目录

[一、什么是Ngnix: 3](#_Toc45739604)

[nginx的作用 3](#_Toc45739605)

[nginx模块 8](#_Toc45739606)

[nginx的进程模型 9](#_Toc45739607)

[配置服务器名称 11](#_Toc45739608)

[配置 HTTPS 服务器 13](#_Toc45739609)

[作为 HTTP 负载均衡器 14](#_Toc45739610)

[二、Ngnix Windows下安装和配置: 19](#_Toc45739611)

[1)下载地址 19](#_Toc45739612)

[2)启动 19](#_Toc45739613)

[3)使用 19](#_Toc45739614)

[4)进程 20](#_Toc45739615)

[5)配置 21](#_Toc45739616)

[6)服务器集群-负载均衡 23](#_Toc45739617)

[7)session共享 26](#_Toc45739618)

[8)上传文件大小限制(413 Request Entity Too Large) 26](#_Toc45739619)

[9)可以反向代理外网地址 27](#_Toc45739620)

[10)TCP反向代理 27](#_Toc45739621)

[11)root访问静态文件 30](#_Toc45739622)

[12) nginx默认首页是404或者自定义页面 31](#_Toc45739623)

[13)指定配置文件启动 31](#_Toc45739624)

[14)查看当前nginx进程 32](#_Toc45739625)

[15)重启 32](#_Toc45739626)

[16) Nginx错误页面隐藏版本号 33](#_Toc45739627)

[17) 检查Nginx配置文件是否正确 34](#_Toc45739628)

[18) 防爬虫策略 34](#_Toc45739629)

[https://blog.csdn.net/shangrila\_kun/article/details/89501343 34](#_Toc45739630)

[三、Ngnix Linux下安装和配置 36](#_Toc45739631)

[四、补充 37](#_Toc45739632)

[reload和restart的区别 39](#_Toc45739633)

[五、常见问题解答 40](#_Toc45739634)

[swap file “\*.swp”already exists！ 40](#_Toc45739635)

[PID file /var/run/nginx.pid not readable 40](#_Toc45739636)

[nginx: [emerg] "server" directive is not allowed here in /etc/nginx/conf.d/proxy.conf:142 41](#_Toc45739637)

[nginx服务启动，页面无法访问 42](#_Toc45739638)

[windows下nginx一闪而过，无法启动 43](#_Toc45739639)

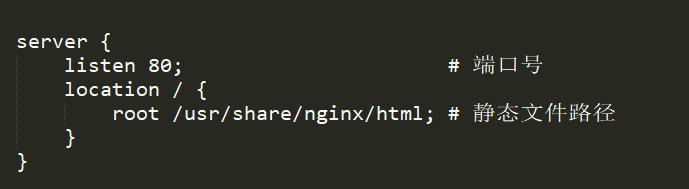
[windows下openssl提示丢失MSVCR120.dll 43](#_Toc45739640)

