# Docker学习

# Docker

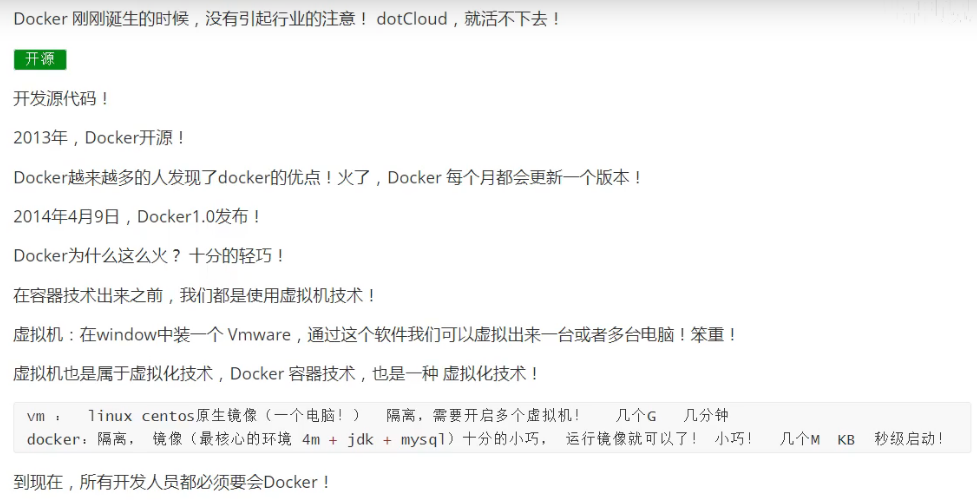
Docker的思想来源于集装箱。

隔离：Docker的核心思想！打包装箱，每个箱子都是相互隔离的，可以将服务利用到极致。

开发----运维的关系，我在我的电脑上可以运行，版本更新导致服务不可用，环境配置十分麻烦。

解决：软件可以带环境安装？也就是说，安装的时候，把原始环境一模一样地复制过来。

# Docker的历史



基于go语言开发的，开源项目。

官网：<https://www.docker.com/> 文档：<https://docs.docker.com/>

# Docker能干什么

虚拟技术：占资源，冗余，步骤多，启动慢。

容器化技术：容器技术不是模拟一个完整的操作系统，一个容器等于：应用加运行环境。容器没有内核，也没有虚拟我们的硬件。轻便。每个容器互相隔离，互不影响、

Docker；一键运行打包测试。在使用容器后，我们的开大测试环境都是高度一致的

## 镜像（image）：

Docker镜像好比一个模板，可以通过这个模板创建镜像服务，

Tomcat镜像->run->tomcat01(容器)提供服务

## 容器：

Docker·运用容器技术，独立运行一个或一组应用，通过镜像来创建，

启动，停止，删除，基本命令。

目前可以把容器理解为一个简易的linux系统()。容器里相互隔离。

## 仓库

存放镜像，仓库分为共有仓库，私有仓库。

## 安装

帮助文档：百度

## Docker常用命令

Docker version #显示版本信息

### 镜像命令：

Docker images #查看本地所有的镜像

Docker search # 搜索镜像

Docker pull # 拉取镜像

Docker rmi image id # 删除镜像

docker inspect 查看镜像详情信息

#查看latest镜像的具体版本 latest表示最新版本的镜像，再次pull 回更新至最新版本

docker image inspect mysql:latest | grep -i version

### 容器命令

Docker run # 运行容器

Docker run –it centos /bin/bash

Exit # 退出容器，容器停止

Ctrl+p+q # 退出容器，容器不停止

Docker run –d 容器名称 # 以守护进程运行

docker run -d -p 3344:80 --name nginx02 nginx # 开启容器，并映射外网地址3344，名称nginx02

Docker ps # 查看正在运行的容器 -a显示所有运行过的容器（状态已退出，正在运行）

删除容器docker rm

docker rm -f $(docker ps -aq) # 删除所有的容器

启动停止容器：

Docker start

Docker restart

Docker stop

Docker kill



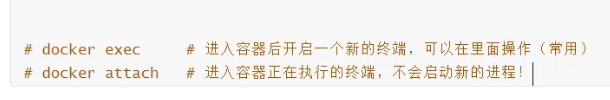
Docker inspect 容器id #查看镜像的元数据

Docker top 容器id # 查看容器进程

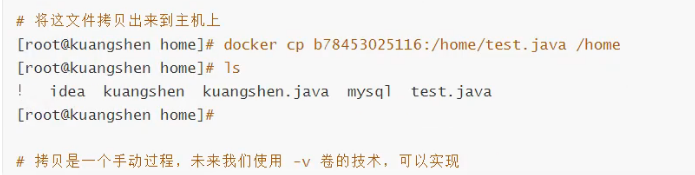
# 进入当前正在运行的容器：

方式一：Docker exec -it 容器id /bin/bash

方式二：docker attach 容器id



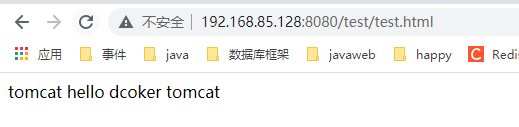
# 把b78453025116容器里 test.java 文件拷贝到 主机/home里



