Tomcat服务部署

一、Tomcat服务部署--前期了解

1.中间件产品

2.Tomcat

3.Tomcat应用场景

4.JDK软件简介

二、安装配置Tomcat

1.所有主机关闭防火墙和selinux

2.查看JDK是否安装

3.卸载rpm方式安装的jdk

4.JDK安装

5.解压apache-tomcat-8.5.16.tar.gz包

6.Tomcat配置目录及文件说明

1.ps : 面试题，描述 netstat -lnpt每一列代表的含义

2.tomcat主目录介绍

3.webapps 目录介绍

4.conf 目录介绍

ps:判断某一台主机的一个端口是否开启?

??????????

实验：自定义默认网站目录

Nginx+Tomcat负载均衡集群(群集 Cluster)

一、nginx服务器：网站服务、代理

Tomcat服务部署

一、Tomcat服务部署--前期了解

1.中间件产品

- **Tomcat　---中小型公司，开源，免费**
- **中间件产品: RedHat JBoss、Oracle Tuxedo、caucho Resin（开源，免费；也有收费版本）**

- **WebSphere、 Weblogic 、 RedHat JBoss、 Oracle Tuxedo**

---商业版，Weblogic ----大公司（银行、证券），购买服务，厂家支持平台更稳定

2.Tomcat

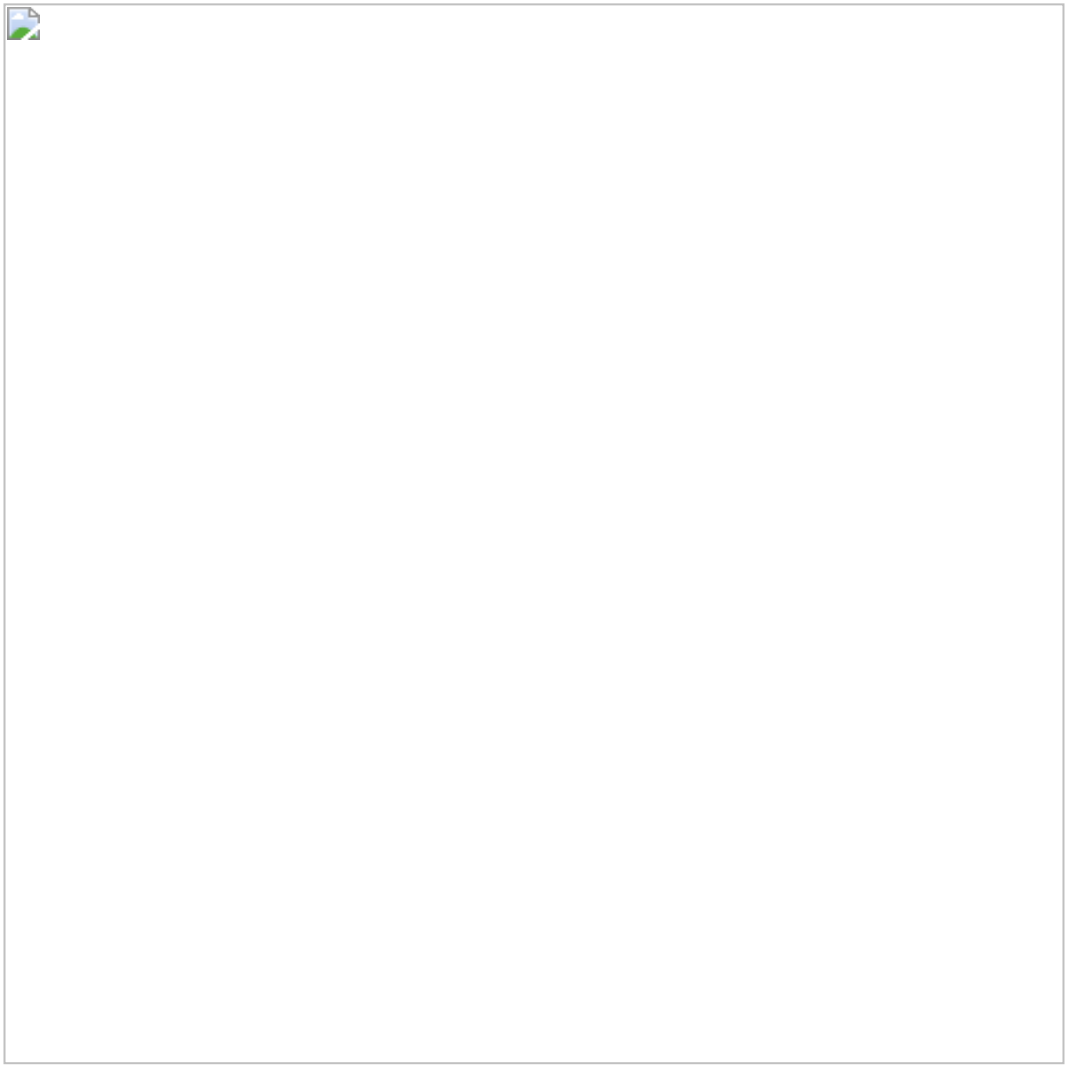
- Apache 软件基金会（Apache Software Foundation）
- 产品：httpd
- Apache HTTP Server
- O'Reilly出版的介绍Tomcat的书籍
- Tomcat的Logo兼吉祥物被设计为一只公猫
- 早期的Tomcat的Logo项目的名字叫Catalina.
- 所以当安装完Tomcat,安装路径下有很多Catalina有关的目录和文件。
- 这些文件是我们配置或使用Tomcat的重要文件

3.Tomcat应用场景

- LAMP LNMP 提供了PHP页面解析
- **Tomcat 提供了jsp页面（Java）**
- 动态网站开发语言：asp php jsp node.js -- 动态网站开发技术

ps：不同的软件环境提供了不同的页面解析

- Tomcat服务器：免费的开放源代码的web应用服务器（轻量级应用服务器）---实现不同的业务功能
- 应用：中小型系统和并发访问用户不是很多的场合，是开发和调试JSP页面的首选
- Tomcat可处理静态的HTML页面；能力不及Apache或Nginx
- Tomcat通常作为一个servlet和JSP**容器**，单独运行在后端
- Tomcat官网: http://tomcat.apache.org



4.JDK软件简介

- 在安装源码包之前，先安装Java解释器，gcc gcc-c++
- 在安装Tomcat之前必须先安装JDK
- JDK 全称是Java Development Kit
- java语言的软件开发工具包
- 编写java源程序,编译产生java字节码
- Java虚拟机(JVM)----解释字节码文件,保证Java 的跨平台性
- CentOS7 系统中默认已经安装了JDK

JDK包含用于Java开发的组件，包括：

javac

Java compiler 是位于JDK安装目录/bin的javac工具。

编译器，将后缀名为.java的源代码编译成后缀名为".class"的字节码

java

运行工具，运行.class的字节码。

jar

基于ZIP和ZLIB压缩格式的存档和压缩工具，把相关文件打包成一个文件

jps

查看jave程序运行进程的相关信息

二、安装配置Tomcat

- 安装时候选择tomcat软件版本要与程序开发使用的版本一致
- jdk版本要进行与tomcat保持一致

1.所有主机关闭防火墙和selinux

```
[root@localhost ~]# systemctl stop firewalld
[root@localhost ~]# iptables -F
[root@localhost ~]# setenforce 0
setenforce: SELinux is disabled
```

