

镜像相当于类 而容器相当于实例

# 安装docker

udo apt-get install docker.io

Systemctl start docker

Systemctl status docker

Docker info 查看消息

Systemctl stop docker 停止docker

Systemctl enable docker 开机启动docker

查看镜像 docker images

拉取镜像 docker pull

Automated dockerhub 注册中心 公共仓库创建出来的

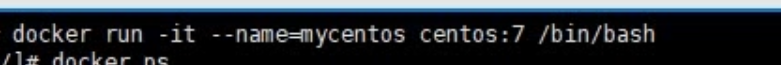


删除镜像 docker rmi name/id





在容器中启动镜像centos7



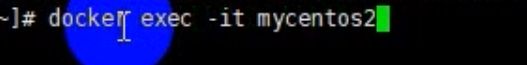
Exit 退出容器

Docker ps -a check all docker container

后台守护进程启动

# Execute one container

Docker exec -it id/DockerName



## 启动和停止

Docker starrt id

Docker stop id

### Copy file to docker

Cp filename DockerId:/dir

Copy file from docker

Cp dockerid:/dir/file name filename default:current filename

## Dir 挂载



## Check the runtime info of docker

Docker inspect --format name/id

Check the IPAdress

Docker inspect --format=’{{.NetworkSettings.IPAddress}}’ dockername/id

# Delete container

Docker rm name/id

# Mysql deploy

Docker pull mysql:5.7

Docker run -di --name=mysql -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=root mysql:5.7

Port mapping root pwd protect thread interminal mode -e means environment

Mysql -uroot -proot --default

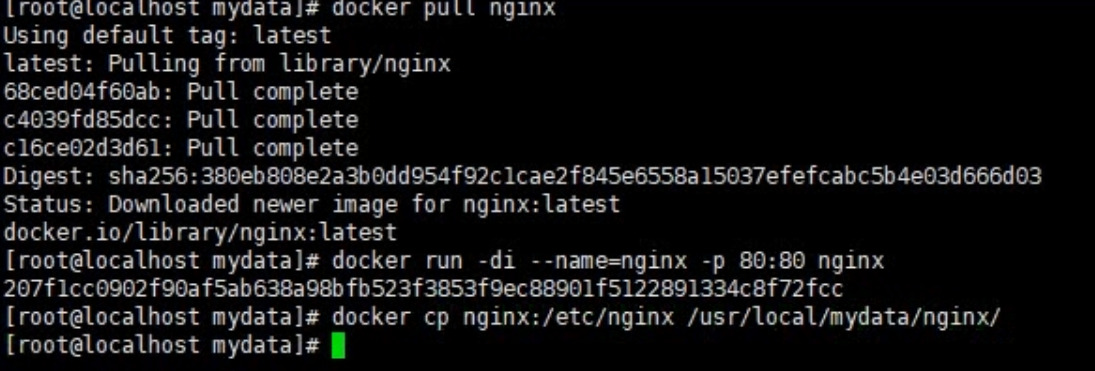


连接mysql

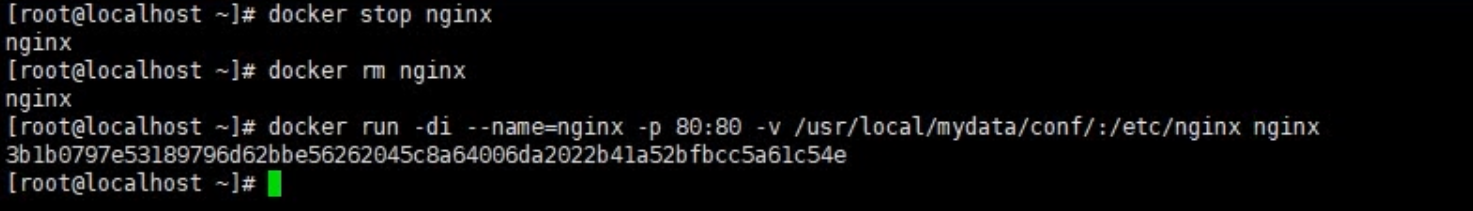
Navicat connect hostmachine mysql in docker



