# Centos安装nginx

1. 先去官网下载文件，例如：nginx-1.19.2.tar.gz，然后丢在/home/software文件夹下
2. 安装gcc环境：yum install gcc-c++
3. 安装PCRE库，用于解析正则表达式：yum install -y pcre pcre-devel
4. zlib压缩和解压缩依赖：yum install -y zlib zlib-devel

SSL 安全的加密的套接字协议层，用于HTTP安全传输，也就是https：yum install -y openssl openssl-devel

1. 解压，需要注意，解压后得到的是源码，源码需要编译后才能安装：tar -zxvf nginx-1.19.2.tar.gz
2. 编译之前，先创建nginx临时目录，如果不创建，在启动nginx的过程中会报错：mkdir /var/temp/nginx -p
3. 在nginx目录，输入如下命令进行配置，目的是为了创建makefile文件，先：cd nginx-1.19.2，然后执行下面语句

|  |
| --- |
| ./configure \  --prefix=/usr/local/nginx \  --pid-path=/var/run/nginx/nginx.pid \  --lock-path=/var/lock/nginx.lock \  --error-log-path=/var/log/nginx/error.log \  --http-log-path=/var/log/nginx/access.log \  --with-http\_gzip\_static\_module \  --http-client-body-temp-path=/var/temp/nginx/client \  --http-proxy-temp-path=/var/temp/nginx/proxy \  --http-fastcgi-temp-path=/var/temp/nginx/fastcgi \  --http-uwsgi-temp-path=/var/temp/nginx/uwsgi \  --http-scgi-temp-path=/var/temp/nginx/scgi |

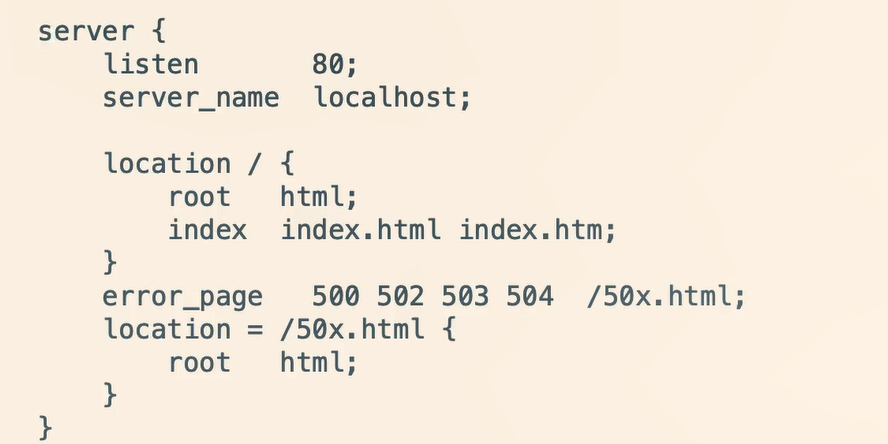


1. make编译：make
2. 安装：make install
3. 然后输入whereis nginx 查看nginx在哪，然后cd 路劲
4. 进入sbin目录
   1. 启动nginx：./nginx
   2. 停止：./nginx -s stop
   3. 重新加载：./nginx -s reload
5. 由于nginx默认端口是80，安装完毕之后直接输入端口查看是否安装成功

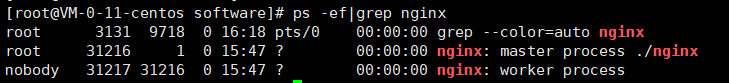
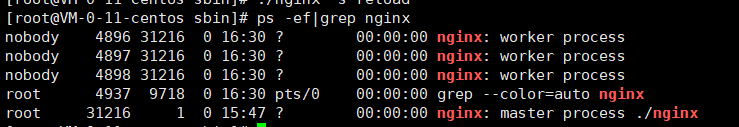


