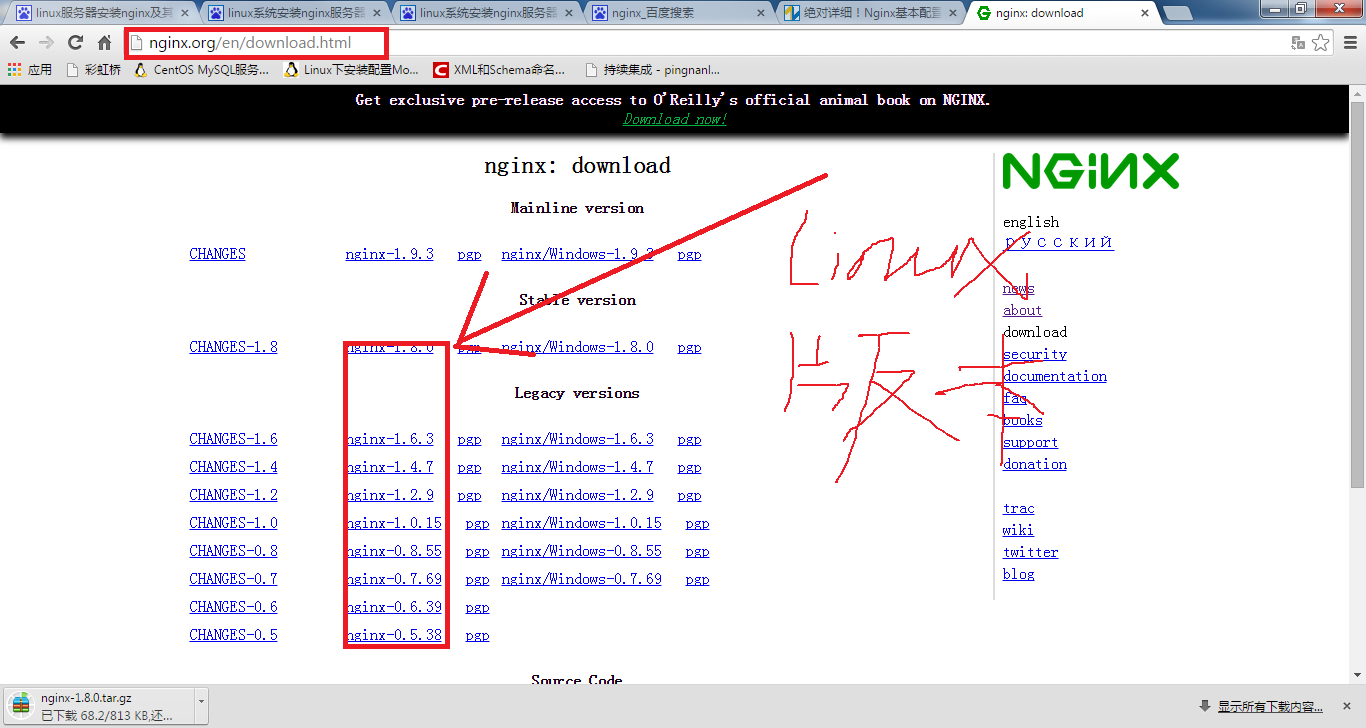
Linux中安装和配置nginx

# 下载nginx的tar包



# 安装nginx

## 安装编译nginx

|  |
| --- |
| #安装依赖库  yum -y install gcc pcre-devel zlib-devel  #编译安装nginx  tar -zxvf nginx-1.8.0.tar.gz  cd nginx-1.8.0  ./configure --user=www --group=www --prefix=/usr/local/nginx --with-http\_stub\_status\_module --with-http\_gzip\_static\_module  make && make install  #配置环境变量  cat >> /etc/profile<<EOF  export NGINX\_HOME=/usr/local/nginx  export PATH=\$PATH:\$NGINX\_HOME/sbin  EOF  source /etc/profile  #备份和修改配置  useradd -s /sbin/nologin www  mkdir /data  cp -f /usr/local/nginx/conf/nginx.conf /usr/local/nginx/conf/nginx.bak  cat > /usr/local/nginx/conf/nginx.conf <<EOF  user www;  worker\_processes 4;  error\_log logs/error.log notice;  pid /var/run/nginx.pid;  worker\_rlimit\_nofile 65536;  events {  use epoll;  worker\_connections 65536;  }  http {  include mime.types;  default\_type application/octet-stream;  charset UTF-8;  limit\_conn\_zone \$binary\_remote\_addr zone=one:10m;  log\_format main '\$remote\_addr - \$remote\_user [\$time\_local] '  '"\$request" \$status \$bytes\_sent '  '"\$http\_referer" "\$http\_user\_agent" '  '"\$gzip\_ratio"';  log\_format download '\$remote\_addr - \$remote\_user [\$time\_local] '  '"\$request" \$status \$bytes\_sent '  '"\$http\_referer" "\$http\_user\_agent" '  '"\$http\_range" "\$sent\_http\_content\_range"';  access\_log logs/access.log main;  client\_max\_body\_size 20m;  client\_header\_buffer\_size 32k;  large\_client\_header\_buffers 4 128k;  sendfile on;  tcp\_nopush on;  tcp\_nodelay on;  keepalive\_timeout 60;  client\_header\_timeout 10;  client\_body\_timeout 10;  send\_timeout 10;  client\_body\_buffer\_size 512k;  proxy\_connect\_timeout 5;  proxy\_read\_timeout 60;  proxy\_send\_timeout 5;  proxy\_buffer\_size 16k;  proxy\_buffers 4 64k;  proxy\_busy\_buffers\_size 128k;  proxy\_temp\_file\_write\_size 64k;  gzip on;  gzip\_min\_length 1k;  gzip\_buffers 4 16k;  gzip\_http\_version 1.1;  gzip\_comp\_level 2;  gzip\_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript;  gzip\_vary on;  proxy\_cache\_path /data/cache levels=1:2 keys\_zone=cache\_one:4096m max\_size=3g inactive=600m;  proxy\_temp\_path /data/temp;  open\_file\_cache max=65536 inactive=20s;  open\_file\_cache\_valid 30s;  open\_file\_cache\_min\_uses 2;  open\_file\_cache\_errors on;  upstream webServer{  server 10.161.171.195:80 weight=3 max\_fails=3 fail\_timeout=20s;  server 10.161.158.22:80 weight=3 max\_fails=3 fail\_timeout=20s;  server 10.161.132.101:80 weight=3 max\_fails=3 fail\_timeout=20s;  }  server{  listen 80;  server\_name helpedu.com 218.244.141.167;  index index.html index.htm index.jsp  #root /home/k12;  charset UTF-8;  access\_log logs/access.log main;  #错误页面处理  error\_page 404 500 502 503 504 /404.html;  location /404.html{  #root /home/k12;  }  #静态请求处理  location ~ .