### 部署Tomcat;

1,拉取Tomcat镜像，运行

2，在webapps文件夹下创建test/test.html

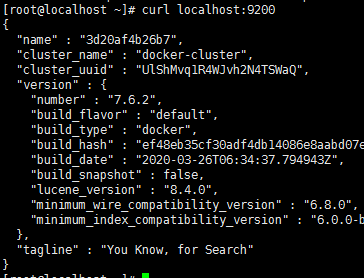
3, 访问：

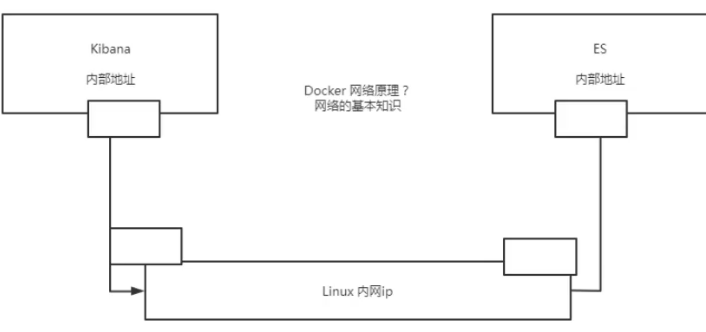
Docker stats 查看docker内存情况

### 安装elasticSearch，kibana

<https://www.jianshu.com/p/fdfead5acc23>

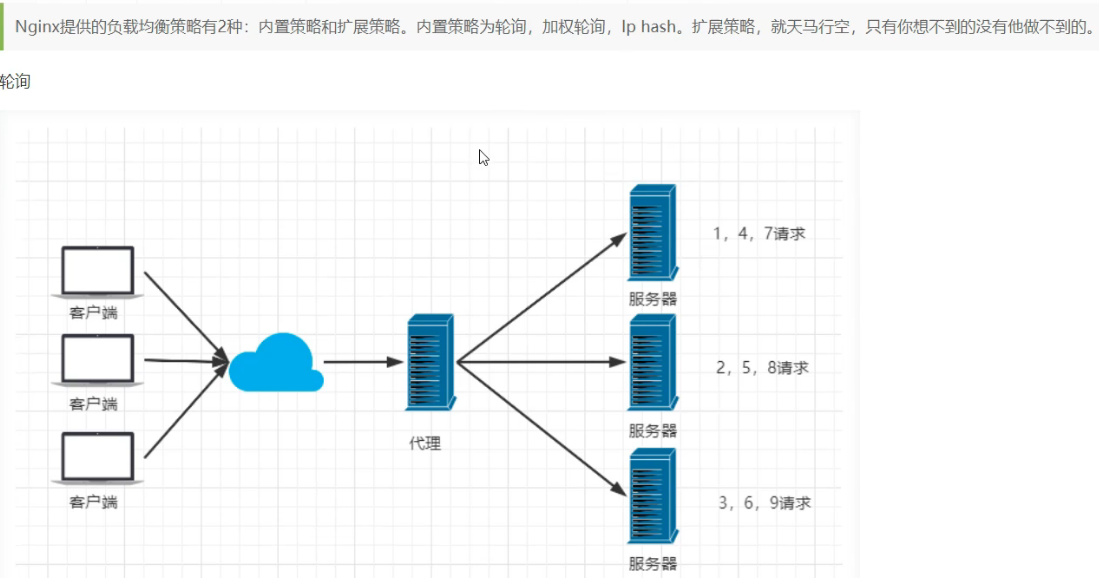






### 安装nginx:

Nginx (engine x) 是一个高性能(并发能力高，4~5万的并发，tomcat只有200~400并发)的HTTP和反向代理（代理服务端）web服务器，同时也提供了IMAP/POP3/SMTP服务

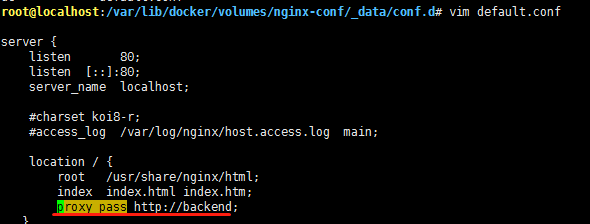