# Nginx



目录挂载 先停止并移除启动的nginx



# Redis docer

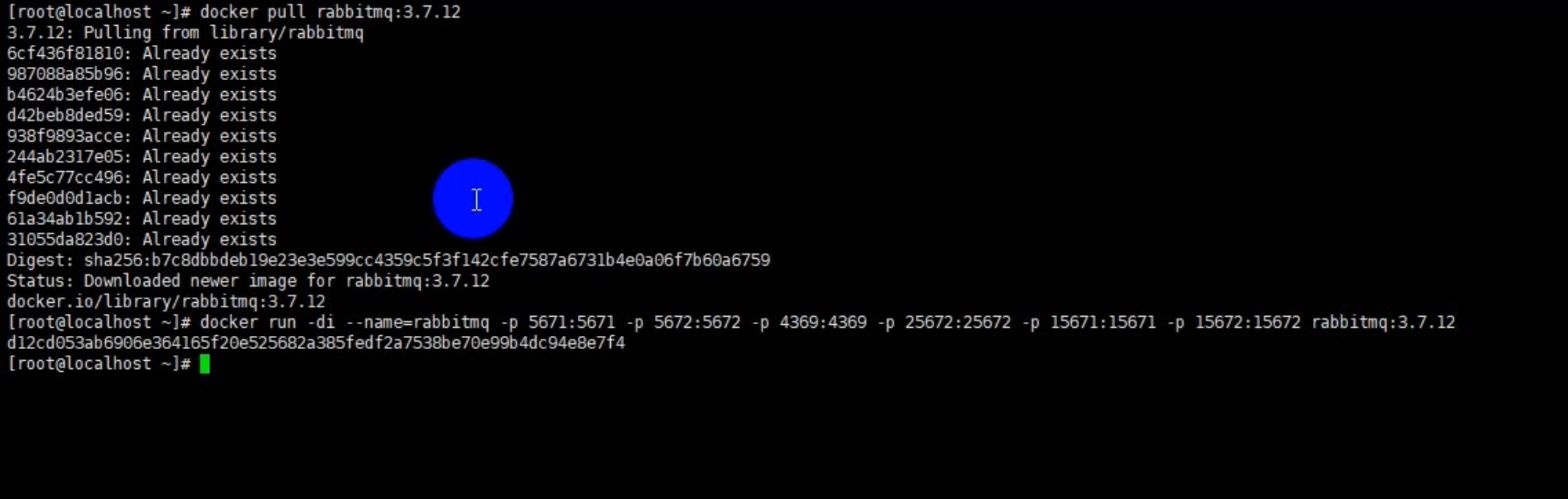
先pull 镜像

然后启动

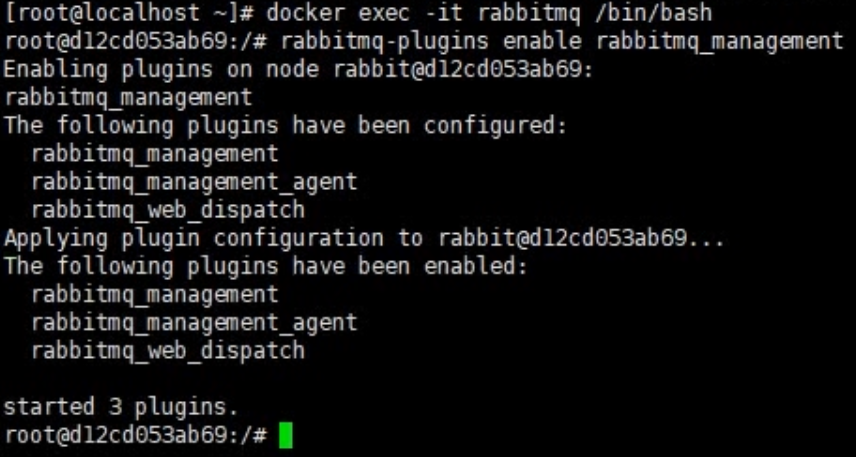
docker run -di --name=redis -p 6388:6379 redis

