# 4.nginx

## 4.1.什么是nginx

早期由俄国团队开发与apache服务器抗衡的可以处理动静分离，处理负载均衡的http反向代理服务器。

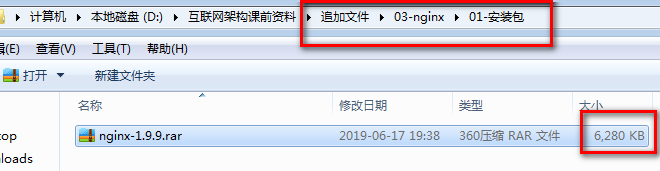
动静分离：动态资源访问tomcat，静态资源（html、css、js）可以存储在nginx中

负载均衡：通过nginx决定访问的后端服务器逻辑

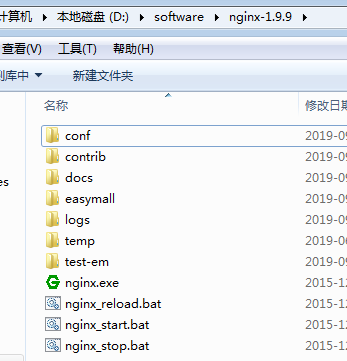
## 4.2.安装和启动nginx（window环境）

### 4.2.1.安装

将课前资料追加文件中的nginx安装包解压到没有中文、空格的目录下（d:/software）



目录结构：



conf：配置文件的目录（后面会修改里面的nginx.conf）

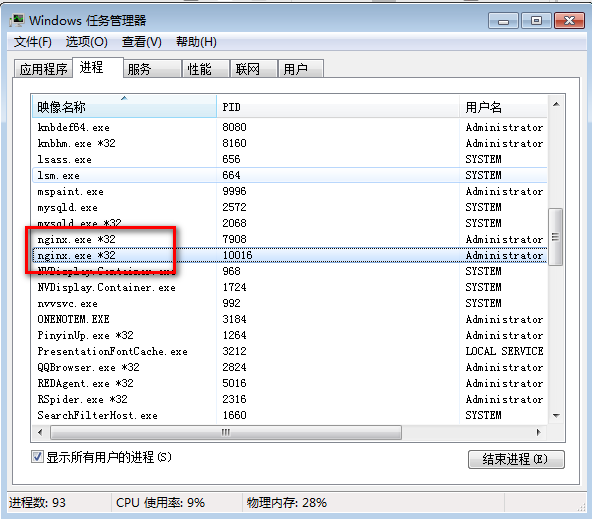
easymall：保存了easymall的静态文件

test-eam:保存了测试项目的静态文件

logs：记录日志的目录，access.log记录了访问nginx的请求，error.log记录启动运行的各种错误

### 4.2.2.启动和停止nginx

**启动：**双击根目录下的nginx\_start.bat，启动任务管理器查看，有两个nginx的进程，则说明启动成功



如果启动失败，则可以到logs下的error.log查看报错原因

例如报错：2019/09/25 14:37:34 [emerg] 1388#9556: bind() to 0.0.0.0:80 failed (10013: An attempt was made to access a socket in a way forbidden by its access permissions)

启动80端口失败，原因是tomcat没有关，占用着80端口

**停止：**双击nginx\_stop.bat则停止

重启不要使用nginx\_reload.bat，先执行stop，查看进程是否消失，然后再start

## 4.3.nginx入门案例-转发功能

### 4.3.1.转发功能配置

将order-user的系统启动在8090端口，通过www.ou.com实现访问order-user系统

首先，将order-user中pom文件的tomcat插件中的80端口改为8090

<port>8090</port>

然后修改nginx的conf文件夹下的nginx.conf 配置文件

nginx.conf的结构：

http{

当前nginx的所有配置内容

}

我们需要做的就是在nginx.conf配置文件中配置一个server

server { //一个server表示nginx内存中启动加载的一个虚拟服务器，可以根据配置逻辑处理请求到nginx的http请求

listen 80;//监听的端口号

server\_name www.ou.com;//监听的域名，可以在request的host头中判断

location / {//location匹配的是当前url地址中过滤域名、端口后其他的字符串内容;”/”就是通配，所有的字符串都能匹配

例如www.ou.com/order/pay?orderId=a(a地址可以匹配，因为域名匹配，端口号匹配，后面的内容由于配置的是”/”，所以通配)

[www.ou.com(b地址可以)](http://www.ou.com(b地址可以))

www.ou.cn（不能匹配）

proxy\_pass http://127.0.0.1:8090;//进入location后，转发到后端服务器的地址，转发规则：location之前匹配到的全部去掉，之后的拼接到proxy\_pass后面。当前配置将会在location过滤之后拼接在域名和端口之后的东西

