**Morning：**

1. Java简介

-Java：跨平台、面向对象的语言。通用、高效、平台移植、安全。

-JDK：Java语言的软件开发工具包。包括Java运行环境、Java工具（编译、排错、打包等）、Java基础的类库。Oracle（jdk）收费，推荐openjdk（开放、开源）

-JRE：Java运行环境。不包括编译、排错、打包等。

-Java Servlet容器：IBM websphere（银行）、Oracle weblogic、Apache tomcat、RedHat Jboss（铁道部）

1. Tomcat的安装与部署
2. **安装jdk（默认已安装）**

yum list | grep openjdk

yum -y install java-1.8.0-openjdk java-1.8.0-openjdk-headless

1. **安装tomcat**

[root@web1 ~]# tar -xzf apache-tomcat-8.0.30.tar.gz

[root@web1 ~]# mv apache-tomcat-8.0.30 /usr/local/tomcat

#由于tomcat是java写的，无需编译，只需有java解释器

[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

1. **启动Tomcat服务**

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

[root@web1 ~]# netstat -antulp | grep java

#查看java解释器是否启动，8080、8005、8009三个端口才正常。

Tips：由于8005启动时需要在/dev/random找随机码，可能存在启动缓慢的情况，此时可以执行以下操作：

mv /dev/random /dev/random.bak

ln -s /dev/urandom /dev/random

**4）测试网页**

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

[root@web1 ~]# firefox 192.168.2.100:8080/test.jsp

1. 案例：Tomcat部署虚拟主机

**-实验要求**

· 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，自动跳转到/var/www/html目录下页面

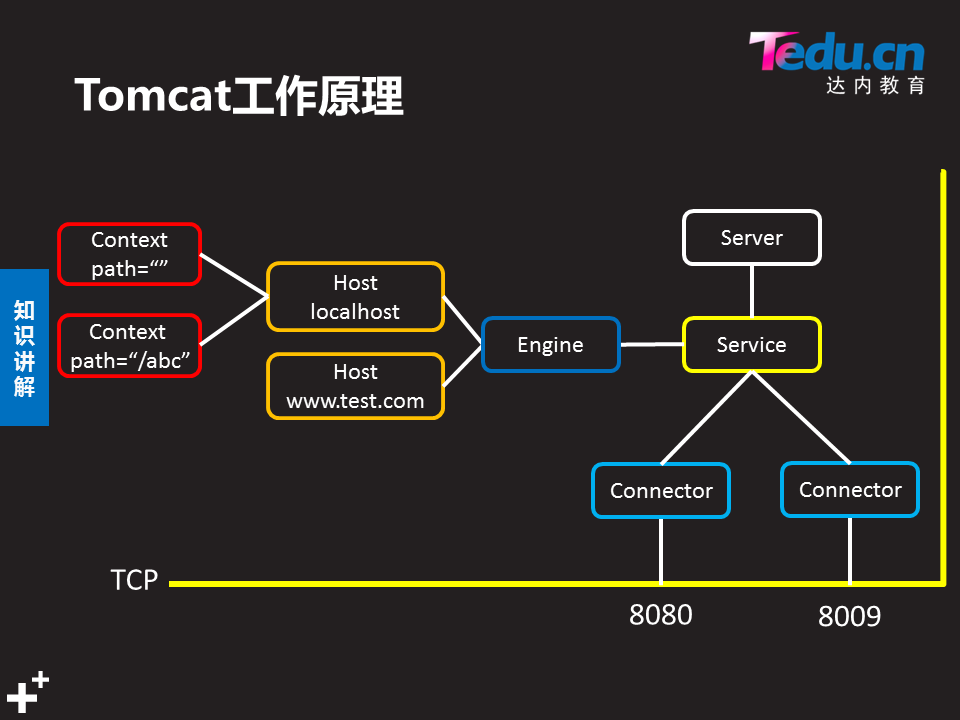
·访问页面时支持SSL加密通讯

·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

**-Tomcat工作原理**



**-主配置文件架构(严格区分大小写)**

cat /usr/local/tomcat/conf/server.xml

<Server>

<Service>

<Connector port=8080 />

#port：默认端口

<Connector port=8009 />

<Engine name="Catalina" defaultHost="localhost">

#defaultHost：用户使用ip访问时默认网站名

<Host name="www.a.com" appBase="a" unpackWARS="true" autoDeploy="true">

#name：域名，appBase：网页根目录，unpackWARS：支持WAR文件压缩

</Host>

<Host name="www.b.com" appBase="b" unpackWARS="true" autoDeploy="true">

</Host>

cat /usr/local/tomcat/conf/web.xml

<welcome-file>定义默认首页</welcome-file>

**步骤1：配置服务器虚拟主机**

1. 修改主配置文件，创建虚拟主机

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

<Host name="www.a.com" appBase="a" unpack......"true">

</Host>

<Host name="www.b.com" appBase="b"unpack......"true">

</Host>

1. 创建根路径及网页文件

[root@web1 ~]# mkdir -p /usr/local/tomcat/{a,b}/ROOT

[root@web1 ~]# echo "AAA" > /usr/local/tomcat/a/ROOT/index.html

[root@web1 ~]# echo "BBB" > /usr/local/tomcat/b/ROOT/index.html

1. 重启Tomcat服务器

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

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

4）配置/etc/hosts文件，测试页面

[root@proxy ~]# vim /etc/hosts

[root@proxy ~]# firefox http://www.a.com:8080/

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

**步骤2：修改www.b.com网站的首页目录为base**

1）修改默认网站首页路径（默认为ROOT）

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

<Host name="www.b.com" appBase="b" unpack..."true">

<Context path="" docBase="base" reloadable="true"/>

#docBase：默认网站首页的绝对路径/相对路径（基于appBase）reloadable：网页有修改时是否重新加载

</Host>

1. 创建新的根路径及网页文件

[root@web1 ~]# mkdir /usr/local/tomcat/b/base

[root@web1 ~]# echo "BASE">/usr/../b/base/index.html

1. 重启tomcat服务

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

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

4）测试www.b.com页面

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

**步骤3：跳转**

1. 对www.a.com配置跳转页面

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