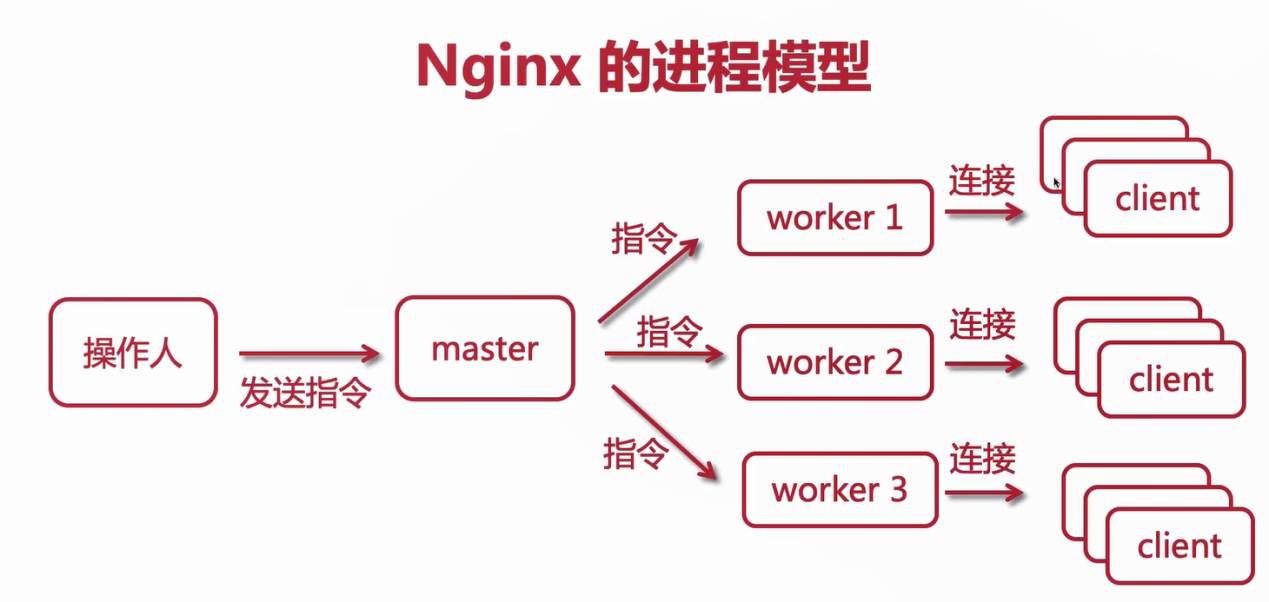
# Nginx默认首页解析过程



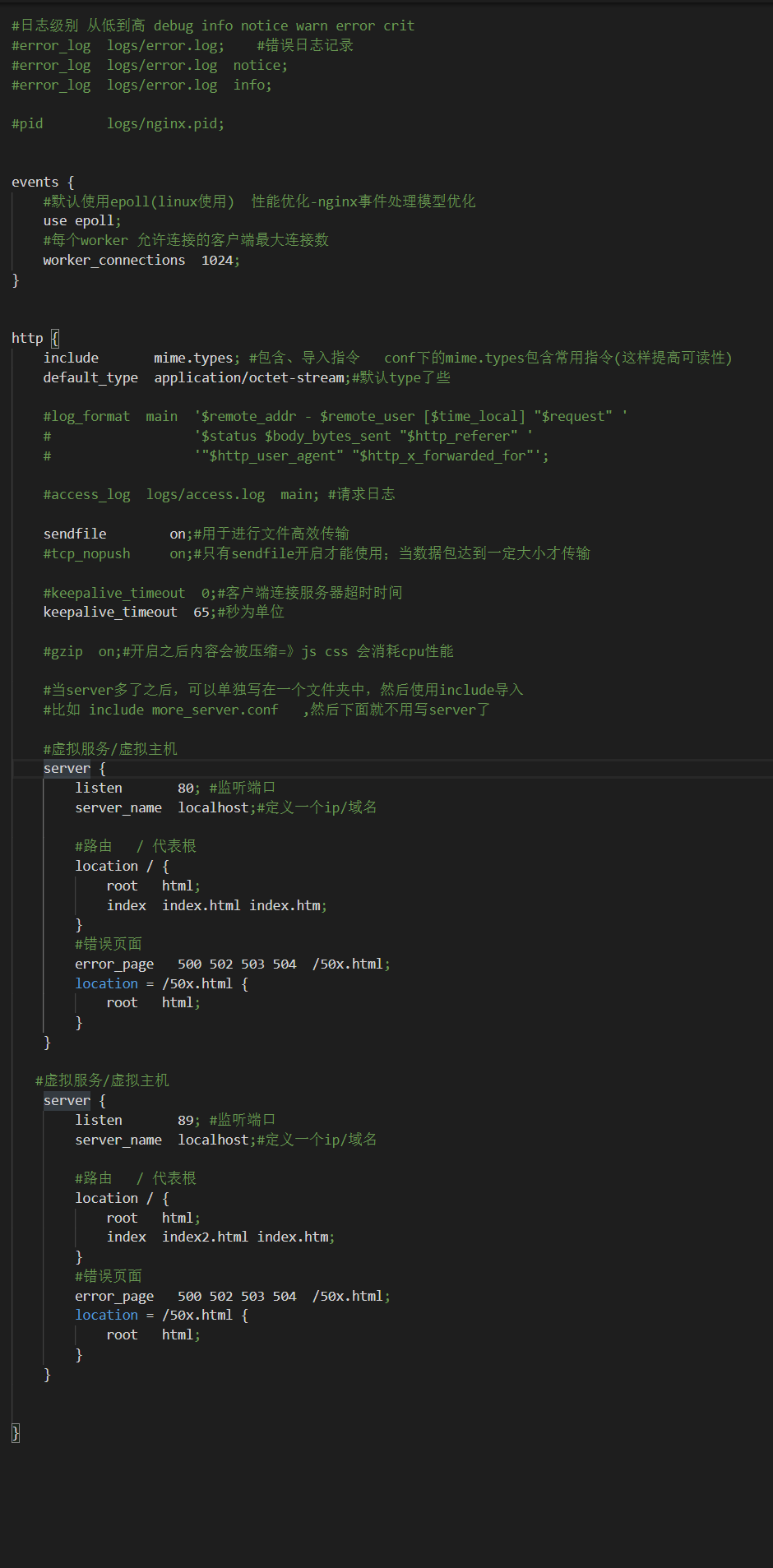


# Nginx进程模型

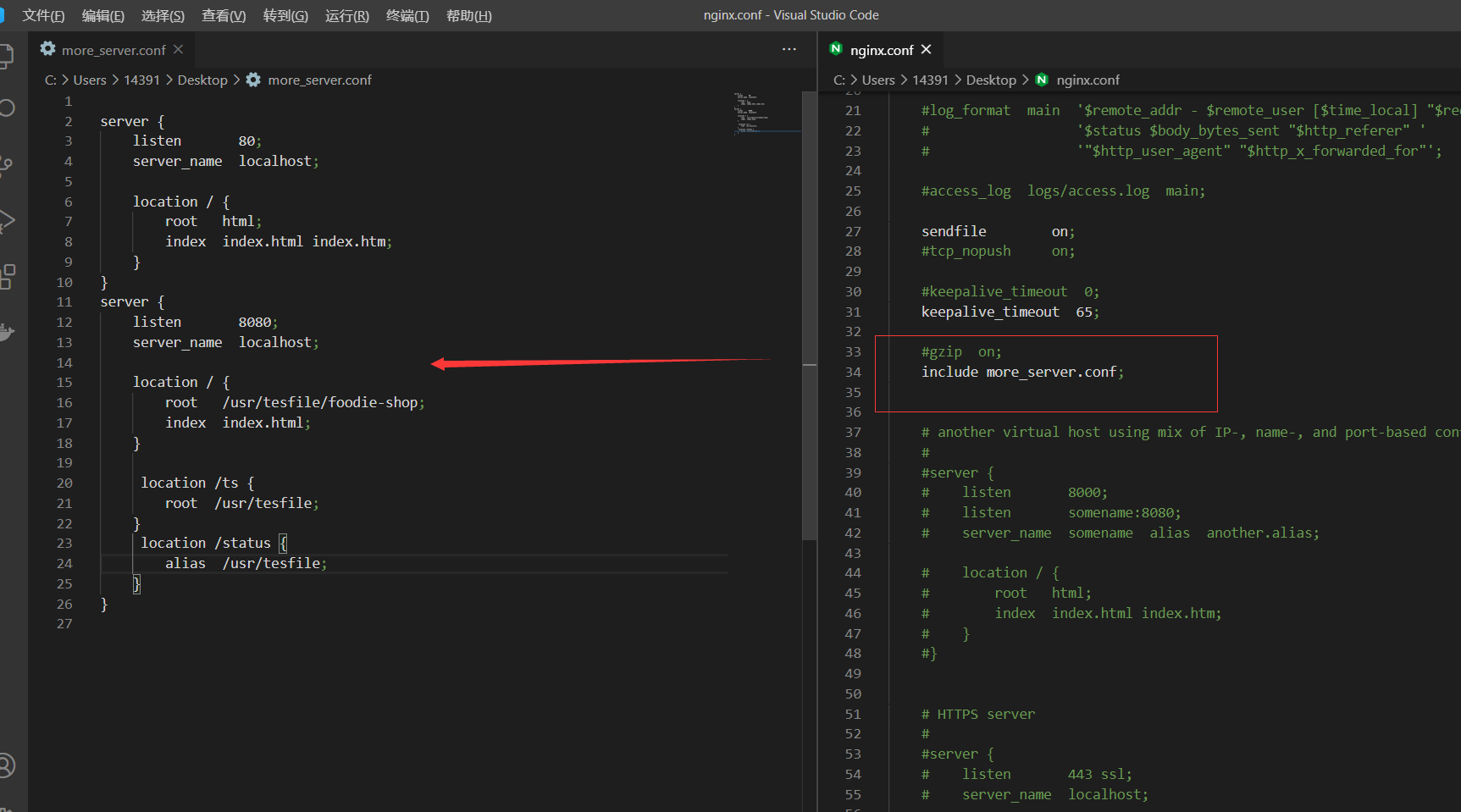
1. 启动nginx之后会有两个进程：
   1. master进程：主进程
   2. worker进程：工作进程
   3. 再任何地方输入，可以看见nginx进程信息：ps -ef|grep nginx
      1. 
   4. 修改工作进程数量：在nginx的conf文件下的nginx.conf ，修改完配置文件需要在sbin目录下重启nginx
   5. 
   6. 修改后
      1. 