例如a地址：<http://127.0.0.1:8090/order/pay?orderId=a>

b地址：http://127.0.0.1:8090

}

}

### 4.3.2.nginx测试

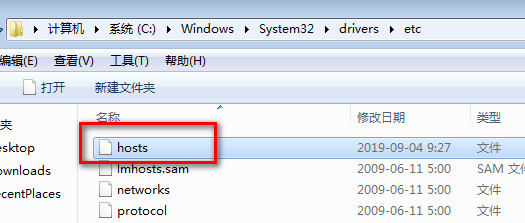
重启nginx，启动8090端口的tomcat

直接访问www.ou.com是无法访问的，因为浏览器会去找dns服务器找对应域名的ip

但是我们并未将域名与ip写到dns服务器中

hosts文件：

浏览器在访问dns之前，首先会到本地的hosts文件中找域名与ip的对应关系，所以，我们可以配置hosts文件完成域名与ip的对应



在hosts文件中添加一行（ip和域名之间用制表符tab分隔）：

127.0.0.1 www.ou.com

如果觉得打开hosts文件麻烦，可以使用一个软件（课前资料中提供的）



清浏览器缓存快捷键：ctrl+shift+delete

### 4.3.3.访问流程

1.浏览器输入www.ou.com/index.html

2.hosts映射，可以请求到nginx

3.nginx根据server匹配规则，匹配到域名和端口号，匹配成功

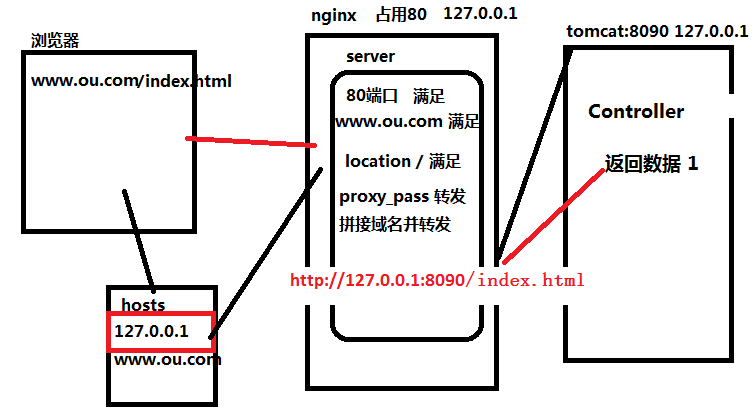
4.根据location匹配域名和端口后面的内容，这里配置的是’/’，所以通配

5.进入location的proxy\_pass，过滤掉location匹配之前的内容，拼接后面的内容，然后访问后台服务器

例如www.ou.com/index.html，去掉www.ou.com（相当于域名端口和location”/”）

拼接后为：http://127.0.0.1:8090/index.html

6.最终经过服务器处理，返回响应到nginx，然后响应到客户端



## 4.4.nginx入门案例-动静分离

动静分离：主要指前后端分离，前端的静态页面文件交给独立的服务器处理，后端的tomcat只处理请求的动态数据，没有页面访问的处理逻辑

nginx作为http服务器，不能处理动态请求，比如jsp，但是可以处理所有的静态文件访问，可以利用nginx的这个特点，将所有的静态资源存放在nginx中保存，提供访问的响应，如果有动态资源的请求，交给后端的tomcat服务器

### 4.4.1.nginx静态资源配置

在nginx.conf配置文件中配置一个server

server {

listen 80;

server\_name www.staticfile.com;

location / {

//proxy\_pass转发动态资源，root可以指向nginx服务器中任意的文件地址，比如D:/html文件夹

例如：www.staticfile.com/1.html,转发路径：D:/html/1.html

注意，这里可以写相对路径，例如root staticdir，如果写相对路径，会去nginx的根目录寻找此文件夹

一般这里用相对路径比较好

root D:/html;

index 2.html;//设置默认主页

}

}

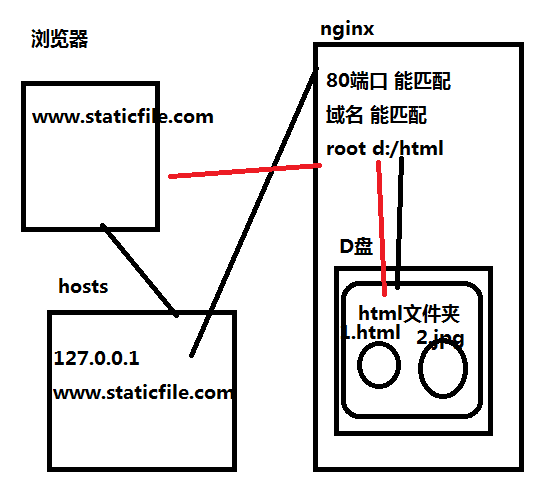
需要在d盘创建html文件夹，并添加一些静态文件（1.html）,修改hosts中的映射