2.查看JDK是否安装

```
[root@localhost ~]# java -version      #系统自带的rpm安装的软件包
openjdk version "1.8.0_161"
OpenJDK Runtime Environment (build 1.8.0_161-b14)
OpenJDK 64-Bit Server VM (build 25.161-b14, mixed mode)
```

安装功能更加全面的软件包

3.卸载rpm方式安装的jdk

方法一：将生成的主程序删除

```
[root@localhost ~]# which java
/usr/bin/java
[root@localhost ~]# rm -rf /usr/bin/java
[root@localhost ~]# java -version
bash: java: 未找到命令...
```

方法二：完全卸载

```
[root@localhost ~]# rpm -qa | grep -i openjdk
java-1.8.0-openjdk-1.8.0.161-2.b14.el7.x86_64
java-1.7.0-openjdk-1.7.0.171-2.6.13.2.el7.x86_64
java-1.8.0-openjdk-headless-1.8.0.161-2.b14.el7.x86_64
java-1.7.0-openjdk-headless-1.7.0.171-2.6.13.2.el7.x86_64
[root@localhost ~]# rpm -e java-1.8.0-openjdk --nodeps   #为什么这里不用卸载依赖关系
[root@localhost ~]# rpm -e java-1.8.0-openjdk-headless
[root@localhost ~]# rpm -qa | grep -i openjdk
```

4.JDK安装

```
[root@localhost ~]# tar xf jdk-8u191-linux-x64.tar.gz
[root@localhost ~]# mv jdk1.8.0_191/ /usr/local/java
[root@localhost ~]# vim /etc/profile
  # 末尾加入两行
  # 这里修改PATH变量的值，而PATH变量是命令搜索路径，一定要核对好
export JAVA_HOME=/usr/local/java
export PATH=$PATH:$JAVA_HOME/bin                    ? ? ? ?
[root@localhost ~]# source /etc/profile
[root@localhost ~]# java -version
java version "1.8.0_191"
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)
```

5.解压apache-tomcat-8.5.16.tar.gz包

```
[root@localhost ~]# tar xf apache-tomcat-8.5.40.tar.gz
[root@localhost ~]# mv apache-tomcat-8.5.40 /usr/local/tomcat8
    #这里的Tomcat8的名字随意，8代表版本
[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh
Using CATALINA_BASE:   /usr/local/tomcat8
Using CATALINA_HOME:   /usr/local/tomcat8
Using CATALINA_TMPDIR: /usr/local/tomcat8/temp
Using JRE_HOME:        /usr/local/java
Using CLASSPATH:
/usr/local/tomcat8/bin/bootstrap.jar:/usr/local/tomcat8/bin/tomcat-juli.jar
Tomcat started.
[root@localhost ~]# netstat -lnpt | grep :8080
tcp6       0      0 :::8080                 :::*                    LISTEN      4174/java
```

关闭服务：

```
[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh
```

6.Tomcat配置目录及文件说明

1.ps：面试题，描述 netstat -lnpt每一列代表的含义

```
[root@localhost ~]# netstat -lnpt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      708/rpcbind
tcp        0      0 192.168.122.1:53        0.0.0.0:*               LISTEN      1472/dnsmasq
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1069/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      1064/cupsd
```

```
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      1178/master
tcp        0      0 127.0.0.1:6010          0.0.0.0:*               LISTEN      1543/sshd: root@pts
tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN      4174/java
tcp6       0      0 :::8009                 :::*                    LISTEN      4174/java
tcp6       0      0 :::111                  :::*                    LISTEN      708/rpcbind
tcp6       0      0 :::8080                 :::*                    LISTEN      4174/java
tcp6       0      0 :::22                   :::*                    LISTEN      1069/sshd
tcp6       0      0 ::1:631                 :::*                    LISTEN      1064/cupsd
tcp6       0      0 ::1:25                  :::*                    LISTEN      1178/master
tcp6       0      0 ::1:6010                :::*                    LISTEN      1543/sshd: root@pts
 协议
 队列 （queue）： 暂时非零可接受，长期非零有问题
 监听地址及端口
 目标地址及端口
 状态
 PID号及进程名
[root@localhost ~]# ls /usr/local/tomcat8
bin         conf        lib     logs    README.md      RUNNING.txt  webapps
BUILDING.txt  CONTRIBUTING.md  LICENSE  NOTICE  RELEASE-NOTES  temp
work
[root@localhost ~]# ls -l /usr/local/tomcat8/
总用量 124
drwxr-x--- 2 root root  4096 4月   6 12:25 bin
-rw-r----- 1 root root 19539 4月  10 2019 BUILDING.txt
drwx------ 3 root root   254 4月   6 12:26 conf
-rw-r----- 1 root root  6090 4月  10 2019 CONTRIBUTING.md
drwxr-x--- 2 root root  4096 4月   6 12:25 lib
-rw-r----- 1 root root 57092 4月  10 2019 LICENSE
drwxr-x--- 2 root root   197 4月   6 12:26 logs
-rw-r----- 1 root root  1726 4月  10 2019 NOTICE
-rw-r----- 1 root root  3255 4月  10 2019 README.md
-rw-r----- 1 root root  7139 4月  10 2019 RELEASE-NOTES
-rw-r----- 1 root root 16262 4月  10 2019 RUNNING.txt
drwxr-x--- 2 root root    30 4月   6 12:25 temp
drwxr-x--- 7 root root    81 4月  10 2019 webapps
drwxr-x--- 3 root root    22 4月   6 12:26 work

[root@localhost ~]# ls /usr/local/tomcat8/bin    # 程序文件所在目录
bootstrap.jar       commons-daemon-native.tar.gz  digest.sh         startup.bat
tool-wrapper.sh
catalina.bat        configtest.bat                setclasspath.bat  startup.sh
version.bat
catalina.sh         configtest.sh                 setclasspath.sh   tomcat-juli.jar
version.sh
catalina-tasks.xml  daemon.sh                     shutdown.bat      tomcat-native.tar.gz
commons-daemon.jar  digest.bat                    shutdown.sh       tool-wrapper.bat
```

.sh　　linux里面用的脚本
.bat　　windows里面用的批处理文件

2.tomcat主目录介绍

bin　//存放windows或linux平台上启动或关闭的Tomcat的脚本文件。

conf　/存放Tomcat的各种全局配置文件，其中最主要的是server.xml和web.xml

lib　　//存放Tomcat运行需要的库文件(UARS),功能包--jar包

　　linux 库文件 .so　　windows 库文件 .dll

logs　//存放Tomcat执行时的日志文件

webapps　　//Tomcat的主要Web发布目录(存放网页)、类似于nginx 的html (包括应用实例)。

3.webapps 目录介绍

```
[root@localhost ~]# ls -l /usr/local/tomcat8/webapps/
总用量 4
drwxr-x--- 14 root root 4096 4月   6 12:25 docs          # Tomcat的帮助文档
drwxr-x---  6 root root   83 4月   6 12:25 examples      # web应用实例
drwxr-x---  5 root root   87 4月   6 12:25 host-manager   #主机管理
drwxr-x---  5 root root  103 4月   6 12:25 manager       # 管理类配置
drwxr-x---  3 root root  306 4月   6 12:25 ROOT          # 真正的网页存放地址（默认站点根目录）
```

4.conf 目录介绍

```
[root@localhost ~]# ls -l /usr/local/tomcat8/conf
-rw------- 1 root root  1338 4月  10 2019 context.xml      # 帮助Tomcat指定额外的网站目录
-rw------- 1 root root  7511 4月  10 2019 server.xml       # 主配置文件
-rw------- 1 root root  2164 4月  10 2019 tomcat-users.xml      # 配置Tomcat的管理用户信息
```

