**SpringBoot + maven + Jenkins + docker 实现docker集群一键自动化部署**

整体可分为以下几个步骤:

1,创建springboot项目

2,搭建docker私服库

3,build镜像并上传到私服库

4,搭建docker swarm集群

5,搭建jenkins并构建部署

演示系统使用centos7,jdk1.8

# 1,创建Springboot项目:

## 1.1 eclipse生成项目

在eclipse上新建maven项目,在pom.xml中引入相应的包,如下所示:

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.spring</groupId>

<artifactId>mySpringboot</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>mySpringboot</name>

<url>http://maven.apache.org</url>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.3.0.RELEASE</version>

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<java.version>1.8</java.version>

<docker.image.prefix>ramer</docker.image.prefix>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<version>1.3.0.RELEASE</version>

<executions>

<execution>

<goals>

<goal>repackage</goal>

</goals>

</execution>

</executions>

</plugin>

<!-- 添加docker插件 -->

<plugin>

<groupId>com.spotify</groupId>

<artifactId>docker-maven-plugin</artifactId>

<version>0.4.13</version>

<executions>

<execution>

<id>build-image</id>

<phase>package</phase>

<goals>

<goal>build</goal>

</goals>

</execution>

</executions>

<configuration>

<imageName>192.168.230.10:5000/learn:mySwarm</imageName>

<forceTags>true</forceTags>

<baseImage>docker.io/java</baseImage>

<entryPoint>["java", "-jar", "/${project.build.finalName}.jar"]</entryPoint>

<resources>

<resource>

<targetPath>/</targetPath>

<directory>${project.build.directory}</directory>

<include>${project.build.finalName}.jar</include>

</resource>

</resources>

</configuration>

</plugin>

</plugins>

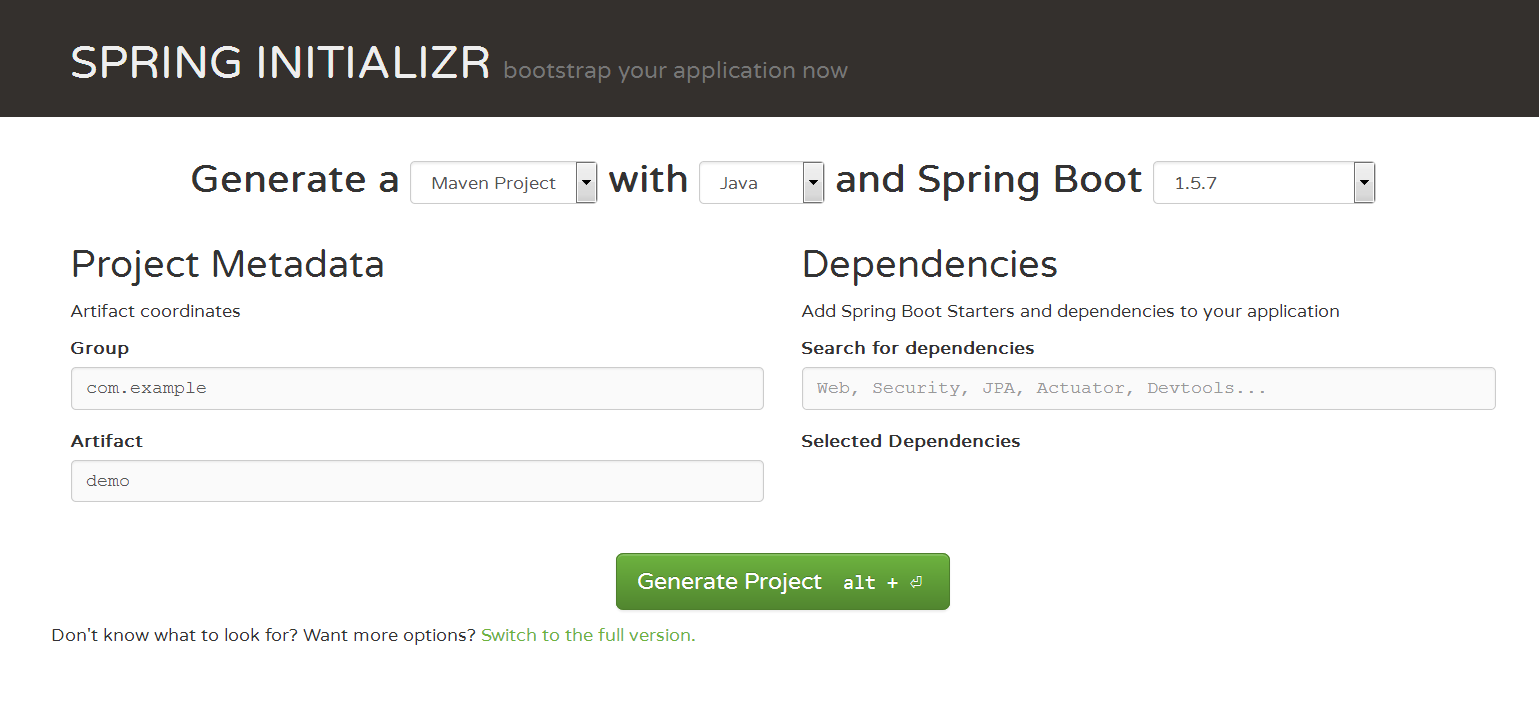
</build>

</project>

提示:docker-maven插件中可以修改imageName的值,其他配置可固定不变

## 1.2 在线生成项目

在浏览器中输入<https://start.spring.io/>,可以在线生成springboot项目结构,如下图所示:



生成的初始pom.xml中会包含springboot项目所需的包,然后再添加docker-maven插件



## 1.3 插件生成项目

可以使用eclipse集成的springboot插件生成springboot项目,由于本人eclipse中没有安装,所以这里不作介绍,有兴趣的小伙伴可以百度一下

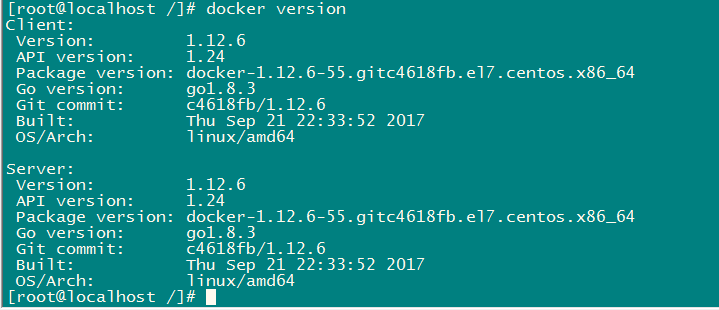
# 2, 搭建docker私服库

## 2.1 下载安装docker环境

直接使用在线安装,在终端输入命令: yum install docker

下载完成后启动docker服务:service docker start

成功启动后输入:docker version查看docker信息,如下图所示:



## 2.2 docker常用命令

docker run :执行镜像生成容器

docker ps –a:查看所有容器

docker pull:拉取镜像

docker push:上传镜像

docker start:运行容器

docker stop:停止容器

docker rm:删除容器

docker rmi:删除镜像

docker logs:查看容器日志

docker tag:修改镜像名

docker images:查看所有镜像

以上知识平时用的最多的docker命令,想了解更多的命令请百度

## 2.3 拉取私服库镜像

在终端输入: docker pull registry

把镜像下载完后修改镜像名:

docker tag docker.io/:registry 192.168.230.10:5000/learn:registry

说明:docker tag 后面分为两部分中间以空格隔开,空格的左边是要修改的原始镜像,右边是修改后的镜像,每个镜像名的组成都由registry + tag形成唯一标识,例如192.168.230.10:5000/learn这个是镜像存放的具体路径,registry这个是镜像标签名tag,必须要按照这种格式命名,否则无法上传到私服库

修改镜像名后输入: docker run -p 192.168.230.10:5000:5000 -v /opt/data/registry:/var/lib/registry -i -t -d docker.io/registry

生成并运行容器,这样一个docker私服库就已经搭建好了

我们把刚才修改镜像名后的镜像上传到docker私服库

docker push 192.168.230.10:5000/learn:registry

这里会报错,编辑配置文件添加以下内容:

vi /etc/docker/daemon.json



设置搭建好的docker私服库IP与端口

再次push刚才的镜像,成功上传

输入: curl <http://192.168.230.10:5000/v2/_catalog> 可以查询私服库信息,如下图所示



learn就是私服库存放镜像的文件夹,再查询这个文件夹下的镜像,输入:

curl <http://192.168.230.10:5000/v2/learn/tags/list>



通过docker pull拉取镜像,如:docker pull 192.168.230.10:5000/learn:registry

这里是拉取私服库中的镜像,根据你的镜像命名去私服库中查找,如果只提供了tag名则默认去docker官方的仓库中拉取,如我们之前拉取的服务库镜像,docker pull registry

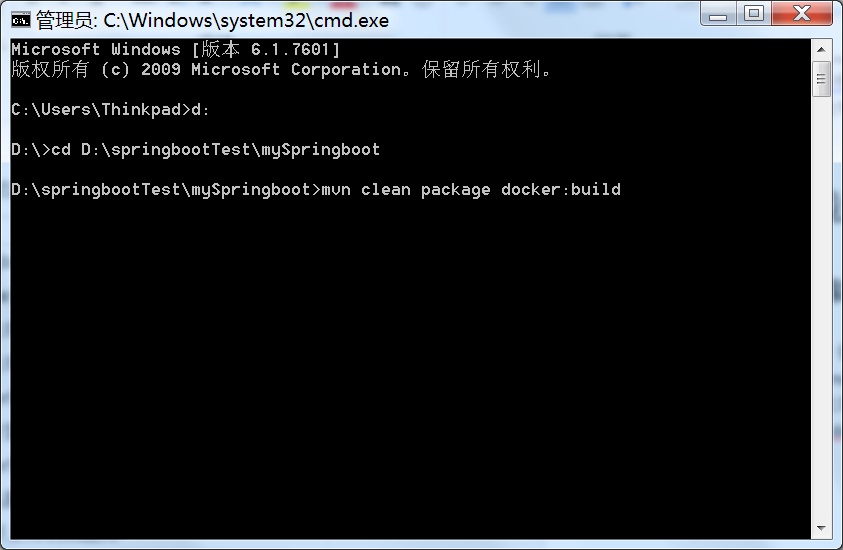
# 3,build镜像并上传到私服库

## 3.1 build镜像并上传到dock服务器

回到我们之前创建的springboot项目,找到本地项目根目录,如我的是D:\springbootTest\mySpringboot

打开cmd进入到此目录中输入以下命令:

mvn clean package docker:build,如下图所示



注意:这里会报错,原因是本地没有安装docker,无法执行docker命令

## 3.2 获取docker命令

通常有2种情况获取docker命令

1,本地安装docker

2,开启docker远程api

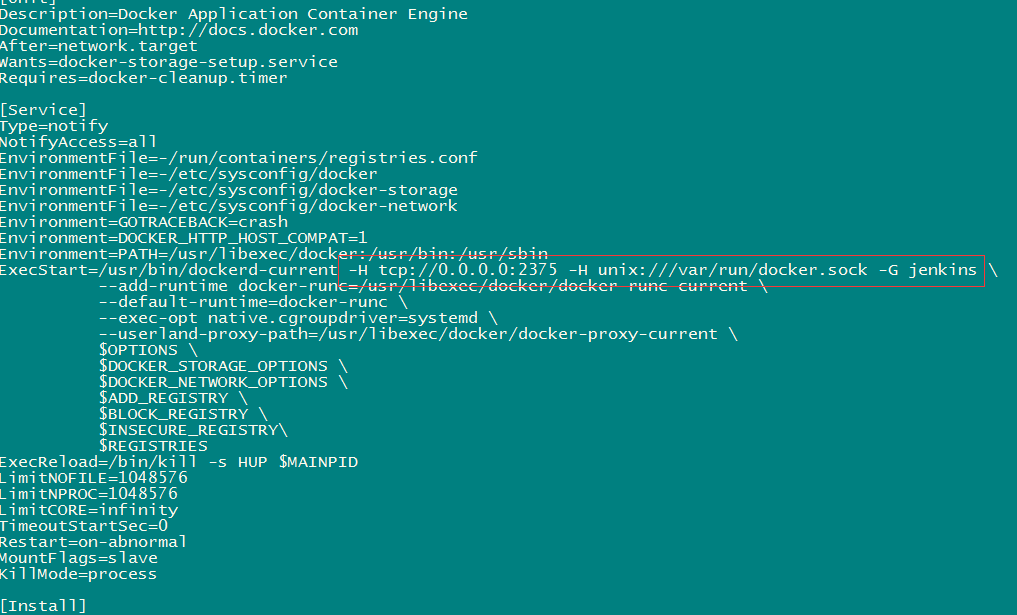
这里我们使用之前安装好的docker环境

1,先在本地环境变量中加入docker\_host变量,如下图所示



2,然后在docker服务中,编辑配置文件

vi /lib/systemd/system/docker.service,如下图所示在红框位置添加红框内的内容



然后重启docker

Systemctl daemon-reload

Systemctl restart docker.service

再在cmd中执行mvn clean package docker:build,还是会报错,原因是对应的2375端口没有被开放,进入到/sbin目录中执行 iptables -I INPUT -p tcp --dport 2375 -j ACCEPT然后再次打包,成功上传

然后在docker服务器输入docker images可以看到镜像已经上传到服务器,然后通过

docker push命令把镜像上传到私服库

# 4,搭建docker swarm集群

为了演示,我们用两台服务器就好了

192.168.230.10(主)

192.168.230.11(从)

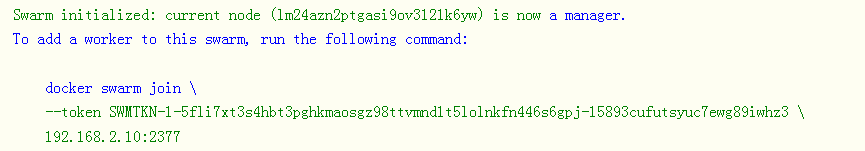
## 4.1 pull docker swarm镜像

在主机上执行:docker pull swarm,安装swarm插件

安装好插件后执行:

docker swarm init --advertise-addr 192.168.230.10 初始化一个集群的master服务器

返回以下信息:



记住这里的token 这个token是集群的唯一标识,只有根据这个token才能加入docker集群中

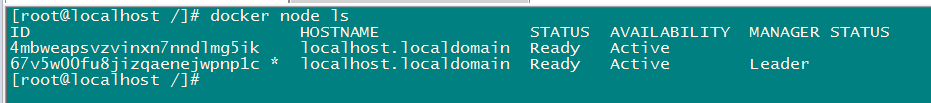
然后在从机上输入:

docker swarm join --token SWMTKN-1-5fli7xt3s4hbt3pghkmaosgz98ttvmnd1t5lolnkfn446s6gpj-15893cufutsyuc7ewg89iwhz3 192.168.2.10:2377

使服务器加入到这个docker集群中

然后在主机上输入

docker node ls 查看集群的信息,如下图



可以看到标识有leader的是集群的master服务器,其他的是从机

## 4.2 docker swarm常用命令

Docker swarm init:初始化一个集群

Docker swarm join:加入一个集群

Docker node ls:查看集群的信息

Docker service create:创建一个集群服务

Docker service ls:查看集群的服务

Docker service rm:删除一个集群服务

还有更多swarm命令,请百度

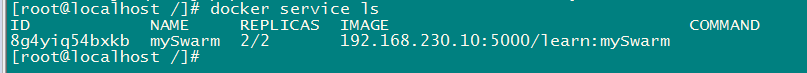
## 4.3 部署一个docker swarm集群项目

用之前从docker私服库中拉取的镜像,在主机上输入:

docker service create –replicas 2 –p 8000:8000 –name mySwarm 192.168.230.10:5000/learn:mySwarm

这样就创建了一个docker swarm集群服务,这里的replicas是部署几台服务器,swarm集群会先从本地找有没有对应的镜像,如果有则直接run该镜像,如果没有则从docker私服库上拉取,然后一个服务器一个服务器run镜像生成容器

然后输入docker service ls:可以看到目前服务的情况



这样,我们就简单的初步实现了一个docker swarm的集群

# 5,搭建jenkins并构建部署

## 5.1 拉取镜像

直接在终端机(jenkins可随意搭建在任意一台服务器)输入:

Docker pull Jenkins

下载完镜像后输入;

docker run -p 192.168.230.10:8080:8080 -e "DOCKER\_HOST=tcp://192.168.230.10:2375" --name myJenkins -u root -v /jenkins/:/var/jenkins\_home -d docker.io/Jenkins

这里的docker\_host跟前面说的原理一样,因为jenkins镜像run出来的容器本身是没有安装docker的,所以我们直接从docker服务器获取docker命令

这里的-u root是为了把当前运行jenkins的用户换做root

然后输入docker ps –a可以发现容器并没有跑起来,输入docker logs + 容器ID,查看容器日志提示没有权限,输入setenforce 0 然后再执行容器就成功run起来了

## 5.2 配置jenkins

Jenkins的初始化请参照百度,这里就不重复了

先安装要用到的插件,进入系统管理-管理插件-可选插件,在搜索框中输入要安装的插件

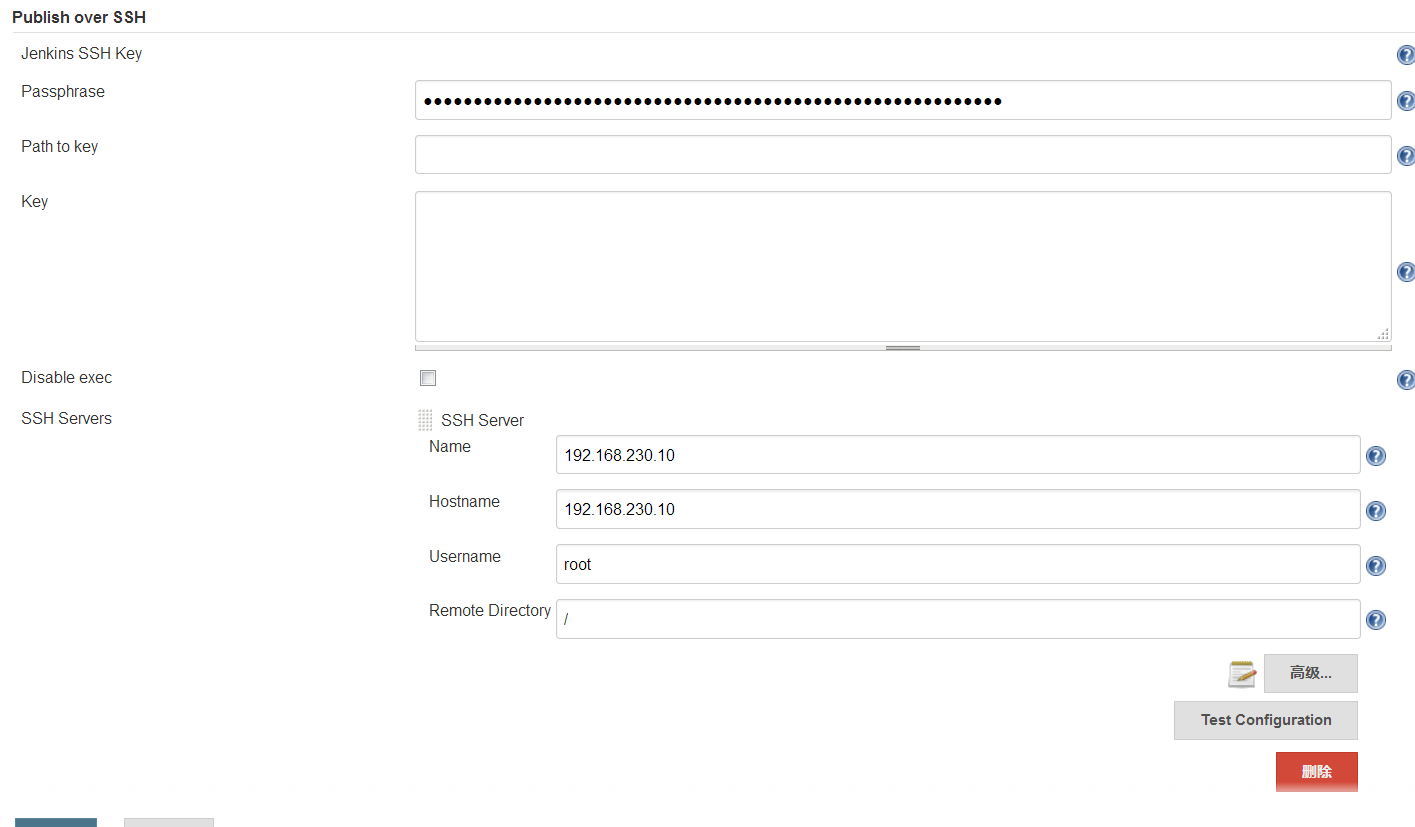
|  |  |
| --- | --- |
| [Maven Integration plugin](http://wiki.jenkins-ci.org/display/JENKINS/Maven+Project+Plugin) |  |

这个插件是为了可以新建一个maven项目

[Publish Over SSH](http://wiki.jenkins-ci.org/display/JENKINS/Publish+Over+SSH+Plugin)

