|  |
| --- |
| KCloudy |
| **SpringBoot In Linux** |
| [文档副标题] |

|  |
| --- |
| 田 长军  [日期] |

目录

[**1.** **部署前准备** 1](#_Toc8141440)

[**1.1** **安装Java** 1](#_Toc8141441)

[**1.2** **安装Maven** 1](#_Toc8141442)

[**1.2.1** **下载Maven安装包** 1](#_Toc8141443)

[**1.2.2** **解压Maven安装包** 2](#_Toc8141444)

[**1.2.3** **配置Maven路径** 2](#_Toc8141445)

[**1.2.4** **验证Maven是否配置成功** 2](#_Toc8141446)

[**1.2.5** **配置国内阿里云Maven镜像** 2](#_Toc8141447)

[**1.3** **安装Git** 3](#_Toc8141448)

[**1.4** **安装Docker** 3](#_Toc8141449)

[**1.4.1** **使用yum源安装** 3](#_Toc8141450)

[**1.4.2** **安装Docker Compose** 3](#_Toc8141451)

[**1.4.3** **安装Docker Machine** 3](#_Toc8141452)

[**1.4.4** **使用国内Docker镜像源** 4](#_Toc8141453)

[**1.4.5** **启动及测试** 4](#_Toc8141454)

[**1.5** **安装Jenkins** 5](#_Toc8141455)

[**1.5.1** **安装前准备** 5](#_Toc8141456)

[**1.5.2** **安装Jenkins** 6](#_Toc8141457)

[**1.5.3** **Jenkins相关配置信息** 6](#_Toc8141458)

[**1.5.4** **启动Jenkins及安装插件** 7](#_Toc8141459)

[**1.5.5** **配置Nginx的反向代理** 9](#_Toc8141460)

[**1.5.6** **配置Java JDK、Maven、Git** 11](#_Toc8141461)

[**2.** **SpringBoot** 11](#_Toc8141462)

[**2.1** **SpringBoot启动流程及自动配置** 11](#_Toc8141463)

[**2.1.1** **SpringBoot启动总体流程** 11](#_Toc8141464)

[**2.1.2** **SpringBoot应用程序入口** 12](#_Toc8141465)

[**2.1.3** **创建SpringApplication对象** 13](#_Toc8141466)

[**2.1.4** **启动SpringBoot应用** 13](#_Toc8141467)

[**2.1.5** **SpringBoot的自动配置** 17](#_Toc8141468)

[**2.2** **Spring MVC流程** 21](#_Toc8141469)

[**2.2.1** **流程图** 21](#_Toc8141470)

[**2.2.2** **其他** 21](#_Toc8141471)

[**2.3** **Web容器（默认使用Tomcat）** 21](#_Toc8141472)

[**2.3.1** **Jetty容器** 21](#_Toc8141473)

[**2.3.2** **undertow容器** 22](#_Toc8141474)

[**2.3.3** **其他** 23](#_Toc8141475)

[**2.4** **其他** 23](#_Toc8141476)

[**3.** **SpringBoot相关插件** 23](#_Toc8141477)

[**3.1** **Console支持彩色输出插件：ANSI Escape in Console** 23](#_Toc8141478)

[3.1.1 进入Eclipse Marketplace，搜索ANSI Escape in Console并插件 23](#_Toc8141479)

[3.1.2 设置application.properties 23](#_Toc8141480)

[3.1.3 配置日志（logback.xml） 23](#_Toc8141481)

[3.1.4 运行结果 26](#_Toc8141482)

[**3.2** **其他** 27](#_Toc8141483)

[**4.** **使用Docker部署SpringBoot Web项目** 27](#_Toc8141484)

[**4.1** **创建SpringBoot的Web项目** 27](#_Toc8141485)

[**4.2** **Web项目添加Docker支持** 28](#_Toc8141486)

[**4.2.1** **添加Docker插件** 28](#_Toc8141487)

