yum -y install php-pecl-zendopcache-7.0.5-2.el7.x86\_64.rpm 可以缓存php代码的优化器，让php更快

实验一：部署java环境与tomcat

1，安装openjdk

yum -y install java-1.8.0-openjdk 安装java环境

2，安装tomcat

cd ~/lnmp\_soft

tar -xf apache-tomcat-8.0.30.tar.gz 找到tomcat软件包并释放

mv apache-tomcat-8.0.30 /usr/local/tomcat 拷贝到/usr/local目录并修改名字

logs 存放日志

conf 存放配置文件

bin 存放执行程序，可以开启或关闭服务

webapps 存放网站页面

temp 存放临时文件

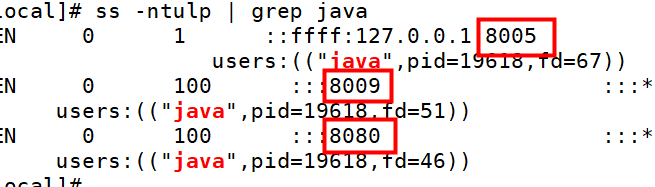
work 存放jsp代码

lib 存放类库

3，启动tomcat测试

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

ss -ntulp | grep java



看到3个端口是正常，如果某个端口没开，可以使用tomcat安装目录下的bin中的shutdown.sh关闭程序

yum -y install psmisc 安装killall命令软件包

killall java 如果tomcat关闭异常，可以杀死程序

如果tomcat无法正常启动很有可能是系统随机字符不够多造成

解决方法1：

mv /dev/random /dev/random.bak 将产生随机字符的random文件改名备份

ln -s /dev/urandom /dev/random 使用可以产生海量随机字符的文件替代

解决方法2：

yum install rng-tools 安装可以产生随机字符的软件，如果第1中方法可行，则无需方法2

systemctl start rngd

systemctl enable rngd

再次开启tomcat服务

ss -ntulp | grep java 再次查询

<http://192.168.2.100:8080/> 然后使用火狐浏览器访问

编写测试页面

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

<html>

<body>

<center>

Now time is: <%=new java.util.Date()%>

</center>

</body>

</html>

然后使用浏览器访问<http://192.168.2.100:8080/test.jsp>

实验二：使用tomcat搭建虚拟主机

httpd搭建虚拟主机

<virtualhost \*:80>

servername [www.a.com](http://www.a.com)

documentroot /var/www/html

</virtualhost>

nginx搭建虚拟主机

http {

server {

listen :80;

server\_name www.a.com;

location / {

root html;

index index.html index.htm;

}

}

}

1，tomcat搭建虚拟主机

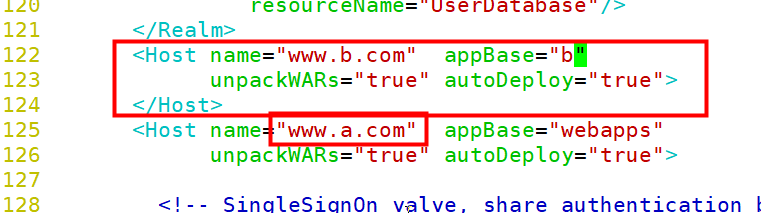
vim conf/server.xml 123gg 到123行

<Host XXXXXXXX > 在tomcat配置文件中，一个Host就是一个虚拟主机

</Host> 不要忘记结尾，且严格区分大小写

可以按照默认虚拟主机创建，修改新域名为[www.b.com](http://www.b.com)

appBase可以指定网站页面存放位置



[root@web1 tomcat]# bin/shutdown.sh 关闭服务，过程中如果报错就去检查配置文件

[root@web1 tomcat]# bin/startup.sh 再开启服务，开启后不报错也不代表服务100%没问题

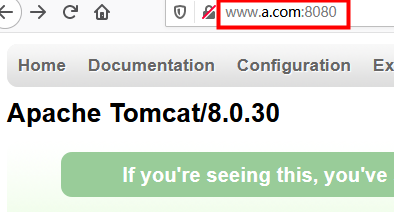
2，测试

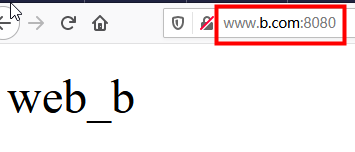
mkdir -p b/ROOT 再tomcat目录创建 b/ROOT目录

echo web\_b > b/ROOT/index.html 创建b网站测试页面

C:\Windows\System32\drivers\etc 修改真机的hosts文件

192.168.2.100 [www.a.com](http://www.a.com) [www.b.com](http://www.b.com) [www.c.com](http://www.c.com)





