一　tomcat

常见servlet容器

IBM websphere 银行居多

Oracle weblogic

Apache tomcat

RedHat Jboss

12306 jboss 阿里云

2，部署Tomcat服务器软件

1）使用RPM安装JDK环境

[root@web1 ~]# yum -y install java-1.8.0-openjdk //安装JDK

[root@web1 ~]# yum -y install java-1.8.0-openjdk-headless //安装JDK

[root@web1 ~]# java -version //查看JAVA版

2）安装Tomcat（apache-tomcat-8.0.30.tar.gz软件包，在lnmp\_soft中有提供）

[root@web1 ~]# tar -xf apache-tomcat-8.0.30.tar.gz

[root@web1 ~]# mv apache-tomcat-8.0.30 /usr/local/tomcat

[root@web1 ~]# ls /usr/local/tomcat

bin/ //主程序目录

lib/ //库文件目录

logs/ //日志目录

temp/ //临时目录

work/ //自动编译目录jsp代码转换servlet

conf/ //配置文件目录

webapps/ //页面目录

3）启动服务

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

[root@web1 ~]# firewall-cmd --set-default-zone=trusted

[root@web1 ~]# setenforce 0

4）服务器验证端口信息

[root@web1 ~]# netstat -nutlp |grep java //查看java监听的端口

tcp 0 0 :::8080 :::\* LISTEN 2778/java

tcp 0 0 ::ffff:127.0.0.1:8005 :::\* LISTEN 2778/java

提示：如果检查端口时，8005端口启动非常慢，可用使用下面的命令用urandom替换random（非必须操作）。

[root@web1 ~]# mv /dev/random /dev/random.bak

[root@web1 ~]# ln -s /dev/urandom /dev/random

5）客户端浏览测试页面

[root@client ~]# firefox <http://192.168.2.100:8080>

3 配置服务器虚拟主机

1）修改server.xml配置文件，创建虚拟主机

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

… …

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

</Host>

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

</Host>

2）创建虚拟主机对应的页面根路径

[root@web1 ~]# mkdir -p /usr/local/tomcat/{a,b}/ROOT

[root@web1 ~]# echo "AAA" > /usr/local/tomcat/a/ROOT/index.html

[root@web1 ~]# echo "BBB" > /usr/local/tomcat/b/ROOT/index.html

3）重启Tomcat服务器

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

4）客户端设置host文件，并浏览测试页面进行测试(proxy充当客户端角色)

注意：ssh远程连接时使用使用-X参数才可以！！！

[root@proxy ~]# vim /etc/hosts

… …

192.168.2.100 www.a.com www.b.com

[root@proxy ~]# firefox http://www.a.com:8080/ //注意访问的端口为8080

[root@proxy ~]# firefox <http://www.b.com:8080/>

4 修改www.b.com网站的首页目录为base

1）使用docBase参数可以修改默认网站首页路径

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

… …

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

</Host>

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

<Context path="" docBase="base" reloadable="true"/> #决定网站的根目录，默认ROOT

</Host>

… …

[root@web1 ~]# mkdir /usr/local/tomcat/b/base

[root@web1 ~]# echo "BASE" > /usr/local/tomcat/b/base/index.html

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

reloadable:如果这个属性设为true，tomcat服务器在运行状态下会监视在WEB-INF/classes和WEB-INF/lib目录下class文件的改动，如果监测到有class文件被更新的，服务器会自动重新加载Web应用。在开发阶段将reloadable属性设为true，有助于调试servlet和其它的class文件，但这样用加重服务器运行负荷，建议在Web应用的发存阶段将reloadable设为false。

2）测试查看页面是否正确(proxy充当客户端角色)

[root@proxy ~]# firefox http://www.b.com:8080/ //结果为base目录下的页面内容

5 tomcat跳转

1）当用户访问http://www.a.com/test打开/var/www/html目录下的页面

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

… …

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

<Context path="/test" docBase="/var/www/html/" />

</Host>

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

<Context path="" docBase="base" />

</Host>

… …

[root@web1 ~]# echo "Test" > /var/www/html/index.html

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

2）测试查看页面是否正确(proxy充当客户端角色)

[root@proxy ~]# firefox http://www.a.com:8080/test

//返回/var/www/html/index.html的内容

//注意，访问的端口为8080

6 配置Tomcat支持SSL加密网站

[root@web1 jluo]# keytool --help

-certreq 生成证书请求

-changealias 更改条目的别名

-delete 删除条目

-exportcert 导出证书

-genkeypair 生成密钥对

-genseckey 生成密钥

-gencert 根据证书请求生成证书

-importcert 导入证书或证书链

-importpass 导入口令

-importkeystore 从其他密钥库导入一个或所有条目

-keypasswd 更改条目的密钥口令

-list 列出密钥库中的条目

-printcert 打印证书内容

-printcertreq 打印证书请求的内容

-printcrl 打印 CRL 文件的内容

-storepasswd 更改密钥库的存储口令

[root@web1 jluo]# keytool -genkeypair -help

-alias <alias> 要处理的条目的别名

-keyalg <keyalg> 密钥算法名称

-keysize <keysize> 密钥位大小

-sigalg <sigalg> 签名算法名称

-destalias <destalias> 目标别名

-dname <dname> 唯一判别名

-startdate <startdate> 证书有效期开始日期/时间

-ext <value> X.509 扩展

-validity <valDays> 有效天数

-keypass <arg> 密钥口令

-keystore <keystore> 密钥库名称

-storepass <arg> 密钥库口令

-storetype <storetype> 密钥库类型

-providername <providername> 提供方名称

-providerclass <providerclass> 提供方类名

-providerarg <arg> 提供方参数

-providerpath <pathlist> 提供方类路径

-v 详细输出

-protected 通过受保护的机制的口令

1）创建加密用的私钥和证书文件

[root@web1 ~]# keytool -genkeypair -alias tomcat -keyalg RSA -keystore /usr/local/tomcat/keystore //提示输入密码为:123456

