Docker应用部署方案

# 参考资料

[Docker Docs](https://docs.docker.com/)

[docker常用命令、实操（包含实用配置）\_docker批量启动容器-CSDN博客](https://blog.csdn.net/Acegem/article/details/123250271)

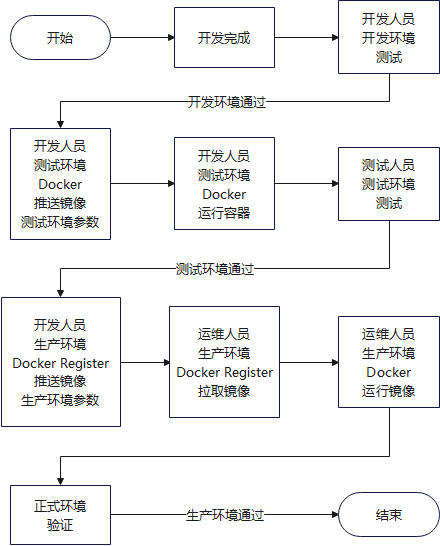
[IDEA整合Docker实现一键部署及处理远程连接导致的问题\_idea连接docker-CSDN博客](https://blog.csdn.net/qq_34383510/article/details/131335119)

[Docker 图形化管理工具，必须推荐给你！\_51CTO博客\_docker可视化管理工具](https://blog.51cto.com/mingongge/5118020)

# 目前生产环境应用部署现状

* 使用Tomcat部署运行
* 使用Jar直接执行

# 使用Docker的测试和上线流程参考



# 环境说明

测试环境：192.168.0.6

生产环境：192.168.0.34

# Docker 安装

**1. 删除老版本**

# 卸载原docker

yum remove docker-ce \

docker-client \

docker-client-latest \

docker-common \

docker-latest \

docker-latest-logrotate \

docker-logrotate \

docker-engine

# 卸载podman

yum remove pdman

**2. 安装工具包并设置存储库**

yum install -y yum-utils

yum-config-manager \

--add-repo \

https://download.docker.com/linux/centos/docker-ce.repo

**3. 安装/升级 docker引擎**

# 安装，使用yum

yum install docker-ce docker-ce-cli containerd.io

# 升级，使用yum，注意升级后使用原文件保存的目录

yum update docker-ce docker-ce-cli containerd.io

**4. 迁移docker目录**

停止docker服务并移动原有docker文件到新目录，配置软链

systemctl stop docker docker.socket

mv /var/lib/docker /opt/

ln -s /opt/docker /var/lib/docker

systemctl start docker

# Docker 安装 Portainer

Portainer 是 Docker 的图形化管理工具，提供状态显示面板、应用模板快速部署、容器镜像网络数据卷的基本操作（包括上传下载镜像，创建容器等操作）、事件日志显示、容器控制台操作、Swarm 集群和服务等集中管理和操作、登录用户管理和控制等功能。功能十分全面，基本能满足个人用户对容器管理的全部需求。

docker volume create portainer\_data

# 版本号可以在 hub.docker.com 上查询。简单的话可用latest代替

docker pull portainer/portainer-ce:2.19.4

docker run -d \

-p 8000:8000 \

-p 9443:9443 \

--name portainer \

--restart=always \

-v /var/run/docker.sock:/var/run/docker.sock \

-v portainer\_data:/data \

portainer/portainer-ce:2.19.4

访问地址：<https://192.168.0.34:9443>

账号：haoyu/haoyu123456789

# Docker 安装 Register

Docker pull的镜像默认来自官方的镜像仓库Docker Hub，一般我们会配置国内镜像仓库如阿里云、网易云等，这些仓库都是公用的，或者托管在第三方平台上的。但商业软件出于保密的需求，需要建立一个私有仓库。用户可以使用Dokcer官方提供的镜像registry来搭建私有仓库。

## 1.安装Docker Register

访问地址：<http://192.168.0.34:5000/v2/_catalog>

docker volume create registryr\_data

# 版本号可以在 hub.docker.com 上查询。简单的话可用latest代替

docker pull registry:2.8.3

docker run -d \

-p 5000:5000 \

--name=local\_registry \

--restart=always \

--privileged=true \

-v registryr\_data:/var/lib/registry \

-e "REGISTRY\_HTTP\_HEADERS\_Access-Control-Origin=[http://192.168.0.34]" \

-e "REGISTRY\_HTTP\_HEADERS\_Access-Control-Allow-Methods=[HEAD,GET,OPTIONS,DELETE]" \

-e "REGISTRY\_HTTP\_HEADERS\_Access-Control-Credentials=[true]" \

-e "REGISTRY\_HTTP\_HEADERS\_Access-Control-Allow-Headers=[Authorization,Accept,Cache-Control]" \

-e "REGISTRY\_HTTP\_HEADERS\_Access-Control-Expose-Headers=[Docker-Content-Digest]" \