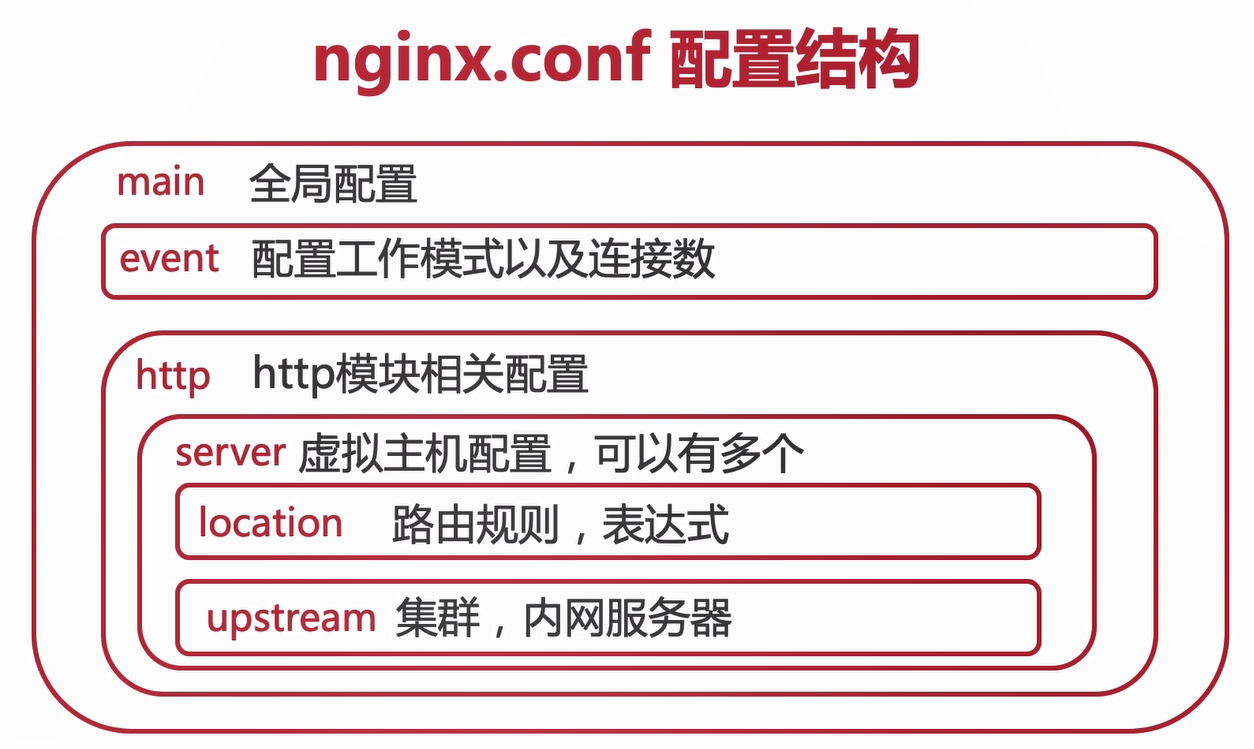


# Nginx配置文件=>nginx.conf

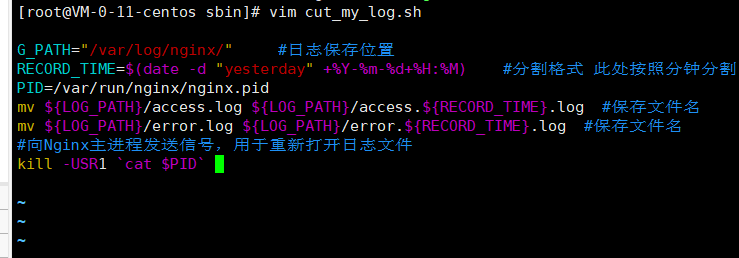


|  |
| --- |
| #user nobody; #工作进程默认nobody执行，这里可以指定为root执行=》user root  worker\_processes 1; #进程数量和cpu相关，cpu好可以配置多个  #日志级别 从低到高 debug info notice warn error crit  #error\_log logs/error.log; #错误日志记录  #error\_log logs/error.log notice;  #error\_log logs/error.log info;  #pid logs/nginx.pid;  events {  #默认使用epoll(linux使用) 性能优化-nginx事件处理模型优化  use epoll;  #每个worker 允许连接的客户端最大连接数  worker\_connections 1024;  }  http {  include mime.types; #包含、导入指令 conf下的mime.types包含常用指令(这样提高可读性)  default\_type application/octet-stream;#默认type了些  #log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '  # '$status $body\_bytes\_sent "$http\_referer" '  # '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  #access\_log logs/access.log main; #请求日志  sendfile on;#用于进行文件高效传输  #tcp\_nopush on;#只有sendfile开启才能使用；当数据包达到一定大小才传输  #keepalive\_timeout 0;#客户端连接服务器超时时间  keepalive\_timeout 65;#秒为单位  #gzip on;#开启之后内容会被压缩=》js css 会消耗cpu性能  #当server多了之后，可以单独写在一个文件夹中，然后使用include导入  #比如 include more\_server.conf; ,然后下面就不用写server了  #虚拟服务/虚拟主机  server {  listen 80; #监听端口  server\_name localhost;#定义一个ip/域名  #路由 / 代表根  location / {  root html;  index index.html index.htm;  }  #错误页面  error\_page 500 502 503 504 /50x.html;  location = /50x.html {  root html;  }  }  #虚拟服务/虚拟主机  server {  listen 89; #监听端口  server\_name localhost;#定义一个ip/域名  #路由 / 代表根  location / {  root html;  index index2.html index.htm;  }  #错误页面  error\_page 500 502 503 504 /50x.html;  location = /50x.html {  root html;  }  }    } |

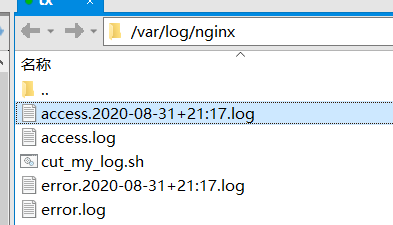




# Nginx日志切割

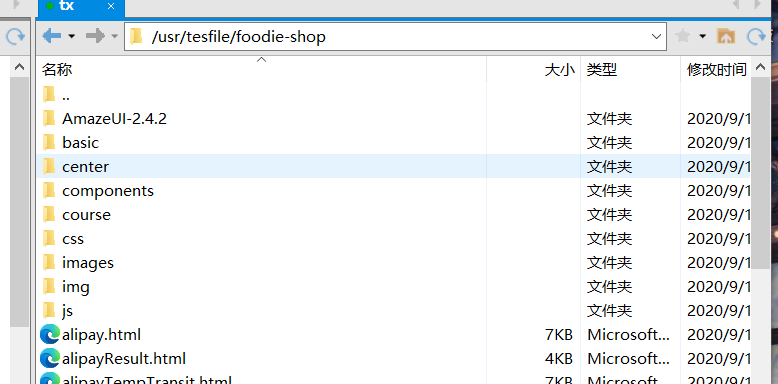
1. 进入nginx的sbin目录下，然后输入 vim cut\_my\_log.sh创建一个可执行文件
2. 然后输入
   1. 

|  |
| --- |
| #!/bin/bash  LOG\_PATH="/var/log/nginx/" #日志保存位置  RECORD\_TIME=$(date -d "yesterday" +%Y-%m-%d+%H:%M) #分割格式 此处按照分钟分割  PID=/var/run/nginx/nginx.pid  mv ${LOG\_PATH}/access.log ${LOG\_PATH}/access.${RECORD\_TIME}.log #access.log重命名  mv ${LOG\_PATH}/error.log ${LOG\_PATH}/error.${RECORD\_TIME}.log #error.log重命名  #向Nginx主进程发送信号，用于重新打开日志文件  kill -USR1 `cat $PID` |