注意默认端口号 docket container 一定是6379

# Rabbitmq



启动ui 界面管理

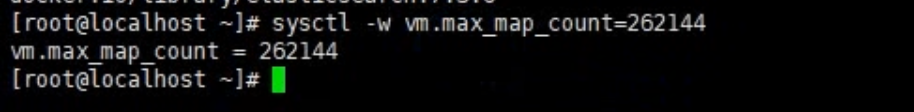




# Elastic search

Pull elasticsearch 7.5.0

修改虚拟内存大小



创建容器

docker run -di --name=elasticsearch -p 9200:9200 -p 9300:9300 -e

"discovery.type=single-node" -e "cluster.name=elasticsearch" -v

/mydata/elasticsearch/plugins:/usr/share/elasticsearch/plugins

elasticsearch:7.5.0

安装中文分词器

docker exec -it elasticsearch /bin/bash

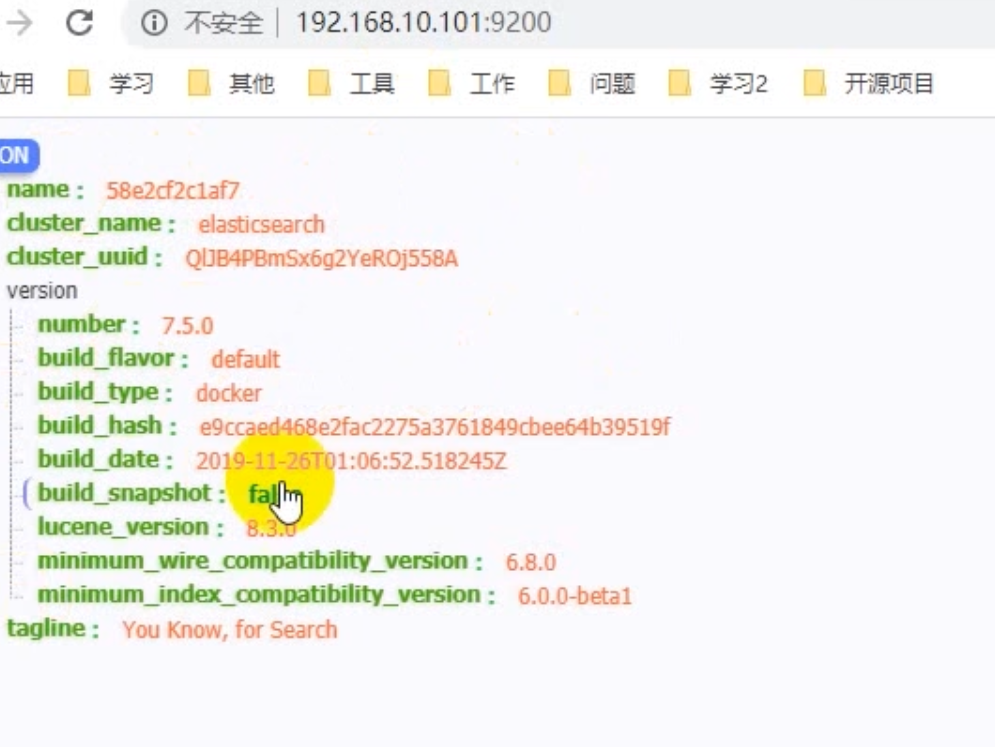
#此命令需要在容器中运行

elasticsearch-plugin install https://github.com/medcl/elasticsearch-analysis-

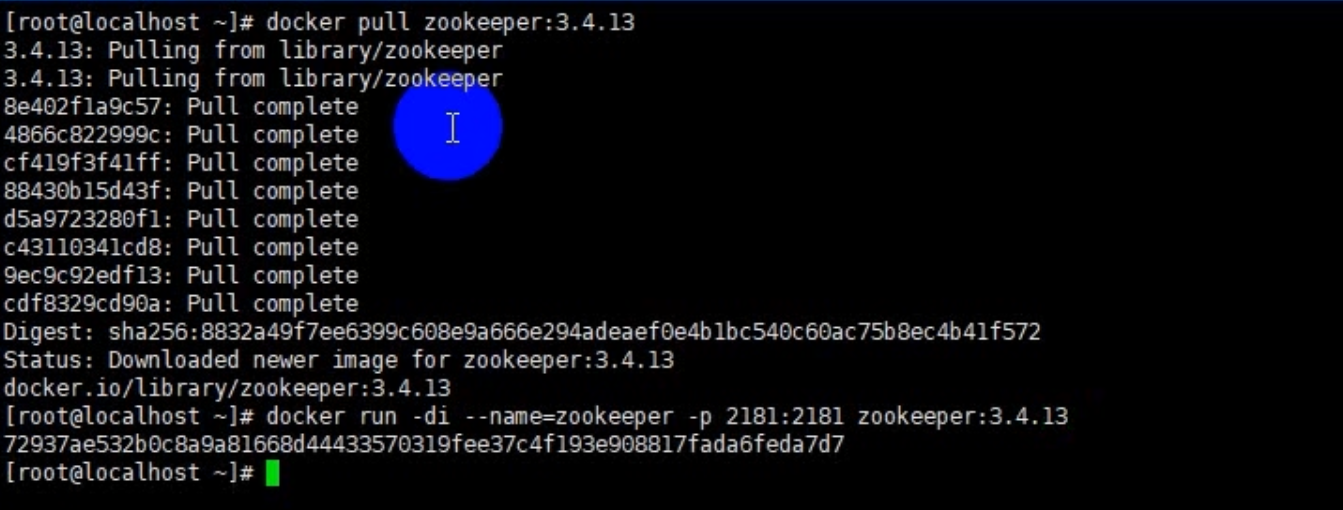
ik/releases/download/v7.5.0/elasticsearch-analysis-ik-7.5.0.zip

