6_Operation05Tomcat 服务器应用案例 Varnish 代理服务器

一 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，具体要求如下：

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

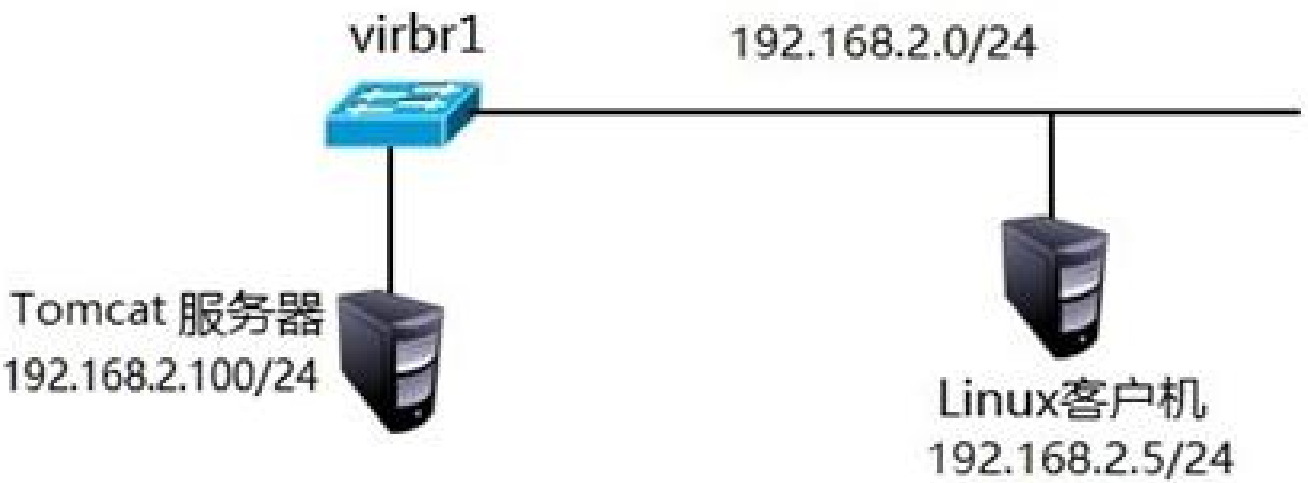
然后客户机访问此 Web 服务器验证效果：

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，浏览默认测试页面

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

1.3.1 部署 Tomcat 服务器软件(192.168.2.100/24)

1）使用 RPM 安装 JDK 环境

```
web1 ~]# yum -y install java-1.8.0-openjdk  #安装 JDK
web1 ~]# yum -y install java-1.8.0-openjdk-headless  #安装 JDK
web1 ~]# java -version  #查看 JAVA 版本
```

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

```
web1 ~]# tar -xf  apache-tomcat-8.0.30.tar.gz
web1 ~]# mv apache-tomcat-8.0.30  /usr/local/tomcat #移动并重命名
web1 ~]# ls /usr/local/tomcat
bin/          #主程序目录,内含 startup.sh shutdown.sh 两个重要文件
lib/          #库文件目录,类库(类),相当于 shell 的函数库
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,网页根目录
```

3）启动服务

```
web1 ~]# /usr/local/tomcat/bin/startup.sh
```

4）服务器验证端口信息

```
web1 ~]# netstat -nutlp |grep java  #查看 java 监听的端口
tcp  0  0  :::8080  :::*  LISTEN  2778/java
tcp  0  0  :::8009  :::*  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，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）。

```
web1 ~]# mv /dev/random  /dev/random.bak
web1 ~]# ln -s /dev/urandom  /dev/random
strings /dev/random    #查看软连接/dev/random 的内容
```

另外，还可以使用方案二解决：

```
web1 ~]# yum install rng-tools
web1 ~]# systemctl start rngd
web1 ~]# systemctl enable rngd
```

5）客户端浏览测试页面(proxy 作为客户端)

```
proxy ~]# firefox http://192.168.2.100:8080
```

1.3.2 修改 Tomcat 配置文件

1）创建测试 JSP 页面

```
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>
```

1.3.3 验证测试

1）客户端浏览测试页面(proxy 充当客户端角色)

```
proxy ~]# firefox http://192.168.2.100:8080
proxy ~]# firefox http://192.168.2.100:8080/test.jsp
```

二 案例 2：使用 Tomcat 部署虚拟主机

2.1 问题

沿用练习二，使用 Tomcat 部署加密虚拟主机，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.xml 配置文件，创建两个域名的虚拟主机，修改如下两个参数块：

```
# cat /usr/local/tomcat/conf/server.xml
<Server>    #基本配置
   <Service>
     <Connector port=8080 />   #监听端口 8080
     <Connector port=8009 />   #监听端口 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>
  </Service>
</Server>
```

2.3 步骤

2.3.1 配置服务器虚拟主机

1）修改 **server.xml** 配置文件，创建虚拟主机

