

淘淘商城

第十二天

**讲师：传智.入云龙**

# 课程目标

1. 系统架构理解
2. 网络拓扑结构

3、掌握nginx+tomcat反向代理的使用方法。

4、掌握nginx作为负载均衡器的使用方法。

# 系统架构

## 系统架构图



## 网络拓扑图



# nginx介绍

## 什么是nginx

Nginx是一款高性能的http 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器。由俄罗斯的程序设计师Igor Sysoev所开发，官方测试nginx能够支支撑5万并发链接，并且cpu、内存等资源消耗却非常低，运行非常稳定。

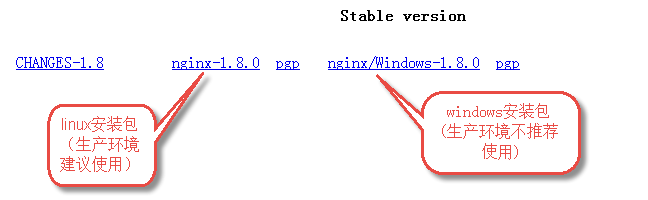
## Nginx的应用场景

1. http服务器。Nginx是一个http服务可以独立提供http服务。可以做网页静态服务器。
2. 虚拟主机。可以实现在一台服务器虚拟出多个网站。例如个人网站使用的虚拟主机。
3. 反向代理，负载均衡。当网站的访问量达到一定程度后，单台服务器不能满足用户的请求时，需要用多台服务器集群可以使用nginx做反向代理。并且多台服务器可以平均分担负载，不会因为某台服务器负载高宕机而某台服务器闲置的情况。

# nginx安装

## 下载

进入http://nginx.org/en/download.html 下载nginx1.8.0版本（当前最新稳定版本）。



## 安装

### nginx安装环境

nginx是C语言开发，建议在linux上运行，本教程使用Centos6.5作为安装环境。

* gcc

安装nginx需要先将官网下载的源码进行编译，编译依赖gcc环境，如果没有gcc环境，需要安装gcc：yum install gcc-c++

* PCRE

PCRE(Perl Compatible Regular Expressions)是一个Perl库，包括 perl 兼容的正则表达式库。nginx的http模块使用pcre来解析正则表达式，所以需要在linux上安装pcre库。

**yum install -y pcre pcre-devel**

注：pcre-devel是使用pcre开发的一个二次开发库。nginx也需要此库。

* zlib

zlib库提供了很多种压缩和解压缩的方式，nginx使用zlib对http包的内容进行gzip，所以需要在linux上安装zlib库。

**yum install -y zlib zlib-devel**

* openssl

OpenSSL 是一个强大的安全套接字层密码库，囊括主要的密码算法、常用的密钥和证书封装管理功能及SSL协议，并提供丰富的应用程序供测试或其它目的使用。

nginx不仅支持http协议，还支持https（即在ssl协议上传输http），所以需要在linux安装openssl库。

**yum install -y openssl openssl-devel**

### 编译安装

将nginx-1.8.0.tar.gz拷贝至linux服务器。

解压：

tar -zxvf nginx-1.8.0.tar.gz

cd nginx-1.8.0

1. configure

./configure --help查询详细参数（参考本教程附录部分：nginx编译参数）

参数设置如下：

./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

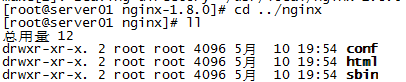
**注意：上边将临时文件目录指定为/var/temp/nginx，需要在/var下创建temp及nginx目录**