# 一、什么是Ngnix:

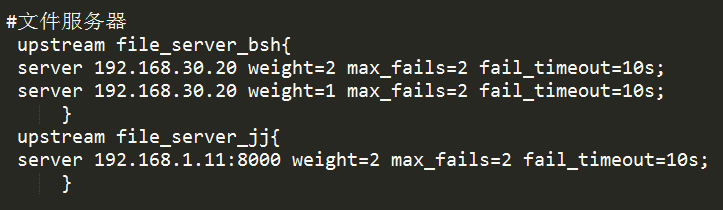
## nginx的作用

nginx是由俄罗斯人开发的一款高性能的http和反向代理服务器，也可以用来作为邮件代理。相比较于其他的服务器，具有占用内存少，稳定性高等优势。

首先，Nginx是一个HTTP服务器，可以将服务器上的静态文件（如HTML、图片）通过HTTP协议展现给客户端，配置：



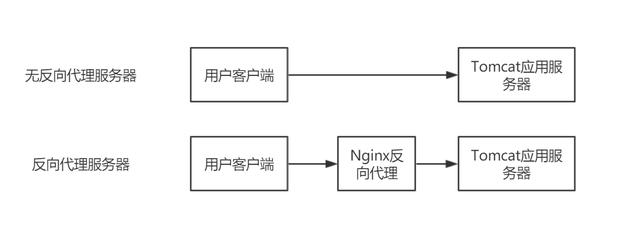
也可以设计独立的文件服务器：



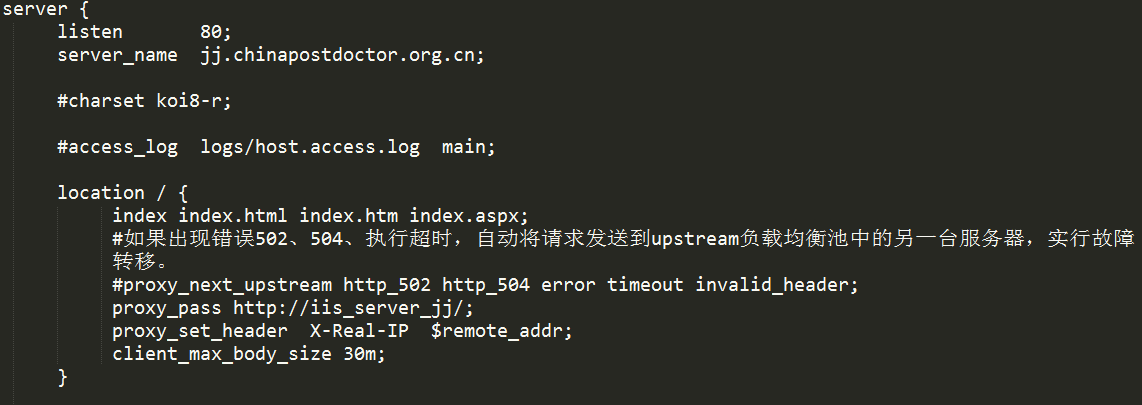




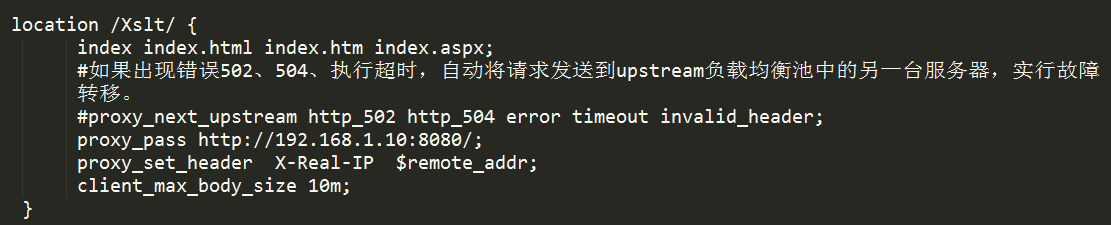
其次，Nginx是一个反向代理服务器，客户端本来可以直接通过HTTP协议访问某网站应用服务器，如果网站管理员在中间加上一个Nginx，客户端请求Nginx，Nginx请求应用服务器，然后将结果返回给客户端，此时Nginx就是反向代理服务器。



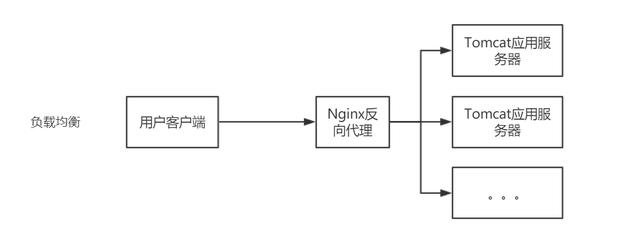
反向代理



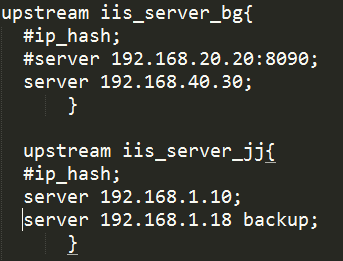
反向代理配置

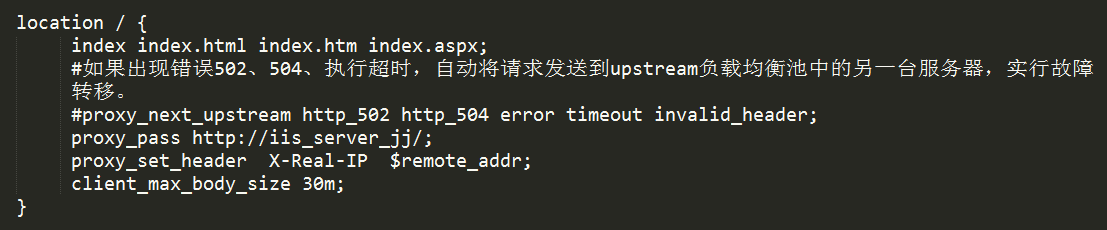


Nginx也可以实现负载均衡。当网站访问量非常大，网站站长开心赚钱的同时，也摊上事儿了。因为网站越来越慢，一台服务器已经不够用了。于是将相同的应用部署在多台服务器上，将大量用户的请求分配给多台机器处理。同时带来的好处是，其中一台服务器万一挂了，只要还有其他服务器正常运行，就不会影响用户使用。



负载均衡





Nginx也可用作虚拟主机。有些网站访问量大，需要负载均衡。然而并不是所有网站都深受用户爱好，有的网站，由于访问量太小，需要节省成本，将多个网站部署在同一台服务器上。

例如将www.asd.com和www.zxc.com两个网站部署在同一台服务器上，两个域名解析到同一个IP地址，但是用户通过两个域名却可以打开两个完全不同的网站，互相不影响，就像访问两个服务器一样，所以叫两个虚拟主机。



上图proxy\_pass <http://localhost:8080/;少一个/>。

在服务器8080和8090分别开了一个应用，客户端通过不同的域名访问，根据server\_name可以反向代理到对应的应用服务器。

虚拟主机的原理是通过HTTP请求头中的Host是否匹配server\_name来实现的，另外，server\_name配置还可以过滤有人恶意将某些域名指向你的主机服务器。

## nginx模块

Nginx的模块从结构上分为核心模块、基础模块和第三方模块：

核心模块：HTTP模块、EVENT模块和MAIL模块

基础模块：HTTP Access模块、HTTP FastCGI模块、HTTP Proxy模块和HTTP Rewrite模块

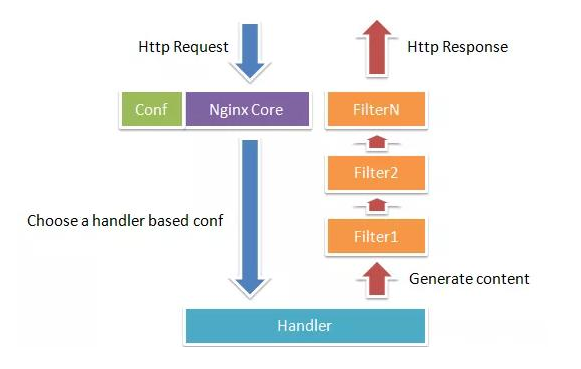
第三方模块：HTTP Upstream Request Hash模块、Notice模块和HTTP Access Key模块。

