系统中会以daemon的方式在后台运行，后台进程包含一个master进程和多个worker进程。

**master进程主要用来管理worker进程**，包含：接收来自外界的信号，向各worker进程发送信号，监控worker进程的运行状态，当worker进程退出后(异常情况下)，会自动重新启动新的worker进程。

而基本的网络事件，则是放在worker进程中来处理了。多个worker进程之间是对等的，他们同等竞争来自客户端的请求，各进程互相之间是独立的。

一个请求，只可能在一个worker进程中处理，一个worker进程，不可能处理其它进程的请求。worker进程的个数是可以设置的，一般我们会设置与机器cpu核数一致。



* 1. Nginx服务器的优点 (了解)
     + 1. **高并发连接**

官方测试能支撑5万并发连接，在实际生产环境中跑到2～3万并发连接数。Nginx服务器专为性能优化而开发。它支持内核Poll模型，能经受高负载的考验。有报告表明能支持高达50,000个并发连接数。

* + - 1. **内存消耗少**

Nginx服务器采取了分阶段资源分配技术，这使得它的CPU与内存占用率非常低。Nginx服务器官方表示: 保持10,000个没有活动的连接，它只占2.5M内存。 类似DOS这样的攻击对Nginx服务器来说毫无用处。 在3万并发连接下，开启的10个Nginx服务器进程才消耗150M内存（15M\*10=150M）。

* + - 1. **配置文件简单**
      2. **成本低**

Nginx服务器是开源软件。 而购买F5 BIG-IP、NetScaler等硬件负载均衡交换机则需要十万人民币以上。

* + - 1. **支持Rewrite重写规则**

能根据域名、URL的不同，将 HTTP 请求分到不同的后端服务器群组。

* + - 1. **内置的健康检查功能.**

若 Nginx服务器 Proxy 后端的某台 Web 服务器宕机，不会影响前端访问。

* + - 1. **节省带宽.**

支持 GZIP 压缩，可以添加浏览器本地缓存的 Header 头。

* + - 1. **稳定性高.**

Nginx服务器的稳定性很高。其它HTTP服务器在遇到访问峰值，或有人恶意发起慢速连接时，很可能会导致服务器物理内存耗尽，频繁交换，失去响应(只能重启服务器)。

另外, Nginx服务器采用master-slave模型，能充分利用SMP的优势，且能减少工作进程在磁盘I/O的阻塞延迟。

Nginx服务器代码质量很高，很规范, 模块扩展也很容易。Nginx服务器采用了一些os提供的最新特性如对sendfile (Linux2.2+)，accept-filter(FreeBSD4.1+)，TCP\_DEFER\_ACCEPT (Linux 2.4+)的支持，从而大大提高了性能。

* 1. Nginx和Apache的区别

**相同点**:

* + - 1. 都是HTTP服务器软件；
      2. 功能实现上都采用模块化结构设计；
      3. 都支持通用的语言接口(如PHP、Perl、Python)；
      4. 都支持正、反向代理，虚拟主机，URL重写，压缩传输，SSL加密传输等；

**差别**:

* + - 1. Apache处理速度较慢，且占用很多内存资源，而Nginx服务器却相反；
      2. 在功能实现上，Apache的所有模块都支持动、静态编译，而Nginx服务器模块都是静态编译的;
      3. 在处理连接方式上，Nginx支持epoll，而Apache不支持；
      4. Nginx服务器安装包很小,只有几百KB；

1. 安装Nginx
   1. Linux下yum安装

目前Nginx官网已经支持了很多LTS版本的的yum源，点击“download”里有很清楚的配置方法。

* + - 1. 产生yum配置文件/etc/yum.repos.d/nginx.repo
      2. 编写如下内容：

[nginx]

name=nginx repo

baseurl=http://nginx.org/packages/OS/OSRELEASE/$basearch/

gpgcheck=0

enabled=1

其中OS可以选择：

“rhel” or “centos”