1.Tomcat主配置文件说明

server.xml 主要配置文件

- 可修改启动端口
- 设置修改更换网站根目录
- 虚拟主机
- 多实例
- 开启https加密等功能

2.server.xml的结构的构成:

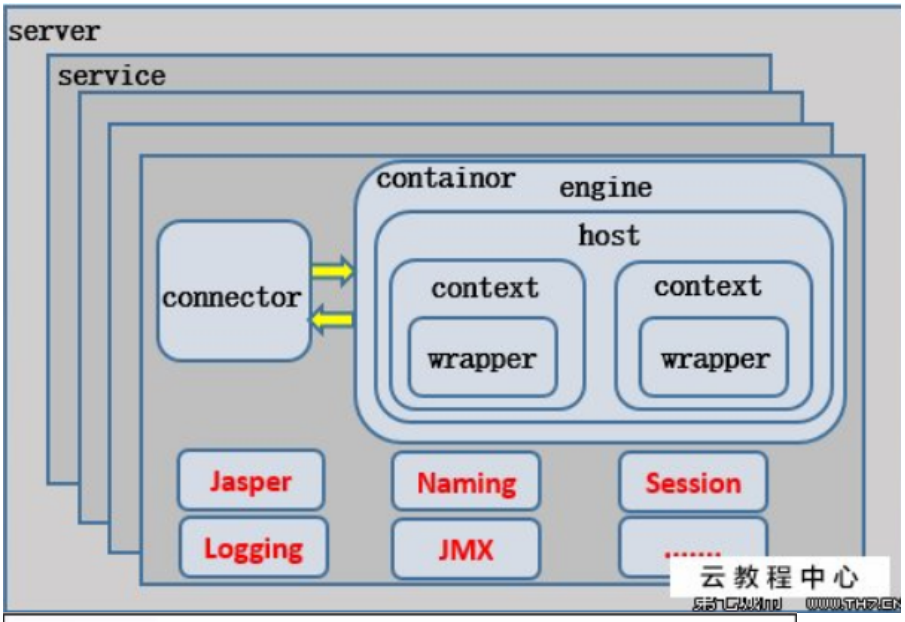
<Server>

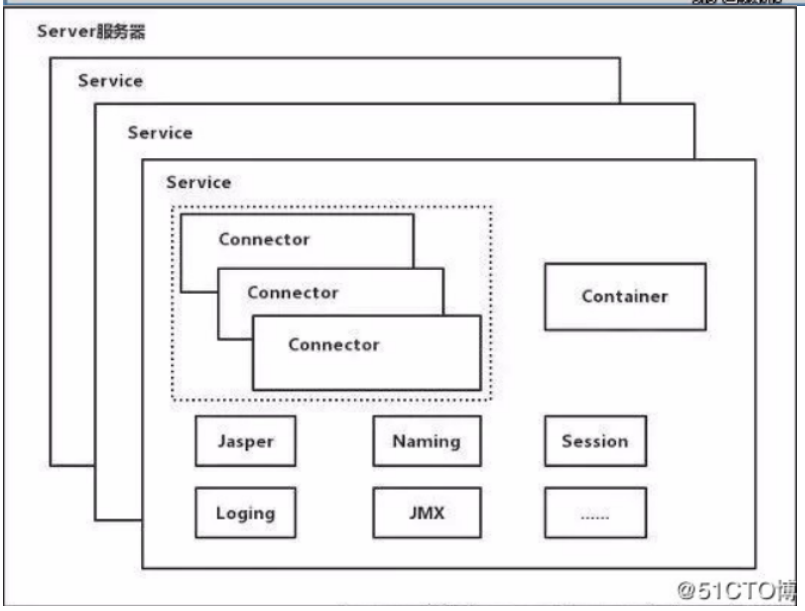
```
    <Service>
        <Connector />
        <Engine>
            <Host>      # 虚拟主机
                <Context></Context>    # 指定网页存放位置
            </Host>
        </Engine>
    </Service>
</Server>
```

<!-- -->内的内容是注释信息

- 修改时配置文件用vim，有颜色作区分

server.xml 主要参数说明





- connector:接收用户请求，且有监听端口
- engine：将connector建立好的连接接收过来进行处理
- 交给匹配的虚拟主机（对外提供服务），有context指定网页存放位置

host参数详解

```
<Host name="localhost"  appBase="webapps"          # 域名及网站存放目录
            unpackWARs="true" autoDeploy="true">
            <Context docBase="/web/webapp" path="" reloadable="flase" >
            </Context>
```

unpackWARs="true"

autoDeploy="true">

网站代码=.war压缩包

自动解压，自动部署

docBase

指定网页存放目录

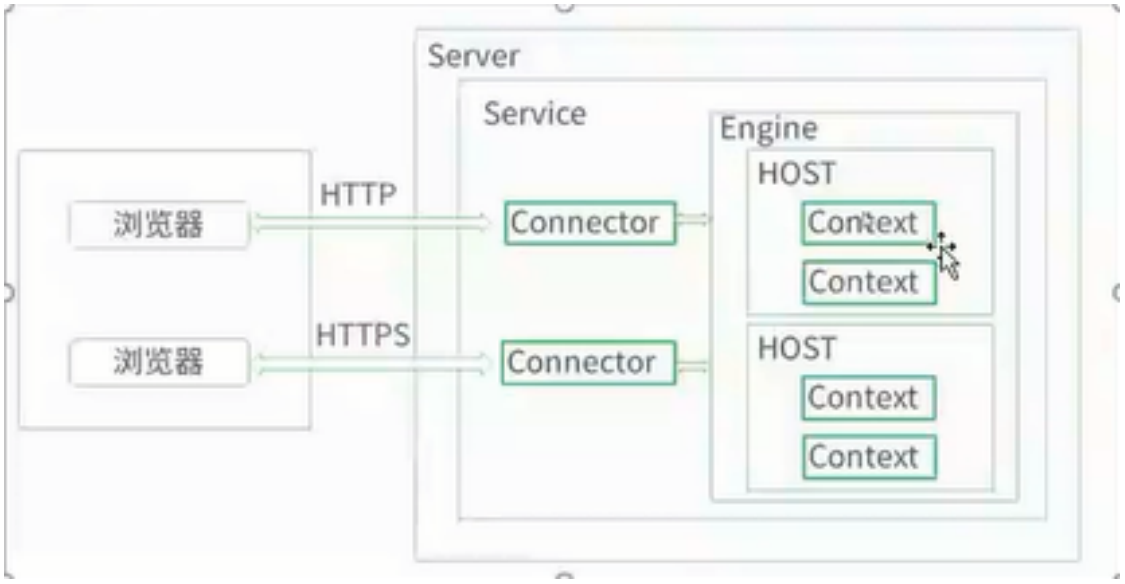
path=""

URL前缀

reloadable="flase"

如果为true，则tomcat会自动检测应用程序的/WEB-INF/lib和/WEB-INF/classes 目录的变化，

自动装载新的应用程序，可以在不重启tomcat的情况下改变应用程序



处理流程:

用户发送请求到WEB服务器，该请求会被正在监听的Connector连接器接收，

并把该请求交给Service下的Engine 来处理，并等待Engine处理的结果。

Engine 获得请求后会根据请求的主机信息来匹配相应的Host主机，
Host 主机会根据请求的路径匹配对应的Context，
Context web应用匹配上之后就构建request. response 请求对象，
调用指定的Servlet来处理请求。
请求处理完成后会将response 对象返回给Host主机，
Host主机将response对象返回给Engine引擎，
Engine 再将response 对象返回给Connector 链接器，
最后Connector连接器将response返回给浏览器

