**tomcat服务器**

概述

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性、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）

java me（移动版）

jdk简介

jdk是sun争对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的运行环境

java工具（如编译、排错、打包等工具）

java基础类库

JRE，java运行环境，JRE是JDK的子集

JRE包括

java虚拟机（jvm）

java核心类库

不包括开发环境（jdk）--编译器、调试器和其他工具

java servlet

servlet是java扩展web服务器功能的组件规范

常见servlet容器

IBM websphere

oracle weblogic

apache tomcat

redhat jboss

安装tomcat

安装jdk1.8

yum -y install java-1.8.0-openjdk

yum -y install java-1.8.0-openjdk-headless

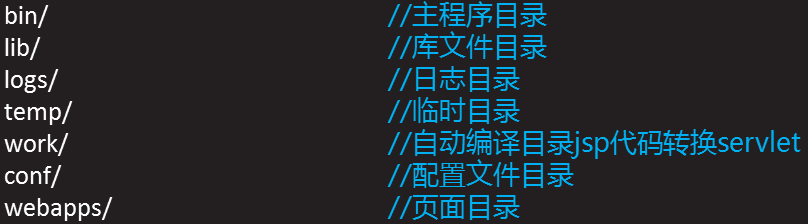
安装部署tomcat

安装tomcat

tar -xf apache-tomcat-8.0.30.tar.gz

mv apache-tomcat-8.0.30 /usr/local/tomcat

ls /usr/local/tomcat



启动tomcat

/usr/local/tomcat/bin/startup.sh

测试服务器

localhost:8080

查看端口，如果启动8005端口较慢，可使用以下命令（非必须命令）

mv /dev/random /dev/random.bak

ln -s /dev/urandom /dev/random

再重启服务

**tomcat应用案例**

tomcat虚拟机

主配置文件（conf/server.xml）解析

server.xml配置文件框架（严格区分大小写）

<Server>

<Service>

<connector port=8080>

<connector port=8009>

<Engine defaulthost=www.b.com>（优先级最低，默认的）

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

</Host>

<Host name=www.b.com>

</Host>

</Engine>

</Service>

</Server>

基于域名的虚拟主机

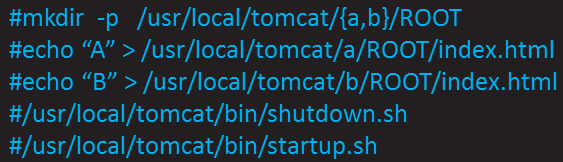
修改主配置文件，添加host主机

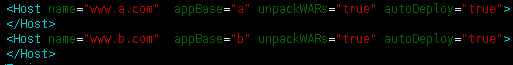
<Host name=”www.a.com” appBase=”a” unpackWARs=”true” autoDeploy=”true ” >

</Host>

<Host name=”www.b.com” appBase=”a” unpackWARs=”true” autoDeploy=”true ” >

</Host>





修改主配置文件，给host添加context

<Host name=”www.a.com” appBaase=”a” unpackWArs=”true” autoDeploy=”true ” >

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

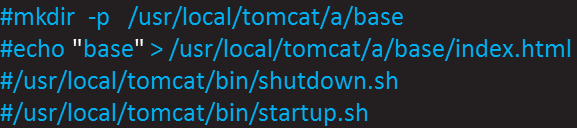
</Host>

<Host name=”www.b.com” appBaase=”a” unpackWArs=”true” autoDeploy=”true ” >

</Host>

appBase定义基础目录，基础目录下可以有很多项目，默认项目webapps

docBase定义首页路径，默认为ROOT





修改主配置文件，给host添加context

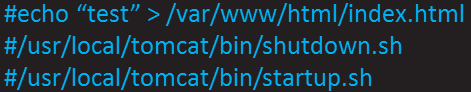
<Host name=”www.a.com” appBaase=”a” unpackWArs=”true” autoDeploy=”true ” >