<https://www.cnblogs.com/qiqiloved/p/13470064.html>

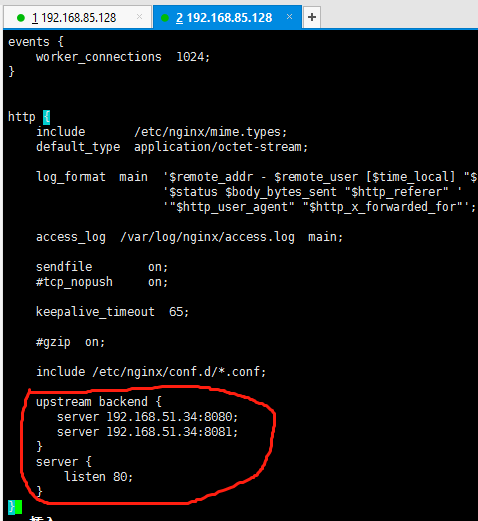
在配置nginx负载均衡的时候有一个大坑：

配置文件docker被分为两部分：nginx.conf 和 default.conf, default.conf,文件包裹nginx.conf

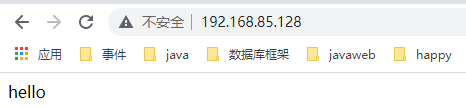
default.conf配置如下：



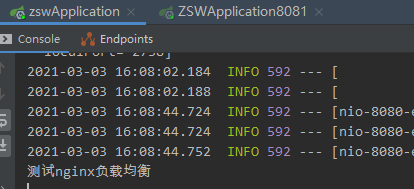
nginx.conf说加配置如下：

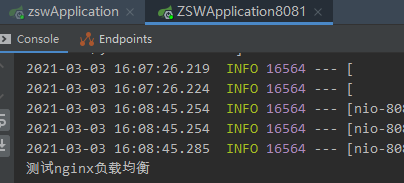


192.168.51.34为我的电脑ip，我通过电脑<http://192.168.85.128:80>（宿主机ip） 访问nginx，nginx负载均衡到192.168.51.34:8080和192.168.51.34:8081



