<https://zhuanlan.zhihu.com/p/89312003?from_voters_page=true>

**手把手教你使用 Jenkins+Docker 实现持续集成**

[](https://www.zhihu.com/people/jvzy)

[**小知**](https://www.zhihu.com/people/jvzy)

我的微信公众号：Java知音

14 人赞同了该文章

作者：乐之终曲  
[https://blog.csdn.net/qq\_37143673/](https://link.zhihu.com/?target=https%3A//blog.csdn.net/qq_37143673/)

对于 Jenkins 我只能用两个字形容，难用。就不过多吐槽了，本篇是基于 docker 环境的使用。

**1.安装**

寻找需要的 Jenkins 镜像：

[https://hub.docker.com/r/jenkinsci/blueocean](https://link.zhihu.com/?target=https%3A//hub.docker.com/r/jenkinsci/blueocean)

docker pull jenkinsci/blueocean

我选择的镜像是 Jenkins-blueocean Jenkins 海洋版，为什么选这个？

* 踩坑：普通的 Jenkins 在部署的时候不少人都遇到过，插件下不下来，但是在海洋版没有这个问题（最主要原因）
* blueocean 的页面更加人性化，流程的监控上看着让人舒服的多，当然普通 Jenkins 也可以通过安装插件添加这个功能

**启动镜像**

docker run

--name jenkinsci-blueocean -u root

--rm -d -p 7005:8080 -p 50000:50000

-v /data/jenkins:/var/jenkins\_home

-v /var/run/docker.sock:/var/run/docker.sock jenkinsci/blueocean

* -u root：以 root 权限启动，防止出现权限问题
* -p 7005:8080：端口映射，服务器的 7005 端口映射容器的 8080 端口
* -p 50000:50000：Jenkins代理默认通过TCP端口50000与Jenkins主机通信
* -v /data/jenkins:/var/jenkins\_home：把容器内的 Jenkins 目录挂载到服务器的 /data/jenkins 目录以防容器没了，数据也没了
* -v /var/run/docker.sock:/var/run/docker.sock：保证容器内的 docker 与 服务器上 docker 的通讯