1. 为 cut\_my\_log.sh 添加可执行的权限：chmod +x cut\_my\_log.sh (chmod +x 文件名)
2. 然后输入：./cut\_my\_log.sh 查看结果
   1. 
3. 启用nginx定时任务
   1. 先输入： yum install crontabs
   2. 查看有无定时任务： crontab -l
   3. 创建定时任务： crontab -e
      1. 在其中输入 \*/1 \* \* \* \* /usr/local/nginx/sbin/cut\_my\_log.sh
   4. 然后重启：service crond restart

# 静态文件配置(centos)

配置html页面

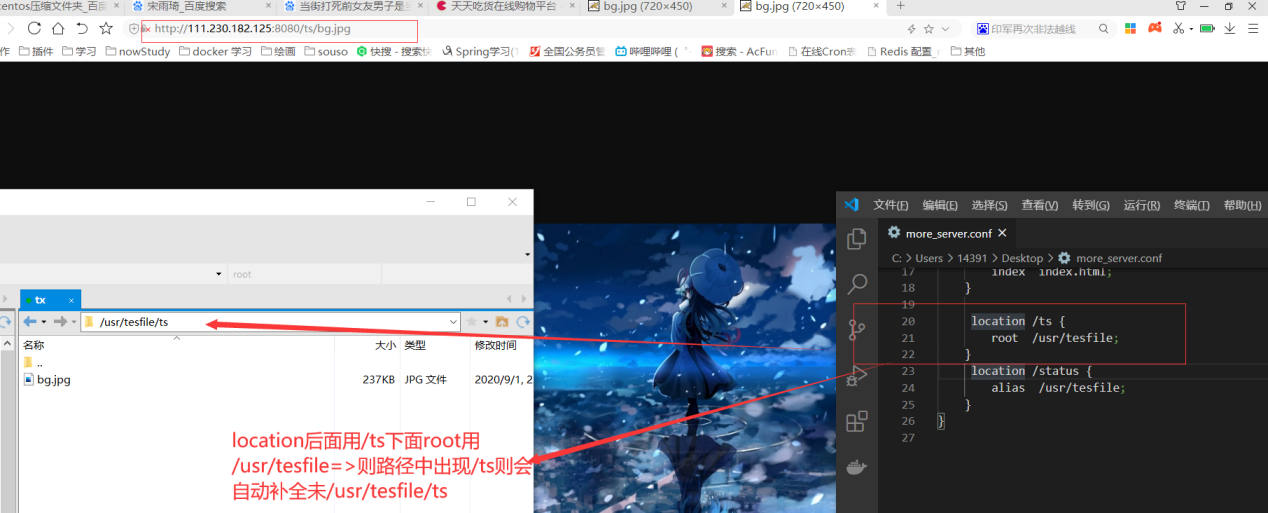


(下列中 location / 代表根目录)

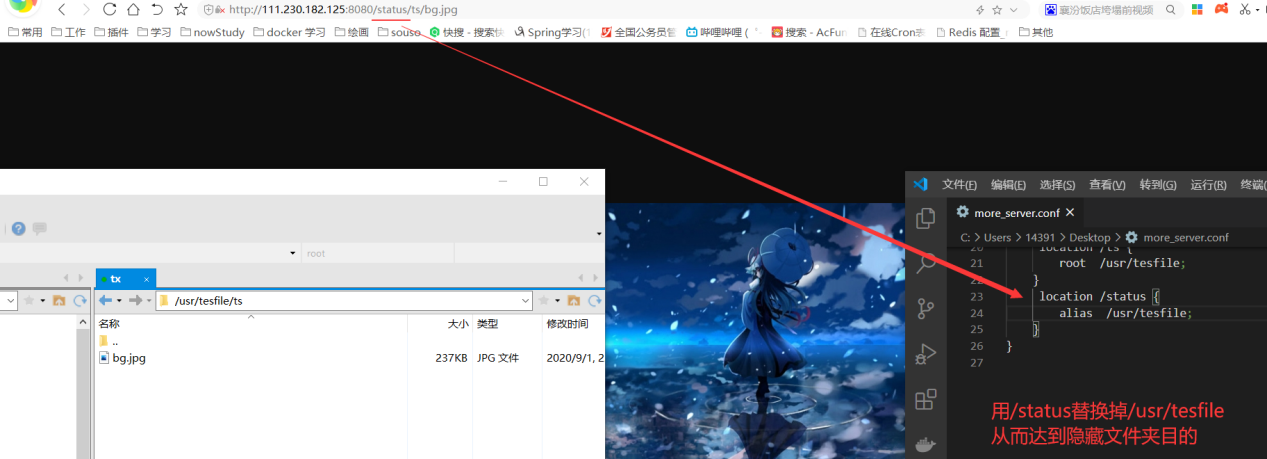




静态其他文件(图片等等)



alias隐藏路径



# 压缩文件 gzip

开启后消耗服务器cpu



# Nginx匹配规则

## 一 Nginx的location语法

|  |  |
| --- | --- |
| 1 | location [=|~|~\*|^~] /uri/ { … } |

=         严格匹配。如果请求匹配这个location，那么将停止搜索并立即处理此请求

~         区分大小写匹配(可用正则表达式)

~\*       不区分大小写匹配(可用正则表达式)