Nginx的模块从功能上分为如下三类：

Handlers（处理器模块）。此类模块直接处理请求，并进行输出内容和修改headers信息等操作。Handlers处理器模块一般只能有一个。

Filters （过滤器模块）。此类模块主要对其他处理器模块输出的内容进行修改操作，最后由Nginx输出。

Proxies （代理类模块）。此类模块是Nginx的HTTP Upstream之类的模块，这些模块主要与后端一些服务比如FastCGI等进行交互，实现服务代理和负载均衡等功能。



## nginx的进程模型

NGINX 启动后，有一个主进程（master process）和一个或多个工作进程（worker process），主进程的作用主要是读入和检查NGINX的配置信息，以及维护工作进程；工作进程才是真正处理客户端请求的进程。具体要启动多少个工作进程，可以在 NGINX 的配置文件nginx.conf中通过worker\_processes指令指定。

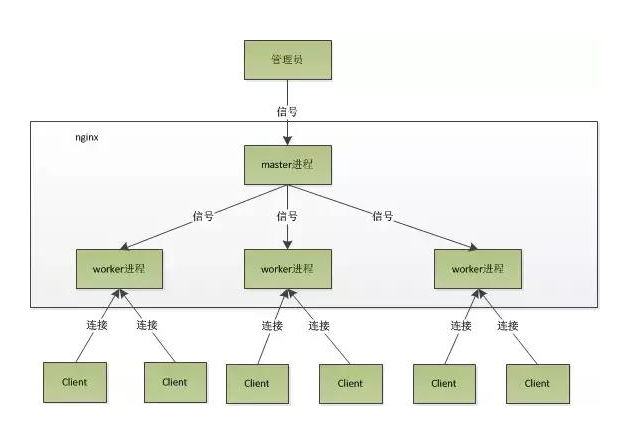
Nginx默认采用多进程工作方式，Nginx启动后，会运行一个master进程和多个worker进程。

master进程充当整个进程组与用户的交互接口，同时对进程进行监护，管理。

worker进程来实现重启服务、平滑升级、更换日志文件、配置文件实时生效等功能。

worker用来处理基本的网络事件，worker之间是平等的，他们共同竞争来处理来自客户端的请求。

nginx的进程模型如图所示：



## 配置服务器名称

服务器名称是用server\_name指令来定义的，并且它决定了哪一个server块将用来处理给定的请求。可以使用精确名称、通配符、正则表达式来定义服务器名称。

server {

listen 80;

server\_name example.org www.example.org;

...

}

server {

listen 80;

server\_name \*.example.org;

...

}

server {

listen 80;

server\_name mail.\*;

...

}

server {

listen 80;

server\_name ~^(?<user>.+)\.example\.net$;

...

}

当寻找一个虚拟服务器的名字，如果指定的名称匹配多个变体，例如，通配符和正则表达式都匹配，将会按照以下的顺序选择第一个匹配的变体：

精确名称

以星号（\*）开头的最长的通配符，例如“\*.example.org”

以星号（\*）结尾的最长的通配符，例如“mail.\*”

第一个匹配的正则表达式（根据在配置文件中出现的顺序）

## 配置 HTTPS 服务器

修改 conf/nginx.conf 文件，必须在配置文件 server 块中的监听指令 listen 后启用 ssl 参数，并且指定服务器证书 ssl\_certificate 和私钥 ssl\_certificate\_key 的位置：

server {

listen 443 ssl;

server\_name www.example.com;

ssl\_certificate www.example.com.crt;

ssl\_certificate\_key www.example.com.key;

ssl\_protocols TLSv1 TLSv1.1 TLSv1.2;

ssl\_ciphers HIGH:!aNULL:!MD5;

...

}

服务器证书是一个公共实体，它被发送给连接到服务器的每一个客户机。私钥是一个安全实体，应该存储在具有受限访问的文件中，但它必须可被nginx主进程读取。私钥也可以存储在与服务器证书相同的文件中：

ssl\_certificate www.example.com.cert;

ssl\_certificate\_key www.example.com.cert;

在这种情况下，这个证书文件的访问权限也应受到限制。虽然证书和密钥存储在一个文件中，但只有证书被发送到客户端。

指令 ssl\_protocols 和 ssl\_ciphers 可用于限制仅包括强版本和密码的 SSL/TLS 连接。 默认情况下，NGINX 使 用ssl\_protocols TLSv1 TLSv1.1 TLSv1.2版本和ssl\_ciphers HIGH:!aNULL:!MD5密码，因此通常不需要显式地配置它们。需要注意的是，这些指令的默认值在不同的版本里面已经变更好几次了。

## 作为 HTTP 负载均衡器

跨多个应用程序实例的负载均衡是优化资源利用率，最大限度地提高吞吐量，降低延迟，并确保容错配置一个常用的技术。

NGINX 支持如下负载均衡的机制（或方法）：

1. 轮询

如果没有指定负载均衡的方法，那么 NGINX 默认采用的是轮询的方式。最简单的负载均衡配置如下：

http {

upstream myapp1 {

server srv1.example.com;

server srv2.example.com;

server srv3.example.com;

}

server {

listen 80;

location / {

proxy\_pass http://myapp1;