[root@web1 tomcat]# vim /etc/hosts 还可以修改web1主机hosts文件

192.168.2.100 www.a.com www.b.com www.c.com

[root@web1 tomcat]# curl www.b.com:8080

测试war包

yum -y install java-1.8.0-openjdk-devel 安装支持jar命令的软件包

cd /

jar -cf abc.war /var/log 使用jar命令制作war包

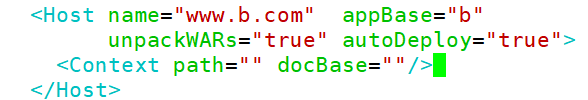
cp abc.war /usr/local/tomcat/b 拷贝到虚拟主机的目录

cd /usr/local/tomcat/b 到目录查看

ls 发现war包已经自动释放

实验三：测试tomcat页面的访问路径

测试1，在docBase后面定义为空的话，那么网站页面就会存在appBase的目录中

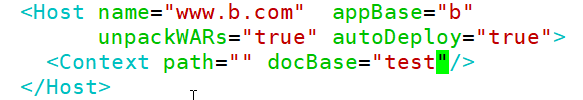


重启tomcat服务

[root@web1 tomcat]# echo "b/index.html" > b/index.html

curl www.b.com:8080

测试2，在docBase后面定义为test，那么b网站页面文件就存在了b/test中



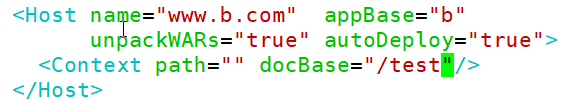
重启tomcat服务

[root@web1 tomcat]# mkdir b/test

echo "b/test/index.html" > b/test/index.html

再使用火狐或者curl [www.b.com:8080](http://www.b.com:8080) 测试

测试3，在docBase后面定义为/test，那么b网站页面文件就存在了/test中



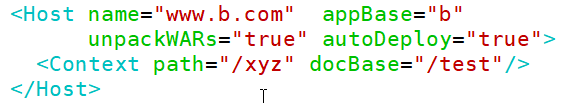
重启tomcat服务

[root@web1 tomcat]# mkdir /test

echo "/test/index.html" > /test/index.html

再使用火狐或者curl www.b.com:8080 测试

测试4，在path后面定义了/xyz之后，用户访问192.168.2.100:8080/xyz 则看到的是docBase定义的/test下的页面

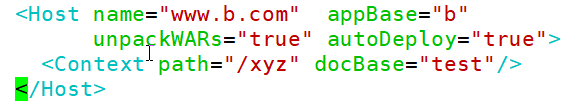


重启tomcat服务

curl [www.b.com:8080/xyz/](http://www.b.com:8080/xyz/)

curl [www.b.com:8080](http://www.b.com:8080) 看到的内容恢复为b/ROOT的页面

测试5，在上述实验基础上删除/test的/，那么用户访问192.168.2.100:8080/xyz 则看到的是b/test下的页面



重启tomcat服务

curl www.b.com:8080/xyz/

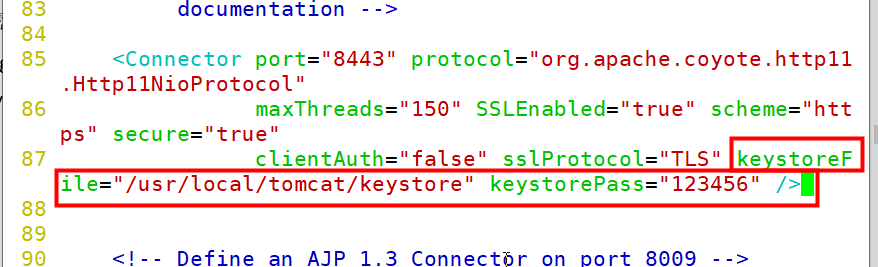
curl www.b.com:8080

实验四：在tomcat中配置ssl网站

1，创建密钥对

keytool -genkeypair -alias tomcat -keyalg RSA -keystore /usr/local/tomcat/keystore

打开tomcat配置文件修改为以下状态其中keystorefile是密钥对的路径，keystonepass是刚才为密钥对设置的密码



重启tomcat服务

ss -ntulp | grep java 检查服务，发现多了8443端口

<https://www.b.com:8443/> 使用火狐访问，看到风险提示则成功