**附带下删除 jenkinsci/blueocean 容器**

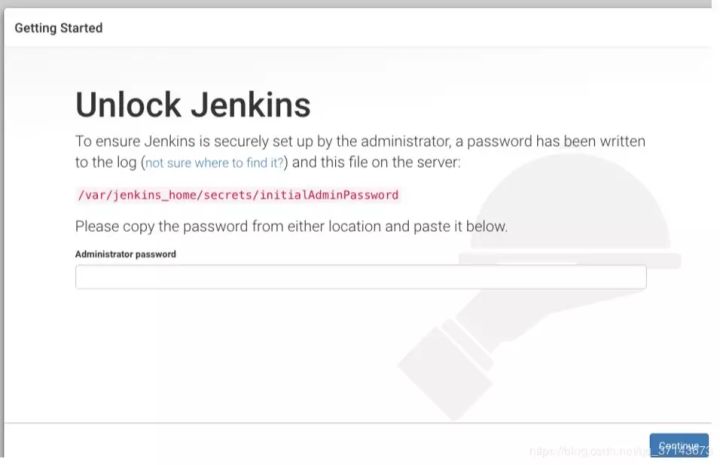
# 删除对应绑定网桥

docker network disconnect --force bridge jenkinsci-blueocean

# 删除 jenkinsci-blueocean 容器，xxxx 容器 ID

docker rm -f xxxx

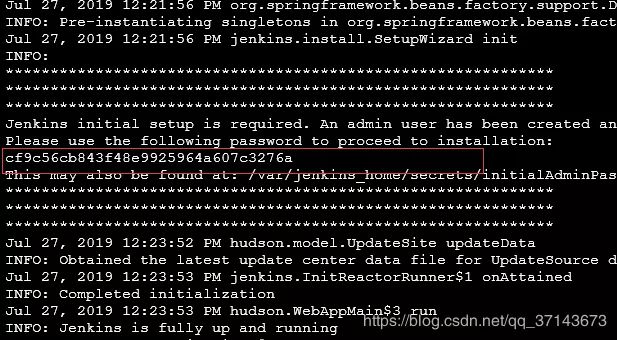
输入密码进入



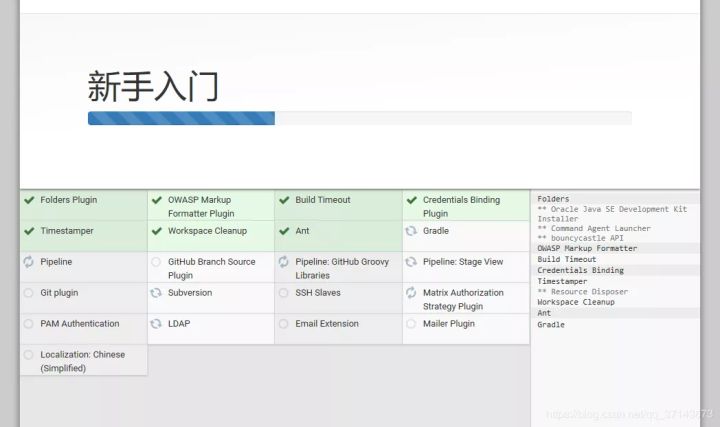
由于我们挂载映射到服务器，所以可以直接通过服务器路径找密码

cat /data/jenkins/secrets/initialAdminPassword

或者直接通过容器日志可以看到密码：docker logs xxxx（容器 ID）



接下来，我选择推荐插件安装



创建第一个账号，我用的 root 123456

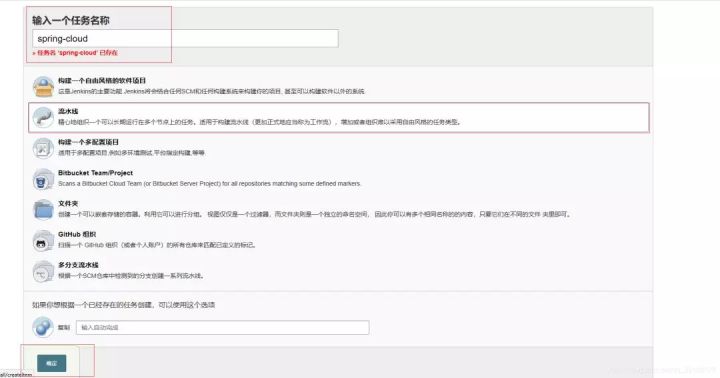


然后完成安装，由于是用的是镜像，所以安装起来非常的简单，海洋版也没有出现插件无法下载的问题

**2.创建项目与持续集成**

踩坑PS：为什么说难用，就是在这。一开始我是把 jdk 和 maven 也挂载到容器里的，但是配置完全局变量后，根本无法使用，可能是容器内也要配置好这些才行，我就搞不懂容器内配了那还要专门配个全局环境变量干啥。。。自动安装也是完全没用，所以我放弃了，用我自己的方法进行持续集成部署。