-e "REGISTRY\_STORAGE\_DELETE\_ENABLED=true" \

registry:2.8.3

## 2.安装界面docker-registry-ui

访问地址：<http://192.168.0.34:5001/>

# 版本号可以在 hub.docker.com 上查询。简单的话可用latest代替

docker pull joxit/docker-registry-ui:2.5.6

docker run -d \

-p 5001:80 \

--name=registry-ui \

--restart=always \

--privileged=true \

-e "SINGLE\_REGISTRY=true" \

-e "REGISTRY\_TITLE=Docker Registry UI" \

-e "DELETE\_IMAGES=true" \

-e "SHOW\_CONTENT\_DIGEST=true" \

-e "NGINX\_PROXY\_PASS\_URL=http://192.168.0.34:5000" \

-e "SHOW\_CATALOG\_NB\_TAGS=true" \

-e "CATALOG\_MIN\_BRANCHES=1" \

-e "CATALOG\_MAX\_BRANCHES=1" \

-e "TAGLIST\_PAGE\_SIZE=100" \

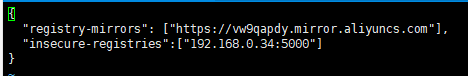
-e "REGISTRY\_SECURED=false" \

-e "CATALOG\_ELEMENTS\_LIMIT=1000" \

joxit/docker-registry-ui:2.5.6

## 2.配置docker信任私有镜像仓库地址

编辑/etc/docker/daemon.json文件，添加insecure-registries配置项



# 1.编辑daemon.json

vim /etc/docker/daemon.json

# 2.添加insecure-registries配置项，其中阿里云加速地址不要修改

{

"registry-mirrors":["https://vw9qapdy.mirror.aliyuncs.com"],

"insecure-registries":["192.168.0.34:5000"]

}

# 3.重启

systemctl daemon-reload

systemctl restart docker

## 3.常用操作和命令

上传、下载镜像

创建好私有仓库之后，就可以使用 docker tag 来标记一个镜像，然后推送它到仓库。

#1.先在本机查看已有的镜像，选择其中一个进行上传

docker images

#2.镜像标记(push前必须先标记镜像)

##镜像标记语法，自定义repository 不能有大写字母

docker tag IMAGE[:TAG] [REGISTRY\_HOST[:REGISTRY\_PORT]/]REPOSITORY[:TAG]

docker tag 自定义镜像名[:版本] 私服IP:端口/仓库名[:版本]

例:

docker tag ruoyi/ruoyi-server:4.7.0 192.168.0.34:5000/ruoyi/ruoyi-server:4.7.0

#标记完后再执行查看所有镜像，会多出一个镜像

docker images

192.168.0.34:5000/ruoyi/ruoyi-server

#3.镜像上传

docker push 192.168.0.34:5000/ruoyi/ruoyi-server:4.7.0

#4.注册服务器查看命令

#用curl 查看注册服务器的所有镜像

curl 192.168.0.34:5000/v2/\_catalog

#列出指定镜像有哪些tag

curl 192.168.0.34:5000/v2/ruoyi/ruoyi-server/tags/list

#5.删除本地的镜像

#删除指定镜像

docker rmi 192.168.0.34:5000/ruoyi/ruoyi-server:4.7.0

#删除所有镜像

docker rmi -f $(docker images -qa)