我本地开启两个实例8080和8081





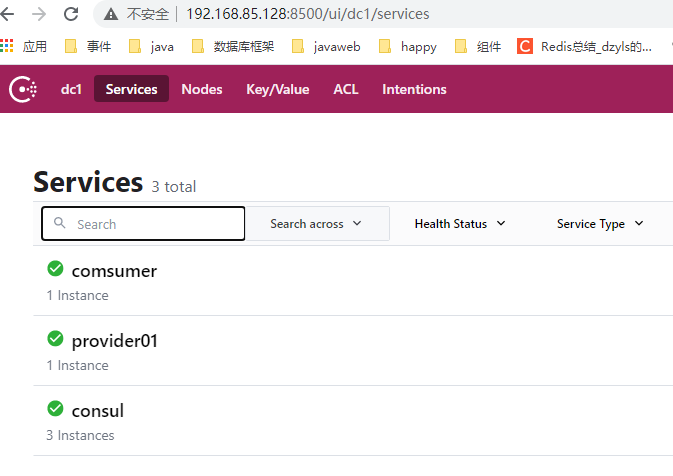
### docker 搭建consul 集群：

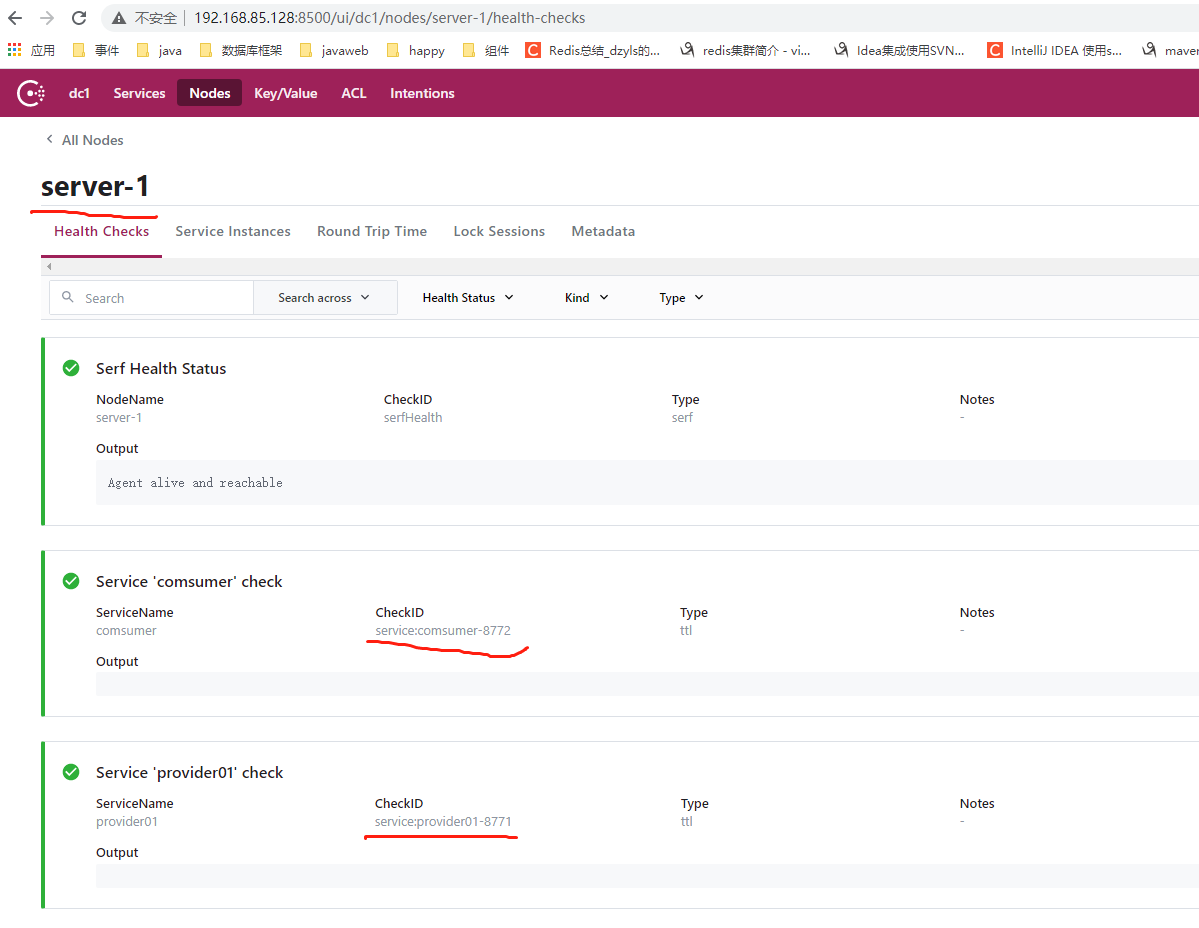
docker脚本：

搭建docker 脚本如下：  
docker run --name consul1 -d -v /home/consul/data:/consul/data -v /home/consul/config:/consul/config -p 8500:8500 consul agent -node server-1 -server -bootstrap-expect 2 -ui -bind=0.0.0.0 -client=0.0.0.0  
  