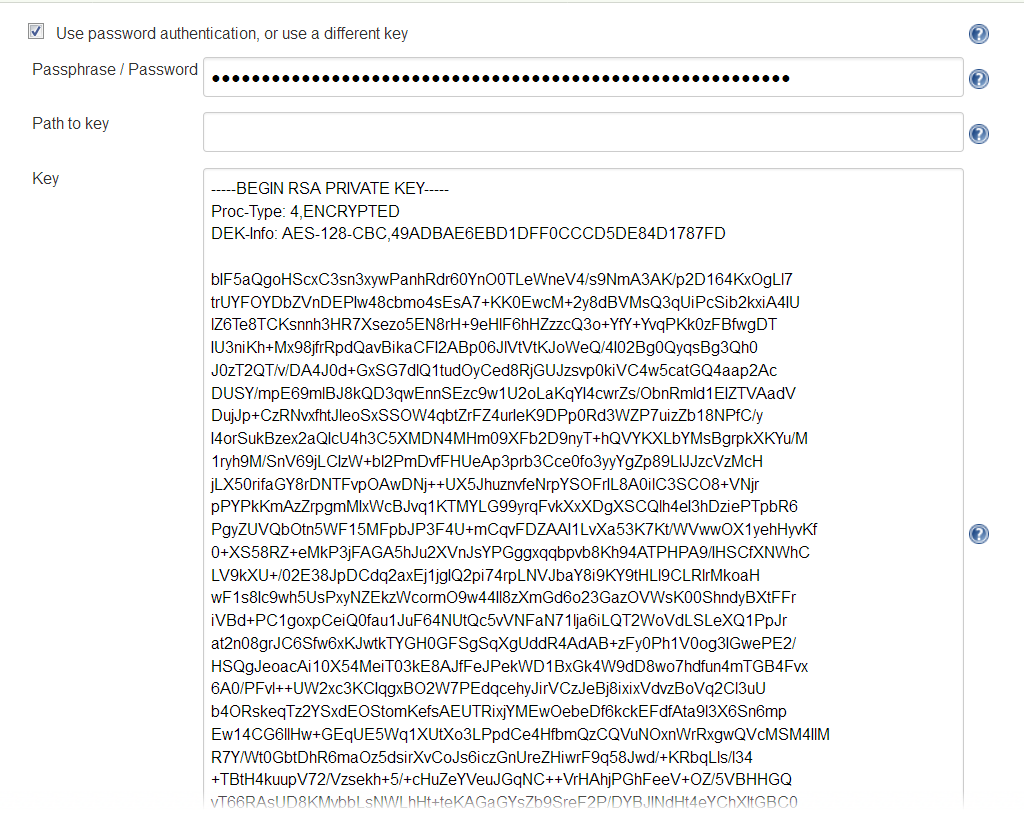
这个插件是为了jenkins构建完之后远程登录到docker swarm集群中的master机器统一部署

配置 SSH:

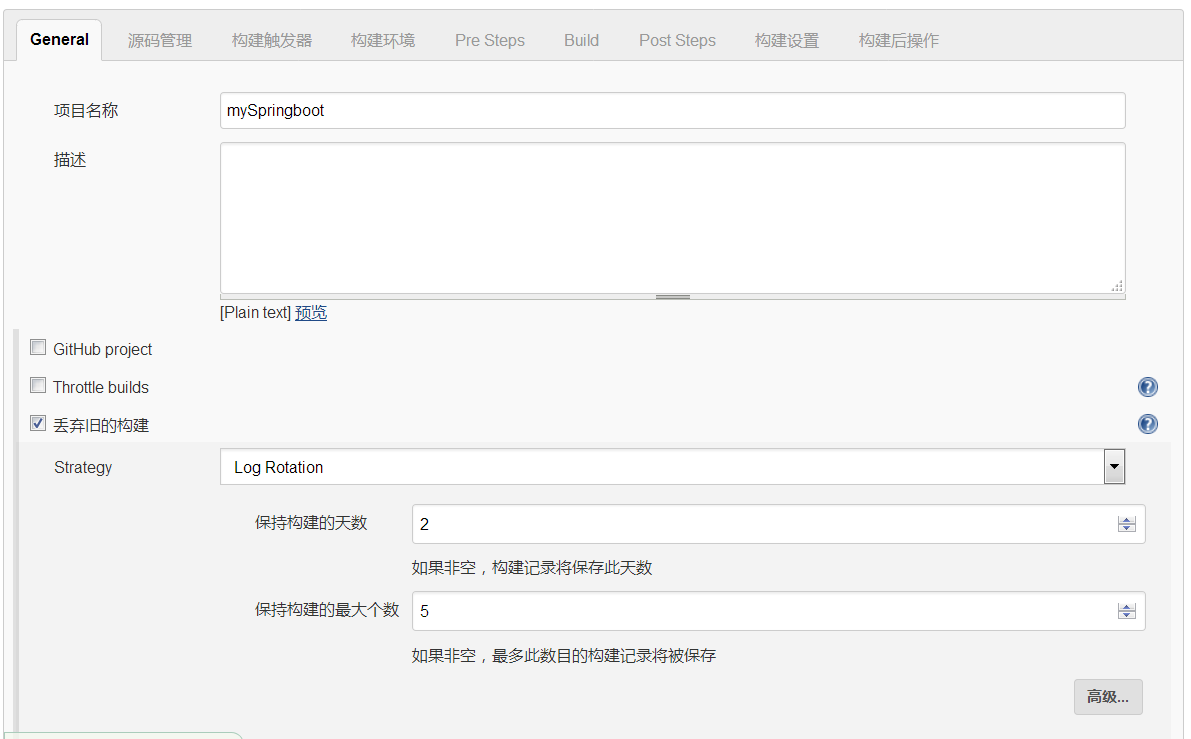


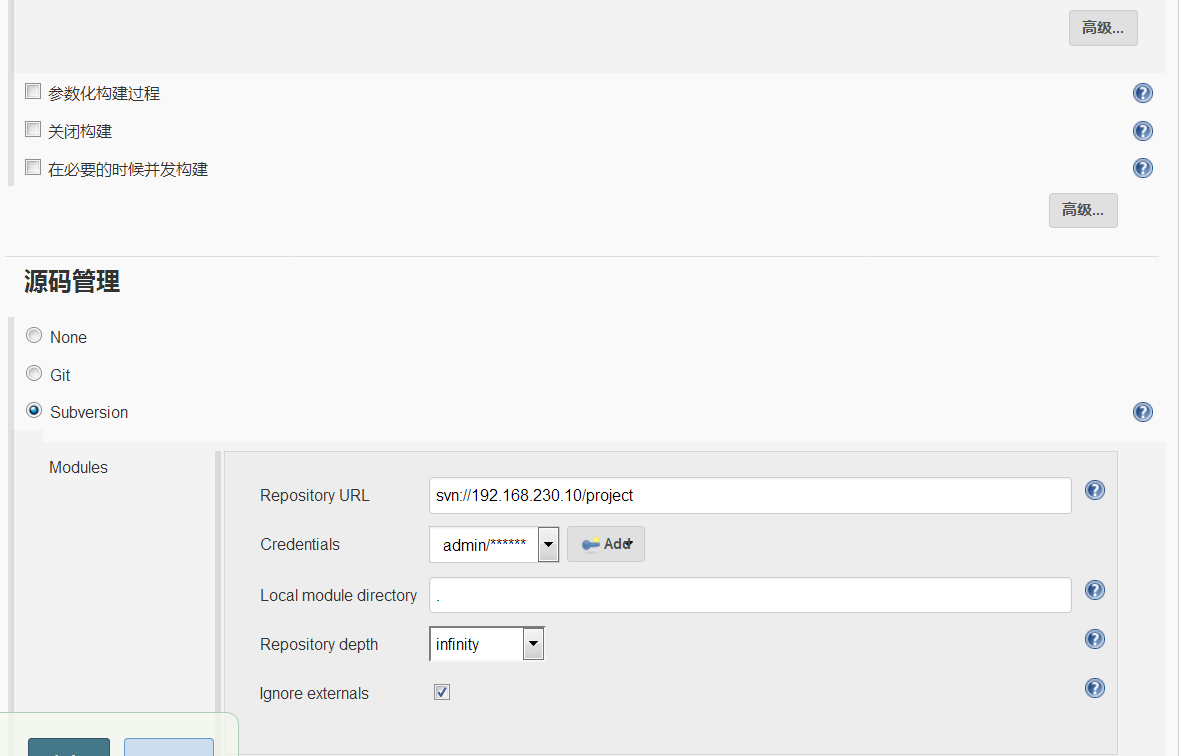
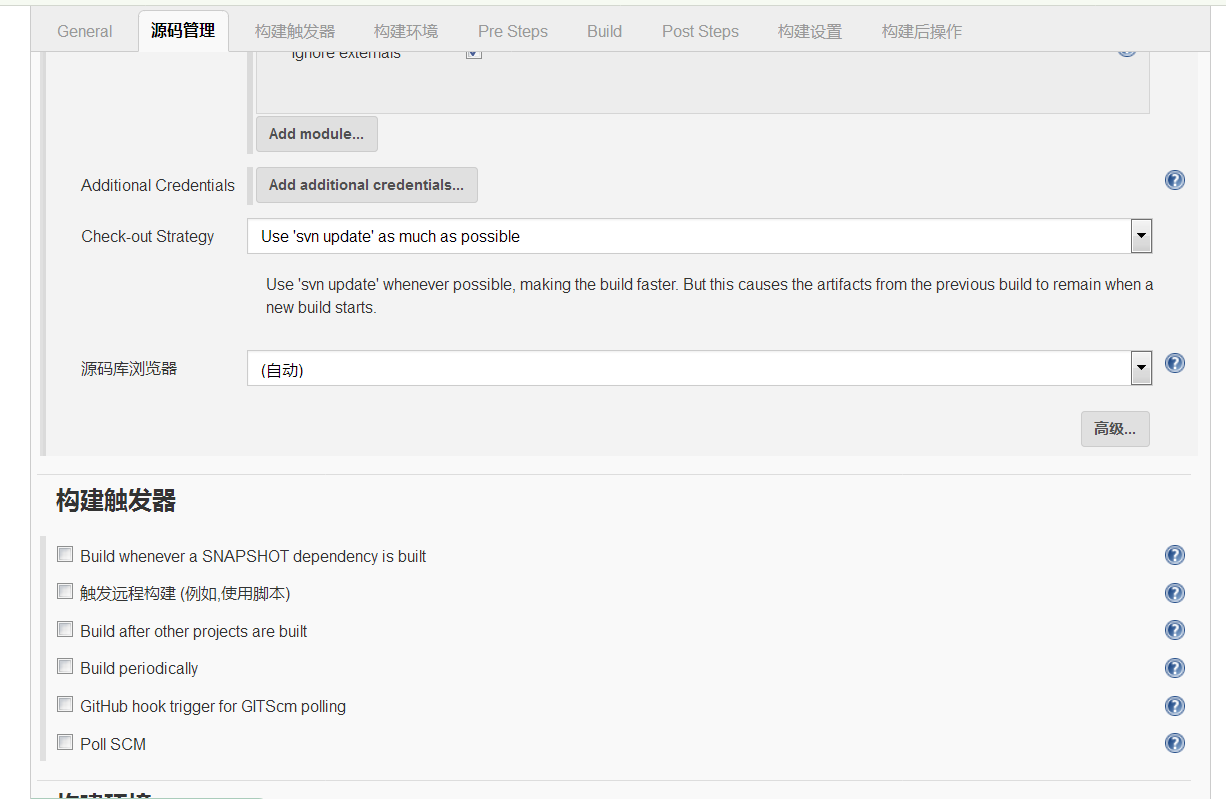
参照百度生成秘钥的方式生成公钥和私钥,