}

}

}

3个同样实例的应用（srv1-srv3）是采用轮询方式。所有请求被代理到一组服务myapp1，同时，NGINX 运用 HTTP 负载均衡来分发请求。

反向代理被应用在 NGINX 内，包括负载均衡针对 HTTP、HTTPS、FASTCGI、uwsgi、SCGI 以及 memcached。

配置负载均衡针对 HTTPS 替代 HTTP 的话，仅仅使用 https 协议即可（proxy\_pass https://myapp1）。

在为 FASTCGI、uwsgi、SCGI 或 memcached 设置负载均衡时，分别使用 fastcgi\_pass、uwsgi\_pass、scgi\_pass 和 memcached\_pass 指令。

2. 最少连接

在一些请求需要更长时间才能完成的情况下，最少连接可以更公正地控制应用程序实例的负载。

使用最少连接的负载平衡，NGINX 将不会加重一个有过多请求的应用服务负担，而是将它分发新的请求给最不繁忙的服务器。

在 NGINX 中需要通过设置least\_conn来激活最少连接的负载均衡策略配置：

upstream myapp1 {

least\_conn;

server srv1.example.com;

server srv2.example.com;

server srv3.example.com;

}

3. IP 哈希（会话持久）

注意，采用轮询或者最少连接的负载均衡策略，每个客户端的后续请求可能被分配带不同的服务器，不能保证同一个客户端总是指向同一个服务。如果需要告诉客户端分配到一个特定的应用服务，换句话，就是保持客户端的会话粘性（sticky）或者会话持久性（persitent），即总是尝试选着同一个特定的服务器，IP 哈希 负载均衡机制可以被使用。

采用 IP 哈希的策略，客户端的 IP 地址被用作一个哈希 key，决定哪个服务应该被选中来服务客户端的请求。这种方式，确保了同一个客户端来的请求将总是被指向同一个服务，除非这个服务不可用了。

配置IP 哈希负载均衡，只需要通过设置ip\_hash来激活：

upstream myapp1 {

ip\_hash;

server srv1.example.com;

server srv2.example.com;

server srv3.example.com;

}

权重（weight）

可以通过使用服务器的权重来影响 NGINX 的负载均衡算法，在上述轮询、最少请求、基于IP 哈希负载均衡配置中，服务器的权重没有配置，意味着所有服务器的权重都是一样的。特别是轮询，它意味着或多或少平等的分发请求到服务器（请求够多，并且请求以均匀方式进行处理，并完成够快）

当配置了一个 weight 变量到一个指定的服务后，权重被作为一个 NGINX 的负载均衡的决定的一部分：

upstream myapp1 {

server srv1.example.com weight=3;

server srv2.example.com;

server srv3.example.com;

}

采用上面的配置，如果来了5个请求，3个到srv1，1个到srv2,1个到srv3。在最近的NGINX版本中，同样可以使用权重针对最少连接和IP 哈希的负载均衡策略。

# 二、Ngnix Windows下安装和配置:

1)下载地址

[http://nginx.org](http://nginx.org/)

2)启动

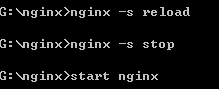
解压至c:\nginx，运行nginx.exe(即nginx -c conf\nginx.conf)，默认使用80端口，日志见文件夹C:\nginx\logs

3)使用

[http://localhost](http://localhost/)

输入locahost，显示下图所示，则说明安装已成功。



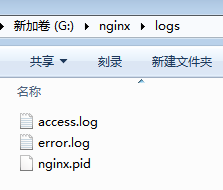


如果运行命令的时候出现错误：



可以查看一下文件是否存在中文名或设置一下文件夹logs的权限。

4)进程

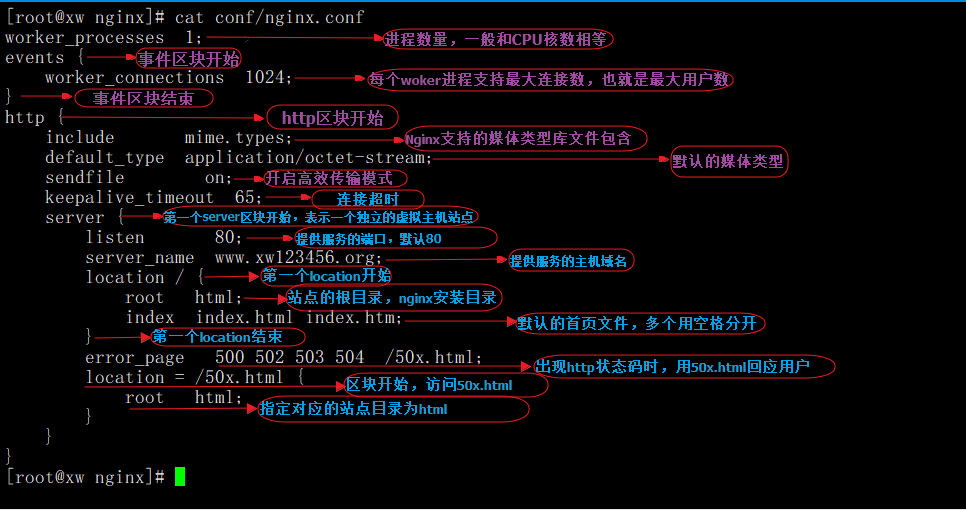


如果文件夹logs中出现nginx.pid，则说明nginx进程已启动，stop后该文件会消失。

5)配置

nginx的安装时配置文件默认在nginx程序安装目录的conf目录下，启动主配置文件默认为为nginx.conf。

安装的时候可以通过—conf-path来指定配置文件的放置路径，同时启动的时候可以通过 -c指令来另行指定启动的配置文件。



在配置文件中，修改：

location / {

root html;

index index.html index.htm;

