operation day 05

================================================

**使用tomcat搭建网站服务**

**由于tomcat的运行需要依赖java，要先安装jdk工具包**

**1，配置java环境并拷贝tomcat程序到指定目录(该目录位置可以自定义)**

yum -y install java-1.8.0-openjdk

cd ~/lnmp\_soft

tar -xf apache-tomcat-8.0.30.tar.gz //在lnmp\_soft目录下释放tomcat软件包

cp -r apache-tomcat-8.0.30 /usr/local/tomcat //拷贝tomcat到指定目录

ls /usr/local/tomcat

**tomcat目录介绍**

bin 存放tomcat主要程序

logs 存放日志

conf 存放配置文件

webapps 网站页面的目录

work 自动编译的目录

temp 临时文件目录

lib 库文件目录

cd /usr/local/tomcat

bin/startup.sh //开启服务

bin/shutdown.sh //关闭服务

**2，运行前准备**

**由于tomcat运行需要random中有海量随机数，有时随机数不足会导致8005端口**

**无法启动**

**解决方案一：**

mv /dev/random /dev/random.bak //备份原有random文件

ln -s /dev/urandom /dev/random //使用urandom创建软连接替代random即可

[root@web1 tomcat]# bin/shutdown.sh //关闭tomcat

[root@web1 tomcat]# bin/startup.sh //重新开启tomcat服务

ss -ntulp | grep java //查询8005端口是否开启，该端口在服务关闭时需要，如果没有启动则服务不能使用，另外还有8080端口是提供网站服务的端口，8009端口可以做内部测试使用，也可以连接其他网站服务，目前的实验不使用，但必须看到这些端口都开启才说明tomcat服务正常

yum -y install psmisc //安装支持killall命令的软件包,如果tomcat异常可以使用

killall java

**解决方案二：**

yum -y install rng-tools //如果8005端口依然无法开启就安装该软件包

[root@web1 ~]# systemctl start rngd //开启服务

**3，测试tomcat服务**

http://192.168.2.100:8080/ //使用浏览器访问tomcat默认页，由于tomcat没有

使用默认的80端口而是8080，所以地址后面不要忘记写

**测试tomcat自定义静态页面与动态页面**

vim webapps/ROOT/test.jsp //编写基于java语言的动态页面

<html>

<body>

<center>

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

</center>

</body>

</html>

http://192.168.2.100:8080/test.jsp //使用浏览器访问可以看到当前时间

--------------------------------------------------------------

**利用tomcat创建虚拟主机**

**回顾：**

**httpd的创建方法是写多个virtualhost**

<virtualhost \*:80>

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

documentroot /var/www/html

</virtualhost>

**nginx的创建方法是写多个server**

http {

server {

listen :80;

server\_name [www.a.com](http://www.a.com);

location / {

root html;

index index.html index.htm;

}

}

}

**tomcat中配置虚拟主机是写多个host**

<Host name=www.a.com appBase=webapps >

</Host>

-----------------------------------------------------------------------------

**在tomcat中创建虚拟主机实际操作**

[root@web1 tomcat]# vim conf/server.xml //打开tomcat主配置文件，在122行添加

<Host name="www.b.com" appBase="web\_b"> //name后面写域名，appbase后面写网站的页面存放目录，另外在后面也就是默认的虚拟主机中的name由localhost修改为www.a.com

</Host> //虚拟主机配置的结尾

然后重启tomcat服务

mkdir -p web\_b/ROOT //在/usr/local/tomcat目录下创建b网站的测试目录

echo "web\_a" > webapps/ROOT/index.html //创建a网站测试页面

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

vim /etc/hosts //之后修改hosts文件添加

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

curl www.a.com:8080 //访问测试

curl www.b.com:8080

C:\Windows\System32\drivers\etc\hosts 修改windows系统的hosts文件添加

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) 如果域名被2.5占用,要将2.5修改为2.100即可

**再次修改b网站的虚拟主机配置成以下状态:**

<Host name="www.b.com" appBase="web\_b"

unpackWARs="true" autoDeploy="true">

重启tomcat服务（先关闭后开启）

yum -y install java-1.8.0-openjdk-devel

**autoDeploy和unpackWARs其功能解释如下：**

**autoDeploy //自动更新网站功能，有助于开发工程师**

**unpackWARs //自动解war包，通常开发工程师习惯将网站文件打成war包(类似tar包)上传到服务器，但是还需要解包很麻烦，但有了这个功能就可以自动解开**

**测试自动解war包功能：**

yum -y install java-1.8.0-openjdk-devel //安装创建war包的工具

cd /

[root@web1 /]# jar -cf xyz.war /var/log 创建war包，名字是xyz.war，内容是/var/log

cp xyz.war /usr/local/tomcat/web\_b/ //把war包扔到虚拟主机的网页目录下

ls /usr/local/tomcat/web\_b //查看b网站目录里面会自动解开xyz.war包，并产

生xyz目录，前提是tomcat服务正常开启中

---------------------------------------------------------------

**tomcat页面路径测试**

**默认情况下，tomcat网站页面的文件存放在webapps目录的ROOT下，如果需要自定义路径可以按以下几种方式进行配置**

**测试一:**

<Host name="www.b.com" appBase="web\_b"