1. 编译安装

make

make install

安装成功查看安装目录 ：



## 启动nginx

cd /usr/local/nginx/sbin/

./nginx

查询nginx进程：



15098是nginx主进程的进程id，15099是nginx工作进程的进程id

**注意：执行./nginx启动nginx，这里可以-c指定加载的nginx配置文件，如下：**

**./nginx -c /usr/local/nginx/conf/nginx.conf**

**如果不指定-c，nginx在启动时默认加载conf/nginx.conf文件，此文件的地址也可以在编译安装nginx时指定./configure的参数（--conf-path= 指向配置文件（nginx.conf））**

## 停止nginx

方式1，快速停止：

cd /usr/local/nginx/sbin

./nginx -s stop

此方式相当于先查出nginx进程id再使用kill命令强制杀掉进程。

方式2，完整停止(建议使用)：

cd /usr/local/nginx/sbin

./nginx -s quit

此方式停止步骤是待nginx进程处理任务完毕进行停止。

## 重启nginx

方式1，先停止再启动（建议使用）：

对nginx进行重启相当于先停止nginx再启动nginx，即先执行停止命令再执行启动命令。

如下：

./nginx -s quit

./nginx

方式2，重新加载配置文件：

当nginx的配置文件nginx.conf修改后，要想让配置生效需要重启nginx，使用-s reload不用先停止nginx再启动nginx即可将配置信息在nginx中生效，如下：

./nginx -s reload

## 安装测试

nginx安装成功，启动nginx，即可访问虚拟机上的nginx：



到这说明nginx基本上安装成功。

# 配置虚拟主机

## 什么是虚拟主机

虚拟主机是一种特殊的软硬件技术，它可以将网络上的每一台计算机分成多个虚拟主机，每个虚拟主机可以独立对外提供www服务，这样就可以实现一台主机对外提供多个web服务，每个虚拟主机之间是独立的，互不影响的。

如下图：

一台物理服务器

www.itcast.cn

open.itcast.cn

......

通过nginx可以实现虚拟主机的配置，nginx支持三种类型的虚拟主机配置，1、基于ip的虚拟主机， 2、基于域名的虚拟主机 3、基于端口的虚拟主机

## nginx配置文件的结构

nginx的配置文件结构如下：

......

events {

.......

}

http{

.......

server{

.......

}

server{

.......