\*\.(htm|html|js|css|less|json|gif|jpg|jpeg|png|bmp|swf|ioc|rar|zip|txt|flv|mid|doc|ppt|pdf|xls|mp3|wma|apk|zip|rar)\$ {  root /home/k12/static;  expires 30d;  }  location ~^/(upload|html)/{  root /home/k12;  expires 30d;  }  #location /download/ {  # limit\_conn one 1;  # valid\_referers none blocked server\_names \*.example.com;  # if (\$invalid\_referer) {  # return 403;  # }  # rewrite ^/(download/.\*)/mp3/(.\*)\..\*\$ /\$1/mp3/\$2.mp3 break;  # root /home/k12;  # access\_log logs/download.log download;  #}  #新旧地址交替  #location /old\_stuff/ {  # rewrite ^/old\_stuff/(.\*)\$ /new\_stuff/\$1 permanent;  #}  #防锁链  #location ~\* \.(gif|jpg|png|swf|flv)\$ {  # valid\_referers none blocked \*.nginxcn.com;  # if (\$invalid\_referer) {  # rewrite ^/ www.nginx.cn;  # }  #}  location /NginxStatus {  stub\_status on;  access\_log off;  #allow可以有多个  allow 122.225.227.163;  deny all;  }  location = / {  #proxy\_pass http://web/index  }  location /{  proxy\_cache cache\_one;  proxy\_cache\_key \$host\$uri\$is\_args\$args;  proxy\_redirect off;  proxy\_set\_header HOST \$host;  proxy\_set\_header X-Real-IP \$remote\_addr;  proxy\_set\_header X-Forwarded-For \$proxy\_add\_x\_forwarded\_for;  #proxy\_pass http://webServer/web;  proxy\_next\_upstream http\_500 http\_502 http\_503 error timeout invalid\_header;  }  }  }  EOF  #启动nginx  nginx -c /usr/local/nginx/conf/nginx.conf |

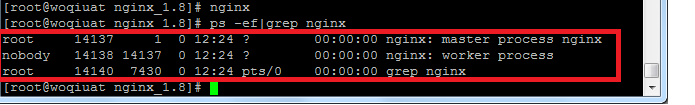
## 启动服务观察是否成功

### nginx服务管理

|  |
| --- |
| #检查配置文件,-c指定配置文件  nginx –t  nginx -t -c /usr/nginx/conf/nginx.conf  #查看版本  nginx –v/V |
| #nginx的启动关闭和重启  nginx #启动  nginx –s reload #重启  fuser -k 80/tcp #命令关闭占用80端口的程序  nginx -s stop #停止服务  nginx对进程的控制能力非常强大，可以通过信号命令控制进程。常用的有:   * QUIT：处理完当前请求后关闭。 * HUP：重新加载配置，关闭原有进程并开启新进程（不会中断用户请求，可以实现平滑重启） * USR1：用于nginx日志切换，也就是重新打开日志文件，每天生成新的日志文件时候可以使用这个信号来切换。 * USR2：用于平滑升级可执行程序 * WINCH：从容关闭工作进程   #kill –xxx pid  #获取进程号  ps -ef | grep "master process" | grep -v "grep" | awk '{print $2}'  cat /usr/local/nginx/logs/nginx.pid  平滑启动nginx:  kill -HUP $(cat /usr/local/nginx/logs/nginx.pid) |

### 查看是否成功

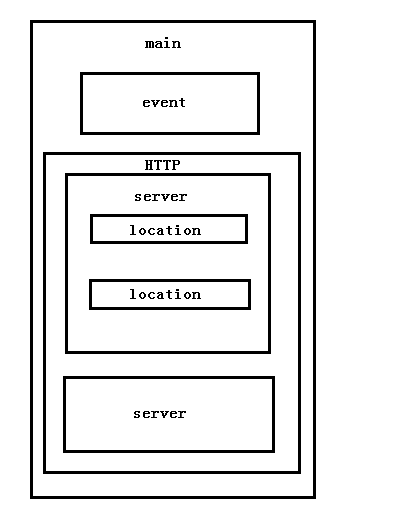
ps -ef|grep nginx



* --prefix=path 定义一个目录，存放服务器上的文件 ，也就是nginx的安装目录。默认使用 /usr/local/nginx。
* --sbin-path=path 设置nginx的可执行文件的路径，默认为 prefix/sbin/nginx.
* --conf-path=path 设置在nginx.conf配置文件的路径。nginx允许使用不同的配置文件启动，通过命令行中的-c选项。默认为prefix/conf/nginx.conf.