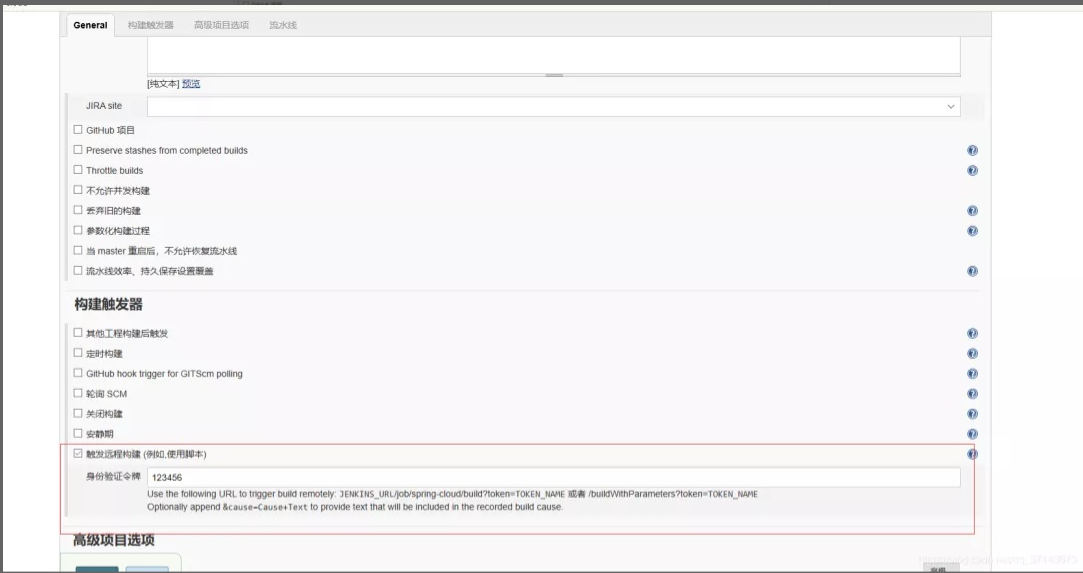
**接下来配置项目，选择流水线**

****

**配置**

这里我们通过 webhook，让项目提交后，自动调用钩子函数进行部署，进行持续集成

**配置构建触发器，也就是钩子函数**

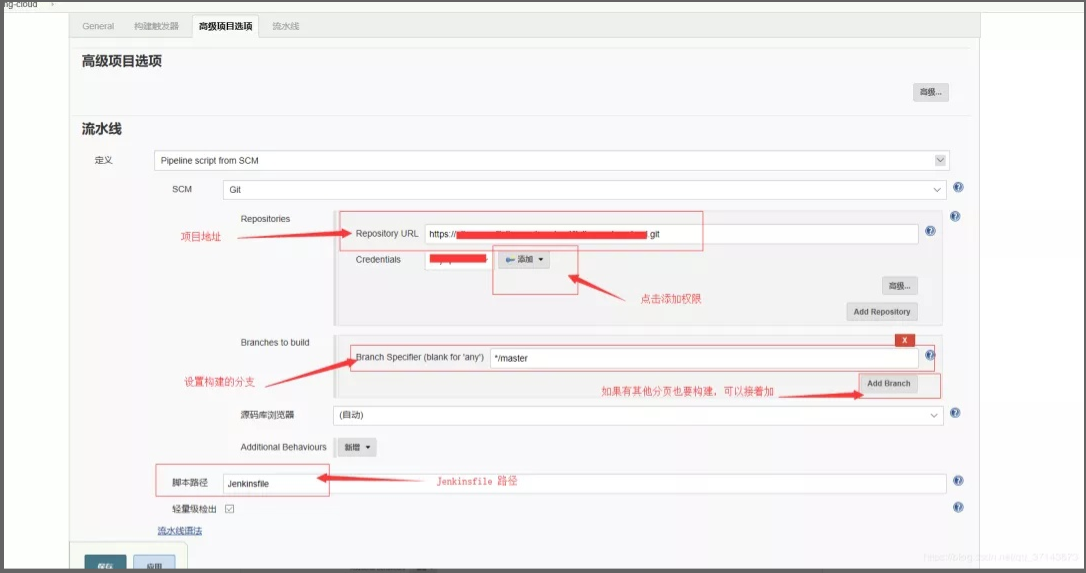


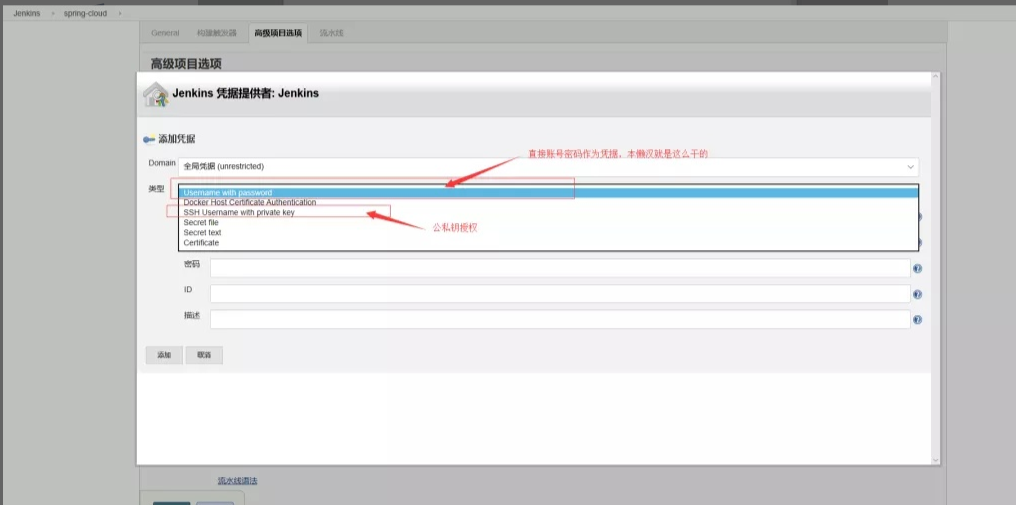
身份验证令牌自己写的，所以这里钩子函数的路径就是

http://127.0.0.1:7005/job/spring-cloud/build?token=123456

