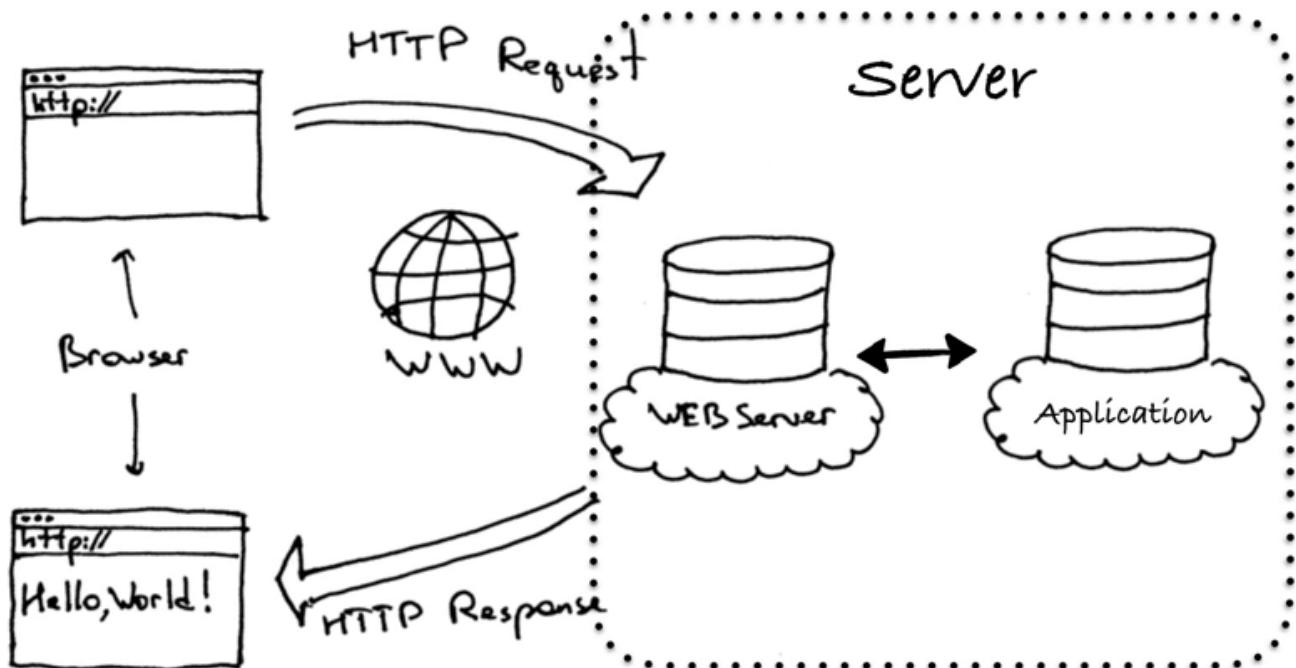


# Nginx

## 一：Web服务基础介绍：

正常情况下的单次web服务访问流程：



### 1.1：互联网发展历程回顾：

1993年3月2日，中国科学院高能物理研究所租用AT&T公司的国际卫星信道建立的接入美国SLAC国家实验室的64K专线正式开通，成为我国连入Internet的第一根专线。

1995年马云开始创业并推出了一个web网站<<中国黄页>>，1999年创建阿里巴巴 [www.alibaba.com](http://www.alibaba.com)，2003年5月10日创立淘宝网，2004年12月，马云创立第三方网上支付平台支付宝（蚂蚁金服旗下，共有蚂蚁金服支付宝、余额宝、招财宝、蚂蚁聚宝、网商银行、蚂蚁花呗、芝麻信用等子业务板块。），2009年开始举办双十一购物狂欢节，以下是历年交易成交额：

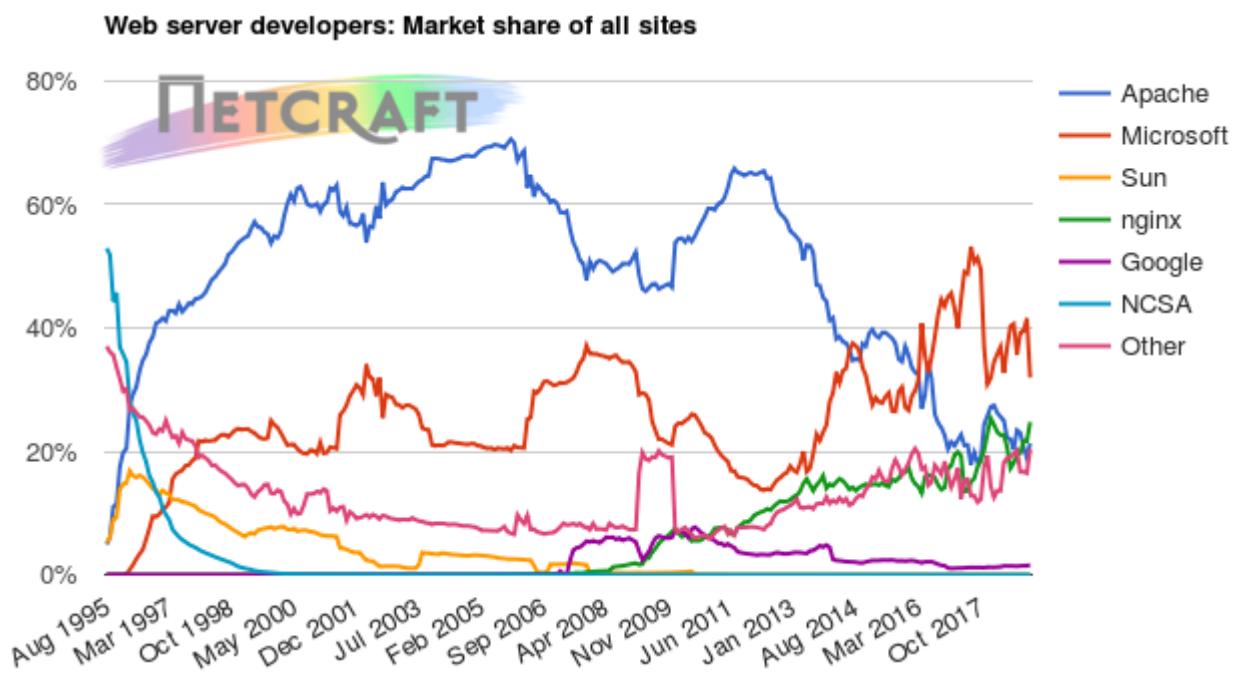
- |    |                     |
|----|---------------------|
| 1  | 2009年双十一：5000万元；    |
| 2  | 2010年双十一：9.36亿元；    |
| 3  | 2011年双十一：33.6亿元；    |
| 4  | 2012年双十一：191亿元；     |
| 5  | 2013年双十一：350亿元；     |
| 6  | 2014年双十一：571亿元；     |
| 7  | 2015年双十一：912.17亿元；  |
| 8  | 2016年双十一：1207亿元；    |
| 9  | 2017年双十一：1682.69亿元； |
| 10 | 2018年双十一：2135亿元；    |

2012年1月11日淘宝商城正式更名为“天猫”。2014年9月19日阿里巴巴集团于纽约证券交易所正式挂牌上市。2018年福布斯统计马云财富346亿美元。

## 1.2 : web服务介绍：

Netcraft公司于1994年底在英国成立，多年来一直致力于互联网市场以及在线安全方面的咨询服务，其中在国际上最具影响力的当属其针对网站服务器，域名解析/主机提供商，以及SSL市场所做的客观严谨的分析研究。

<https://news.netcraft.com/>

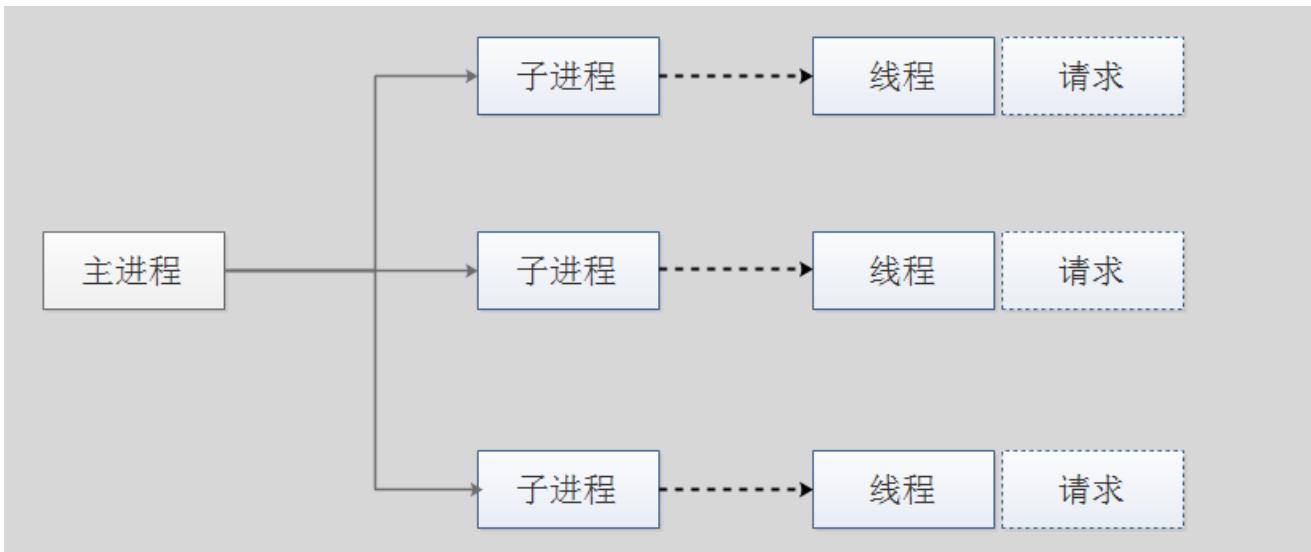


### 1.2.1 : Apace-早期的web服务端：

Apache起初由美国的伊利诺伊大学香槟分校的国家超级计算机应用中心开发，目前经历了两大版本分别是1.X和2.X，其可以通过编译安装实现特定的功能，官方网站：<http://www.apache.org>

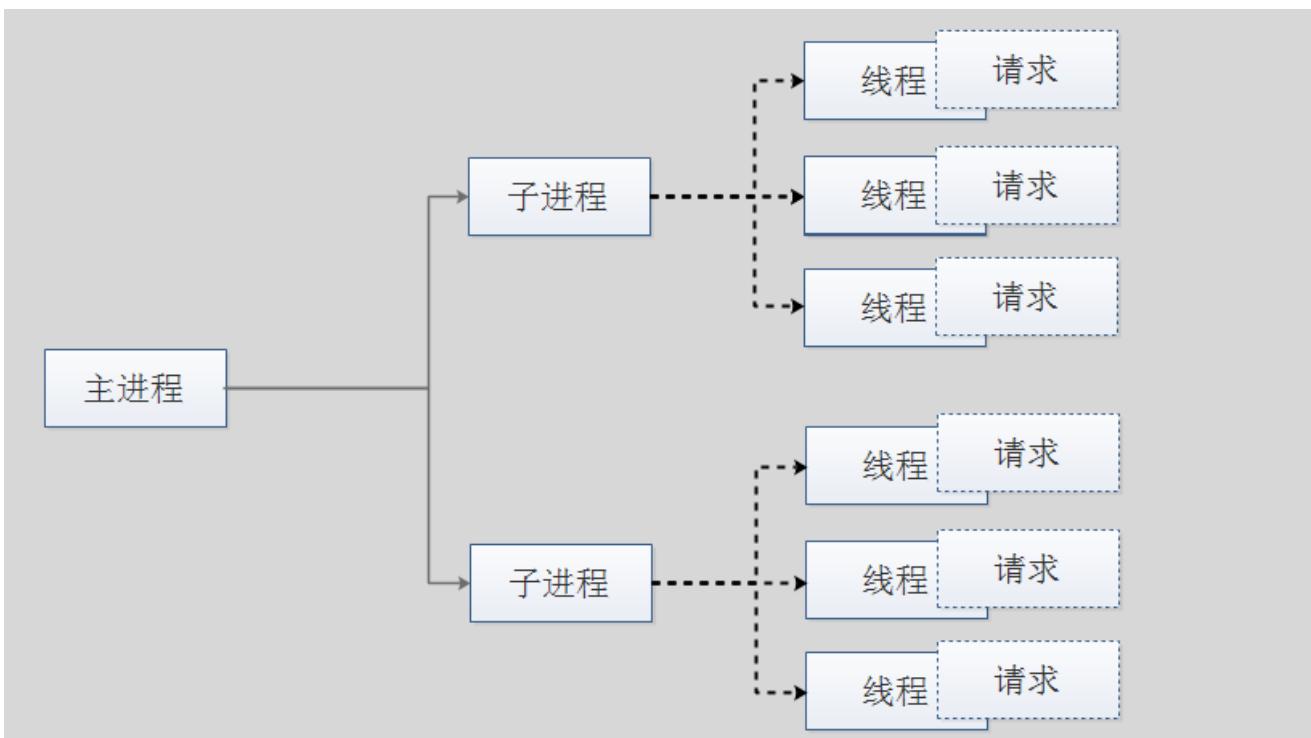
#### 1.2.1.1 : Apache prefork模型：

预派生模式，有一个主控制进程，然后生成多个子进程，使用select模型，最大并发1024，每个子进程有一个独立的线程响应用户请求，相对比较占用内存，但是比较稳定，可以设置最大和最小进程数，是最古老的一种模式，也是最稳定的模式，适用于访问量不是很大的场景。优点：稳定 缺点：慢，占用资源，1024个进程不适用于高并发场景



### 1.2.1.2 : Apache worker模型 :

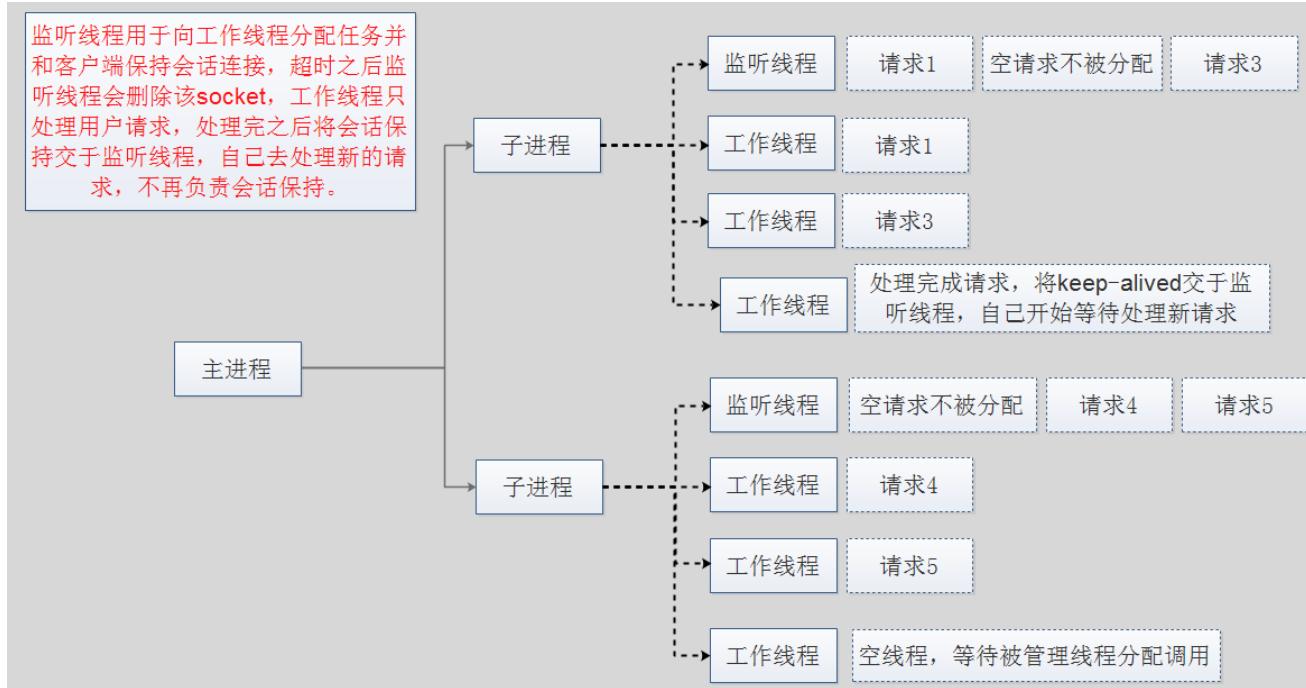
一种多进程和多线程混合的模型，有一个控制进程，启动多个子进程，每个子进程里面包含固定的线程，使用线程来处理请求，当线程不够使用的时候会再启动一个新的子进程，然后在进程里面再启动线程处理请求，由于其使用了线程处理请求，因此可以承受更高的并发。优点：相比prefork 占用的内存较少，可以同时处理更多的请求 缺点：使用keepalive的长连接方式，某个线程会一直被占据，即使没有传输数据，也需要一直等待到超时才会被释放。如果过多的线程，被这样占据，也会导致在高并发场景下的无服务线程可用。（该问题在prefork模式下，同样会发生）



### 1.2.1.3 : Apache event模型 :

Apache中最新的模式，2012年发布的apache 2.4.X系列正式支持event 模型，属于事件驱动模型(epoll)，每个进程响应多个请求，在现在版本里的已经是稳定可用的模式。它和worker模式很像，最大的区别在于，它解决了keepalive场景下，长期被占用的线程的资源浪费问题（某些线程因为被keepalive，空挂在哪里等待，中间几乎没有请求过来，甚至等到超时）。event MPM中，会有一个专门的线程来管理这些keepalive类型的线程，当有真实请求过来的时候，将请求传递给服务线程，执行完毕后，又允许它释放。这样增强了高并发场景下的请求处理能

力。优点：单线程响应多请求，占据更少的内存，高并发下表现更优秀，会有一个专门的线程来管理keep-alive类型的线程，当有真实请求过来的时候，将请求传递给服务线程，执行完毕后，又允许它释放 缺点：没有线程安全控制



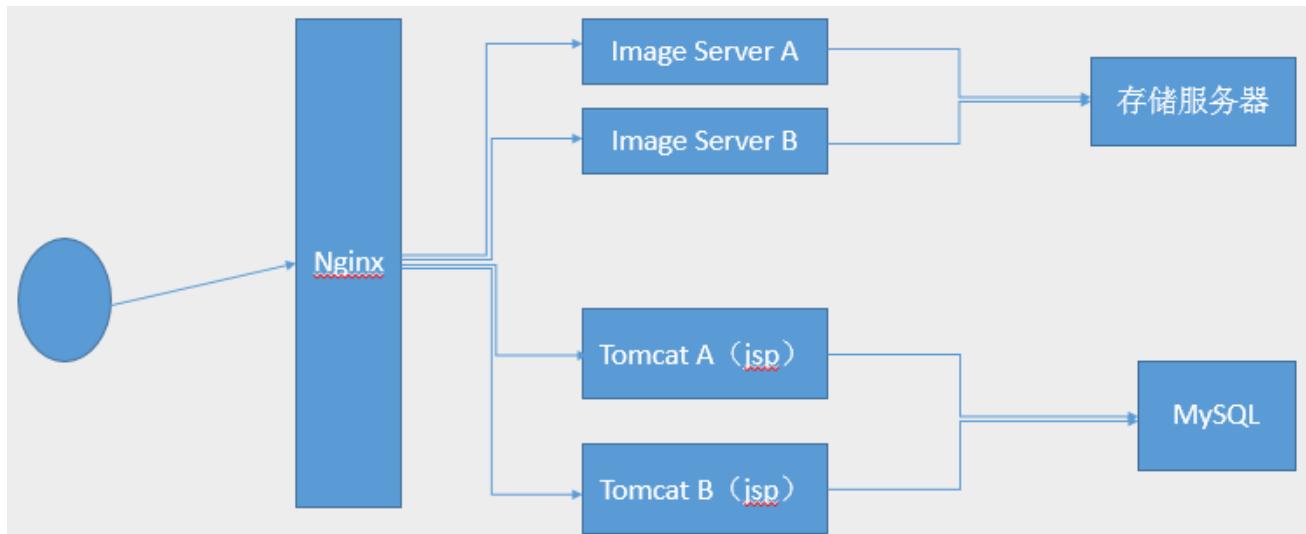
## 1.2.2 : Nginx-高性能的web服务端：

Ninx是由1994年毕业于俄罗斯国立莫斯科鲍曼科技大学的同学为俄罗斯rambler.ru公司开发的，开发工作最早从2002年开始，第一次公开发布时间是2004年10月4日，版本号是0.1.0，官网地址 [www.nginx.org](http://www.nginx.org)

Nginx历经十几年的迭代更新 (<https://nginx.org/en/CHANGES>)，目前功能已经非常完善且运行稳定，另外Nginx的版本分为开发版、稳定版和过期版，nginx以功能丰富著称，它既可以作为http服务器，也可以作为反向代理服务器或者邮件服务器，能够快速的响应静态网页的请求，支持FastCGI/SSL/Virtual Host/URL Rewrite/Gzip/HTTP Basic Auth/http或者TCP的负载均衡(1.9版本以上且开启stream模块)等功能，并且支持第三方的功能扩展。

为什么使用Nginx：天猫 淘宝 小米 163 京东新浪等一线互联网公司都在用Nginx或者进行二次开发

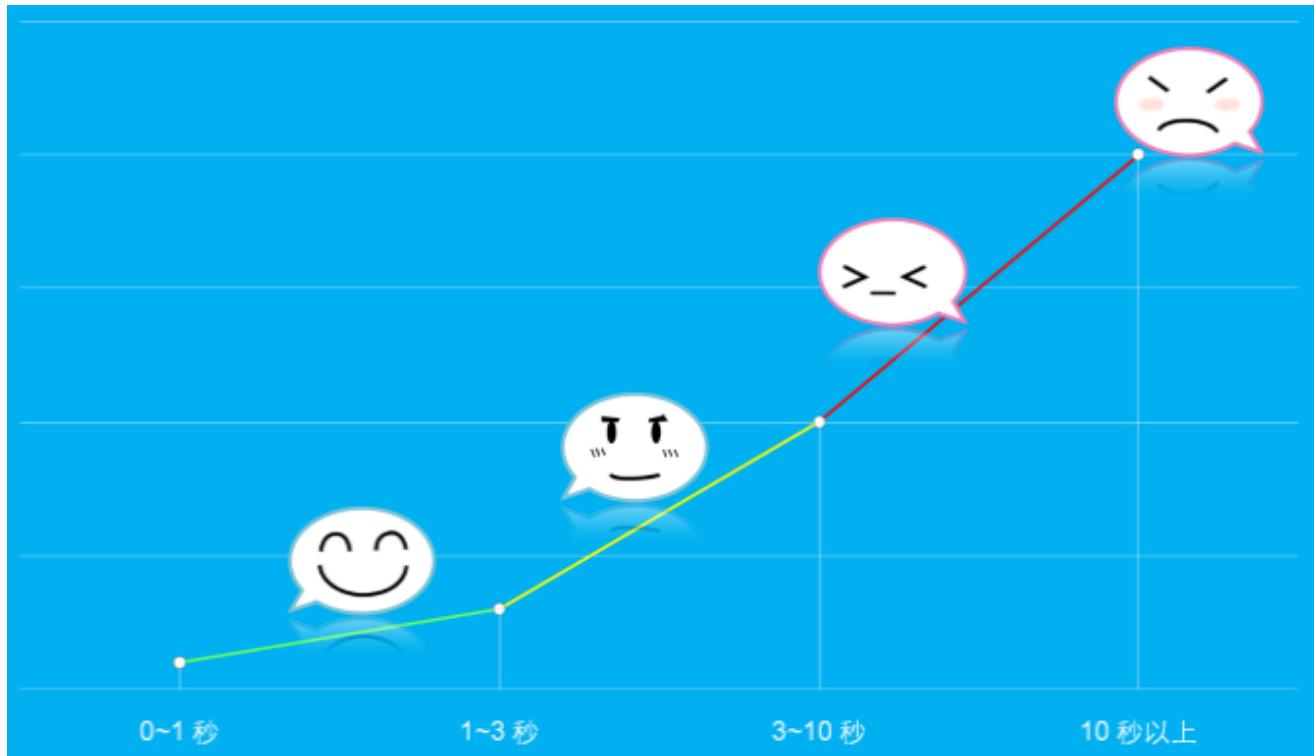
基于Nginx的访问流程如下：



### 1.2.3 : 用户访问体验统计 :

互联网存在用户速度体验的1-3-10原则，即1秒最优，1-3秒较优，3~10秒比较慢，10秒以上用户无法接受。用户放弃一个产品的代价很低，只是换一个URL而已。

全球最大搜索引擎 Google : 慢500ms = 20% 将放弃访问。 全球最大的电商零售网站 亚马逊 : 慢100ms = 1% 将放弃交易



### 1.2.4 : 性能影响 :

有很多研究都表明，性能对用户的行为有很大的影响：79%的用户表示不太可能再次打开一个缓慢的网站 47%的用户期望网页能在2秒钟以内加载 40%的用户表示如果加载时间超过三秒钟，就会放弃这个网站 页面加载时间延迟一秒可能导致转换损失7%，页面浏览量减少11% 8秒定律：用户访问一个网站时，如果等待网页打开的时间超过8秒，会有超过30%的用户放弃等待

#### 1.2.4.1 : 影响用户体验的几个因素 :

据说马云在刚开始创业在给客户演示时，打开一个网站花了两个多小时。

- 1 客户端硬件配置
- 2 客户端网络速率
- 3 客户端与服务端距离
- 4
- 5 服务端网络速率
- 6 服务端硬件配置
- 7 服务端架构设计
- 8 服务端应用程序工作模式
- 9 服务端并发数量
- 10 服务端响应文件大小及数量
- 11 服务端I/O压力

### 1.2.4.2 : 应用程序工作模式 :

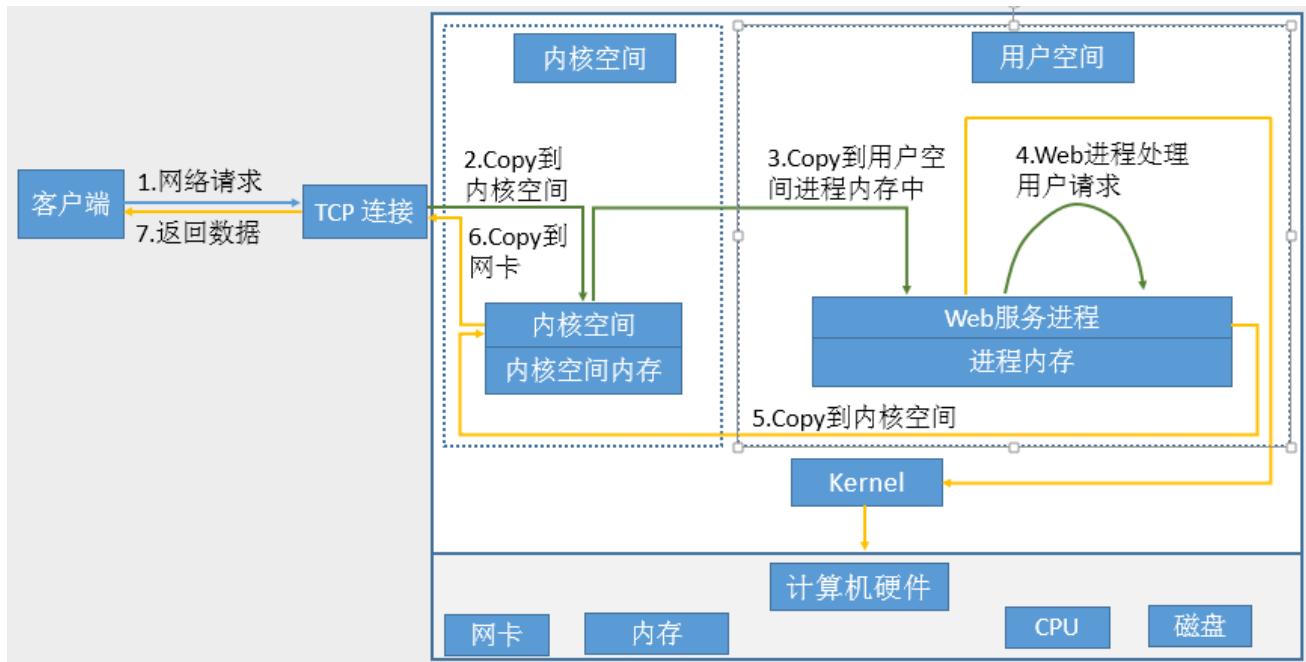
```
1 httpd MPM ( Multi-Processing Module , 多进程处理模块 ) 模式 :
2 prefork : 进程模型 , 两级结构 , 主进程master负责生成子进程 , 每个子进程负责响应一个请求
3 worker : 线程模型 , 三级结构 , 主进程master负责生成子进程 , 每个子进程负责生成多个线程 , 每个线程响应一个请求
4 event : 线程模型 , 三级结构 , 主进程master负责生成子进程 , 每个子进程生成多个线程 , 每个线程响应一个请求 , 但是增加了一个监听线程 , 用于解决在设置了keep-alived场景下线程的空等待问题。
5
6 Nginx ( Master+Worker ) 模式 :
7 主进程
8 工作进程 #直接处理客户的请求
9
10 线程验证方式 :
11 # cat /proc/PID/status
12 # pstree -p PID
```

### 1.2.4.3 : 服务端I/O :

I/O在计算机中指Input/Output，IOPS (Input/Output Per Second)即每秒的输入输出量(或读写次数)，是衡量磁盘性能的主要指标之一。IOPS是指单位时间内系统能处理的I/O请求数量，一般以每秒处理的I/O请求数量为单位，I/O请求通常为读或写数据操作请求。

一次完整的I/O是用户空间的进程数据与内核空间的内核数据的报文的完整交换，但是由于内核空间与用户空间是严格隔离的，所以其数据交换过程中不能由用户空间的进程直接调用内核空间的内存数据，而是需要经历一次从内核空间中的内存数据copy到用户空间的进程内存当中，所以说I/O就是把数据从内核空间中的内存数据复制到用户空间中进程的内存当中。

而网络通信就是网络协议栈到用户空间进程的IO就是网络IO



磁盘I/O是进程向内核发起系统调用，请求磁盘上的某个资源比如是文件或者是图片，然后内核通过相应的驱动程序将目标图片加载到内核的内存空间，加载完成之后把数据从内核内存再复制给进程内存，如果是比较大的数据也需要等待时间。

- 1 每次IO，都要经由两个阶段：
  - 2 第一步：将数据从文件先加载至内核内存空间（缓冲区），等待数据准备完成，时间较长
  - 3 第二步：将数据从内核缓冲区复制到用户空间的进程的内存中，时间较短

## 1.3：系统I/O模型：

同步/异步：关注的是消息通信机制，即在等待一件事情的处理结果时，被调用者是否提供完成通知。同步：synchronous，调用者等待被调用者返回消息后才能继续执行，如果被调用者不提供消息返回则为同步，同步需要调用者主动询问事情是否处理完成。异步：asynchronous，被调用者通过状态、通知或回调机制主动通知调用者被调用者的运行状态

- 1 同步：进程发出请求调用后，等内核返回响应以后才继续下一个请求，即如果内核一直不返回数据，那么进程就一直等。
- 2 异步：进程发出请求调用后，不等内核返回响应，接着处理下一个请求，Nginx是异步的。

阻塞/非阻塞：关注调用者在等待结果返回之前所处的状态 阻塞：blocking，指IO操作需要彻底完成后才返回到用户空间，调用结果返回之前，调用者被挂起，干不了别的事情。非阻塞：nonblocking，指IO操作被调用后立即返回给用户一个状态值，无需等到IO操作彻底完成，最终的调用结果返回之前，调用者不会被挂起，可以去做别的事情。

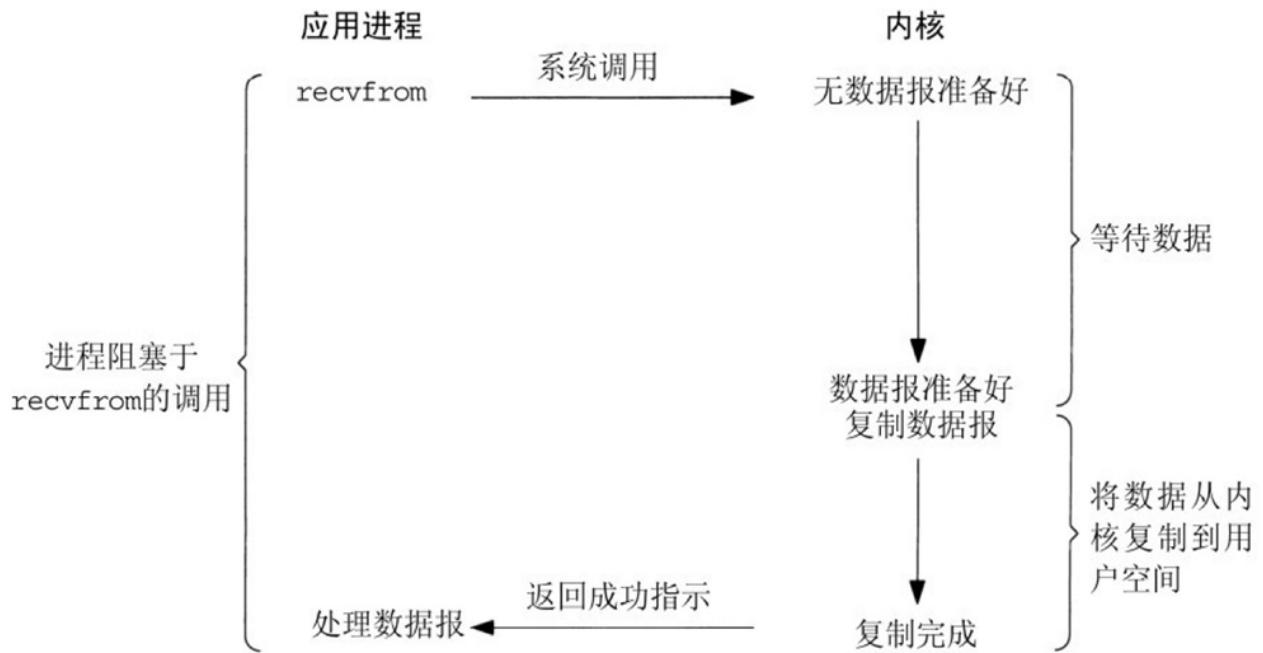
## 1.4：网络I/O模型：

阻塞型、非阻塞型、复用型、信号驱动型、异步

### 1.4.1：同步阻塞型IO模型（blocking IO）：

阻塞IO模型是最简单的IO模型，用户线程在内核进行IO操作时被阻塞 用户线程通过系统调用read发起IO读操作，由用户空间转到内核空间。内核等到数据包到达后，然后将接收的数据拷贝到用户空间，完成read操作 用户需要等待read将数据读取到buffer后，才继续处理接收的数据。整个IO请求的过程中，用户线程是被阻塞的，这导致用户在发起IO请求时，不能做任何事情，对CPU的资源利用率不够 优点：程序简单，在阻塞等待数据期间进程/线程挂起，基本不会占用 CPU 资源 缺点：每个连接需要独立的进程/线程单独处理，当并发请求量大时为了维护程序，内存、线程切换开销较大，apache 的prefork使用的是这种模式。