127.0.0.1 [www.staticfile.com](http://www.staticfile.com)

重启nginx

在浏览器访问http://www.staticfile.com/1.html

### 4.4.2.静态资源转发流程



## 4.5.location匹配规则

根据location的匹配，具有优先级关系逻辑，以下案例优先级从高到低（上面的优先级高于下面的）

**精确匹配:**

location =/images;域名端口后的字符串必须完全等于location匹配的路径

**有修饰符的字符串前缀匹配：**

location ^~/images;所有字符串以/images开头的都能匹配

**无修饰符的字符串前缀匹配：**

location /images;可以匹配多级路径

**正则匹配：**

location ~正则表达式

**通配:**

location /

优先级相同时，匹配的字符串长度越多，则优先级越高

优先级相同，例如正则，谁在上面先匹配谁

例如：

location =/images{return 0}

location ^~/images{return 1}

location /images{return 2}

location ^~/images/test{return 3}

location ~.jpg${return 4}

location ~{.png|.jpg}${return 5}

location /{return 6}

请求路径:

localhost 6

localhost/images 0

localhost/images.jpg 1

localhost/images/test.png 3

location/user/a.jpg 4

## 4.6.nginx负载均衡

### 4.6.1.什么是负载均衡：

负载：服务器承受的访问压力

均衡：

物理均衡：平均分配，每台机器分配都一样

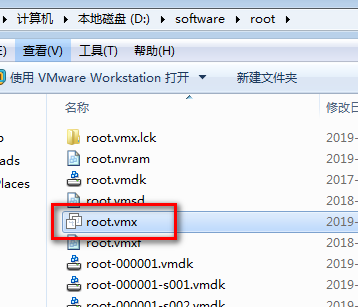
逻辑均衡：根据服务器性能分配访问的权重，性能越高，分配权重越大

### 4.6.2.在虚拟机上搭建三个tomcat集群

使用课前资料提供的虚拟机，因为里面已经安装了一些基础软件，例如jdk、node.js

将课前资料中的root.rar解压

登录时选other，用户名：root，密码：root



双击root.vmx即可在vmware中打开

确保虚拟机可以启动，并且xshell可以连接虚拟机

昨日回忆

利用ssm框架搭建了订单用户系统

单体系统

发现了一些问题：

并发集中在一个tomcat

程序员可能需要掌握整个系统的所有功能

静态资源没有分离

nginx：

http服务器反向代理技术

请求转发

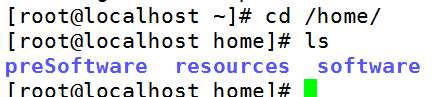
动静分离

负载均衡

## 4.7.负载均衡

用到了课前提供的虚拟机

### 4.7.1.在虚拟机/home目录中的目录结构



/home/preSoftware:里面是提前安装好的各种软件和环境资源

/home/resources：里面是互联网架构需要的安装包、配置文件

/home/software：我们将来安装的所有软件，都安装到此目录

### 4.7.2.在虚拟机安装tomcat

**将tomcat安装包拷贝到/home/software中**

[root@localhost software]# cp ../resources/apache-tomcat-9.0.6.tar.gz ./

**解压tar包**

[root@localhost software]# tar -xvf apache-tomcat-9.0.6.tar.gz

**删除tar包和改名**

[root@localhost software]# rm -rf apache-tomcat-9.0.6.tar.gz

[root@localhost software]# mv apache-tomcat-9.0.6/ t1

**删除t1中webapps中的所有内容**

[root@localhost webapps]# pwd

/home/software/t1/webapps

[root@localhost webapps]# rm -rf \*

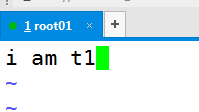
**在webapps中创建一个目录ROOT，在ROOT中创建一个index.html**