<Host name="www.a.com" appBase="a" unpack..."true">

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

#path：地址栏，docBase：跳转页面

</Host>

1. 创建页面并重启

[root@web1 ~]# echo "Test" > /var/www/html/index.html

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

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

3）测试www.a.com/test页面

[root@proxy ~]# firefox http://www.a.com:8080/test

**步骤4：SSL加密网站**

Tips：虚拟主机和端口不是绑定关系，只需对一个端口加密，即可将所有网站都加密

1. 创建私钥和证书文件

[root@web1 ~]# keytool -genkeypair -alias tomcat -keyalg RSA -keystore /usr/local/tomcat/keystore

#java环境自带密钥工具keytool

#-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 定义密钥文件存在:/usr/local/tomcat/keystore

#需要非交互输入密钥口令、个人信息等

1. 修改配置文件，创建支持加密连接的Connector

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

<Connector port="8443" protocol="org.apache...ioProtocol"

maxThreads="150" SSLEnabled="true" scheme="https" secure="true"

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

#去掉注释<!-- -->，添加keystoreFile、keystorePass

1. 重启Tomcat服务器
2. 测试https页面

[root@proxy ~]# firefox https://www.a.com:8443/

[root@proxy ~]# firefox https://www.b.com:8443/

[root@proxy ~]# firefox https://192.168.2.100:8443/

**步骤5：配置Tomcat日志**

1）为每个虚拟主机设置不同的日志文件

[root@web1 ~]# vim /usr/local/tomcat/conf/server.xml

<Host>

... ...

<Valve className="org.apache....Valve" directory="logs"

prefix=" a\_access" suffix=".txt"

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

#className调用日志类库，directory日志存放位置

#prefix文件名，suffix扩展名，pattern文件内容

</Host>

1. 重启Tomcat服务器，查看日志文件

[root@web1 ~]# ls /usr/local/tomcat/logs/

**步骤6：配置Tomcat集群（扩展实验）**

1. 在proxy主机配置Nginx调度器：

参考：阶段二 --> OPERATION --> DAY03 --> Page1

1. 在web2主机上配置Tomcat服务器

参考：本文档Page1

1. 启动Tomcat服务
2. 客户端验证

**Afternoon：**

1、Varnish缓存服务器

**-CDN（Content delivery network）原理**

在源站搭建Web集群以及部署网站，通过CDN缓存到各地，各地的用户通过Local DNS访问到对应的边缘节点CDN。首次访问时，代理服务器会把网页内容缓存到本地，从而起到加速的作用。

推荐：Junneyang博客园（服务器的发展历史）



**步骤1：在web1搭建httpd服务并建立测试页面**

**步骤2：部署Varnish缓存服务器（192.168.4.5）**

1. 安装软件依赖包，源码安装Varnish包

[root@proxy ~]# yum -y install gcc readline-devel ncurses-devel pcre-devel

#安装软件依赖包

[root@proxy ~]# cd /root/lnmp\_soft/

[root@proxy ~]# yum -y install python-docutils....noarch.rpm

#安装python-docutils

[root@proxy ~]# useradd -s /sbin/nologin varnish

#创建账户

[root@proxy ~]# tar -xf varnish-5.2.1.tar.gz

[root@proxy ~]# cd varnish-5.2.1

[root@proxy varnish-5.2.1]# ./configure

[root@proxy varnish-5.2.1]# make && make install

#源码安装Varnish包

1. 复制启动脚本及配置文件

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

1. 修改代理配置文件

[root@proxy ~]# vim /usr/local/etc/default.vcl

backend default {

.host = "192.168.2.100"; #后台服务器IP、端口

.port = "80";

}

1. 启动Varnish服务

[root@proxy ~]# varnishd -f /usr/local/etc/default.vcl

#varnishd –s malloc,128M 使用内存作为缓存，空间为128M

#varnishd –s file,/var/lib/varnish\_storage.bin,1G 使用文件作为缓存（存硬盘）

**步骤3：客户端测试&其他操作**

1）client客户机访问varnish缓存服务器

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

1. 查看varnish日志

[root@proxy ~]# varnishlog #varnish日志

[root@proxy ~]# varnishncsa #访问日志

#输入日志命令后为空，需在客户机执行访问再查看

1. 更新缓存数据（默认也会自动更新，但非实时更新）

[root@proxy ~]# varnishadm

varnish> ban req.url ~ .\* #清空缓存数据，支持正则（文件名）

？？？Varnish如何重启配置