PS：身份验证令牌框下面就是钩子函数的请求路径说明，不太明白的，可以自己看看，我上面给的这个是示例

**配置项目**





公私密钥的可参考：

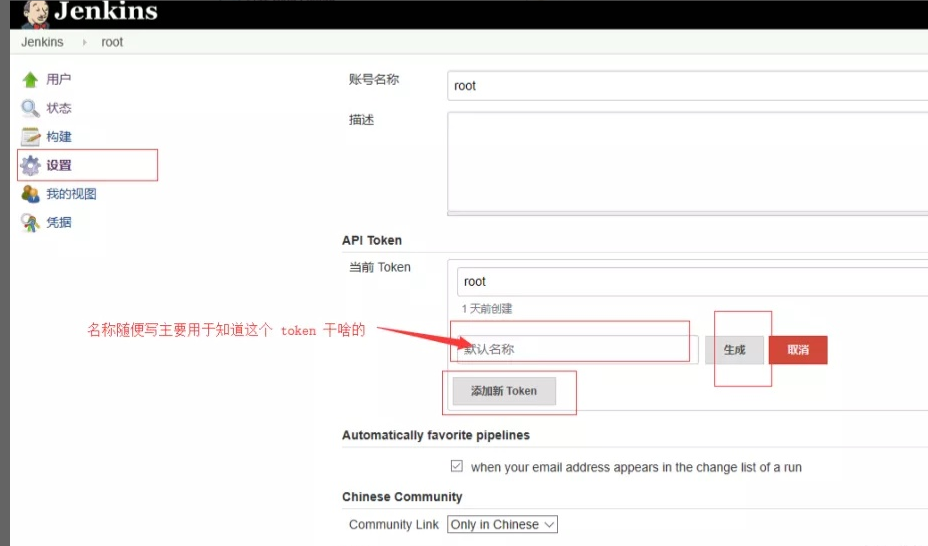
[https://blog.csdn.net/weixin\_41235146/article/details/81780894](https://link.zhihu.com/?target=https%3A//blog.csdn.net/weixin_41235146/article/details/81780894)