[**4.2.2** **添加Dockerfile文件** 30](#_Toc8141488)

[**4.3** **将项目部署至Docker** 30](#_Toc8141489)

[**4.3.1** **检测环境** 30](#_Toc8141490)

[**4.3.2** **测试项目** 30](#_Toc8141491)

[**4.3.3** **生成Docker镜像** 31](#_Toc8141492)

[**4.3.4** **运行Docker镜像** 31](#_Toc8141493)

[**4.3.5** **测试Docker项目** 32](#_Toc8141494)

[**4.4** **相关问题** 32](#_Toc8141495)

[**4.4.1** **mvn docker:build fails with "{}->unix://localhost:80: Connection reset by peer** 32](#_Toc8141496)

[**4.4.2** **other** 32](#_Toc8141497)

[**4.5** **相关资料** 32](#_Toc8141498)

[**4.6** **其他** 32](#_Toc8141499)

[**5.** **其他** 32](#_Toc8141500)

**SpringBoot In Linux**

1. **部署前准备**
   1. **安装Java**

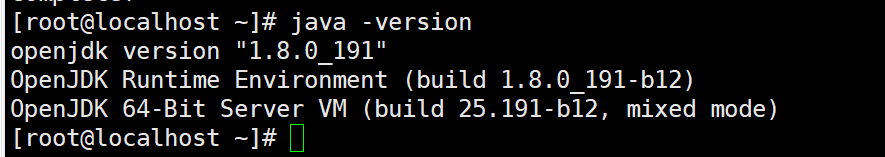
sudo yum -y install java-1.8.0-openjdk #jre

sudo yum -y install java-1.8.0-openjdk-devel #jdk

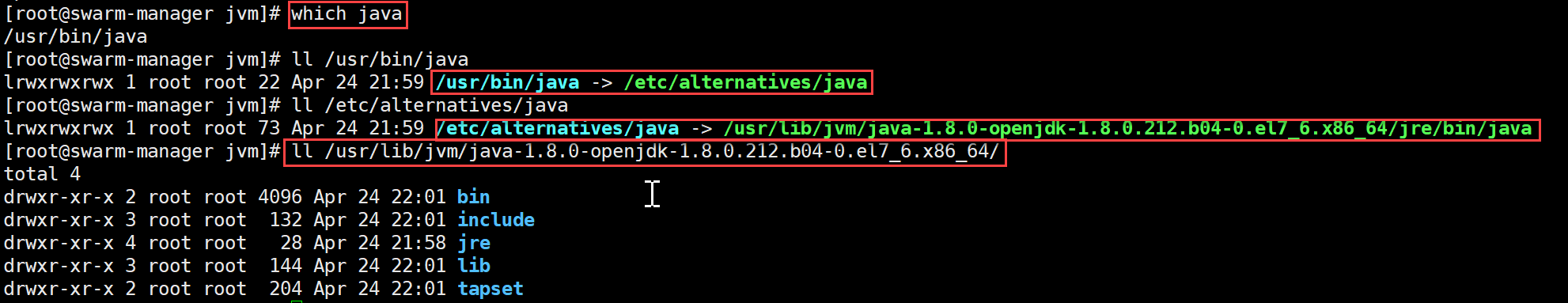
但对于centos有个问题，不可以安装成gcj(例如：gij(GNU libgcj) version 4.4.6 20110731(Red Hat 4.4.6-3)),导致[Jekins不工作](https://issues.jenkins-ci.org/browse/JENKINS-743),如果查看java版本类似上例，需要卸载，安装其他版本

java -version #查看Java版本

yum remove java #卸载异常版本



which java #查看java执行路径，备注：java实际的路径为-/usr/lib/jvm



设置java路径：vi /etc/profile 在文件最后添加下列配置

export JAVA\_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.212.b04-0.el7\_6.x86\_64

export JRE\_HOME=$JAVA\_HOME/jre

export CLASSPATH=$JAVA\_HOME/lib:$JRE\_HOME/lib:$CLASSPATH

export PATH=$JAVA\_HOME/bin:$JRE\_HOME/bin:$PATH

使配置生效： source /etc/profile

* 1. **安装Maven**
     1. **下载Maven安装包**

wget <https://mirrors.tuna.tsinghua.edu.cn/apache/maven/maven-3/3.6.2/binaries/apache-maven-3.6.2-bin.tar.gz>