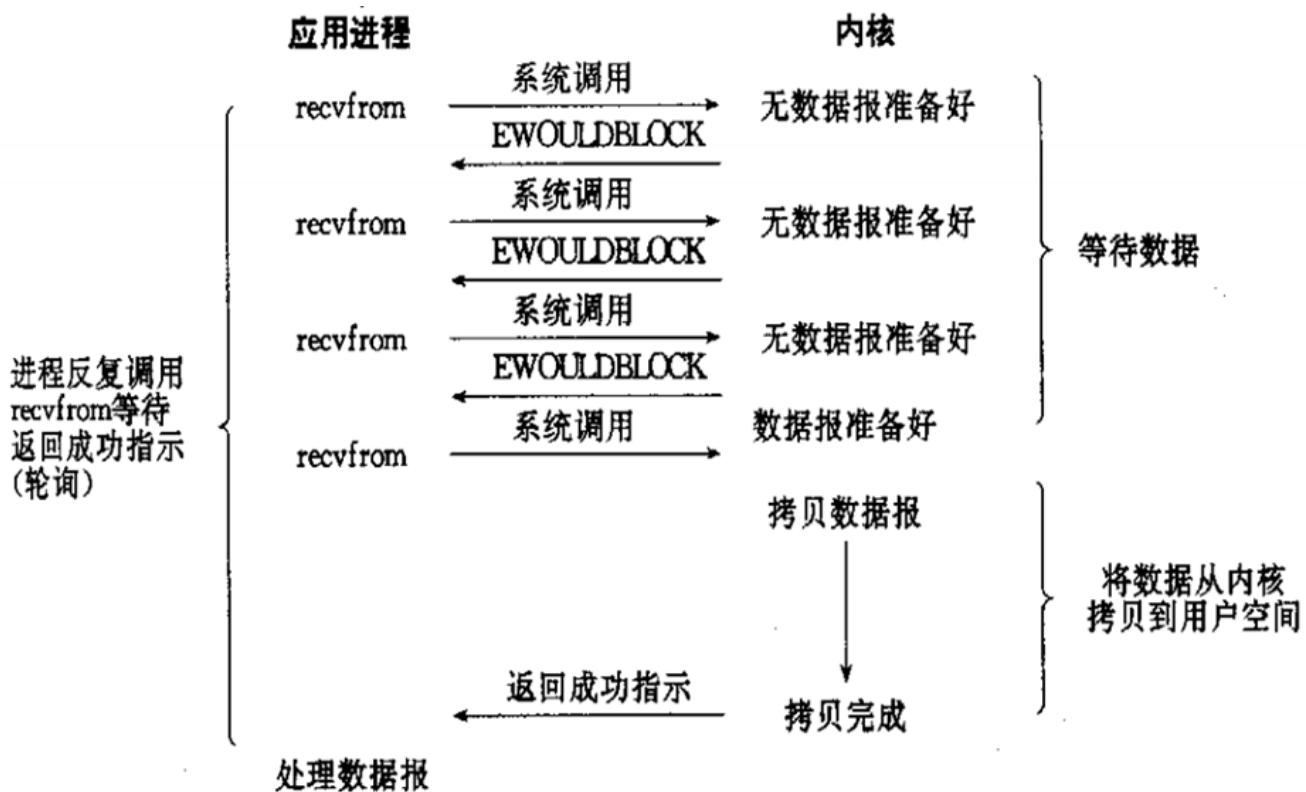
- 1 同步阻塞：程序向内核发送IO请求后一直等待内核响应，如果内核处理请求的IO操作不能立即返回，则进程将一直等待并不再接受新的请求，并由进程轮训查看IO是否完成，完成后进程将IO结果返回给client，在IO没有返回期间进程不能接受其他客户的请求，而且是有进程自己去查看IO是否完成，这种方式简单，但是比较慢，用的比较少。



### 1.4.2 : 同步非阻塞型I/O模型(nonblocking IO) :

用户线程发起IO请求时立即返回。但并未读取到任何数据，用户线程需要不断地发起IO请求，直到数据到达后，才真正读取到数据，继续执行。即“轮询”机制存在两个问题：如果有大量文件描述符都要等，那么就得一个一个的read。这会带来大量的Context Switch ( read是系统调用，每调用一次就得在用户态和核心态切换一次)。轮询的时间不好把握。这里是要猜多久之后数据才能到。等待时间设的太长，程序响应延迟就过大；设的太短，就会造成过于频繁的重试，干耗CPU而已，是比较浪费CPU的方式，一般很少直接使用这种模型，而是在其他IO模型中使用非阻塞IO这一特性。

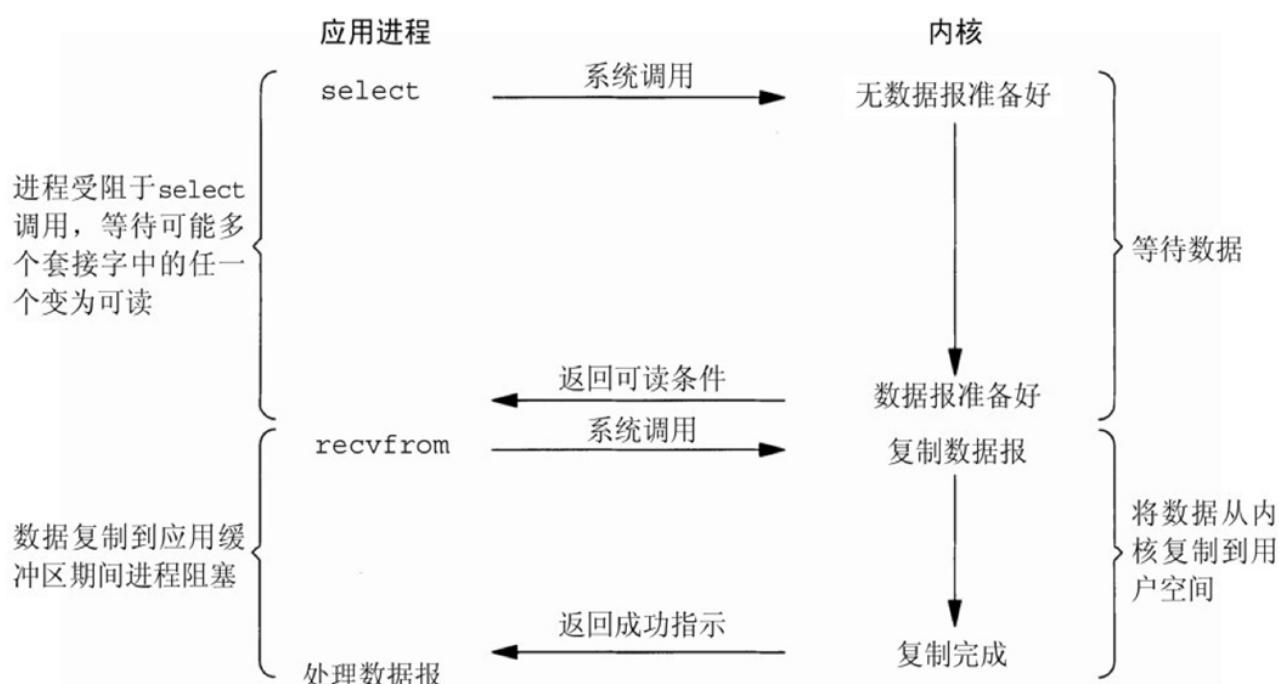
1 | 同步非阻塞：程序向内核发送请IO求后一直等待内核响应，如果内核处理请求的IO操作不能立即返回IO结果，进程将不再等待，而且继续处理其他请求，但是仍然需要进程隔一段时间就要查看内核IO是否完成。



### 1.4.3 : IO多路复用型(IO multiplexing) :

IO multiplexing就是我们说的select , poll , epoll , 有些地方也称这种IO方式为event driven IO。 select epoll的好处就在于单个process就可以同时处理多个网络连接的IO。它的基本原理就是select , poll , epoll这个function会不断的轮询所负责的所有socket , 当某个socket有数据到达了 , 就通知用户进程。当用户进程调用了select , 那么整个进程会被block , 而同时 , kernel会“监视”所有select负责的socket , 当任何一个socket中的数据准备好了 , select就会返回。这个时候用户进程再调用read操作 , 将数据从kernel拷贝到用户进程。

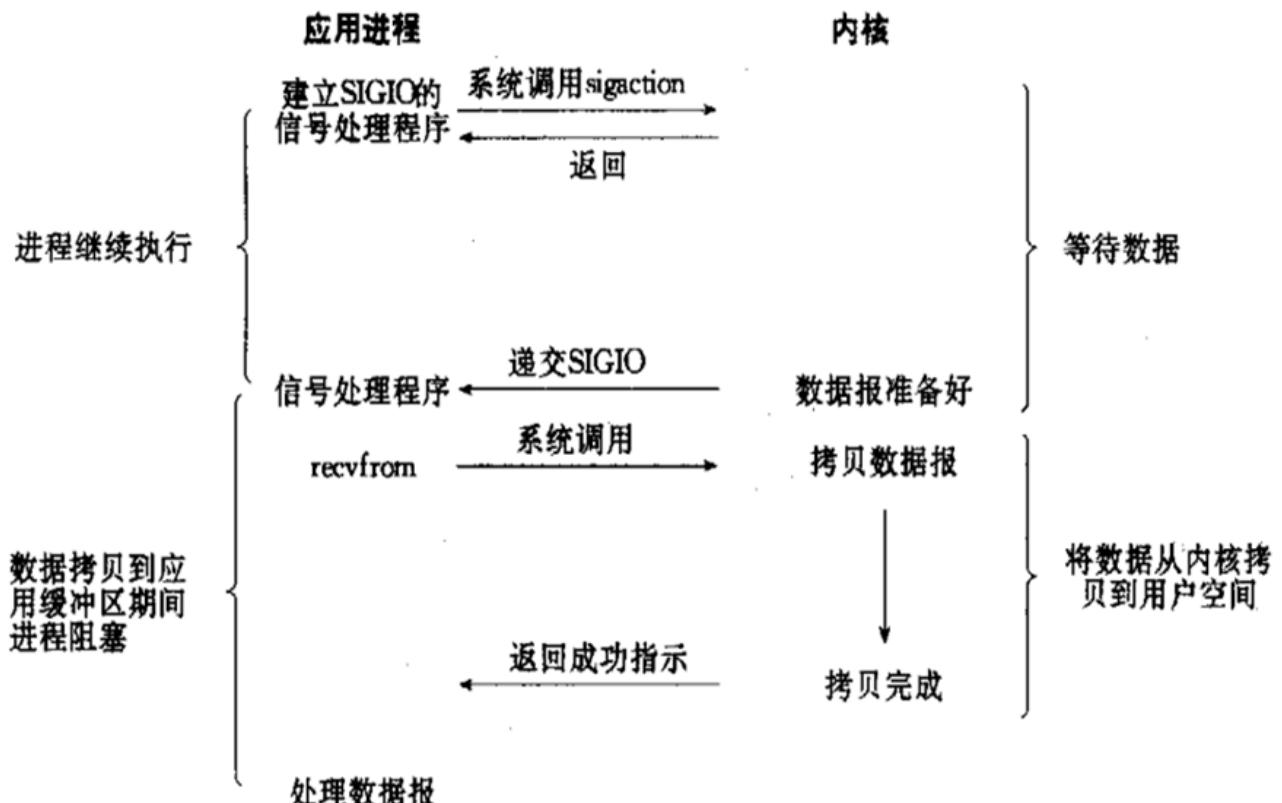
1 | Apache prefork是此模式的select , work是poll模式。



#### 1.4.4 : 信号驱动式IO(signal-driven IO):

信号驱动IO : signal-driven I/O 用户进程可以通过sigaction系统调用注册一个信号处理程序，然后主程序可以继续向下执行，当有IO操作准备就绪时，由内核通知触发一个SIGIO信号处理程序执行，然后将用户进程所需要的数据从内核空间拷贝到用户空间 此模型的优势在于等待数据报到达期间进程不被阻塞。用户主程序可以继续执行，只要等待来自信号处理函数的通知。优点：线程并没有在等待数据时被阻塞，内核直接返回调用接收信号，不影响进程继续处理其他请求因此可以提高资源的利用率 缺点：信号 I/O 在大量 IO 操作时可能会因为信号队列溢出导致没法通知

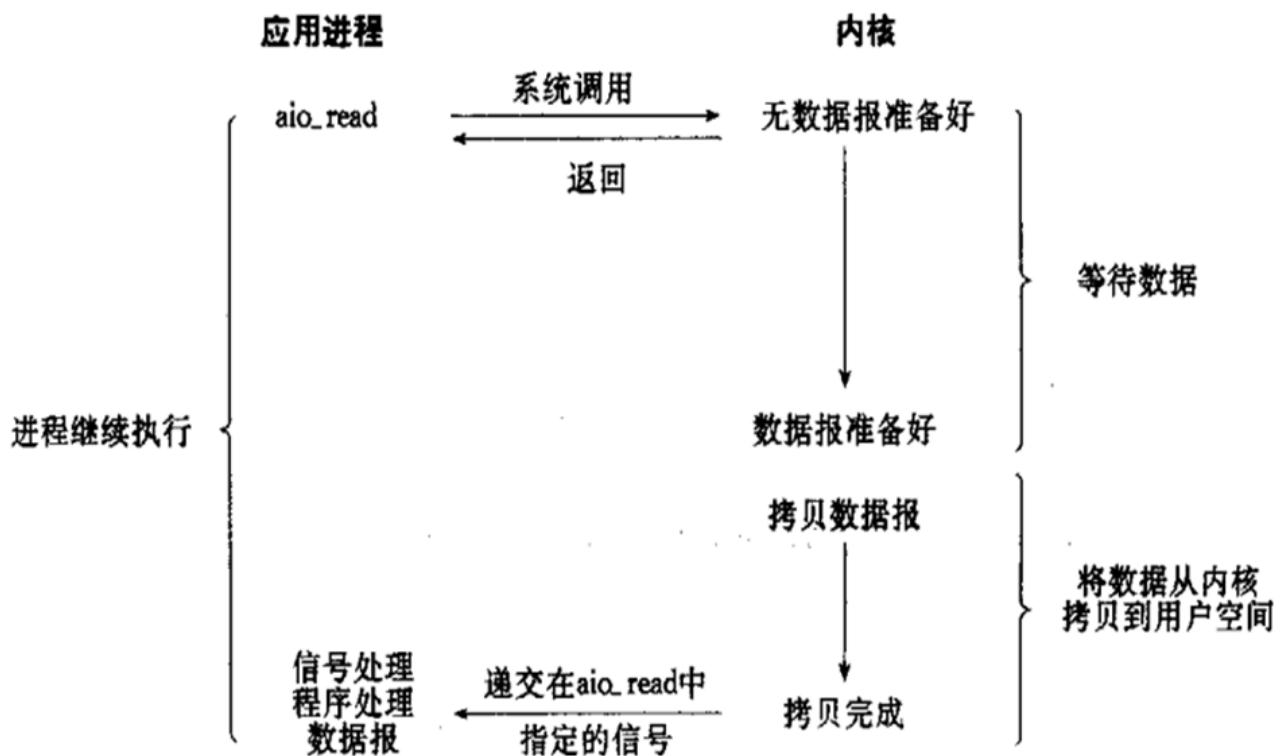
- 1 异步阻塞：程序进程向内核发送IO调用后，不用等待内核响应，可以继续接受其他请求，内核收到进程请求后进行的IO如果不能立即返回，就由内核等待结果，直到IO完成后内核再通知进程，apache event是此模式。



#### 1.4.5 : 异步(非阻塞) IO(asynchronous IO) :

相对于同步IO，异步IO不是顺序执行。用户进程进行aio\_read系统调用之后，无论内核数据是否准备好，都会直接返回给用户进程，然后用户态进程可以去做别的事情。等到socket数据准备好了，内核直接复制数据给进程，然后从内核向进程发送通知。IO两个阶段，进程都是非阻塞的。Linux提供了AIO库函数实现异步，但是用的很少。目前有很多开源的异步IO库，例如libevent、libev、libuv。异步过程如下图所示：

- 1 异步非阻塞：程序进程向内核发送IO调用后，不用等待内核响应，可以继续接受其他请求，内核调用的IO如果不能立即返回，内核会继续处理其他事物，直到IO完成后将结果通知给内核，内核在将IO完成的结果返回给进程，期间进程可以接受新的请求，内核也可以处理新的事物，因此相互不影响，可以实现较大的同时并实现较高的IO复用，因此异步非阻塞使用最多的一种通信方式。



#### 1.4.6：IO对比：

这五种 I/O 模型中，越往后，阻塞越少，理论上效率也是最优前四种属于同步 I/O，因为其中真正的 I/O 操作 (recvfrom) 将阻塞进程/线程，只有异步 I/O 模型才与 POSIX 定义的异步 I/O 相匹配。

#### 1.4.7：实现方式：

Nginx支持在多种不同的操作系统实现不同的事件驱动模型，但是其在不同的操作系统甚至是不同的系统版本上面的实现方式不尽相同，主要有以下实现方式：

```
1 1、select：  
2 select库是在linux和windows平台都基本支持的 事件驱动模型库，并且在接口的定义也基本相同，只是部分参数的含义略有差异，最大并发限制1024，是最早期的事件驱动模型。  
3 2、poll：  
4 在Linux 的基本驱动模型，windows不支持此驱动模型，是select的升级版，取消了最大的并发限制，在编译  
nginx的时候可以使用--with-poll_module和--without-poll_module这两个指定是否编译select库。  
5 3、epoll：  
6 epoll是库是Nginx服务器支持的最高性能的事件驱动库之一，是公认的非常优秀的事件驱动模型，它和select  
和poll有很大的区别，epoll是poll的升级版，但是与poll的效率有很大的区别。  
7 epoll的处理方式是创建一个待处理的事件列表，然后把这个列表发给内核，返回的时候在去轮训检查这个表，以  
判断事件是否发生，epoll支持一个进程打开的最大事件描述符的上限是系统可以打开的文件的最大数，同时  
epoll库的IO效率不随描述符数目增加而线性下降，因为它只会对内核上报的“活跃”的描述符进行操作。  
8 4、rtsig：  
9 不是一个常用事件驱动，最大队列1024，不是很常用  
10 5、kqueue：  
11 用于支持BSD系列平台的高校事件驱动模型，主要用在FreeBSD 4.1及以上版本、OpenBSD 2.0级以上版本，  
NetBSD级以上版本及Mac OS X 平台上，该模型也是poll库的变种，因此和epoll没有本质上的区别，都是通过  
避免轮训操作提供效率。  
12 6、/dev/poll：  
13 用于支持unix衍生平台的高效事件驱动模型，主要在solaris 平台、HP/UX，该模型是sun公司在开发Solaris  
系列平台的时候提出的用于完成事件驱动机制的方案，它使用了虚拟的/dev/poll设备，开发人员将要识别的文件  
描述符加入这个设备，然后通过ioctl()调用来获取事件通知，因此运行在以上系列平台的时候请使  
用/dev/poll事件驱动机制。  
14 7、eventport：  
15 该方案也是sun公司在开发solaris的时候提出的事件驱动库，只是solaris 10以上的版本，该驱动库看防止内  
核崩溃等情况的发生。  
16 8、Iocp：  
17 windows系统上的实现方式，对应第5种（异步I/O）模型。
```

#### 1.4.8：常用模型汇总：

\	select	poll	epoll
操作方式	遍历	遍历	回调
底层实现	数组	链表	哈希表
IO效率	每次调用都进行线性遍历，时间复杂度为O(n)	每次调用都进行线性遍历，时间复杂度为O(n)	事件通知方式，每当fd就绪，系统注册的回调函数就会被调用，将就绪fd放到rdlist里面。时间复杂度O(1)
最大连接数	1024 ( x86 ) 或 2048 ( x64 )	无上限	无上限
fd拷贝	每次调用select，都需要把fd集合从用户态拷贝到内核态	每次调用poll，都需要把fd集合从用户态拷贝到内核态	调用epoll_ctl时拷贝进内核并保存，之后每次epoll_wait不拷贝

#### 1.4.9：常用模型对比：

水平触发--单次通知

边缘触发--多次通知



```
1 | Select :  
2 | POSIX所规定，目前几乎在所有的平台上支持，其良好跨平台支持也是它的一个优点，本质上是通过设置或者检查存  
放fd标志位的数据结构来进行下一步处理  
3 | 缺点  
4 | 单个进程能够监视的文件描述符的数量存在最大限制，在Linux上一般为1024，可以通过修改宏定义  
FD_SETSIZE，再重新编译内核实现，但是这样也会造成效率的降低  
5 | 单个进程可监视的fd数量被限制，默认是1024，修改此值需要重新编译内核  
6 | 对socket是线性扫描，即采用轮询的方法，效率较低  
7 | select 采取了内存拷贝方法来实现内核将 FD 消息通知给用户空间，这样一个用来存放大量fd的数据结构，这样  
会使得用户空间和内核空间在传递该结构时复制开销大
```

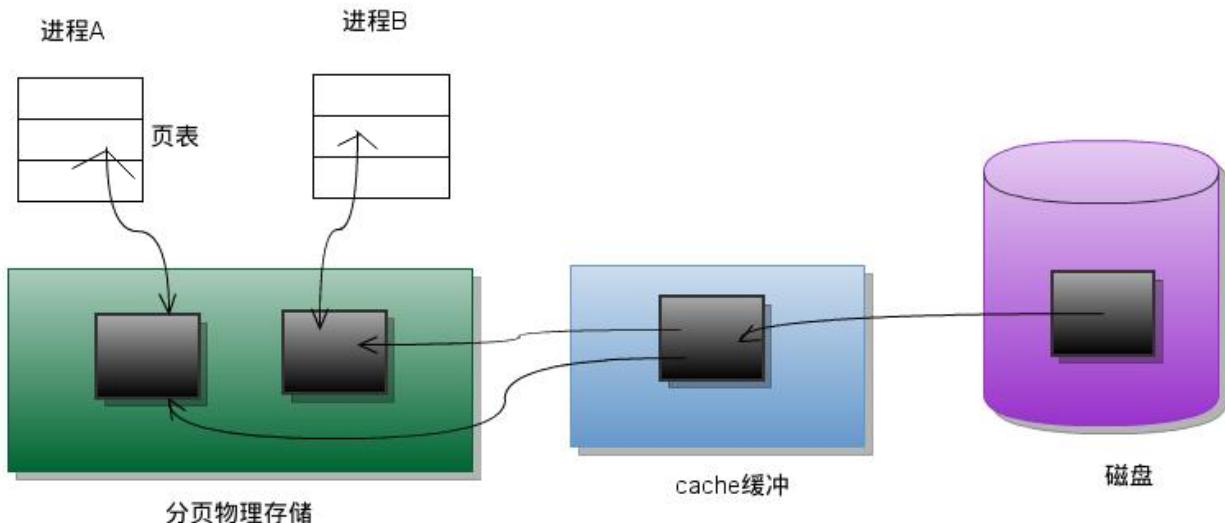
```
1 | poll :  
2 | 本质上和select没有区别，它将用户传入的数组拷贝到内核空间，然后查询每个fd对应的设备状态  
3 | 其没有最大连接数的限制，原因是它是基于链表来存储的  
4 | 大量的fd的数组被整体复制于用户态和内核地址空间之间，而不管这样的复制是不是有意义  
5 | poll特点是“水平触发”，如果报告了fd后，没有被处理，那么下次poll时会再次报告该fd  
6 | select是边缘触发即只通知一次
```

```
1 epoll:  
2 在Linux 2.6内核中提出的select和poll的增强版本  
3 支持水平触发LT和边缘触发ET，最大的特点在于边缘触发，它只告诉进程哪些fd刚刚变为就绪态，并且只会通知一次  
4 使用“事件”的就绪通知方式，通过epoll_ctl注册fd，一旦该fd就绪，内核就会采用类似callback的回调机制来激活该fd，epoll_wait便可以收到通知  
5 优点：  
6 没有最大并发连接的限制：能打开的FD的上限远大于1024(1G的内存能监听约10万个端口)，具体查看/proc/sys/fs/file-max，此值和系统内存大小相关  
7 效率提升：非轮询的方式，不会随着FD数目的增加而效率下降；只有活跃可用的FD才会调用callback函数，即epoll最大的优点就在于它只管理“活跃”的连接，而跟连接总数无关  
8 内存拷贝，利用mmap(Memory Mapping)加速与内核空间的消息传递；即epoll使用mmap减少复制开销
```

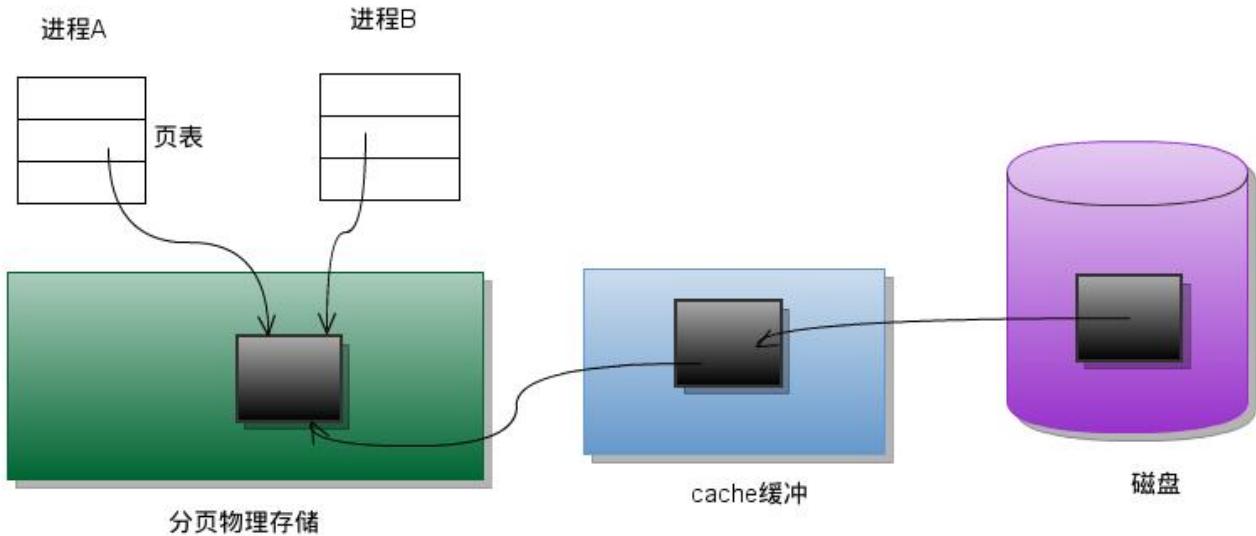
## 1.4.10 : MMAP介绍：

mmap()系统调用使得进程之间通过映射同一个普通文件实现共享内存。普通文件被映射到进程地址空间后，进程可以向访问普通内存一样对文件进行访问。

### 1.4.10.1 传统方式copy数据：



### 1.4.10.2 : mmap方式：



## 二：Nginx基础：

Nginx : engine X , 2002年 , 开源 , 商业版 Nginx是免费的、开源的、高性能的HTTP和反向代理服务器、邮件代理服务器、以及TCP/UDP代理服务器 解决C10K问题 ( 10K Connections ) , <http://www.ideawu.net/blog/archives/740.html> Nginx官网 : <http://nginx.org> nginx的其它的二次发行版 : Tengine : 由淘宝网发起的Web服务器项目。它在Nginx的基础上 , 针对大访问量网站的需求 , 添加了很多高级功能和特性。Tengine的性能和稳定性已经在大型的网站如淘宝网 , 天猫商城等得到了很好的检验。它的最终目标是打造一个高效、稳定、安全、易用的Web平台。从2011年12月开始 , Tengine成为一个开源项目 , 官网 <http://tengine.taobao.org/> OpenResty : 基于 Nginx 与 Lua 语言的高性能 Web 平台 , 章亦春团队开发 , 官网 : <http://openresty.org/cn/>

### 2.1 : Nginx功能介绍 :

静态的web资源服务器html , 图片 , js , css , txt等静态资源 结合FastCGI/uWSGI/SCGI等协议反向代理动态资源请求 http/https协议的反向代理 imap4/pop3协议的反向代理 tcp/udp协议的请求转发 ( 反向代理 )

#### 2.1.1 : 基础特性 :

- ```

1 | 特性 :
2 | 模块化设计 , 较好的扩展性
3 | 高可靠性
4 | 支持热部署 : 不停机更新配置文件 , 升级版本 , 更换日志文件
5 | 低内存消耗 : 10000个keep-alive连接模式下的非活动连接 , 仅需2.5M内存
6 | event-driven,aio,mmap , sendfile
7 |
8 | 基本功能 :
9 | 静态资源的web服务器
10 | http协议反向代理服务器
11 | pop3/imap4协议反向代理服务器
12 | FastCGI(LNMP) , uWSGI(python)等协议
13 | 模块化 ( 非DSO ) , 如zip , SSL模块

```

#### 2.1.2 : 和web服务相关的功能 :

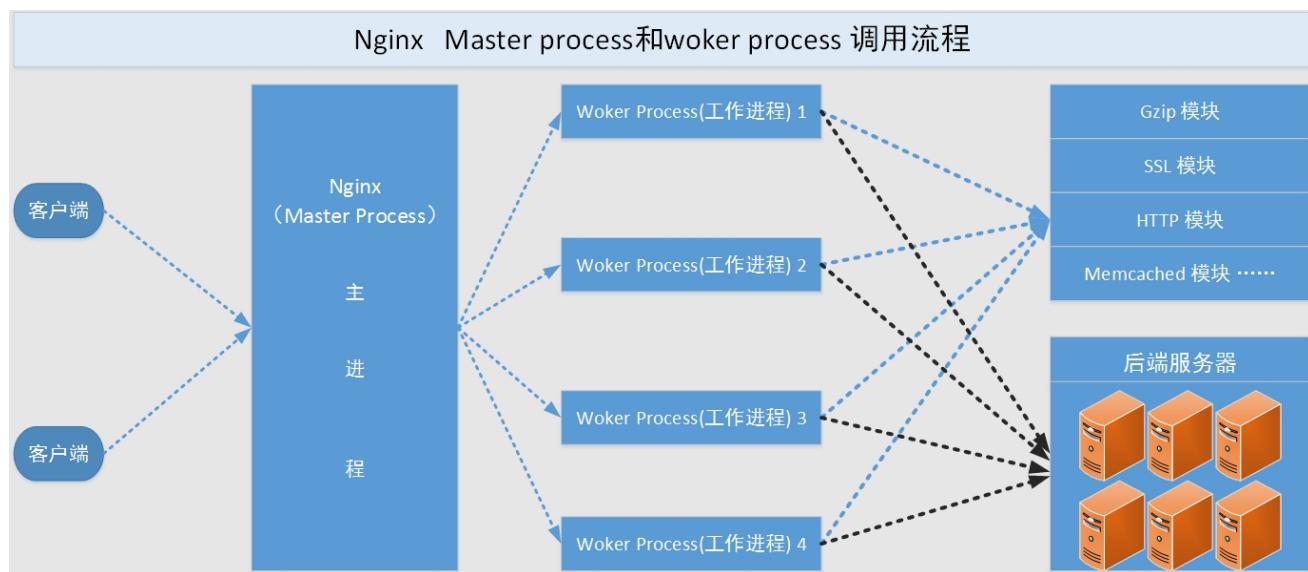
- 1 虚拟主机 ( server )
- 2 支持 keep-alive 和管道连接(利用一个连接做多次请求)
- 3 访问日志 ( 支持基于日志缓冲提高其性能 )
- 4 url rewrite
- 5 路径别名
- 6 基于IP及用户的访问控制
- 7 支持速率限制及并发数限制
- 8 重新配置和在线升级而无须中断客户的工作进程

## 2.2 : Nginx组织结构 :

web请求处理机制： 1、多进程方式：服务器每接收到一个客户端请求就有服务器的主进程生成一个子进程响应客户端，直到用户关闭连接，这样的优势是处理速度快，子进程之间相互独立，但是如果访问过大会导致服务器资源耗尽而无法提供请求。 2、多线程方式：与多进程方式类似，但是每收到一个客户端请求会有服务进程派生出一个线程来个客户方进行交互，一个线程的开销远远小于一个进程，因此多线程方式在很大程度减轻了web服务器对系统资源的要求，但是多线程也有自己的缺点，即当多个线程位于同一个进程中工作的时候，可以相互访问同样的内存地址空间，所以他们相互影响，一旦主进程挂掉则所有子线程都不能工作了，IIS服务器使用了多线程的方式，需要间隔一段时间就重启一次才能稳定。

### 2.2.1 : 组织模型 :

Nginx是多进程组织模型，而且是一个由Master主进程和Worker工作进程组成。

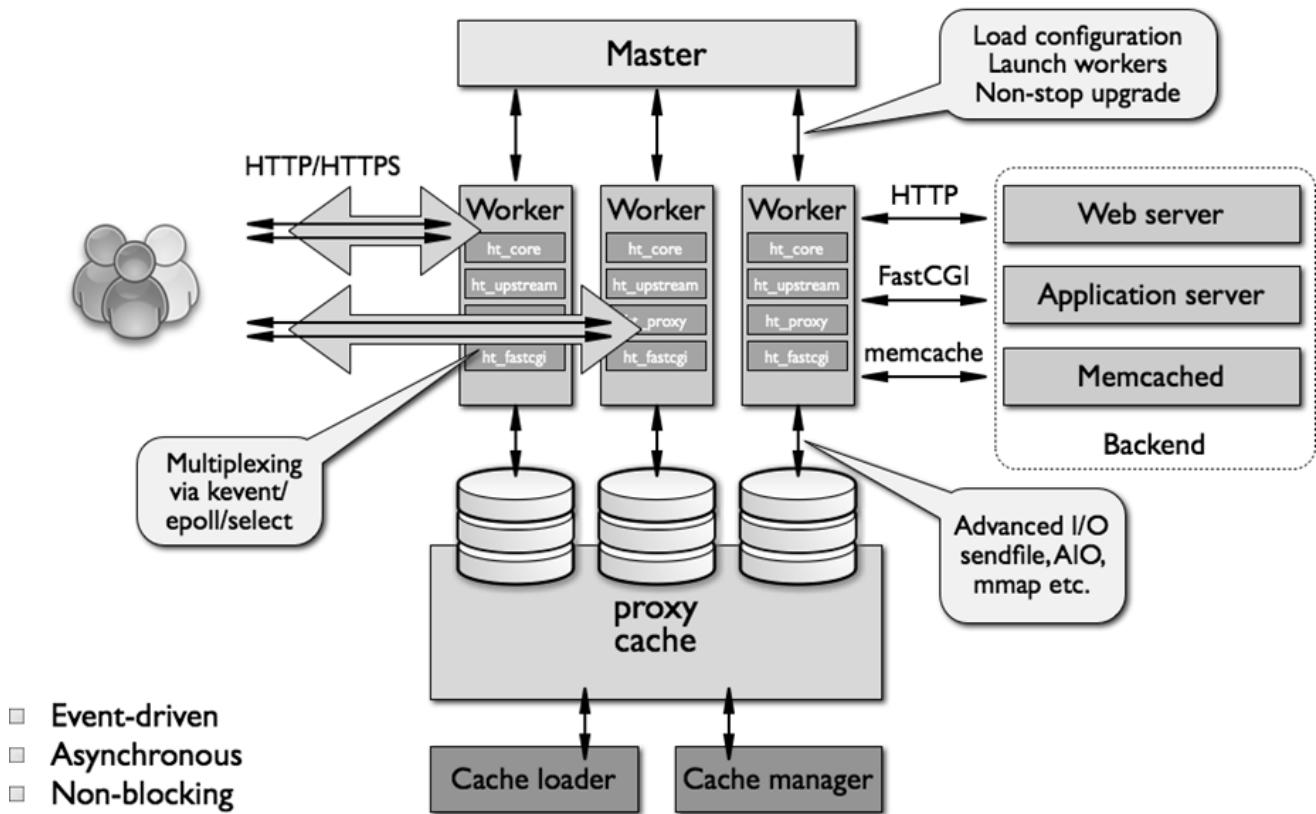


主进程(master process)的功能：

- 1 读取Nginx 配置文件并验证其有效性和正确性
- 2 建立、绑定和关闭socket连接
- 3 按照配置生成、管理和结束工作进程
- 4 接受外界指令，比如重启、升级及退出服务器等指令
- 5 不中断服务，实现平滑升级，重启服务并应用新的配置
- 6 开启日志文件，获取文件描述符
- 7 不中断服务，实现平滑升级，升级失败进行回滚处理
- 8 编译和处理perl脚本