<https://blog.csdn.net/weixin_41235146/article/details/81780894>

然后保存，这里暂时就配好了

**root 创建 API token**

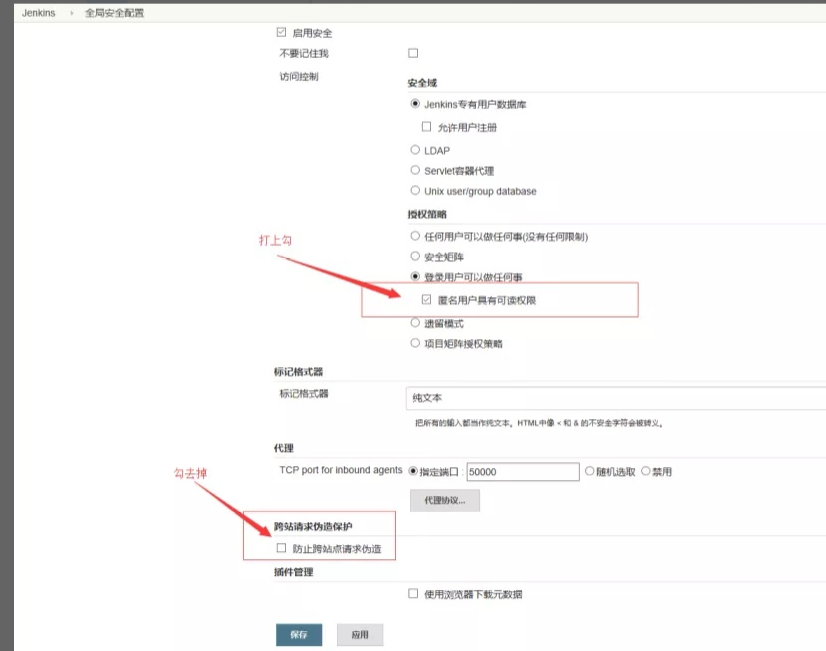
****



****

记住这串 token，后面要和钩子函数一起使用，离开这个页面后再进入就看不到这串 token 了

**系统管理 => 全局安全配置**



PS踩坑：这里不配置的话，后面钩子函数的请求都会 403 失败

高版本jenkins不能界面禁用跨站请求伪造保护。

禁用跨站请求伪造保护操作如下：

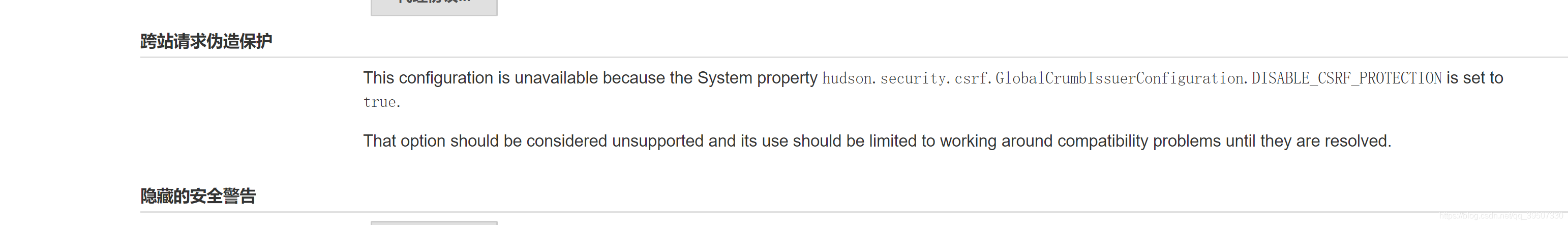
修改jenkins的配置文件。vim /etc/sysconfig/jenkins



JENKINS\_JAVA\_OPTIONS="-Djava.awt.headless=true -Dhudson.security.csrf.GlobalCrumbIssuerConfiguration.DISABLE\_CSRF\_PROTECTION=true"

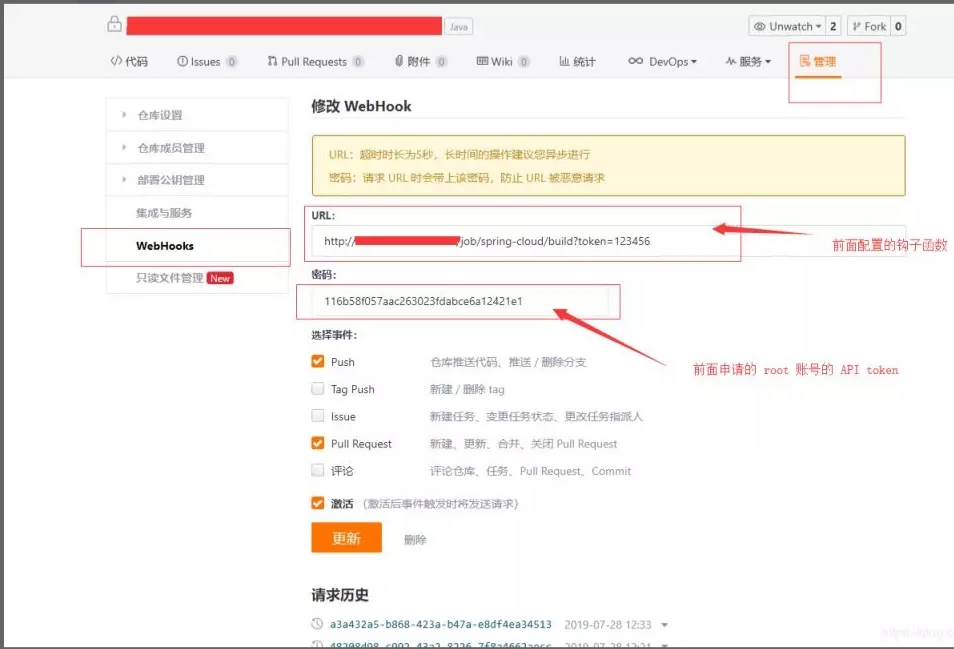
配置后重启jenkins。service jenkins restart

访问网页查看：如下图效果



接下来配置钩子函数的请求了，这里以 gitee 为例

**进入项目 => 管理 => WebHook**



请求历史可以看到请求成功 201



先说下本人用于测试的这个项目是父子项目，打包时候把每个子项目打成 jar 包，然后通过 Dockerfile 生产 docker 镜像，然后启动镜像生成容器，这就是我的部署流程