[root@localhost webapps]# mkdir ROOT

[root@localhost webapps]# cd ROOT

[root@localhost ROOT]# vim index.html

i进入编辑模式



保存退出：esc，然后:wq

**修改三个tomcat占用的端口号**

每个tomcat启动时，会占用三个端口

8080（69行左右）：请求的端口号

8005（22行左右）：关闭软件时执行的端口

8009（116行左右）：和其他服务器进行数据调用的端口

t1:8091,8015,8019

t2:8092,8025,8029

t3:8093,8035,8039

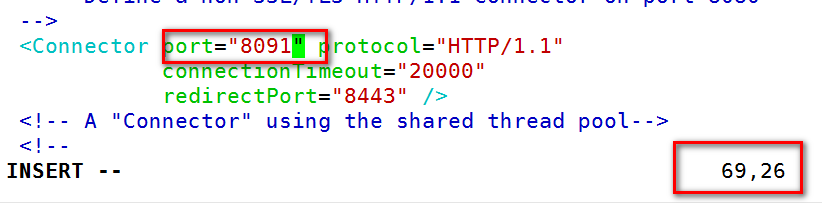
修改t1端口：

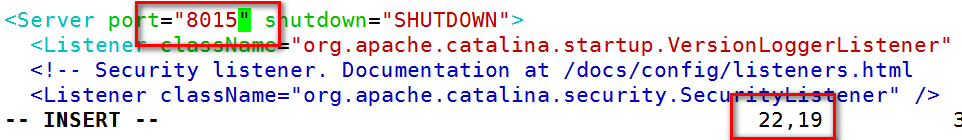
进入t1根目录下的conf目录，修改server.xml

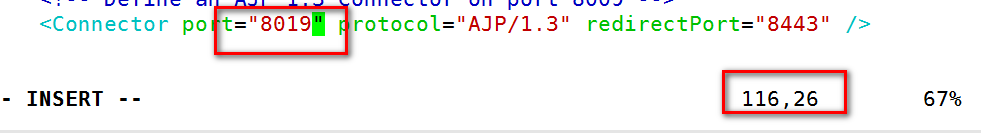
[root@localhost conf]# pwd

/home/software/t1/conf

[root@localhost conf]# vim server.xml







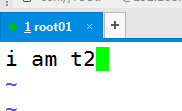
三个地方，改完后保存退出

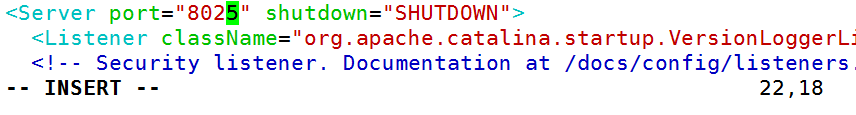
**复制t1为t2、t3**

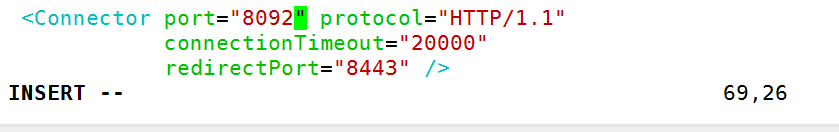
[root@localhost software]# cp -r t1 t2

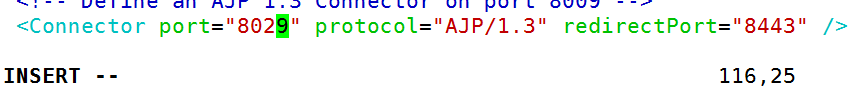
[root@localhost software]# cp -r t1 t3

**修改t2的index.html，3个端口号**









t3同上

**启动三个tomcat**

调用bin目录下的startup.sh文件（需要关闭防火墙）

关闭防火墙：

[root@localhost software]# chkconfig iptables off //永久关，重启后有效

[root@localhost software]# service iptables stop//临时关

启动tomcat

[root@localhost software]# t1/bin/startup.sh

[root@localhost software]# t2/bin/startup.sh

[root@localhost software]# t3/bin/startup.sh

关闭tomcat

[root@localhost software]# t1/bin/shutdown.sh

[root@localhost software]# t2/bin/shutdown.sh

[root@localhost software]# t3/bin/shutdown.sh

启动的所有日志都会在tomcat根目录下的logs目录下catalina.out文件中

### 4.7.3.轮询（roundBin）

nginx默认转发到后台服务器的逻辑-轮询：平均分配（物理均衡）

nginx中配置一个集群：通过关键字upstream配置（和server同级）

upstream tomcluster{//upstream是关键字，不能随便改，tomcluster是给这个集群起的名称，可以自定义（不要有特殊符号的字符串）,相当于虚拟的域名

server 192.168.171.138:8091;

server 192.168.171.138:8092;

server 192.168.171.138:8093;

}

要使用这个集群，只需在server中的proxy\_pass属性转发到此集群的名称即可

server {

listen 80;

server\_name www.ou.com;

location / {

#proxy\_pass http://127.0.0.1:8090;

proxy\_pass http://tomcluster;