```
web1 ~]# vim /usr/local/tomcat/conf/server.xml
<Host name="localhost"  appBase="webapps"
unpackWARs="true" autoDeploy="true">    #复制此 2 两行,并在其上面粘贴
      <Host name="www.a.com" appBase="a"
      unpackWARS="true" autoDeploy="true">
      </Host>    #手动添加此行,并严格注意大小写
      <Host name="www.b.com" appBase="b"
       unpackWARS="true" autoDeploy="true">
      </Host>
```

appBase 定义基础目录,基础目录下可以有很多项目,默认项目 **ROOT**

docBase 定义首页路径,默认为 **ROOT**

unpackWARS 定义自动解压 **war** 包

autoDeploy 自动更新网页内容

path 类似于 **nginx** 的地址跳转

2）创建虚拟主机对应的页面根路径

```
web1 ~]# mkdir -p  /usr/local/tomcat/{a,b}/ROOT
web1 ~]# echo "AAA"   > /usr/local/tomcat/a/ROOT/index.html
web1 ~]# echo "BBB" > /usr/local/tomcat/b/ROOT/index.html
```

3）重启 Tomcat 服务器

```
web1 ~]# /usr/local/tomcat/bin/shutdown.sh    #关闭
web1 ~]# /usr/local/tomcat/bin/startup.sh     #启动
```

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)

```
proxy ~]# vim /etc/hosts
192.168.2.100      www.a.com  www.b.com
proxy ~]# firefox http://www.a.com:8080/    #注意访问的端口为 8080
proxy ~]# firefox http://www.b.com:8080/
```

2.3.2 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

```
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>
web1 ~]# mkdir  /usr/local/tomcat/b/base
web1 ~]# echo "BASE" > /usr/local/tomcat/b/base/index.html
```

```
web1 ~]# /usr/local/tomcat/bin/shutdown.sh    #关闭
web1 ~]# /usr/local/tomcat/bin/startup.sh     #启动
```

2）测试查看页面是否正确(proxy 充当客户端角色)

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

2.3.3 跳转

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

```
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>
web1 ~]# echo "Test" > /var/www/html/index.html
web1 ~]# /usr/local/tomcat/bin/shutdown.sh    #关闭
web1 ~]# /usr/local/tomcat/bin/startup.sh     启动
```

2）测试查看页面是否正确(proxy 充当客户端角色)

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

2.3.4 配置 Tomcat 支持 SSL 加密网站(1 次部署,使用于所有网站)

1）创建加密用的私钥和证书文件

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

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

#默认这段 Connector 被注释掉了<!--　-->，打开注释，添加密钥信息即可

#添加时注意空格,keystorePass 为上一步按提示输入的密码

3）重启 Tomcat 服务器

```
web1 ~]# /usr/local/tomcat/bin/shutdown.sh
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)

```
proxy ~]# vim /etc/hosts
192.168.2.100      www.a.com  www.b.com
proxy ~]# firefox https://www.a.com:8443/
proxy ~]# firefox https://www.b.com:8443/
proxy ~]# firefox https://192.168.2.100:8443/
```

2.3.5 配置 Tomcat 日志

catalina.2019-08-13.log\catalina.out 主日志,每天重新生成 1 个;其他为访问日志

1)为每个虚拟主机设置不同的日志文件

```
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>    #为 www.a.com 设置日志文件
<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>    #为 www.b.com 设置日志文件
```

2）重启 Tomcat 服务器

```
web1 ~]# /usr/local/tomcat/bin/shutdown.sh
web1 ~]# /usr/local/tomcat/bin/startup.sh
```

3）查看服务器日志文件

```
web1 ~]# ls /usr/local/tomcat/logs/
```

2.3.6 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）

```
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：

```
web1 ~]# yum -y install  java-1.8.0-openjdk  #安装 JDK
web1 ~]# yum -y install java-1.8.0-openjdk-headless  #安装 JDK
web1 ~]# tar -xzf  apache-tomcat-8.0.30.tar.gz
web1 ~]# mv apache-tomcat-8.0.30  /usr/local/tomcat
```

3）启动服务