# 基本配置nginx

默认的配置文件在：prefix/conf/nginx.conf。Nginx配置文件的结构层次：



nginx配置文件主要分为四个部分：

main(全局设置)：全局设置，命令影响所有其他的设置

server(主机设置)：主要用于指定主机和端口

upstream(负载均衡服务器设置)：指定一系列后端服务器

location(URL特定位置设定)：用于匹配网页位置。

## 配置解析

|  |
| --- |
| #worker进程运行用户以及用户组  user www;  #nginx要启动的进程数  worker\_processes 4;  #用来定义全局的错误日志文件(日志级别：debug,info,notice,warn,error,crit)  #其中debug日志最详细，crit日志最少  error\_log logs/error.log notice;  #指定 pid 存放的路径  pid /var/run/nginx.pid;  #用于绑定worker进程和cpu  worker\_rlimit\_nofile 65536;  #用来设定nginx工作模式和链接上限数  events {  #指定工作模式(select,poll,kqueue,epoll,rtsig,/dev/poll)  #select,poll都是标准的工作模式;  #kqueue,epoll是高效的工作模式,epoll首选  use epoll;  #每个进程最大连接数,和worker\_processes决定最大客户端连接数  #worker\_connections生效：ulimit -n 65536  worker\_connections 65536;  }  http {  #主模块命令，实现配置文件所包含文件的设定。  include conf/mime.types;  #当文件类型未定义时候使用使用这种方式  #application/octet-stream二进制流可下载  default\_type application/octet-stream;  #默认字符编码格式  charset UTF-8;  #定义一个叫“one”的记录区，总容量为 10M，  #以变量 $binary\_remote\_addr 作为会话的判断基准（即一个地址一个会话）  limit\_conn\_zone $binary\_remote\_addr zone=one:10m;  #nginx日志输出格式  log\_format main '$remote\_addr - $remote\_user [$time\_local] '  '"$request" $status $bytes\_sent '  '"$http\_referer" "$http\_user\_agent" '  '"$gzip\_ratio"';  log\_format download '$remote\_addr - $remote\_user [$time\_local] '  '"$request" $status $bytes\_sent '  '"$http\_referer" "$http\_user\_agent" '  '"$http\_range" "$sent\_http\_content\_range"';  #nginx日志是否启用  access\_log logs/access.log main;  #设置允许客户端请求的最大单个文件字节数  client\_max\_body\_size 20m  #指定来自客户端请求头的headerbuffer大小  #一般1k够用，如果包含的cookie,增大缓冲区  client\_header\_buffer\_size 32k  #客户端请求中较大的消息头的缓存最大数量和大小  #最大缓存量是4个4k  large\_client\_header\_buffers 4 128k;  #开启高效文件传输模式  sendfile on  #tcp\_nopush和tcp\_nodelay设置on,用于防止网络阻塞  tcp\_nopush on;  tcp\_nodelay on;  #客户端连接保持活动时间，超过时间服务器将关闭连接  keepalive\_timeout 60;  #设置客户端请求头读取超时时间，超过时间报408  client\_header\_timeout 10;  #设置客户端请求主题读取超时时间，超过时间报408  client\_body\_timeout 10;  #设置响应客户端的超时时间  send\_timeout 10;  client\_body\_buffer\_size 512k;  #nginx跟后端服务器连接超时时间(代理连接超时)  proxy\_connect\_timeout 5;  #连接成功后，后端服务器响应时间(代理接收超时)  proxy\_read\_timeout 60;  #后端服务器数据回传时间(代理发送超时)  proxy\_send\_timeout 5;  #设置代理服务器（nginx）保存用户头信息的缓冲区大小  proxy\_buffer\_size 16k;  #proxy\_buffers缓冲区  proxy\_buffers 4 64k;  #高负荷下缓冲大小（proxy\_buffers\*2）  proxy\_busy\_buffers\_size 128k;  #设定缓存文件夹大小，大于这个值，将从upstream服务器传  proxy\_temp\_file\_write\_size 64k;  #httpGzip模块  #用于设置开启和关闭gzip模块，gzip on表示开启gzip压缩  #实时压缩数据流量  gzip on;  #设置允许压缩页面的最小字节，页面的字节数从header的Content-length  #中获取。默认为0，表示不管多大都压缩。建议设置成1k  gzip\_min\_length 1k;  #表示申请4个单位为16k的内存作为压缩结果流缓存，默认申请  #原始数据大小相同的内存空间开存储GZip压缩结果  gzip\_buffers 4 16k;  #用于设置识别的http版本  gzip\_http\_version 1.1;  #用来设置Gzip的压缩比，1为压缩比最小，处理速度最快；  #9表示压缩比最大，处理速度最慢，也消耗cpu资源  gzip\_comp\_level 2  #指定压缩类型，text/html总是被压缩  gzip\_types text/plain text/css application/json application/x-javascript text/xml application/xml application/xml+rss text/javascript;  #让前段的缓存服务器缓存经过Gzip压缩的页面，例如Squid缓存经过Nginx的数据  gzip\_vary on;  #proxy\_cache\_path:用于设置缓存目录，后面跟缓存路径。最好放在独立的硬盘  #level=1:2用来设置目录深度，这里是两层，第一层是一个字符，第二层是两个字符  #key\_zone：用来设置web缓存区名称，这里是cache-one,后面接缓存空间大小  #inactive=1d:表示自动清理缓存时间，d表示天，h表示小时，m表示分钟  #max\_size:表示硬盘空间可使用的最大值，内存不够用时候，nginx将不常访问的数据放在硬盘中  proxy\_cache\_path /data/cache levels=1:2 keys\_zone=cache\_one:4096m max\_size=3g inactive=600m;  #指定临时缓存文件的存储路径，和proxy\_cache\_path指定的路径必须在同一磁盘分区  proxy\_temp\_path /data/temp;  #设置打开缓存的同时也指定了缓存最大数目，以及缓存的时间。  open\_file\_cache max=65536 inactive=20s;  #指定检测正确信息的间隔时间  open\_file\_cache\_valid 30s;  #定义了open\_file\_cache中指令参数不活动时间期间里最小的文件数。  