OSRELEASE，选择6，或者7即可。

* + - 1. yum install nginx就可以安装了。
  1. Linux下源码安装（了解）

Nginx有模块依赖性，需要依赖下面3个包，才能源码编译安装。

* gzip 模块需要zlib库
* rewrite 模块需要pcre库
* ssl 功能需要openssl库
  1. 控制服务

如果使用官网已经安装好的对应版本，默认将安装在/usr/sbin/目录下会有nginx可执行文件，通过命令：ps -aux | grep nginx可以来判断当前系统是否有nginx进程运行。

若没有运行，可以使用如下方法运行：

方法1：

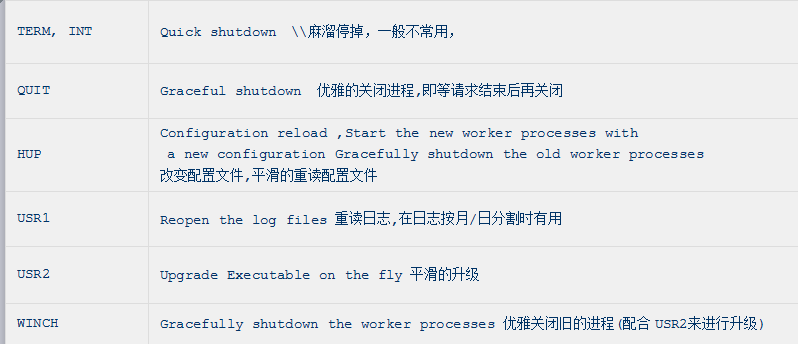
sudo /etc/init.d/nginx restart/start/stop

方法2：

sudo /usr/sbin/nginx

* + 1. 信号控制（了解）

所有的Nginx版本都支持linux的信号控制master进程。



* + 1. 使用命令参数来控制

从nginx0.8版本以后，推荐使用命令参数来控制，方便记忆。

/usr/local/nginx/sbin/nginx -s stop(quit、reload)

其他参数：

-t 检测配置文件语法是否有问题

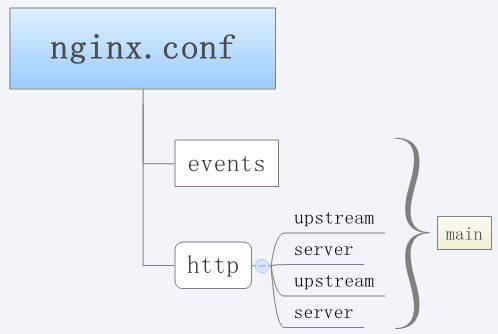
1. Nginx配置文件
   1. Nginx服务器的主配置文件
      1. 认识配置文件

Linux下基本上每个服务都会有它的主配置文件，该文件定义服务应该如何运行， 使用什么参数? 启用什么功能? 会涉及到的相关操作文件在哪? 主配置文件对服务至关重要。

Nginx服务器的主配置文件默认位于/etc/nginx/nginx.conf中。

* + 1. **Nginx的配置文件结构**

nginx.conf由多个块组成。最外面的一块是main，main包含events和http，http包含多个upstream和多个server，server又包含多个location块：



main（全局设置）、server（虚拟主机设置）、upstream（负载均衡服务器设置）和 location（URL匹配特定位置的设置）。

* main块设置的指令将影响其他所有设置；
* server块的指令主要用于指定主机和端口,以及网站路径；
* upstream指令主要用于负载均衡，设置一系列的后端服务器；
* location块用于匹配网页位置；

这四者之间的关系：server继承main，location继承server，upstream既不会继承其他设置也不会被继承。

这四部分的每个部分都包含若干指令，这些指令主要包含Nginx服务器的主模块指令、事件模块指令和HTTP核心模块指令。 每个部分还可使用其他HTTP模块指令， 如Http SSL模块， HttpGzip Static模块和Http Addition模块等。