注1：通过docker images命令可以看到，本地已经没有了192.168.217.143:5000/test1:1.0.0

#6.再从registry下载镜像

docker pull 192.168.0.34:5000/ruoyi/ruoyi-server:4.7.0

# IDEA配置Docker

## 1.开启Docker远程连接

idea整合docker，需要连接到docker，需要开启docker的远程连接

# 修改docker服务启动配置文件

vi /usr/lib/systemd/system/docker.service

# 在 ExecStart= 最后加上，见下图示意

-H unix:///var/run/docker.sock -H tcp://0.0.0.0:2375

# 重启docker

systemctl daemon-reload

systemctl restart docker

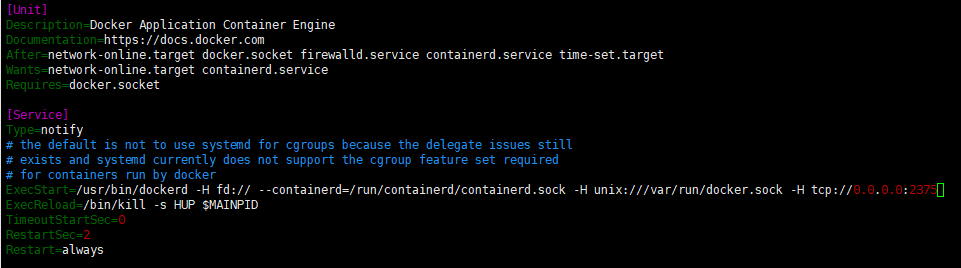
# 查看端口是否开放

netstat -ntlp |grep 2375

# 防火墙开放2375端口

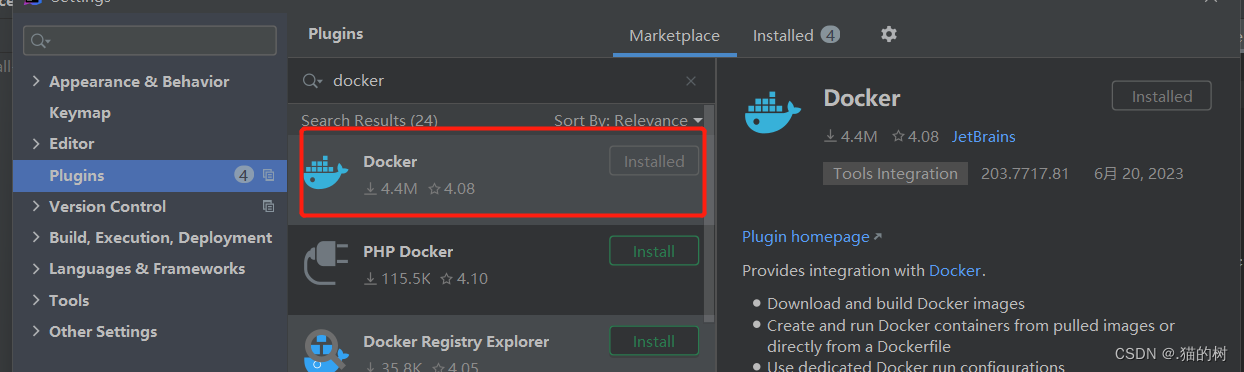
# 注意此方式仅用于测试环境打包和上传镜像。和正式环境打包镜像

# 注意生产环境严禁配置docker的远程连接，严禁使用此方式打包和上传镜像



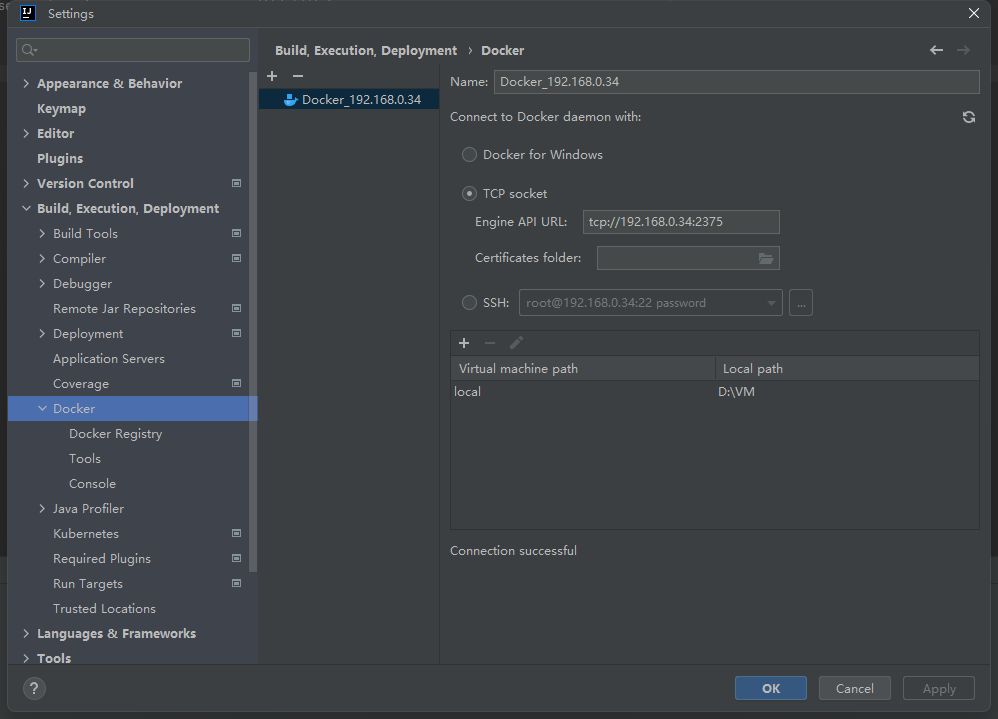
## 2.配置IDEA的Docker插件

### 2.1.IDEA安装Docker插件

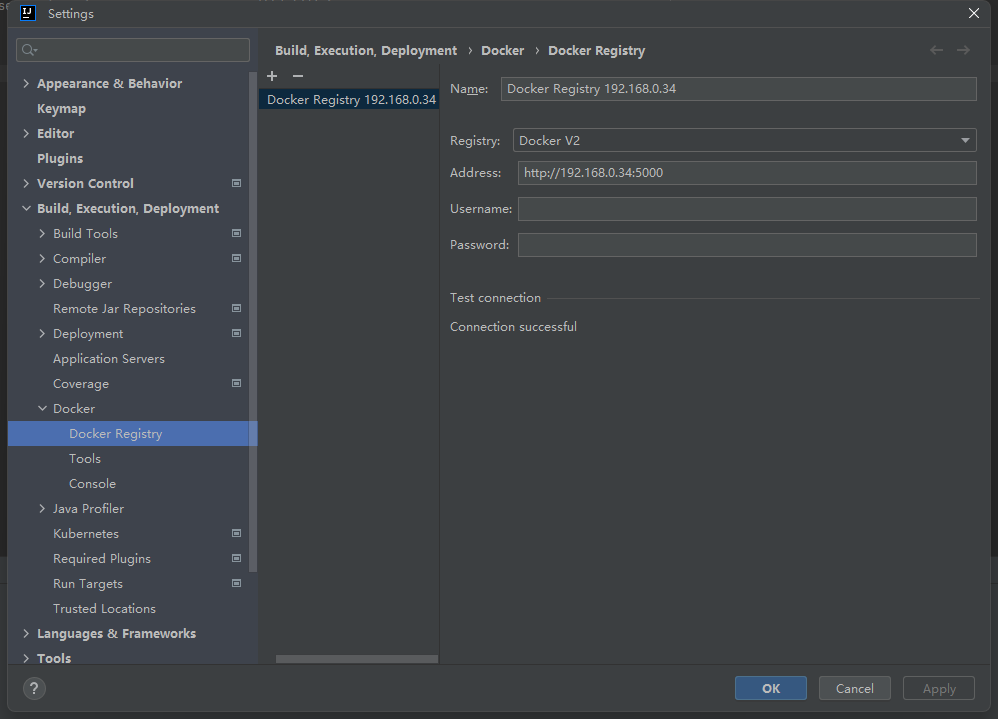