* + 1. **解压Maven安装包**

tar -zxvf apache-maven-3.6.2-bin.tar.gz

mv apache-maven-3.6.2 /usr/local/maven3

* + 1. **配置Maven路径**

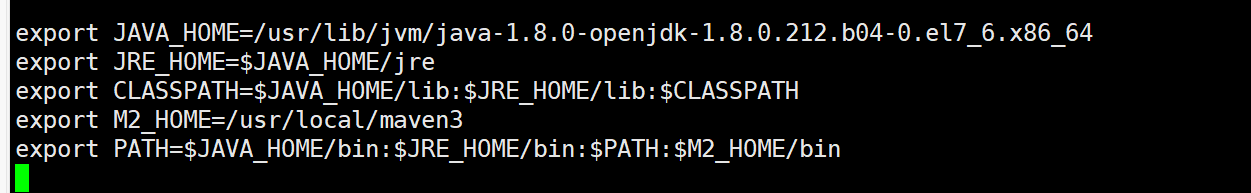
vi /etc/profile #最后添加以下内容

export M2\_HOME=/usr/local/maven3

export PATH=$PATH:$JAVA\_HOME/bin:$M2\_HOME/bin #保存退出后运行下面的命令使配置生效，或者重启服务器生效

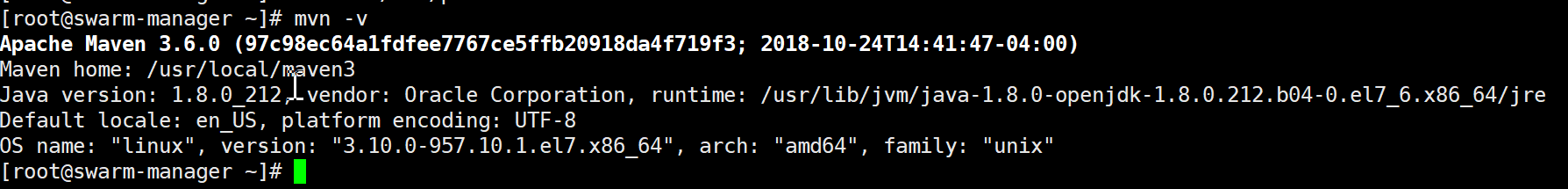
source /etc/profile

配置结果如下图：

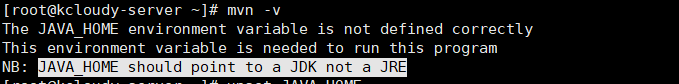


* + 1. **验证Maven是否配置成功**

mvn -v



* + 1. **使用mvn -v时，出现错误：JAVA\_HOME should point to a JDK not a JRE**



unset JAVA\_HOME #重置下JAVA\_HOME

* + 1. **配置国内阿里云Maven镜像**

vi /usr/local/maven3/conf/settings.xml

配置mirrors的子节点，添加如下mirror

<mirror>

<id>nexus-aliyun</id>

<mirrorOf>central</mirrorOf>

<name>Nexus aliyun</name>

<url>http://maven.aliyun.com/nexus/content/groups/public</url>

</mirror>



* 1. **安装Git**

sudo yum -y install git

git --version

* 1. **安装Docker**
     1. **使用yum源安装**

sudo wget -P /etc/yum.repos.d/ <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

sudo yum install -y docker-ce-18.09.3-3.el7 docker-ce-cli-18.09.3 containerd.io-18.09.3

