[Top](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY05/CASE/01/index.html" \l "page_top_case)

# NSD OPERATION DAY05

1. [案例1：安装部署Tomcat服务器](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY05/CASE/01/index.html" \l "case1)
2. [案例2：使用Tomcat部署虚拟主机](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY05/CASE/01/index.html" \l "case2)
3. [案例3：使用Varnish加速Web](http://tts.tmooc.cn/ttsPage/LINUX/NSDTN201904/OPERATION/DAY05/CASE/01/index.html" \l "case3)

## 1 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-1所示。

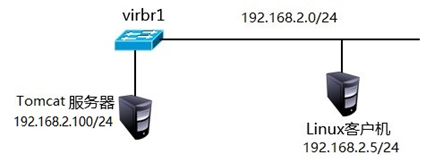


图-1

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

实现此案例需要按照如下步骤进行。

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

1. [root@web1 ~]# yum -y install java-1.8.0-openjdk                //安装JDK
2. [root@web1 ~]# yum -y install java-1.8.0-openjdk-headless        //安装JDK
3. [root@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）

1. [root@web1 ~]# tar -xf apache-tomcat-8.0.30.tar.gz
2. [root@web1 ~]# mv apache-tomcat-8.0.30 /usr/local/tomcat
3. [root@web1 ~]# ls /usr/local/tomcat
4. bin/                                            //主程序目录
5. lib/                                            //库文件目录
6. logs/                                         //日志目录
7. temp/                                         //临时目录
8. work/                                        //自动编译目录jsp代码转换servlet
9. conf/                                        //配置文件目录
10. webapps/                                        //页面目录

3）启动服务

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

4）服务器验证端口信息

1. [root@web1 ~]# ss -nutlp |grep java        //查看java监听的端口
2. tcp 0 0 :::8080 :::\* LISTEN 2778/java
3. tcp 0 0 :::8009 :::\* LISTEN 2778/java
4. 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）。

1. [root@web1 ~]# mv /dev/random /dev/random.bak
2. [root@web1 ~]# ln -s /dev/urandom /dev/random

另外，还可以使用方案二解决：

1. [root@web1 ~]# yum install rng-tools
2. [root@web1 ~]# systemctl start rngd
3. [root@web1 ~]# systemctl enable rngd

5）客户端浏览测试页面(proxy作为客户端)

1. [root@proxy ~]# 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

1. [root@web1 ~]# vim /usr/local/tomcat/webapps/ROOT/test.jsp
2. <html>
3. <body>
4. <center>
5. Now time is: <%=new java.util.Date()%>            //显示服务器当前时间
6. </center>
7. </body>
8. </html>

步骤三：验证测试

1）客户端浏览测试页面(proxy充当客户端角色)

1. [root@proxy ~]# firefox http://192.168.2.100:8080
2. [root@proxy ~]# firefox http://192.168.2.100:8080/test.jsp

## 2 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：

1. # cat /usr/local/tomcat/conf/server.xml
2. <Server>
3. <Service>
4. <Connector port=8080 />
5. <Connector port=8009 />
6. <Engine name="Catalina" defaultHost="localhost">
7. <Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">
8. </Host>
9. <Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">
10. </Host>
11. … …

### 2.3 步骤

实现此案例需要按照如下步骤进行。

步骤一：配置服务器虚拟主机

1）修改server.xml配置文件，创建虚拟主机

1. [root@web1 ~]# vim /usr/local/tomcat/conf/server.xml
2. … …
3. <Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">
4. </Host>
5. <Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">
6. </Host>

2）创建虚拟主机对应的页面根路径

1. [root@web1 ~]# mkdir -p /usr/local/tomcat/{a,b}/ROOT
2. [root@web1 ~]# echo "AAA" > /usr/local/tomcat/a/ROOT/index.html
3. [root@web1 ~]# echo "BBB" > /usr/local/tomcat/b/ROOT/index.html

3）重启Tomcat服务器

1. [root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh
2. [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！！！

1. [root@proxy ~]# vim /etc/hosts
2. … …
3. 192.168.2.100     www.a.com www.b.com
4. [root@proxy ~]# firefox http://www.a.com:8080/        //注意访问的端口为8080
5. [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

1. [root@web1 ~]# vim /usr/local/tomcat/conf/server.xml
2. … …
3. <Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">
4. </Host>
5. <Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">
6. <Context path="" docBase="base"/>
7. </Host>
8. … …
9. [root@web1 ~]# mkdir /usr/local/tomcat/b/base
10. [root@web1 ~]# echo "BASE" > /usr/local/tomcat/b/base/index.html
11. [root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh
12. [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)

1. [root@proxy ~]# firefox http://www.b.com:8080/     //结果为base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

1. [root@web1 ~]# vim /usr/local/tomcat/conf/server.xml
2. … …
3. <Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">
4. <Context path="/test" docBase="/var/www/html/" />
5. </Host>
6. <Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">
7. <Context path="" docBase="base" />
8. </Host>
9. … …
10. [root@web1 ~]# echo "Test" > /var/www/html/index.html
11. [root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh
12. [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)

1. [root@proxy ~]# firefox http://www.a.com:8080/test
2. //返回/var/www/html/index.html的内容
3. //注意，访问的端口为8080