}

}

每个server就是一个虚拟主机。

## 基于ip的虚拟主机配置

Linux操作系统允许添加IP别名，IP别名就是在一块物理网卡上绑定多个lP地址。这样就能够在使用单一网卡的同一个服务器上运行多个基于IP的虚拟主机。

### 需求

一台nginx服务器绑定两个ip：192.168.101.3、192.168.101.103

访问不同的ip请求不同的html目录，即：

访问http://192.168.101.3将访问“html3”目录下的html网页

访问http://192.168.101.103将访问“html103”目录下的html网页

### 准备环境

创建192.168.101.3虚拟机，保证本地电脑和虚拟网络通畅。

在192.168.101.3上安装nginx。

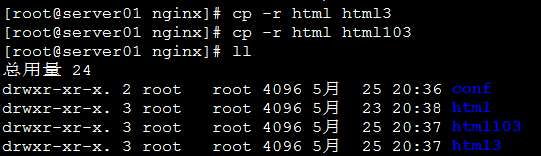
### html目录创建

将原来nginx的html目录拷贝两个目录 “html3”和“html103”，为了方便测试需要修改每个目录下的index.html内容使之个性化。

cd /usr/local/nginx

cp -r html html3

cp -r html html103

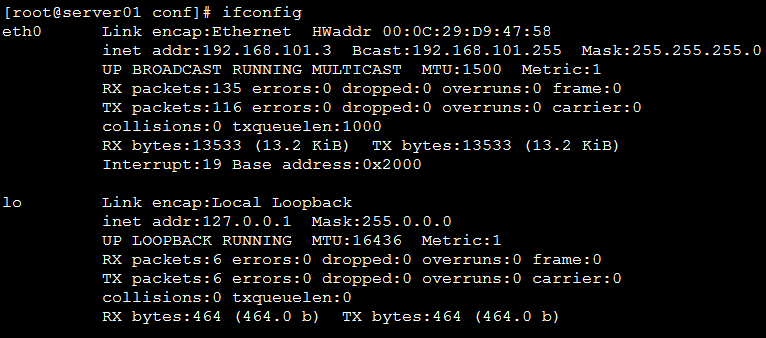


### 绑定多ip

**方法一：**

使用标准的网络配置工具（比如ifconfig和route命令）添加lP别名：

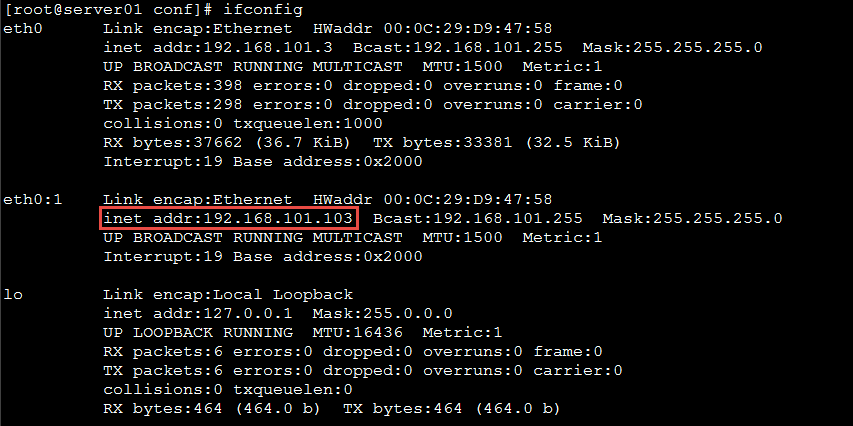
当前ip配置情况：



在eth0网卡再绑定一个ip：192.168.101.103

/sbin/ifconfig eth0:1 192.168.101.103 broadcast 192.168.101.255 netmask 255.255.255.0 up

/sbin/route add -host 192.168.101.103 dev eth0:1



**方法二：**

1、将/etc/sysconfig/network-scripts/ifcfg-eth0文件复制一份，命名为ifcfg-eth0:1

修改其中内容：

DEVICE=eth0:1

IPADDR=192.168.25.103

其他项不用修改

2、重启系统

### 配置虚拟主机

修改/usr/local/nginx/conf/nginx.conf文件，添加两个虚拟主机，如下：

#user nobody;

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

#配置虚拟主机192.168.101.3

server {

#监听的ip和端口，配置192.168.101.3:80

listen 80;

#虚拟主机名称这里配置ip地址

server\_name 192.168.101.3;

#所有的请求都以/开始，所有的请求都可以匹配此location

location / {

#使用root指令指定虚拟主机目录即网页存放目录

#比如访问http://ip/test.html将找到/usr/local/html3/test.html

#比如访问http://ip/item/test.html将找到/usr/local/html3/item/test.html

root /usr/local/nginx/html3;

#指定欢迎页面，按从左到右顺序查找

index index.html index.htm;

}

}

#配置虚拟主机192.168.101.103

server {

listen 80;

server\_name 192.168.101.103;

location / {

root /usr/local/nginx/html103;

index index.html index.htm;