docker inspect --format='{{.NetworkSettings.IPAddress}}' consul1 #查看端口  
  
docker run --name consul2 -d -v /home/consul/data:/consul/data2 -v /home/consul/config:/consul/config2 -p 8501:8500 consul agent -node server-2 -server -ui -bind=0.0.0.0 -client=0.0.0.0 -join 172.18.0.6

docker run --name consul3 -d -v /home/consul/data:/consul/data3 -v /home/consul/config:/consul/config3 -p 8502:8500 consul agent -node server-3 -server -ui -bind=0.0.0.0 -client=0.0.0.0 -join 172.18.0.6

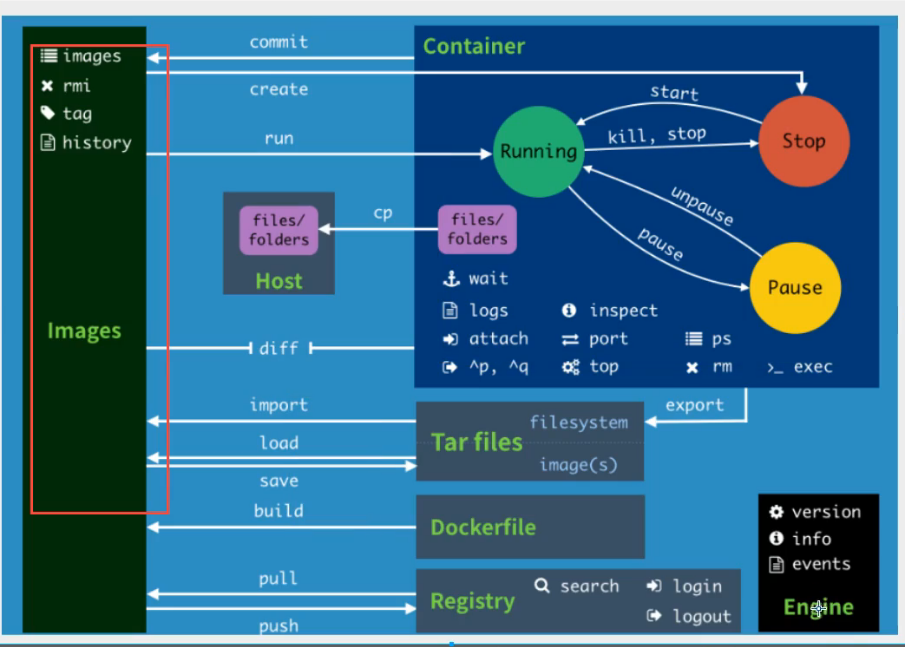
docker run --name consul5 -d -v /home/consul/data:/consul/data5 -v /home/consul/config:/consul/config5 -p 8505:8500 consul agent -node client-1 -client -ui -bind=0.0.0.0 -client=0.0.0.0 -join 172.18.0.6  
  
docker exec -it consul1 consul members # 查看集群信息  
  
项目启动访问：http://192.168.85.128:8500 默认 consul1主节点  
  
坑点：注意join 加入节点的ip地址。-bootstrap-expect 2 这个只能在默认主节点加，推举自己为主节点要先启动。其他服务挂载路径不能一样，要不然会无法搭建集群， 启动一个client个 3个server端