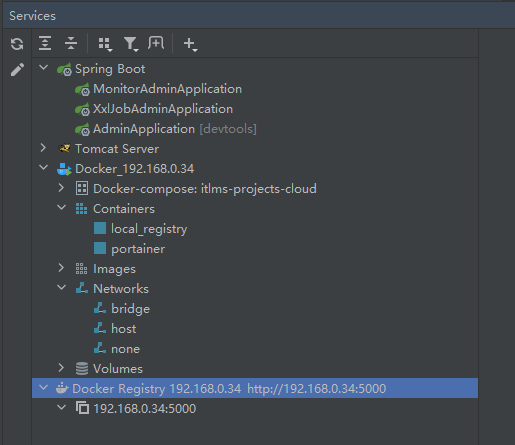
### 2.2.配置Docker信息

配置Docker：



配置Docker Register：

打开IDEA底部services，已经能看到连接上Docker和所有的镜像和容器信息，DockerRegister信息。



## 3.打包镜像：打包镜像并推送到测试环境Docker

Dockerfile示例：

FROM anapsix/alpine-java:8\_server-jre\_unlimited

MAINTAINER Haoyu

RUN mkdir -p /opt/server/logs \

/opt/server/temp \

/opt/skywalking/agent

WORKDIR /opt/server

# 端口设置

ENV SERVER\_PORT=9060

# 配置文件

ENV PROFILE=dev

EXPOSE ${SERVER\_PORT}

ADD ./target/itlms\_projects\_plus\_servers-admin.jar ./app.jar

ENTRYPOINT ["java", \

"-Djava.security.egd=file:/dev/./urandom", \

"-Dserver.port=${SERVER\_PORT}", \

# 应用名称 如果想区分集群节点监控 改成不同的名称即可

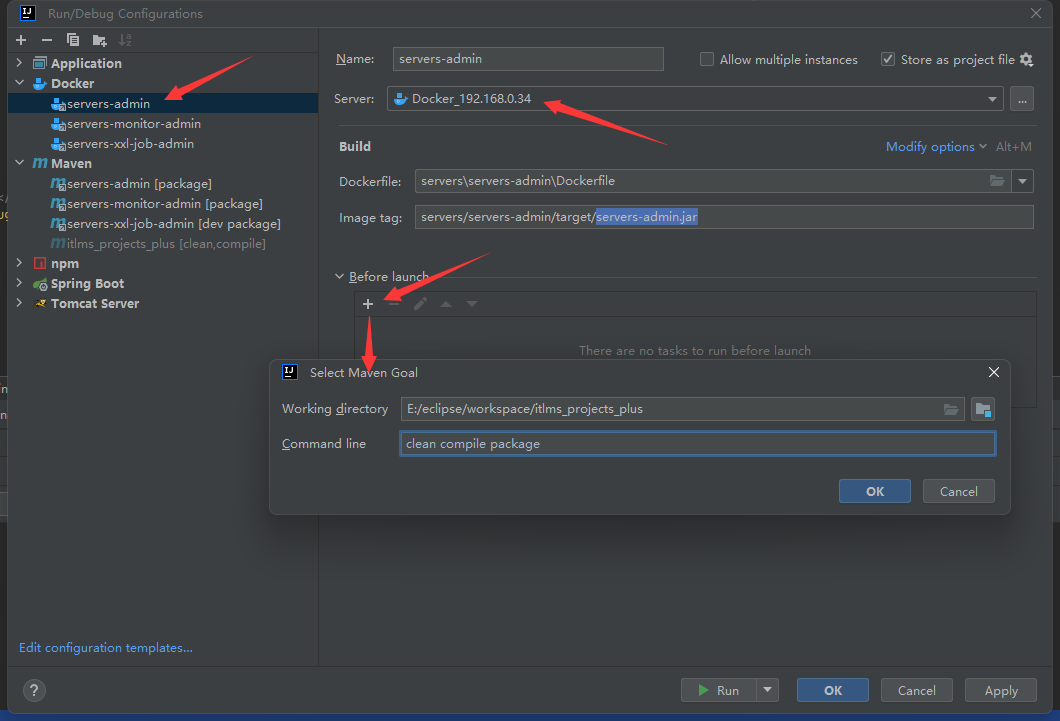
# "-Dskywalking.agent.service\_name=servers-server", \