工作进程 ( woker process ) 的功能：

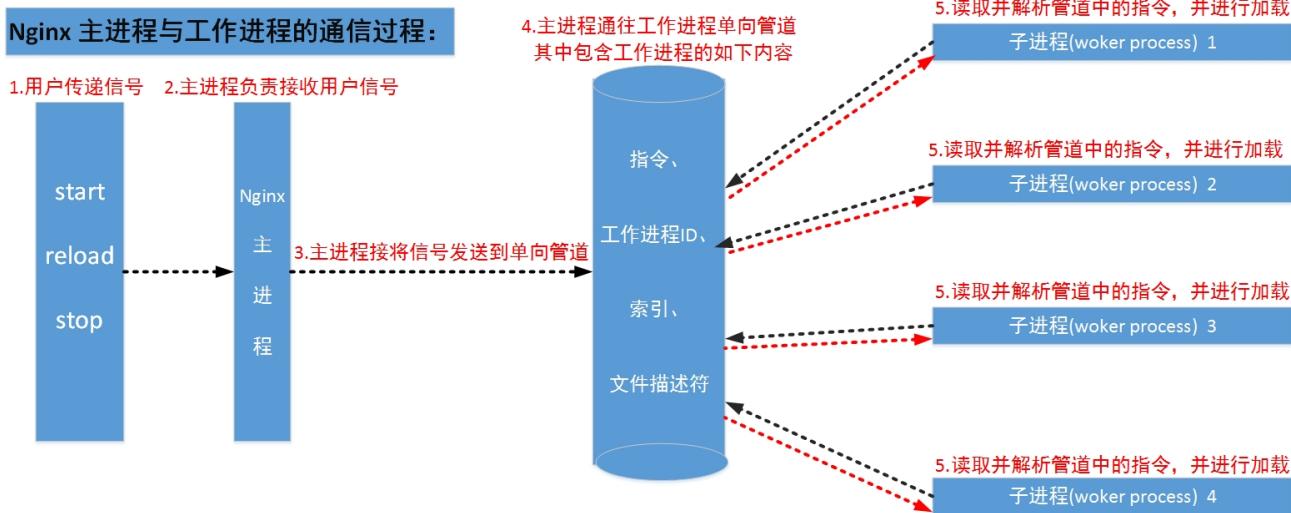
- 1 接受处理客户的请求
- 2 将请求以此送入各个功能模块进行处理
- 3 IO调用，获取响应数据
- 4 与后端服务器通信，接收后端服务器的处理结果
- 5 缓存数据，访问缓存索引，查询和调用缓存数据
- 6 发送请求结果，响应客户的请求
- 7 接收主程序指令，比如重启、升级和退出等



## 2.2.2 : 进程间通信 :

工作进程是有主进程生成的，主进程使用fork()函数，在Nginx服务器启动过程中主进程根据配置文件决定启动工作进程的数量，然后建立一张全局的工作表用于存放当前未退出的所有的工作进程，主进程生成工作进程后会将新生成的工作进程加入到工作进程表中，并建立一个单向的管道并将其传递给工作进程，该管道与普通的管道不同，它是由主进程指向工作进程的单项通道，包含了主进程想工作进程发出的指令、工作进程ID、工作进程在工作进程表中的索引和必要的文件描述符等信息。 主进程与外界通过信号机制进行通信，当接收到需要处理的信号时，它通过管道向相关的工作进程发送正确的指令，每个工作进程都有能力捕获管道中的可读事件，当管道中有可读事件的时候，工作进程就会从管道中读取并解析指令，然后采取相应的执行动作，这样就完成了主进程与工作进程的交互。

- 1 工作进程之间的通信原理基本上和主进程与工作进程之间的通信是一样的，只要工作进程之间能够取得彼此的信息，建立管道即可通信，但是由于工作进程之间是完全隔离的，因此一个进程想要知道另外一个进程的状态信息就只能通过主进程来设置了。
- 2 为了实现工作进程之间的交互，主进程在生成工作进程之后，在工作进程表中进行遍历，将该新进程的ID以及针对该进程建立的管道句柄传递给工作进程中的其他进程，为工作进程之间的通信做准备，当工作进程1向工作进程2发送指令的时候，首先在主进程给它的其他工作进程工作信息中找到2的进程ID，然后将正确的指令写入指向进程2的管道，工作进程2捕获到管道中的事件后，解析指令并进行相关操作，这样就完成了工作进程之间的通信。



## 2.3 : Nginx模块介绍：

核心模块：是 Nginx 服务器正常运行必不可少的模块，提供 错误日志记录、 配置文件解析、 事件驱动机制、 进程管理 等核心功能 标准HTTP模块：提供 HTTP 协议解析相关的功能，比如：端口配置、 网页编码设置、 HTTP响应头设置 等等 可选HTTP模块：主要用于扩展标准的 HTTP 功能，让 Nginx 能处理一些特殊的服务，比如：Flash 多媒体传输、 解析 Geoloc 请求数、 网络传输压缩、 安全协议 SSL 支持等 邮件服务模块：主要用于支持 Nginx 的 邮件服务，包括对 POP3 协议、 IMAP 协议和 SMTP 协议的支持 第三方模块：是为了扩展 Nginx 服务器应用，完成开发者自定义功能，比如：Json 支持、 Lua 支持等

nginx高度模块化，但其模块早期不支持DSO机制；1.9.11版本支持动态装载和卸载 模块分类：

```

1 核心模块：core module
2 标准模块：
3 HTTP 模块： ngx_http_*
4     HTTP Core modules    默认功能
5     HTTP Optional modules 需编译时指定
6 Mail 模块    ngx_mail_*
7 stream 模块  ngx_stream_*
8 第三方模块

```



Nginx 模块图

## 2.4 : Nginx安装：

Nginx的安装版本分为开发版、稳定版和过期版，Nginx安装可以使用yum或源码安装，但是推荐使用源码，一是yum的版本比较旧，二是编译安装可以更方便自定义相关路径，三是使用源码编译可以自定义相关功能，更方便业务的上的使用，源码安装需要提前准备标准的编译器，GCC的全称是（GNU Compiler collection），其有GNU开发，并以GPL即LGPL许可，是自由的类UNIX即苹果电脑Mac OS X操作系统的标准编译器，因为GCC原本只能处理C语言，所以原名为GNU C语言编译器，后来得到快速发展，可以处理C++,Fortran , pascal , objective-C , java以及Ada等其他语言，此外还需要Automake工具，以完成自动创建Makefile的工作，Nginx的一些模块需要依赖第三方库，比如pcre（支持rewrite）， zlib（支持gzip模块）和openssl（支持ssl模块）等。

### 2.4.1 : Nginx yum安装：

需要提前配置好epel源

```

1 [root@s1 ~]# yum install -y nginx
2 [root@s1 ~]# rpm -q1 nginx
3 /etc/logrotate.d/nginx
4 /etc/nginx/fastcgi.conf
5 /etc/nginx/fastcgi.conf.default
6 /etc/nginx/fastcgi_params
7 /etc/nginx/fastcgi_params.default
8 /etc/nginx/koi-utf
9 /etc/nginx/koi-win
10 /etc/nginx/mime.types

```

```
11 /etc/nginx/mime.types.default
12 /etc/nginx/nginx.conf
13 /etc/nginx/nginx.conf.default
14 /etc/nginx/scgi_params
15 /etc/nginx/scgi_params.default
16 /etc/nginx/uwsgi_params
17 /etc/nginx/uwsgi_params.default
18 /etc/nginx/win-utf
19 /usr/bin/nginx-upgrade
20 /usr/lib/systemd/system/nginx.service
21 /usr/lib64/nginx/modules
22 /usr/sbin/nginx
23 /usr/share/doc/nginx-1.12.2
24 .....
25 /var/lib/nginx
26 /var/lib/nginx/tmp
27 /var/log/nginx
28
29 [root@s1 ~]# which nginx
30 /usr/sbin/nginx
```

#### 2.4.1.1 : 检查安装 :

查看nginx安装包信息

```
1 [root@s1 ~]# rpm -qi nginx
2 Name        : nginx
3 Epoch       : 1
4 Version     : 1.12.2
5 Release    : 2.el7
6 Architecture: x86_64
7 Install Date: Fri 15 Feb 2019 05:01:32 PM CST
8 Group       : System Environment/Daemons
9 Size        : 1574949
10 License     : BSD
11 Signature   : RSA/SHA256, Tue 06 Mar 2018 05:44:06 PM CST, Key ID 6a2faea2352c64e5
12 Source RPM  : nginx-1.12.2-2.el7.src.rpm
13 Build Date  : Tue 06 Mar 2018 05:27:44 PM CST
14 Build Host  : buildhw-02.phx2.fedoraproject.org
15 Relocations : (not relocatable)
16 Packager    : Fedora Project
17 Vendor      : Fedora Project
18 URL         : http://nginx.org/
19 Bug URL     : https://bugz.fedoraproject.org/nginx
20 Summary     : A high performance web server and reverse proxy server
21 Description :
22 Nginx is a web server and a reverse proxy server for HTTP, SMTP, POP3 and
23 IMAP protocols, with a strong focus on high concurrency, performance and low
24 memory usage.
```

#### 2.4.1.2 : 查看帮助 :

使用安装完成的二进制文件nginx

```
1 [root@s1 ~]# nginx -h
2 nginx version: nginx/1.12.2
3 Usage: nginx [-?hvvtTq] [-s signal] [-c filename] [-p prefix] [-g directives]
4
5 Options:
6 -?, -h : this help
7 -v : show version and exit
8 -V : show version and configure options then exit #显示版本和编译参数
9 -t : test configuration and exit #测试配置文件是否异常
10 -T : test configuration, dump it and exit #测试并打印
11 -q : suppress non-error messages during configuration testing #静默模式
12 -s signal : send signal to a master process: stop, quit, reopen, reload #发送信号
13 -p prefix : set prefix path (default: /usr/share/nginx/) #指定Nginx 目录
14 -c filename : set configuration file (default: /etc/nginx/nginx.conf) #配置文件路径
15 -g directives : set global directives out of configuration file #设置全局指令
```

#### 2.4.1.3 : 验证Nginx :

```
1 [root@s1 ~]# nginx -t
2 nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
3 nginx: configuration file /etc/nginx/nginx.conf test is successful
4 [root@s1 ~]# nginx -V
5 nginx version: nginx/1.12.2
6 built by gcc 4.8.5 20150623 (Red Hat 4.8.5-16) (GCC)
7 built with OpenSSL 1.0.2k-fips 26 Jan 2017
8 TLS SNI support enabled
9 configure arguments: --prefix=/usr/share/nginx --sbin-path=/usr/sbin/nginx --modules-
path=/usr/lib64/nginx/modules --conf-path=/etc/nginx/nginx.conf --error-log-
path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --http-
client-body-temp-path=/var/lib/nginx/tmp/client_body --http-proxy-temp-
path=/var/lib/nginx/tmp/proxy --http-fastcgi-temp-path=/var/lib/nginx/tmp/fastcgi --
http uwsgi-temp-path=/var/lib/nginx/tmp/uwsgi --http-scgi-temp-
path=/var/lib/nginx/tmp/scgi --pid-path=/run/nginx.pid --lock-
path=/run/lock/subsys/nginx --user=nginx --group=nginx --with-file-aio --with-ipv6 --
with-http_auth_request_module --with-http_ssl_module --with-http_v2_module --with-
http_realip_module --with-http_addition_module --with-http_xslt_module=dynamic --
with-http_image_filter_module=dynamic --with-http_geoip_module=dynamic --with-
http_sub_module --with-http_dav_module --with-http_fltv_module --with-http_mp4_module
--with-http_gunzip_module --with-http_gzip_static_module --with-
http_random_index_module --with-http_secure_link_module --with-
http_degradation_module --with-http_slice_module --with-http_stub_status_module --
with-http_perl_module=dynamic --with-mail=dynamic --with-mail_ssl_module --with-pcre
--with-pcre-jit --with-stream=dynamic --with-stream_ssl_module --with-
google_perftools_module --with-debug --with-cc-opt='-O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-protector-strong --param=ssp-buffer-size=4 -frecord-gcc-switches -specs=/usr/lib/rpm/redhat/redhat-hardened-cc1 -m64 -mtune=generic' --with-ld-opt=' -Wl,-z,relro -specs=/usr/lib/rpm/redhat/redhat-
hardened-ld -Wl,-E'
```

#### 2.4.1.4 : Nginx启动脚本 :

```

1 [root@s1 ~]# cat /usr/lib/systemd/system/nginx.service
2 [Unit]
3 Description=The nginx HTTP and reverse proxy server
4 After=network.target remote-fs.target nss-lookup.target
5
6 [Service]
7 Type=forking
8 PIDFile=/run/nginx.pid
9 # Nginx will fail to start if /run/nginx.pid already exists but has the wrong
10 # SELinux context. This might happen when running `nginx -t` from the cmdline.
11 # https://bugzilla.redhat.com/show_bug.cgi?id=1268621
12 ExecStartPre=/usr/bin/rm -f /run/nginx.pid
13 ExecStartPre=/usr/sbin/nginx -t
14 ExecStart=/usr/sbin/nginx
15 ExecReload=/bin/kill -s HUP $MAINPID
16 KillSignal=SIGQUIT
17 TimeoutStopSec=5
18 KillMode=process
19 PrivateTmp=true
20
21 [Install]
22 WantedBy=multi-user.target

```

#### 2.4.1.5 : 配置Nginx :

默认配置文件 : /etc/nginx/nginx.conf , , 默认配置如下 :

```

1 [root@s1 ~]# grep -v "#" /etc/nginx/nginx.conf | grep -v "^\$"
2 user nginx;
3 worker_processes auto;
4 error_log /var/log/nginx/error.log;
5 pid /run/nginx.pid;
6 include /usr/share/nginx/modules/*.conf;
7 events {
8     worker_connections 1024;
9 }
10 http {
11     log_format main '$remote_addr - $remote_user [$time_local] "$request" '
12                     '$status $body_bytes_sent "$http_referer" '
13                     '"$http_user_agent" "$http_x_forwarded_for"';
14     access_log /var/log/nginx/access.log main;
15     sendfile          on;
16     tcp_nopush        on;
17     tcp_nodelay       on;
18     keepalive_timeout 65;
19     types_hash_max_size 2048;
20     include           /etc/nginx/mime.types;
21     default_type      application/octet-stream;
22     include /etc/nginx/conf.d/*.conf;
23     server {
24         listen          80 default_server;
25         listen          [::]:80 default_server;

```

```

26     server_name _;
27     root          /usr/share/nginx/html;
28     include /etc/nginx/default.d/*.conf;
29     location / {
30     }
31     error_page 404 /404.html;
32         location = /40x.html {
33     }
34     error_page 500 502 503 504 /50x.html;
35         location = /50x.html {
36     }
37 }
38 }
```

#### 2.4.1.6 : 启动Nginx :

```

1 [root@s1 ~]# systemctl start nginx
2 [root@s1 ~]# systemctl status nginx
3 ● nginx.service - The nginx HTTP and reverse proxy server
4   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor preset:
disabled)
5     Active: active (running) since Fri 2019-02-15 17:43:27 CST; 2s ago
6       Process: 8648 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
7       Process: 8645 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
8       Process: 8643 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited,
status=0/SUCCESS)
9     Main PID: 8650 (nginx)
10    CGroup: /system.slice/nginx.service
11        ├─8650 nginx: master process /usr/sbin/nginx
12        ├─8651 nginx: worker process
13        └─8652 nginx: worker process
14
15 Feb 15 17:43:27 s1.example.com systemd[1]: Starting The nginx HTTP and reverse
proxy server...
16 Feb 15 17:43:27 s1.example.com nginx[8645]: nginx: the configuration file
/etc/nginx/nginx.conf syntax is ok
17 Feb 15 17:43:27 s1.example.com nginx[8645]: nginx: configuration file
/etc/nginx/nginx.conf test is successful
18 Feb 15 17:43:27 s1.example.com systemd[1]: Started The nginx HTTP and reverse proxy
server.
19
20 [root@s1 ~]# ps -ef | grep nginx
21 root      8650      1  0 17:43 ?  00:00:00 nginx: master process /usr/sbin/nginx
22 nginx     8651  8650  0 17:43 ?  00:00:00 nginx: worker process
23 nginx     8652  8650  0 17:43 ?  00:00:00 nginx: worker process
24 root      8858  2785  0 17:47 pts/0    00:00:00 grep --color=auto nginx
```

#### 2.4.1.7 : 访问Nginx :

## Welcome to nginx on Fedora!

is used to test the proper operation of the **nginx** HTTP server after it has been installed. If you can read this page, it means that the web server installed at this site is working properly.

### Website Administrator

This is the default `index.html` page that is distributed with **nginx** on Fedora. It is located in `/usr/share/nginx/html`.

You should now put your content in a location of your choice and edit the `root` configuration directive in the **nginx** configuration file `/etc/nginx/nginx.conf`.



## 2.4.2 : Nginx 编译安装：

准备编译安装的基础环境：

```

1 [root@s2 ~]# yum install -y vim lrzsz tree screen psmisc lsof tcpdump wget ntpdate
  gcc gcc-c++ glibc glibc-devel pcre pcre-devel openssl openssl-devel systemd-devel
  net-tools iotop bc zip unzip zlib-devel bash-completion nfs-utils automake libxml2
  libxml2-devel libxml2 libxml2-devel perl perl-ExtUtils-Embed

2
3 gcc为GNU Compiler Collection的缩写，可以编译C和C++源代码等，它是GNU开发的C和C++以及其他很多种
语言的编译器（最早的时候只能编译C，后来很快进化成一个编译多种语言的集合，如Fortran、Pascal、
Objective-C、Java、Ada、Go等。）

4     gcc 在编译C++源代码的阶段，只能编译 C++ 源文件，而不能自动和 C++ 程序使用的库链接（编译过程分为
编译、链接两个阶段，注意不要和可执行文件这个概念搞混，相对可执行文件来说有三个重要的概念：编译
（compile）、链接（link）、加载（load）。源程序文件被编译成目标文件，多个目标文件连同库被链接成一个
最终的可执行文件，可执行文件被加载到内存中运行）。因此，通常使用 g++ 命令来完成 C++ 程序的编译和连
接，该程序会自动调用 gcc 实现编译。

5     gcc-c++也能编译C源代码，只不过把会把它当成C++源代码，后缀为.c的，gcc把它当作是C程序，而g++当作
是c++程序；后缀为.cpp的，两者都会认为是c++程序，注意，虽然c++是c的超集，但是两者对语法的要求是有区
别的。

6     automake是一个从Makefile.am文件自动生成Makefile.in的工具。为了生成Makefile.in，automake
还需要用到perl，由于automake创建的发布完全遵循GNU标准，所以在创建中不需要perl。libtool是一款方便生
成各种程序库的工具。

7     pcre pcre-devel：在Nginx编译需要 PCRE(Perl Compatible Regular Expression)，因为Nginx
的Rewrite模块和HTTP 核心模块会使用到PCRE正则表达式语法。

8     zlib zlib-devel：nginx启用压缩功能的时候，需要此模块的支持。

9     openssl openssl-devel：开启SSL的时候需要此模块的支持。

```

### 2.4.2.1 : 安装Nginx：

官方源码包下载地址：

<https://nginx.org/en/download.html>

```

1 [root@s2 ~]# cd /usr/local/src/
2 [root@s2 src]# wget https://nginx.org/download/nginx-1.12.2.tar.gz
3 [root@s2 src]# tar xf nginx-1.12.2.tar.gz
4 [root@s2 src]# cd nginx-1.12.2/
5
6 编译是为了检查系统环境是否符合编译安装的要求，比如是否有gcc编译工具，是否支持编译参数当中的模块，并
根据开启的参数等生成Makefile文件为下一步做准备：

```

```
7 [root@s2 nginx-1.12.2]# ./configure --prefix=/apps/nginx \
8 --user=nginx \
9 --group=nginx \
10 --with-http_ssl_module \
11 --with-http_v2_module \
12 --with-http_realip_module \
13 --with-http_stub_status_module \
14 --with-http_gzip_static_module \
15 --with-pcre \
16 --with-stream \
17 --with-stream_ssl_module \
18 --with-stream_realip_module
19
20 [root@s2 nginx-1.12.2]# make #编译步骤，根据Makefile文件生成相应的模块
21 [root@s2 nginx-1.12.2]# make install #创建目录，并将生成的模块和文件复制到相应的目录：
22 [root@s2 nginx-1.12.2]# useradd nginx -s /sbin/nologin -u 2000
23 [root@s2 nginx-1.12.2]# chown nginx.nginx -R /apps/nginx/
```

备注：nginx完成安装以后，有四个主要的目录：

- 1 conf：保存nginx所有的配置文件，其中nginx.conf是nginx服务器的最核心最主要的配置文件，其他的.conf则是用来配置nginx相关的功能的，例如fastcgi功能使用的是fastcgi.conf和fastcgi\_params两个文件，配置文件一般都有个样板配置文件，是文件名.default结尾，使用的使用将其复制为并将default去掉即可。
- 2 html目录中保存了nginx服务器的web文件，但是可以更改为其他目录保存web文件，另外还有一个50x的web文件是默认的错误页面提示页面。
- 3 logs：用来保存nginx服务器的访问日志错误日志等日志，logs目录可以放在其他路径，比如/var/logs/nginx里面。
- 4 sbin：保存nginx二进制启动脚本，可以接受不同的参数以实现不同的功能。

#### 2.4.2.2：验证版本及编译参数：

```
1 [root@s2 nginx-1.12.2]# /apps/nginx/sbin/nginx -v
2 nginx version: nginx/1.12.2
3 built by gcc 4.8.5 20150623 (Red Hat 4.8.5-36) (GCC)
4 built with OpenSSL 1.0.2k-fips 26 Jan 2017
5 TLS SNI support enabled
6 configure arguments: --prefix=/apps/nginx --user=nginx --group=nginx --with-
http_ssl_module --with-http_v2_module --with-http_realip_module --with-
http_stub_status_module --with-http_gzip_static_module --with-pcre --with-stream --
with-stream_ssl_module --with-stream_realip_module
```

#### 2.4.2.3：访问编译安装的nginx web界面：

# Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](http://nginx.org).  
Commercial support is available at [nginx.com](http://nginx.com).

*Thank you for using nginx.*

## 2.4.2.4：创建Nginx自启动脚本：

```

1 [root@s1 ~]# cat /usr/lib/systemd/system/nginx.service
2 [Unit]
3 Description=The nginx HTTP and reverse proxy server
4 After=network.target remote-fs.target nss-lookup.target
5
6 [Service]
7 Type=forking
8 PIDFile=/apps/nginx/logs/nginx.pid
9 # Nginx will fail to start if /run/nginx.pid already exists but has the wrong
10 # SELinux context. This might happen when running `nginx -t` from the cmdline.
11 # https://bugzilla.redhat.com/show_bug.cgi?id=1268621
12 ExecStartPre=/usr/bin/rm -f /apps/nginx/logs/nginx.pid
13 ExecStartPre=/apps/nginx/sbin/nginx -t
14 ExecStart=/apps/nginx/sbin/nginx
15 ExecReload=/bin/kill -s HUP $MAINPID
16 #KillSignal=SIGQUIT
17 #TimeoutStopSec=5
18 KillMode=process
19 PrivateTmp=true
20
21 [Install]
22 WantedBy=multi-user.target

```

## 2.4.2.5：验证Nginx自启动脚本：

```

1 [root@s2 nginx-1.12.2]# systemctl daemon-reload
2 [root@s2 nginx-1.12.2]# systemctl start nginx
3 [root@s2 nginx-1.12.2]# systemctl enable nginx
4 Created symlink from /etc/systemd/system/multi-user.target.wants/nginx.service to
5 /usr/lib/systemd/system/nginx.service.
6 [root@s2 nginx-1.12.2]# systemctl status nginx
7 ● nginx.service - The nginx HTTP and reverse proxy server
8     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset:
9     Active: active (running) since Mon 2019-02-18 11:11:38 CST; 10s ago
10    Main PID: 6348 (nginx)
11      CGroup: /system.slice/nginx.service
12                  └─6348 nginx: master process /apps/nginx/sbin/nginx

```

```
12          |-6349 nginx: worker process
13          |└-6350 nginx: worker process
14
15 Feb 18 11:11:38 s2.example.com systemd[1]: Starting The nginx HTTP and reverse
16 proxy server...
16 Feb 18 11:11:38 s2.example.com nginx[6343]: nginx: the configuration file
17 /apps/nginx/conf/nginx.conf syntax is ok
17 Feb 18 11:11:38 s2.example.com nginx[6343]: nginx: configuration file
18 /apps/nginx/conf/nginx.conf test is successful
18 Feb 18 11:11:38 s2.example.com systemd[1]: Started The nginx HTTP and reverse proxy
server.
```

### 2.5.2.6：配置Nginx：

Nginx的配置文件的组成部分：主配置文件：nginx.conf，子配置文件 include conf.d/\*.conf

```
1 fastcgi , uwsgi , scgi等协议相关的配置文件
2 mime.types : 支持的mime类型 , MIME(Multipurpose Internet Mail Extensions)多用途互联网邮件扩展类型 , MIME消息能包含文本、图像、音频、视频以及其他应用程序专用的数据 , 是设定某种扩展名的文件用一种应用程序来打开的方式类型 , 当该扩展名文件被访问的时候 , 浏览器会自动使用指定应用程序来打开。多用于指定一些客户端自定义的文件名 , 以及一些媒体文件打开方式。
3
4 Nginx主配置文件的配置指令方式 :
5 directive value [value2 ...];
6 注意 :
7 (1) 指令必须以分号结尾
8 (2) 支持使用配置变量
9     内建变量 : 由Nginx模块引入 , 可直接引用
10    自定义变量 : 由用户使用set命令定义
11        set variable_name value;
12    引用变量 : $variable_name
```

MIME参考文档：[https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Basics\\_of\\_HTTP/MIME\\_Types](https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Basics_of_HTTP/MIME_Types)

### 2.5.2.7：默认配置文件：

```
1 [root@s2 ~]# grep -v "#" /apps/nginx/conf/nginx.conf | grep -v "^\$"
2 #全局配置端 , 对全局生效 , 主要设置nginx的启动用户/组 , 启动的工作进程数量 , 工作模式 , Nginx的PID路径 , 日志路径等。
3 user    nginx nginx;
4 worker_processes 1;      #启动工作进程数数量
5 events { #events设置快 , 主要影响nginx服务器与用户的网络连接 , 比如是否允许同时接受多个网络连接 ,
6   使用哪种事件驱动模型处理请求 , 每个工作进程可以同时支持的最大连接数 , 是否开启对多工作进程下的网络连接
7   进行序列化等。
8       worker_connections 1024;      #设置单个nginx工作进程可以接受的最大并发 , 作为web服务器的时候最大并发数为worker_connections * worker_processes , 作为反向代理的时候为
9       (worker_connections * worker_processes)/2
7   }
8 http { #http块是Nginx服务器配置中的重要部分 , 缓存、代理和日志格式定义等绝大多数功能和第三方模块都
9   可以在这设置 , http块可以包含多个server块 , 而一个server块中又可以包含多个location块 , server块可以配置文件引入、 MIME-Type定义、 日志自定义、 是否启用sendfile、 连接超时时间和单个链接的请求上限等。
9       include      mime.types;
```

```

10    default_type application/octet-stream;
11    sendfile          on; #作为web服务器的时候打开sendfile加快静态文件传输，指定是否使用
12    sendfile系统调用来传输文件，sendfile系统调用在两个文件描述符之间直接传递数据(完全在内核中操作)，从
13    而避免了数据在内核缓冲区和用户缓冲区之间的拷贝，操作效率很高，被称之为零拷贝，硬盘 >> kernel
14    buffer (快速拷贝到kernel socket buffer) >>协议栈。
15
16    keepalive_timeout 65; #长连接超时时间，单位是秒
17
18    server { #设置一个虚拟机主机，可以包含自己的全局快，同时也可以包含多个location模块。比如本虚
19    拟机监听的端口、本虚拟机的名称和IP配置，多个server 可以使用一个端口，比如都使用80端口提供web服务、
20    listen      80; #配置server监听的端口
21    server_name localhost; 本server的名称，当访问此名称的时候nginx会调用当前server内部
22    的配置进程匹配。
23
24    location / { #location其实是server的一个指令，为nginx服务器提供比较多而且灵活的指
25    令，都是在location中提现的，主要是基于nginx接受到的请求字符串，对用户请求的URL进行匹配，并对特定
26    的指令进行处理，包括地址重定向、数据缓存和应答控制等功能都是在这部分实现，另外很多第三方模块的配置也
27    是在location模块中配置。
28
29    root   html; #相当于默认页面的目录名称，默认是相对路径，可以使用绝对路径配置。
30    index  index.html index.htm; #默认的页面文件名称
31
32    }
33
34    error_page   500 502 503 504  /50x.html; #错误页面的文件名称
35
36    location = /50x.html { #location处理对应的不同错误码的页面定义到/50x.html，这个跟对
37    应其server中定义的目录下。
38
39    root   html; #定义默认页面所在的目录
40
41    }
42
43    }
44
45
46    #和邮件相关的配置
47    #mail {
48    #
49    #           ...
50    #       }         mail 协议相关配置段
51
52
53    #tcp代理配置，1.9版本以上支持
54    #stream {
55    #
56    #           ...
57    #       }         stream 服务器相关配置段
58
59
60    #导入其他路径的配置文件
61    #include /apps/nginx/conf.d/*.conf
62
63    }
64
65

```

## 三：Nginx 核心配置详解：

### 3.1：全局配置：

```

1 user nginx nginx; #启动Nginx工作进程的用户和组
2 worker_processes [number | auto]; #启动Nginx工作进程的数量
3 worker_cpu_affinity 00000001 00000010 00000100 00001000; #将Nginx工作进程绑定到指定的
4 CPU核心，默认Nginx是不进行进程绑定的，绑定并不是意味着当前nginx进程独占以一核心CPU，但是可以保证
此进程不会运行在其他核心上，这就极大减少了nginx的工作进程在不同的cpu核心上的来回跳转，减少了CPU对
进程的资源分配与回收以及内存管理等，因此可以有效的提升nginx服务器的性能。
5 [root@s2 ~]# ps axo pid,cmd,psr | grep nginx

```

```

5 20061 nginx: master process /apps    0
6 20062 nginx: worker process          0
7 20063 nginx: worker process          1
8 20097 grep --color=auto nginx      0
9
10 #错误日志记录配置，语法：error_log file [debug | info | notice | warn | error | crit |
11   alert | emerg]
12 #error_log logs/error.log;
13 #error_log logs/error.log notice;
14 error_log /apps/nginx/logs/error.log error;
15
16 #pid文件保存路径
17 pid      /apps/nginx/logs/nginx.pid;
18
19 worker_priority 0; #工作进程优先级，-20~19
20 worker_rlimit_nofile 65536; #这个数字包括Nginx的所有连接（例如与代理服务器的连接等），而不仅仅是与客户端的连接，另一个考虑因素是实际的并发连接数不能超过系统级别的最大打开文件数的限制。
21 [root@s2 ~]# watch -n1 'ps -axo pid,cmd,nice | grep nginx' #验证进程优先级
22
23 daemon off; #前台运行Nginx服务用于测试、docker等环境。
24 master_process off|on; #是否开启Nginx的master-worker工作模式，仅用于开发调试场景。
25
26 events {
27     worker_connections 65536; #设置单个工作进程的最大并发连接数
28     use epoll; #使用epoll事件驱动，Nginx支持众多的事件驱动，比如select、poll、epoll，只能设置在events模块中设置。
29     accept_mutex on; #优化同一时刻只有一个请求而避免多个睡眠进程被唤醒的设置，on为防止被同时唤醒，默认为off，全部唤醒的过程也成为“惊群”，因此nginx刚安装完以后要进行适当的优化。
30     multi_accept on; Nginx服务器的每个工作进程可以同时接受多个新的网络连接，但是需要在配置文件中配置，此指令默认为关闭，即默认为一个工作进程只能一次接受一个新的网络连接，打开后几个同时接受多个。
31 }

```

## 3.2 : http详细配置：

```

1 http {
2     include      mime.types; #导入支持的文件类型
3     default_type application/octet-stream; #设置默认的类型，会提示下载不匹配的类型文件
4
5 #日志配置部分
6     log_format main '$remote_addr - $remote_user [$time_local] "$request" '
7     '#           "$status $body_bytes_sent \"$http_referer\" '
8     '#           '\"$http_user_agent\" \"$http_x_forwarded_for\"';'
9     access_log  logs/access.log  main;
10
11 #自定义优化参数
12     sendfile      on;
13     #tcp_nopush    on; #在开启了sendfile的情况下，合并请求后统一发送给客户端。
14     #tcp_nodelay   off; #在开启了keepalive模式下的连接是否启用TCP_NODELAY选项，当为off时，延迟0.2s发送，默认on时，不延迟发送，立即发送用户相应报文。
15     #keepalive_timeout 0;
16     keepalive_timeout 65 65; #设置会话保持时间
17     #gzip      on; #开启文件压缩

```

```
18
19     server {
20         listen      80; #设置监听地址和端口
21         server_name localhost; #设置server name , 可以以空格隔开写多个并支持正则表达式 , 如
*.magedu.com      www.magedu.* ~^www\d+\.\magedu\.\com$ default_server
22         #charset koi8-r; #设置编码格式 , 默认是俄语格式 , 可以改为utf-8
23         #access_log  logs/host.access.log  main;
24         location / {
25             root    html;
26             index   index.html index.htm;
27         }
28
29         #error_page  404          /404.html;
30         # redirect server error pages to the static page /50x.html
31         #
32         error_page  500 502 503 504  /50x.html; #定义错误页面
33         location = /50x.html {
34             root    html;
35         }
36
37         # proxy the PHP scripts to Apache listening on 127.0.0.1:80
38         #
39         #location ~ \.php$ { #以http的方式转发php请求到指定web服务器
40             proxy_pass    http://127.0.0.1;
41         }
42
43         # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
44         #
45         #location ~ \.php$ { #以fastcgi的方式转发php请求到php处理
46             root        html;
47             fastcgi_pass 127.0.0.1:9000;
48             fastcgi_index index.php;
49             fastcgi_param SCRIPT_FILENAME  /scripts$fastcgi_script_name;
50             include      fastcgi_params;
51         }
52
53         # deny access to .htaccess files, if Apache's document root
54         # concurs with nginx's one
55         #
56         #location ~ /\.ht { #拒绝web形式访问指定文件 , 如很多的网站都是通过.htaccess文件来改
变自己的重定向等功能。
57             #    deny  all;
58         }
59         location ~ /passwd.html {
60             deny  all;
61         }
62
63     }
64
65     # another virtual host using mix of IP-, name-, and port-based configuration
66     #
67     #server { #自定义虚拟server
68         #    listen      8000;
```

```

69      #     listen      somename:8080;
70      #     server_name  somename alias another.alias;
71
72      #     location / {
73      #         root   html;
74      #         index  index.html index.htm; #指定默认网页文件，此指令由
    ngx_http_index_module模块提供
75
76      #     }
77      #}
78
79      # HTTPS server
80      #
81      #server { #https服务器配置
82      #     listen      443 ssl;
83      #     server_name  localhost;
84
85      #     ssl_certificate      cert.pem;
86      #     ssl_certificate_key  cert.key;
87
88      #     ssl_session_cache    shared:SSL:1m;
89      #     ssl_session_timeout  5m;
90
91      #     ssl_ciphers  HIGH:!aNULL:!MD5;
92      #     ssl_prefer_server_ciphers  on;
93
94      #     location / {
95      #         root   html;
96      #         index  index.html index.htm;
97      #     }
98      #}

```

## 3.3：核心配置示例：

基于不同的IP、不同的端口以及不用得域名实现不同的虚拟主机，依赖于核心模块ngx\_http\_core\_module实现。

### 3.3.1：新建一个PC web站点：

```

1 [root@s2 ~]# mkdir  /apps/nginx/conf/conf.d
2 [root@s2 ~]# cat /apps/nginx/conf/conf.d/pc.conf
3 server {
4     listen 80;
5     server_name www.magedu.net;
6     location / {
7         root /data/nginx/html/pc;
8     }
9 }
10
11 [root@s2 ~]# mkdir /data/nginx/html/pc -p
12 [root@s2 ~]# echo "pc web" > /data/nginx/html/pc/index.html
13 [root@s2 ~]# vim /apps/nginx/conf/nginx.conf
14     include /apps/nginx/conf/conf.d/*.conf;