docker restart elasticsearch

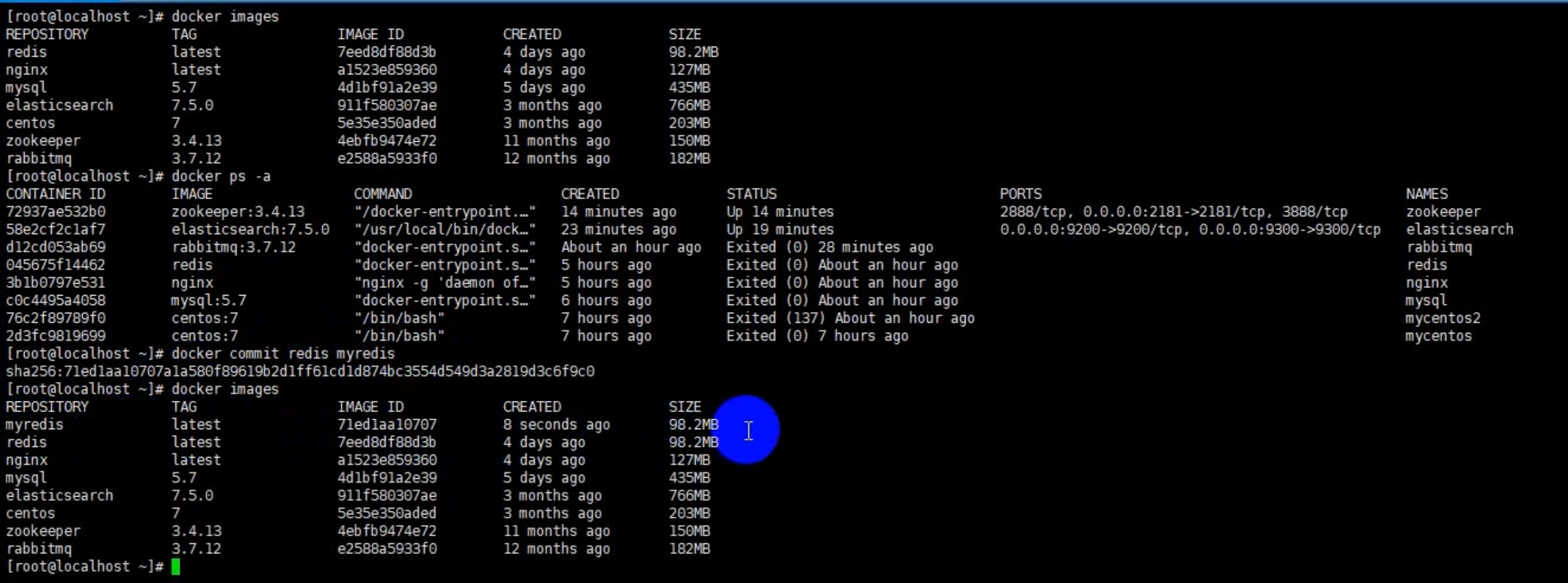




# Zookeeper



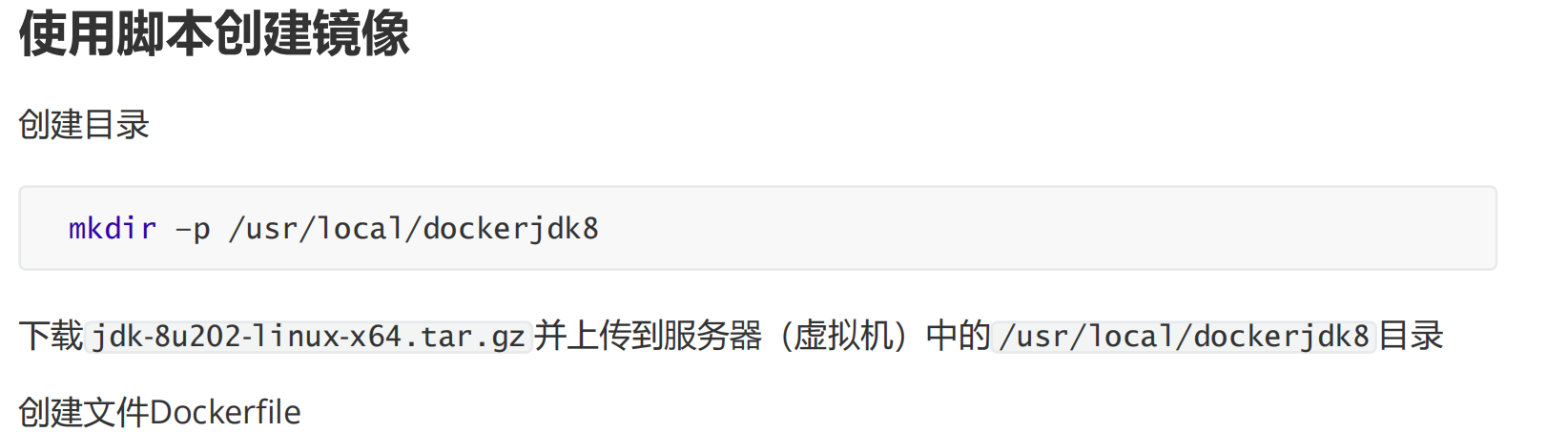
# 镜像保存





# Dockerfile





vim Dockerfile

#依赖镜像名称和ID

FROM centos:7

#指定镜像创建者信息

MAINTAINER xxxx

#切换工作目录

WORKDIR /usr

RUN mkdir /usr/local/java

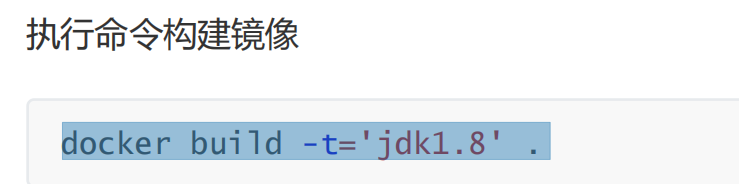
#ADD 是相对路径jar,把java添加到容器中

ADD jdk-8u202-linux-x64.tar.gz /usr/local/java/

#配置java环境变量

ENV JAVA\_HOME /usr/local/java/jdk1.8.0\_202

ENV PATH $JAVA\_HOME/bin:$PATH



# Docker 私有仓库





# Maven docker 插件

通过maven 将项目打包 生成镜像 上传到docker 私有仓库

本质上还是使用dockerfile

**DockerMaven插件**

微服务部署有两种方法：

手动部署：首先基于源码打包生成jar包（或war包）,将jar包（或war包）上传至虚拟机并拷贝至

JDK容器。

通过Maven插件自动部署。

对于数量众多的微服务，手动部署无疑是非常麻烦的做法，并且容易出错。所以我们这里学习如何自动

部署，这也是企业实际开发中经常使用的方法。

Maven插件自动部署步骤：

**修改宿主机的docker配置，让其可以远程访问**

修改以 ExecStart 开头的行

修改后如下：

{"insecure-registries":["192.168.10.101:5000"]}

systemctl restart docker

docker tag jdk1.8 192.168.10.101:5000/jdk1.8

docker start registry

docker push 192.168.10.101:5000/jdk1.8

vim /lib/systemd/system/docker.service

ExecStart=

ExecStart=/usr/bin/dockerd -H fd:// -H tcp://0.0.0.0:2375**刷新配置，重启服务**

**在工程pom.xml 增加配置**

systemctl daemon-reload

systemctl restart docker

docker start registry

<!--docker的maven插件，官网：https://github.com/spotify/docker-maven-plugin-->

<plugin>

<groupId>com.spotify</groupId>

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

<version>1.1.0</version>

<executions>

<execution>

<id>build-image</id>

<phase>package</phase>

<goals>

<goal>build</goal>

</goals>

</execution>

</executions>

<configuration>

<imageName>ego/${project.artifactId}:${project.version}</imageName>

<dockerHost>http://192.168.10.101:2375</dockerHost>

<baseImage>java:8</baseImage>