TOMCAT服务

案例1:安装部署tomcat服务器

1.1问题

本案例要求部署tomcat服务器,集体要求如下:

\*安装部署JDK基础环境

\*安装部署tomcat服务器

\*创建JSP测试页面,文件名为test.jsp,显示服务器当前时间

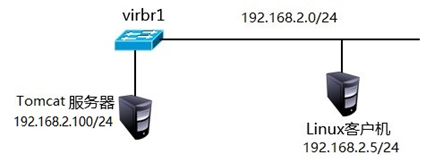
然后客户及访问此WEB服务验证效果:

\*使用火狐浏览器访问Tomcat服务器的8080端口,浏览默认首页

\*使用火狐浏览器访问Tomcat服务器的8080端口,浏览默认测试页

1.2方案

使用2台RHEL7虚拟机，其中一台作为Tomcat服务器（192.168.2.100）、另外一台作为测试用的Linux客户机（192.168.2.5），如图-1所示



使用RPM安装JDK基础环境

使用源码安装部署Tomcat服务器

1.3步骤

步骤一:部署tomcat服务器软件(192.168.2.100/24)

[1]使用RPM安装JDK环境

]#yum -y install java-1.8.0-openjdk

]#yum -y install java-1.8.0-openjdk-headless

]#java -version

[2]安装Tomcat(apache-tomcat-8.0.30.tar.gz)

[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

]#/usr/local/tomcat/bin/start.sh

[4]服务验证端口信息

]#netstat -nutlp | grep java

提示：如果检查端口时，8005端口启动非常慢，默认tomcat启动需要从/dev/random读取大量的随机数据，默认该设备生成随机数据的速度很慢，可用使用下面的命令用urandom替换random（非必须操作）。

]#mv /dev/random /dev/random.bak

]#ln -s /dev/urandom /dev/random

另外,还可以使用的方案二解决

[root@web1 ~]# yum install rng-tools

[root@web1 ~]# systemctl start rngd

[root@web1 ~]# systemctl enable rngd

[5]客户端列蓝测试页面(proxy作为客户端)

firefox <http://192.168.2.100:8080>

步骤二:修改tomcat配置文件

[1]创建测试JSP页面

[root@web1 ~]# vim /usr/local/tomcat/webapps/ROOT/test.jsp

<html>

<body>

<center>

Now time is: <%=new java.util.Date()%>            //显示服务器当前时间

</center>

</body>

</html>

步骤三:验证测试

[1]客户端浏览器测试页面(proxy充当客户端角色)

]#firefox <http://192.168.2.100:8080>

]#firefox <http://192.168.2.100:8080>/test.jsp

案例2:使用Tomcat部署虚拟机

**2.1问题**

沿用练习二,使用tomcat部署加密虚拟机,实现以下要求:

\*实现两个基于域名的虚拟主机，域名分别为：www.a.com和 www.b.com

\*使用www.a.com域名访问的页面根路径为/usr/local/tomcat/a/ROOT

\*使用www.b.com域名访问的页面根路径为/usr/local/tomcat/b/base

\*访问www.a.com/test时，页面自动跳转到/var/www/html目录下的页面

\*访问页面时支持SSL加密通讯

\*私钥、证书存储路径为/usr/local/tomcat/conf/cert

\*每个虚拟主机都拥有独立的访问日志文件

\*配置tomcat集群环境

**2.2方案**

修改server.xm配置文件,创建两个域的虚拟机,修改如下连个参数

# cat /usr/local/tomcat/conf/server.xml

<Server>

<Service>

<Connector port=8080 />

<Connector port=8009 />

<Engine name="Catalina" defaultHost="localhost">

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

</Host>

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

</Host>

… …

2.3步骤

**步骤一:配置服务器虚拟机**

[1]修改server.xml配置文件,创建虚拟机

[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Tomct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,并浏览测试页进行测试,(proxy充当客户端角色)

注意:ssh远程连接时使用-X参数才行可以!

[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/

**步骤二: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"/>

</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

[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> //结果为basse目录下的页面内容

**步骤三: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)

]#firefox ://www.a.com:8080/test

//返回/var/www/html/index.html的内容

//注意,访问的端口为8080

**步骤四:配置Tomcat支持SSL加密证书**

[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

[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,并浏览测试页进行测试,(proxy充当客户端角色)

**步骤五:部署Tomcat日志**

[1]为每个虚拟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"

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

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]查看服务日志文件

]#ls /usr/local/tomcat/logs

**步骤六: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

]#/use/local/tomcat/bin/start.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。

**案例3:使用Varnish加速Web**

3.1问题

通过Varnish缓存服务器,实现如下目录:

使用Varnish加速后端Web服务

代理服务器可以将远程的Web服务器页面缓存在本地

远程Web服务器对客户端用户是透明的

利用缓存机制提高网站的响应速度

使用varnishadm命令管理缓存页面

使用varnishstat命令查看Varnish状态

3.2方案

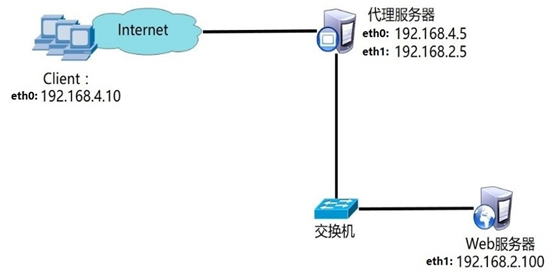
通过源码编译安装Varnish缓存服务

编译安装Varnish软件

修改配置文件,缓存代理源web服务器,实现web加速功能

使用3台RHEL7虚拟机，其中一台作为Web服务器（192.168.2.100）、一台作为Varnish代理服务器（192.168.4.5,192.168.2.5)，另外一台作为测试用的Linux客户机（192.168.4.10），

如图-2所示。



对于Web服务器的部署，此实验中仅需要安装nginx或者httpd软件、启动服务，并生成测试首页文件即可，默认httpd网站根路径为/var/www/html，首页文档名称为index.html，默认nginx网站根路径为/usr/local/nginx/html，默认首页为index.html。下面的实验我们以httpd为例作为Web服务器。

3.3:步骤

步骤一:构建Web服务器

[1]使用yum安装web软件包

]#yum -y install httpd

[2]启用服务(注意要关闭nginx,否则端口冲突)

]#systemctl start httpd

http服务默认通过TCP80端口监听客户端请求

ss -anptu |grep httpd

[3]为Web访问建立测试文件

]#cat /var/www/html/index.html

[4]测试页面是否正常(代理服务器测试后台web)

]#firefox:192.168.2.100

步骤二:部署varnish缓存服务器(192.168.4.5)

[1]编译安装软件

[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         //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(注意相对路径与绝对路径)

]#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

步骤三:客户端测试

[1]客户端开启浏览器访问

]#curl <http://192.168.4.5>

步骤四:其他操作

[1]查看Varnish日志

]#varnishlog //varnish日志

]#varnishlog //访问日志

[2]更新缓存数据,在后台web服务器更新页面内容后,用户访问代理服务器看到的还是之前的数据

]#varnishadm

varnish>ban req.url~.\*

//清空缓存数据,支持正则表达式