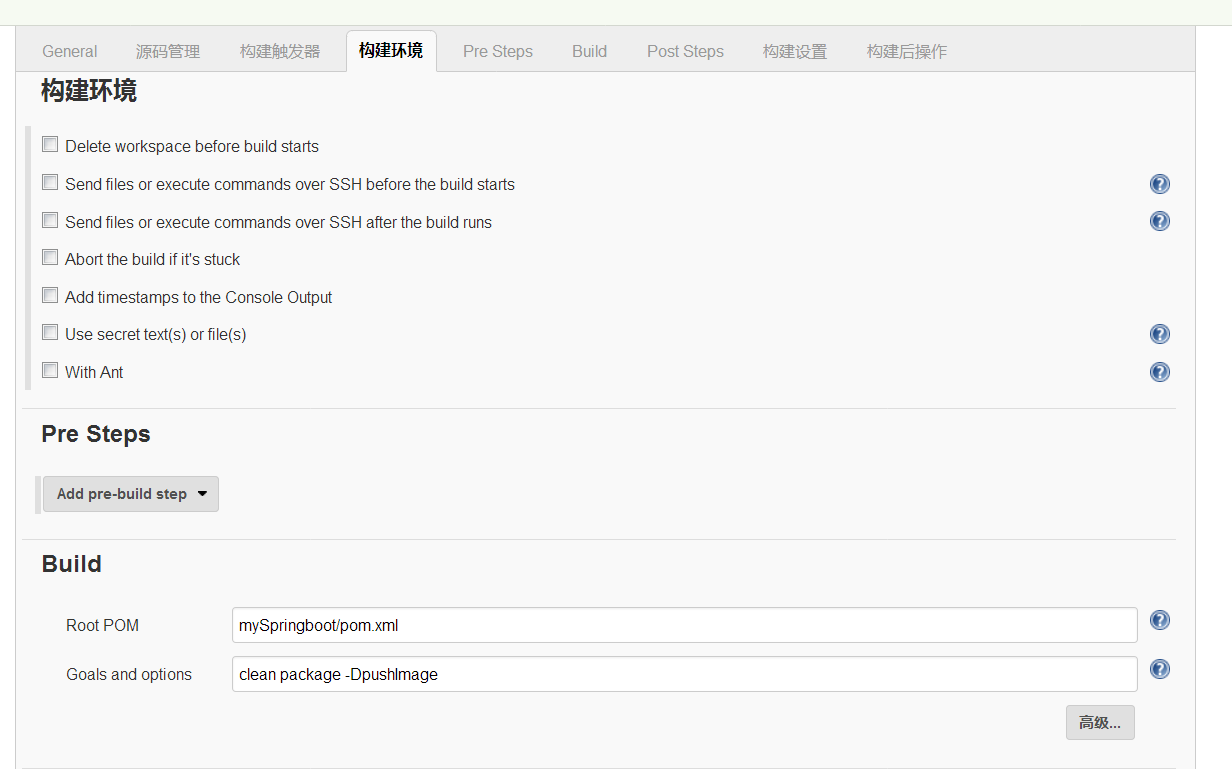
<http://blog.csdn.net/u010947098/article/details/61922969>