```

```
15 [root@s2 ~]# systemctl reload nginx  
16 访问测试
```

### 3.3.2：新建一个Mobile web站点：

```
1 [root@s2 ~]# cat /apps/nginx/conf/conf.d/mobile.conf  
2 server {  
3     listen 80;  
4     server_name mobile.magedu.net;  
5     location / {  
6         root /data/nginx/html/mobile;  
7     }  
8 }  
9 [root@s2 ~]# mkdir /data/nginx/html/mobile -p  
10 [root@s2 ~]# echo "mobile web" >> /data/nginx/html/mobile/index.html  
11  
12 [root@s2 ~]# systemctl reload nginx
```

### 3.3.3：root与alias：

root：指定web的家目录，在定义location的时候，文件的绝对路径等于 root+location，如：

```
1 server {  
2     listen 80;  
3     server_name www.magedu.net;  
4     location / {  
5         root /data/nginx/html/pc;  
6     }  
7  
8     location /about {  
9         root /data/nginx/html/pc; #必须要在html目录中创建一个about目录才可以访问，否则报错。  
10        index index.html;  
11    }  
12 }  
13  
14 [root@s2 ~]# mkdir /data/nginx/html/pc/about  
15 [root@s2 ~]# echo about > /data/nginx/html/pc/about/index.html  
16  
17 重启Nginx并访问测试
```

alias：定义路径别名，会把访问的路径重新定义到其指定的路径，如：

```
1  
2 server {  
3     listen 80;  
4     server_name www.magedu.net;  
5     location / {  
6         root /data/nginx/html/pc;  
7     }  
8 }
```

```
9 location /about { #使用alias的时候uri后面如果加了斜杠则下面的路径配置必须加斜杠，否则403
10     alias /data/nginx/html/pc; #当访问about的时候，会显示alias定义的/data/nginx/html/pc
11     index index.html;
12 }
13 }
14
15 重启Nginx并访问测试
```

### 3.3.4 : location的详细使用：

在没有使用正则表达式的时候，nginx会先在server中的多个location选取匹配度最高的一个uri，uri是用户请求的字符串，即域名后面的web文件路径，然后使用该location模块中的正则url和字符串，如果匹配成功就结束搜索，并使用此location处理此请求。

```
1 语法规则： location [=|~|~*|^~] /uri/ { ... }
2
3 = #用于标准uri前，需要请求字串与uri精确匹配，如果匹配成功就停止向下匹配并立即处理请求。
4 ~ #用于标准uri前，表示包含正则表达式并且区分大小写
5 ~* #用于标准uri前，表示包含正则表达式并且不区分大小写
6 !~ #用于标准uri前，表示包含正则表达式并且区分大小写不匹配
7 !~* #用于标准uri前，表示包含正则表达式并且不区分大小写不匹配
8 ^~ #用于标准uri前，表示包含正则表达式并且匹配以什么开头
9 $ #用于标准uri前，表示包含正则表达式并且匹配以什么结尾
10 \ #用于标准uri前，表示包含正则表达式并且转义字符。可以转. * ?等
11 * #用于标准uri前，表示包含正则表达式并且代表任意长度的任意字符
```

#### 3.3.4.1 : 匹配案例-精确匹配：

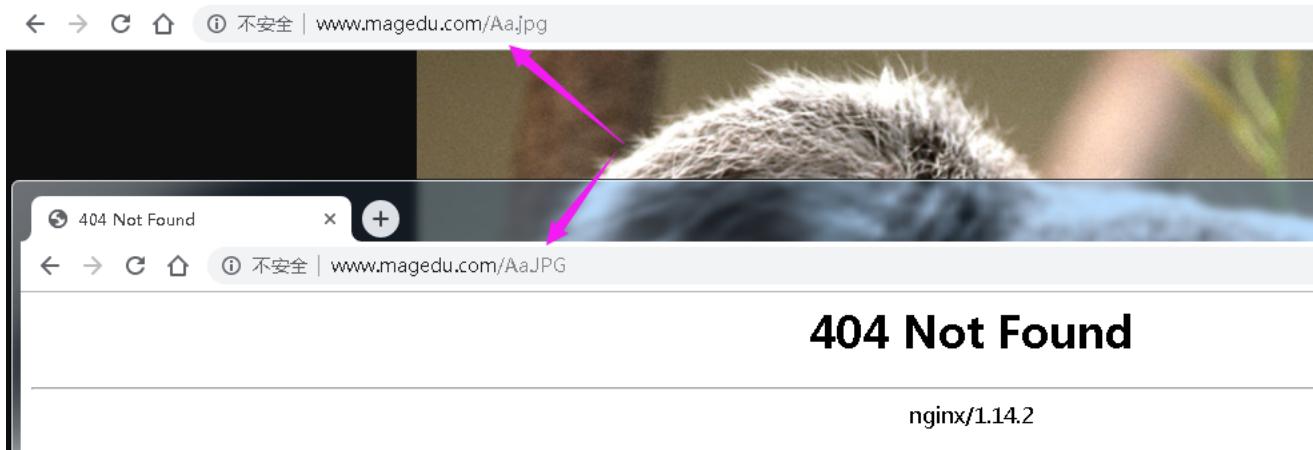
在server部分使用location配置一个web界面，要求：当访问nginx 服务器的/login的时候要显示指定html文件的内容：

```
1 [root@s2 ~]# cat /apps/nginx/conf/conf.d/pc.conf
2 server {
3     listen 80;
4     server_name www.magedu.net;
5     location / {
6         root /data/nginx/html/pc;
7     }
8     location = /1.jpg {
9         root /var/www/nginx/images;
10        index index.html;
11    }
12 }
13 上传图片到/var/www/nginx/images，重启Nginx并访问测试
14 访问测试：http://www.magedu.net/1.jpg
```

#### 3.3.4.2 : 匹配案例-区分大小写：

如果uri中包含大写字母，组此条件不匹配

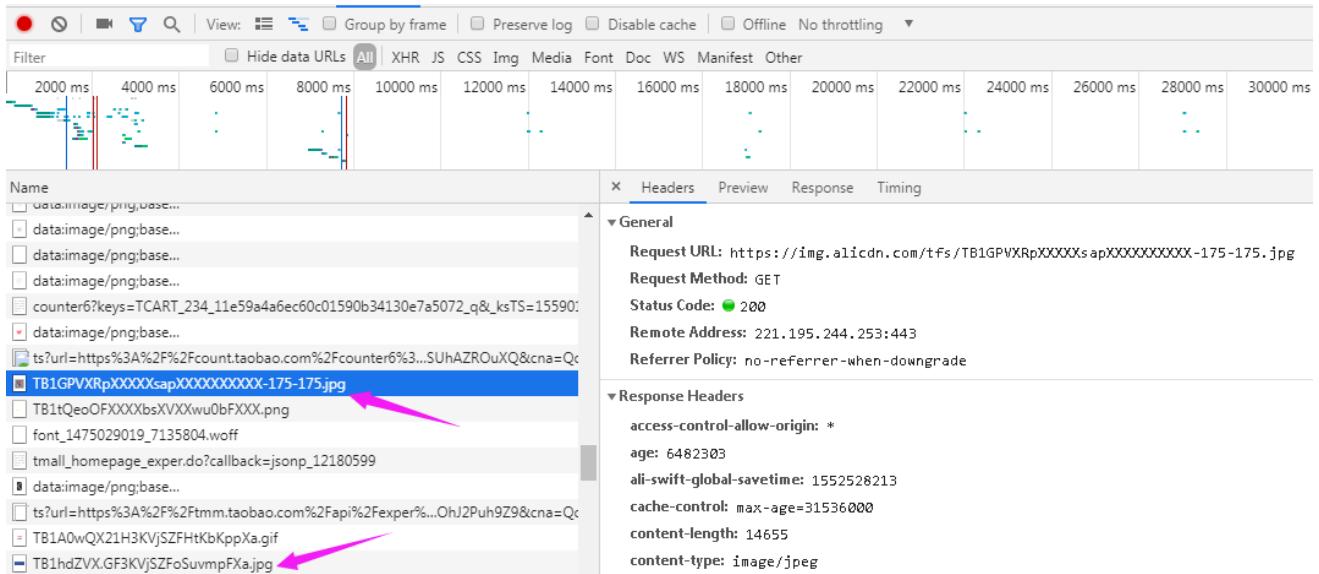
```
1 location ~ /A.?\.jpg {  
2     index index.html;  
3     root /opt/nginx/html/image;  
4 }  
5  
6 重启Nginx并访问测试  
7 将只能访问以小写字符的AX.jpg图片，不能识别大写的JPG结尾的图片
```



### 3.3.4.3：匹配案例-不区分大小写：

对用户请求的uri做模糊匹配，也就是uri中无论都是大写、都是小写或者大小写混合，此模式也都会匹配，通常使用此模式匹配用户request中的静态资源并继续做下一步操作。

```
1 正则表达式匹配：  
2 # location ~ /A.?\.jpg {  
3 #     index index.html;  
4 #     root /opt/nginx/html/image;  
5 # }  
6  
7 location ~* /A.?\.jpg {  
8     index index.html;  
9     root /opt/nginx/html/image;  
10 }  
11 ~~~~~  
12 精确匹配指定名称：  
13 # location ~ /aa.jpg {  
14 #     index index.html;  
15 #     root /opt/nginx/html/image;  
16 # }  
17  
18 location ~* /aa.jpg {  
19     index index.html;  
20     root /opt/nginx/html/image;  
21 }  
22  
23  
24 重启Nginx并访问测试  
25 对于不区分大小写的location，则可以访问任意大小写结尾的图片文件，如区分大小写则只能访问aa.jpg，不区分大小写则可以访问aa.jpg以外的资源比如Aa.JPG、aA.JPG这样的混合名称文件
```



### 3.3.4.4 : 匹配案例-URI开始 :

```

1 location ^~ /images {
2     root /data/nginx;
3     index index.html;
4 }
5
6 location /images1 {
7     alias /data/nginx/html/pc;
8     index index.html;
9 }
10 重启Nginx并访问测试，实现效果是访问images和images1返回不同的结果
11 [root@s2 images]# curl http://www.magedu.net/images/
12 images
13 [root@s2 images]# curl http://www.magedu.net/images1/
14 pc web

```

### 3.3.4.5 : 匹配案例-文件名后缀 :

```

1 [root@s2 ~]# mkdir /data/nginx/images1
2 #上传一个和images目录不一样内容的图片1.jpg到/data/nginx/images1
3
4 location ~* \.(gif|jpg|jpeg|bmp|png|tiff|tif|ico|wmf|js)$ {
5     root /data/nginx/images1;
6     index index.html;
7 }
8 重启Nginx并访问测试

```

### 3.3.4.6 : 匹配案例-优先级 :

```

1 location /1.jpg {
2     root /var/www/nginx/images;
3     index index.html;
4 }

```

```

5
6 #   location  /1.jpg {
7 #     root /var/www/nginx/images;
8 #     index index.html;
9 #   }
10
11 #   location ~* \.(gif|jpg|jpeg|bmp|png|tiff|tif|ico|wmf|js)$ {
12 #     root /data/nginx/images1;
13 #     index index.html;
14 #   }
15
16 [root@s2 ~]# mkdir /var/www/nginx/images -p
17 上传图片到/var/www/nginx/images并重启nginx访问测试
18
19 匹配优先级 :=, ^~, ~/~*, /
20 location优先级：(location =) > (location 完整路径) > (location ^~ 路径) > (location ~,~* 正则顺序) > (location 部分起始路径) > (/)

```

### 3.3.4.7 : 生产使用案例:

```

1 直接匹配网站根会加速Nginx访问处理：
2 location = / {
3   ....;
4 }
5
6 location / {
7   ....;
8 }
9
10 静态资源配置：
11 location ^~ /static/ {
12   ....;
13 }
14 # 或者
15 location ~* \.(gif|jpg|jpeg|png|css|js|ico)$ {
16   ....;
17 }
18
19 多应用配置
20 location ~* /app1 {
21   ....;
22 }
23 location ~* /app2 {
24   ....;
25 }
26

```

### 3.3.5 : Nginx 四层访问控制：

访问控制基于模块ngx\_http\_access\_module实现，可以通过匹配客户端源IP地址进行限制。

```
1 location = /login/ {
2     root /data/nginx/html/pc;
3 }
4
5 location /about {
6     alias /data/nginx/html/pc;
7     index index.html;
8     deny 192.168.1.1;
9     allow 192.168.1.0/24;
10    allow 10.1.1.0/16;
11    allow 2001:0db8::/32;
12    deny all; #先允许小部分，再拒绝大部分
13 }
```

### 3.3.6：Nginx账户认证功能：

```
1 [root@s2 ~]# yum install httpd-tools -y
2 [root@s2 ~]# htpasswd -cbm /apps/nginx/conf/.htpasswd user1 123456
3 Adding password for user user1
4 [root@s2 ~]# htpasswd -bm /apps/nginx/conf/.htpasswd user2 123456
5 Adding password for user user2
6 [root@s2 ~]# tail /apps/nginx/conf/.htpasswd
7 user1:$apr1$Rjm0u2Kr$VHvkAIc5OYg.3ZoaGwaGq/
8 user2:$apr1$nIqnxoJB$LR9W1DTJT.viDjhxa6wHv.
9
10 [root@s2 ~]# vim /apps/nginx/conf/conf.d/pc.conf
11 location = /login/ {
12     root /data/nginx/html/pc;
13     index index.html;
14     auth_basic "login password";
15     auth_basic_user_file /apps/nginx/conf/.htpasswd;
16 }
17
18 重启Nginx并访问测试
```

### 3.3.7：自定义错误页面：

```
1 listen 80;
2 server_name www.magedu.net;
3 error_page 500 502 503 504 404 /error.html;
4 location = /error.html {
5     root html;
6 }
7 重启nginx并访问不存在的页面进行测试
```

### 3.3.8：自定义访问日志：

```

1 [root@s2 ~]# mkdir /data/nginx/logs
2   listen 80;
3   server_name www.magedu.net;
4   error_page 500 502 503 504 404 /error.html; #默认目录下面创建error.html页面
5   access_log /data/nginx/logs/www-magedu-net_access.log;
6   error_log /data/nginx/logs/www-magedu-net_error.log;
7   location = /error.html {
8     root html;
9   }
10 重启nginx并访问不存在的页面进行测试并验证是在指定目录生成新的日志文件

```

### 3.3.9 : 检测文件是否存在 :

try\_files会按顺序检查文件是否存在，返回第一个找到的文件或文件夹（结尾加斜线表示为文件夹），如果所有文件或文件夹都找不到，会进行一个内部重定向到最后一个参数。只有最后一个参数可以引起一个内部重定向，之前的参数只设置内部URI的指向。最后一个参数是回退URI且必须存在，否则会出现内部500错误。

```

1 location /about {
2   root /data/nginx/html/pc;
3   #alias /data/nginx/html/pc;
4   index index.html;
5   #try_files $uri /about/default.html;
6   #try_files $uri $uri/index.html $uri.html /about/default.html;
7   try_files $uri $uri/index.html $uri.html =489;
8 }
9 [root@s2 ~]# echo "default" >> /data/nginx/html/pc/about/default.html
10 重启nginx并测试，当访问到http://www.magedu.net/about/xx.html等不存在的uri会显示default，如果是自定义的状态码则会显示在返回数据的状态码中，如：
11 [root@s2 about]# curl --head http://www.magedu.net/about/xx.html
12 HTTP/1.1 489 #489就是自定义的状态返回码
13 Server: nginx
14 Date: Thu, 21 Feb 2019 00:11:40 GMT
15 Content-Length: 0
16 Connection: keep-alive
17 Keep-Alive: timeout=65

```

### 3.3.10 : 长连接配置 :

keepalive\_timeout number; #设定保持连接超时时长，0表示禁止长连接，默认为75s，通常配置在http字段作为站点全局配置  
keepalive\_requests number; #在一次长连接上所允许请求的资源的最大数量，默认为100次

```

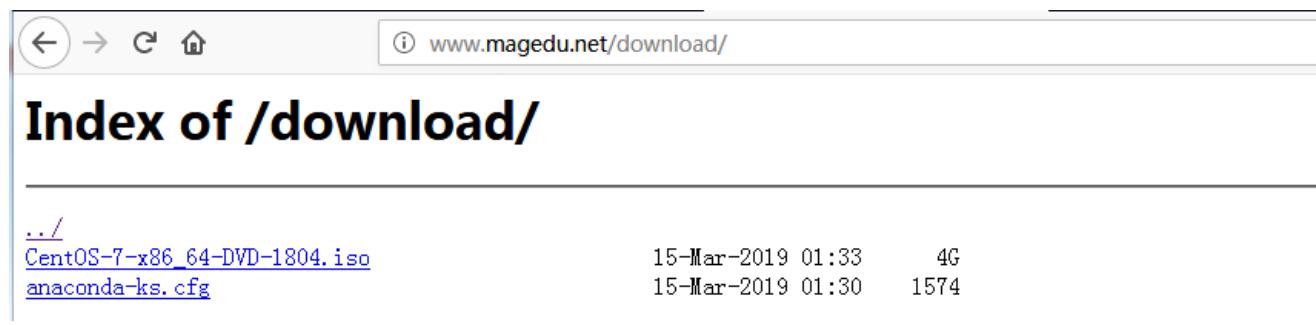
1   keepalive_requests 3;
2   keepalive_timeout 65 65;
3   启开长连接后，返回客户端的会话保持时间为60s，单次长连接累计请求达到指定次数请求或65秒就会被断开，后面的60为发送给客户端应答报文头部中显示的超时时间设置为60s：如不设置客户端将不显示超时时间。
4   Keep-Alive:timeout=60 #浏览器收到的服务器返回的报文
5
6   如果设置为0表示关闭会话保持功能，将如下显示：
7   Connection:close #浏览器收到的服务器返回的报文
8   使用命令测试：
9   [root@s3 apps]# telnet www.magedu.net 80

```

```
10 Trying 172.18.200.102...
11 Connected to www.magedu.net.
12 Escape character is '^]'.
13 GET / HTTP/1.1
14 HOST: www.magedu.net
15
16 #Response Headers(响应头信息) :
17 HTTP/1.1 200 OK
18 Server: nginx/1.14.2
19 Date: Thu, 14 Mar 2019 17:23:46 GMT
20 Content-Type: text/html
21 Content-Length: 7
22 Last-Modified: Thu, 14 Mar 2019 14:54:50 GMT
23 Connection: keep-alive
24 Keep-Alive: timeout=60
25 ETag: "5c8a6b3a-7"
26 Accept-Ranges: bytes
27
28 #页面内容
29 pc web
```

### 3.3.11：作为下载服务器配置：

```
1 [root@s2 about]# mkdir /data/nginx/html/pc/download
2 #download不需要index.html文件
3 [root@s2 about]# vim /apps/nginx/conf/conf.d/pc.conf
4   location /download {
5     autoindex on;      #自动索引功能
6     autoindex_exact_size on; #计算文件确切大小 (单位bytes) , off只显示大概大小 (单位kb、
7     mb、gb)
8     autoindex_localtime on; #显示本机时间而非GMT(格林威治)时间
9     root /data/nginx/html/pc;
10   }
11 [root@s2 pc]# cp /root/anaconda-ks.cfg /data/nginx/html/pc/download/
12 重启Nginx并访问测试下载页面
```



The screenshot shows a web browser window with the URL [www.magedu.net/download/](http://www.magedu.net/download/) in the address bar. The page title is "Index of /download/". Below the title, there is a table listing two files:

|     | File                                         | Date              | Size |
|-----|----------------------------------------------|-------------------|------|
| ... | <a href="#">CentOS-7-x86_64-DVD-1804.iso</a> | 15-Mar-2019 01:33 | 4G   |
|     | <a href="#">anaconda-ks.cfg</a>              | 15-Mar-2019 01:30 | 1574 |

```
1 limit_rate rate; #限制响应给客户端的传输速率，单位是bytes/second，默认值0表示无限制
2 限速与不限速的对比：
3   limit_rate 10k;
```

```
[root@s3 ~]# wget http://www.magedu.net/download/CentOS-7-x86_64-DVD-1804.iso
--2019-03-15 18:15:03--  http://www.magedu.net/download/CentOS-7-x86_64-DVD-1804.iso
Resolving www.magedu.net (www.magedu.net)... 172.18.200.102
Connecting to www.magedu.net (www.magedu.net)|172.18.200.102|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4470079488 (4.2G) [application/octet-stream]
Saving to: ' CentOS-7-x86_64-DVD-1804.iso'

0% [=====] 163,840      10.5KB/s eta 4d 19h
```

```
[root@s3 ~]# wget http://www.magedu.net/download/CentOS-7-x86_64-DVD-1804.iso
--2019-03-15 18:18:15--  http://www.magedu.net/download/CentOS-7-x86_64-DVD-1804.iso
Resolving www.magedu.net (www.magedu.net)... 172.18.200.102
Connecting to www.magedu.net (www.magedu.net)|172.18.200.102|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4470079488 (4.2G) [application/octet-stream]
Saving to: ' CentOS-7-x86_64-DVD-1804.iso.3'

3% [==>-----] 168,498,088 100MB/s
```

### 3.3.12：作为上传服务器：

```
1 client_max_body_size 1m; #设置允许客户端上传单个文件的最大值，默认值为1m
2 client_body_buffer_size size; #用于接收每个客户端请求报文的body部分的缓冲区大小；默认16k；
3 超出此大小时，其将被暂存到磁盘上的由下面client_body_temp_path指令所定义的位置
4 client_body_temp_path path [level1 [level2 [level3]]];
5 #设定存储客户端请求报文的body部分的临时存储路径及子目录结构和数量，目录名为16进制的数字，使用hash
6 之后的值从后往前截取1位、2位、2位作为文件名：
7 [root@s3 ~]# md5sum /data/nginx/html/pc/index.html
8 95f6f65f498c74938064851b1bb 96 3d 4 /data/nginx/html/pc/index.html
9 1级目录占1位16进制，即 $2^4=16$ 个目录 0-f
10 2级目录占2位16进制，即 $2^8=256$ 个目录 00-ff
11 3级目录占2位16进制，即 $2^8=256$ 个目录 00-ff
12 配置示例：
13     client_max_body_size 10m;
14     client_body_buffer_size 16k;
15     client_body_temp_path /apps/nginx/temp 1 2 2; #reload Nginx会自动创建temp目录
```

```
[root@iZ62ovkwaktZ ~]# tree /usr/local/nginx/tmp/
/usr/local/nginx/tmp/
└── 1
    └── 54
        └── 07
└── 5
    └── 56
        └── 07

6 directories, 0 files
[root@iZ62ovkwaktZ ~]# _
```

### 3.3.13：其他配置：

```
1 keepalive_disable none | browser ...;
2 #对哪种浏览器禁用长连接
```

```
1 limit_except method ... { ... }, 仅用于location
2 限制客户端使用除了指定的请求方法之外的其它方法
3     method:GET, HEAD, POST, PUT, DELETE , MKCOL, COPY, MOVE, OPTIONS, PROPFIND,
4     PROPPATCH, LOCK, UNLOCK, PATCH
5     limit_except GET {
```

```

5      allow 192.168.0.0/24;
6      allow 192.168.7.101;
7      deny all;
8  }
9
10 #除了GET和HEAD之外其它方法仅允许192.168.1.0/24网段主机使用
11 [root@s2 about]# mkdir /data/nginx/html/pc/upload
12 [root@s2 about]# echo "upload" > /data/nginx/html/pc/upload/index.html
13 [root@s2 about]# vim /apps/nginx/conf/conf.d/pc.conf
14 location /upload {
15     root /data/magedu/pc;
16     index index.html;
17     limit_except GET {
18         allow 172.18.200.101;
19         deny all;
20     }
21 }
22
23
24 #重启Nginx并进行测试上传文件
25 [root@s2 pc]# systemctl restart nginx
26 [root@s2 pc]#
27 [root@s2 pc]# curl -XPUT /etc/issue http://www.magedu.net/about
28 curl: (3) <url> malformed
29 <html>
30 <head><title>405 Not Allowed</title></head> #Nginx已经允许但是程序未支持上传功能
31 <body bgcolor="white">
32 <center><h1>405 Not Allowed</h1></center>
33 <hr><center>nginx</center>
34 </body>
35 </html>
36 [root@s1 ~]# curl -XPUT /etc/issue http://www.magedu.net/upload
37 curl: (3) <url> malformed
38 <html>
39 <head><title>403 Forbidden</title></head> #Nginx拒绝上传
40 <body bgcolor="white">
41 <center><h1>403 Forbidden</h1></center>
42 <hr><center>nginx</center>
43 </body>
44 </html>

```

```

1 aio on | off #是否启用asynchronous file I/O(AIO)功能，需要编译开启
2 Linux 2.6以上内核提供以下几个系统调用来支持aio：
3 1、SYS_io_setup：建立aio 的context
4 2、SYS_io_submit：提交I/O操作请求
5 3、SYS_io_getevents：获取已完成的I/O事件
6 4、SYS_io_cancel：取消I/O操作请求
7 5、SYS_io_destroy：毁销aio的context

```

```

1 directio size | off; #操作完全和aio相反，aio是读取文件而directio是写文件到磁盘，启用直接I/O，默认为关闭，当文件大于等于给定大小时，例如directio 4m，同步（直接）写磁盘，而非写缓存。

```

```
1 open_file_cache off; #是否缓存打开过的文件信息
2 open_file_cache max=N [inactive=time];
3     nginx可以缓存以下三种信息：
4     (1) 文件元数据：文件的描述符、文件大小和最近一次的修改时间
5     (2) 打开的目录结构
6     (3) 没有找到的或者没有权限访问的文件的相关信息
7     max=N：可缓存的缓存项上限数量；达到上限后会使用LRU(Least recently used，最近最少使用)算法实
8     现管理
9     inactive=time：缓存项的非活动时长，在此处指定的时长内未被命中的或命中的次数少于
10    open_file_cache_min_uses指令所指定的次数的缓存项即为非活动项，将被删除
```

```
1 open_file_cache_errors on | off;
2     是否缓存查找时发生错误的文件一类的信息
3     默认值为off
```

```
1 open_file_cache_min_uses number;
2     open_file_cache指令的inactive参数指定的时长内，至少被命中此处指定的次数方可被归类为活动项
3     默认值为1
```

```
1 open_file_cache_valid time;
2     缓存项有效性的检查验证频率，默认值为60s
3
4 open_file_cache max=10000 inactive=60s; #最大缓存10000个文件，非活动数据超时时长60s
5 open_file_cache_valid 60s; #每间隔60s检查一下缓存数据有效性
6 open_file_cache_min_uses 5; #60秒内至少被命中访问5次才被标记为活动数据
7 open_file_cache_errors on; #缓存错误信息
```

```
1 server_tokens off; #隐藏Nginx server版本。
```

## 四：Nginx 高级配置：

### 4.1：Nginx 状态页：

基于nginx模块ngx\_http\_auth\_basic\_module实现，在编译安装nginx的时候需要添加编译参数--with-http\_stub\_status\_module，否则配置完成之后监测会是提示语法错误。

```
1 配置示例：
2 location /nginx_status {
3     stub_status;
4     allow 192.168.0.0/16;
5     allow 127.0.0.1;
6     deny all;
7 }
8
9 状态页用于输出nginx的基本状态信息：
10    输出信息示例：
11    Active connections: 291
```

```
12 server accepts handled requests
13     16630948 16630948 31070465
14     上面三个数字分别对应accepts,handled,requests三个值
15 Reading: 6 Writing: 179 Waiting: 106
16
17 Active connections : 当前处于活动状态的客户端连接数，包括连接等待空闲连接数。
18 accepts : 统计总值，Nginx自启动后已经接受的客户端请求的总数。
19 handled : 统计总值，Nginx自启动后已经处理完成的客户端请求的总数，通常等于accepts，除非有因
worker_connections限制等被拒绝的连接。
20 requests : 统计总值，Nginx自启动后客户端发来的总的请求数。
21 Reading : 当前状态，正在读取客户端请求报文首部的连接的连接数。
22 Writing : 当前状态，正在向客户端发送响应报文过程中的连接数。
23 Waiting : 当前状态，正在等待客户端发出请求的空闲连接数，开启 keep-alive的情况下，这个值等于
active - (reading+writing) ,
```

## 4.2 : Nginx 第三方模块 :

第三模块是对nginx 的功能扩展，第三方模块需要在编译安装Nginx 的时候使用参数--add-module=PATH指定路径添加，有的模块是由公司的开发人员针对业务需求定制开发的，有的模块是开源爱好者开发好之后上传到github进行开源的模块，nginx支持第三方模块需要从源码重新编译支持，比如开源的echo模块 <https://github.com/openresty/echo-nginx-module>：

```
1 [root@s2 pc]# systemctl stop nginx
2 [root@s2 pc]# vim /apps/nginx/conf/conf.d/pc.conf
3 location /main {
4     index index.html;
5     default_type text/html;
6     echo "hello world,main-->";
7     echo_reset_timer;
8     echo_location /sub1;
9     echo_location /sub2;
10    echo "took $echo_timer_elapsed sec for total.";
11 }
12 location /sub1 {
13     echo_sleep 1;
14     echo sub1;
15 }
16 location /sub2 {
17     echo_sleep 1;
18     echo sub2;
19 }
20
21 [root@s2 pc]# /apps/nginx/sbin/nginx -t
22 nginx: [emerg] unknown directive "echo_reset_timer" in
23 /apps/nginx/conf/conf.d/pc.conf:86
24 nginx: configuration file /apps/nginx/conf/nginx.conf test failed
25 #解决以上报错问题
26 [root@s2 src]# yum install git -y
27 [root@s2 src]# git clone https://github.com/openresty/echo-nginx-module.git
28 [root@s2 src]# cd nginx-1.12.2/
29 [root@s2 src]# ./configure \
```

```

30 --prefix=/apps/nginx \
31 --user=nginx --group=nginx \
32 --with-http_ssl_module \
33 --with-http_v2_module \
34 --with-http_realip_module \
35 --with-http_stub_status_module \
36 --with-http_gzip_static_module \
37 --with-pcre \
38 --with-stream \
39 --with-stream_ssl_module \
40 --with-stream_realip_module \
41 --with-http_perl_module \
42 --add-module=/usr/local/src/echo-nginx-module
43 [root@s2 src]# make && make install
44
45 #确认语法检测通过
46 [root@s2 pc]# /apps/nginx/sbin/nginx -t
47 nginx: the configuration file /apps/nginx/conf/nginx.conf syntax is ok
48 nginx: configuration file /apps/nginx/conf/nginx.conf test is successful
49
50 #重启nginx访问测试：
51 [root@s2 pc]# systemctl restart nginx
52 [root@s2 pc]# curl http://www.magedu.net/main
53 hello world,main-->
54 sub1
55 sub2
56 took 2.010 sec for total.