open\_file\_cache\_min\_uses 2;  #指定了当搜索一个文件时是否缓存错误信息，也包括再次给配置中添加文件。  open\_file\_cache\_errors on;  ###动态页面负载均衡配置，可添加多台配置  upstream webServer{  ###Tomcat服务  server 10.161.171.195:80 weight=3 max\_fails=3 fail\_timeout=20s;  server 10.161.158.22:80 weight=3 max\_fails=3 fail\_timeout=20s;  server 10.161.132.101:80 weight=3 max\_fails=3 fail\_timeout=20s;  }  #虚拟主机配置  server{  #虚拟主机服务端口  listen 80;  #用来指定ip地址或者域名,多个域名用空格隔开  server\_name helpedu.com 218.244.141.167;  #用来设定默认首页地址  index index.html index.htm index.jsp  #设定虚拟主机网页根目录  root /home/k12  #默认字符编码格式  charset UTF-8;  #虚拟主机的访问日志存放位置  access\_log logs/access.log main;  #错误页面处理  error\_page 404 500 502 503 504 /404.html;  location /404.html{  root /home/k12;  }  #静态文件处理  location ~ .\*\.(htm|html|js|css|less|json|gif|jpg|jpeg|png|bmp|swf|ioc|rar|zip|txt|flv|mid|doc|ppt|pdf|xls|mp3|wma|apk|zip|rar)$ {  root /home/k12/static;  expires 30d  }  location ~^/(upload|html)/{  root /home/k12;  expires 30d;  }  #下载处理  location /download/ {  #一个IP只能发起一个连接，多过一个，一律503。  limit\_conn one 1;  valid\_referers none blocked server\_names \*.example.com;  if ($invalid\_referer) {  #rewrite ^/http://www.example.com/;  return 403;  }  #rewrite\_log on;  # rewrite /download/\*/mp3/\*.any\_ext to /download/\*/mp3/\*.mp3  rewrite ^/(download/.\*)/mp3/(.\*)\..\*$ /$1/mp3/$2.mp3 break;  root /spool/www;  #autoindex on;  access\_log logs/download.log download;  }  #新旧网站交替  location /old\_stuff/ {  rewrite ^/old\_stuff/(.\*)$ /new\_stuff/$1 permanent;  }  #防盗链  #valid\_referers指令在referer头的基础上为 $invalid\_referer 变量赋值，其值为0或1。  #可以使用这个指令来实现防盗链功能，如果valid\_referers列表中没有Referer头的值，  #$invalid\_referer将被设置为1。  #参数可以使如下形式：  #none 意为不存在的Referer头(表示空的，也就是直接访问，比如直接在浏览器打开一个图片)  #blocked 意为根据防火墙伪装Referer头，如：“Referer: XXXXXXX”。  #server\_names 为一个或多个服务器的列表，0.5.33版本以后可以在名称中使用“\*”通配符。  location ~\* \.(gif|jpg|png|swf|flv)$ {  valid\_referers none blocked \*.nginxcn.com;  if ($invalid\_referer) {  rewrite ^/ www.nginx.cn  #return 403;  }  }  #Nginx配置状态,查看nginx的运行状态http://ip/NginxStatus  location /NginxStatus {  stub\_status on;  access\_log off;  #Nginx的一中认证方式  #auth\_basic "NginxStatus"  #指定认证的密码文件  #/usr/local/apache/bin/htpasswd -c /usr/local/nginx/conf/htpasswd webadmin  #auth\_basic\_user\_file ../htpasswd;  #allow SOME.IP.ADD.RESS;  allow 122.225.227.163  deny all;  }  #反向代理  location = /{  proxy\_pass http://webServer;  proxy\_redirect off;  proxy\_set\_header HOST $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  }  location /{  #反向代理缓存设置命令，默认为off。将proxy\_cache放在location中  #这样匹配的url才会被缓存  proxy\_cache cache\_one;  #对不同的http状态设置不同缓存时间  proxy\_cache\_valid 200 304 12h;  #这个命令设置什么样的参数得到缓存文件名。默认是#$scheme$proxy\_host$request\_uri  #表示协议，主机名，请求uri包含参数作为MD5得出缓存文件名  #这里使用域名，uri，参数组合成web缓存的key,nginx根据key值哈希，存储缓存内容到二级缓存目录  proxy\_cache\_key $host$uri$is\_args$args;  #如果需要修改从被代理服务器传来的应答头中的"Location"和"Refresh"字段，可以用这个指令设置。  proxy\_redirect off;  #添加头信息  proxy\_set\_header HOST $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  #反向代理，proxy\_pass转发的路径后是否带 “/”有不同的结果  #带"/"结尾，nginx会把匹配到的路径当做根路径,不带"/",nginx会把匹配到的路径当做一个目录  #例如location /web/{  proxy\_pass http://webServer/ } 当访问http://proxy\_pass/web，会跳转到http://server/  #例如location /web/{  proxy\_pass http://webServer } 当访问http://proxy\_pass/web，会跳转到http://server/web  proxy\_pass http://webServer/web;  #用来定义故障转移，当后台服务器返回500,502,503,504和花草还是错误时候，  #自动将请求转发到upstream负载均衡的另外几台服务器上  proxy\_next\_upstream http\_500 http\_502 http\_503 error timeout invalid\_header;  }  location ^~ /web/{  proxy\_pass http://webServer;  proxy\_redirect off;  proxy\_set\_header HOST $host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  }    }  } |