}

}

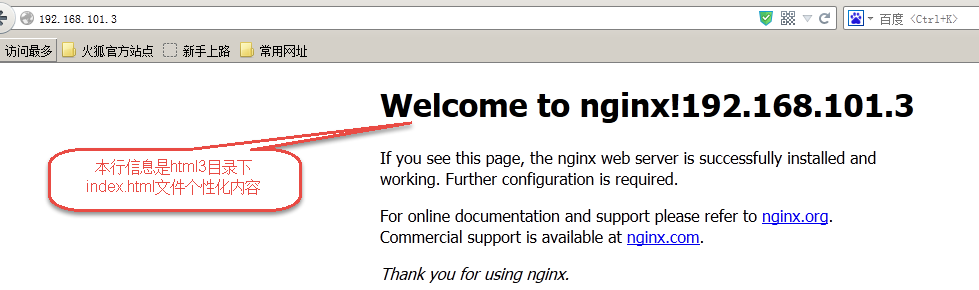
}

### 测试

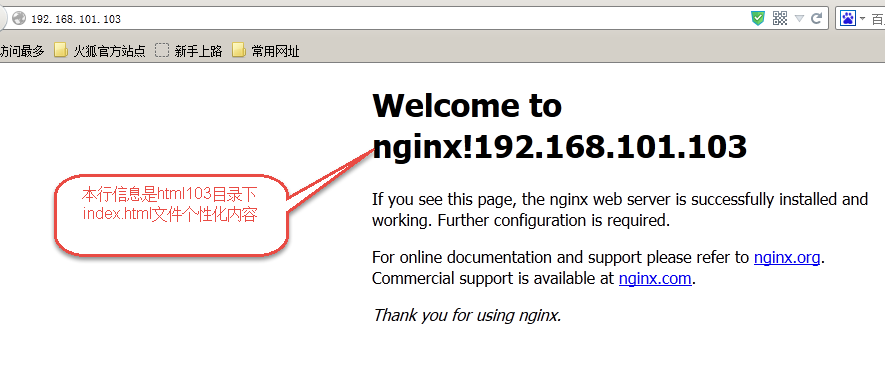
启动nginx，观察端口监听状态：



访问http://192.168.101.3



访问http://192.168.101.103



## 基于端口的虚拟主机

### 需求

nginx对外提供80和8080两个端口监听服务。

请求80端口则请求html80目录下的html

请求8080端口则请求html8080目录下的html

### 准备环境

创建192.168.101.3虚拟机，保证本地电脑和虚拟网络通畅。

在192.168.101.3上安装nginx。

### html目录创建

将原来nginx的html目录拷贝两个目录 “html80”和“html8080”，为了方便测试需要修改每个目录下的index.html内容使之个性化。

参考基于ip或域名的虚拟主机章节。

### 配置虚拟主机

修改/usr/local/nginx/conf/nginx.conf文件，添加两个虚拟主机，如下：

#user nobody;

worker\_processes 1;

events {

worker\_connections 1024;

}

http {

include mime.types;

default\_type application/octet-stream;

sendfile on;

keepalive\_timeout 65;

#配置虚拟主机

server {

#监听的ip和端口，配置80

listen 80;

#虚拟主机名称这里配置ip地址

server\_name 192.168.101.3;

#所有的请求都以/开始，所有的请求都可以匹配此location

location / {

#使用root指令指定虚拟主机目录即网页存放目录

#比如访问http://ip/test.html将找到/usr/local/html3/test.html

#比如访问http://ip/item/test.html将找到/usr/local/html3/item/test.html

root /usr/local/nginx/html80;

#指定欢迎页面，按从左到右顺序查找

index index.html index.htm;

}

}

#配置虚拟主机

server {

listen 8080;

server\_name 192.168.101.3;

location / {

root /usr/local/nginx/html8080;

index index.html index.htm;