```

## 4.3 : Nginx 变量使用：

nginx的变量可以在配置文件中引用，作为功能判断或者日志等场景使用，变量可以分为内置变量和自定义变量，内置变量是由nginx模块自带，通过变量可以获取到众多的与客户端访问相关的值。

### 4.3.1：内置变量：

```

1 $remote_addr;
2 #存放了客户端的地址，注意是客户端的公网IP，也就是一家人访问一个网站，则会显示为路由器的公网IP。

```

```

1 $args ;
2 #变量中存放了URL中的指令，例如http://www.magedu.net/main/index.do?
   id=20190221&partner=search中的id=20190221&partner=search

```

```

1 $document_root ;
2 #保存了针对当前资源的请求的系统根目录，如/apps/nginx/html。

```

```

1 $document_uri ;
2 #保存了当前请求中不包含指令的URI，注意是不包含请求的指令，比如
   http://www.magedu.net/main/index.do?id=20190221&partner=search会被定义为/main/index.do
   。

```

```
1 $host ;  
2 #存放了请求的host名称。
```

```
1 $http_user_agent ;  
2 #客户端浏览器的详细信息
```

```
1 $http_cookie ;  
2 #客户端的cookie信息。
```

```
1 limit_rate 10240;  
2 echo $limit_rate;  
3 #如果nginx服务器使用limit_rate配置了显示网络速率，则会显示，如果没有设置，则显示0。
```

```
1 $remote_port ;  
2 #客户端请求Nginx服务器时随机打开的端口，这是每个客户端自己的端口。
```

```
1 $remote_user ;  
2 #已经经过Auth Basic Module验证的用户名。
```

```
1 $request_body_file ;  
2 #做反向代理时发给后端服务器的本地资源的名称。
```

```
1 $request_method ;  
2 #请求资源的方式，GET/PUT/DELETE等
```

```
1 $request_filename ;  
2 #当前请求的资源文件的路径名称，由root或alias指令与URI请求生成的文件绝对路径，  
如/apps/nginx/html/main/index.html
```

```
1 $request_uri ;  
2 #包含请求参数的原始URI，不包含主机名，如：/main/index.do?id=20190221&partner=search。
```

```
1 $scheme ;  
2 #请求的协议，如ftp，https，http等。
```

```
1 $server_protocol ;  
2 #保存了客户端请求资源使用的协议的版本，如HTTP/1.0，HTTP/1.1，HTTP/2.0等。
```

```
1 $server_addr ;  
2 #保存了服务器的IP地址。
```

```
1 $server_name ;  
2 #请求的服务器的主机名。
```

```
1 $server_port ;  
2 #请求的服务器的端口号。
```

### 4.3.2 : 自定义变量 :

假如需要自定义变量名称和值，使用指令set \$variable value;，则方法如下：

Syntax: set \$variable value; Default: — Context: server, location, if

```
1      set $name magedu;  
2      echo $name;  
3      set $my_port $server_port;  
4      echo $my_port;  
5      echo "$server_name:$server_port";
```

## 4.4 : Nginx 自定义访问日志 :

访问日志是记录客户端即用户的具体请求内容信息，全局配置模块中的error\_log是记录nginx服务器运行时的日志保存路径和记录日志的level，因此有着本质的区别，而且Nginx的错误日志一般只有一个，但是访问日志可以在不同server中定义多个，定义一个日志需要使用access\_log指定日志的保存路径，使用log\_format指定日志的格式，格式中定义要保存的具体日志内容。

### 4.4.1 : 自定义默认格式日志 :

如果是要保留日志的源格式，只是添加相应的日志内容，则配置如下：

```
1      log_format nginx_format1 '$remote_addr - $remote_user [$time_local]  
2      "$request" '  
3              '$status $body_bytes_sent "$http_referer" '  
4              '"$http_user_agent" "$http_x_forwarded_for"'  
5              '$server_name:$server_port';  
6  
6      access_log logs/access.log nginx_format1;  
7  
8 #重启nginx并访问测试日志格式  
9 ==> /apps/nginx/logs/access.log <=  
10 192.168.0.1 - - [22/Feb/2019:08:44:14 +0800] "GET /favicon.ico HTTP/1.1" 404 162 "-  
" "Mozilla/5.0 (Windows NT 6.1; win64; x64; rv:65.0) Gecko/2  
11 0100101 Firefox/65.0" "-"www.magedu.net:80
```

### 4.4.2 : 自定义json格式日志 :

Nginx 的默认访问日志记录内容相对比较单一，默认的格式也不方便后期做日志统计分析，生产环境中通常将nginx日志转换为json日志，然后配合使用ELK做日志收集-统计-分析。

```
1      log_format access_json '["@timestamp":'$time_iso8601','  
2          '"host":'$server_addr','  
3          '"clientip":'$remote_addr','  
4          '"size":'$body_bytes_sent,'  
5          '"responsetime":'$request_time,'
```

```

6      '"upstreamtime":"$upstream_response_time",'
7      '"upstreamhost":"$upstream_addr",'
8      '"http_host":"$host",'
9      '"uri":"$uri",'
10     '"domain":"$host",'
11     '"xff":"$http_x_forwarded_for",'
12     '"referer":"$http_referer",'
13     '"tcp_xff":"$proxy_protocol_addr",'
14     '"http_user_agent":"$http_user_agent",'
15     '"status":"$status"}';
16 access_log /apps/nginx/logs/access_json.log access_json;
17
18 #重启Nginx并访问测试日志格式
19 {"@timestamp": "2019-02-
22T08:55:32+08:00", "host": "192.168.7.102", "clientip": "192.168.0.1", "size": 162, "resp
onsetime": 0.000, "upstreamtime": "-", "upstreamhost": "-",
", "http_host": "www.magedu.net", "uri": "/favicon.ico", "domain": "www.magedu.net", "xff"
: "-", "referer": "-", "tcp_xff": "", "http_user_agent": "Mozilla/5.0 (Windows NT 6.1;
Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0", "status": "404"}

```

#### 4.4.3 : json格式的日志访问统计 :

```

1 #!/usr/bin/env python
2 #coding:utf-8
3 #Author:Zhang Shijie
4 status_200= []
5 status_404= []
6 with open("access_json.log") as f:
7     for line in f.readlines():
8         line = eval(line)
9         if line.get("status") == "200":
10             status_200.append(line.get)
11         elif line.get("status") == "404":
12             status_404.append(line.get)
13         else:
14             print("状态码 ERROR")
15 f.close()
16
17 print "状态码200的有--:",len(status_200)
18 print "状态码404的有--:",len(status_404)
19
20 #保存日志文件到指定路径并进测试：
21 [root@s2 ~]# python nginx_json.py
22 状态码200的有--: 1910
23 状态码404的有--: 13

```

### 4.5 : Nginx 压缩功能 :

Nginx支持对指定类型的文件进行压缩然后再传输给客户端，而且压缩还可以设置压缩比例，压缩后的文件大小将比源文件显著变小，这样有助于降低出口带宽的利用率，降低企业的IT支出，不过会占用相应的CPU资源。

Nginx对文件的压缩功能是依赖于模块ngx\_http\_gzip\_module，官方文档：[https://nginx.org/en/docs/http/ngx\\_http\\_gzip\\_module.html](https://nginx.org/en/docs/http/ngx_http_gzip_module.html)，配置指令如下：

```
1 #启用或禁用gzip压缩，默认关闭
2 gzip on | off;
3
4 #压缩比由低到高从1到9，默认为1
5 gzip_comp_level level;
6
7 #禁用IE6 gzip功能
8 gzip_disable "MSIE [1-6]\.";
9
10 #gzip压缩的最小文件，小于设置值的文件将不会压缩
11 gzip_min_length 1k;
12
13 #启用压缩功能时，协议的最小版本，默认HTTP/1.1
14 gzip_http_version 1.0 | 1.1;
15
16 #指定Nginx服务需要向服务器申请的缓存空间的个数*大小，默认32 4k|16 8k;
17 gzip_buffers number size;
18
19 #指明仅对哪些类型的资源执行压缩操作；默认为gzip_types text/html，不用显示指定，否则出错
20 gzip_types mime-type ...;
21
22 #如果启用压缩，是否在响应报文首部插入“Vary: Accept-Encoding”
23 gzip_vary on | off;
24
25 #重启nginx并进行访问测试压缩功能
26 [root@s2 pc]# cp /apps/nginx/logs/access.log /data/nginx/html/pc/test.html
27 [root@s2 pc]# echo "test1" > /data/nginx/html/pc/test1.html #小于1k的文件测试是否会压缩
28 [root@s2 pc]# vim /apps/nginx/conf/nginx.conf
29     gzip on;
30     gzip_comp_level 5;
31     gzip_min_length 1k;
32     gzip_types text/plain application/javascript application/x-javascript
text/cssapplication/xml text/javascript application/x-httpd-php image/jpeg
image/gif image/png;
33     gzip_vary on;
34
35 重启Nginx并访问测试：
36 [root@s2 pc]# curl --head --compressed http://www.magedu.net
37 HTTP/1.1 200 OK
38 Server: nginx
39 Date: Thu, 21 Feb 2019 13:06:18 GMT
40 Content-Type: text/html
41 Content-Length: 7
42 Last-Modified: Tue, 19 Feb 2019 04:09:32 GMT
43 Connection: keep-alive
44 Keep-Alive: timeout=65
45 ETag: "5c6b817c-7"
46 Accept-Ranges: bytes
47
48 [root@s2 ~]# curl --head --compressed http://www.magedu.net/test.html
```

```

49 HTTP/1.1 200 OK
50 Server: nginx
51 Date: Fri, 22 Feb 2019 01:52:23 GMT
52 Content-Type: text/html
53 Last-Modified: Thu, 21 Feb 2019 10:31:18 GMT
54 Connection: keep-alive
55 Keep-Alive: timeout=65
56 Vary: Accept-Encoding
57 ETag: W/"5c6e7df6-171109"
58 Content-Encoding: gzip #压缩传输

```

#验证不压缩访问的文件大小：

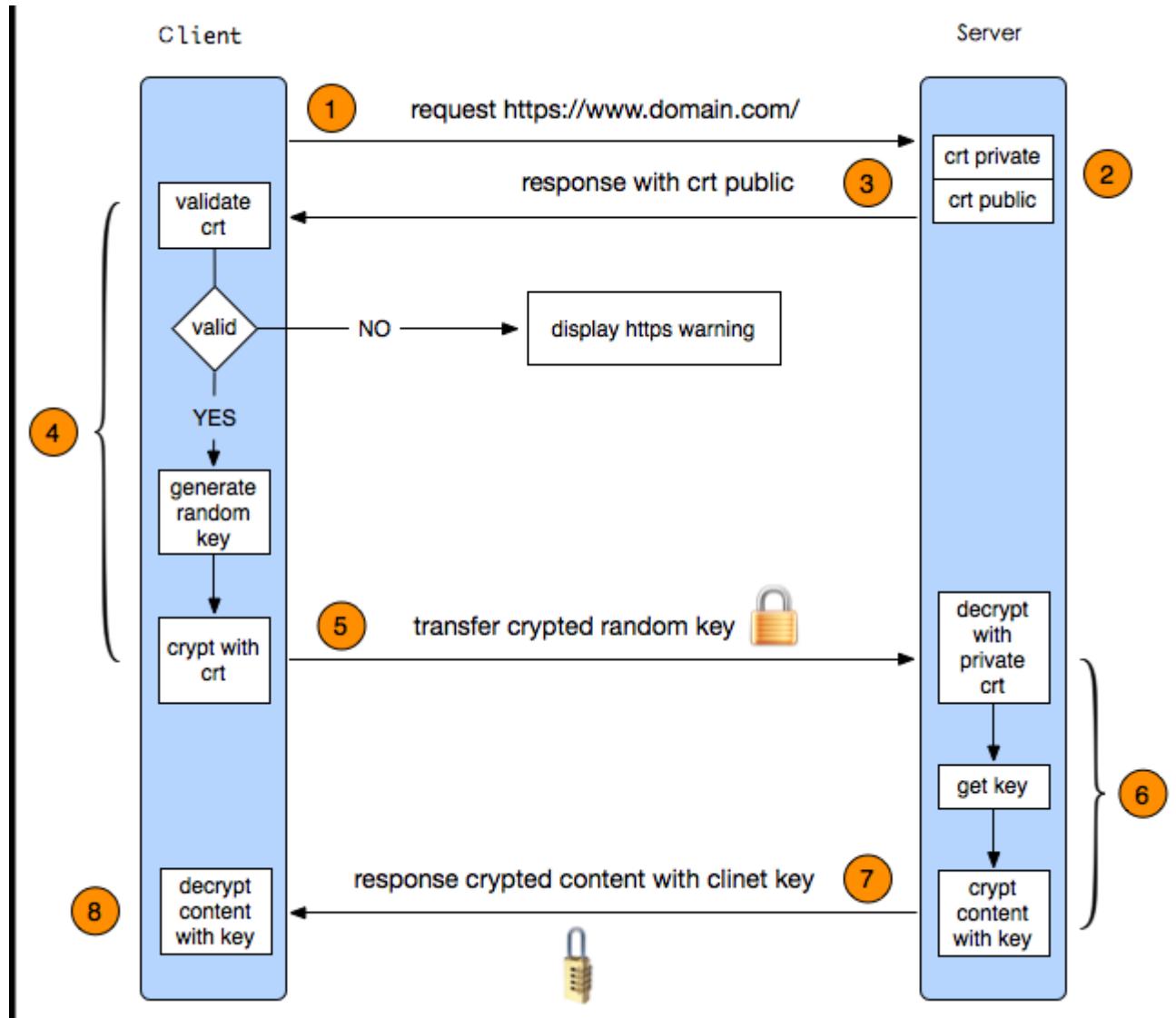
| Name        | Status | Type     | Initiator | Size   | Time   | Waterfall |
|-------------|--------|----------|-----------|--------|--------|-----------|
| access.html | 200    | document | Other     | 1.4 MB | 232 ms |           |

#验证压缩后访问的文件大小：

| Name        | Status | Type     | Initiator | Size    | Time   | Waterfall |
|-------------|--------|----------|-----------|---------|--------|-----------|
| access.html | 200    | document | Other     | 30.6 KB | 275 ms |           |

## 4.6 : https 功能 :

Web网站的登录页面都是使用https加密传输的，加密数据以保障数据的安全，HTTPS能够加密信息，以免敏感信息被第三方获取，所以很多银行网站或电子邮箱等等安全级别较高的服务都会采用HTTPS协议，HTTPS其实是有两部分组成：HTTP + SSL / TLS，也就是在HTTP上又加了一层处理加密信息的模块。服务端和客户端的信息传输都会通过TLS进行加密，所以传输的数据都是加密后的数据。



- 1 https 实现过程如下：
2. 客户端发起HTTPS请求：  
客户端访问某个web端的https地址，一般都是443端口
3. 服务端的配置：  
采用https协议的服务器必须要有一套证书，可以通过一些组织申请，也可以自己制作，目前国内很多网站都自己做的，当你访问一个网站的时候提示证书不可信任就表示证书是自己做的，证书就是一个公钥和私钥匙，就像一把锁和钥匙，正常情况下只有你的钥匙可以打开你的锁，你可以把这个送给别人让他锁住一个箱子，里面放满了钱或秘密，别人不知道里面放了什么而且别人也打不开，只有你的钥匙是可以打开的。
4. 传送证书：  
服务端给客户端传递证书，其实就是公钥，里面包含了很多信息，例如证书得到颁发机构、过期时间等等。

```
11 4.客户端解析证书：  
12 这部分工作是有客户端完成的，首先回验证公钥的有效性，比如颁发机构、过期时间等等，如果发现异常则会弹出  
13 一个警告框提示证书可能存在问题，如果证书没有问题就生成一个随机值，然后用证书对该随机值进行加密，就像  
14 2步骤所说把随机值锁起来，不让别人看到。  
15 5.传送4步骤的加密数据：  
16 就是将用证书加密后的随机值传递给服务器，目的就是为了让服务器得到这个随机值，以后客户端和服务端的通信  
17 就可以通过这个随机值进行加密解密了。  
18 6.服务端解密信息：  
19 服务端用私钥解密5步骤加密后的随机值之后，得到了客户端传过来的随机值(私钥)，然后把内容通过该值进行对  
20 称加密，对称加密就是将信息和私钥通过算法混合在一起，这样除非你知道私钥，不然无法获取其内部的内容，  
21 而正好客户端和服务端都知道这个私钥，所以只要机密算法够复杂就可以保证数据的安全性。  
22 7.传输加密后的信息：  
23 服务端将用私钥加密后的数据传递给客户端，在客户端可以被还原出原数据内容。  
24 8.客户端解密信息：  
25 客户端用之前生成的私钥获解密服务端传递过来的数据，由于数据一直是加密的，因此即使第三方获取到数据也无法  
26 知道其详细内容。
```

#### 4.6.1 : ssl 配置参数：

nginx 的https 功能基于模块ngx\_http\_ssl\_module实现，因此如果是编译安装的nginx要使用参数  
ngx\_http\_ssl\_module开启ssl功能，但是作为nginx的核心功能，yum安装的nginx默认就是开启的，编译安装的  
nginx需要指定编译参数--with-http\_ssl\_module开启，官方文档：[https://nginx.org/en/docs/http/ngx\\_http\\_ssl\\_module.html](https://nginx.org/en/docs/http/ngx_http_ssl_module.html)，配置参数如下：

```
1  ssl on | off;  
2  #为指定的虚拟主机配置是否启用ssl功能，此功能在1.15.0废弃，使用listen [ssl]替代。  
3  
4  ssl_certificate /path/to/file;  
5  #当前虚拟主机使用使用的公钥文件，一般是crt文件  
6  
7  ssl_certificate_key /path/to/file;  
8  #当前虚拟主机使用的私钥文件，一般是key文件  
9  
10 ssl_protocols [SSLv2] [SSLv3] [TLSv1] [TLSv1.1] [TLSv1.2];  
11 #支持ssl协议版本，早期为ssl现在是TSL，默认为后三个  
12  
13 ssl_session_cache off | none | [builtin[:size]] [shared:name:size];  
14 #配置ssl缓存  
15     off: 关闭缓存  
16     none: 通知客户端支持ssl session cache，但实际不支持  
17     builtin[:size]: 使用OpenSSL内建缓存，为每worker进程私有  
18     [shared:name:size]: 在各worker之间使用一个共享的缓存，需要定义一个缓存名称和缓存空间大小，  
19     一兆可以存储4000个会话信息，多个虚拟主机可以使用相同的缓存名称。  
20     ssl_session_timeout time;#客户端连接可以复用ssl session cache中缓存的有效时长，默认5m
```

#### 4.6.2 : 自签名证书：

```
1 #自签名CA证书
2 [root@s2 ~]# cd /apps/nginx/
3 [root@s2 nginx]# mkdir certs
4 [root@s2 nginx]# cd certs/
5 [root@s2 nginx]# openssl req -newkey rsa:4096 -nodes -sha256 -keyout ca.key -
x509 -days 3650 -out ca.crt #自签名CA证书
6 Generating a 4096 bit RSA private key
7 .....++
8 .....
9 Country Name (2 letter code) [XX]:CN #国家代码
10 State or Province Name (full name) []:Beijing #省份
11 Locality Name (eg, city) [Default City]:Beijing #城市名称
12 Organization Name (eg, company) [Default Company Ltd]:magedu.Ltd #公司名称
13 Organizational Unit Name (eg, section) []:magedu #部门
14 Common Name (eg, your name or your server's hostname) []:magedu.ca #通用名称
15 Email Address []:2973707860@qq.com #邮箱
16 [root@s2 certs]# ll ca.crt
17 -rw-r--r-- 1 root root 2118 Feb 22 12:10 ca.crt
18
19 #自制key和csr文件
20 [root@s2 certs]# openssl req -newkey rsa:4096 -nodes -sha256 -keyout
www.magedu.net.key -out www.magedu.net.csr
21 Generating a 4096 bit RSA private key
22 ..... ++
23 .....
24 Country Name (2 letter code) [XX]:CN
25 State or Province Name (full name) []:Beijing
26 Locality Name (eg, city) [Default City]:Beijing
27 Organization Name (eg, company) [Default Company Ltd]:magedu.net
28 Organizational Unit Name (eg, section) []:magedu.net
29 Common Name (eg, your name or your server's hostname) []:www.magedu.net
30 Email Address []:2973707860@qq.com
31
32 Please enter the following 'extra' attributes
33 to be sent with your certificate request
34 A challenge password []:
35 An optional company name []:
36 [root@s2 certs]# ll
37 total 16
38 -rw-r--r-- 1 root root 2118 Feb 22 12:10 ca.crt
39 -rw-r--r-- 1 root root 3272 Feb 22 12:10 ca.key
40 -rw-r--r-- 1 root root 1760 Feb 22 12:18 www.magedu.net.csr
41 -rw-r--r-- 1 root root 3272 Feb 22 12:18 www.magedu.net.key
42
43 #签发证书
44 [root@s2 certs]# openssl x509 -req -days 3650 -in www.magedu.net.csr -CA ca.crt -
CAkey ca.key -CAcreateserial -out www.magedu.net.crt
45 Signature ok
46 subject=/C=CN/ST=BeiJing/L=BeiJing/O=magedu.net/OU=magedu.net/CN=www.magedu.net/ema
ilAddress=2973707860@qq.com
47 Getting CA Private Key
48
49 #验证证书内容
```

```

50 [root@s2 certs]# openssl x509 -in www.magedu.net.crt -noout -text
51 Certificate:
52     Data:
53         Version: 1 (0x0)
54         Serial Number:
55             bb:76:ea:fe:f4:04:ac:06
56         Signature Algorithm: sha256WithRSAEncryption
57         Issuer: C=CN, ST=BeiJing, L=Beijing, O=magedu.Ltd, OU=magedu,
58 CN=magedu.ca/emailAddress=2973707860@qq.com
59         Validity
60             Not Before: Feb 22 06:14:03 2019 GMT
61             Not After : Feb 22 06:14:03 2020 GMT
62         Subject: C=CN, ST=BeiJing, L=Beijing, O=magedu.net, OU=magedu.net,
63 CN=www.magedu.net/emailAddress=2973707860@qq.com
64         Subject Public Key Info:
65             Public Key Algorithm: rsaEncryption
66                 Public-Key: (4096 bit)

```

### 4.6.3 : Nginx证书配置：

```

1 listen 80;
2 listen 443 ssl;
3 ssl_certificate /apps/nginx/certs/www.magedu.net.crt;
4 ssl_certificate_key /apps/nginx/certs/www.magedu.net.key;
5 ssl_session_cache shared:sslcache:20m;
6 ssl_session_timeout 10m;
7 #重启Nginx并访问验证

```

pc web

The screenshot shows the browser's address bar with the URL <https://www.magedu.net>. Below the address bar, there are standard navigation buttons (back, forward, search). The main content area displays two windows related to the certificate.

**Page Information - https://www.magedu.net/**

- 常规(G)**: Basic information about the website.
- 权限(P)**: Permissions or security settings.
- 安全(S)**: Security-related details.

**网站身份**

- 网站 : www.magedu.net
- 所有者 : 此网站未提供所有者信息。
- 验证者 : magedu.Ltd
- 过期时间 : 2020年2月22日

**隐私和历史记录**

- 我之前访问过该网站吗 ?
- 此网站在我的计算机上存储了信息吗 ?
- 我保存过该网站的任何密码吗 ?

**技术细节**

- 连接已加密 ( TLS\_ECDHE\_RSA\_WITH\_AES )
- 您当前查看的页面在传送到互联网之前已被加密使得未经授权的人难以查看计算机之间取此页面。

**证书查看器 : "www.magedu.net"**

**基本信息(G) 详细信息(D)**

**颁发给**

- 一般名称 (CN) www.magedu.net
- 组织 (O) magedu.net
- 组织单位 (OU) magedu.net
- 序列号 00:BB:76:EA:FE:F4:04:AC:06

**颁发者**

- 一般名称 (CN) magedu.ca
- 组织 (O) magedu.Ltd
- 组织单位 (OU) magedu

**有效期**

- 起始时间 2019年2月22日
- 过期时间 2020年2月22日

**指纹**

- SHA-256 指纹 66:0D:B7:3F:79:7E:44:D9:F0:56:87:70:6D:47:D7:C5:  
01:8B:08:C7:52:D2:26:6F:08:DE:D7:32:C1:3F:11:FF
- SHA1 指纹 D6:8D:E8:81:70:FB:F9:55:E1:89:6A:4E:05:98:79:3F:84:04:85:F9

### 4.6.4 : 实现多域名HTTPS :

Nginx支持基于单个IP实现多域名的功能，并且还支持单IP多域名的基础之上实现HTTPS，其实是基于Nginx的SNI ( Server Name Indication ) 功能实现，SNI是为了解决一个Nginx服务器内使用一个IP绑定多个域名和证书的功能，其具体功能是客户端在连接到服务器建立SSL链接之前先发送要访问站点的域名 ( Hostname )，这样服务器再根据这个域名返回给客户端一个合适的证书。

```
1 #制作key和csr文件
2 [root@s2 certs]# openssl req -newkey rsa:4096 -nodes -sha256 -keyout
mobile.magedu.net.key -out mobile.magedu.net.csr
3
4 Generating a 4096 bit RSA private key
5 .....
6 Country Name (2 letter code) [XX]:CN
7 State or Province Name (full name) []:Beijing
8 Locality Name (eg, city) [Default City]:Beijing
9 Organization Name (eg, company) [Default Company Ltd]:magedu
10 Organizational Unit Name (eg, section) []:magedu
11 Common Name (eg, your name or your server's hostname) []:mobile.magedu.net
12 Email Address []:2973707860@qq.com
13
14 Please enter the following 'extra' attributes
15 to be sent with your certificate request
16 A challenge password []:
17 An optional company name []:
18
19 #签名证书
20 [root@s2 certs]# openssl x509 -req -days 3650 -in mobile.magedu.net.csr -CA
ca.crt -CAkey ca.key -CAcreateserial -out mobile.magedu.net.crt
21 Signature ok
22 subject=/C=CN/ST=Beijing/L=Beijing/O=magedu/OU=magedu/CN=mobile.magedu.net/emailAdd
ress=2973707860@qq.com
23 Getting CA Private Key
24
25 #验证证书内容
26 [root@s2 certs]# openssl x509 -in mobile.magedu.net.crt -noout -text
27 Certificate:
28     Data:
29         Version: 1 (0x0)
30         Serial Number:
31             bb:76:ea:fe:f4:04:ac:07
32         Signature Algorithm: sha256WithRSAEncryption
33         Issuer: C=CN, ST=Beijing, L=Beijing, O=magedu.Ltd, OU=magedu,
CN=magedu.ca/emailAddress=2973707860@qq.com
34         Validity
35             Not Before: Feb 22 13:50:43 2019 GMT
36             Not After : Feb 19 13:50:43 2029 GMT
37         Subject: C=CN, ST=Beijing, L=Beijing, O=magedu, OU=magedu,
CN=mobile.magedu.net/emailAddress=2973707860@qq.com
38         Subject Public Key Info:
39             Public Key Algorithm: rsaEncryption
40                 Public-Key: (4096 bit)
41 ...
42
43 #Nginx 配置
```

```
44 [root@s2 certs]# cat /apps/nginx/conf/conf.d/mobile.conf
45 server {
46     listen 80 default_server;
47     server_name mobile.magedu.net;
48     rewrite ^(.*)$ https://$server_name$1 permanent;
49 }
50
51 server {
52     listen 443 ssl;
53     server_name mobile.magedu.net;
54     ssl_certificate /apps/nginx/certs/mobile.magedu.net.crt;
55     ssl_certificate_key /apps/nginx/certs/mobile.magedu.net.key;
56     ssl_session_cache shared:sslcache:20m;
57     ssl_session_timeout 10m;
58     location / {
59         root "/data/nginx/html/mobile";
60     }
61     location /mobile_status {
62         stub_status;
63         #allow 172.16.0.0/16;
64         #deny all;
65     }
66 }
```

The screenshot shows a browser window with a certificate warning and an open certificate viewer.

**Browser Warning:**

- Address bar: `https://mobile.magedu.net`
- Message: "您的连接并不安全"
- Details:
  - 报告此类错误
  - 添加安全例外
  - 永久保存此证书
- Text: "该证书因为其颁发者未知，该服务器可能未受信任。可能需要导入额外的根证书。"
- Code: 错误代码 : SEC\_ERROR\_UNKNOWN\_ISSUER

**Certificate Viewer:**

- Title: 证书查看器：“mobile.magedu.net”
- Tab: 基本信息(G) [ 详细信息(D) ]
- Message: 无法验证此证书，因为颁发者未知。
- 颁发者:**
  - 一般名称 (CN) mobile.magedu.net
  - 组织 (O) magedu
  - 组织单位 (OU) magedu
  - 序列号 00:BB:76:EA:FE:F4:04:AC:07
- 颁发者:**
  - 一般名称 (CN) magedu.ca
  - 组织 (O) magedu.Ltd
  - 组织单位 (OU) magedu
- 有效期:**
  - 起始时间 2019年2月22日
  - 过期时间 2029年2月19日
- 指纹:**
  - SHA-256 指纹 D2:E9:21:9E:50:09:15:D9:02:D4:45:2C:2D:8D:FF:3F:1A:80:83:F6:73:C8:8D:CB:99:FD:53:47:6E:C9:CE:A9
  - SHA1 指纹 52:21:4F:57:2D:B1:0F:9D:63:70:46:B0:80:97:03:49:D7:AE:13:ED

## 4.7 : 关于favicon.ico :

favicon.ico 文件是浏览器收藏网址时显示的图标，当客户端使用浏览器访问页面时，浏览器会自己主动发起请求获取页面的favicon.ico文件，但是当浏览器请求的favicon.ico文件不存在时，服务器会记录404日志，而且浏览器也会显示404报错。

解决办法：

```
1 #一：服务器不记录访问日志：
2     location = /favicon.ico {
3         #log_not_found off;
4         #access_log off;
5     }
6
7 #二：将图标保存到指定目录访问：
8     location ~ ^/favicon\.ico$ {
9         location = /favicon.ico {
10            root   /data/nginx/html/pc/images;
11        }
12    }
```

## 五：Nginx Rewrite相关功能：

Nginx服务器利用ngx\_http\_rewrite\_module 模块解析和处理rewrite请求，此功能依靠 PCRE(perl compatible regular expression)，因此编译之前要安装PCRE库，rewrite是nginx服务器的重要功能之一，用于实现URL的重写，URL的重写是非常有用的功能，比如它可以在我们改变网站结构之后，不需要客户端修改原来的书签，也无需其他网站修改我们的链接，就可以设置为访问，另外还可以在一定程度上提高网站的安全性。

### 5.1 : ngx\_http\_rewrite\_module模块指令：

[https://nginx.org/en/docs/http/ngx\\_http\\_rewrite\\_module.html](https://nginx.org/en/docs/http/ngx_http_rewrite_module.html)

#### 5.1.1 : if指令：

用于条件匹配判断，并根据条件判断结果选择不同的Nginx配置，可以配置在server或location块中进行配置，Nginx的if语法仅能使用if做单次判断，不支持使用if else或者if elif这样的多重判断，用法如下：

```
1 if ( 条件匹配 ) {
2     action
3 }
4
5 #实例-1：
6     location /main {
7         index index.html;
8         default_type text/html;
9         if ( $scheme = http ){
10            echo "if----> $scheme";
11        }
12        if ( $scheme = https ){
13            echo "if ----> $scheme";
14        }
15    }
```

```
15 | }
```

使用正则表达式对变量进行匹配，匹配成功时if指令认为条件为true，否则认为false，变量与表达式之间使用以下符号链接：

```
1 = : #比较变量和字符串是否相等，相等时if指令认为该条件为true，反之为false。
2 != : #比较变量和字符串是否不相等，不相等时if指令认为条件为true，反之为false。
3 ~ : #表示在匹配过程中区分大小写字符，(可以通过正则表达式匹配)，满足匹配条件为真，不满足为假。
4 ~* : #表示在匹配过程中不区分大小写字符，(可以通过正则表达式匹配)，满足匹配条件为真，不满足为假。
5 !~ : #区分大小写不匹配，不满足为真，满足为假，不满足为真。
6 !~* : #为不区分大小写不匹配，满足为假，不满足为真。
7
8 -f 和 ! -f: 判断请求的文件是否存在和是否不存在
9 -d 和 ! -d: #判断请求的目录是否存在和是否不存在。
10 -x 和 ! -x: #判断文件是否可执行和是否不可执行。
11 -e 和 ! -e: #判断请求的文件或目录是否存在和是否不存在(包括文件，目录，软链接)。
```

注：如果\$变量的值为空字符串或是以0开头的任意字符串，则if指令认为该条件为false，其他条件为true。

### 5.1.2 : set指令：

指定key并给其定义一个变量，变量可以调用Nginx内置变量赋值给key，另外set定义格式为set \$key \$value，及无论是key还是value都要加\$符号。

```
1 location /main {
2     root /data/nginx/html/pc;
3     index index.html;
4     default_type text/html;
5     set $name magedu;
6     echo $name;
7     set $my_port $server_port;
8     echo $my_port;
9 }
```

### 5.1.3 : break指令：

用于中断当前相同作用域(location)中的其他Nginx配置，与该指令处于同一作用域的Nginx配置中，位于它前面的配置生效，位于后面的指令配置就不再生效了，Nginx服务器在根据配置处理请求的过程中遇到该指令的时候，回到上一层作用域继续向下读取配置，该指令可以在server块和location块以及if块中使用，使用语法如下：

```
1 location /main {
2     root /data/nginx/html/pc;
3     index index.html;
4     default_type text/html;
5     set $name magedu;
6     echo $name;
7     break;
8     set $my_port $server_port;
9     echo $my_port;
10 }
```

## 5.1.4 : return指令：

从nginx版本0.8.2开始支持，return用于完成对请求的处理，并直接向客户端返回响应状态码，比如其可以指定重定向URL(对于特殊重定向状态码，301/302等)或者是指定提示文本内容(对于特殊状态码403/500等)，处于此指令后的所有配置都将不被执行，return可以在server、if和location块进行配置，用法如下：

```
1  return code; #返回给客户端指定的HTTP状态码
2  return code (text); #返回给客户端的状态码及响应体内容，可以调用变量
3  return code URL; #返回给客户端的URL地址
4
5 例如：
6  location /main {
7      root /data/nginx/html/pc;
8      default_type text/html;
9      index index.html;
10     if ( $scheme = http ){
11         #return 666;
12         #return 666 "not allow http";
13         #return 301 http://www.baidu.com;
14         return 500 "service error";
15         echo "if-----> $scheme"; #return后面的将不再执行
16     }
17     if ( $scheme = https ){
18         echo "if ----> $scheme";
19 }
```

## 5.1.5 : rewrite\_log指令：

设置是否开启记录ngx\_http\_rewrite\_module模块日志记录到error\_log日志文件当中，可以配置在http、server、location或if当中，需要日志级别为notice。

```
1  location /main {
2      index index.html;
3      default_type text/html;
4      set $name magedu;
5      echo $name;
6      rewrite_log on;
7      break;
8      set $my_port $server_port;
9      echo $my_port;
10 }
11
12 #重启nginx，访问并验证error_log：
13 [root@s2 ~]# tail -f /apps/nginx/logs/error.log
14 2019/02/27 15:10:02 [warn] 5815#0: *3 using uninitialized "my_port" variable,
  client: 192.168.0.1, server: www.magedu.net, request: "GET /main HTTP/1.1", host:
  "www.magedu.net"
```

## 5.2 : rewrite指令：

通过正则表达式的匹配来改变URI，可以同时存在一个或多个指令，按照顺序依次对URI进行匹配，rewrite主要是针对用户请求的URL或者是URI做具体处理，以下是URL和URI的具体介绍：

```
1 URI(universal resource identifier) : 通用资源标识符，标识一个资源的路径，可以不带协议。  
2 URL(uniform resource location): 统一资源定位符，是用于在Internet中描述资源的字符串，是URI的子集，主要包括传输协议(scheme)、主机(IP、端口号或者域名)和资源具体地址(目录和文件名)等三部分，一般格式为 scheme://主机名[:端口号] [/资源路径]，如：http://www.a.com:8080/path/file/index.html就是一个URL路径，URL必须带访问协议。  
3 每个URL都是一个URI，但是URI不都是URL。  
4  
5 例如：  
6 http://example.org/path/to/resource.txt #URI/URL  
7 ftp://example.org/resource.txt #URI/URL  
8 /absolute/path/to/resource.txt #URI
```

rewrite的官方介绍地址：[https://nginx.org/en/docs/http/ngx\\_http\\_rewrite\\_module.html#rewrite](https://nginx.org/en/docs/http/ngx_http_rewrite_module.html#rewrite)， rewrite可以配置在server、location、if，其具体使用方式为：

```
1 rewrite regex replacement [flag];
```

rewrite将用户请求的URI基于regex所描述的模式进行检查，匹配到时将其替换为表达式指定的新的URI。注意：如果在同一级配置块中存在多个rewrite规则，那么会自下而上逐个检查；被某条件规则替换完成后，会新一轮的替换检查，隐含有循环机制，但不超过10次；如果超过，提示500响应码，[flag]所表示的标志位用于控制此循环机制，如果替换后的URL是以http://或https://开头，则替换结果会直接以重向返回给客户端，即永久重定向301

### 5.2.1 : rewrite flag使用介绍：

利用nginx的rewrite的指令，可以实现url的重新跳转，rewrite有四种不同的flag，分别是redirect(临时重定向)、permanent(永久重定向)、break和last。其中前两种是跳转型的flag，后两种是代理型，跳转型是指有客户端浏览器重新对新地址进行请求，代理型是在WEB服务器内部实现跳转的。

Syntax: rewrite regex replacement [flag]; #通过正则表达式处理用户请求并返回替换后的数据包。 Default: —  
Context: server, location, if

```
1 redirect ;  
2 #临时重定向，重写完成后以临时重定向方式直接返回重写后生成的新URL给客户端，由客户端重新发起请求；使用  
3 相对路径，或者http://或https://开头，状态码：302  
4  
5 permanent ;  
6 #重写完成后以永久重定向方式直接返回重写后生成的新URL给客户端，由客户端重新发起请求，状态码：301  
7  
8 last ;  
9 #重写完成后停止对当前URI在当前location中后续的其它重写操作，而后对新的URL启动新一轮重写检查，不建  
10 议在location中使用  
11 break ;  
12 #重写完成后停止对当前URL在当前location中后续的其它重写操作，而后直接跳转至重写规则配置块之后的其它  
13 配置；结束循环，建议在location中使用
```

### 5.2.2 : rewrite案例-域名永久与临时重定向：

要求：因业务需要，将访问源域名 [www.magedu.net](http://www.magedu.net) 的请求永久重定向到[www.magedu.com](http://www.magedu.com)。

临时重定向不会缓存域名解析记录(A记录)，但是永久重定向会缓存。

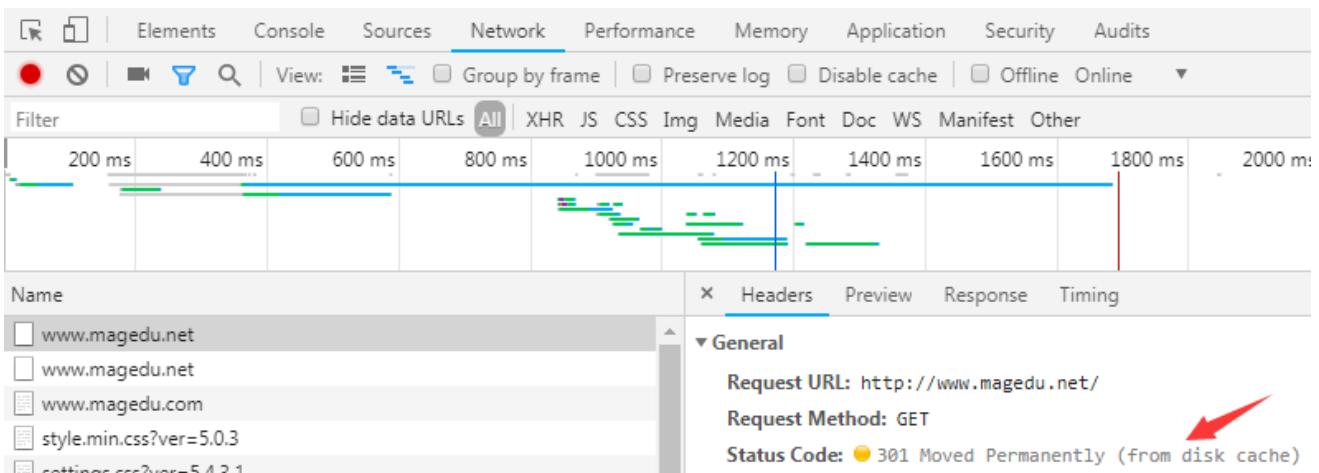
```
1 location / {  
2     root /data/nginx/html/pc;  
3     index index.html;  
4     rewrite / http://www.magedu.com permanent;  
5     #rewrite / http://www.magedu.com redirect;  
6 }  
7 #重启Nginx并访问域名www.magedu.net进行测试
```

The screenshot shows a browser window for '马哥教育官网-专业Linux云计算' with the URL 'www.magedu.com'. The page content includes the company logo and navigation links for '首页', 'Linux云计算培训', 'Python全能培训', and 'linux教程'. Below the page, the browser's developer tools Network tab is open, displaying network requests. A specific row for a 301 redirect is highlighted in blue, showing the request from 'www.magedu.com' to its homepage. Other requests listed include various static files and scripts from other domains like Baidu and JD.