}

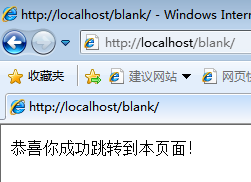
location /blank/ {

proxy\_pass http://localhost:8080/;

index index.html index.htm;

}

此时访问http://localhost/blank/，即可：



该页面实际发布在IIS中，端口为8080，<http://localhost:8080/>。

在使用IP地址的时候，ASP.NET中的Response.Redirect有时会出现路径问题，建议使用Response.Write(“<script language=’javascript’>window.location=’V1/Manage/Default.aspx’</script>”);可以解决。

6)服务器集群-负载均衡

在http节点下添加upstream（服务器集群），server设置的是集群服务器的信息，我这里搭建了两个站点，配置了两条信息。

#服务器集群名称为Jq\_one

upstream Jq\_one {  
 　　server  127.0.0.1:9000;   
 　　server  127.0.0.1:8082;   
    }

在http节点下找到location节点修改：

  location / {  
            root   html;  
            index  index.aspx index.html index.htm; #修改主页为index.aspx  
     #其中jq\_one 对应着upstream设置的集群名称  
     proxy\_pass         http://Jq\_one;   
     #设置主机头和客户端真实地址，以便服务器获取客户端真实IP  
     proxy\_set\_header   Host             $host;   
     proxy\_set\_header   X-Real-IP        $remote\_addr;   
     proxy\_set\_header   X-Forwarded-For  $proxy\_add\_x\_forwarded\_for;  
 }



感觉讲解的比较好的文章有《[nginx+iis实现负载均衡](http://www.cnblogs.com/yanweidie/p/4658136.html)》(<http://www.cnblogs.com/yanweidie/p/4658136.html>)

//举例，以下IP，端口无效

upstream test{

server 11.22.333.11:6666 weight=1;

server 11.22.333.22:8888 down;

server 11.22.333.33:8888 backup;

server 11.22.333.44:5555 weight=2;

}

//down 表示当前的server临时不參与负载.

//weight 默觉得1.weight越大，负载的权重就越大

//backup： 其他全部的非backup机器down或者忙的时候，请求backup机器。所以这台机器压力会最轻

后面的 weight=1，weight=2 是表示权重的意思，数字越大，权重越高，在该例中 8811 这个端口权重就是 8855 的两倍，比如三次请求，大概就是两次分发给 8811 一次分发给 8855 ，其实这个是不需要写的，upstream 模块默认就是轮询法，每个ip分发一次，设置权重（加权轮询法）的意义上面已经解释过了，可以看下。

什么是源地址哈希法，就是对访问用户的IP进行hash后的结果进行分配，这样每一个用户固定请求同一个后端服务器，能够解决session的问题。  
直接上代码吧：

upstream guwenjie\_http {

ip\_hash;

server \*\*.\*\*\*.\*\*\*.\*\*\*:8855;

server \*\*.\*\*\*.\*\*\*.\*\*\*:8811;

}

详见：

<https://blog.csdn.net/gu_wen_jie/article/details/82149003>

7)session共享

如果站点使用了session，请求平均分配到两个站点，那么必然存在session共享问题，该如何解决？

(1)使用数据库保存session信息

(2)使用nginx将同一ip的请求分配到固定服务器，修改如下。ip\_hash会计算ip对应hash值，然后分配到固定服务器。

upstream Jq\_one {

 server  127.0.0.1:9000;

 server  127.0.0.1:8082;

    ip\_hash;

}

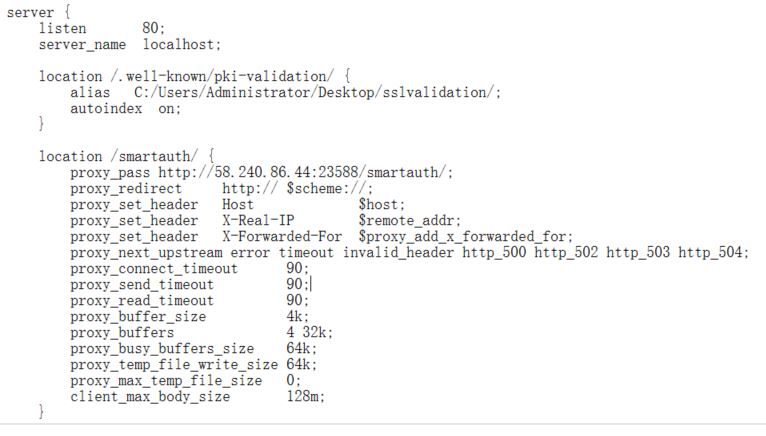
(3)搭建一台Redis服务器，对session的读取都从该Redis服务器上读取。后面的文章将介绍分布式缓存Redis的使用。

8)上传文件大小限制(413 Request Entity Too Large)



client\_max\_body\_size 改为100m甚至更大。

9)可以反向代理外网地址



上文中的smartauth改为smartDesign

http://58.240.86.44:23588/smartDesign/index.html#

就可以通过[http://ck.jsptpd.com/smartDesign/index.html#](http://ck.jsptpd.com/smartDesign/index.html)来访问了。