}

}

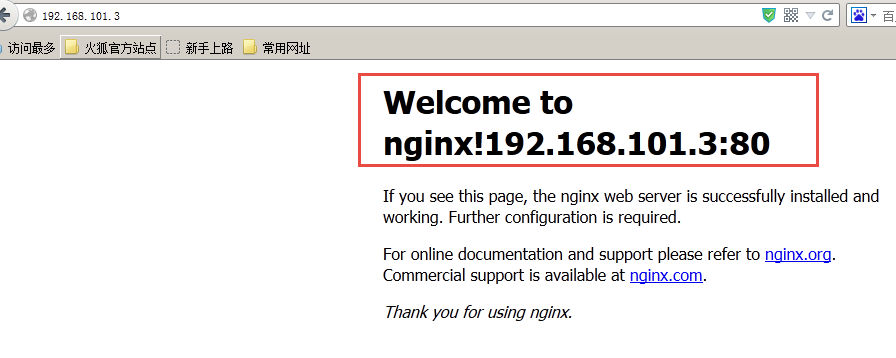
}

### 测试

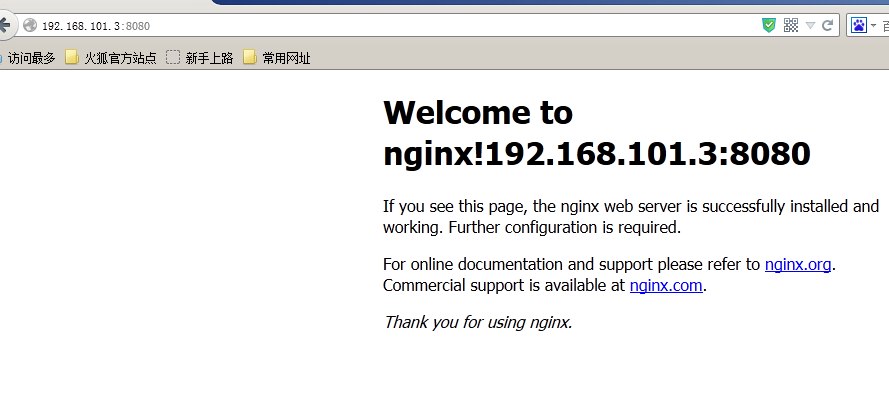
启动 nginx，查看端口监听状态：



访问http://192.168.101.3



访问http://192.168.101.3:8080



## 基于域名的虚拟主机配置

### 需求

两个域名指向同一台nginx服务器，用户访问不同的域名显示不同的网页内容。

两个域名是aaa.test.com和bbb.test.com

nginx服务器使用虚拟机192.168.101.3

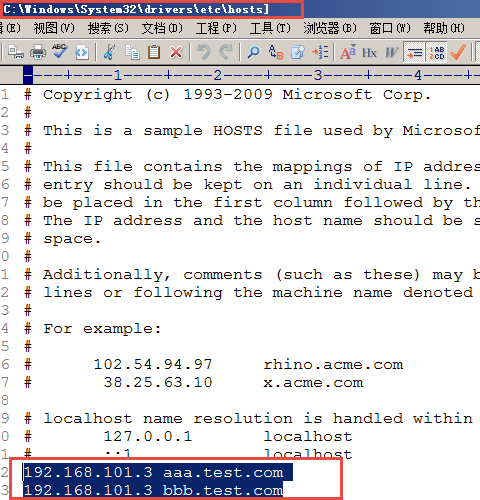
### 准备环境

创建192.168.101.3虚拟机，保证本地电脑和虚拟网络通畅。

在192.168.101.3上安装nginx。

通过host文件指定aaa.test.com和bbb.test.com对应192.168.101.3虚拟机：

修改window的hosts文件：（C:\Windows\System32\drivers\etc）



### html目录创建

在192.168.101.3上创建/usr/local/aaa\_html，此目录为aaa.test.com域名访问的目录

在192.168.101.3上创建/usr/local/bbb\_html，此目录为bbb.test.com域名访问的目录

目录中的内容使用nginx自带的html文件，将/usr/local/nginx/html中的内容拷贝分别拷贝到上边两个目录中，并且将aaa\_html目录中的index.html内容改为：“Welcome to aaa nginx!”

将bbb\_html目录中的index.html内容改为“Welcome to bbb nginx!”

### 配置虚拟主机

修改/usr/local/nginx/conf/nginx.conf文件，添加两个虚拟主机，如下：

#配置虚拟主机aaa.test.com

server {

#监听的ip和端口，配置本机ip和端口

listen 192.168.101.3:80;

#虚拟主机名称是aaa.test.com，请求域名aaa.test.com的url将由此server配置解析

server\_name aaa.test.com;

#所有的请求都以/开始，所有的请求都可以匹配此location

location / {

#使用root指令指定虚拟主机目录即网页存放目录

#比如访问http://ip/test.html将找到/usr/local/aaa\_html/test.html

#比如访问http://ip/item/test.html将找到/usr/local/aaa\_html/item/test.html

root /usr/local/aaa\_html;

#指定欢迎页面，按从左到右顺序查找

index index.html index.htm;

}

}

#配置虚拟主机bbb.test.com

server {

listen 192.168.101.3:80;

server\_name bbb.test.com;

location / {

root /usr/local/bbb\_html;

index index.html index.htm;

}

}

### 测试

访问aaa.test.com、bbb.test.com

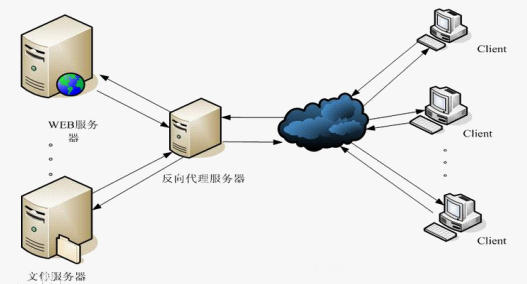
# nginx反向代理

## 什么是反向代理

通常的代理服务器，只用于代理内部网络对Internet的连接请求，客户机必须指定代理服务器,并将本来要直接发送到Web服务器上的http请求发送到代理服务器中由代理服务器向Internet上的web服务器发起请求，最终达到客户机上网的目的。

而反向代理（Reverse Proxy）方式是指以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个反向代理服务器。