# 反向代理

## 正向代理与反向代理

**正向代理**：已知要访问本地无法访问到的服务器，我们可以连接正向代理服务器，通过它去访问远程服务器，并把信息返回给本地用户。

**反向代理**：本地用户直接访问反向代理服务器，反向代理服务器根据访问的请求，向远端服务器获取自己需要的内容，然后返回给本地用户。本地用户并不知道远端服务器的存在。

**正向代理和反向代理的区别**：

正向代理的典型用途是为在防火墙内的局域网客户端提供访问Internet的途径，可以使用缓冲特性减少网络使用率。另外，从安全性来讲：正向代理允许客户端通过它访问任意网站并且隐藏客户端自身，因此你必须采取安全措施以确保仅为经过授权的客户端提供服务。

反向代理的典型用途是将防火墙后面的服务器提供给Internet用户访问。

反向代理还可以为后端的多台服务器提供负载平衡，或为后端较慢的服务器提供缓冲服务。另外，反向代理还可以启用高级URL策略和管理技术。另外，从安全上方面反向代理对外都是透明的，访问者并不知道自己访问的是一个代理。

## Nginx代理模块

#这个指令设置被代理服务器的地址和被映射的URI，地址可以使用主机名或IP加端口号的形式,要注意：url后面有无"/"的区别，有/代表会把正则配置的uri变成/，没有使用/结尾在uri后面加上匹配的字符。语法：proxy\_pass url

|  |
| --- |
| #实际访问的地址是http://172.16.130.124:8080/  location ^~ /3t-admin/ {  proxy\_pass http://172.16.130.124:8080/;  } |

#指令允许将发送到被代理服务器的请求头重新定义或者增加一些字段。

proxy\_set\_header

## location语法规则

语法规则： location [=|~|~\*|^~] /uri/ { … }

* = 开头表示精确匹配
* ^~ 开头表示uri以某个常规字符串开头，理解为匹配 url路径即可。nginx不对url做编码，因此请求为/static/20%/aa，可以被规则^~ /static/ /aa匹配到（注意是空格）。
* ~ 开头表示区分大小写的正则匹配
* ~\* 开头表示不区分大小写的正则匹配
* !~和!~\*分别为区分大小写不匹配及不区分大小写不匹配的正则
* / 通用匹配，任何请求都会匹配到。

多个location配置的情况下匹配顺序为：

1. 普通location(无顺序)：严格匹配 =(一旦匹配，不会在匹配正则location)，其次最大前缀匹配^~(一旦匹配，不会在匹配正则location)，无任何前缀匹配（会继续匹配正则location）
2. 其次是按正则匹配的配置顺序进行正则匹配，匹配到了就停止匹配。
3. 最后是交给 / 通用匹配。

## root指令和alias的区别

alias指定的目录是当前目录，而root指定的是根目录，然后根据uri对应目录一级一级往下找。

|  |
| --- |
| #访问/img/目录里面的文件时，nginx自动去/var/www/image/目录找文件location /img/ {  alias /var/www/image/;  }  #访问url：http://ip:port /admin/static/upload/image/appfaq/source.jpg  会到目录/var/ftp/3tbetter/admin/static/upload/image/appfaq/查找  location ~ .\*\.(htm|html|js|css|less|json|gif|jpg|jpeg|png|bmp|swf|ioc|rar|zip|txt|flv|mid|doc|ppt|pdf|xls|mp3|wma|apk|zip|rar|map)$ {  root /var/ftp/3tbetter/;  }  #访问/img/目录的文件时，nginx会去/var/www/image/img/目录下找文件。  location /img/ {  root /var/www/image;  } |

## Nginx代理常用功能

|  |
| --- |
| location /{  #目录列表  autoindex on;  #ip访问权限，需要使用ngx\_http\_access\_module模块  deny 192.168.55.80;  allow 192.16.88.0/16;  deny all;  }  #访问目录控制权限  location ~^ /(WEB-INF)/{  deny all;  }  #访问文件控制权限  location ~\* \.(txt|doc){  root /home/k12;  deny all;  } |

# Nginx之URL重写

这个模块允许使用正则表达式重写URI（需PCRE库），并且可以根据相关变量重定向和选择不同的配置。如果这个指令在server字段中指定，那么将在被请求的location确定之前执行，如果在指令执行后所选location中有其他的重写规则，那么它们也被执行。如果在location中执行这个指令产生了新的URI，那么location又一次确定了新的URI。这样的循环可以最多执行10次，超过以后nginx将返回500错误。

## If命令

|  |
| --- |
| 语法：if (condition) { … }  默认值：none  使用字段：server, location  匹配条件： |
| 正则表达式的匹配  ~为区分大小写的匹配。  ~\*不区分大小写的匹配（firefox匹配FireFox）。   !~和!~\*意为“不匹配的”。  注意：正则表达式的一部分可以用圆括号，方便之后按照顺序用$1-$9来引用。  文件和目录匹配   使用-f和!-f检查一个文件是否存在。   使用-d和!-d检查一个目录是否存在。   使用-e和!-e检查一个文件，目录或者软链接是否存在。   使用-x和!-x检查一个文件是否为可执行文件。 |