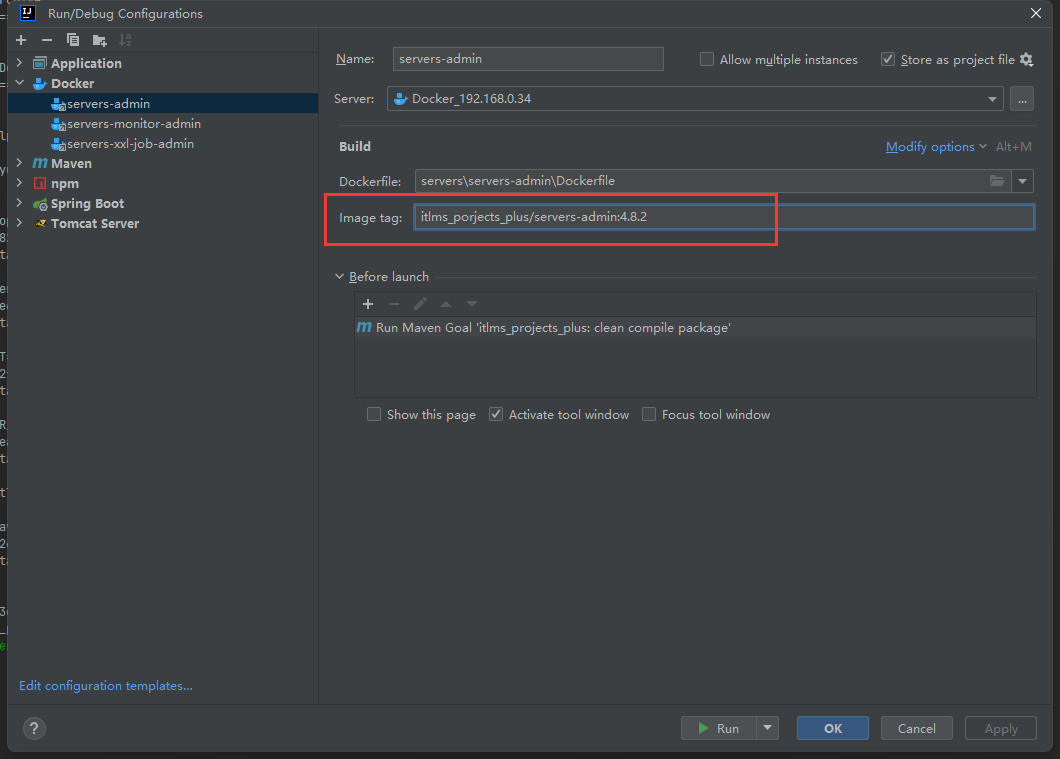
# "-javaagent:/ruoyi/skywalking/agent/skywalking-agent.jar", \