!~       区分大小写不匹配

!~\*     不区分大小写不匹配

^~      如果把这个前缀用于一个常规字符串,那么告诉nginx 如果路径匹配那么不测试正则表达式

示例1：

location / { }

匹配任意请求

 示例2：

location ~\* .(gif|jpg|jpeg)$ ｛

rewrite .(gif|jpg|jpeg)$ /logo.png;

｝

不区分大小写匹配任何以gif、jpg、jpeg结尾的请求，并将该请求重定向到 /logo.png请求

示例3：

location ~ ^.+\.txt$ {

root /usr/local/nginx/html/;

}

区分大小写匹配以.txt结尾的请求，并设置此location的路径是/usr/local/nginx/html/。也就是以.txt结尾的请求将访问/usr/local/nginx/html/ 路径下的txt文件

## 二 alias与root的区别

root    实际访问文件路径会拼接URL中的路径

alias   实际访问文件路径不会拼接URL中的路径

示例如下：

location ^~ /sta/ {

alias /usr/local/nginx/html/static/;

}

请求：http://test.com/sta/sta1.html

实际访问：/usr/local/nginx/html/static/sta1.html 文件

location ^~ /tea/ {

root /usr/local/nginx/html/;

}

# Nginx跨域问题

在server的server\_name下写入以下代码

|  |
| --- |
| #允许跨域请求的域，\*代表所有  add\_header 'Access-Control-Allow-Origin' \*;  #允许带上cookie请求  add\_header 'Access-Control-Allow-Credentials' 'true';  #允许请求的方法，比如 GET/POST/PUT/DELETE  add\_header 'Access-Control-Allow-Methods' \*;  #允许请求的header  add\_header 'Access-Control-Allow-Headers' \*; |



# 防盗链

#对源站点验证 (如果不是baidu.com 则会返回404)

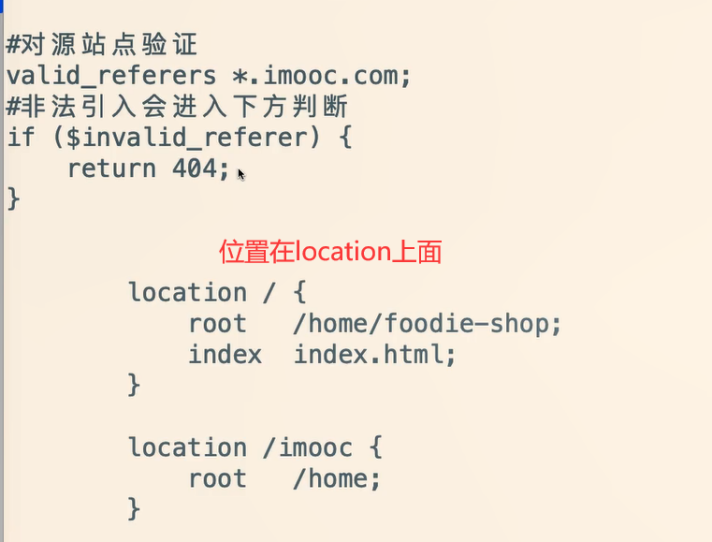
valid\_referers \*.baidu.com;

#非法引入会进入下方判断

if ($invalid\_referer) {

return 404;

}



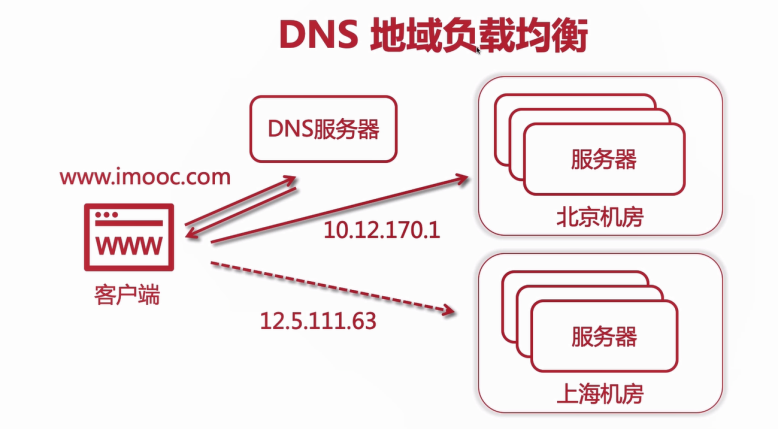
# 负载均衡

## 四层负载均衡

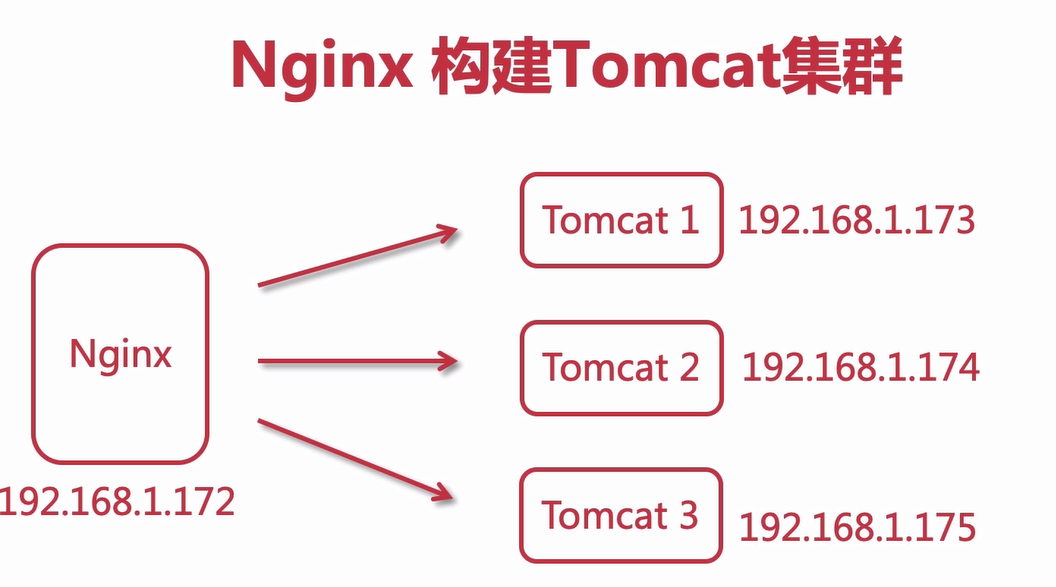
* F5硬负载均衡：重点用户硬件负载方面
* LVS四层负载均衡
* Haproxy四层负载均衡
* Nginx四层负载均衡