| 状态  | 方法   | 域名             | 文件      | 触发源...      | 类型   | 传输       | 大小        | 耗时    |
|-----|------|----------------|---------|-------------|------|----------|-----------|-------|
| 301 | GET  | www.magedu.com | /       | document    | html | 20.28 KB | 136.54 KB | 49毫秒  |
| 200 | GET  | www.magedu.com | /       | document    | html | 20.34 KB | 136.54 KB | 167毫秒 |
| 200 | GET  | tag.baidu.com  | v.js... | script      | html | 184字节    | 0字节       | 276毫秒 |
| 502 | GET  | accwww6.53...  | se...   | script      | html | 338字节    | 166字节     | 222毫秒 |
| 200 | GET  | www6.53kf.c... | we...   | subdocum... | html | 47.91 KB | 204.98 KB |       |
| 200 | GET  | www6.53kf.c... | ifra... | subdocum... | html | 710字节    | 897字节     |       |
| 200 | GET  | www6.53kf.c... | wn...   | xhr         | html | 219字节    | 0字节       |       |
| 200 | POST | wait.53kf.com  | se...   | xhr         | html | 201字节    | 15字节      |       |
| 200 | POST | www6.53kf.c... | co...   | xhr         | html | 266字节    | 48字节      |       |
| 200 | POST | wait.53kf.com  | se...   | xhr         | html | 251字节    | 65字节      |       |
| 200 | GET  | accwww6.53...  | se...   | script      | html | 161字节    | 0字节       |       |

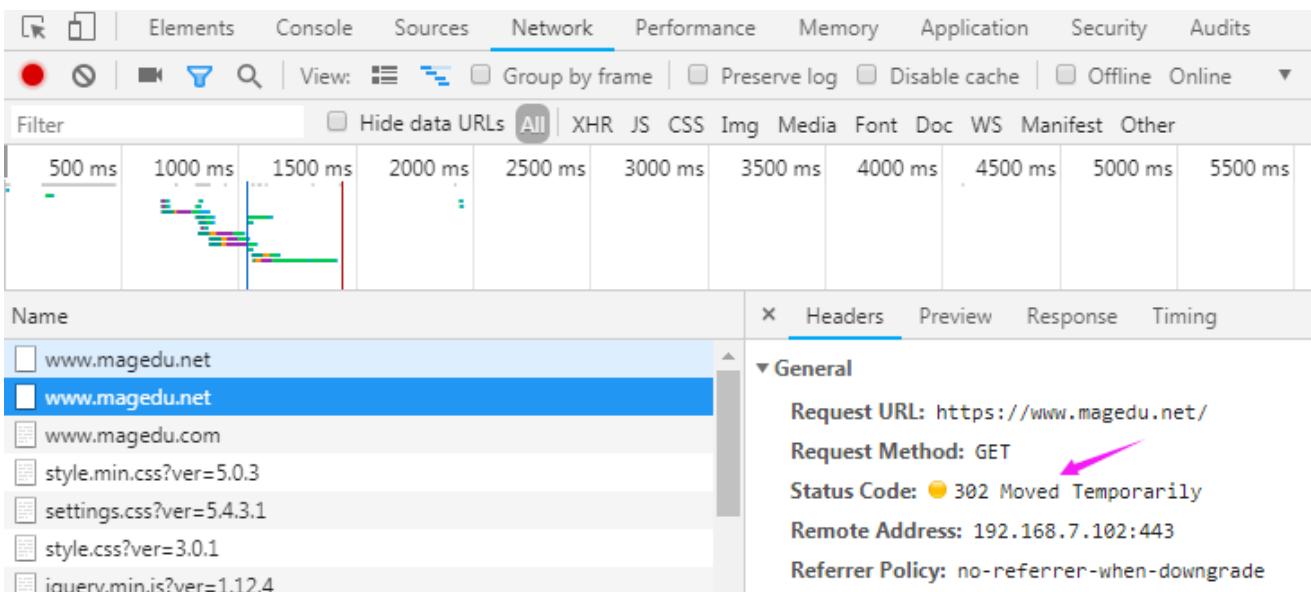
### 5.2.2.1：永久重定向：

域名永久重定向，京东早期的域名 [www.360buy.com](http://www.360buy.com) 由于与360公司类似，于是后期永久重定向到了[www.jd.com](http://www.jd.com)，永久重定向会缓存DNS解析记录。



### 5.2.2.2 : 临时重定向 :

域名临时重定向，告诉浏览器域名不是固定重定向到当前目标域名，后期可能随时会更改，因此浏览器不会缓存当前域名的解析记录，而浏览器会缓存永久重定向的DNS解析记录，这也是临时重定向与永久重定向最大的本质区别。



### 5.2.3 : rewrite案例--URI 重定向 :

要求：访问about的请求被转发至images，而访问images传递请求再次被转发至images1，以此测试last和break分别有什么区别：

#### 5.2.3.1 : last与break :

```

1 #Nginx配置：
2 [root@s2 conf.d]# vim pc.conf
3     location /break {
4         rewrite ^/break/(.*) /test$1 break;  #break不会跳转到其他的location
5         return 666 "break";
6     }
7
8     location /last {
9         rewrite ^/last/(.*) /test$1 last; #last会跳转到其他的location继续匹配新的URI

```

```
10     return 888 "last";
11 }
12
13 location /test {
14     return 999 "test";
15 }
```

重启Nginx并访问测试：

```
1 [root@s3 ~]# curl -L -i http://www.magedu.net/break/ #brak不会跳转至location /test中
2 HTTP/1.1 404 Not Found
3 Server: nginx
4 Date: Fri, 15 Mar 2019 19:31:17 GMT
5 Content-Type: text/html
6 Content-Length: 162
7 Connection: keep-alive
8 Keep-Alive: timeout=60
9
10<html>
11<head><title>404 Not Found</title></head>
12<body bgcolor="white">
13<center><h1>404 Not Found</h1></center>
14<hr><center>nginx</center>
15</body>
16</html>
17
18 [root@s3 ~]# curl -L -i http://www.magedu.net/last/ #last会跳转到location /test继续执行匹配操作
19 HTTP/1.1 999
20 Server: nginx
21 Date: Fri, 15 Mar 2019 19:34:34 GMT
22 Content-Type: application/octet-stream
23 Content-Length: 4
24 Connection: keep-alive
25 Keep-Alive: timeout=60
26
27 test
```

## 5.2.4 : rewrite案例-自动跳转https:

要求：基于通信安全考虑公司网站要求全站https，因此要求将在不影响用户请求的情况下将http请求全部自动跳转至https，另外也可以实现部分location跳转。

```
1
2 server {
3     listen 443 ssl;
4     listen 80;
5     ssl_certificate /apps/nginx/certs/www.magedu.net.crt;
6     ssl_certificate_key /apps/nginx/certs/www.magedu.net.key;
7     ssl_session_cache shared:sslcache:20m;
8     ssl_session_timeout 10m;
9     server_name www.magedu.net;
```

```
10 location / {
11     root /data/nginx/html/pc;
12     index index.html;
13     if ($scheme = http){ #未加条件判断，会导致死循环
14         rewrite / https://www.magedu.net permanent;
15     }
16 }
17 }
18 #重启Nginx并访问测试
19 [root@s3 ~]# curl -L -k -i https://www.magedu.net/
20 HTTP/1.1 200 OK
21 Server: nginx
22 Date: Fri, 15 Mar 2019 19:53:07 GMT
23 Content-Type: text/html
24 Content-Length: 7
25 Last-Modified: Thu, 14 Mar 2019 14:54:50 GMT
26 Connection: keep-alive
27 Keep-Alive: timeout=60
28 ETag: "5c8a6b3a-7"
29 Accept-Ranges: bytes
30
31 pc web
```

如果是因为规则匹配问题导致的陷入死循环，则报错如下：



## 5.2.5 : rewrite案例-判断文件是否存在：

要求：当用户访问到公司网站时输入了一个错误的URL，可以将用户重定向至官网首页。

```
1 location / {
2     root /data/nginx/html/pc;
3     index index.html;
4     if (!-f $request_filename) {
5         #return 404 "linux35";
6         rewrite (.*) http://www.magedu.net/index.html;
7     }
8 }
9 #重启Nginx并访问测试
```

## 5.3 : Nginx防盗链：

防盗链基于客户端携带的referer实现，referer是记录打开一个页面之前记录是从哪个页面跳转过来的标记信息，如果别人只链接了自己网站图片或某个单独的资源，而不是打开了网站的整个页面，这就是盗链，referer就是之前的那个网站域名，正常的referer信息有以下几种：

```
1 none : 请求报文首部没有referer首部，比如用户直接在浏览器输入域名访问web网站，就没有referer信息。
2 blocked : 请求报文有referer首部，但无有效值，比如为空。
3 server_names : referer首部中包含本主机名及即nginx 监听的server_name。
4 arbitrary_string : 自定义指定字符串，但可使用*作通配符。
5 regular expression : 被指定的正则表达式模式匹配到的字符串，要使用~开头，例如：
~.*\magedu\.com。
```

正常通过搜索引擎搜索web 网站并访问该网站的referer信息如下：

```
1 #通过搜索引擎访问web网站的referer信息：
2 ==> /apps/nginx/logs/access_json.log <==
3 {"@timestamp": "2019-02-
28T13:58:46+08:00", "host": "192.168.7.102", "clientip": "192.168.0.1", "size": 0, "responsetime": 0.000, "upstreamtime": "-", "upstreamhost": "-",
4 ", "http_host": "www.magedu.net", "uri": "/index.html", "domain": "www.magedu.net", "xff": "-",
", "referer": "https://www.baidu.com/s?ie=utf-
8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd=www.magedu.net&oq=www.mageedu.net&rsv_pq=d630606
80002eb69&rsv_t=de01TwnmyTdcJqph7sfI1hxgxLJxssfUPCQ3QkwdJk%2FLNrN95ih3xohbRs4&rqlang=
cn&rsv_enter=1&inputT=321&rsv_sug3=41&rsv_sug2=0&rsv_sug4=1626", "tcp_xff": "", "http_user_agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.36", "status": "304"}
```

### 5.3.1：实现web盗链：

在一个web 站点盗链另一个站点的资源信息，比如图片、视频等。

```
1 [root@s2 conf.d]# pwd
2 /apps/nginx/conf/conf.d
3 [root@s2 conf.d]# cat mageedu.net.conf
4 server {
5   listen 80;
6   server_name www.mageedu.net;
7
8   location / {
9     index index.html;
10    root "/data/nginx/html/mageedu";
11    access_log /apps/nginx/logs/www.mageedu.net.log access_json;
12  }
13 }
14
15 #准备盗链web页面：
16 [root@s2 conf.d]# mkdir /data/nginx/html/mageedu
17 [root@s2 conf.d]# cat /data/nginx/html/mageedu/index.html
18 <!DOCTYPE html>
19 <html lang="en">
20 <head>
21   <meta charset="UTF-8">
22   <title>盗链页面</title>
23 </head>
24 <body>
25   <a href="http://www.magedu.net">测试盗链</a>
```

```

26 
27 </body>
28 </html>
29
30 #重启Nginx并访问http://www.mageedu.net/测试
31 #验证两个域名的日志，是否会在被盗链的web站点的日志中出现以下盗链日志信息：
32 ==> /apps/nginx/logs/access_json.log <==
33 {"@timestamp":"2019-02-
28T13:27:37+08:00","host":"192.168.7.102","clientip":"192.168.0.1","size":0,"respon
setime":0.000,"upstreamtime":"-","upstreamhost":"-","http_host":"w
34 ww.magedu.net","uri":"/images/1.jpg","domain":"www.magedu.net","xff":"-",
"referer":"http://www.mageedu.net/","tcp_xff":"","http_user_agent":"Mozilla/5.0
(Windows NT 6.1; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0","status":"304"}}
35

```

```

==> /apps/nginx/logs/www.mageedu.net.log <== 盗链方, 取巧者
{"@timestamp":"2019-02-28T13:37:28+08:00","host":"192.168.7.102","clientip":"192.168.0.1","size":0,"responsetime":0.000,"upstreamtime":"-","upstreamhost":"-","http_host":"www.mageedu.net",
"uri":"/index.html","domain":"www.mageedu.net","xff":"-","referer":"-","tcp_xff":"","http_user
_agent":"Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0","statu
s":"304"}}

==> /apps/nginx/logs/access_json.log <== 被盗链方, 受害者
{"@timestamp":"2019-02-28T13:37:28+08:00","host":"192.168.7.102","clientip":"192.168.0.1","size":0,"responsetime":0.000,"upstreamtime":"-","upstreamhost":"-","http_host":"www.magedu.net",
"uri":"/images/1.jpg","domain":"www.magedu.net","xff":"-","referer":"http://www.mageedu.net/","tcp_xff":"","http_user_agent":"Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:65.0) Gecko/2010010
1 Firefox/65.0","status":"304"}

```

### 5.3.2 : 实现防盗链 :

基于访问安全考虑，nginx支持通过ngx\_http\_referer\_module模块 [https://nginx.org/en/docs/http/ngx\\_http\\_referer\\_module.html#valid\\_referers](https://nginx.org/en/docs/http/ngx_http_referer_module.html#valid_referers) 检查访问请求的referer信息是否有效实现防盗链功能，定义方式如下：

```

1 [root@s2 ~]# vim /apps/nginx/conf/conf.d/pc.conf
2 location /images {
3     root /data/nginx/html/pc;
4     index index.html;
5     valid_referers none blocked server_names
6         *.example.com example.* www.example.org/galleries/
7         ~\.\google\.;
8
9     if ($invalid_referer) {
10         return 403;
11     }

```

定义防盗链：

```

1 location ^~ /images {
2     root /data/nginx;
3     index index.html;
4     valid_referers none blocked server_names *.magedu.com www.magedu.*;
5     if ($invalid_referer) { #假如是使用其他的无效的referer访问：
6         return 403; #返回状态码403
7     }
8 }
9 #重启Nginx并访问测试

```

使用浏览器访问盗链网站 [www.mageedu.net](http://www.mageedu.net)，验证是否提前状态码403：

The screenshot shows the Network tab of the Chrome DevTools. A request for '1.jpg' from 'www.mageedu.net' has failed with a status code of 403 Forbidden. A pink arrow points to the status code in the details panel.

| Name            | Headers | Preview | Response                                                                                                                                                                                                                                | Timing |
|-----------------|---------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
| www.mageedu.net |         |         |                                                                                                                                                                                                                                         |        |
| <b>1.jpg</b>    |         |         | <p>Request URL: http://www.magedu.net/images/1.jpg<br/>         Request Method: GET<br/> <b>Status Code: 403 Forbidden</b> ←<br/>         Remote Address: 192.168.7.102:80<br/>         Referrer Policy: no-referrer-when-downgrade</p> |        |
| favicon.ico     |         |         |                                                                                                                                                                                                                                         |        |

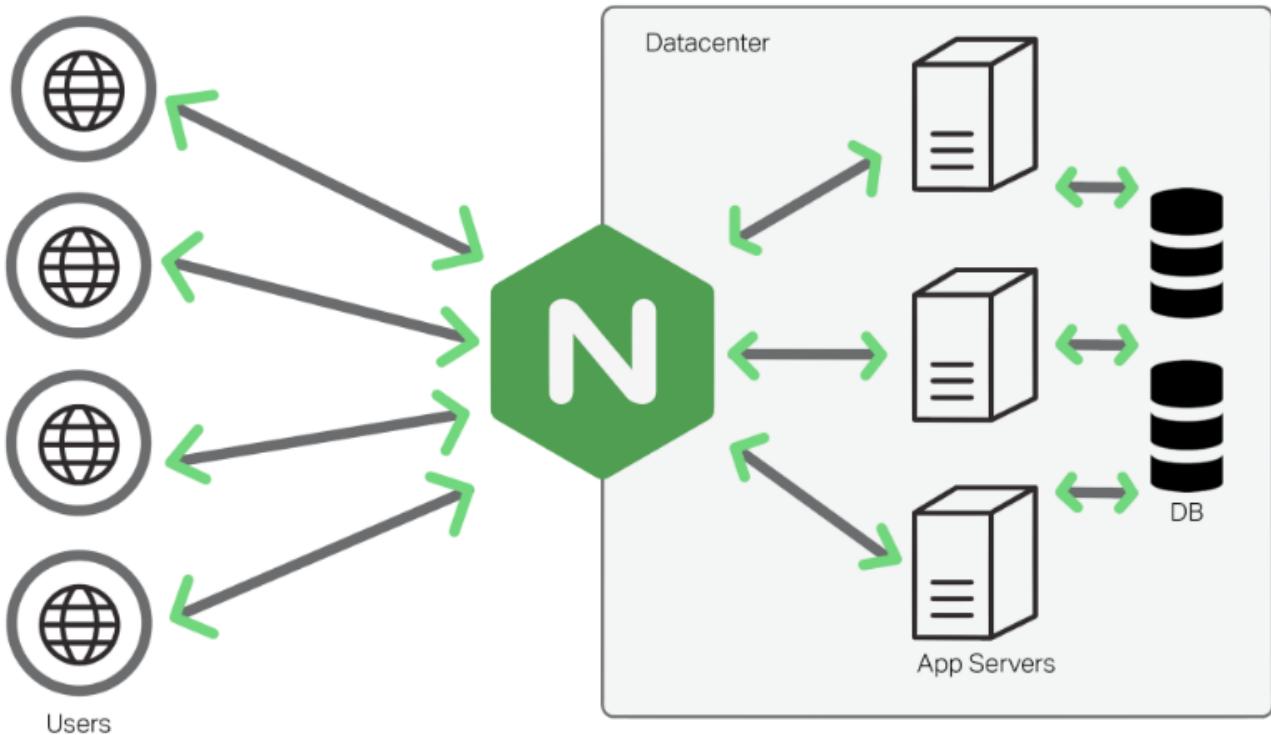
## 六：Nginx 反向代理功能：

反向代理：反向代理也叫reverse proxy，指的是代理外网用户的请求到内部的指定web服务器，并将数据返回给用户的一种方式，这是用的比较多的一种方式。

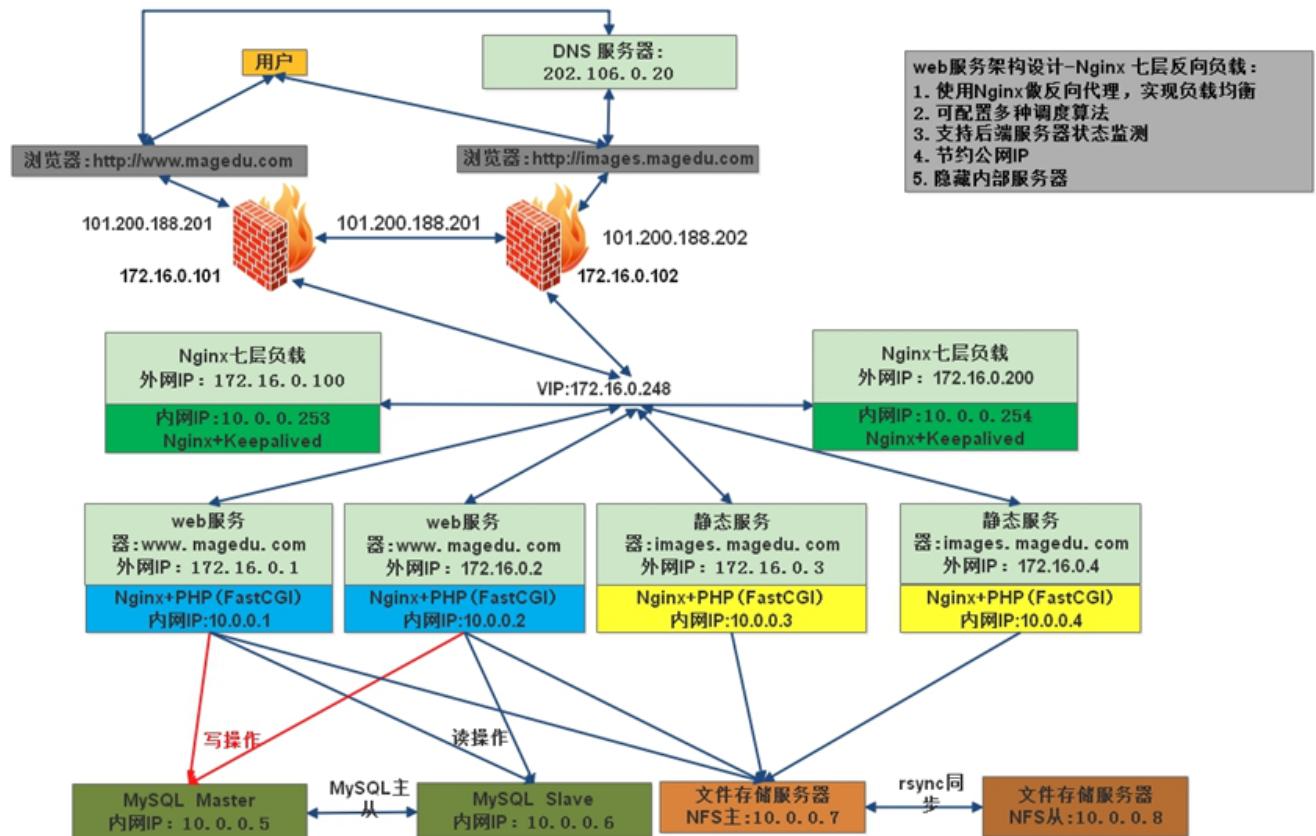
Nginx除了可以在企业提供高性能的web服务之外，另外还可以将本身不具备的请求通过某种预定义的协议转发至其它服务器处理，不同的协议就是Nginx服务器与其他服务器进行通信的一种规范，主要在不同的场景使用以下模块实现不同的功能：

- 1 `ngx_http_proxy_module`：将客户端的请求以http协议转发至指定服务器进行处理。
- 2 `ngx_stream_proxy_module`：将客户端的请求以tcp协议转发至指定服务器处理。
- 3 `ngx_http_fastcgi_module`：将客户端对php的请求以fastcgi协议转发至指定服务器助理。
- 4 `ngx_http_uwsgi_module`：将客户端对Python的请求以uwsgi协议转发至指定服务器处理。

逻辑调用关系：



生产环境部署结构：



## 6.1：实现http反向代理：

要求：将用户对域 [www.nagedu.net](https://www.nagedu.net) 的请求转发给后端服务器处理，官方文档：[https://nginx.org/en/docs/http/ngx\\_http\\_proxy\\_module.html](https://nginx.org/en/docs/http/ngx_http_proxy_module.html)，

环境准备：

```
1 192.168.7.102 #Nginx 代理服务器
2 192.168.7.103 #后端web A , Apache部署
3 192.168.7.104 #后端web B , Apache部署
```

访问逻辑图：



### 6.1.1：部署后端Apache服务器：

```
1 [root@s3 ~]# yum install httpd -y
2 [root@s3 ~]# echo "web1 192.168.7.103" > /var/www/html/index.html
3 [root@s3 ~]# systemctl start httpd && systemctl enable httpd
4
5 [root@s4 ~]# yum install httpd -y
6 [root@s4 ~]# echo "web2 192.168.7.104" >> /var/www/html/index.html
7 [root@s4 ~]# systemctl start httpd && systemctl enable httpd
8
9 #访问测试
10 [root@s4 ~]# curl http://192.168.7.103
11 web1 192.168.7.103
12 [root@s3 ~]# curl http://192.168.7.104
13 web2 192.168.7.104
```

### 6.1.2：Nginx http 反向代理入门：

官方文档：[https://nginx.org/en/docs/http/ngx\\_http\\_proxy\\_module.html#proxy\\_pass](https://nginx.org/en/docs/http/ngx_http_proxy_module.html#proxy_pass)

#### 6.2.1.1：反向代理配置参数：

```
1 proxy_pass ;
2 #用来设置将客户端请求转发给的后端服务器的主机，可以是主机名、IP地址：端口的方式，也可以代理到预先设置的主机群组，需要模块gx_http_upstream_module支持。
3 location /web {
4     index index.html;
5     proxy_pass http://192.168.7.103:80;
6     #不带斜线将访问的/web，等于访问后端服务器 http://192.168.7.103:80/web/index.html，即后端服务器配置的站点根目录要有web目录才可以被访问，这是一个追加/web到后端服务器http://servername:port/WEB/INDEX.HTML的操作
7
8     proxy_pass http://192.168.7.103:80/;
9     #带斜线，等于访问后端服务器的http://192.168.7.103:80/index.html 内容返回给客户端
10 }
11
12 #重启Nginx测试访问效果：
13 # curl -L http://www.magedu.net/web/index.html
```

```
1 proxy_hide_header ;
2 #用于nginx作为反向代理的时候，在返回给客户端http响应的时候，隐藏后端服务版本相应头部的信息，可以设置
3 在http/server或location块：
4     location /web {
5         index index.html;
6         proxy_pass http://192.168.7.103:80/;
7         proxy_hide_header ETag;
8     }
```

```
1 proxy_pass_request_body on | off ;
2 #是否向后端服务器发送HTTP包体部分，可以设置在http/server或location块，默认即为开启
```

```
1 proxy_pass_request_headers on | off ;
2 #是否将客户端的请求头部转发给后端服务器，可以设置在http/server或location块，默认即为开启
```

```
1 proxy_set_header ;
2 #可以更改或添加客户端的请求头部信息内容并转发至后端服务器，比如在后端服务器想要获取客户端的真实IP的时
3 候，就要更改每一个报文的头部，如下：
4     proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
5     #proxy_set_header HOST $remote_addr;
6     #添加HOST到报文头部，如果客户端为NAT上网那么其值为客户端的共用的公网IP地址。
```

```
1 proxy_hide_header field;
2 #用于隐藏后端服务器特定的响应首部，默认nginx在响应报文中不传递后端服务器的首部字段Date, Server, X-
3 Pad, X-Accel等
```

```
1 proxy_connect_timeout time ;
2 #配置nginx服务器与后端服务器尝试建立连接的超时时间，默认为60秒，用法如下：
3 proxy_connect_timeout 60s ;
4 #60s为自定义nginx与后端服务器建立连接的超时时间
```

```
1 proxy_read_timeout time ;
2 #配置nginx服务器向后端服务器或服务器组发起read请求后，等待的超时时间，默认60s
3 proxy_send_timeout time ;
4 #配置nginx项后端服务器或服务器组发起write请求后，等待的超时时间，默认60s
```

```
1 proxy_http_version 1.0 ;
2 #用于设置nginx提供代理服务的HTTP协议的版本，默认http 1.0
```

```
1 proxy_ignore_client_abort off ;
2 #当客户端网络中断请求时，nginx服务器中断其对后端服务器的请求。即如果此项设置为on开启，则服务器会忽略
3 客户端中断并一直等着代理服务执行返回，如果设置为off，则客户端中断后Nginx也会中断客户端请求并立即记录
4 499日志，默认为off。
```

```
1 proxy_headers_hash_bucket_size 64 ;
2 #当配置了 proxy_hide_header和proxy_set_header的时候，用于设置nginx保存HTTP报文头的hash表的上
限。
3 proxy_headers_hash_max_size 512 ;
4 #设置proxy_headers_hash_bucket_size的最大可用空间
5 server_name_hash_bucket_size 512;
6 #server_name hash表申请空间大小
7 server_names_hash_max_szie 512;
8 #设置服务器名称hash表的上限大小
```

### 6.1.2.2：反向代理示例--单台web服务器：

```
1 [root@s2 conf.d]# mv pc.conf pc.conf.bak0304
2 [root@s2 conf.d]# cp mobile.conf pc.conf
3 [root@s2 conf.d]# cat pc.conf
4 server {
5     listen 80;
6     server_name www.magedu.net;
7     location / {
8         proxy_pass http://192.168.7.103:80/;
9     }
10 }
11 #重启Nginx 并访问测试
```

### 6.1.2.3：反向代理示例--指定location：

```
1 server {
2     listen 80;
3     server_name www.magedu.net;
4     location / {
5         index index.html index.php;
6         root /data/nginx/html/pc;
7     }
8
9     location /web {
10         #proxy_pass http://192.168.7.103:80/; #注意有后面的/ ,
11         proxy_pass http://192.168.7.104:80/;
12     }
13 }
14
15 #后端web服务器必须要有相对于的访问URL
16 [root@s3 ~]# mkdir /var/www/html/web
17 [root@s3 ~]# echo "web1 page for apache" > /var/www/html/web/index.html
18 [root@s4 ~]# mkdir /var/www/html/web
19 [root@s4 ~]# echo "web2 page for apache" > /var/www/html/web/index.html
20
21 #重启Nginx并访问测试：
22 [root@s2 ~]# curl -L http://www.magedu.net/
23 pc web
24 [root@s2 ~]# curl -L http://www.magedu.net/web
25 web2 page for apache
```

```
26
27 #Apache的访问日志：
28 [root@s4 ~]# tail -f /var/log/httpd/access_log
29 192.168.7.102 - - [04/Mar/2019:18:52:00 +0800] "GET /web/ HTTP/1.1" 200 21 "-"
30 "curl/7.29.0"
31 192.168.7.102 - - [04/Mar/2019:18:52:00 +0800] "GET /web HTTP/1.0" 301 233 "-"
32 "curl/7.29.0"
33 192.168.7.102 - - [04/Mar/2019:18:52:00 +0800] "GET /web/ HTTP/1.1" 200 21 "-"
34 "curl/7.29.0"
```

### 6.1.2.4：反向代理示例--缓存功能：

缓存功能默认关闭状态

```
1 proxy_cache zone | off; 默认off
2 #指明调用的缓存，或关闭缓存机制；Context:http, server, location
```

```
1 proxy_cache_key string;
2 #缓存中用于“键”的内容，默认值：proxy_cache_key $scheme$proxy_host$request_uri;
```

```
1 proxy_cache_valid [code ...] time;
2 #定义对特定响应码的响应内容的缓存时长，定义在http{...}中
3     示例：
4         proxy_cache_valid 200 302 10m;
5         proxy_cache_valid 404 1m;
```

```
1 proxy_cache_path;
2     定义可用于proxy功能的缓存；Context:http
3         proxy_cache_path path [levels=levels] [use_temp_path=on|off]
4             keys_zone=name:size [inactive=time] [max_size=size] [manager_files=number]
5                 [manager_sleep=time] [manager_threshold=time] [loader_files=number]
6                 [loader_sleep=time] [loader_threshold=time] [purger=on|off] [purger_files=number]
7                 [purger_sleep=time] [purger_threshold=time];
8
9     示例：在http配置定义缓存信息
10        proxy_cache_path /var/cache/nginx/proxy_cache #定义缓存保存路径，proxy_cache会自动创建
11        levels=1:2:2 #定义缓存目录结构层次，1:2:2可以生成2^4x2^8x2^8=1048576个目录
12        keys_zone=proxycache:20m #指内存中缓存的大小，主要用于存放key和metadata（如：使用次数）
13        inactive=120s; #缓存有效时间
14        max_size=1g; #最大磁盘占用空间，磁盘存入文件内容的缓存空间最大值
```

```
1 #调用缓存功能，需要定义在相应的配置段，如server{...}；或者location等
2     proxy_cache proxycache;
3     proxy_cache_key $request_uri;
4     proxy_cache_valid 200 302 301 1h;
5     proxy_cache_valid any 1m;
```

```
1 proxy_cache_use_stale;
2 #在被代理的后端服务器出现哪种情况下，可直接使用过期的缓存响应客户端 ,
3 proxy_cache_use_stale error | timeout | invalid_header | updating | http_500 |
http_502 | http_503 | http_504 | http_403 | http_404 | off ; #默认是off
```

```
1 proxy_cache_methods GET | HEAD | POST ... ;
2 #对哪些客户端请求方法对应的响应进行缓存 , GET和HEAD方法总是被缓存
```

```
1 proxy_set_header field value;
2 #设定发往后端主机的请求报文的请求首部的值
3     Context: http, server, location
4     proxy_set_header X-Real-IP $remote_addr;
5     proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
6         请求报文的标准格式如下 :
7             X-Forwarded-For: client1, proxy1, proxy2
```

#### 6.1.2.4.1 : 非缓存场景压测 :

```
1 [root@s2 pc]# pwd
2 /data/nginx/html/pc
3 [root@s2 pc]# scp /apps/nginx/logs/access.log 192.168.7.104:/var/www/html/log.html
4 [root@s2 pc]# ab -n 2000 -c200 http://www.magedu.net/web/log.html
5      Total transferred:    3059318000 bytes
6      HTML transferred:   3058760000 bytes
7      Requests per second: 155.14 [#/sec] (mean)
8      Time per request:   1289.166 [ms] (mean)
9      Time per request:   6.446 [ms] (mean, across all concurrent requests)
10     Transfer rate:     231747.94 [Kbytes/sec] received
```

#### 6.1.2.4.2 : 准备缓存配置 :

```
1 [root@s2 pc]# vim /apps/nginx/conf/nginx.conf
2 proxy_cache_path /data/nginx/proxycache  levels=1:1:1  keys_zone=proxycache:20m
inactive=120s  max_size=1g; #配置在nginx.conf http配置段
3
4 [root@s2 pc]# vim /apps/nginx/conf/conf.d/pc.conf
5 location /web { #要缓存的URL 或者放在server配置项对所有URL都进行缓存
6     proxy_pass http://192.168.7.104:80/;
7     proxy_set_header clientip $remote_addr;
8     proxy_cache proxycache;
9     proxy_cache_key $request_uri;
10    proxy_cache_valid 200 302 301 1h;
11    proxy_cache_valid any 1m;
12
13 }
14 [root@s2 pc]# /apps/nginx/sbin/nginx -t
15 nginx: the configuration file /apps/nginx/conf/nginx.conf syntax is ok
16 nginx: configuration file /apps/nginx/conf/nginx.conf test is successful
17 [root@s2 pc]# systemctl start nginx
```

#### 6.2.1.4.3 : 访问并验证缓存文件 :

```
1 #访问web并验证缓存目录
2 [root@s2 pc]# curl http://www.magedu.net/web/log.html
3 [root@s2 pc]# ab -n 2000 -c200 http://www.magedu.net/web/log.html
4     Total transferred:      3059318000 bytes
5     HTML transferred:      3058760000 bytes
6     Requests per second:   1922.78 [#/sec] (mean)
7     Time per request:     104.016 [ms] (mean)
8     Time per request:     0.520 [ms] (mean, across all concurrent requests)
9     Transfer rate:        2872259.55 [Kbytes/sec] received
10
11 #验证缓存目录结构及文件大小
12 [root@s2 pc]# tree /data/nginx/proxycache/
13 /data/nginx/proxycache/
14   └── f
15     └── 0
16       └── 6
17         └── 50b643197ae7d66aaaa5e7e1961b060f
18
19 3 directories, 1 file
20 [root@s2 pc]# ll -h /data/nginx/proxycache/f/0/6/50b643197ae7d66aaaa5e7e1961b060f
21 -rw----- 1 nginx nginx 1.5M Mar  7 14:30
22 /data/nginx/proxycache/f/0/6/50b643197ae7d66aaaa5e7e1961b060f
23
24 #验证文件内容 :
25 [root@s2 pc]# head -n100
26 /data/nginx/proxycache/f/0/6/50b643197ae7d66aaaa5e7e1961b060f #会在文件首部添加相应码
27 HTTP/1.1 200 OK
28 Date: Thu, 07 Mar 2019 18:35:37 GMT
29 Server: Apache/2.4.6 (CentOS)
30 Last-Modified: Thu, 07 Mar 2019 14:14:47 GMT
31 ETag: "175624-58381ba95ac05"
32 Accept-Ranges: bytes
33 Content-Length: 1529380
34 Connection: close
35 Content-Type: text/html; charset=UTF-8
36
37 192.168.0.1 - - [18/Feb/2019:10:26:33 +0800] "GET / HTTP/1.1" 200 612
```