10)TCP反向代理

events {

worker\_connections 1024;

}

stream {

server {

listen 4018;

proxy\_pass 172.16.241.115:4018;

}

}

http {

……

}

在1.9.0之后支持TCP反向代理。注意本地防火墙和阿里云控制台防火墙对端口的限制。

也可以设置负载均衡：

stream {

    server {

        listen 1034;

        proxy\_pass app;

    }

    upstream app {

        server 192.168.0.3:1034;

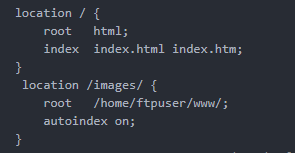
        server 192.168.0.4:1034;

        server 192.168.0.6:1034;

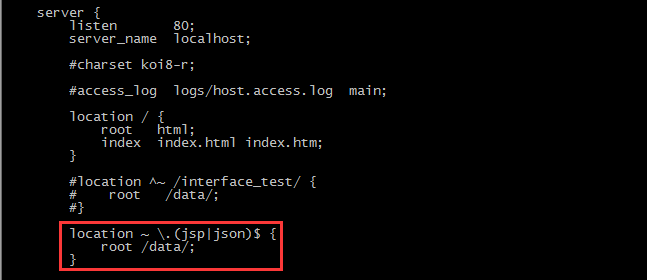
    }

}

11)root访问静态文件



root: 是指资源在你的服务器中的绝对路径,这里千万不能出错,要不然会找不到资源返回404错误,我这里是配置了 /home/ftpuser/www/ ,这个路径在访问的时候不会显性展示出来,自己要记得自己存放文件的路径啊



只要URL中的文件后缀匹配上，则匹配到/data/目录，访问该目录下静态文件。

location / {

alias D:\filePath\public/;

autoindex on;

}

12) nginx默认首页是404或者自定义页面

location / {

return 404;

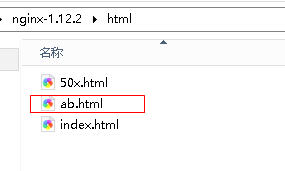
}

error\_page 404 /404.html;

其实还有个非常简单的写法：

server {  
        listen       8001;  
        server\_name  localhost;  
        index        ab.html;

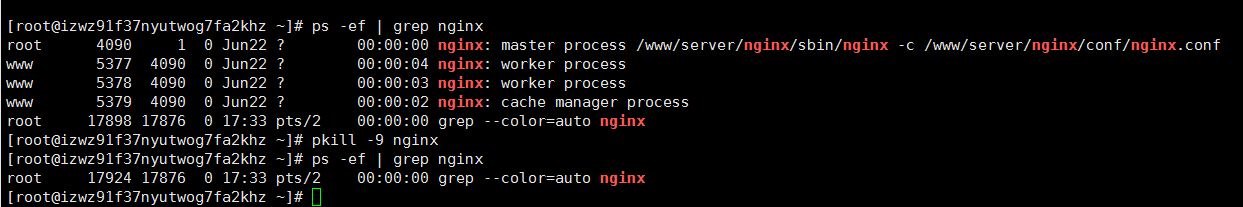
原本index是对应到index.html，现在改成ab.html，将该页面拷贝到nginx的html页面中。



13)指定配置文件启动

-c,具体命令如下：

/usr/sbin/nginx -c /etc/nginx/conf.d/proxy.conf



指定配置文件重启的话，我们可按如下操作关闭nginx:

pkill -9 nginx

14)查看当前nginx进程

命令：ps -ef | grep nginx，操作如下：

[root@unicom conf.d]# ps -ef | grep nginx

root      3277     1  0 19:11 ?        00:00:00 nginx: master process /usr/sbin/nginx -c /etc/nginx/conf.d/proxy.conf

nginx     3278  3277  0 19:11 ?        00:00:00 nginx: worker process

root      3280  3099  0 19:11 pts/0    00:00:00 grep nginx

其实你也能看到使用的nginx配置文件路径。

15)重启

关闭服务：servie nginx stop

重启服务：servie nginx restart

在 nginx 中，如果我修改了一个配置文件，或者新增了一个配置文件。按照群里网友的描述，我需要关闭 nginx，或者重启 nginx，比较麻烦，而且这个过程中会中断服务的提供。那么有没有热配置，或者是不重启就可以生效呢？

答案当然是有的。那就是使用 nginx -s reload 命令。-s 代表的是向主进程发送信号。其中信号有 4 个，stop, quit, reopen, reload。reload 就是重新加载的意思。nginx -s reload 命令，合起来的作用就是重新加载配置文件。

16) Nginx错误页面隐藏版本号

一、隐藏Nginx服务器出现403、404错误访问页面时显示的Nginx版本号

[](https://s5.51cto.com/wyfs02/M01/7A/ED/wKiom1bByXGxQgg8AABE5de79v8656.jpg)

1、在nginx虚拟机主机配置文件中的server模块中增加一行server\_tokens指令，指定关闭（off）显示版本号

server {

listen 80;

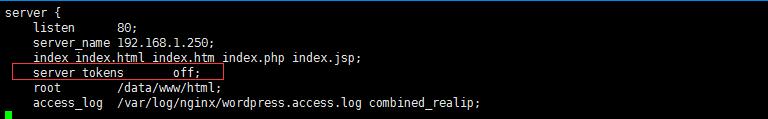
server\_name 192.168.1.250;

index index.html index.htm index.php index.jsp;

server\_tokens off;

root /data/www/html;

access\_log /var/log/nginx/wordpress.access.log combined\_realip;