3.// tomcat 关闭端口

- 默认只对本机地址开放
- 可以在本机通过telnet 127.0.0.1 8005访问
- 对Tomcat进行关闭操作
- telnet：ssh的上一个版本，无加密字符版远程连接，端口号：23

ps:判断某一台主机的一个端口是否开启?

```
[root@localhost ~]# yum -y install talnet
[root@localhost ~]# telnet 192.168.200.109 8080
Trying 192.168.200.109...
Connected to 192.168.200.109.
Escape character is '^]'.
Connection closed by foreign host.
? ? ? ? ? ? ? ?
```

```
[root@localhost ~]# telnet 127.0.0.1 8005
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
SHUTDOWN
Connection closed by foreign host.
[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh
[root@localhost ~]# netstat -lnpt | grep java
tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN      5422/java
tcp6       0      0 :::8009                 :::*                    LISTEN      5422/java
tcp6       0      0 :::8080                 :::*                    LISTEN      5422/java
```

4.// tomcat启动的默认端口号8080，可以根据需要进行修改

```
<Connector port="8080" protocol="HTTP/1.1".
connectionTimeout="20000"。
redirectPort="8443" />。
```

5.// tomcat启动AJP1.3连接器时默认的端口号，可以根据需要进行修改

<Connector port="8009" protocol="AUP/1.3" redirectPort="8443" />.

注释掉

国家信息安全漏洞共享平台

6.// tomcat定义虚拟主机时的配置及日志配置

```
<Host name=localhost"   appBase= 'webapps"
        unpackWARs="true" autoDeploy-="true">
        <Context docBase="/web/webapp path="" reloadable="flase >
        </Context>
    <Valve ClassName="org.apache.catalina.valves.AccessLogValve"
directory="logs"
        prefix="localhost_access_log" suffix=".txt"
        pattern="%h %I %u 9%t &amp;quot;%r&amp;quot; %s %b" />
</Host>
```

??????????

实验：自定义默认网站目录

页面部署及更换页面

1.首先在跟目录下建立一个web目录，并在里面建立一个 webapp目录，用于存放网站文件

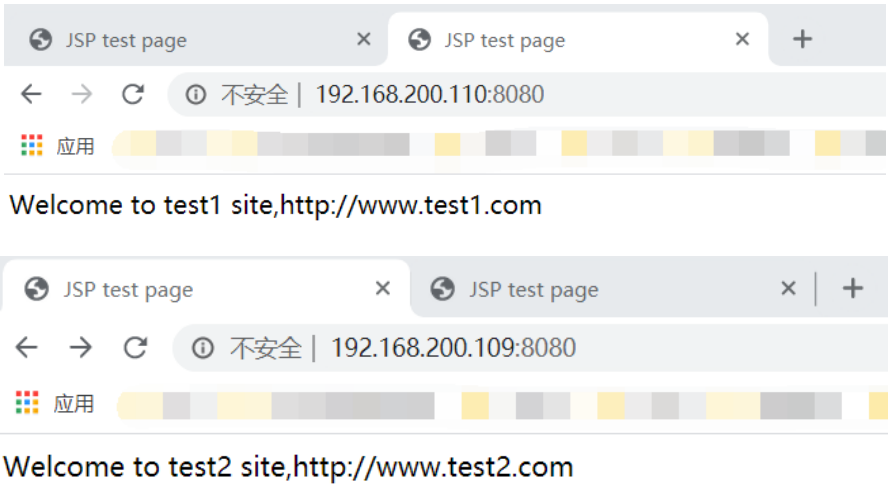
```
[root@localhost ~]# mkdir -pv /web/webapp    # p嵌套创建   v 详细信息
mkdir: 已创建目录 "/web"
mkdir: 已创建目录 "/web/webapp"
[root@localhost ~]# vim /web/webapp/index.jsp
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<html>
<head>
<title>JSP test page</title>
</head>
<body>
<% out.println("Welcome to test1 site,http://www.test1.com");%>
</body>
</html>
[root@localhost ~]# cp /usr/local/tomcat8/conf/server.xml{,.-$(date +%F)}      # 备份时间戳
[root@localhost ~]# vim /usr/local/tomcat8/conf/server.xml
150  <Context docBase="/web/webapp" path="" reloadable="flase" >
```

```
    </Context>
[root@localhost ~]#  /usr/local/tomcat8/bin/shutdown.sh
Using CATALINA_BASE:   /usr/local/tomcat8
Using CATALINA_HOME:   /usr/local/tomcat8
Using CATALINA_TMPDIR: /usr/local/tomcat8/temp
Using JRE_HOME:        /usr/local/java
Using CLASSPATH:
/usr/local/tomcat8/bin/bootstrap.jar:/usr/local/tomcat8/bin/tomcat-juli.jar
[root@localhost ~]#  /usr/local/tomcat8/bin/startup.sh
Using CATALINA_BASE:   /usr/local/tomcat8
Using CATALINA_HOME:   /usr/local/tomcat8
Using CATALINA_TMPDIR: /usr/local/tomcat8/temp
Using JRE_HOME:        /usr/local/java
Using CLASSPATH:
/usr/local/tomcat8/bin/bootstrap.jar:/usr/local/tomcat8/bin/tomcat-juli.jar
Tomcat started.
```

注意：Tomcat服务比较特殊，不能只看端口号，注意查看日志

```
[root@localhost ~]# netstat -lnpt | grep java
    tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN
    5593/java
    tcp6       0      0 :::8009                 :::*                    LISTEN
    5593/java
    tcp6       0      0 :::8080                 :::*                    LISTEN
    5593/java
[root@localhost ~]# tail -f /usr/local/tomcat8/logs/catalina.out
    06-Apr-2020 14:51:00.324 信息 [main]
    org.apache.catalina.startup.Catalina.start Server startup in 1258 ms
```

如果没有这条信息，那么会在上面显示报错信息，进行排错



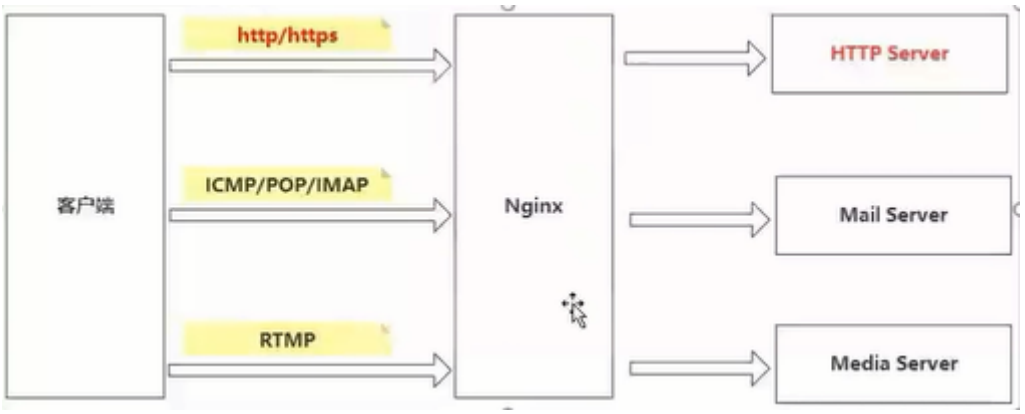
Tomcat：

- **提高并发处理能力**

- 提高服务可靠性

Nginx+Tomcat负载均衡集群(群集 Cluster)