//-genkeypair 生成密钥对

//-alias tomcat 密钥别名

//-keyalg RSA 定义密钥算法为RSA算法

//-keystore 定义密钥文件存储在:/usr/local/tomcat/keystore

2)再次修改server.xml配置文件，创建支持加密连接的Connector

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

… …

<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"

maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

keystoreFile="/usr/local/tomcat/keystore" keystorePass="123456" clientAuth="false" sslProtocol="TLS" />

//备注，默认这段Connector被注释掉了，打开注释，添加密钥信息即可

keystoreFile=”密钥绝对路径”　keystorePass=”密钥密码”

3）重启Tomcat服务器

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

4）客户端设置host文件，并浏览测试页面进行测试(proxy充当客户端角色)

[root@proxy ~]# vim /etc/hosts

… …

192.168.2.100 www.a.com www.b.com

[root@proxy ~]# firefox https://www.a.com:8443/

[root@proxy ~]# firefox https://www.b.com:8443/

[root@proxy ~]# firefox https://192.168.2.100:8443/

7配置Tomcat日志

1)为每个虚拟主机设置不同的日志文件

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

.. ..

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

<Context path="/test" docBase="/var/www/html/" />

#从默认localhost虚拟主机中把Valve这段复制过来，适当修改下即可

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"　　　　　＃日志存放目录

prefix=" a\_access" suffix=".txt"　　＃文件名，扩展名

pattern="%h %l %u %t &quot;%r&quot; %s %b" />

</Host>

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

<Context path="" docBase="base" />

<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"

prefix=" b\_access" suffix=".txt"

pattern="%h %l %u %t &quot;%r&quot; %s %b" />

</Host>

.. ..

2）重启Tomcat服务器

[root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

3）查看服务器日志文件

[root@web1 ~]# ls /usr/local/tomcat/logs/

8　扩展实验(配置Tomcat集群)

1) 在192.168.4.5主机上配置Nginx调度器（具体安装步骤参考前面的章节）

[root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf

http{

upstream toms {

server 192.168.2.100:8080;

server 192.168.2.200:8080;

}

server {

listen 80;

server\_name localhost;

location / {

proxy\_pass http://toms;

}

}

}

2) 在192.168.2.100和192.168.2.200主机上配置Tomcat调度器

以下以Web1为例：

[root@web1 ~]# yum -y install java-1.8.0-openjdk //安装JDK

[root@web1 ~]# yum -y install java-1.8.0-openjdk-headless //安装JDK

[root@web1 ~]# tar -xzf apache-tomcat-8.0.30.tar.gz

[root@web1 ~]# mv apache-tomcat-8.0.30 /usr/local/tomcat

3）启动服务

[root@web1 ~]# /usr/local/tomcat/bin/startup.sh

4) 客户端验证

为了防止有数据缓存，可以使用真实主机的google-chrome访问代理服务器，输入Ctrl+F5刷新页面。

9使用Varnish加速Web

CDN,　content ，delivery， network

CDN，varnish,squid

构建Web服务器

1）使用yum安装web软件包

[root@web1 ~]# yum -y install httpd

2）启用httpd服务（注意需要关闭nginx，否则端口冲突）

[root@web1 ~]# systemctl start httpd

[root@web1 ~]# setenforce 0

[root@web1 ~]# firewall-cmd --set-default-zone=trusted

httpd服务默认通过TCP 80端口监听客户端请求：

[root@web1 ~]# netstat -anptu | grep httpd

tcp 0 0 :::80 :::\* LISTEN 2813/httpd

3）为Web访问建立测试文件

在网站根目录/var/www/html下创建一个名为index.html的首页文件：

[root@web1 ~]# cat /var/www/html/index.html

192.168.2.100

4）测试页面是否正常（代理服务器测试后台web）

[root@proxy ~]# firefox http://192.168.2.100

部署Varnish缓存服务器(192.168.4.5)

1）编译安装软件(python-docutils默认光盘中没有，需要在lnmp\_soft中找)

[root@proxy ~]# yum -y install gcc readline-devel //安装软件依赖包

[root@proxy ~]# yum -y install ncurses-devel //安装软件依赖包

[root@proxy ~]# yum -y install pcre-devel //安装软件依赖包

[root@proxy ~]# yum -y install \

python-docutils-0.11-0.2.20130715svn7687.el7.noarch.rpm //安装软件依赖包

[root@proxy ~]# useradd -s /sbin/nologin varnish //创建账户

[root@proxy ~]# tar -xf varnish-5.2.1.tar.gz

[root@proxy ~]# cd varnish-5.2.1

[root@proxy varnish-5.2.1]# ./configure

[root@proxy varnish-5.2.1]# make && make install

2）复制启动脚本及配置文件

[root@proxy varnish-5.2.1]# cp etc/example.vcl /usr/local/etc/default.vcl

3）修改代理配置文件

[root@proxy ~]# vim /usr/local/etc/default.vcl

backend default {

.host = "192.168.2.100";

.port = "80";

}

4）启动服务

[root@proxy ~]# varnishd -f /usr/local/etc/default.vcl

//varnishd命令的其他选项说明如下：

//varnishd –s malloc,128M 定义varnish使用内存作为缓存，空间为128M

//varnishd –s file,/var/lib/varnish\_storage.bin,1G 定义varnish使用文件作为缓存

10

11