}

}

测试：重启nginx，然后访问www.ou.com，发现每次刷新，转发时按照配置顺序进行访问，请求完全平均分配

### 4.7.4.权重

轮询访问，没有权重值，如果后台服务器性能差别较大，处理能力不同，可能导致性能差的服务器负载过重，所以，引入权重，可以配置性能好的服务器权重值大，反之亦然

权重可以在ip:端口后添加weight关键字，weight=n，n代表权重，性能好的，n大一些，反之亦然

down关键字：永不访问

upstream tomcluster{

server 192.168.171.138:8091 weight=3;

server 192.168.171.138:8092 weight=1;

server 192.168.171.138:8093 down;

}

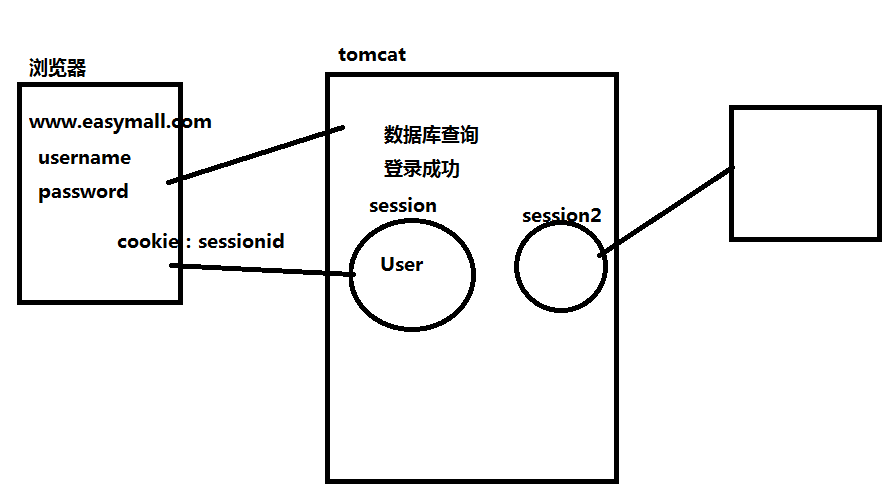
重启nginx测试

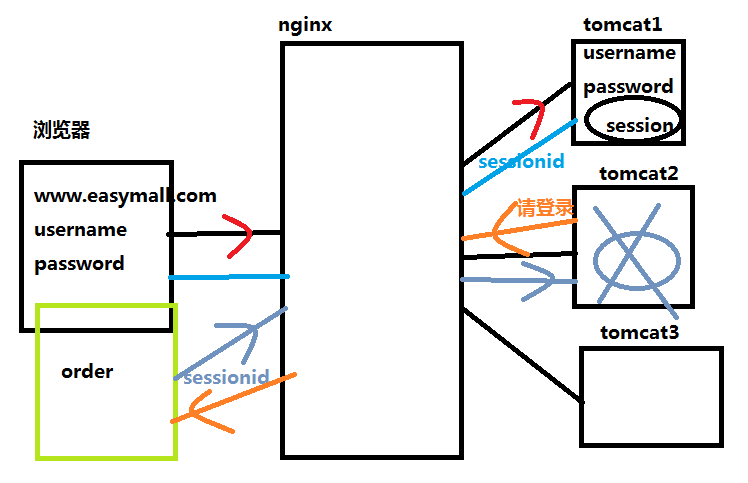
### 4.7.5.ip\_hash黏着负载均衡

session回忆：

session是服务器端会话技术，会话中产生的数据保存在服务器端

cookie是客户端会话技术，将数据保存在客户端





问题：由于nginx转发到任意一个tomcat，例如第一个tomcat登录并生成session，但是第二次访问时，请求到第二个tomcat，此tomcat中并无session，提示需要重新登录，以此类推

解决思路：同一个ip过来转发到同一个tomcat

ip\_hash

任何数据都可以做hash计算得到一个hash值，相同的数据hash值一定相同，不容的数据hash值也不同，可以对tomcat服务器编号，例如0,1,2，然后将hash值对3取余，得到0,1,2的取余结果，和服务器编号对应,这样，同一个ip多次访问，也会进入同一个服务器

配置ip\_hash

在轮询基础上添加一个关键字：ip\_hash

upstream tomcluster{

ip\_hash;

server 192.168.171.138:8091;

server 192.168.171.138:8092;

server 192.168.171.138:8093;

}

重启nginx测试

后期不会使用，redis解决session共享问题