unpackWARs="true" autoDeploy="true">

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

</Host>

[root@web1 tomcat]# echo "web\_b/index.html" > web\_b/index.html //创建测试页

重启tomcat服务

<http://www.b.com:8080> //测试，看到的是web\_b

中的页面，而不是web\_b/ROOT下

**测试二:**

当docBase="xyz"时，访问页面会在web\_b/xyz目录中

[root@web1 tomcat]# mkdir web\_b/xyz 创建测试目录

echo "web\_b/xyz/index.html" > web\_b/xyz/index.html 创建测试

页面

重启tomcat服务

www.b.com:8080 访问测试，看到的是web\_b/xyz中的页面

**测试三:**

<Host name="www.b.com" appBase="web\_b"

unpackWARs="true" autoDeploy="true">

<Context path="" docBase="/web\_b" /> //这里使用绝对路径

</Host>

[root@web1 tomcat]# mkdir /web\_b 创建测试目录

[root@web1 tomcat]# echo "/web\_b/index.html" > / web\_b/index.html 创建

测试页面

重启tomcat服务

www.b.com:8080 访问测试，看到的是/web\_b中的页面

**测试四:**

<Host name="www.b.com" appBase="web\_b"

unpackWARs="true" autoDeploy="true">

<Context path="/abc" docBase="/web\_b" />

</Host>

重启tomcat服务

www.b.com:8080/abc/ 访问测试，看到的是/web\_b中的页面

www.b.com:8080 访问测试，看到的是web\_b/ROOT中的页面

**测试五:**

<Host name="www.b.com" appBase="web\_b"

unpackWARs="true" autoDeploy="true">

<Context path="/abc" docBase="xyz" />

</Host>

重启tomcat服务

curl www.b.com:8080/abc/ 访问测试，看到的是web\_b/xyz/中的

页面

curl www.b.com:8080 访问测试，看到的web\_b/ROOT/中的页面

**练习: 下列需求如何实现？**

访问[www.b.com:8080](http://www.b.com:8080) 看到的是/usr/local/tomcat/abc/a 中的内容

访问[www.b.com:8080/test/](http://www.b.com:8080/test/) 看到的是/var/www/html 中的内容

提示 <Context path="/abc" docBase="xyz" /> 这样的语句在虚拟主机

中可以出现多次,以匹配多种路径需求

答案:

<Host name="www.b.com" appBase="abc"

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

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

重启tomcat服务

------------------------------------------------------------------

**为b网站虚拟主机添加日志**

**tomcat只为默认的虚拟主机添加了日志功能，如果新建的虚拟主机需要该功能就**

**按下列方式配置：**

打开配置文件，在b网站的配置中添加日志功能<Valve ..... />(该配置从默认的

虚拟主机中复制即可，最后一页)，修改prefix日志名字，suffix日志后缀

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

prefix="www\_b\_com" suffix=".log"

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

</Host>

之后保存退出，重启tomcat服务，然后访问几次b网站

ls logs //查看tomcat目录下的日志目录，可以看到b网站的日志

**--------------------------------------------------------------**

**使用tomcat开启安全加密效果**

首先修改配置文件，找到85行左右, 首先删除原有内容的注释 <!-- -->, 然后添加下图内容中的keystoreFile(指定公钥私钥文件)和keystorePass(公钥私钥文件密码)

<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"/>

然后敲命令keytool -genkeypair -alias tomcat -keyalg RSA -keystore /usr/local/to

mcat/keystore //创建秘钥对文件，命令输入完毕后，先输入两次123456的密

码，然后回答问题，最后y确认，然后密码不用输入直接回车。-genkeypair是创

建密钥对，-alias是别名，-keyalg是用什么算法，RSA是一种非对称加密算法，

-keystore是密钥对文件的存储路径

重启tomcat 服务

ss -ntulp | grep java 可以看到8443端口

curl -k <https://www.b.com:8443> 测试最终效果

--------------------------------------------------------

部署后续课程环境：自动化运维

参考视频链接：https://pan.baidu.com/s/12wCiIUBK1pqGeYOUYFPchA 提取码：61U2

使用镜像是rhel-8.0-x86\_64-dvd.iso

所有主机为最小化配置，语言选择英文，安装过程中内存可以设置1G，安装好之

后可以设置512m