* + 1. **Nginx的全局配置**

1. #指定使用的用户和组
2. #user  nginx nginx;
4. #启动进程,通常设置成和cpu的数量相等
5. worker\_processes  1;
7. #全局错误日志
8. #error\_log  logs/error.log;
9. #error\_log  logs/error.log  notice;
10. #error\_log  logs/error.log  info;
12. #PID文件--存放进程号的文件
13. #pid        logs/nginx.pid;
    1. events配置
14. #工作模式及连接数上限
15. events {
16. #单个后台worker process进程的最大并发链接数
17. worker\_connections  1024;
18. #并发总数是 worker\_processes 和 worker\_connections 的乘积
20. #Nginx服务器支持如下处理连接的方法（I/O复用方法），这些方法可以通过 use指令指定.
21. #use [ kqueue | rtsig | epoll | /dev/poll | select | poll];
23. use epoll;  #使用 epoll（linux2.6的性能方式 ）
24. }

注意: 服务器为linux时, 请用use epoll。 利用linux的内核提供性能优化方案。

* 1. http配置

1. http {
2. #Nginx服务器对HTTP服务器相关属性的配置
3. include       mime.types;
4. default\_type  application/octet-stream;
5. #设定虚拟主机配置
6. server {
7. #侦听80端口
8. listen       80;
9. #定义使用 www.itsource.cn访问
10. server\_name  www.itsource.cn;
11. location  {
12. }
13. location  {
14. }
15. …..
16. }
17. server {
18. #侦听80端口
19. listen       80;
20. #定义使用 www.example.cn访问
21. server\_name  www.example.cn;
22. location  {
24. }
25. location  {
26. }
27. …..
28. }
29. }
    * + 1. include是个主模块指令。它实现对配置文件所包含的文件的设定，以减少主配置文件的复杂度。
        2. default\_type属于HTTP核心模块指令. 这里设定默认类型为二进制流，也就是当文件类型未定义时使用这种方式。
        3. server块的指令主要用于指定主机和端口(虚拟主机).
        4. location部分主要用于匹配网页位置,设置不同的功能特征. 比如:缓存,重定向等..
    1. [Nginx重要指令: location](http://www.nginx.cn/115.html)

location主要用于匹配网页位置，设置不同的功能特征. 比如缓存，重定向等。

**实例:**

以.gif、.jpg、.jpeg、.png、.bmp结尾的静态文件都交给Nginx处理，用expires指定静态文件的过期时间(这里是30天)。

1. location ~ .\*\.(gif|jpg|jpeg|png|bmp)$  {
2. root    /wwwroot/www.itsource.com;
3. expires 30d;
4. }

upload/和html/下的所有文件都交给Nginx来处理(upload和html目录包含在/web/wwwroot/www.itsource.cn目录中).

1. location ~ ^/(upload|html)/  {
2. root    /web/wwwroot/www.itsource.com;
3. expires 30d;
4. }

**location的语法:**

~      #执行一个正则匹配，区分大小写

~\*    #执行一个正则匹配，不区分大小写

^~    #普通字符匹配，若该选项匹配，只匹配该选项.不匹配别的选项，一般用来匹配目录

=      #普通字符精确匹配

location中可接受以下语法规则

1. location  = / {
2. # 只匹配"/".
3. [ configuration A ]
4. }
5. location  / {
6. # 匹配任何请求，因为所有请求都以"/"开始,
7. # 但更长字符匹配或正则表达式匹配会优先匹配
8. [ configuration B ]
9. }
10. location ^~ /images/ {
11. # 匹配任何以 /images/ 开始的请求，并停止匹配其它location
12. [ configuration C ]
13. }
14. location ~\* \.(gif|jpg|jpeg)$ {
15. # 匹配以 gif, jpg, or jpeg结尾的请求,
16. # 但所有 /images/ 目录的请求将由 [Configuration C]处理.
17. [ configuration D ]
18. }

**优先级:**

* + - 1. =前缀的指令严格匹配这个查询。若找到，停止搜索；
      2. 所有剩下的常规字符串，最长的匹配。若这个匹配使用^〜前缀，则搜索停止；
      3. 正则表达式，在配置文件中定义的顺序。
      4. 若第3条规则产生匹配的话，则结果被使用。否则，如同从第2条规则被使用。

请求URI例子:

/ -> 符合configuration A

/documents/document.html -> 符合configuration B

/images/1.gif -> 符合configuration C

/documents/1.jpg ->符合 configuration D

1. Nginx虚拟主机
   1. 什么是Nginx虚拟主机?

 若我们可使用特殊的软硬件技术把一台服务器主机分成多个部分, 且每一部分都具有独立的域名，具有完整的Intemet服务器功能（WWW、FTP、Email等）,则各部分都称为该服务器上的一台**虚拟主机**. 同一台主机上的虚拟主机之间是完全独立的.从网站访问者来看，每台虚拟主机和一台独立的主机完全一样.