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

</Host>

<Host name=”www.b.com” appBaase=”a” unpackWArs=”true” autoDeploy=”true ” >

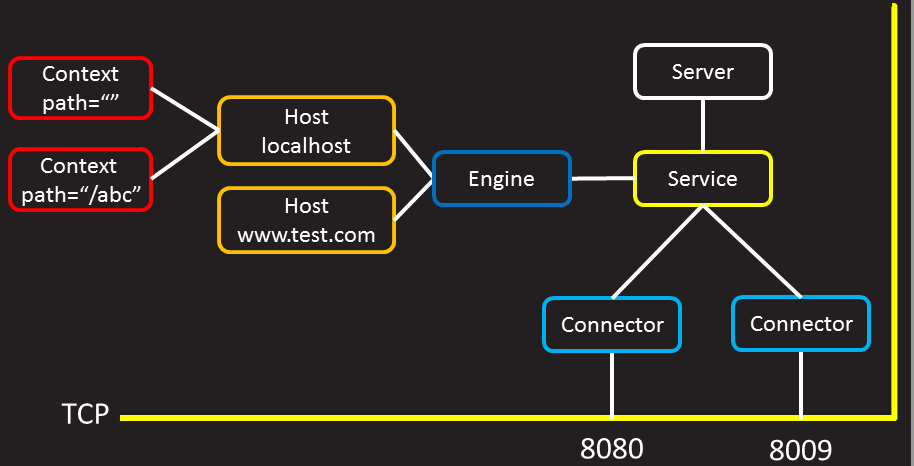
</Host>



path指指定用户访问的URL，docBase指定页面存储的位置



tomcat工作原理



SSL加密站点

生产私钥证书文件（工具是安装上jdk以后就有的）

keytool -genkeypair -alias tomcat -keyalg RSA -keystore /data/tomcat/keystore

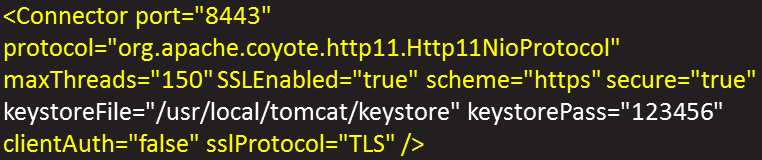
genkeypair 生成密钥对

-alias tomcat 密钥别名

-keyalg RSA 定义密钥算法为RSA算法

-keystore 定义密钥文件存储位置

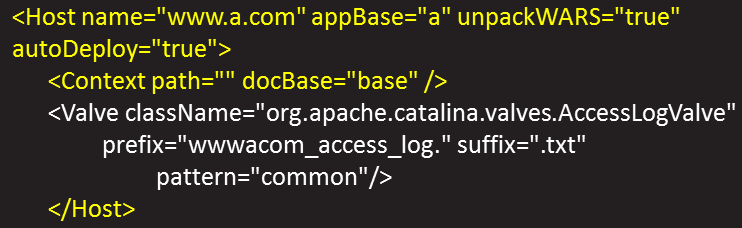
修改主配置文件

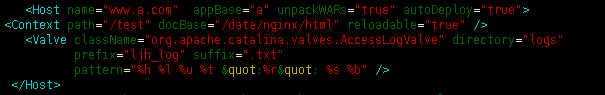




日志

虚拟主机创建独立日志文件





tomcat集群

nginx反向代理tomcat

修改Nginx配置文件

http{

upstream toms {

server 192.168.2.100:8080;

server 192.168.2.200:8080;

}

server {

listen 80;

server\_name localhost;

location / {

proxy\_pass http://toms;

}

}

}

**varnish代理服务器**

代理服务器概述

基本概念

proxy，代理人、经济人、中介

代替用户处理事务

能提供正常情况下不变或无法访问的资源

应用场景

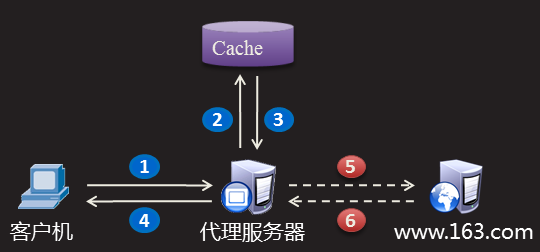
web访问加速（正/反向）

IP伪装、“翻墙”

代理服务器工作机制

cache池的复用

线路转移



varnish服务器（具有缓存功能，用内存缓存）

是一款高性能且开源的反向代理服务器

具有性能高、速度更快、管理更方便等诸多优点

加速后端web服务

可以将远程的web服务器缓存到本地

远程web对客户端是透明的

利用缓存机制提高网站的响应速度

部署varnish

编译安装

编译安装：

yum -y install gcc readline-devel

yum -y install ncurses-devel pcre-devel

yum -y install python-docutils-0.11-0.2.20130715svn7687.el7.noarch.rpm

useradd -s /sbin/nologin varnish

tar -xf varnish-5.2.1.tar.gz

cd varnish-5.2.1

./configure

make && make install

cp etc/example.vcl /usr/local/etc/default.vcl（复制启动脚本和配置文件）



varnish服务器

服务器软件

ls /usr/local/bin/

varnishadm varnishlog varnishstat varnishtop varnishhist varnishncsa varnishtest

ls /usr/local/sbin

varnishd

加速服务器

快速部署varnish

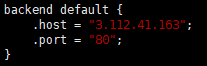
修改配置文件：vim /usr/local/etc/default.vcl

backend default {

.host=”192.168.2.100”;//后端服务器的IP地址

.port=”80”;

}



启动服务：varnishd -f /usr/local/etc/default.vcl

其他选项：varnishd -s malloc,128M//定义varnish使用内存作为缓存，空间为128M

varnishd -s file,/var/lib/varnish\_storage.bin,1G//定义varnish使用文件作为缓存

缓存管理

清楚缓存内容的命令格式：

varnishadm ban.url index.html

清除所有缓存

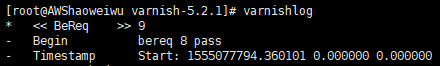
查看缓存清理列表

varnishadm ban.list

varnish日志

varnishlog [-w file] //共享内存的日志

varnishncsa [-w file] //类apache日志





varnish状态

varnishstat

client connections accepted:

表示客户端成功发送连接的总数量

client requests received:

客户端发送HTTP请求的总数

cache hits:

!重要：命中缓存的次数

cache misses

!重要：缓存非命中的个数

代理多台后端服务器

修改配置文件

vcl 4.0

import directors;//新增一行，导入一个directors（管理）

backend web {

.host = ”192.168.2.100”

.port = “80”;

}

backend web2 {

.host = ”192.168.2.200”;

.port = “80”;

}

-------接上面代码接着定义调度算法及制定流量转发--------

sub vcl\_init {#在init子函数中定义

new bar = directors.round\_robin();//random（随机） round\_robin dns

#定义服务器组，让新的bar等于之前定义的变量directors，后面接轮询算法

bar.add\_backend(web1);

bar.add\_backend(web2);

}

sub vcl\_recv {

set req.backend\_h