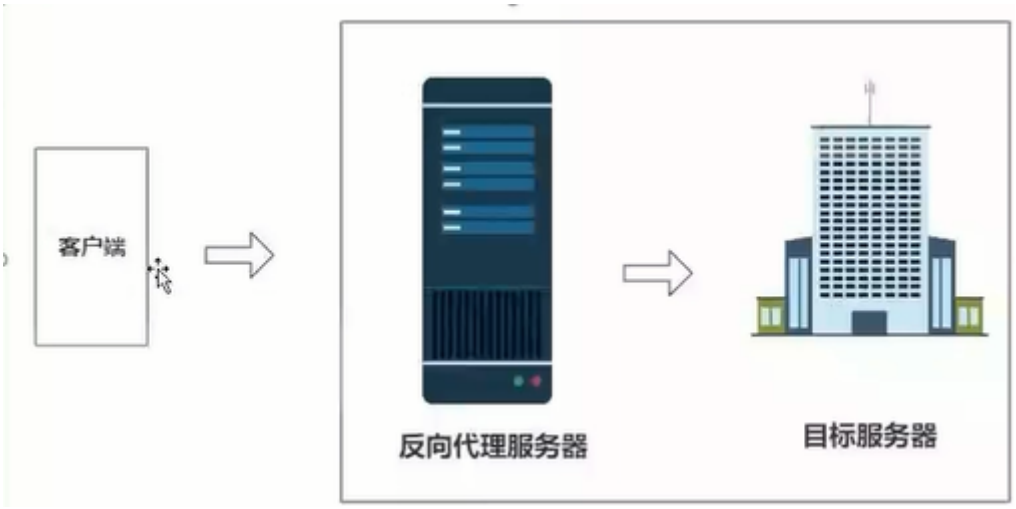
一、nginx服务器：网站服务、代理



- 代理服务器站在客户端那边就是正向代理
- 代理服务器站在原始服务器那边就是反向代理
- Nginx通过proxy_pass设置代理服务

正向代理

反向代理



- 一台Tomcat站点由于可能出现单点故障及无法应付过多客户复杂多样的请求等问题，
- 不能单独应用于生产环境下，所以需要一套可靠的解决方案来完善web站点架构
- Nginx是一款非常优秀的http服务器软件，
- 它能够支持高达50000个并发连接数的响应，
- 拥有强大的静态资源处理能力，运行稳定，并且内存，CPU等系统资源消耗非常低，
- 目前很多大型网站都用Nginx服务器做后端网站程序的反向代理及负载均衡器，
- 来提升整个站点的负载并发能力

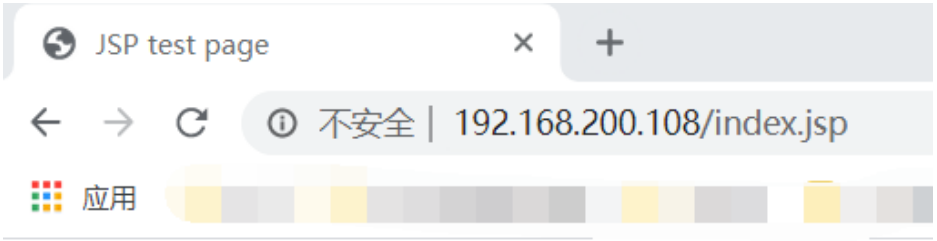
实验开始~

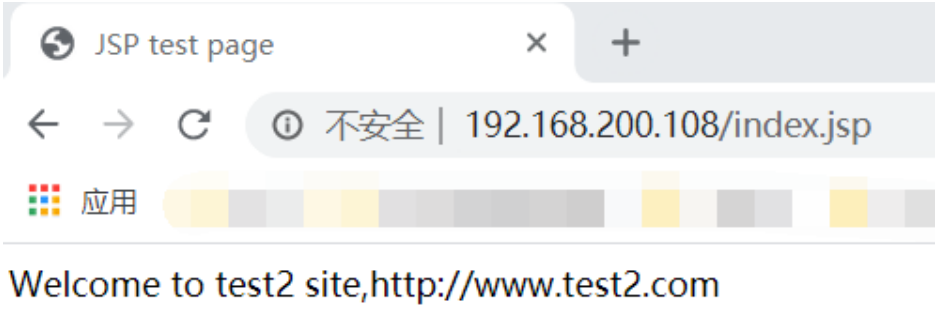
192.168.200.108　nginx

nginx服务配置

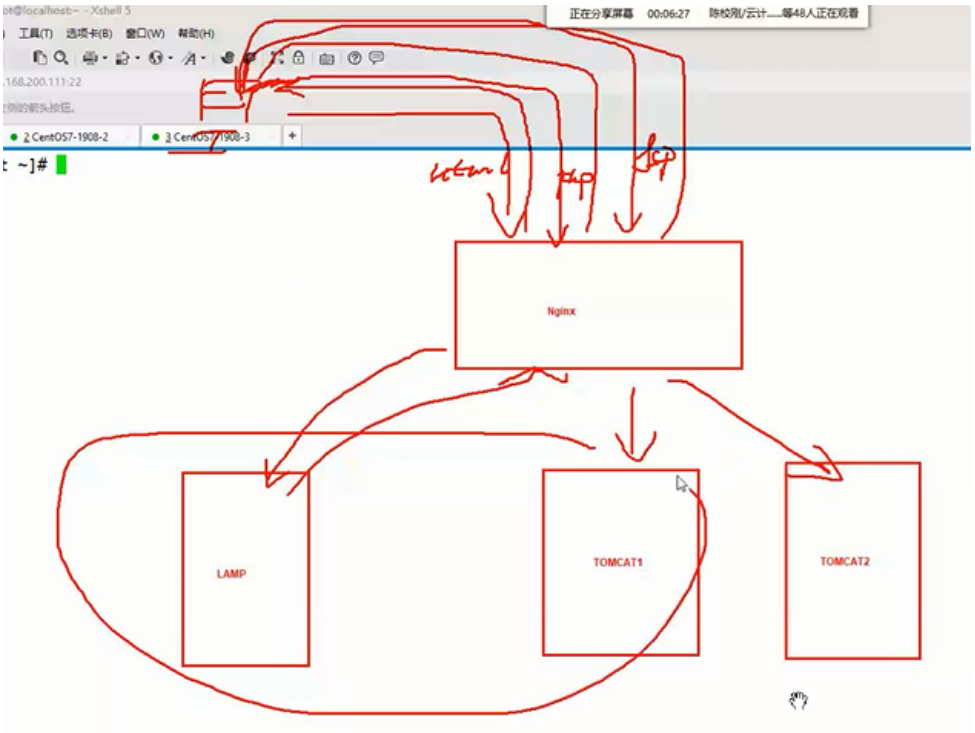
```
[root@localhost ~]# yum -y install pcre-devel zlib-devel openssl-devel
[root@localhost ~]# useradd -s /sbin/nologin -M nginx
[root@localhost ~]# tar xf nginx-1.14.2.tar.gz -C /usr/src
[root@localhost ~]# cd /usr/src/nginx-1.14.2/
[root@localhost nginx-1.14.2]# ./configure --prefix=/usr/local/nginx --user=nginx --group=nginx --with-http_stub_sta
tus_module --with-http_ssl_module --with-http_flv_module --with-http_gzip_static_module && make && make install
[root@localhost nginx-1.14.2]# cp /usr/local/nginx/conf/nginx.conf{,.bak}
[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf
```