#### 6.1.2.5 : 添加头部报文信息 :

nginx基于模块`ngx_http_headers_module`可以实现对头部报文添加指定的key与值 , [https://nginx.org/en/docs/http/ngx\\_http\\_headers\\_module.html](https://nginx.org/en/docs/http/ngx_http_headers_module.html) ,

```
1 Syntax: add_header name value [always];
2 Default: -
3 Context: http, server, location, if in location
4
5 #添加自定义首部，如下：
6 add_header name value [always];
7     add_header x-Via $server_addr;
8     add_header x-Cache $upstream_cache_status;
9     add_header x-Accel $server_name;
10    add_trailer name value [always];
11 添加自定义响应信息的尾部， 1.13.2版后支持
```

#### 6.1.2.5.1 : Nginx配置：

```
1 location /web {
2     proxy_pass http://192.168.7.103:80/;
3     proxy_set_header clientip $remote_addr;
4     proxy_cache proxycache;
5     proxy_cache_key $request_uri;
6     proxy_cache_valid 200 302 301 1h;
7     proxy_cache_valid any 1m;
8     add_header x-Via $server_addr;
9     add_header x-Cache $upstream_cache_status;
10    add_header x-Accel $server_name;
11 }
```

#### 6.1.2.5.2 : 验证头部信息：

```
[root@s2 ~]# curl -I http://www.magedu.net/web/index.html
HTTP/1.1 200 OK
Server: magnginx
Date: Thu, 07 Mar 2019 11:16:40 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 19
Connection: keep-alive
Keep-Alive: timeout=65
Last-Modified: Thu, 28 Feb 2019 17:28:34 GMT
ETag: "13-582f79eaf4d71"
X-Via: 192.168.7.102
X-Cache: MISS ← 第一次访问没有使用缓存
X-Accel: www.magedu.net
Accept-Ranges: bytes
```

```
[root@s2 ~]# curl -I http://www.magedu.net/web/index.html
HTTP/1.1 200 OK
Server: magnginx
Date: Thu, 07 Mar 2019 11:16:45 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 19
Connection: keep-alive
Keep-Alive: timeout=65
Last-Modified: Thu, 28 Feb 2019 17:28:34 GMT
ETag: "13-582f79eaf4d71"
X-Via: 192.168.7.102
X-Cache: HIT ← 第二次访问命中缓存
X-Accel: www.magedu.net
Accept-Ranges: bytes
```

```
[root@s2 ~]# █
```

## 6.1.3 : Nginx http 反向代理高级应用 :

在上一个章节中Nginx可以将客户端的请求转发至单台后端服务器但是无法转发至特定的一组的服务器，而且不能对后端服务器提供相应的服务器状态监测，但是Nginx可以基于ngx\_http\_upstream\_module模块提供服务器分组转发、权重分配、状态监测、调度算法等高级功能，官方文档：[https://nginx.org/en/docs/http/ngx\\_http\\_upstream\\_module.html](https://nginx.org/en/docs/http/ngx_http_upstream_module.html)

### 6.1.3.1 : http upstream配置参数 :

```
1 upstream name {
2
3 }
4 #自定义一组服务器，配置在http内
```

```

1 server address [parameters];
2 #配置一个后端web服务器，配置在upstream内，至少要有一个server服务器配置。
3
4 #server支持的parameters如下：
5 weight=number #设置权重，默认为1。
6 max_conns=number #给当前server设置最大活动链接数，默认为0表示没有限制。
7 max_fails=number #对后端服务器连续监测失败多少次就标记为不可用。
8 fail_timeout=time #对后端服务器的单次监测超时时间，，默认为10秒。
9 backup #设置为备份服务器，当所有服务器不可用时将重新启用次服务器。
10 down #标记为down状态。
11 resolve #当server定义的是主机名的时候，当A记录发生变化会自动应用新IP而不用重启Nginx。

```

```

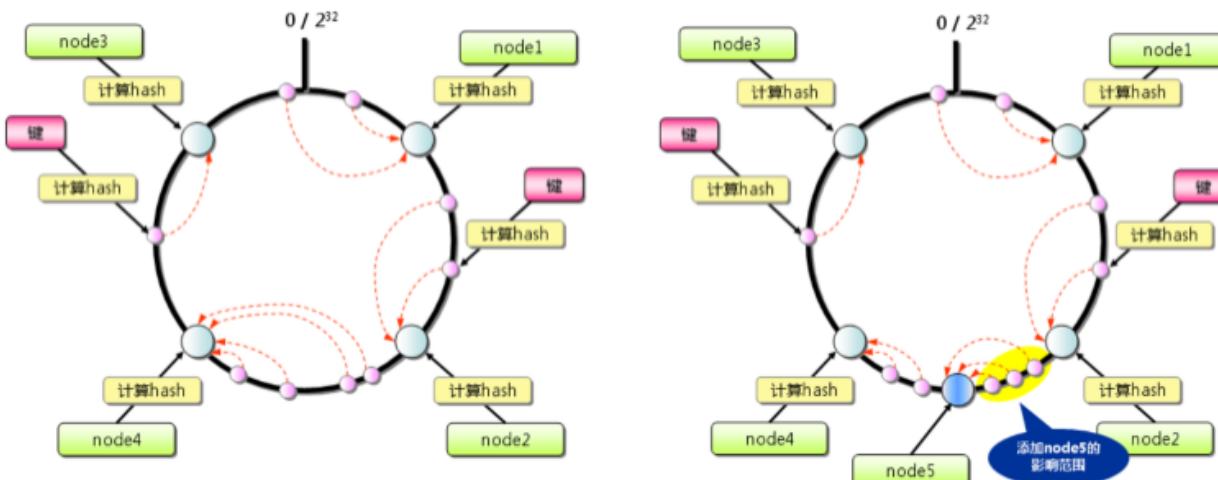
1 hash KEY consistent ;
2 #基于指定key做hash计算，使用consistent参数，将使用ketama一致性hash算法，适用于后端是Cache服务器
  (如varnish) 时使用，consistent定义使用一致性hash运算，一致性hash基于取模运算。
3
4 所谓取模运算，就是计算两个数相除之后的余数，比如 $10 \% 7 = 3$ ,  $7 \% 4 = 3$ 
5
6 hash $request_uri consistent; #基于用户请求的uri做hash

```

```

1 ip_hash ;
2 #源地址hash调度方法，基于的客户端的remote_addr(源地址)做hash计算，以实现会话保持，

```



```

1 least_conn;
2 #最少连接调度算法，优先将客户端请求调度到当前连接最少的后端服务器

```

### 6.1.3.2：反向代理示例--多台web服务器：

```

1 upstream webserver {
2   #hash $request_uri consistent;
3   #ip_hash ;
4   #least_conn;
5   server 192.168.7.103:80 weight=1 fail_timeout=5s max_fails=3; #后端服务器状态监测
6   server 192.168.7.104:80 weight=1 fail_timeout=5s max_fails=3;
7   server 192.168.7.101:80 weight=1 fail_timeout=5s max_fails=3 backup;

```

```

8 }
9
10 server {
11     listen 80;
12     server_name www.magedu.net;
13     location / {
14         index index.html index.php;
15         root /data/nginx/html/pc;
16     }
17
18     location /web {
19         index index.html;
20         proxy_pass http://webserver/;
21     }
22 }
23
24 #重启Nginx 并访问测试
25 [root@s2 ~]# curl http://www.magedu.net/web
26 web1 192.168.7.103
27 [root@s2 ~]# curl http://www.magedu.net/web
28 web2 192.168.7.104

```

关闭192.168.7.103和192.168.7.104，测试nginx backup服务器可用性：

```
[root@s4 ~]# while true;do curl http://www.magedu.net/web;sleep 1;done
```

```

web1 192.168.7.103
web2 192.168.7.104
web1 192.168.7.103
web2 192.168.7.104
web1 192.168.7.103
web1 192.168.7.103
web1 page 192.168.7.101
web1 page 192.168.7.101
web1 page 192.168.7.101
web1 page 192.168.7.101

```

### 6.1.3.3：反向代理示例--客户端IP透传：

```

1 [root@s2 conf.d]# cat pc.conf
2 upstream webserver {
3     #hash $request_uri consistent; #consistent 一致性hash算法
4     #server 192.168.7.103:80 weight=1 fail_timeout=5s max_fails=3;
5     server 192.168.7.104:80 weight=1 fail_timeout=5s max_fails=3;
6     server 192.168.7.101:80 weight=1 fail_timeout=5s max_fails=3 backup;
7 }
8
9 server {
10     listen 80;
11     server_name www.magedu.net;
12     location / {

```

```

13     index index.html index.php;
14     root /data/nginx/html/pc;
15 }
16
17 location /web {
18     index index.html;
19     proxy_pass http://webserver/;
20     proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for; #添加客户端IP到报文头
部
21 }
22 }
23 #重启nginx
24
25 #后端web服务器配置
26 1、Apache：
27 [root@s4 ~]# vim /etc/httpd/conf/httpd.conf
28 LogFormat "%{X-Forwarded-For}i %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
29 #重启apache访问web界面并验证apache日志：
30 192.168.7.104 192.168.7.102 - - [05/Mar/2019:00:36:03 +0800] "GET / HTTP/1.0" 200
31 19 "—" "curl/7.29.0"
32 192.168.0.1 192.168.7.102 - - [05/Mar/2019:00:40:46 +0800] "GET / HTTP/1.0" 200 19
33 "-" "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like
34 Gecko) Chrome/72.0.3626.119 Safari/537.36"
35
36 2、Nginx：
37 [root@s1 conf.d]# cat /apps/nginx/conf/nginx.conf
38 "$http_x_forwarded_for'" #默认日志格式就有此配置
39
40 重启nginx访问web界面并验证日志格式：
41 192.168.7.102 - - [04/Mar/2019:16:33:24 +0800] "GET // HTTP/1.0" 200 24 "-
42 "curl/7.29.0" "192.168.7.104"
43 192.168.7.102 - - [04/Mar/2019:16:40:51 +0800] "GET / HTTP/1.0" 200 24 "-
44 "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
45 Chrome/72.0.3626.119 Safari/537.36" "192.168.0.1"

```

## 6.1.4：实现动静分离：

要求：将客户端对图片的访问

画图

## 6.2：实现Nginx tcp负载均衡：

Nginx在1.9.0版本开始支持tcp模式的负载均衡，在1.9.13版本开始支持udp协议的负载，udp主要用于DNS的域名解析，其配置方式和指令和http代理类似，其基于ngx\_stream\_proxy\_module模块实现tcp负载，另外基于模块ngx\_stream\_upstream\_module实现后端服务器分组转发、权重分配、状态监测、调度算法等高级功能。

官方文档：[https://nginx.org/en/docs/stream/ngx\\_stream\\_core\\_module.html](https://nginx.org/en/docs/stream/ngx_stream_core_module.html)

### 6.2.1：tcp负载均衡配置参数：

```

1 stream { #定义stream
2     upstream backend { #定义后端服务器
3         hash $remote_addr consistent; #定义调度算法
4
5         server backend1.example.com:12345 weight=5; #定义具体server
6         server 127.0.0.1:12345      max_fails=3 fail_timeout=30s;
7         server unix:/tmp/backend3;
8     }
9
10    upstream dns { #定义后端服务器
11        server 192.168.0.1:53535; #定义具体server
12        server dns.example.com:53;
13    }
14
15    server { #定义server
16        listen 12345; #监听IP:PORT
17        proxy_connect_timeout 1s; #连接超时时间
18        proxy_timeout 3s; #转发超时时间
19        proxy_pass backend; #转发到具体服务器组
20    }
21
22    server {
23        listen 127.0.0.1:53 udp reuseport;
24        proxy_timeout 20s;
25        proxy_pass dns;
26    }
27
28    server {
29        listen [::1]:12345;
30        proxy_pass unix:/tmp/stream.socket;
31    }
32 }

```

## 6.2.2 : 负载均衡实例--Redis :

服务器安装redis

```

1 [root@s4 ~]# yum install redis -y
2 [root@s4 ~]# vim /etc/redis.conf
3 bind 0.0.0.0
4 .....
5 [root@s4 ~]# systemctl start redis
6 [root@s4 ~]# systemctl enable redis
7 [root@s4 ~]# ss -tnl | grep 6379
8 LISTEN      0      128          *:6379

```

nginx配置：

```

1 [root@s2 ~]# mkdir /apps/nginx/conf/tcp
2 [root@s2 ~]# cat /apps/nginx/conf/tcp/tcp.conf
3 stream {

```

```

4 upstream redis_server {
5     #hash $remote_addr consistent;
6     server 192.168.7.104:6379 max_fails=3 fail_timeout=30s;
7 }
8
9 server {
10    listen 192.168.7.102:6379;
11    proxy_connect_timeout 3s;
12    proxy_timeout 3s;
13    proxy_pass redis_server;
14 }
15 }
16
17 [root@s2 ~]# vim /apps/nginx/conf/nginx.conf
18 include /apps/nginx/conf/tcp/tcp.conf; #注意此处的include与http模块平级
19
20 #重启nginx并访问测试
21 [root@s2 ~]# systemctl restart nginx
22 [root@s2 ~]# ss -tnl | grep 6379
23 LISTEN      0      128      192.168.7.102:6379          *:*
24
25 #测试通过nginx 负载连接redis：
26 [root@s4 ~]# redis-cli -h 192.168.7.102
27 192.168.7.102:6379> set name jack
28 OK
29 192.168.7.102:6379> get name
30 "jack"
31 192.168.7.102:6379>

```

### 6.2.3：负载均衡实例：MySQL

服务器安装MySQL:

```

1 [root@s4 ~]# yum install mariadb mariadb-server -y
2 [root@s4 ~]# systemctl start mariadb
3 [root@s4 ~]# mysql_secure_installation #安全初始化
4 [root@s4 ~]# systemctl enable mariadb
5 [root@s4 ~]# mysql -uroot -p123456
6 MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '123456'
7 WITH GRANT OPTION;
8 MariaDB [(none)]> FLUSH PRIVILEGES;
9 Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> exit

```

nginx配置：

```

1 [root@s2 ~]# cat /apps/nginx/conf/tcp/tcp.conf
2 stream {
3     upstream redis_server {
4         server 192.168.7.104:6379 max_fails=3 fail_timeout=30s;
5     }
6

```

```

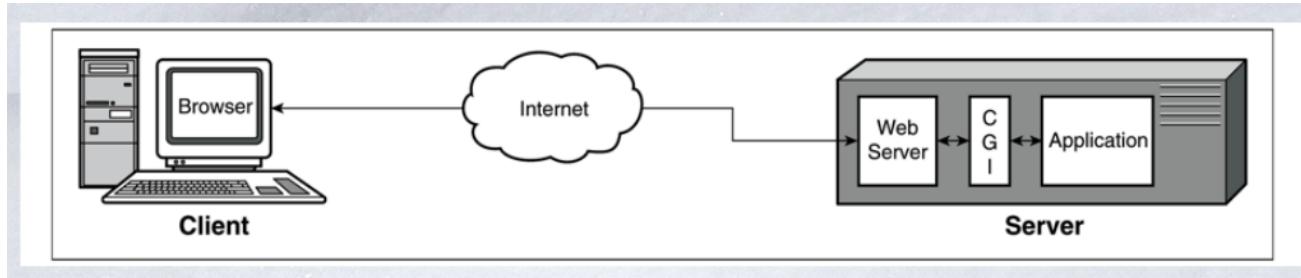
7  upstream mysql_server {
8      least_conn;
9      server 192.168.7.104:3306 max_fails=3 fail_timeout=30s;
10 }
11 #####
12 server {
13     listen 192.168.7.102:3306;
14     proxy_connect_timeout 6s;
15     proxy_timeout 15s;
16     proxy_pass mysql_server;
17 }
18
19 server {
20     listen 192.168.7.102:6379;
21     proxy_connect_timeout 3s;
22     proxy_timeout 3s;
23     proxy_pass redis_server;
24 }
25 }
26
27 #重启nginx并访问测试：
28 [root@s2 ~]# systemctl restart nginx
29
30 #测试通过nginx负载连接MySQL：
31 [root@s4 ~]# mysql -uroot -p123456 -h 192.168.7.102
32 Welcome to the MariaDB monitor. Commands end with ; or \g.
33 Your MariaDB connection id is 32
34 Server version: 5.5.60-MariaDB MariaDB Server
35
36 Copyright (c) 2000, 2018, oracle, MariaDB Corporation Ab and others.
37
38 Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
39
40 MariaDB [(none)]> create database linux34;
41 Query OK, 1 row affected (0.00 sec)
42
43 MariaDB [(none)]> show databases;
44 +-----+
45 | Database           |
46 +-----+
47 | information_schema |
48 | linux34            |
49 | mysql               |
50 | performance_schema |
51 +-----+
52 4 rows in set (0.01 sec)
53
54 MariaDB [(none)]>

```

## 6.3 : 实现FastCGI :

CGI的由来：

最早的Web服务器只能简单地响应浏览器发来的HTTP请求，并将存储在服务器上的HTML文件返回给浏览器，也就是静态html文件，但是后期随着网站功能增多网站开发也越来越复杂，以至于出现动态技术，比如像php(1995年)、java(1995)、python(1991)语言开发的网站，但是nginx/apache服务器并不能直接运行 php、java这样的文件，apache实现的方式是打补丁，但是nginx缺通过与第三方基于协议实现，即通过某种特定协议将客户端请求转发给第三方服务处理，**第三方服务器会新建新的进程处理用户的请求，处理完成后返回数据给Nginx并回收进程**，最后nginx在返回给客户端，那这个约定就是通用网关接口(common gateway interface，简称CGI)，CGI（协议）是web服务器和外部应用程序之间的接口标准，是cgi程序和web服务器之间传递信息的标准化接口。



为什么FastCGI？

CGI协议虽然解决了语言解析器和seb server之间通讯的问题，但是它的效率很低，因为web server每收到一个请求都会创建一个CGI进程，PHP解析器都会解析php.ini文件，初始化环境，请求结束的时候再关闭进程，对于每一个创建的CGI进程都会执行这些操作，所以效率很低，而FastCGI是用来提高CGI性能的，FastCGI每次处理完请求之后不会关闭掉进程，而是保留这个进程，使这个进程可以处理多个请求。这样的话每个请求都不用再重新创建一个进程了，大大提升了处理效率。

什么是PHP-FPM？ PHP-FPM(FastCGI Process Manager：FastCGI进程管理器)是一个实现了Fastcgi的程序，并且提供进程管理的功能。进程包括master进程和worker进程。master进程只有一个，负责监听端口，接受来自web server的请求。worker进程一般会有多个，每个进程中会嵌入一个PHP解析器，进行PHP代码的处理。

### 6.3.1 : FastCGI配置指令：

Nginx基于模块ngx\_http\_fastcgi\_module实现通过fastcgi协议将指定的客户端请求转发至php-fpm处理，其配置指令如下：

```
1 fastcgi_pass address;
2 #转发请求到后端服务器，address为后端的fastcgi server的地址，可用位置：location, if in
3 location
4
5 fastcgi_index name;
6 #fastcgi默认的主页资源，示例：fastcgi_index index.php;
7
8 fastcgi_param parameter value [if_not_empty];
9 #设置传递给FastCGI服务器的参数值，可以是文本，变量或组合，可用于将Nginx的内置变量赋值给自定义key
10 fastcgi_param REMOTE_ADDR      $remote_addr; #客户端源IP
11 fastcgi_param REMOTE_PORT     $remote_port; #客户端源端口
12 fastcgi_param SERVER_ADDR    $server_addr; #请求的服务器IP地址
13 fastcgi_param SERVER_PORT    $server_port; #请求的服务器端口
14 fastcgi_param SERVER_NAME    $server_name; #请求的server name
15
16 Nginx默认配置示例：
17     location ~ \.php$ {
18         root           html;
19         fastcgi_pass  127.0.0.1:9000;
```

```
19     fastcgi_index index.php;
20     fastcgi_param SCRIPT_FILENAME /scripts$fastcgi_script_name; #默认脚本路径
21     include         fastcgi_params;
22 }
```

缓存定义指令：

```
1 fastcgi_cache_path path [levels=levels] [use_temp_path=on|off] keys_zone=name:size
2   [inactive=time] [max_size=size] [manager_files=number] [manager_sleep=time]
3   [manager_threshold=time] [loader_files=number] [loader_sleep=time]
4   [loader_threshold=time] [purger=on|off] [purger_files=number] [purger_sleep=time]
5   [purger_threshold=time];
6 定义fastcgi的缓存；
7   path    #缓存位置为磁盘上的文件系统路径
8   max_size=size #磁盘path路径中用于缓存数据的缓存空间上限
9   levels=levels : 缓存目录的层级数量，以及每一级的目录数量，levels=ONE:TWO:THREE，示例：
10  levels=1:2:2
11   keys_zone=name:size #设置缓存名称及k/v映射的内存空间的名称及大小
12   inactive=time #缓存有效时间，默认10分钟，需要在指定时间满足fastcgi_cache_min_uses 次数被
13  视为活动缓存。
```

缓存调用指令：

```
1 fastcgi_cache zone | off;
2 #调用指定的缓存空间来缓存数据，可用位置：http, server, location
3
4 fastcgi_cache_key string;
5 #定义用作缓存项的key的字符串，示例：fastcgi_cache_key $request_uri;
6
7 fastcgi_cache_methods GET | HEAD | POST ...;
8 #为哪些请求方法使用缓存
9
10 fastcgi_cache_min_uses number;
11 #缓存空间中的缓存项在inactive定义的非活动时间内至少要被访问到此处所指定的次数方可被认作活动项
12
13 fastcgi_keep_conn on | off;
14 #收到后端服务器响应后，fastcgi服务器是否关闭连接，建议启用长连接
15
16 fastcgi_cache_valid [code ...] time;
17 #不同的响应码各自的缓存时长
18
19 fastcgi_hide_header field; #隐藏响应头指定信息
20 fastcgi_pass_header field; #返回响应头指定信息，默认不会将status、X-Accel-...返回
```

## 6.3.2 : FastCGI示例--Nginx与php-fpm在同一服务器：

php安装可以通过yum或者编译安装，使用yum安装相对比较简单，编译安装更方便自定义参数或选项。

### 6.3.2.1 : php环境准备：

使用base源自带的php版本

```

1 #yum安装默认版本php
2 [root@s2 ~]# yum install php-fpm php-mysql -y #默认版本
3 [root@s2 ~]# systemctl start php-fpm && systemctl enable php-fpm
4 [root@s2 ~]# ps -ef | grep php-fpm
5 root      4925      1  0 17:13 ?          00:00:00 php-fpm: master process
6 (/etc/php-fpm.conf)
7 apache    4927    4925  0 17:13 ?          00:00:00 php-fpm: pool www
8 apache    4928    4925  0 17:13 ?          00:00:00 php-fpm: pool www
9 apache    4929    4925  0 17:13 ?          00:00:00 php-fpm: pool www
10 apache   4930    4925  0 17:13 ?          00:00:00 php-fpm: pool www
11 apache   4931    4925  0 17:13 ?          00:00:00 php-fpm: pool www
12 root     4933    3235  0 17:13 pts/0        00:00:00 grep --color=auto php-fpm

```

### 6.3.2.2 : php相关配置优化 :

```

1 [root@s2 ~]# grep "^[a-z]" /etc/php-fpm.conf
2 include=/etc/php-fpm.d/*.conf
3 pid = /run/php-fpm/php-fpm.pid
4 error_log = /var/log/php-fpm/error.log
5 daemonize = yes #是否后台启动
6
7 [root@s2 ~]# cat /etc/php-fpm.d/www.conf
8 [www]
9 listen = 127.0.0.1:9000 #监听地址及IP
10 listen.allowed_clients = 127.0.0.1 #允许客户端从哪个源IP地址访问，要允许所有行首加 ;注释即可
11 user = nginx #php-fpm启动的用户和组，会涉及到后期文件的权限问题
12 group = nginx
13 pm = dynamic #动态模式进程管理
14 pm.max_children = 500 #静态方式下开启的php-fpm进程数量，在动态方式下他限定php-fpm的最大进程数
15 pm.start_servers = 100 #动态模式下初始进程数，必须大于等于pm.min_spare_servers和小于等于pm.max_children的值。
16 pm.min_spare_servers = 100 #最小空闲进程数
17 pm.max_spare_servers = 200 #最大空闲进程数
18 pm.max_requests = 500000 #进程累计请求回收值，会重启
19 pm.status_path = /pm_status #状态访问URL
20 ping.path = /ping #ping访问动地址
21 ping.response = ping-pong #ping返回值
22 slowlog = /var/log/php-fpm/www-slow.log #慢日志路径
23 php_admin_value[error_log] = /var/log/php-fpm/www-error.log #错误日志
24 php_admin_flag[log_errors] = on
25 php_value[session.save_handler] = files #phpsession保存方式及路径
26 php_value[session.save_path] = /var/lib/php/session #当时使用file保存session的文件路径

```

修改配置文件后记得重启php-fpm

```
[root@s2 ~]# systemctl restart php-fpm
```

### 6.2.3.3 : 准备php测试页面 :

```
1 [root@s2 ~]# mkdir /data/nginx/php
2 [root@s2 ~]# cat /data/nginx/php/index.php #php测试页面
3 <?php
4     phpinfo();
5 ?>
```

### 6.3.2.4 : Nginx配置转发 :

Nginx安装完成之后默认生成了与fastcgi的相关配置文件，一般保存在nginx的安装路径的conf目录当中，比如/apps/nginx/conf/fastcgi.conf、/apps/nginx/conf/fastcgi\_params。

```
1 [root@s2 ~]# vim /apps/nginx/conf/conf.d/pc.conf #在指定文件配置fastcgi
2   location ~ \.php$ {
3     root          /data/nginx/php; #$document_root调用root目录
4     fastcgi_pass  127.0.0.1:9000;
5     fastcgi_index index.php;
6     #fastcgi_param SCRIPT_FILENAME  /data/nginx/php$fastcgi_script_name;
7     fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
8     #如果SCRIPT_FILENAME是绝对路径则可以省略root /data/nginx/php;
9     include        fastcgi_params;
10    }
11  #重启Nginx并访问web测试
12  systemctl restart nginx
```

### 6.3.2.5 : 访问验证php测试页面 :

| www.magedu.net/index.php



|                                         |                                                                                       |
|-----------------------------------------|---------------------------------------------------------------------------------------|
| System                                  | Linux s2.example.com 3.10.0-862.el7.x86_64 #1 SMP Fri Apr 20 16:44:24 UTC 2018 x86_64 |
| Build Date                              | Oct 30 2018 19:32:17                                                                  |
| Server API                              | FPM/FastCGI                                                                           |
| Virtual Directory Support               | disabled                                                                              |
| Configuration File (php.ini) Path       | /etc                                                                                  |
| Loaded Configuration File               | /etc/php.ini                                                                          |
| Scan this dir for additional .ini files | /etc/php.d                                                                            |

```
1 常见的错误：
2 File not found. #路径不对
3 502：php-fpm处理超时、服务停止运行等原因导致的无法连接或请求超时
```

### 6.3.2.6 : php-fpm 的运行状态页面 :

访问配置文件里面指定的路径，会返回php-fpm的当前运行状态。

Nginx配置：

```
1 location ~ ^/(pm_status|ping)$ {
2     #access_log off;
3     #allow 127.0.0.1;
4     #deny all;
5     include fastcgi_params;
6     fastcgi_pass 127.0.0.1:9000;
7     fastcgi_param PATH_TRANSLATED $document_root$fastcgi_script_name;
8 }
```

重启Nginx并测试：

```
1 [root@s2 ~]# curl http://www.magedu.net/pm_status
2 pool:          www
3 process manager: dynamic
4 start time:    06/Mar/2019:16:07:40 +0800
5 start since:   620
6 accepted conn: 3
7 listen queue:  0
8 max listen queue: 0
9 listen queue len: 128
10 idle processes: 99
11 active processes: 1
12 total processes: 100
13 max active processes: 1
14 max children reached: 0
15 slow requests: 0
16 [root@s2 ~]# curl http://www.magedu.net/ping
17 ping-pong
18 [root@s2 ~]# curl http://www.magedu.net/pm_status?full
19 [root@s2 ~]# curl http://www.magedu.net/pm_status?html
20 [root@s2 ~]# curl http://www.magedu.net/pm_status?json
```

### 6.3.3 : FastCGI示例--Nginx与php不在同一个服务器：

nginx会处理静态请求，但是会转发动态请求到后端指定的php-fpm服务器，因此代码也需要放在后端的php-fpm服务器，即静态页面放在Nginx上而动态页面放在后端php-fpm服务器，正常情况下，一般都是采用6.3.2的部署方式。

#### 6.3.3.1 : yum安装较新版本php-fpm：

```
1 [root@s4 ~]# rpm -ivh https://mirrors.tuna.tsinghua.edu.cn/remi/enterprise/remi-release-7.rpm
2 [root@s4 ~]# yum install php56-php-fpm php56-php-mysql -y
3 验证安装路径：
4 [root@s4 ~]# rpm -q1 php56-php-fpm
5 /etc/logrotate.d/php56-php-fpm
6 /etc/systemd/system/php56-php-fpm.service.d
7 /opt/remi/php56/root/etc/php-fpm.conf
8 /opt/remi/php56/root/etc/php-fpm.d
9 /opt/remi/php56/root/etc/php-fpm.d/www.conf
10 /opt/remi/php56/root/etc/sysconfig/php-fpm
11 /opt/remi/php56/root/usr/sbin/php-fpm
12 .....
13 /usr/lib/systemd/system/php56-php-fpm.service
```

### 6.3.3.2：修改php-fpm监听配置：

php-fpm默认监听在127.0.0.1的9000端口，也就是无法远程连接，因此要做相应的修改。

```
1 #修改监听配置
2 [root@s4 ~]# vim /opt/remi/php56/root/etc/php-fpm.d/www.conf
3 39 listen = 192.168.7.104:9000 #指定监听IP
4 65 #listen.allowed_clients = 127.0.0.1 #注释运行的客户端
5
```

### 6.3.3.3：准备php测试页面：

```
1 #准备php数据目录
2 [root@s4 ~]# mkdir /data/php -p
3 [root@s4 ~]# vim /data/php/index.php
4 <?php
5     phpinfo();
6 ?>
```

### 6.3.3.4：启动并验证php-fpm：

```
1 #启动php-fpm
2 [root@s4 ~]# systemctl start php56-php-fpm
3 [root@s4 ~]# systemctl enable php56-php-fpm
4
5 #验证php-fpm进程及端口：
6 [root@s4 ~]# ps -ef | grep php-fpm
7 root      5339      1  0 18:57 ?        00:00:00 php-fpm: master process
(./opt/remi/php56/root/etc/php-fpm.conf)
8 apache    5340    5339  0 18:57 ?        00:00:00 php-fpm: pool www
9 apache    5341    5339  0 18:57 ?        00:00:00 php-fpm: pool www
10 apache   5342    5339  0 18:57 ?        00:00:00 php-fpm: pool www
11 apache   5343    5339  0 18:57 ?        00:00:00 php-fpm: pool www
12 apache   5344    5339  0 18:57 ?        00:00:00 php-fpm: pool www
13 root     5364    3235  0 18:58 pts/0    00:00:00 grep --color=auto php-fpm
14 [root@s4 ~]# ss -tnl | grep 9000
```

```

15 LISTEN      0      128    127.0.0.1:9000          *:*
16 [root@s4 ~]# netstat -tuanlp | grep 9000
17  tcp        0      0 127.0.0.1:9000    0.0.0.0:*      LISTEN      5339/php-fpm: maste

```

### 6.3.3.5 : Nginx配置转发 :

```

1 location ~ \.php$ {
2     root          /data/php;
3     fastcgi_pass 192.168.7.104:9000;
4     fastcgi_index index.php;
5     #fastcgi_param SCRIPT_FILENAME /data/php$fastcgi_script_name;
6     fastcgi_param  SCRIPT_FILENAME $document_root$fastcgi_script_name;
7     include        fastcgi_params;
8 }
9 #重启nginx
10 [root@s2 ~]# systemctl restart nginx

```

### 6.3.3.6 : 访问验证php测试页面 :

| www.magedu.net/index.php

| PHP Version 5.6.40                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>System</b>                                  | Linux s4.example.com 3.10.0-862.el7.x86_64 #1 SMP Fri Apr 20 16:44:24 UTC 2018 x86_64                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <b>Build Date</b>                              | Mar 5 2019 09:14:23                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>Server API</b>                              | FPM/FastCGI                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>Virtual Directory Support</b>               | disabled                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Configuration File (php.ini) Path</b>       | /opt/remi/php56/root/etc                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Loaded Configuration File</b>               | /opt/remi/php56/root/etc/php.ini                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Scan this dir for additional .ini files</b> | /opt/remi/php56/root/etc/php.d                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| <b>Additional .ini files parsed</b>            | /opt/remi/php56/root/etc/php.d/20-bz2.ini, /opt/remi/php56/root/etc/php.d/20-calendar.ini, /opt/remi/php56/root/etc/php.d/20-ctype.ini, /opt/remi/php56/root/etc/php.d/20-curl.ini, /opt/remi/php56/root/etc/php.d/20-exif.ini, /opt/remi/php56/root/etc/php.d/20-fileinfo.ini, /opt/remi/php56/root/etc/php.d/20-ftp.ini, /opt/remi/php56/root/etc/php.d/20-gettext.ini, /opt/remi/php56/root/etc/php.d/20-iconv.ini, /opt/remi/php56/root/etc/php.d/20-mysqlind.ini, /opt/remi/php56/root/etc/php.d/20-pdo.ini, /opt/remi/php56/root/etc/php.d/20-phar.ini, /opt/remi/php56/root/etc/php.d/20-sockets.ini, /opt/remi/php56/root/etc/php.d/20-sqlite3.ini, /opt/remi/php56/root/etc/php.d/20-tokenizer.ini, /opt/remi/php56/root/etc/php.d/30-mysqli.ini, /opt/remi/php56/root/etc/php.d/30-mysqli.ini, /opt/remi/php56/root/etc/php.d/30-pdo_sqlite.ini, /opt/remi/php56/root/etc/php.d/40-json.ini, /opt/remi/php56/root/etc/php.d/40-zip.ini |
| <b>PHP API</b>                                 | 20131106                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>PHP Extension</b>                           | 20131226                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Zend Extension</b>                          | 220131226                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

## 七 : 系统参数优化 :

### 7.1 : 系统参数优化 :

1 默认的Linux内核参数考虑的是最通用场景，不符合用于支持高并发访问的web服务器的定义，根据业务特点来进行调整，当Nginx作为静态web内容服务器、反向代理或者提供压缩服务器的服务器时，内核参数的调整都是不同的，此处针对最通用的、使Nginx支持更多并发请求的TCP网络参数做简单的配置，修改/etc/sysctl.conf来更改内核参数  
2 fs.file-max = 1000000  
3 #表示单个进程较大可以打开的句柄数

```
4 net.ipv4.tcp_tw_reuse = 1
5 #参数设置为 1 , 表示允许将TIME_WAIT状态的socket重新用于新的TCP链接 , 这对于服务器来说意义重大 , 因
6 为总有大量TIME_WAIT状态的链接存在
7
8 net.ipv4.tcp_keepalive_time = 600
9 #当keepalive启动时 , TCP发送keepalive消息的频度 ; 默认是2小时 , 将其设置为10分钟 , 可更快的清理无效
10 链接
11
12 net.ipv4.tcp_fin_timeout = 30
13 #当服务器主动关闭链接时 , socket保持在FIN_WAIT_2状态的较大时间
14
15 net.ipv4.tcp_max_tw_buckets = 5000
16 #表示操作系统允许TIME_WAIT套接字数量的较大值 , 如超过此值 , TIME_WAIT套接字将立刻被清除并打印警告信
17 息 , 默认为8000 , 过多的TIME_WAIT套接字会使web服务器变慢
18
19 net.ipv4.ip_local_port_range = 1024 65000
20 #定义UDP和TCP链接的本地端口的取值范围
21
22 net.ipv4.tcp_rmem = 10240 87380 12582912
23 #定义了TCP接受缓存的最小值、默认值、较大值
24
25 net.ipv4.tcp_wmem = 10240 87380 12582912
26 #定义TCP发送缓存的最小值、默认值、较大值
27
28 net.core.netdev_max_backlog = 8096
29 #当网卡接收数据包的速度大于内核处理速度时 , 会有一个列队保存这些数据包。这个参数表示该列队的较大值
30
31 net.core.rmem_default = 6291456
32 #表示内核套接字接受缓存区默认大小
33
34 net.core.wmem_default = 6291456
35 #表示内核套接字发送缓存区默认大小
36
37 net.core.rmem_max = 12582912
38 #表示内核套接字接受缓存区较大小
39
40 注意 : 以上的四个参数 , 需要根据业务逻辑和实际的硬件成本来综合考虑
41
42 net.ipv4.tcp_syncookies = 1
43 #与性能无关。用于解决TCP的SYN攻击
44
45 net.ipv4.tcp_max_syn_backlog = 8192
46 #这个参数表示TCP三次握手建立阶段接受SYN请求列队的较大长度 , 默认1024 , 将其设置的大一些可使出现
47 Nginx繁忙来不及accept新连接时 , Linux不至于丢失客户端发起的链接请求
48
49 net.ipv4.tcp_tw_recycle = 1
50 #这个参数用于设置启用timewait快速回收
51 net.core.somaxconn=262114
```

```
52 #选项默认值是128，这个参数用于调节系统同时发起的TCP连接数，在高并发的请求中，默认的值可能会导致链接超时或者重传，因此需要结合高并发请求数来调节此值。  
53  
54 net.ipv4.tcp_max_orphans=262114  
55 #选项用于设定系统中最多有多少个TCP套接字不被关联到任何一个用户文件句柄上。如果超过这个数字，孤立链接将立即被复位并输出警告信息。这个限制指示为了防止简单的DOS攻击，不用过分依靠这个限制甚至认为的减小这个值，更多的情况是增加这个值
```