## 七层负载均衡

* Nginx七层负载均衡
* Haproxy七层负载均衡
* Apache七层负载均衡

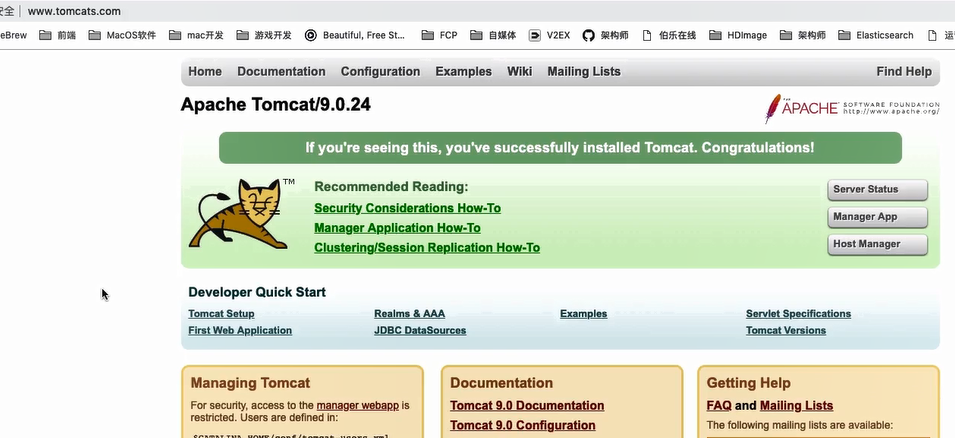


## Nginx构建集群



然后在server下方配置



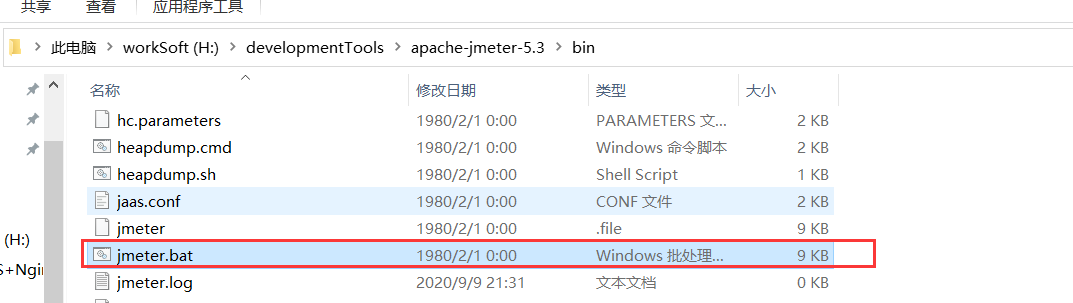


# Jmeter测试

先下载软件

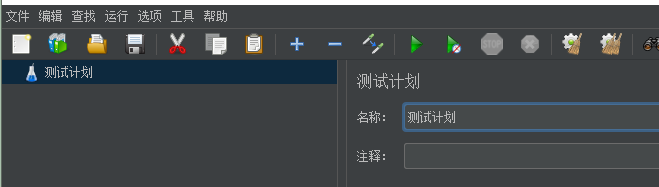


然后在bin文件夹下启动 jmeter.bat

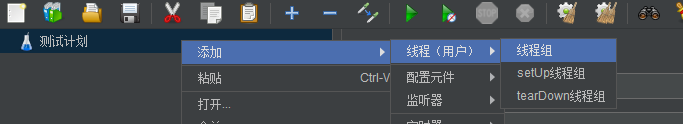