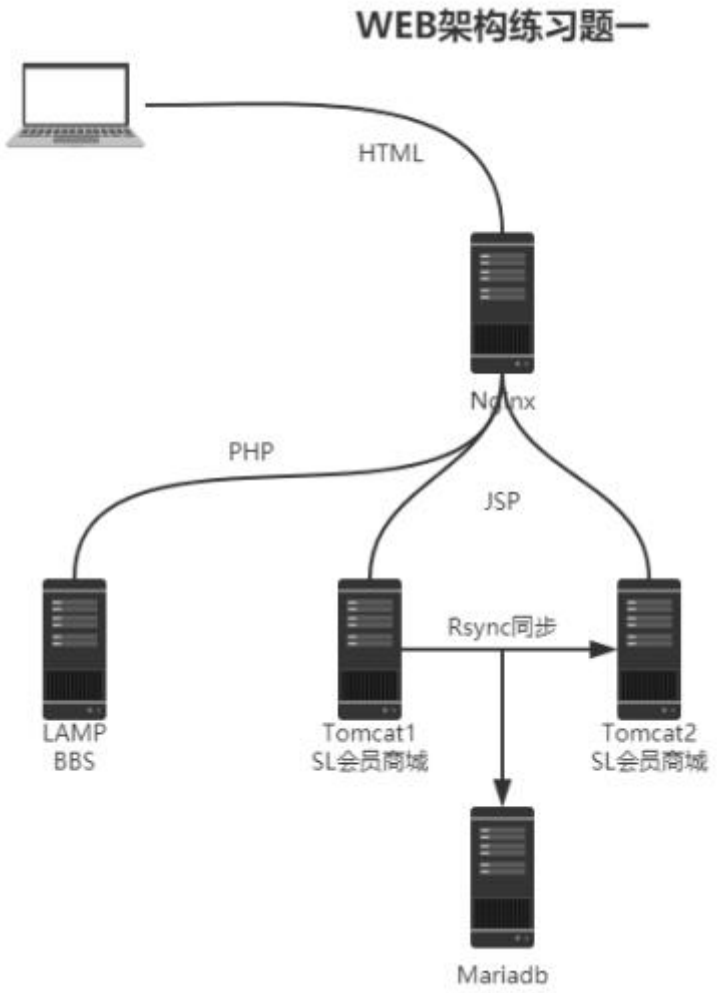
```
35    upstream tomcat {
        server 192.168.200.109:8080 weight=1;
        server 192.168.200.110:8080 weight=1;
      }
40    location ~ \.jsp$ {
            proxy_pass http://tomcat;
        }
[root@localhost ~]# /usr/local/nginx/sbin/nginx -t
[root@localhost ~]# netstat -lnpt | grep :80
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      62274/nginx: master
```





刷新交替处理，轮询





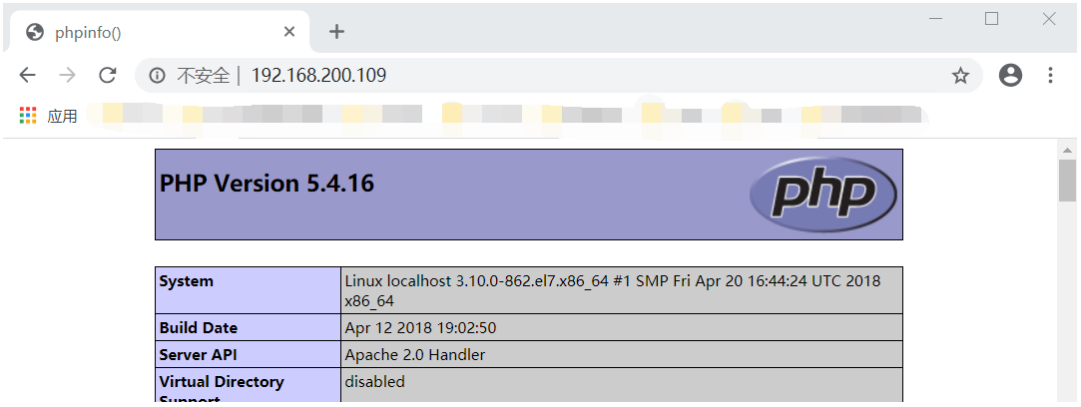
192.168.200.109 Tomcat1

```
[root@localhost ~]# yum -y install httpd mariadb mariadb-server php
[root@localhost ~]# systemctl start httpd
[root@localhost ~]# systemctl enable httpd
[root@localhost ~]# systemctl start mariadb
[root@localhost ~]# systemctl enable mariadb
[root@localhost ~]# mysqladmin -uroot password 123456
[root@localhost ~]# mysql -u root -p123456
MariaDB [(none)]>
MariaDB [(none)]> exit
Bye
```

写测试页：

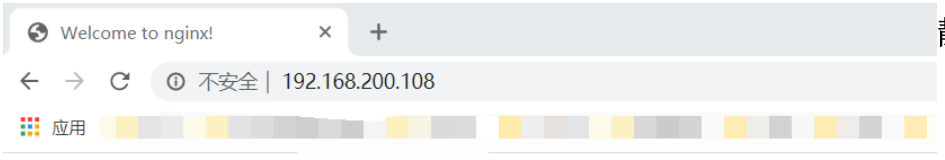
```
[root@localhost ~]# vim /var/www/html/index.php
<?php
phpinfo()
```

?>

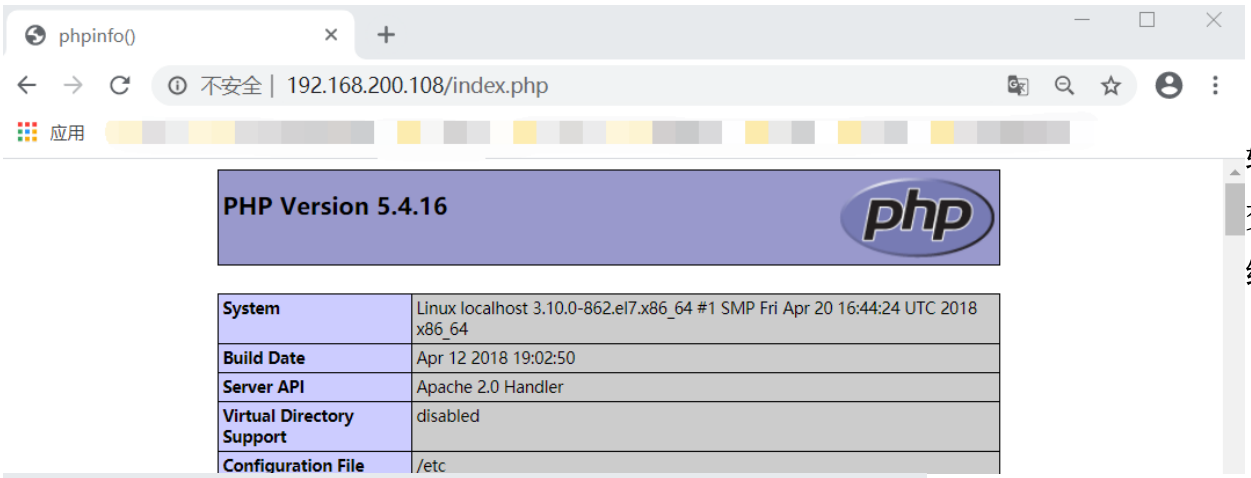


192.168.200.108　nginx

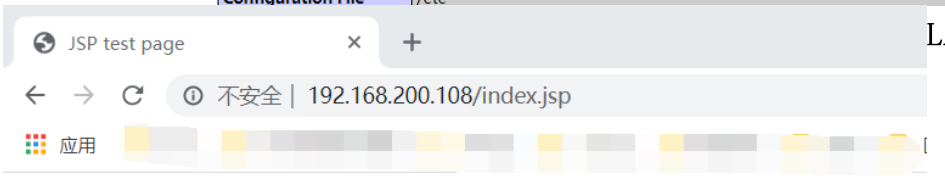
```
[root@localhost ~]# vim /usr/local/nginx/conf/nginx.conf
 57          location ~ \.php$ {
 58              proxy_pass http://192.168.200.109;
 59          }
```