* 1. 标准的虚拟主机配置

1. #默认的虚拟机主机
2. server {
3. listen          80 **default**;
4. server\_name    localhost;
5. access\_log      logs/**default**.access.log main;
6. location / {
7. index index.html;
8. root  /var/www/**default**/htdocs;
9. }
10. }
11. #www.domain1.com对应的虚拟主机
12. server {
13. listen          80;
14. server\_name     www.domain1.com;
15. #access\_log      logs/domain1.access.log main;
16. location / {
17. index index.html;
18. root  /var/www/domain1.com/htdocs;
19. }
20. }
21. #www.domain2.com对应的虚拟主机
22. server {
23. listen          80;
24. server\_name     www.domain2.com;
25. access\_log      logs/domain2.access.log main;
26. location / {
27. index index.html;
28. root  /var/www/domain2.com/htdocs;
29. }
30. }
31. }
    1. 规划虚拟主机的配置文件

以上全部配置在主配置文件中可能不小心修改掉配置文件导致Nginx.conf文件出错. 这种情况下,可使用include指令将每台虚拟主机放在一个单独的配置文件中。

将Nginx服务器目录结构设置为:

nginx

nginx.conf

vhost

-- www.domain1.com.conf

-- www.domain2.com.conf

每个.conf结尾的文件中保存一台虚拟主机(server)的配置信息.在nginx.conf中通过include指令包含这些配置文件.

**在nginx/config/nginx.conf中的配置:**

1. http {
2. #默认的虚拟机主机
3. server {
4. listen          80 **default**;
5. server\_name     \_ \*;
6. access\_log      logs/**default**.access.log main;
7. location / {
8. index index.html;
9. root  /var/www/**default**/htdocs;
10. }
11. }
12. include vhost/\*.conf;
13. }

**在nginx/config/vhost/www.domain1.com.conf中的配置:**

1. server {
2. listen          80;
3. server\_name     www.domain1.com;
4. access\_log      logs/domain1.access.log main;
5. location / {
6. index index.html;
7. root  /var/www/domain1.com/htdocs;
8. }
9. }

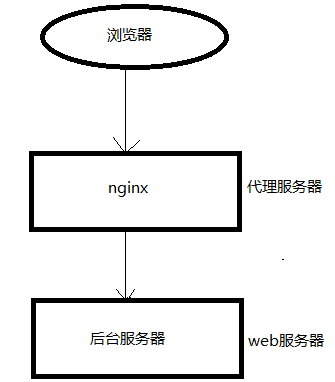
**在nginx/config/vhost/www.domain2.com.conf中的配置:**

1. server {
2. listen          80;
3. server\_name     www.domain2.com;
4. access\_log      logs/domain2.access.log main;
5. location / {
6. index index.html;
7. root  /var/www/domain2.com/htdocs;
8. }

}

1. Nginx的反向代理
   1. 什么是反向代理?

当以代理服务器来接受网络上的连接请求，再将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给网络上发出请求的客户端时，代理服务器对外就表现为一个**反向代理服务器**.



* 1. 两个概念

**web前端服务器**: 代理服务器可以作为前端服务器处理静态资源

**web后端服务器**: web前端服务器无法处理的动态请求发送给后台web服务器处理.

1. 课程总结
   1. 重点
      * 1. Nginx配置文件
        2. 配置WSGI模块
   2. 难点
      * 1. 反向代理
2. 课后练习
   * + 1. 搭建静态虚拟主机
3. 面试题
   * + 1. 什么是web服务器
       2. 什么是WSGI服务器
4. 扩展知识或课外阅读推荐（可选）