**关于Tomcat 简介**

**httpd、nginx + php shell #nginx和php是最佳组合**

**tomcat + java**  **#tomcat 和java是最佳组合**

**Java简介**

**- java是一种跨平台的 面向对象的程序设计语言 java技术具有卓越的通用行 高效性 平台移植性和安全性**

**Java体系**

**-java SE (标准版)**

**-java EE(企业版)**

**Jdk 是sun针对java开发者推出的java语言的软件开发工具包**

**Jdk是整个java的核心**

**-**包括了java运行环境

-java工具（如编译 纠错 打包等工具）

-java基础类库

**Jre jre是jdk的子集**

-java虚拟机

-java核心类库和支持文件

-不包含开发工具（jdk）--编译工具 调试器和其他工具

**Java servlet**

-servlet是扩展web服务器功能的组件规范

-常见servlet容器

--IBM websphere

--Oracle weblogic

--Apache tomacat

--redhat Jboss

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

使用RPM安装JDK基础环境

使用源码安装部署Tomcat服务器

**部署jdk和tomcat**

步骤一：部署Tomcat服务器软件(192.168.2.100/24)

1)使用rpm安装jdk

[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 ~]# 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）

[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

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

4）服务器验证端口信息

root@web1 ~]# ss -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）。

[root@web1 ~]# mv /dev/random /dev/random.bak

[root@web1 ~]# ln -s /dev/urandom /dev/random

另外，还可以使用方案二解决：

root@web1 ~]# yum install rng-tools

[root@web1 ~]# systemctl start rngd

[root@web1 ~]# systemctl enable rngd

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

unpackWARS="true" autoDeploy="true" 代表可以自动解压压缩包 并完成自动部署

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

## **使用Tomcat部署虚拟主机**

* 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

## **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>

**配置服务器虚拟主机**

1. 修改server.xml配置文件，创建虚拟主机
2. [root@web1 ~]# vim /usr/local/tomcat/conf/server.xml
3. … …
4. <Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">
5. </Host>
6. <Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">
7. </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

**修改www.b.com网站的首页目录为base**

情况1：

1. <Host name="www.a.com" appBase="a"
2. unpackWARs="true" autoDeploy="true">
3. <Context path="" docBase="" />
4. </Host>

* 当访问www.a.com:8080时，显示的页面是a下的，此时a目录下需要创建测试页面 echo a\_test > a/index.html

情况2

1. <Host name="www.a.com" appBase="a"
2. unpackWARs="true" autoDeploy="true">
3. <Context path="" docBase="aa" />
4. </Host>

当访问www.a.com:8080时，显示的页面是a/aa下的，此时aa目录下需要创建测试页

echo aa\_test >a/aa/index.html

情况3：

1. <Host name="www.a.com" appBase="a"
2. unpackWARs="true" autoDeploy="true">
3. <Context path="" docBase="/aa" />
4. </Host>

当访问www.a.com:8080时，显示的页面是（根路径）/aa下的，此时要在根下创建/aa目录，然后创建测试页面

echo "/aa\_test" > /aa/index.html

情况4：

1. <Host name="www.a.com" appBase="a"
2. unpackWARs="true" autoDeploy="true">
3. <Context path="/test" docBase="/aa" />
4. </Host>

当访问www.a.com:8080/test时，显示的页面是/aa下的， 当访问www.a.com:8080时,显示的页面是a/ROOT下的

情况5

1. <Host name="www.a.com" appBase="a"
2. unpackWARs="true" autoDeploy="true">
3. <Context path="/test" docBase="aa" />
4. </Host>

当访问www.a.com:8080时,显示的页面是a/ROOT下的

当访问www.a.com:8080/test时，显示的页面是a/aa下的

**开启加密网站（https）功能**

修改主配置文件在85行左右的位置修改 在<Connector内部添以下内容

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" />
7. //备注，默认这段Connector被注释掉了，打开注释，添加密钥信息即可

另外使用keytool命令创建秘钥对，过程可以使用--help看帮助

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

重启tomcat服务

https://www.b.com:8443/ 火狐测试

curl -k https://www.a.com:8443 linux测试

**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**>**

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**/**

**扩展实验(配置Tomcat集群)**

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. }