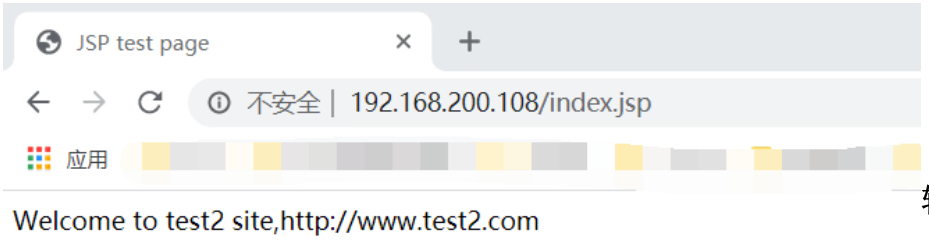
静态页面



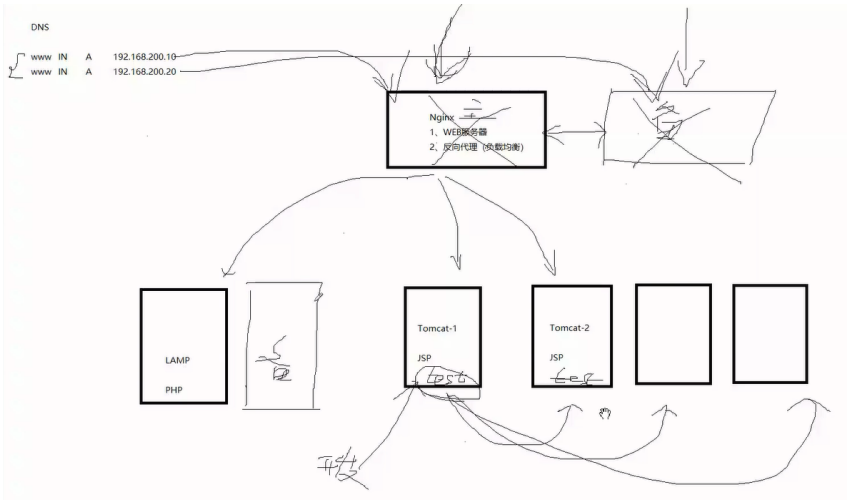
转交给LAMP，192.168.200.109



Welcome to test1 site,http://www.test1.com



转交给Tomcat，
192.168.200.109
且可轮询



二、Nginx 负载均衡算法

1. 轮询(默认，什么都未配置的情况下)

每个请求按时间顺序逐一分配到不同的后端服务,如果有一台死机,自动剔除

weight(轮询权值)

- **weight的值越大分配到的访问概率越高，**
- **主要用于后端每台服务器性能不均衡的情况下**
- **或者仅仅为在主从的情况下设置不同的权值，达到合理有效的地利用主机资源**

问题：用户访问时长不确定，占用资源时间不确定，会造成负载不均衡

2. least_conn　--LC　很多企业用

least-connected方式可以更公平的将负载分配到多个机器上面

使用least-connected，nginx不会将请求分发到繁忙的机器.上面

而且将新的请求分发的较清闲的机器上面

问题：

这种方法对于静态页面还好，

动态不行，动态页面有状态，刷新轮询会影响用户访问体验

3. ip hash

将用户与IP进行捆绑

每个请求按访问IP的哈希结果分配，

使来自同一个IP的访客固定访问一台后端服务器，

并且可以有效解决动态网页存在的session共享问题。

问题：常用IP所在机器出了故障，轮询到其他Tomcat上，且数据永久无法找回

很少用

解决：session同步数据 ---小规模可以，大规模就会繁琐耗时

解决：将所有数据都同步到一个地方（数据备份），需要时来调用即可

4. fair --更加智能

比weight. ip hash更加智能的负载均衡算法，

- fair算法可以根据页面大小和加载时间长短智能地进行负载均衡
- 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 本身不支持fair,如果需要这种调度算法，则必须安装upstream fair 模块

5. url hash

- 按访问的URL的哈希结果来分配请求
- 使每个URL定向到一台后端服务器
- 可以进一步提高后端缓存服务器的效率
- Nginx 本身不支持url hash
- 如果需要这种调度算法，则必须安装Nginx的hash软件包

三、Nginx负载均衡调度状态

在Nginx upstream模块中，可以设定每台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 ，即不工作

使用：软件或业务更新（访问量不大的时候），分组更新

上线时使用

backup: 预留的备份机器,用的不多

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，因此这台机器的访问压力最低

server 192.168.200.109:8080 weight=2 ;

server 192.168.200.110:8080 weight=1;

当上面两个所在机房断电或出故障时，找下面的机器

server 192.168.200.108:8080 weight=1 backup;

这台机器可以跑别的业务，暂时承担上面的服务

后端数据的健康检查：max_fails　　fail_timeout

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:请求失败超时时间，

在经历了max_fails　次失败后，暂停服务的时间。

max_fails和fail_timeout　可以一起使用

使用：检查主机状态存活

server 192.168.200.109:8080 weight=1 max_fails=3 fail_timeout=10s;

server 192.168.200.110:8080 weight=1 max_fails=3 fail_timeout=10s;

四、Tomcat连接数据库

[root@localhost ~]# tar xf SLSaleSystem.tar.gz -C /web/webapp/

[root@localhost ~]# ls /web/webapp/

index.jsp SLSaleSystem

[root@localhost ~]# vim /usr/local/tomcat8/conf/server.xml

150　　<Context docBase="/web/webapp/SLSaleSystem" path=""

reloadable="flase" >

数据库

[root@localhost ~]# yum -y install mariadb mariadb-server

[root@localhost ~]# systemctl start mariadb

[root@localhost ~]# systemctl enable mariadb

Created symlink from /etc/systemd/system/multi-user.target.wants/mariadb.service to /usr/lib/systemd/system/mariadb.service.
[root@localhost ~]# mysql
MariaDB [(none)]> create database slsaledb;
Query OK, 1 row affected (0.00 sec)
MariaDB [(none)]> grant all on slsaledb.* to admin@'%' identified by '123456';
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.00 sec)
MariaDB [(none)]> exit
Bye
数据库配置完成