常用的内置变量：

|  |
| --- |
| $args：变量和请求中的参数相同  $document\_root：相当于当前请求的root命令指定的值  $uri：当前request的URI  $document\_uri：同$uri  $host:和请求头中的host一样  $limit\_rate：用来设置限制连接速率  $request\_method：等同于request的method，get或者post  $remote\_addr：表示客户端ip地址  $remote\_port：表示客户端端口号  $remote\_user：表示用户名  $request\_filename：当前请求的文件的路径名。root或者alias与URI的request组合而成  $request\_uri：有参数的完整的初始URI  $query\_string：此变量和$args一样  $server\_name：表示请求到达服务器的服务器名字  $server\_port：表示请求到达服务器的端口号。 |

## rewrite命令

|  |
| --- |
| 语法：rewrite regex replacement flag  默认值：none  使用字段：server, location, if  按照相关的正则表达式与字符串修改URI，指令按照在配置文件中出现的顺序执行。可以在重写指令后面添加标记。  如果替换的字符串以http://开头，请求将被重定向，并且不再执行多余的rewrite指令。尾部的标记(flag)可以是以下的值：   * last - 完成重写指令，之后搜索相应的URI或location，url不变。 * break - 完成重写指令，不在匹配后面的规则，url不变。 * redirect - 返回302临时重定向，如果替换字段用http://开头则被使用。 * permanent - 返回301永久重定向,浏览器地址栏中url变成跳转URL |
| location /download/ {  rewrite ^(/download/.\*)/media/(.\*)\..\*$ $1/mp3/$2.mp3 break;  rewrite ^(/download/.\*)/audio/(.\*)\..\*$ $1/mp3/$2.ra break;  return 403;  } |

## set命令

|  |
| --- |
| set命令可以为设置一个变量并为其赋值，其值可以是文本，变量或他们的 组合。不能使用set设置$http\_xxx头部变量的值  语法：set var val  默认值：none  使用字段：server，location，if |

## break命令

|  |
| --- |
| 表示完成当前设置的规则后不在匹配后面的重写规则  语法：break  默认值：none  使用字段：server , location ,if |

## return 命令

|  |
| --- |
| 语法：return code  默认值：none  使用字段：server, location, if  这个指令结束执行配置语句并为客户端返回状态代码，可以使用下列的值：  204，400，402-406，408，410, 411, 413, 416与500-504。  此外，非标准代码444将关闭连接并且不发送任何的头部。 |

# Nginx之页面缓存

proxy\_set\_header：设置由后端的服务器获取用户的主机名或者真实IP地址，以及代理者的真实IP地址。

client\_body\_buffer\_size：用于指定客户端请求主体缓冲区大小，可以理解为先保存到本地再传给用户。

proxy\_connect\_timeout：表示与后端服务器连接的超时时间，即发起握手等候响应的超时时间。

proxy\_send\_timeout：表示后端服务器的数据回传时间，即在规定时间之内后端服务器必须传完所有的数据，否则，Nginx将断开这个连接。

proxy\_read\_timeout：设置Nginx从代理的后端服务器获取信息的时间，表示连接建立成功后，Nginx等待后端服务器的响应时间，其实是Nginx已经进入后端的排队之中等候处理的时间。

proxy\_buffer\_size：设置缓冲区大小， 默认,该缓冲区大小等于指令proxy\_buffers设置的大小。

proxy\_buffers：设置缓冲区的数量和大小。nginx从代理的后端服务器获取的响应信息，会放置到缓冲区。

proxy\_busy\_buffers\_size：用于设置系统很忙时可以使用的proxy\_buffers大小，官方推荐的大小为proxy\_buffers\*2。

proxy\_temp\_file\_write\_size：指定proxy缓存临时文件的大小。

## proxy\_cache\_path

语法：proxy\_cache\_path path [levels=number] keys\_zone=zone\_name:zone\_size [inactive=time] [max\_size=size];

默认值：None

使用字段：http

指令指定缓存的路径和一些其他参数，缓存的数据存储在文件中，并且使用代理url的哈希值作为关键字与文件名。levels参数指定缓存的子目录数，例如：

proxy\_cache\_path /data/nginx/cache levels=1:2 keys\_zone=one:10m;

## proxy\_cache

语法：proxy\_cache zone\_name;

默认值：None

使用字段：http, server, location

设置一个缓存区域的名称，一个相同的区域可以在不同的地方使用。

# Nginx之负载均衡

upstream是Nginx的HTTP Upstream模块，这个模块通过一个简单的调度算法来实现客户端IP到后端服务器的负载均衡。在上面的设定中，通过upstream指令指定了一个负载均衡器的名称test.net。这个名称可以任意指定，在后面需要用到的地方直接调用即可。

## upstream 支持的负载均衡算法

***轮询（默认）。***每个请求按时间顺序逐一分配到不同的后端服务器，如果后端某台服务器宕机，故障系统被自动剔除，使用户访问不受影响。Weight 指定轮询权值，Weight值越大，分配到的访问机率越高，主要用于后端每个服务器性能不均的情况下。

***ip\_hash。***每个请求按访问IP的hash结果分配，这样来自同一个IP的访客固定访问一个后端服务器，有效解决了动态网页存在的session共享问题。

***fair。***这是比上面两个更加智能的负载均衡算法。此种算法可以依据页面大小和加载时间长短智能地进行负载均衡，也就是根据后端服务器的响应时间来分配请求，响应时间短的优先分配。Nginx本身是不支持fair的，如果需要使用这种调度算法，必须下载Nginx的upstream\_fair模块。

***url\_hash。***此方法按访问url的hash结果来分配请求，使每个url定向到同一个后端服务器，可以进一步提高后端缓存服务器的效率。Nginx本身是不支持url\_hash的，如果需要使用这种调度算法，必须安装Nginx 的hash软件包。