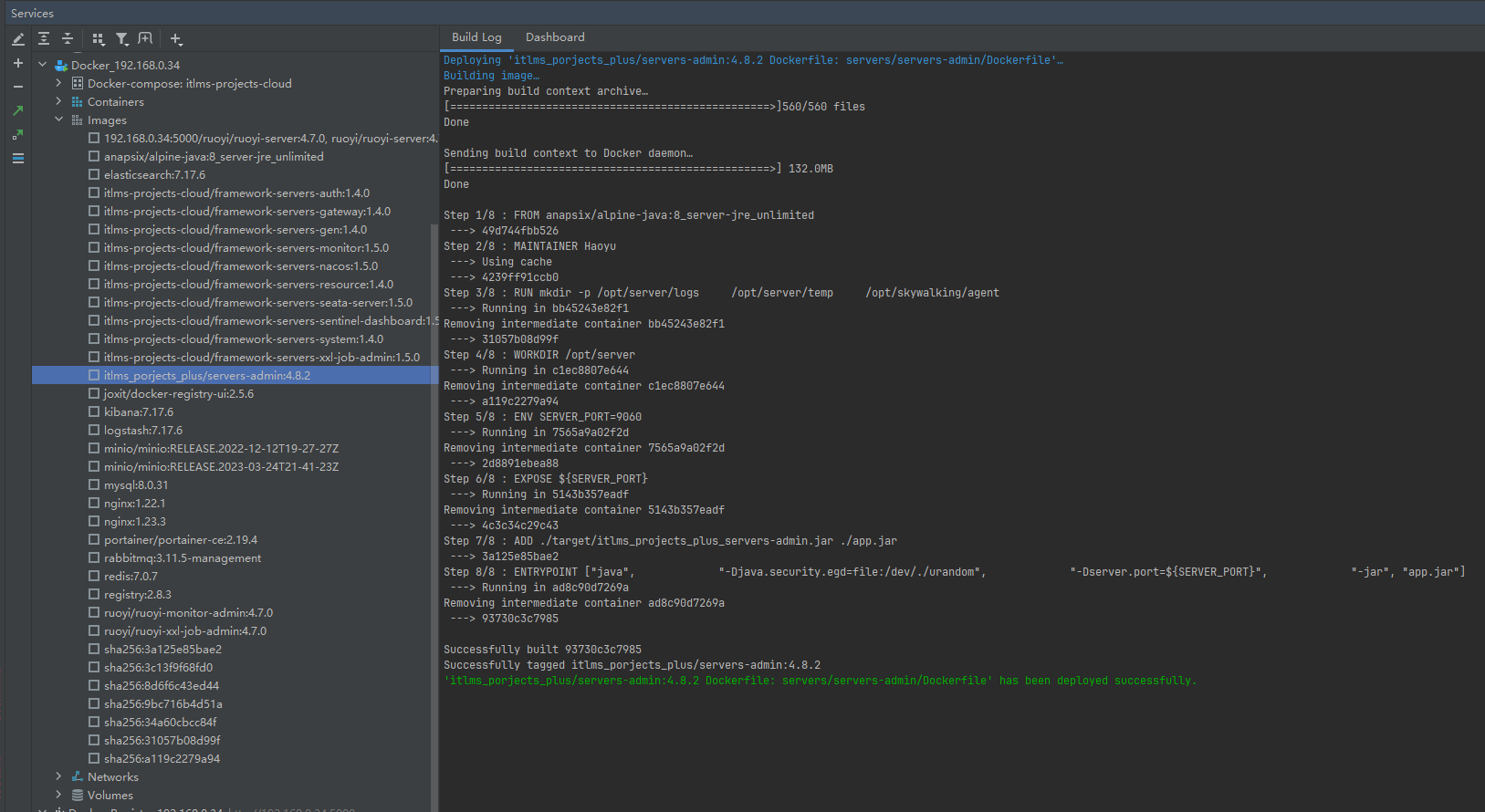
"-jar", "app.jar", \

"--spring.profiles.active=${PROFILE}"]

配置Docker运行参数，运行前先进行打包

镜像名称设定

执行完成后，镜像已经在测试环境的docker中

等同于命令：

cd /opt/src

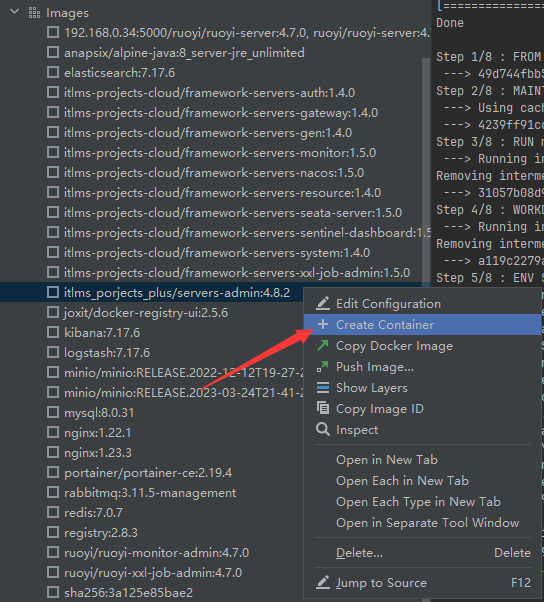
docker build -f /opt/src/Dockerfile -t itlms\_porjects\_plus/servers-admin:4.8.2 .