docker --version

* + 1. **安装Docker Compose**

curl -L https://github.com/docker/compose/releases/download/1.23.2/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

chmod +x /usr/local/bin/docker-compose

docker-compose --version

* + 1. **安装Docker Machine**

sudo wget -P /etc/yum.repos.d/ http://download.virtualbox.org/virtualbox/rpm/rhel/virtualbox.repo

sudo yum install -y VirtualBox-5.1 #安装VirtualBox

curl -L https://github.com/docker/machine/releases/download/v0.16.1/docker-machine-`uname -s`-`uname -m` >/tmp/docker-machine &&

chmod +x /tmp/docker-machine &&

sudo cp /tmp/docker-machine /usr/local/bin/docker-machine

docker-machine -version

* + 1. **使用国内Docker镜像源**

vi /etc/docker/daemon.json #修改为以下内容

{

"exec-opts": ["native.cgroupdriver=systemd"],

"registry-mirrors" : [

"http://ovfftd6p.mirror.aliyuncs.com",

"http://registry.docker-cn.com",

"http://docker.mirrors.ustc.edu.cn",

"http://hub-mirror.c.163.com"

],

"insecure-registries" : [

"registry.docker-cn.com",

"docker.mirrors.ustc.edu.cn"

],

"debug" : true,

"experimental" : true

}

* + 1. **启动及测试**
       1. **启动Docker**

service docker start

chkconfig docker on

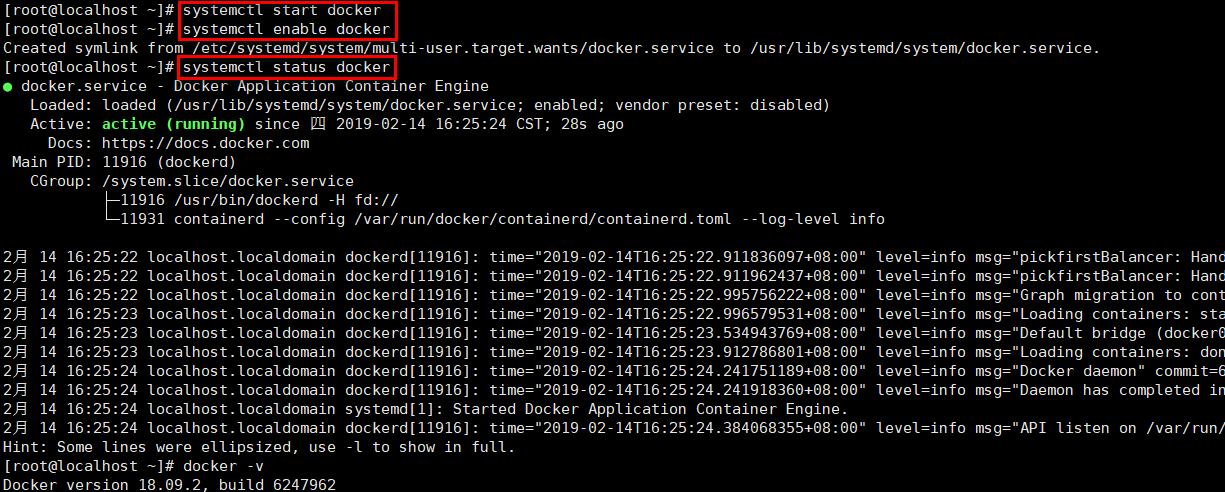
#LCTT 译注：此处采用了旧式的 sysv 语法，如采用CentOS 7中支持的新式 systemd 语法，如下：