步骤四：配置Tomcat支持SSL加密网站

1）创建加密用的私钥和证书文件

1. [root@web1 ~]# keytool -genkeypair -alias tomcat -keyalg RSA -keystore /usr/local/tomcat/keystore                //提示输入密码为:123456
2. //-genkeypair     生成密钥对
3. //-alias tomcat     密钥别名
4. //-keyalg RSA     定义密钥算法为RSA算法
5. //-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

1. [root@web1 ~]# vim /usr/local/tomcat/conf/server.xml
2. … …
3. <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
4. maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
5. keystoreFile="/usr/local/tomcat/keystore" keystorePass="123456" clientAuth="false" sslProtocol="TLS" />
6. //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

1. [root@web1 ~]# /usr/local/tomcat/bin/shutdown.sh
2. [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)

1. [root@proxy ~]# vim /etc/hosts
2. … …
3. 192.168.2.100     www.a.com www.b.com
4. [root@proxy ~]# firefox https://www.a.com:8443/
5. [root@proxy ~]# firefox https://www.b.com:8443/
6. [root@proxy ~]# firefox https://192.168.2.100:8443/

步骤五：配置Tomcat日志

1)为每个虚拟主机设置不同的日志文件

1. [root@web1 ~]# vim /usr/local/tomcat/conf/server.xml
2. .. ..
3. <Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">
4. <Context path="/test" docBase="/var/www/html/" />
5. #从默认localhost虚拟主机中把Valve这段复制过来，适当修改下即可
6. <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
7. prefix="a\_access" suffix=".txt"
8. pattern="%h %l %u %t &quot;%r&quot; %s %b" />
9. </Host>
10. <Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">
11. <Context path="" docBase="base" />
12. <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
13. prefix="b\_access" suffix=".txt"
14. pattern="%h %l %u %t &quot;%r&quot; %s %b" />
15. </Host>
16. .. ..

2）重启Tomcat服务器

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

3）查看服务器日志文件

1. [root@web1 ~]# 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的章节）

1. [root@proxy ~]# vim /usr/local/nginx/conf/nginx.conf
2. http{
3. upstream toms {
4. server 192.168.2.100:8080;
5. server 192.168.2.200:8080;
6. }
7. server {
8. listen 80;
9. server\_name localhost;
10. location / {
11. proxy\_pass http://toms;
12. }
13. }
14. }

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：

1. [root@web1 ~]# yum -y install java-1.8.0-openjdk                //安装JDK
2. [root@web1 ~]# yum -y install java-1.8.0-openjdk-headless        //安装JDK
3. [root@web1 ~]# tar -xzf apache-tomcat-8.0.30.tar.gz
4. [root@web1 ~]# mv apache-tomcat-8.0.30 /usr/local/tomcat

3）启动服务

1. [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。

## 3 案例3：使用Varnish加速Web

### 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-2所示。

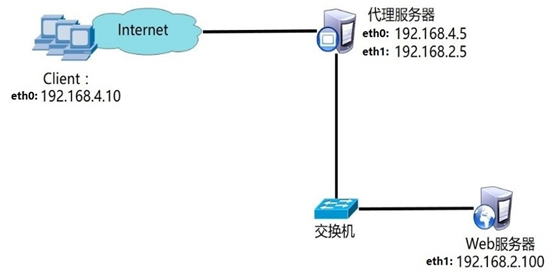


图-2

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

实现此案例需要按照如下步骤进行。

步骤一：构建Web服务器

1）使用yum安装web软件包

1. [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）

1. [root@web1 ~]# systemctl start httpd

httpd服务默认通过TCP 80端口监听客户端请求：

1. [root@web1 ~]# netstat -anptu | grep httpd
2. 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：

1. [root@web1 ~]# cat /var/www/html/index.html
2. 192.168.2.100

4）测试页面是否正常（代理服务器测试后台web）

1. [root@proxy ~]# firefox http://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

1. [root@proxy ~]# yum -y install gcc readline-devel    //安装软件依赖包
2. [root@proxy ~]# yum -y install ncurses-devel         //安装软件依赖包
3. [root@proxy ~]# yum -y install pcre-devel            //安装软件依赖包
4. [root@proxy ~]# yum -y install python-docutils         //安装软件依赖包
5. [root@proxy ~]# useradd -s /sbin/nologin varnish                //创建账户
6. [root@proxy ~]# tar -xf varnish-5.2.1.tar.gz
7. [root@proxy ~]# cd varnish-5.2.1
8. [root@proxy varnish-5.2.1]# ./configure
9. [root@proxy varnish-5.2.1]# make && make install

2）复制配置文件（注意相对路径与绝对路径）

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

3）修改代理配置文件

1. [root@proxy ~]# vim /usr/local/etc/default.vcl
2. backend default {
3. .host = "192.168.2.100";
4. .port = "80";
5. }

4）启动服务

1. [root@proxy ~]# varnishd -f /usr/local/etc/default.vcl
2. //varnishd命令的其他选项说明如下：
3. //varnishd -s malloc,128M        定义varnish使用内存作为缓存，空间为128M
4. //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

1. [root@client ~]# curl http://192.168.4.5

步骤四：其他操作

1）查看varnish日志

1. [root@proxy ~]# varnishlog                        //varnish日志
2. [root@proxy ~]# varnishncsa                    //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，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）。

1. [root@proxy ~]# varnishadm
2. varnish> ban req.url ~ .\*
3. //清空缓存数据，支持正则表达式