-f，--file:指定 dockerfile 路径

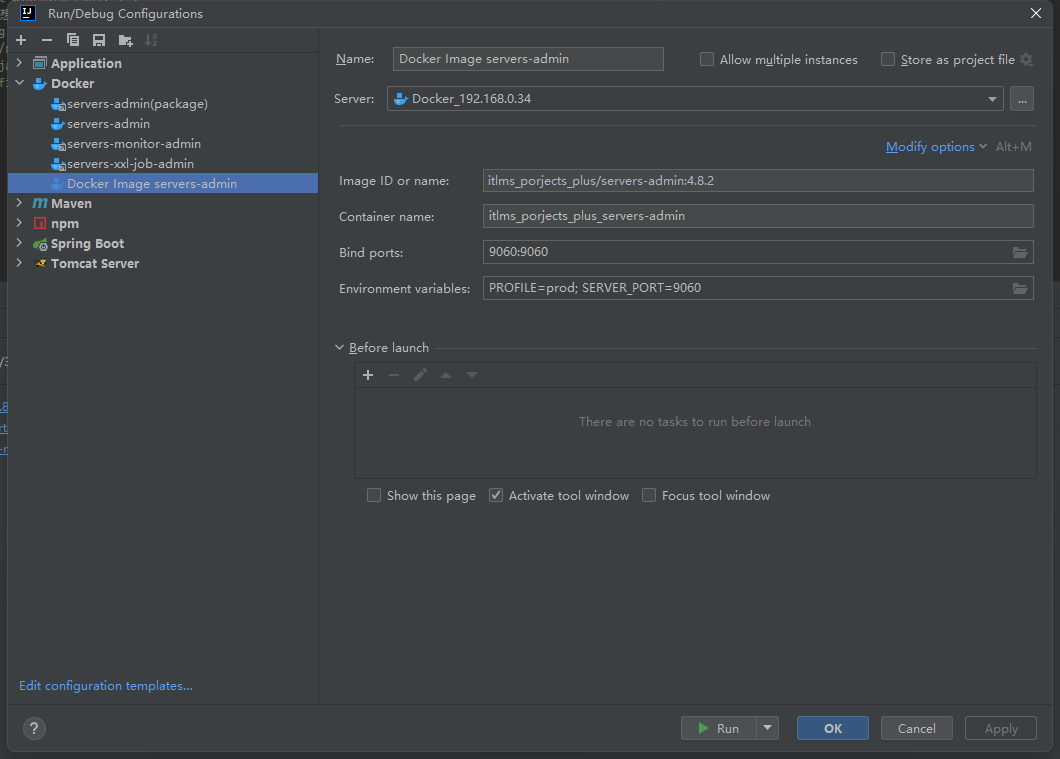
-t，--tag:指定构建的镜像名和 tag，可以指定多个 tag

## 4.运行容器：测试环境Docker建立容器并运行

在镜像上鼠标右键菜单中选择Create Container



简单配置容器名，端口映射，磁盘映射，dockerfile中的环境变量可以在这里配置



等同于命令：

docker run -d \

-p 9060:9060 \

--name itlms\_porjects\_plus\_servers-admin \

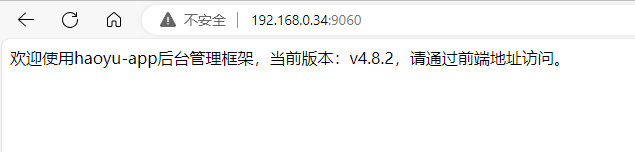
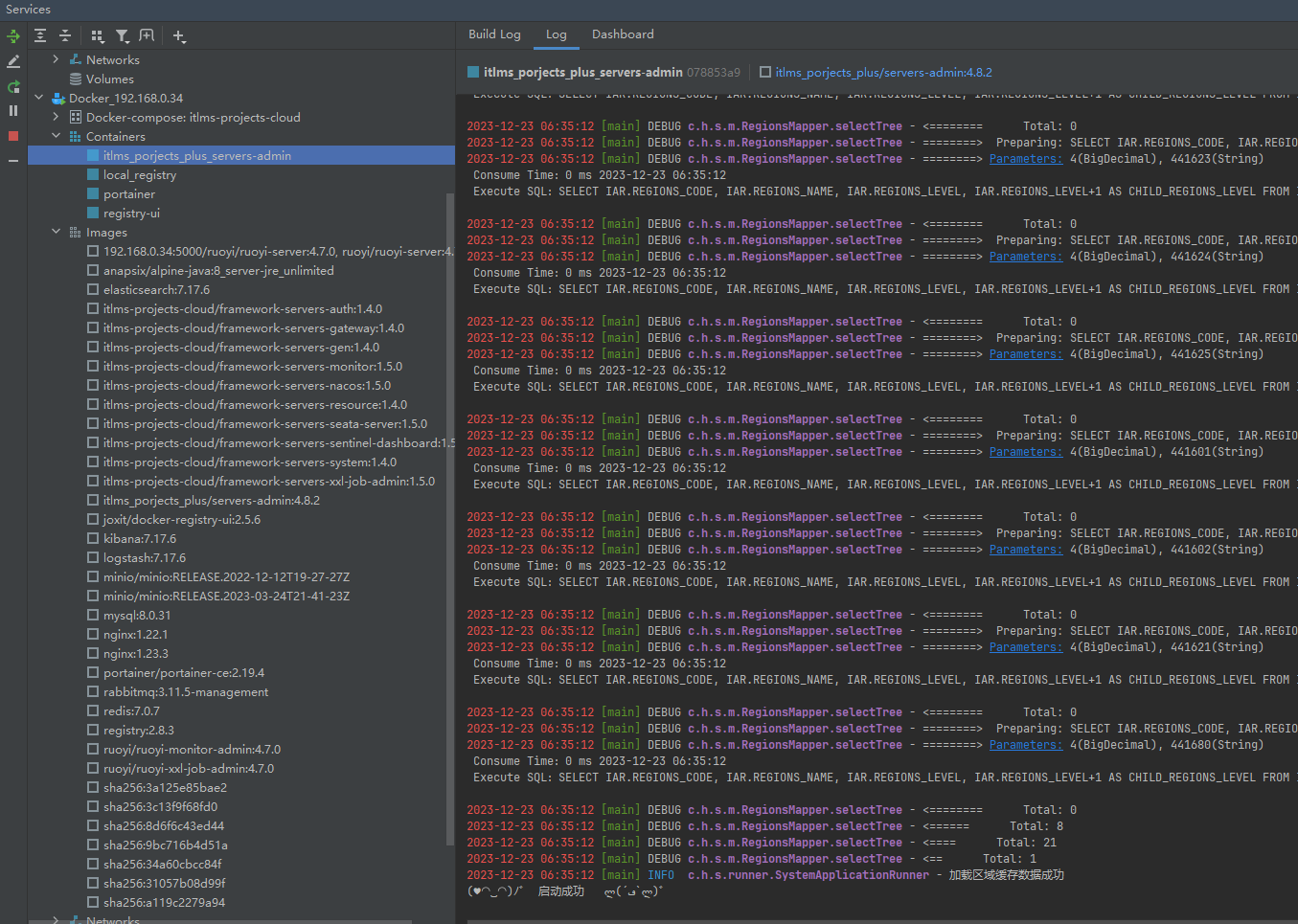
--restart=always \