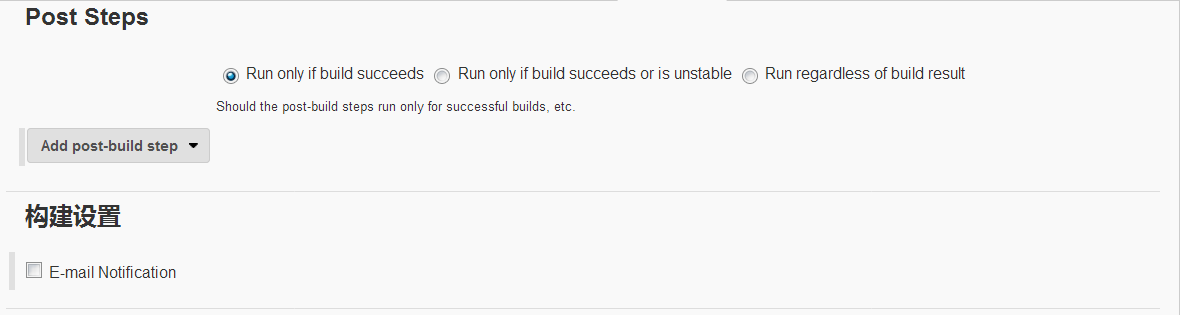


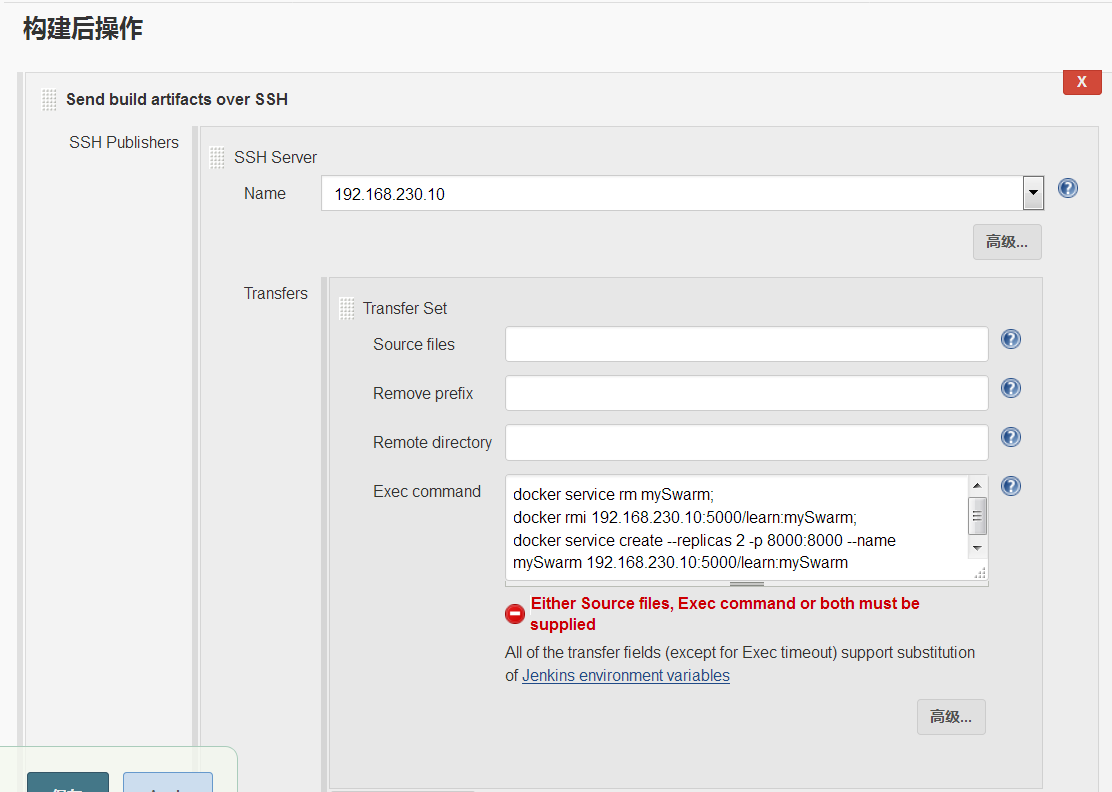
然后我们生成一个maven项目,配置如图所示:









Exec command中的命令:

docker service rm mySwarm;

先删除旧的服务

docker rmi 192.168.230.10:5000/learn:mySwarm;

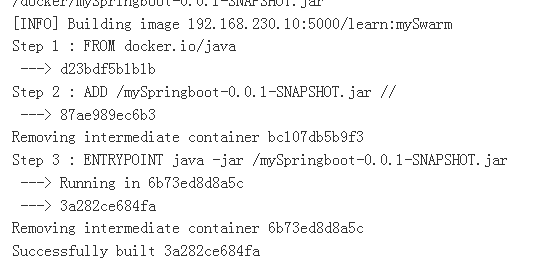
然后删除旧的镜像

docker service create --replicas 2 -p 8000:8000 --name mySwarm 192.168.230.10:5000/learn:mySwarm

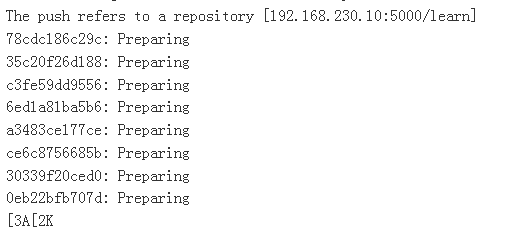
新建新的服务

配置完之后点击立即构建,第一次构建会下载很多包,需要一点时间,请耐心等待,最终如下图所示:

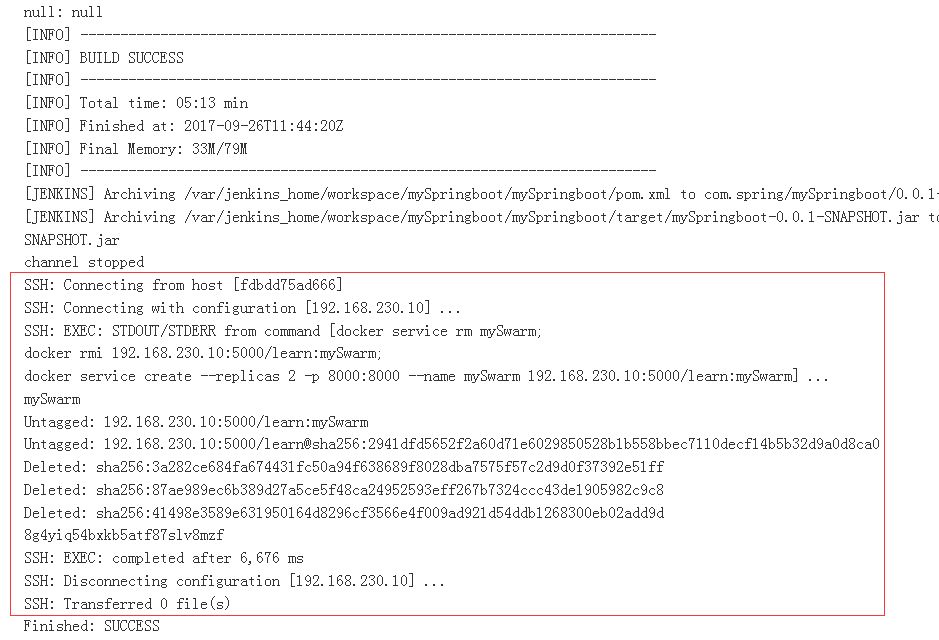
先成功build镜像



然后push镜像到docker私服库



构建完成后通过SSH登录到docker swarm集群的master统一部署代码



至此整个流程就结束了,还有很多细节需要完善….