[](https://s3.51cto.com/wyfs02/M01/7A/ED/wKioL1bBzI3AXAx1AAAylpQ5czE466.jpg)

2、重启或重新加载Nginx服务器，测试访问



17) 检查Nginx配置文件是否正确

nginx -tc /etc/nginx/nginx.conf

18) 防爬虫策略

<https://blog.csdn.net/shangrila_kun/article/details/89501343>

#禁止Scrapy等工具的抓取

if ($http\_user\_agent ~\* (Scrapy|Curl|HttpClient)) {

return 403;

}

#禁止指定UA及UA为空的访问

if ($http\_user\_agent ~ "WinHttp|WebZIP|FetchURL|node-superagent|java/|

FeedDemon|Jullo|JikeSpider|Indy Library|Alexa Toolbar|AskTbFXTV|AhrefsBot|

CrawlDaddy|Java|Feedly|Apache-HttpAsyncClient|UniversalFeedParser|ApacheBench|

Microsoft URL Control|Swiftbot|ZmEu|oBot|jaunty|Python-urllib|

lightDeckReports Bot|YYSpider|DigExt|HttpClient|MJ12bot|heritrix|EasouSpider|Ezooms|BOT/0.1|

YandexBot|FlightDeckReports|Linguee Bot|^$" ) {

return 403;

}

注：^$是UA为空的情况，特殊情况下可以去掉。

19) 修改配置文件

cd /usr/lib/systemd/system

vim nginx.service

修改PID文件地址：

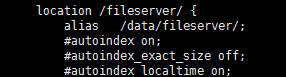
PID file=/usr/local/nginx/logs/nginx.pid

增加以下内容，延缓systemd操作，否则和nginx创建PID文件之前存在冲突：

ExecStartPost=/bin/sleep 0.1

20) 开启目录浏览

autoindex on





21) 关闭目录浏览

autoindex off;

可以放在http段，也可以放在location段。

location /down/ {  
alias /home/wwwroot/lnmp/test/;  
autoindex on;  
}

# 三、Ngnix Linux下安装和配置

可参见“技术-系统-Linux-CentOS”中《Linux（CentOS）下，下载安装Nginx并配置(已验证)》。

如果本地电脑无法访问虚拟机里nginx发布的站点，且ip能够ping通，多半是因为防火墙的问题，请参见“技术-Nginx”中《[nginx开启后主机无法访问虚拟机的nginx解决方案](http://www.cnblogs.com/yomho/p/6074815.html)》

yum install nginx默认安装目录：

# config: /etc/nginx/nginx.conf     安装目录  
# config: /etc/sysconfig/nginx  
# pidfile: /var/run/nginx.pid

日志文件在var/log/nginx



# 四、补充

http://tengine.taobao.org/book/chapter\_02.html

nginx在启动后，会有一个master进程和多个worker进程。master进程主要用来管理worker进程，包含：接收来自外界的信号，向各worker进程发送信号，监控worker进程的运行状态，当worker进程退出后(异常情况下)，会自动重新启动新的worker进程。

感觉woker进程数跟配置文件中的：worker\_processes 2;有关。

在nginx启动后，如果我们要操作nginx，要怎么做呢？从上文中我们可以看到，master来管理worker进程，所以我们只需要与master进程通信就行了。master进程会接收来自外界发来的信号，再根据信号做不同的事情。所以我们要控制nginx，只需要通过kill向master进程发送信号就行了。比如kill -HUP pid，则是告诉nginx，从容地重启nginx，我们一般用这个信号来重启nginx，或重新加载配置，因为是从容地重启，因此服务是不中断的。master进程在接收到HUP信号后是怎么做的呢？首先master进程在接到信号后，会先重新加载配置文件，然后再启动新的worker进程，并向所有老的worker进程发送信号，告诉他们可以光荣退休了。新的worker在启动后，就开始接收新的请求，而老的worker在收到来自master的信号后，就不再接收新的请求，并且在当前进程中的所有未处理完的请求处理完成后，再退出。当然，直接给master进程发送信号，这是比较老的操作方式，nginx在0.8版本之后，引入了一系列命令行参数，来方便我们管理。比如，./nginx -s reload，就是来重启nginx，./nginx -s stop，就是来停止nginx的运行。如何做到的呢？我们还是拿reload来说，我们看到，执行命令时，我们是启动一个新的nginx进程，而新的nginx进程在解析到reload参数后，就知道我们的目的是控制nginx来重新加载配置文件了，它会向master进程发送信号，然后接下来的动作，就和我们直接向master进程发送信号一样了。

现在，我们知道了当我们在操作nginx的时候，nginx内部做了些什么事情，那么，worker进程又是如何处理请求的呢？我们前面有提到，worker进程之间是平等的，每个进程，处理请求的机会也是一样的。当我们提供80端口的http服务时，一个连接请求过来，每个进程都有可能处理这个连接，怎么做到的呢？首先，每个worker进程都是从master进程fork过来，在master进程里面，先建立好需要listen的socket（listenfd）之后，然后再fork出多个worker进程。所有worker进程的listenfd会在新连接到来时变得可读，为保证只有一个进程处理该连接，所有worker进程在注册listenfd读事件前抢accept\_mutex，抢到互斥锁的那个进程注册listenfd读事件，在读事件里调用accept接受该连接。当一个worker进程在accept这个连接之后，就开始读取请求，解析请求，处理请求，产生数据后，再返回给客户端，最后才断开连接，这样一个完整的请求就是这样的了。我们可以看到，一个请求，完全由worker进程来处理，而且只在一个worker进程中处理。