项目根目录下新增 Jenkinsfile

PS踩坑：由于 maven 配置不成功，我也不想进容器配置 maven，因为服务器才 1G ，进容器就死机了，所以我改了依赖 maven 镜像部署，然后再通过另一个任务把 jar 包生成镜像，因为 maven 容器里没有 docker 环境

pipeline {

agent any

stages {

stage('package') {

agent {

docker {

image 'maven'

args '-v /root/.m2:/root/.m2 -v /data/maven/apache-maven-3.6.0/conf/settings.xml:/root/.m2/settings.xml --entrypoint='

}

}

steps {

script{

echo "WORKSPACE：${env.WORKSPACE}"

echo "Branch：${env.NODE\_NAME}"

if ("${env.NODE\_NAME}" == "master") {

sh "sh package-prod.sh"

}

}

}

}

stage('build') {

agent none

steps {

script{

echo "WORKSPACE：${env.WORKSPACE}"

echo "Branch：${env.NODE\_NAME}"

if ("${env.NODE\_NAME}" == "master") {

sh "sh build-prod.sh"

}

}

}

}

}

}

先 pull 个 maven 镜像，以及在构建镜像时候所需要的 jdk8 的镜像

# maven 用于 Jenkins 构建有 maven 环境的容器

docker pull maven

# jdk8 的镜像用于 Dockerfile 中设置构建拥有 jdk8 环境镜像的基础镜像

docker pull kdvolder/jdk8

* image '[http://docker.io/maven](https://link.zhihu.com/?target=http%3A//docker.io/maven)'：基于这个镜像生成容器部署，所以环境有 maven 环境
* args 中 -v /root/.m2:/root/.m2：把依赖挂载导服务器上，不用每次打包都下依赖  
  PS：首先，你要知道这个流程是如何进行的，在你根据下的那个 maven 镜像去打包时，由于内外 docker 建立了通讯，所有服务器的 docker 就会生成一个你 maven 的容器，而你的命令都会在这个容器里进行，所以这里挂载就能直接把 maven 这个临时容器里的依赖挂载到服务器上，打包的过程中，你用 docker ps 是可以看到这个 maven 容器的
* args 中 -v /data/maven/apache-maven-3.6.0/conf/settings.xml:/root/.m2/settings.xml：阿里环境配置挂进去  
  PS踩坑：这里注意，挂载进去的 settings.xml 文件必须放在 .m2 文件夹下，不是放在 maven 的 conf 下面，那样不会生效
* --entrypoint=：配置下好像就不会打开进入容器了，没深入研究
* ${env.WORKSPACE}：工作空间路径
* ${env.NODE\_NAME}：进行的分支

Jenkins 内置环境变量可参考：

[https://www.cnblogs.com/puresoul/p/4828913.htm](https://link.zhihu.com/?target=https%3A//www.cnblogs.com/puresoul/p/4828913.htm)

可以看到，我这部署先判度为 master 分支的情况下，package 任务中调用脚本进行打包，build 任务下调用脚本部署

下面贴下这两个脚本，大伙可以参考下

**package-prod.sh**

#!/bin/sh

#进入文件根目录

#cd "$WORKSPACE"

#项目打包

mvn clean install package '-Dmaven.test.skip=true'

**build-prod.sh**

#!/bin/sh

#进入文件根目录

#cd "$WORKSPACE"