打开软件之后开始新建测试

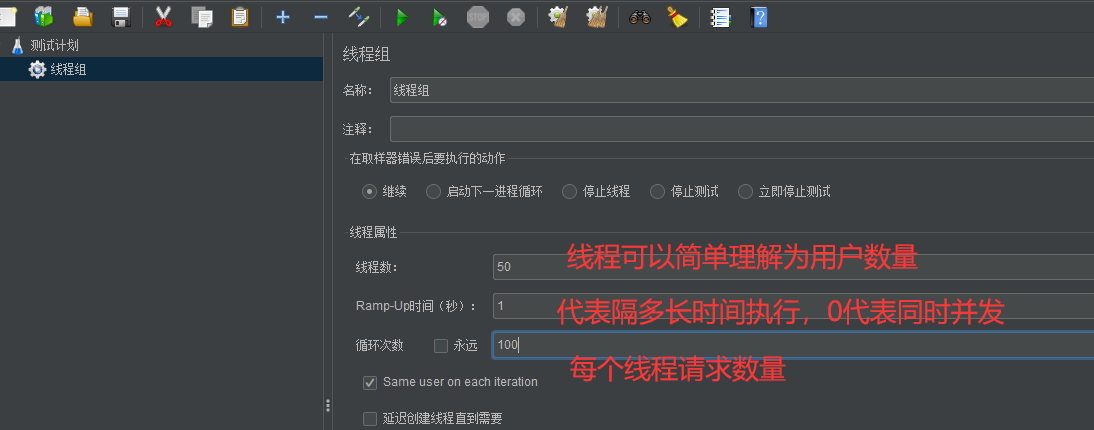




然后在【测试计划】右键-选择 线程组



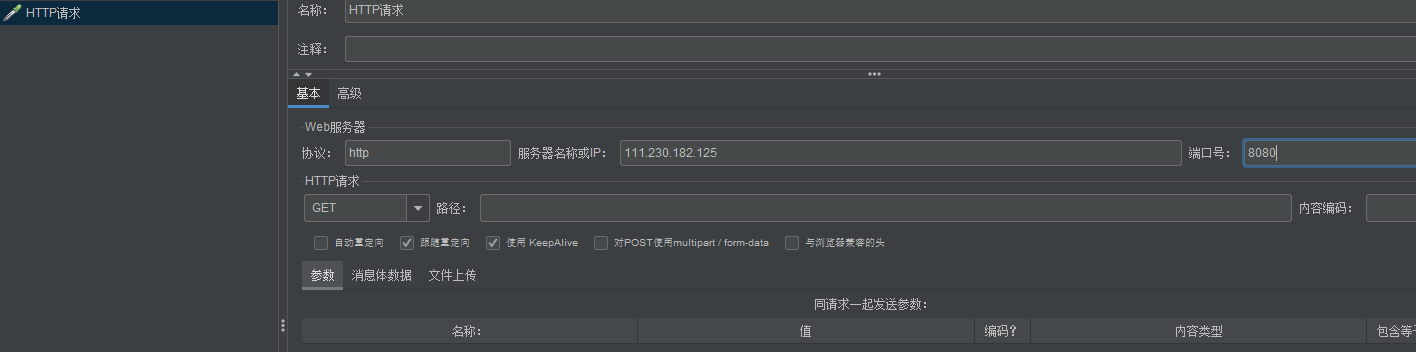
然后设置基础数据



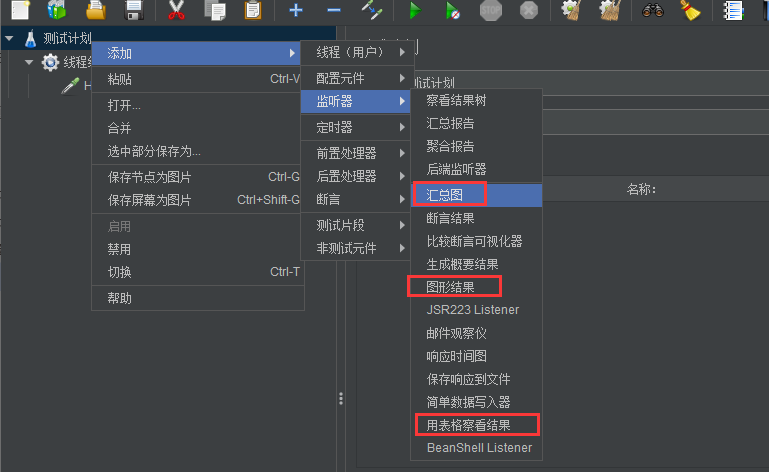
设置完成之后添加取样器

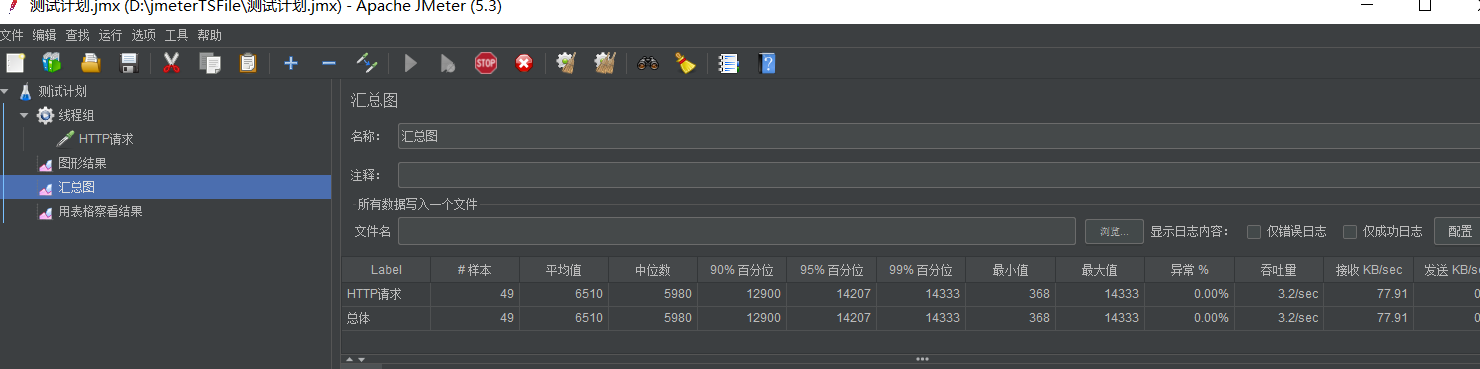


然后设置相对应参数



设置完成之后在 【测试计划】 右键添加监听器，根据自己需要添加





# 部署nginx

# 看到