很多人会误解worker\_connections这个参数的意思，认为这个值就是nginx所能建立连接的最大值。其实不然，这个值是表示每个worker进程所能建立连接的最大值，所以，一个nginx能建立的最大连接数，应该是worker\_connections \* worker\_processes。当然，这里说的是最大连接数，对于HTTP请求本地资源来说，能够支持的最大并发数量是worker\_connections \* worker\_processes，而如果是HTTP作为反向代理来说，最大并发数量应该是worker\_connections \* worker\_processes/2。因为作为反向代理服务器，每个并发会建立与客户端的连接和与后端服务的连接，会占用两个连接。

reload和restart的区别

reload，重新加载的意思，reload会重新加载配置文件，nginx服务不会中断，而且reload时会测试conf语法等，如果出错会rollback用上一次正确配置文件保持正常运行。  
 restart，重启，会重启nginx服务。这个重启会造成服务一瞬间的中断，当然如果配置文件出错会导致服务启动失败，那就是更长时间的服务中断了。

所以，根据不同情况使用不同命令最好不过了。

# 五、常见问题解答

swap file “\*.swp”already exists！

在 Linux 下的 vim 编辑过程中，由于某种原因异常退出正在编辑的文件，再次编辑该文件时，会出现如下提示



使用vim编辑文件实际是先 copy 一份临时文件并映射到内存给你编辑， 编辑的是临时文件， 当执行：w 后才保存临时文件到原文件，执行：q 后才删除临时文件。

可以通过rm .proxy.conf.swp删除的方式解决。

PID file /var/run/nginx.pid not readable

cd /usr/lib/systemd/system

vim nginx.service

修改PID文件地址：

PID file=/usr/local/nginx/logs/nginx.pid

增加以下内容，延缓systemd操作，否则和nginx创建PID文件之前存在冲突：

ExecStartPost=/bin/sleep 0.1

nginx: [emerg] "server" directive is not allowed here in /etc/nginx/conf.d/proxy.conf:142

这种错误一般都是语法错误。

nginx升级到1.16后，必须要有如下：

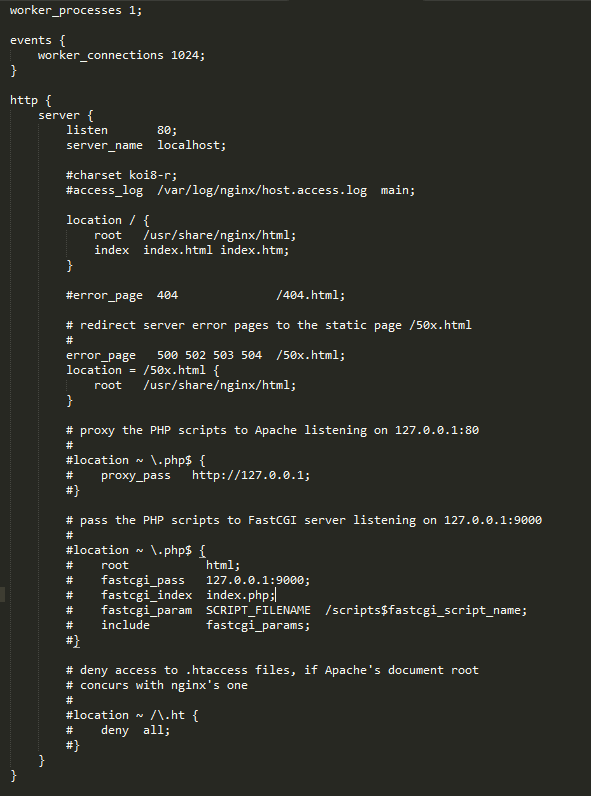
worker\_processes 1;

events {

worker\_connections 1024;

}

同时，server块要写到http {中，如：



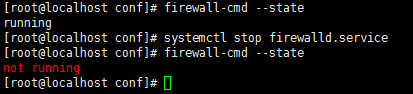
nginx服务启动，页面无法访问

这种错误先看防火墙是否开着。

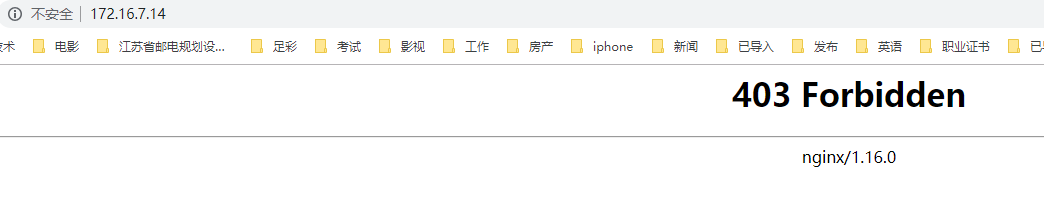
telnet 172.16.7.14 80

如果不通。

firewall-cmd –state



如果是running，按照上图方法关闭：



这时就OK了。

windows下nginx一闪而过，无法启动

这种错误先看nginx目录下的logs，能找到原因的，error。

windows下openssl提示丢失MSVCR120.dll

如果提示丢失MSVCR120.dll，安装VC运行库。链接：https://pan.baidu.com/s/1q61JWYlw0gQcWLom3ssKzQ 密码：musy，或者直接搜索vcredist\_x64.exe，安装VC++组件库。