如下图：



## nginx+tomcat反向代理

### 需求

两个tomcat服务通过nginx反向代理，本例子使用三台虚拟机进行测试，

nginx服务器：192.168.101.3

tomcat1服务器：192.168.101.5

tomcat2服务器：192.168.101.6

如下图：

nginx 192.168.101.3

tomcat1 192.168.101.5

tomcat2 192.168.101.6

请求aaa.test.com

请求bbb.test.com

aaa.test.com由tomcat1提供服务

bbb.test.com由tomcat2提供服务

### 启动tomcat

tomcat使用apache-tomcat-7.0.57版本，在192.168.101.5和192.168.101.6虚拟机上启动tomcat。

### nginx反向代理配置

根据上边的需求在nginx.conf文件中配置反向代理，如下：

#配置一个代理即tomcat1服务器

upstream tomcat\_server1 {

server 192.168.101.5:8080;

}

#配置一个代理即tomcat2服务器

upstream tomcat\_server2 {

server 192.168.101.6:8080;

}

#配置一个虚拟主机

server {

listen 80;

server\_name aaa.test.com;

location / {

#域名aaa.test.com的请求全部转发到tomcat\_server1即tomcat1服务上

proxy\_pass http://tomcat\_server1;

#欢迎页面，按照从左到右的顺序查找页面

index index.jsp index.html index.htm;

}

}

server {

listen 80;

server\_name bbb.test.com;

location / {

#域名bbb.test.com的请求全部转发到tomcat\_server2即tomcat2服务上

proxy\_pass http://tomcat\_server2;

index index.jsp index.html index.htm;

}

}

### 测试

分别修改两个tomcat下的webapps/ROOT/index.jsp的内容，使用tomcat1和tomcat2两个服务首页显示不同的内容，如下：

tomcat1下的index.jsp修改后：



tomcat2下的index.jsp修改后：



分别访问aaa.test.com、bbb.test.com测试反向代理。

请求访问aaa.test.com通过nginx代理访问tomcat1，请求访问bbb.test.com通过nginx代理访问tomcat2。

# 负载均衡

## 什么是负载均衡

负载均衡 建立在现有网络结构之上，它提供了一种廉价有效透明的方法扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性。

负载均衡，英文名称为Load Balance，其意思就是分摊到多个操作单元上进行执行，例如Web服务器、FTP服务器、企业关键应用服务器和其它关键任务服务器等，从而共同完成工作任务。

## nginx实现负载均衡

### 需求

nginx作为负载均衡服务器，用户请求先到达nginx，再由nginx根据负载配置将请求转发至 tomcat服务器。

nginx负载均衡服务器：192.168.101.3

tomcat1服务器：192.168.101.5

tomcat2服务器：192.168.101.6

nginx 192.168.101.3

负载均衡服务器

tomcat1 192.168.101.5

tomcat2 192.168.101.6

请求aaa.test.com

请求aaa.test.com

请求经过负载均衡至tomcat1

请求经过负载均衡至tomcat2

### 配置

根据上边的需求在nginx.conf文件中配置负载均衡，如下：

upstream tomcat\_server\_pool{

server 192.168.101.5:8080 weight=10;

server 192.168.101.6:8080 weight=10;

}

server {

listen 80;

server\_name aaa.test.com;

location / {

proxy\_pass http://tomcat\_server\_pool;

index index.jsp index.html index.htm;

}

}

|  |
| --- |
| **节点说明：**  在http节点里添加:  #定义负载均衡设备的 Ip及设备状态  upstream myServer {  server 127.0.0.1:9090 down;  server 127.0.0.1:8080 weight=2;  server 127.0.0.1:6060;  server 127.0.0.1:7070 backup;  }  在需要使用负载的Server节点下添加  proxy\_pass http://myServer;  upstream 每个设备的状态:  down 表示单前的server暂时不参与负载  weight 默认为1.weight越大，负载的权重就越大。  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:max\_fails 次失败后，暂停的时间。  backup： 其它所有的非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。 |

### 测试

请求aaa.test.com，通过nginx负载均衡，将请求转发到tomcat服务器。

通过观察tomcat的访问日志或tomcat访问页面即可知道当前请求由哪个tomcat服务器受理。