#启用 prod 配置

ActiveProfiles=prod

#基本信息需要配置

#内部端口

targetPort=8081

#旧镜像版本号

oldVendor=1.0.0

#镜像版本号

vendor=1.0.0

#项目名

projectName=eureka

#进入target文件夹

#直接的构建是再容器里，这个是在 Jenkins 容器里，所以空间不一样

#容器的空间是原空间路径后面多了 @2

cd $WORKSPACE@2/$projectName/target

#创建Dockerfile文件

#-jar -Duser.timezone=GMT+08 保证生成出来的容器的时区与服务器一致

cat << EOF > Dockerfile

FROM kdvolder/jdk8

MAINTAINER $projectName

VOLUME /tmp

LABEL app="$projectName" version="$vendor" by="$projectName"

COPY $projectName.jar $projectName.jar

EXPOSE $targetPort

CMD -Xmx100m -Xms100m -jar -Duser.timezone=GMT+08 $projectName.jar --spring.profiles.active=$ActiveProfiles

ENTRYPOINT java

EOF

#删除镜像下所有容器

docker rm -f $(docker ps -a | grep "$projectName" | awk '{print $1}')

#删除旧镜像

docker rmi -f $projectName:$oldVendor

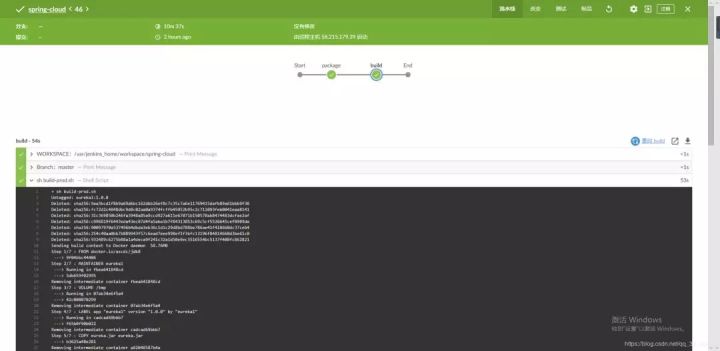
#创建镜像

docker build -t $projectName:$vendor .

#启动镜像生成容器

docker run --name $projectName -d -p $targetPort:$targetPort $projectName:$vendor

我写配置喜欢统一性配置，后面改起来简单，上面参数换下下面都不需要改，这里我的文件夹名，镜像名和生产的 jar 名都用一样的也是这个目的



打包成功，这画面看的好看不少

本文将讲解如何将 Jenkins 设置为中文，但是要注意，Jenkins 的中文翻译并不完全。

1. 安装 locale 插件，该插件安装后才可以在系统配置中设置语言；



安装Localization Chinese插件，可以搜索Chinese安装；



 重启 Jenkins (安装插件时就可以选择重启，否则就重启 Jenkins服务即可)；

 进入系统管理Manage Jenkins > Configure System，找到 Locale 配置，设置为 zh\_cn。



#### 浏览器进入Jenkins，登录

http://localhost:8080/

* 1

#### 关闭Jenkins

http://localhost:8080/exit

* 1

#### 重启Jenkies

http://localhost:8080/restart

* 1

#### 重新加载配置信息

http://localhost:8080/reload

### docker搭建jenkins工程访问路径遇到的坑

当我们在用docker-compose创建jenkins镜像的后，默认访问端口是8080，而且是不带工程名称的，当我们用nginx进行代理的时候就会遇到问题。  
比如我们想让nginx实现这样的访问http://192.169.0.1/jenkins，这样是不行的。下面我贴出我的配置来解决这个问题：

version: '3'

services:

jenkins:

build:

context: ./

dockerfile: ./Dockerfile

image: my-jenkins:version\_2.0

container\_name: jenkins

restart: always

user: root

ports:

- '8080:8080'

command: ["--httpPort=8080","--prefix=/jenkins"]

volumes:

- './common/jenkins\_home:/var/jenkins\_home'

- './common/package:/usr/local/package'

- './common/maven/repo:/opt/maven/repo'

- '/var/run/docker.sock:/var/run/docker.sock'

- '/home/docker/lib:/usr/bin/docker'

- '/bin/docker:/bin/docker'

我们可以看到下面这段关键的配置

command: ["--httpPort=8080","--prefix=/jenkins"]

* 1

这句代码为jenkins设置了端口（也可以设置我们自定义的端口），并设置了访问工程名称  
想深入研究的朋友们，可以进入容器里面，看看/usr/local/bin/jenkins.sh这个脚本，执行启动是允许传入参数的

另外nginx的配置我就不贴了。。。