```
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。

三 案例 3：使用 Varnish 加速 Web

CDN:Content Delivery Network,内容分发网络

Varnish 软件:代理+缓存功能

3.1 问题

通过配置 Varnish 缓存服务器，实现如下目标：

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

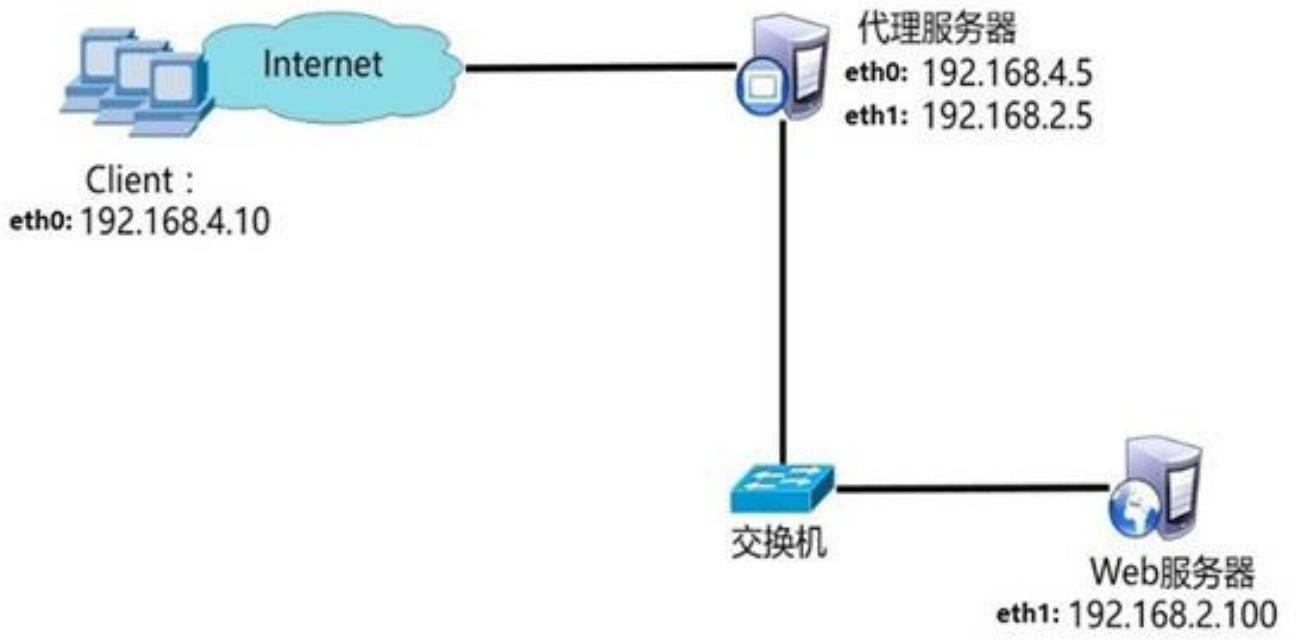
3.2 方案

通过源码编译安装 Varnish 缓存服务器

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

3.3.1 构建 Web 服务器

1）使用 yum 安装 web 软件包

```
web1 ~]# yum  -y  install  httpd
```

2）启用 httpd 服务（注意需要关闭 nginx，否则端口冲突）

```
web1 ~]# systemctl start httpd
```

httpd 服务默认通过 TCP 80 端口监听客户端请求：

```
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：

```
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）

```
proxy ~]# firefox http://192.168.2.100
```

3.3.2 部署 Varnish 缓存服务器(192.168.4.5)

1）编译安装软件

```
proxy ~]# yum -y install gcc readline-devel  #安装软件依赖包
proxy ~]# yum -y install ncurses-devel  #安装软件依赖包
proxy ~]# yum -y install pcre-devel  #安装软件依赖包
proxy ~]# yum -y install python-docutils  #安装软件依赖包
 proxy ~]# useradd -s /sbin/nologin varnish  #创建账户
proxy ~]# tar -xf varnish-5.2.1.tar.gz
proxy ~]# cd varnish-5.2.1
proxy varnish-5.2.1]# ./configure
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）

```
proxy varnish-5.2.1]# cp  ./etc/example.vcl
/usr/local/etc/default.vcl#拷贝并重命名 varnish config language
```

3）修改代理配置文件

```
proxy ~]# vim  /usr/local/etc/default.vcl
backend default {
    .host = "192.168.2.100";  #修改此处为 192.168.2.100
    .port = "80";  #修改此处为 80
 }
```

4）启动服务

```
 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

3.3.3 客户端测试

1）客户端开启浏览器访问

```
client ~]# curl http://192.168.4.5
```

3.3.4 其他操作

1）查看 varnish 日志

```
proxy ~]# varnishlog        #varnish 详细日志,进入后不退出
client ~]# curl 192.168.4.5  #客户端测试,再返回 proxy 查看日志信息
proxy ~]# varnishncsa       #varnish 简化日志
```

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，说明缓存中的数据过期了需要更新（默认也会自动更新，但非实时更新）。

```
proxy ~]# varnishadm

varnish> ban req.url ~ .*

#清空缓存数据，支持正则表达式
```