systemctl start docker.service #开机自启 Docker CE

systemctl enable docker.service

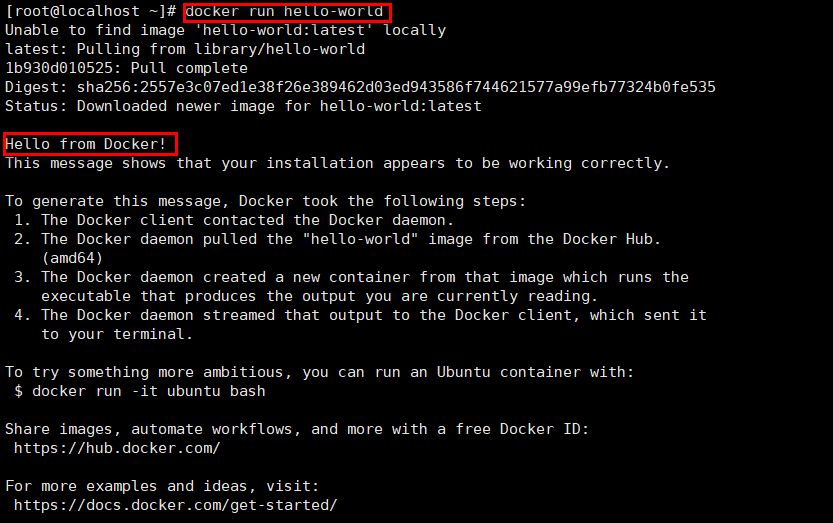
systemctl status docker #查看Docker状态

docker -v #查看docker版本



* + - 1. **测试Docker是否安装正确**

docker run hello-world



1. **Spring Consul**
   1. **安装Consul**
      1. **安装三个Consul服务**

192.168.56.12:8300作为Consul node1(leader)

192.168.56.12:8400作为Consul node2

192.168.56.12:8500作为Consul node3

* + 1. **使用Docker安装Consul**
* 搜索镜像

docker search consul

* 下载镜像

docker pull progrium/consul

* 安装镜像

#consul 服务端 node1

docker run -d -h node1 --name consul\_node1 --restart=always \

-v /data/consul:/consul/data \

-p 8300:8300 -p 8301:8301 -p 8301:8301/udp -p 8302:8302 -p 8302:8302/udp -p 8400:8400 -p 8500:8500 \

-e 'CONSUL\_LOCAL\_CONFIG={"skip\_leave\_on\_interrupt": true}' \

-e CONSUL\_BIND\_INTERFACE=eth0 \

progrium/consul -server -bootstrap

#获取 node1 的IP

JOIN\_IP="$(docker inspect -f '{{.NetworkSettings.IPAddress}}' consul\_node1)"

#consul 服务端 node2

docker run --name consul\_node2 -d --restart=always \

-e 'CONSUL\_LOCAL\_CONFIG={"skip\_leave\_on\_interrupt": true}' \

-e CONSUL\_BIND\_INTERFACE=eth0 \

progrium/consul -h node2 –server -join $JOIN\_IP

#consul 服务端 node3

docker run --name consul\_node3 -d --restart=always \

-e 'CONSUL\_LOCAL\_CONFIG={"skip\_leave\_on\_interrupt": true}' \

-e CONSUL\_BIND\_INTERFACE=eth0 \

progrium/consul –h node3 -server -join $JOIN\_IP

agent : 表示启动 Agent 进程。

-server：表示启动 Consul Server 模式。

-client：表示启动 Consul Cilent 模式。

-bootstrap：表示这个节点是 Server-Leader ，每个数据中心只能运行一台服务器。技术角度上讲 Leader 是通过 Raft 算法选举的，但是集群第一次启动时需要一个引导 Leader，在引导群集后，建议不要使用此标志。

-ui：表示启动 Web UI 管理器，默认开放端口 8500，所以上面使用 Docker 命令把 8500 端口对外开放。

-node：节点的名称，集群中必须是唯一的。

-client：表示 Consul 将绑定客户端接口的地址，0.0.0.0 表示所有地址都可以访问。

-join：表示加入到某一个集群中去。 如：-json=192.168.1.23

* 看集群成员

docker exec -t consul\_node1 consul members

docker exec -t consul\_node1 consul operator raft list-peers

* + 1. **其他**
  1. **其他**

1. **其他**