## 八：LNMP项目实战-WordPress站点搭建：

LNMP项目实战：

L : Linux ( centos 7.6 ) [http://mirrors.cqu.edu.cn/CentOS/7.6.1810/isos/x86\\_64/](http://mirrors.cqu.edu.cn/CentOS/7.6.1810/isos/x86_64/)

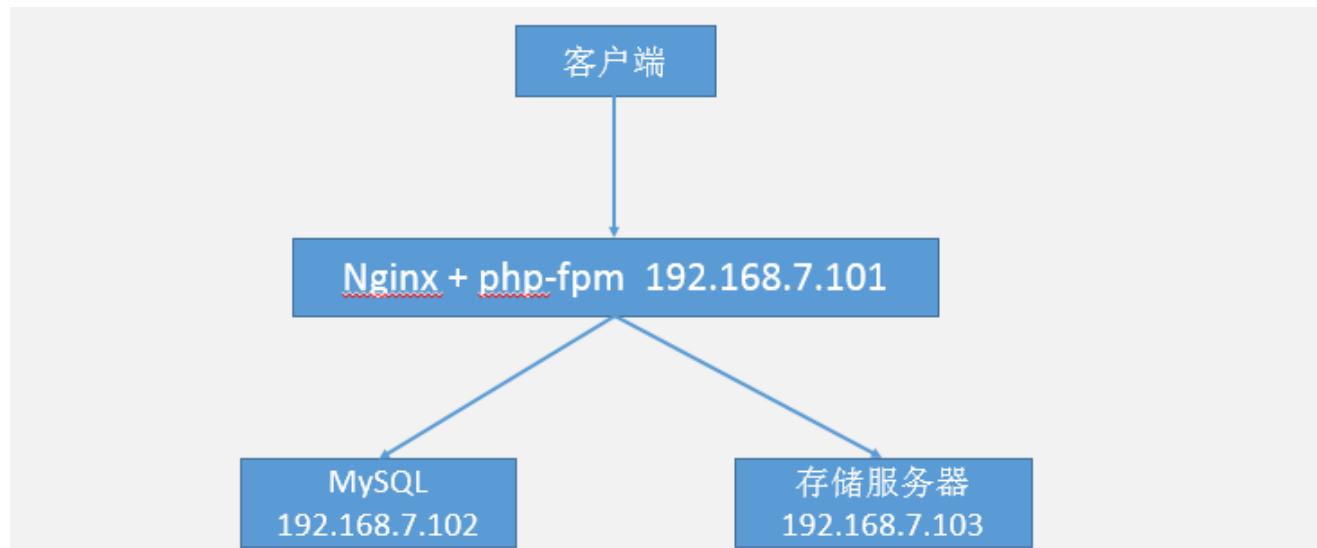
N : Nginx ( 1.12.2 ) <https://nginx.org/en/download.html>

M : MySQL ( 5.6.43 ) <https://dev.mysql.com/downloads/mysql/5.6.html#downloads>

P : PHP ( 7.2.15 ) <http://php.net/downloads.php>

Wordpress ( 5.0.3 ) : <https://cn.wordpress.org/download/>

```
1 部署规划：  
2 192.168.7.101 : Nginx  php-fpm 运行web服务  
3 192.168.7.102 : 运行MySQL数据库  
4 192.168.7.103 : NFS存储服务器，存储上传的图片
```



### 8.1：部署数据库：

#### 8.1.1：二进制部署MySQL数据库：

```
1 [root@s2 ~]# yum install vim gcc gcc-c++ wget autoconf net-tools lrzs iotop lsof  
iotop bash-completion curl policycoreutils openssh-server openssh-clients postfix  
-y  
2 [root@s2 ~]# cd /usr/local/src/
```

```

3 [root@s2 src]# tar xf mysql-5.6.43-linux-glibc2.12-x86_64.tar.gz
4 [root@s2 src]# tar xf mysql-5.6.43-linux-glibc2.12-x86_64.tar.gz
5 [root@s2 src]# ln -sv /usr/local/src/mysql-5.6.43-linux-glibc2.12-x86_64
6 /usr/local/mysql
7 '/usr/local/mysql' -> '/usr/local/src/mysql-5.6.43-linux-glibc2.12-x86_64'
8 [root@s2 mysql]# useradd mysql -s /sbin/nologin
9 [root@s2 mysql]# mkdir -pv /data/mysql /var/lib/mysql
10 [root@s2 mysql]# chown -R mysql.mysql /data /var/lib/mysql -R
11 [root@s2 mysql]# /usr/local/mysql/scripts/mysql_install_db --user=mysql --
12 datadir=/data/mysql --basedir=/usr/local/mysql
13 [root@s2 mysql]# cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld
14 [root@s2 mysql]# chmod a+x /etc/init.d/mysqld
15 [root@s2 mysql]# vim /etc/my.cnf
16 [mysqld]
17 socket=/data/mysql/mysql.sock
18 user=mysql
19 symbolic-links=0
20 datadir=/data/mysql
21 innodb_file_per_table=1
22 max_connections=10000
23
24 [client]
25 port=3306
26 socket=/var/lib/mysql/mysql.sock
27
28 [mysqld_safe]
29 log-error=/var/log/mysqld.log
pid-file=/tmp/mysql.sock

```

## 8.1.2 : 创建数据库并授权 :

```

1 #启动mysql并创建数据库
2 [root@s2 ~]# /etc/init.d/mysqld start
3 [root@s2 ~]# ln -sv /data/mysql/mysql.sock /var/lib/mysql/mysql.sock
4 [root@s2 ~]# /usr/local/mysql/bin/mysql
5 Welcome to the MySQL monitor. Commands end with ; or \g.
6 Your MySQL connection id is 2
7 Server version: 5.6.43 MySQL Community Server (GPL)
8
9 Copyright (c) 2000, 2019, oracle and/or its affiliates. All rights reserved.
10
11 Oracle is a registered trademark of Oracle Corporation and/or its
12 affiliates. Other names may be trademarks of their respective
13 owners.
14 Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
15
16 mysql> CREATE DATABASE wordpress;
17 Query OK, 1 row affected (0.00 sec)
18
19 mysql> GRANT ALL PRIVILEGES ON wordpress.* TO "wordpress"@"192.168.7.%" IDENTIFIED
BY "123456";

```

```
20 | Query OK, 0 rows affected (0.00 sec)
21 |
22 | mysql> FLUSH PRIVILEGES;
23 | Query OK, 0 rows affected (0.00 sec)
24 |
25 | mysql> FLUSH PRIVILEGES;
26 | Query OK, 0 rows affected (0.00 sec)
27 |
28 | mysql> show databases;
29 +-----+
30 | Database      |
31 +-----+
32 | information_schema |
33 | mysql          |
34 | performance_schema |
35 | test           |
36 | wordpress      |
37 +-----+
38 | 5 rows in set (0.00 sec)
```

### 8.1.3 : 验证MySQL账户权限 :

在WordPress服务器使用授权的MySQL账户远程登录测试权限

```
1 | [root@s1 etc]# mysql -uwordpress -h192.168.7.102 -p123456
2 | Welcome to the MariaDB monitor. Commands end with ; or \g.
3 | Your MySQL connection id is 5
4 | Server version: 5.6.43 MySQL Community Server (GPL)
5 |
6 | Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
7 |
8 | Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
9 |
10| MySQL [(none)]> show databases;
11+-----+
12| Database      |
13+-----+
14| information_schema |
15| test           |
16| wordpress      |
17+-----+
18| 3 rows in set (0.00 sec)
```

## 8.2 : 部署PHP :

### 8.2.1 : 编译安装php 7.2.15

```

1 [root@s1 ~]# yum -y install wget vim pcre pcre-devel openssl openssl-devel libicu-
  devel gcc gcc-c++ autoconf libjpeg libjpeg-devel libpng libpng-devel freetype
  freetype-devel libxml2 libxml2-devel zlib zlib-devel glibc glibc-devel glib2 glib2-
  devel ncurses ncurses-devel curl curl-devel krb5-devel libidn libidn-devel openldap
  openldap-devel nss_ldap jemalloc-devel cmake boost-devel bison automake libevent
  libevent-devel gd gd-devel libtool* libmcrypt libmcrypt-devel mcrypt mhash libxslt
  libxslt-devel readline readline-devel gmp gmp-devel libcurl libcurl-devel openjpeg-
  devel
2
3 [root@s1 src]# pwd
4 /usr/local/src
5 [root@s1 src]# tar xf php-7.2.15.tar.gz
6 [root@s1 src]# cd php-7.2.15
7 [root@s1 php-7.2.15]# ./configure --prefix=/apps/php --enable-fpm --with-fpm-
  user=www --with-fpm-group=www --with-pear --with-curl --with-png-dir --with-
  freetype-dir --with-iconv --with-mhash --with-zlib --with-xmlrpc --with-xsl --
  with-openssl --with-mysqli --with-pdo-mysql --disable-debug --enable-zip --enable-
  sockets --enable-soap --enable-inline-optimization --enable-xml --enable-ftp --
  enable-exif --enable-wddx --enable-bcmath --enable-calendar --enable-shmop --
  enable-dba --enable-sysvsem --enable-sysvshm --enable-sysvmsg
8 [root@s1 php-7.2.15]# make -j 2
9 [root@s1 php-7.2.15]# make install

```

## 8.2.2 : 准备PHP配置文件 :

```

1 #生成配置文件
2 [root@s1 php-7.2.15]# cd /apps/php/etc/php-fpm.d/
3 [root@s1 php-fpm.d]# cp www.conf.default www.conf
4 [root@s1 php-fpm.d]# cp /usr/local/src/php-7.2.15/php.ini-production
  /apps/php/etc/php.ini
5 [root@s1 php-fpm.d]# useradd www -s /sbin/nologin -u 1001
6 [root@s1 php-fpm.d]# grep -v ";" www.conf | grep -v "^$"
7 [www]
8 user = www
9 group = www
10 listen = 127.0.0.1:9000
11 listen.allowed_clients = 127.0.0.1
12 pm = dynamic
13 pm.max_children = 50
14 pm.start_servers = 30
15 pm.min_spare_servers = 30
16 pm.max_spare_servers = 35
17 pm.status_path = /pm_status
18 ping.path = /ping
19 ping.response = pong
20 access.log = log/$pool.access.log
21 slowlog = log/$pool.log.slow
22
23 [root@s1 etc]# mkdir /apps/php/log/ #日志文件路径
24 [root@s1 etc]# cd /apps/php/etc/
25 [root@s1 etc]# cp php-fpm.conf.default php-fpm.conf
26

```

## 8.2.3 : 启动并验证php-fpm :

```
1 #检测语法并启动php-fpm :  
2 [root@s1 etc]# /apps/php/sbin/php-fpm -t  
3 [06-Mar-2019 18:44:46] NOTICE: configuration file /apps/php/etc/php-fpm.conf test  
is successful  
4  
5 #验证php-fpm :  
6 [root@s1 etc]# /apps/php/sbin/php-fpm -c /apps/php/etc/php.ini  
7 [root@s1 etc]# ps -ef | grep php-fpm  
8 root      115708      1  0 18:45 ?          00:00:00 php-fpm: master process  
(/apps/php/etc/php-fpm.conf)  
9 www       115709 115708  0 18:45 ?          00:00:00 php-fpm: pool www  
10 www      115710 115708  0 18:45 ?          00:00:00 php-fpm: pool www  
11  
12 [root@s1 etc]# netstat -tanlp | grep php-fpm  
13 tcp      0      0 127.0.0.1:9000    0.0.0.0:*      LISTEN      115708/php-fpm: mas
```

## 8.3 : 部署Nginx:

要求自定义显示返回给客户端的server信息并隐藏nginx 版本。

### 8.3.1 : 下载nginx源码 :

```
1 [root@s1 ~]# yum install -y vim lrzsz tree screen psmisc lsof tcpdump wget ntpdate  
gcc gcc-c++ glibc glibc-devel pcre pcre-devel openssl openssl-devel systemd-devel  
net-tools iotop bc zip unzip zlib-devel bash-completion nfs-utils automake libxml2  
libxml2-devel libxml2 libxml2-devel perl perl-ExtUtils-Embed  
2 [root@s1 ~]# cd /usr/local/src/  
3 [root@s1 src]# wget https://nginx.org/download/nginx-1.12.2.tar.gz  
4 [root@s1 src]# tar xf nginx-1.12.2.tar.gz  
5 [root@s1 src]# cd nginx-1.12.2
```

### 8.3.2 : 自定义server信息 :

自定义Response Headers中server信息：

```
1 [root@s1 nginx-1.12.2]# vim src/core/nginx.h  
2 13 #define NGINX_VERSION      "1.2"  
3 14 #define NGINX_VER           "magesrv/" NGINX_VERSION #开启server_tokens显示此信息  
4  
5 [root@s1 nginx-1.12.2]# vim src/http/ngx_http_header_filter_module.c  
6 49 static u_char ngx_http_server_string[] = "Server: magnginx" CRLF; #关闭  
server_tokens显示此信息
```

### 8.3.3. : 编译安装Nginx :

```
1 [root@s1 nginx-1.12.2]# ./configure --prefix=/apps/nginx \  
2 > --user=www \
```

```
3 > --group=www \
4 > --with-http_ssl_module \
5 > --with-http_v2_module \
6 > --with-http_realip_module \
7 > --with-http_stub_status_module \
8 > --with-http_gzip_static_module \
9 > --with-pcre \
10 > --with-stream \
11 > --with-stream_ssl_module \
12 > --with-stream_realip_module
13 [root@s1 nginx-1.12.2]# make
14 [root@s1 nginx-1.12.2]# make install
```

### 8.3.4：准备php测试页：

```
1 [root@s1 ~]# mkdir /data/nginx/wordpress -p
2 [root@s1 ~]# vim /data/nginx/wordpress/index.php
3 <?php
4     phpinfo();
5 ?>
```

### 8.3.5：配置Nginx：

```
1 [root@s1 ~]# grep -v "#" /apps/nginx/conf/nginx.conf | grep -v "^\$"
2     server {
3         listen      80;
4         server_name www.magedu.net;
5         location / {
6             root   /data/nginx/wordpress;
7             index  index.php index.html index.htm;
8             if ($http_user_agent ~ "ApacheBench|WebBench|TurnitinBot|Sogou web
spider|Grid Service") {
9                 #proxy_pass http://www.baidu.com;
10                return 403;
11            }
12        }
13        location ~ \.php$ {
14            root           /data/nginx/wordpress;
15            fastcgi_pass  127.0.0.1:9000;
16            fastcgi_index index.php;
17            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
18            include       fastcgi_params;
19        }
20        error_page   500 502 503 504  /50x.html;
21        location = /50x.html {
22            root   html;
23        }
24    }
```

### 8.3.6：重启nginx并访问php状态页：

```

1 [root@s1 ~]# /apps/nginx/sbin/nginx -t
2 nginx: the configuration file /apps/nginx/conf/nginx.conf syntax is ok
3 nginx: configuration file /apps/nginx/conf/nginx.conf test is successful
4 [root@s1 ~]# /apps/nginx/sbin/nginx -s reload

```

| www.magedu.net/index.php

The screenshot shows a PHP configuration page with the title "PHP Version 7.2.15". It includes a "php" logo. Below the title, there is a table with the following data:

|                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>System</b>                            | Linux s1.example.com 3.10.0-862.el7.x86_64 #1 SMP Fri Apr 20 16:44:24 UTC 2018 x86_64                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <b>Build Date</b>                        | Mar 6 2019 18:25:34                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>Configure Command</b>                 | './configure' '--prefix=/apps/php' '--enable-fpm' '--with-fpm-user=www' '--with-fpm-group=www' '--with-pear' '--with-curl' '--with-png-dir' '--with-freetype-dir' '--with-iconv' '--with-mhash' '--with-zlib' '--with-xmlrpc' '--with-xsl' '--with-openssl' '--with-mysqli' '--with-pdo-mysql' '--disable-debug' '--enable-zip' '--enable-soap' '--enable-inline-optimization' '--enable-xml' '--enable-ftp' '--enable-exif' '--enable-wddx' '--enable-bcmath' '--enable-calendar' '--enable-shmop' '--enable-dba' '--enable-sysvsem' '--enable-sysvshm' '--enable-sysvmsg' |
| <b>Server API</b>                        | FPM/FastCGI                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <b>Virtual Directory Support</b>         | disabled                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| <b>Configuration File (php.ini) Path</b> | /apps/php/lib                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

## 8.4 : 部署WordPress :

### 8.4.1 : 部署WordPress :

```

1 [root@s1 ~]# cd /data/nginx/wordpress/
2 [root@s1 wordpress]# mv index.php /opt/
3 [root@s1 wordpress]# unzip wordpress-5.0.3-zh_CN.zip
4 [root@s1 wordpress]# mv wordpress/* .
5 [root@s1 wordpress]# mv wordpress wordpress-5.0.3-zh_CN.zip /opt/
6 [root@s1 wordpress]# cp wp-config-sample.php wp-config.php
7 [root@s1 wordpress]# vim wp-config.php
8 // ** MySQL 设置 - 具体信息来自您正在使用的主机 ** //
9 /** WordPress数据库的名称 */
10 define('DB_NAME', 'wordpress');
11
12 /** MySQL数据库用户名 */
13 define('DB_USER', 'wordpress');
14
15 /** MySQL数据库密码 */
16 define('DB_PASSWORD', '123456');
17
18 /** MySQL主机 */
19 define('DB_HOST', '192.168.7.102');
20
21 [root@s1 wordpress]# chown www.www /data/nginx/wordpress/ /apps/nginx/ -R
22 [root@s1 wordpress]# /apps/nginx/sbin/nginx -s reload
23

```

### 8.4.2 : 访问web页面 :

<http://www.magedu.net/index.php>



**欢迎**

欢迎使用著名的WordPress五分钟安装程序！请简单地填写下面的表格，来开始使用这个世界上最具扩展性、最大的个人信息发布平台。

**需要信息**

您需要填写一些基本信息。无需担心填错，这些信息以后可以再次修改。

|                                   |                                                  |
|-----------------------------------|--------------------------------------------------|
| 站点标题                              | <input type="text"/>                             |
| 用户名                               | admin                                            |
| 用户名只能含有字母、数字、空格、下划线、连字符、句号和“@”符号。 |                                                  |
| 密码                                | <input type="password"/> !CcK^iVrJ3ZKH*Xx1k<br>强 |
| 重要：您将需要此密码来登录，请将其保存在安全的位置。        |                                                  |

### 8.4.3 : 初始化配置WordPress :

初始化过程会在数据库创建表、生成管理员账户等。

## 欢迎

欢迎使用著名的WordPress五分钟安装程序！请简单地填写下面的表格，来开始使用这个世界上最具扩展性、最强大的个人信息发布平台。

## 需要信息

您需要填写一些基本信息。无需担心填错，这些信息以后可以再次修改。

站点标题

马哥教育Linux架构师

用户名

admin

用户名只能含有字母、数字、空格、下划线、连字符、句号和“@”符号。

密码

123456

 隐藏

非常弱

重要：您将需要此密码来登录，请将其保存在安全的位置。

确认密码

确认使用弱密码

您的电子邮件

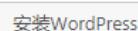
2973707860@qq.com

请仔细检查电子邮件地址后再继续。

对搜索引擎的可见性

建议搜索引擎不索引本站点

搜索引擎将本着自觉自愿的原则对待WordPress提出的请求。并不是所有搜索引擎都会遵守这类请求。

 安装WordPress

## 8.4.4：初始化完成：



成功！

WordPress安装完成。谢谢！

用户名 admin

密码 您设定的密码。

 登录

## 8.4.5：验证数据库：

```
1 mysql> use wordpress;
2 Database changed
3 mysql> show tables;
4 +-----+
5 | Tables_in_wordpress |
6 +-----+
7 | wp_commentmeta      |
8 | wp_comments         |
9 | wp_links            |
10 | wp_options          |
11 | wp_postmeta         |
12 | wp_posts             |
13 | wp_term_relationships |
14 | wp_term_taxonomy    |
15 | wp_termmeta         |
16 | wp_terms             |
17 | wp_usermeta         |
18 | wp_users             |
19 +-----+
20 12 rows in set (0.00 sec)
```

## 8.4.6 : 登录WordPress :

| www.magedu.net/wp-login.php



忘记密码 ?

← 返回到马哥教育Linux架构师

### 8.4.7：后台管理界面：

The screenshot shows the WordPress admin dashboard. The left sidebar has a dark theme with white icons and text. It includes links for Home, Updates (6), Posts, Media, Pages, Comments, Appearance, Plugins (1), Users, Tools, and Settings. A blue header bar at the top indicates an insecure connection to www.magedu.net/wp-admin/. A yellow banner at the top right says "WordPress 5.1现已可用！请现在更新。". The main content area is titled "仪表盘" (Dashboard) and features a "欢迎使用WordPress！" (Welcome to WordPress!) message with a link to "开始使用" (Get Started). Below this are two buttons: "自定义您的站点" (Customize your site) and "或更换主题" (Or change theme). To the right, there's a "接下来" (Next) section with links to "撰写您的第一篇博文" (Write your first post), "添加“关于”页面" (Add a "About" page), and "查看站点" (View site). At the bottom, there are two tabs: "概览" (Overview) which shows "1篇文章" (1 post) and "1个页面" (1 page), and "快速草稿" (Quick Draft) which shows a "标题" (Title) field.

### 8.4.8：前端访问界面：

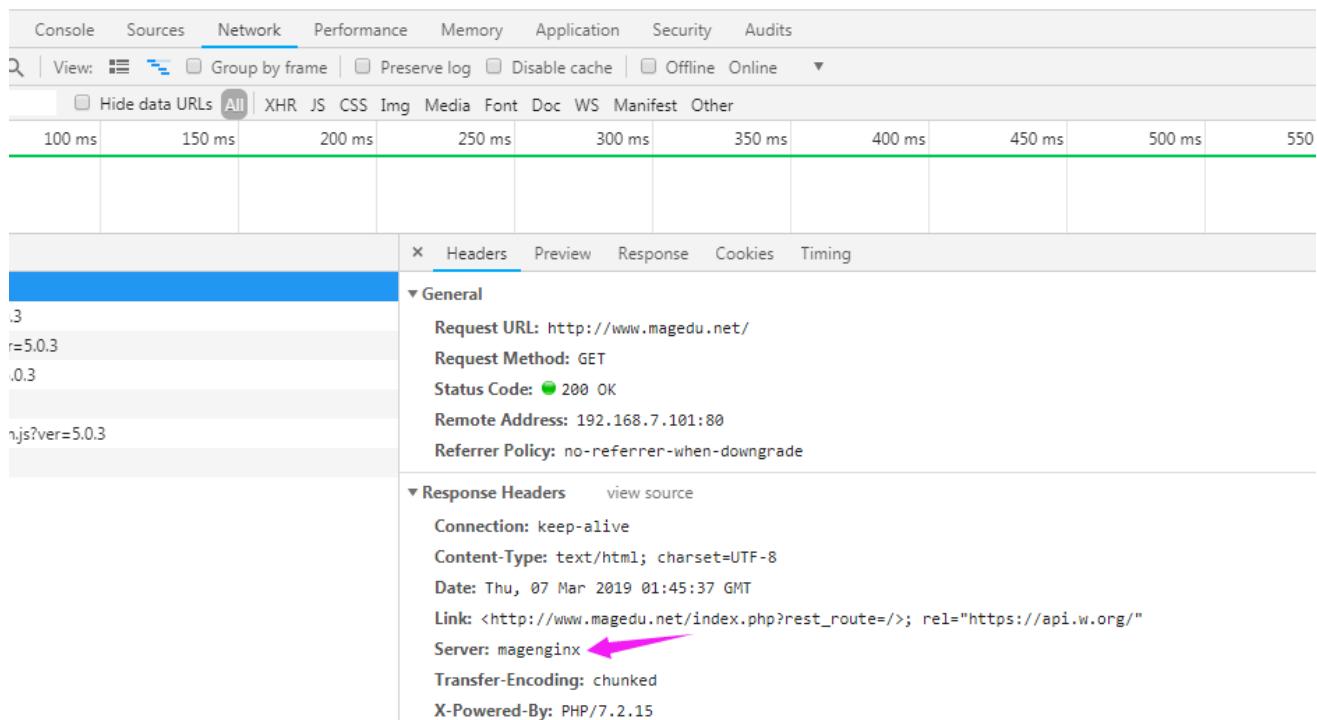
The screenshot shows the WordPress frontend. The top navigation bar shows an insecure connection to www.magedu.net. The main title is "马哥教育Linux架构师 — 又一个WordPress站点" (MaGeduo Education Linux Architecture - Another WordPress site). Below the title, a large heading says "世界，您好！" (Hello, World!). A paragraph of text below it reads "欢迎使用WordPress。这是您的第一篇文章。编辑或删除它，然后开始写作吧！" (Welcome to WordPress. This is your first article. Edit or delete it, then start writing!). At the bottom, there is a footer with author information: "admin" (作者), "2019年3月7日" (发布日期), "未分类" (Category), and "有1条评论" (Comments).

### 8.4.9：验证自定义server信息：

```
1 [root@s1 nginx-1.12.2]# vim /apps/nginx/conf/nginx.conf
2 server_tokens off; #http配置项
3 [root@s1 nginx-1.12.2]# /apps/nginx/sbin/nginx -s reload
```

① 不安全 | www.magedu.net

## 马哥教育Linux架构师 — 又一个WordPress站点



The screenshot shows the Network tab in Chrome DevTools. A request to `http://www.magedu.net/` is selected. In the Headers section, under Response Headers, the `Server` header is listed as `magenginx`, with a pink arrow pointing to it.

| Request URL:      | http://www.magedu.net/                                                         |
|-------------------|--------------------------------------------------------------------------------|
| Request Method:   | GET                                                                            |
| Status Code:      | 200 OK                                                                         |
| Remote Address:   | 192.168.7.101:80                                                               |
| Referrer Policy:  | no-referrer-when-downgrade                                                     |
| Header            | Connection: keep-alive                                                         |
|                   | Content-Type: text/html; charset=UTF-8                                         |
|                   | Date: Thu, 07 Mar 2019 01:45:37 GMT                                            |
|                   | Link: <http://www.magedu.net/index.php?rest_route=/>; rel="https://api.w.org/" |
| Server: magenginx | ← Pink arrow pointing here                                                     |
|                   | Transfer-Encoding: chunked                                                     |
|                   | X-Powered-By: PHP/7.2.15                                                       |

### 8.4.10：隐藏PHP版本：

```
1 [root@s1 nginx-1.12.2]# vim /apps/nginx/conf/nginx.conf
2 location ~ \.php$ {
3     root          /data/nginx/wordpress;
4     fastcgi_pass  127.0.0.1:9000;
5     fastcgi_index index.php;
6     fastcgi_hide_header X-Powered-By;
7     fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
8     include        fastcgi_params;
9 }
10 #重启nginx并验证是否隐藏PHP版本：
11 [root@s1 nginx-1.12.2]# /apps/nginx/sbin/nginx -t
12 nginx: the configuration file /apps/nginx/conf/nginx.conf syntax is ok
13 nginx: configuration file /apps/nginx/conf/nginx.conf test is successful
14 [root@s1 nginx-1.12.2]# /apps/nginx/sbin/nginx -s reload
```

## 马哥教育Linux架构师 — 又一个WordPress站点

Console Sources Network Performance Memory Application Security Audits

View: Group by frame Preserve log Disable cache Offline Online ▾

Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

1000 ms 1500 ms 2000 ms 2500 ms 3000 ms 3500 ms 4000 ms

|              | x | Headers                                                                        | Preview | Response | Timing |  |
|--------------|---|--------------------------------------------------------------------------------|---------|----------|--------|--|
| 3            |   | General                                                                        |         |          |        |  |
| 0.3          |   | Request URL: http://www.magedu.net/                                            |         |          |        |  |
| js?ver=5.0.3 |   | Request Method: GET                                                            |         |          |        |  |
| =5.0.3       |   | Status Code: 200 OK                                                            |         |          |        |  |
|              |   | Remote Address: 192.168.7.101:80                                               |         |          |        |  |
|              |   | Referrer Policy: no-referrer-when-downgrade                                    |         |          |        |  |
|              |   | ▼ Response Headers view source                                                 |         |          |        |  |
|              |   | Connection: keep-alive                                                         |         |          |        |  |
|              |   | Content-Type: text/html; charset=UTF-8                                         |         |          |        |  |
|              |   | Date: Thu, 07 Mar 2019 01:48:52 GMT                                            |         |          |        |  |
|              |   | Link: <http://www.magedu.net/index.php?rest_route=/>; rel="https://api.w.org/" |         |          |        |  |
|              |   | Server: magenginx                                                              |         |          |        |  |
|              |   | Transfer-Encoding: chunked                                                     |         |          |        |  |
|              |   | ▼ Request Headers view source                                                  |         |          |        |  |

确认响应头部中已经没有PHP版本信息

## 8.5 : PHP 扩展session模块redis :

<http://pecl.php.net/package-stats.php>

### 8.5.1 : 编译安装过程 :

```
1 [root@s1 ~]# yum install php-pecl-redis
2 [root@s1 ~]# cd /usr/local/src/
3 [root@s1 src]# tar xf phppredis-4.2.0.tar.gz
4 [root@s1 src]# cd phppredis-4.2.0
5
6 #生成配置文件
7 [root@s1 phppredis-4.2.0]# ll | wc -l
8 37
9 [root@s1 phppredis-4.2.0]# /apps/php/bin/phpize
10 Configuring for:
11 PHP Api Version:      20170718
12 Zend Module Api No:   20170718
13 Zend Extension Api No: 320170718
14 [root@s1 phppredis-4.2.0]# ll | wc -l
15 52
16
17 #编译安装
18 [root@s1 phppredis-4.2.0]# ./configure --with-php-config=/apps/php/bin/php-config
```

```
19   creating libtool
20   appending configuration tag "CXX" to libtool
21   configure: creating ./config.status
22   config.status: creating config.h
23   [root@s1 phppredis-4.2.0]#
24 [root@s1 phppredis-4.2.0]# make && make install
25   Build complete.
26   Don't forget to run 'make test'.
27   Installing shared extensions:      /apps/php/lib/php/extensions/no-debug-non-
28 zts-20170718/
29
30 #验证redis模块
31 [root@s1 phppredis-4.2.0]# ll /apps/php/lib/php/extensions/no-debug-non-zts-
32 20170718/
33 total 7372
34 -rwxr-xr-x 1 www www 3586572 Mar 6 18:28 opcache.a
35 -rwxr-xr-x 1 www www 1974184 Mar 6 18:28 opcache.so
36 -rwxr-xr-x 1 root root 1984768 Mar 7 10:54 redis.so
37
38 #编辑php.ini配置文件，扩展redis.so模块
39 [root@s1 phppredis-4.2.0]# vim /apps/php/etc/php.ini
40 1928 extension=/apps/php/lib/php/extensions/no-debug-non-zts-20170718/redis.so
41
42 #重启php-fpm：
43 [root@s1 phppredis-4.2.0]# pkill php-fpm
44 [root@s1 phppredis-4.2.0]# /apps/php/sbin/php-fpm -t
45 [07-Mar-2019 11:10:08] NOTICE: configuration file /apps/php/etc/php-fpm.conf test
46 is successful
47 [root@s1 phppredis-4.2.0]# /apps/php/sbin/php-fpm -c /apps/php/etc/php.ini
48
49 #通过pid文件操作php-fpm，前提是配置了php-fpm文件路径
50 [root@s1 phppredis-4.2.0]# kill -INT `cat /apps/php/var/run/php-fpm.pid`
51 [root@s1 phppredis-4.2.0]# kill -USR2 `cat /apps/php/var/run/php-fpm.pid`
```

## 8.5.2：验证加载redis模块：

准备状态页：

```
1 [root@s1 phppredis-4.2.0]# cat /data/nginx/wordpress/php-status.php
2 <?php
3   phpinfo();
4 ?>
```

| Revision                    | \$Id: 0a764bab332255746424a1e6cfbaaeebab998e4c \$ |              |
|-----------------------------|---------------------------------------------------|--------------|
| redis                       |                                                   |              |
| Redis Support               | enabled                                           |              |
| Redis Version               | 4.2.0                                             |              |
| Available serializers       | php                                               |              |
| Directive                   | Local Value                                       | Master Value |
| redis.arrays.autorehash     | 0                                                 | 0            |
| redis.arrays.connecttimeout | 0                                                 | 0            |
| redis.arrays.distributor    | no value                                          | no value     |
| redis.arrays.functions      | no value                                          | no value     |
| redis.arrays.hosts          | no value                                          | no value     |
| redis.arrays.index          | 0                                                 | 0            |
| redis.arrays.lazyconnect    | 0                                                 | 0            |
| redis.arrays.names          | no value                                          | no value     |
| redis.arrays.pconnect       | 0                                                 | 0            |
| redis.arrays.previous       | no value                                          | no value     |

### 8.5.3 : 将session写入redis :

```

1 [root@s1 phppredis-4.2.0]# yum install redis
2 [root@s1 phppredis-4.2.0]# systemctl start redis
3 [root@s1 phppredis-4.2.0]# systemctl enable redis

```

### 8.5.4 : 配置php.ini :

```

1 1328 [Session]
2 1331 session.save_handler = redis
3 1399 session.save_path = "tcp://127.0.0.1:6379"
4 1400 ;session.save_path = "tcp://IP:6379?auth=PASSWORD"
5
6 #重启php-fpm
7 [root@s1 phppredis-4.2.0]# /apps/php/sbin/php-fpm -t
8 [root@s1 phppredis-4.2.0]# kill -USR2 `cat /apps/php/var/run/php-fpm.pid`

```

### 8.5.5 : 准备session写入web页面 :

```

1 [root@s1 php-fpm.d]# cat /data/nginx/wordpress/session.php
2 <?php
3 echo "test session page?";
4 session_start();
5 $_SESSION['a'] = 'magedu test php write session to redis';
6 ?>

```

### 8.5.6 : 访问web页面 :

← → C ⌂ ⓘ 不安全 | www.magedu.net/session.php

test session page

## 8.5.7 : redis验证session数据 :

```
1 [root@s1 phpcache-4.2.0]# redis-cli  
2 127.0.0.1:6379> KEYS *  
3 1) "PHPREDIS_SESSION:q12hvhqocuh7g3o4sr7o490i18"  
4 127.0.0.1:6379> get PHPREDIS_SESSION:q12hvhqocuh7g3o4sr7o490i18  
5 "a|s:38:\"magedu test php write session to redis\";"  
6 127.0.0.1:6379>
```