## upstream 支持的状态参数

在HTTP Upstream模块中，可以通过server指令指定后端服务器的IP地址和端口，同时还可以设定每个后端服务器在负载均衡调度中的状态。常用的状态有：

* down，表示当前的server暂时不参与负载均衡。
* backup，预留的备份机器。当其他所有的非backup机器出现故障或者忙的时候，才会请求backup机器，因此这台机器的压力最轻。
* max\_fails，允许请求失败的次数，默认为1。当超过最大次数时，返回proxy\_next\_upstream 模块定义的错误。
* fail\_timeout，在经历了max\_fails次失败后，暂停服务的时间。max\_fails可以和fail\_timeout一起使用。

注，当负载调度算法为ip\_hash时，后端服务器在负载均衡调度中的状态不能有backup和weight。

# 其他问题

## Nginx内核优化

|  |
| --- |
| #参数用来设定timewait的数量，默认是180000，这里设为6000。  net.ipv4.tcp\_max\_tw\_buckets = 6000  #选项用来设定允许系统打开的端口范围。  net.ipv4.ip\_local\_port\_range = 1024 65000  #选项用于设置启用timewait快速回收。  net.ipv4.tcp\_tw\_recycle = 1  #选项用于设置开启重用，允许将TIME-WAIT sockets重新用于新的TCP连接。  net.ipv4.tcp\_tw\_reuse = 1  #选项用于设置开启SYN Cookies，当出现SYN等待队列溢出时，启用cookies进行处理。  net.ipv4.tcp\_syncookies = 1  #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，需要结合并发请求数来调节此值。  net.core.somaxconn = 262144  #选项表示当每个网络接口接收数据包的速率比内核处理这些包的速率快时，  #允许发送到队列的数据包的最大数目。  net.core.netdev\_max\_backlog = 262144  #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，  #孤立连接将立即被复位并打印出警告信息。这个限制只是为了防止简单的DoS攻击。不能过分依靠这个限制甚至人为减小这个值，  #更多的情况是增加这个值。  net.ipv4.tcp\_max\_orphans = 262144  #选项用于记录那些尚未收到客户端确认信息的连接请求的最大值。对于有128MB内存的系统而言，  #此参数的默认值是1024，对小内存的系统则是128。  net.ipv4.tcp\_max\_syn\_backlog = 262144  #参数的值决定了内核放弃连接之前发送SYN+ACK包的数量。  net.ipv4.tcp\_synack\_retries = 1  #选项表示在内核放弃建立连接之前发送SYN包的数量。  net.ipv4.tcp\_syn\_retries = 1  #选项决定了套接字保持在FIN-WAIT-2状态的时间。默认值是60秒。正确设置这个值非常重要，  #有时候即使一个负载很小的Web服务器，也会出现因为大量的死套接字而产生内存溢出的风险。  net.ipv4.tcp\_fin\_timeout = 1  #选项表示当keepalive启用的时候，TCP发送keepalive消息的频度。默认值是2（单位是小时）。  net.ipv4.tcp\_keepalive\_time = 30 |
| 在/etc/sysctl.conf中：  net.ipv4.tcp\_max\_tw\_buckets = 6000  net.ipv4.ip\_local\_port\_range = 1024 65000  net.ipv4.tcp\_tw\_recycle = 1  net.ipv4.tcp\_tw\_reuse = 1  net.ipv4.tcp\_syncookies = 1  net.core.somaxconn = 262144  net.core.netdev\_max\_backlog = 262144  net.ipv4.tcp\_max\_orphans = 262144  net.ipv4.tcp\_max\_syn\_backlog = 262144  net.ipv4.tcp\_synack\_retries = 1  net.ipv4.tcp\_syn\_retries = 1  net.ipv4.tcp\_fin\_timeout = 1  net.ipv4.tcp\_keepalive\_time = 30  执行：/sbin/ sysctl -p |

## Java代码如何获取远程主机ip

Nginx中：

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

tomcat中：

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs" prefix="localhost\_access\_log." suffix=".txt" pattern='commont' resolveHosts="false"/>把patten改成

pattern='%{X-Forwarded-For}i %h %l %u %t "%r" %s %b'

java代码：

|  |
| --- |
| public static String getAddr(HttpServletRequest request) {  String ip = request.getHeader("x-forwarded-for");  if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {  ip = request.getHeader("Proxy-Client-IP");  }  if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {  ip = request.getHeader("WL-Proxy-Client-IP");  }  if (ip == null || ip.length() == 0 || "unknown".equalsIgnoreCase(ip)) {  ip = request.getRemoteAddr();  }  return ip;  } |