-e "PROFILE=prod" \

-e "SERVER\_PORT=9060" \

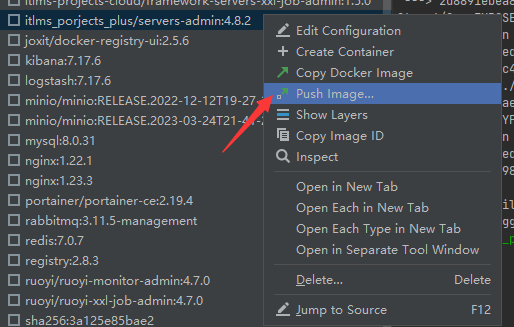
itlms\_porjects\_plus/servers-admin:4.8.2

启动完毕

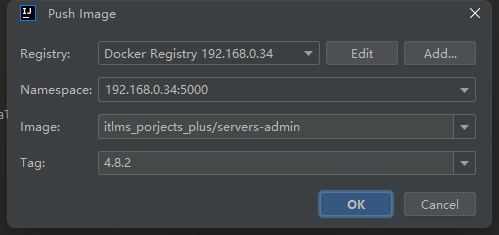


## 5.推送镜像到镜像仓库

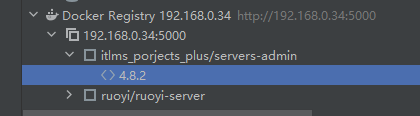
在镜像上鼠标右键菜单中选择Push Image

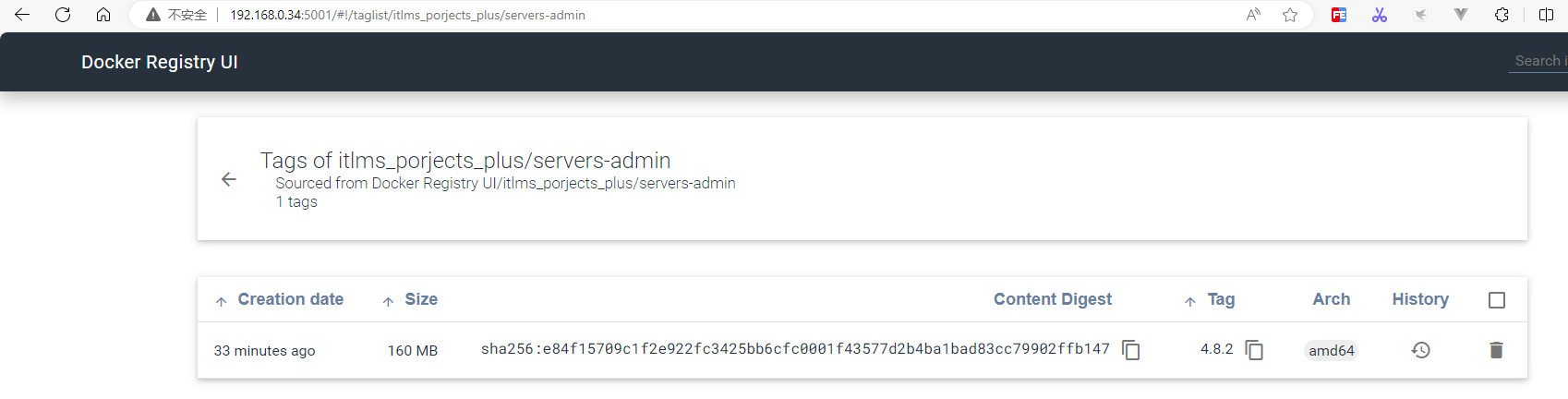


选择私有镜像仓库地址（可以考虑将生产环境的仓库地址配置到公网，配置账号密码）



推送完毕，可以看到镜像到仓库中





# 测试(生产)环境使用私有镜像仓库

测试(生产)环境可以通过拉取私有镜像仓库中的镜像生成容器运行

# 拉取镜像

docker pull 192.168.0.34:5000/itlms\_porjects\_plus/servers-admin:4.8.2

# 生成容器并运行

docker run -d \

-p 9060:9060 \

--name=itlms\_porjects\_plus\_servers-admin \

--restart=always \

--privileged=true \

-e "PROFILE=prod" \

-e "SERVER\_PORT=9060" \

192.168.0.34:5000/itlms\_porjects\_plus/servers-admin:4.8.2

运行结果：