导入数据库文件

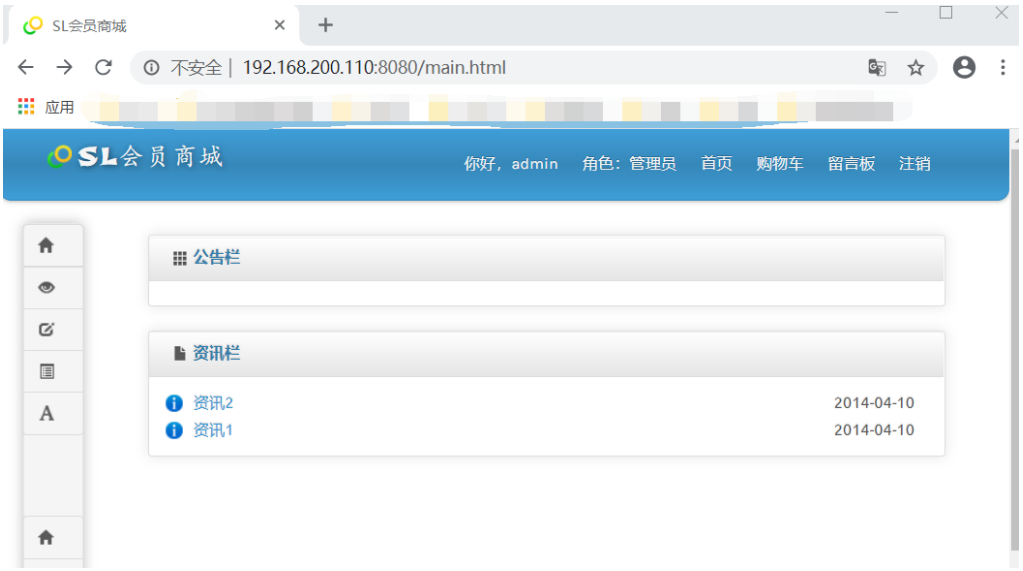
[root@localhost ~]# mysql -uroot < slsaledb-2014-4-10.sql

修改设置：

[root@localhost ~]# vim /web/webapp/SLSaleSystem/WEB-INF/classes/jdbc.properties

[root@localhost ~]# /usr/local/tomcat8/bin/shutdown.sh

[root@localhost ~]# /usr/local/tomcat8/bin/startup.sh



五、FPM制作rpm包

软件的安装方式：rpm

rpm的管理工具：yum　　无法定义软件包功能

源码包安装：功能自定义

------把自己用的源码包环境，基于此，做一个属于自己的定制完成的软件包----rpm包

-----以rpm包的安装速度安装，但环境是源码包的环境

fpm环境

实验目的：将目录做成一个rpm源码包

概述

1、支持的源类型包

- ➢dir:将目录打包成所需要的类型，可以用于源码编译安装的软件包
- ➢rpm:对rpm进行转换
- ➢gem:对rubygem包进行转换
- ➢python: 将Python模块打包成相应的类型

2、支持目标类型包

- ➢rpm:转换为rpm包
- ➢deb:转换为deb包
- ➢solaris:转换为solaris包
- ➢puppet:转换为puppet包

3、FPM常用参数

-s: 指定源类型

-t: 指定目标类型，即想要制作为什么包

-n: 指定包的名字

-V: 指定包的版本号。

-C: 指定打包的相对路径。

-d: 指定依赖于哪些包。

-f: 第二次包时目录下如果有同名安装包存在，则覆盖它。

-p: 制作的rpm安装包存放路径，不想放在当前目录下就需要指定

--post-install: 软件包安装完成之后所要运行的脚本;同--offer-install

--pre-install: 软件包安装完成之前所要运行的脚本;同--before-install

--post-uninstall: 软件包卸载完成之后所要运行的脚本;同--offer-remove

--pre-uninstall: 软件包卸载完成之前所要运行的脚本:同- -before-remove

--prefix: 制作好的rpm包默认安装路径

4、基于FPM制作RPM包

4.1、 搭建Epel Yum源

```
[root@localhost yum.repos.d]# ls
bakup  epel-release-latest-7.noarch.rpm  local.repo
```

4.2、安装ruby环境和gem命令FPM

gem命令是从rubygem仓库安装软件，类似yum从yum仓库安装软件

```
[root@localhost ~]# yum install -y ruby rubygems ruby-devel
[root@localhost ~]# gem sources -l
*** CURRENT SOURCES ***
https://rubygems.org/
[root@localhost ~]# gem install rubygems-update -v 2.3.0
Fetching: rubygems-update-2.3.0.gem (100%)
Successfully installed rubygems-update-2.3.0
Parsing documentation for rubygems-update-2.3.0
Installing ri documentation for rubygems-update-2.3.0
1 gem installed
[root@localhost ~]# gem update --system
Updating rubygems-update
Fetching: rubygems-update-3.1.2.gem (100%)
ERROR:  Error installing rubygems-update:
        rubygems-update requires Ruby version >= 2.3.0.
Installing RubyGems 2.3.0
RubyGems 2.3.0 installed
Parsing documentation for rubygems-2.3.0
Installing ri documentation for rubygems-2.3.0
[root@localhost ~]# gem source -a http://mirrors.aliyun.com/rubygems/        #
添加国内阿里源
http://mirrors.aliyun.com/rubygems/ added to sources
[root@localhost ~]# gem sources --remove https://rubygems.org/            # 移除
国外源，加快下载速度
https://rubygems.org/ removed from sources
[root@localhost ~]# gem sources -l       #查看当前源
*** CURRENT SOURCES ***
http://mirrors.aliyun.com/rubygems/
[root@localhost ~]# gem install fpm         # 安装FPM工具
```

4.3 编译nginx

```
[root@localhost ~]# tar xf nginx-1.16.0.tar.gz -C /usr/src/
[root@localhost ~]# cd /usr/src/nginx-1.16.0/
[root@localhost nginx-1.16.0]# yum -y install pcre-devel zlib-devel
```

```
[root@localhost nginx-1.16.0]# ./configure --prefix=/usr/local/nginx --with-
pcre && make && make install
```

4.4 打包nginx生成RPM包

nginx配置脚本

```
[root@localhost ~]# vim nginx.sh
#!/bin/bash
useradd -M -s /sbin/nologin nginx
ln -s /usr/local/nginx/sbin/nginx  /sbin/
echo www.kitty.com > /usr/local/nginx/html/index.html
nginx
[root@localhost ~]# fpm -s dir -t rpm -n nginx -v 1.16.0 -d 'pcre-devel,zlib-devel' -f
--post-install /
root/nginx.sh /usr/local/nginx
Doing `require 'backports'` is deprecated and will not load any backport in the next
major release.
Require just the needed backports instead, or 'backports/latest'.
Created package {:path=>"nginx-1.16.0-1.x86_64.rpm"}
[root@localhost ~]# ls
anaconda-ks.cfg                initial-setup-ks.cfg      nginx-1.16.0.tar.gz       sbin
epel-release-latest-7.noarch.rpm    nginx-1.16.0-1.x86_64.rpm   nginx.sh
[root@localhost ~]# rpm -qpi nginx-1.16.0-1.x86_64.rpm       # 查看未安装的软件包
Name        : nginx
Version     : 1.16.0
Release     : 1
Architecture: x86_64
Install Date: (not installed)
Group       : default
Size        : 3856910
License     : unknown
Signature   : (none)
Source RPM  : nginx-1.16.0-1.src.rpm
Build Date  : 2020年04月07日 星期二 19时50分22秒
Build Host  : localhost
Relocations : /
Packager    : <root@localhost>
Vendor      : root@localhost
URL         : http://example.com/no-uri-given
Summary     : no description given
Description :
no description given
```

注意：

- 安装这个软件包的前提：[root@localhost ~]# yum -y install pcre-devel zlib-devel
- 事先准备好nginx脚本，放在/root下

企业使用这各方法的原因：

1.根据需求自定义软件包；

安装方便快捷（在做大规模部署时）

2.生产环境的软件的迁移

作业：WEB架构部署