## Nginx实现无缝tomcat部署

|  |
| --- |
| --部署  #!/bin/bash  . /etc/profile  tomcatDir=/usr/local/tomcat7  FTP\_DIR=/home/k12/ftp/sshPackage  prepTomcatHosts=$(cat /home/k12/prepTomcatHosts)  otherTocmatHosts=$(cat /home/k12/otherTocmatHosts)  pacDir=/mnt/tongguan/bakup\_war  uploadPacDate=$(/bin/date +%Y%m%d%H%M)  #将所有的war包放到每台服务器的/mnt/tongguan/bakup\_war  if test -e $FTP\_DIR/k12-web-1.0.0-SNAPSHOT-PROD\_ALL.war;then  pscp -h /etc/pssh/all\_hosts $FTP\_DIR/k12-web-1.0.0-SNAPSHOT-PROD\_ALL.war $pacDir/ROOT.war.$uploadPacDate  fi  if test -e $FTP\_DIR/k12-web-1.0.0-SNAPSHOT-PROD\_ZJ.war;then  pscp -h /etc/pssh/zj\_hosts $FTP\_DIR/k12-web-1.0.0-SNAPSHOT-PROD\_ZJ.war $pacDir/ROOT.war.$uploadPacDate  fi  if test -e $FTP\_DIR/k12-web-1.0.0-SNAPSHOT-PROD\_SC.war;then  pscp -h /etc/pssh/sc\_hosts $FTP\_DIR/k12-web-1.0.0-SNAPSHOT-PROD\_SC.war $pacDir/ROOT.war.$uploadPacDate  fi  echo '---upload package successful!---'  echo '----------使用第一个tomcat,部署其他两个tomcat-------------'  for host in $otherTocmatHosts  do  mussh -h 218.244.141.167 -m -c "sed -i 's/'\"server ${host}\"'/'\"#server ${host}\"'/g' /usr/local/nginx/conf/nginx.conf"  done  mussh -h 218.244.141.167 -m -c " /usr/local/nginx/sbin/nginx -s reload"  #部署all,zj,sc中后两个tomcat服务器  mussh -H /home/k12/otherTocmatHosts -m -c "killall -9 java ; rm -rf $tomcatDir/webapps/ROOT\* ; mv $pacDir/ROOT.war.$uploadPacDate $tomcatDir/webapps/ROOT ; $tomcatDir/bin/startup.sh"  echo '等待tomcat重启完成，大概需要60s......'  sleep 60  #修改nginx.conf(启用后两个tomcat)  echo '----------使用其他两台tomcat,部署预备tomcat-------------'  for host in $otherTocmatHosts  do  mussh -h 218.244.141.167 -m -c "sed -i 's/'\"##\*server ${host}\"'/'\"server ${host}\"'/g' /usr/local/nginx/conf/nginx.conf"  done  for host in $prepTomcatHosts  do  mussh -h 218.244.141.167 -m -c "sed -i 's/'\"server ${host}\"'/'\"#server ${host}\"'/g' /usr/local/nginx/conf/nginx.conf"  done  mussh -h 218.244.141.167 -m -c " /usr/local/nginx/sbin/nginx -s reload"  #部署预备的tomcat  mussh -H /home/k12/prepTomcatHosts -m –c "killall -9 java ; rm -rf $tomcatDir/webapps/ROOT\* ; mv $pacDir/ROOT.war.$uploadPacDate $tomcatDir/webapps/ROOT ; $tomcatDir/bin/startup.sh"  echo '等待tomcat重启完成，大概需要60s......'  sleep 60  #修改nginx.conf(启用所有的tomcat)  for host in $prepTomcatHosts  do  mussh -h 218.244.141.167 -m -c "sed -i 's/'\"##\*server ${host}\"'/'\"server ${host}\"'/g' /usr/local/nginx/conf/nginx.conf"  done  mussh -h 218.244.141.167 -m -c " /usr/local/nginx/sbin/nginx -s reload"  echo '--------------------部署完毕-----------------------' |

|  |
| --- |
| --restore  #!/bin/bash  . /etc/profile  tomcatDir=/usr/local/tomcat7  FTP\_DIR=/home/k12/ftp/sshPackage  prepTomcatHosts=$(cat /home/k12/prepTomcatHosts)  otherTocmatHosts=$(cat /home/k12/otherTocmatHosts)  pacDir=/mnt/tongguan/bakup\_war  packages=$(ls /mnt/tongguan/bakup\_war)  function deploy(){  echo '----------使用第一个tomcat,部署其他两个tomcat-------------'  for host in $otherTocmatHosts  do  mussh -h 218.244.141.167 -m -c "sed -i 's/'\"server ${host}\"'/'\"#server ${host}\"'/g' /usr/local/nginx/conf/nginx.conf"  done  mussh -h 218.244.141.167 -m -c " /usr/local/nginx/sbin/nginx -s reload"  #部署all,zj,sc中后两个tomcat服务器  mussh -H /home/k12/otherTocmatHosts -m -c "killall -9 java ; rm -rf $tomcatDir/webapps/ROOT\* ; mv $pacDir/$1 $tomcatDir/webapps/ROOT ; $tomcatDir/bin/startup.sh"  echo '等待tomcat重启完成，大概需要60s......'  sleep 60  #修改nginx.conf(启用后两个tomcat)  echo '----------使用其他两台tomcat,部署预备tomcat-------------'  for host in $otherTocmatHosts  do  mussh -h 218.244.141.167 -m -c "sed -i 's/'\"##\*server ${host}\"'/'\"server ${host}\"'/g' /usr/local/nginx/conf/nginx.conf"  done  for host in $prepTomcatHosts  do  mussh -h 218.244.141.167 -m -c "sed -i 's/'\"server ${host}\"'/'\"#server ${host}\"'/g' /usr/local/nginx/conf/nginx.conf"  done  mussh -h 218.244.141.167 -m -c " /usr/local/nginx/sbin/nginx -s reload"  #部署预备的tomcat  mussh -H /home/k12/prepTomcatHosts -m -c "killall -9 java ; rm -rf $tomcatDir/webapps/ROOT\* ; mv $pacDir/$1 $tomcatDir/webapps/ROOT ; $tomcatDir/bin/startup.sh"  echo '等待tomcat重启完成，大概需要60s......'  sleep 60  #修改nginx.conf(启用所有的tomcat)  for host in $prepTomcatHosts  do  mussh -h 218.244.141.167 -m -c "sed -i 's/'\"##\*server ${host}\"'/'\"server ${host}\"'/g' /usr/local/nginx/conf/nginx.conf"  done  mussh -h 218.244.141.167 -m -c " /usr/local/nginx/sbin/nginx -s reload"  echo '--------------------部署完毕-----------------------'  }  echo "请选择需要的版本："  select version in $packages  do  deploy $version  break  done |