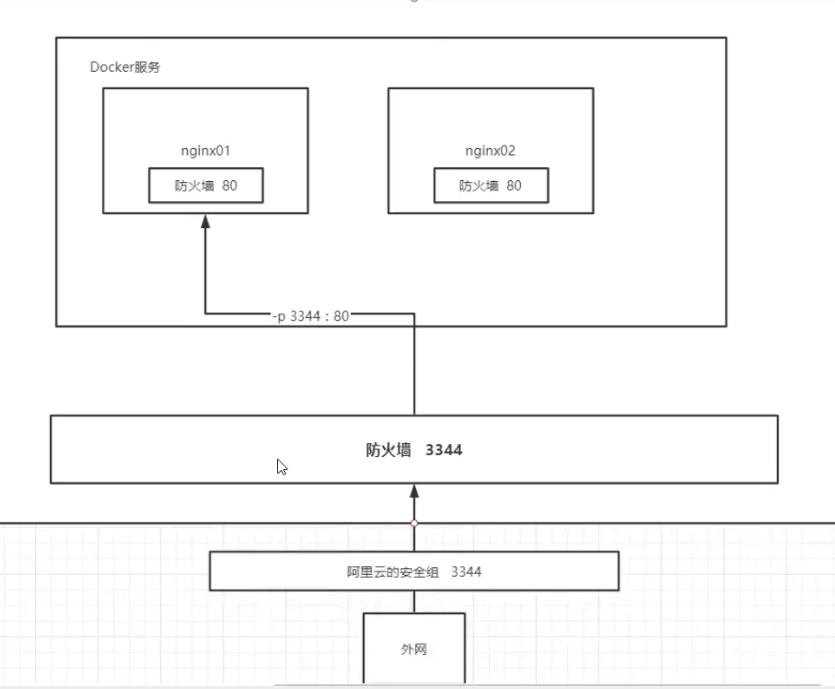


8772（服务消费者） 和 8771（服务提供者）注册到 server1 服务中，consul 下3个server。1 个 client

### 命令总结：



### 端口暴露的概念：



### 可视化

Portainer:docker 的界面图形化工具

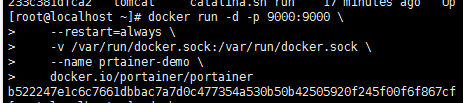
docker run -d -p 9000:9000 \

--restart=always \

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

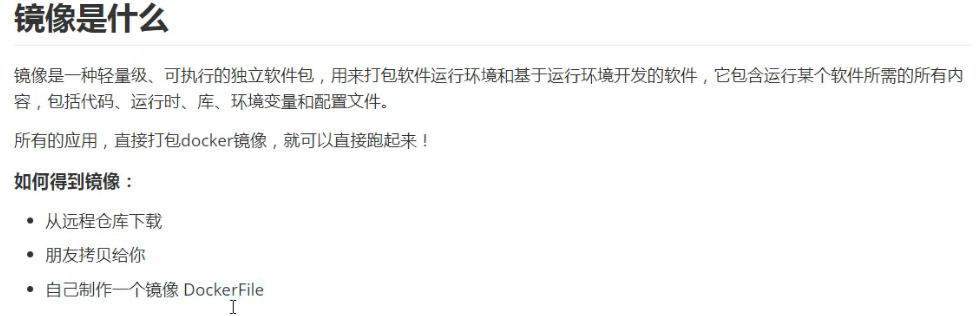
--name prtainer-demo \

docker.io/portainer/portainer





访问方式：[http://127.0.0.1:9000](http://127.0.0.1:9000/)，用户名admin 密码12345678

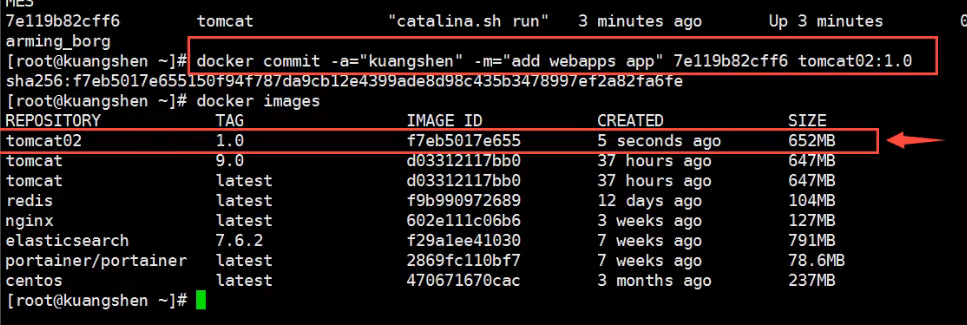


### 镜像加载原理，分层的理解：

## Commit镜像

Docker commit 提交一个容器成为一个新的副本(本地镜像)

Docker commit –m =“提交的描述信息” –a =“作者” 容器id 目标镜像名:tag



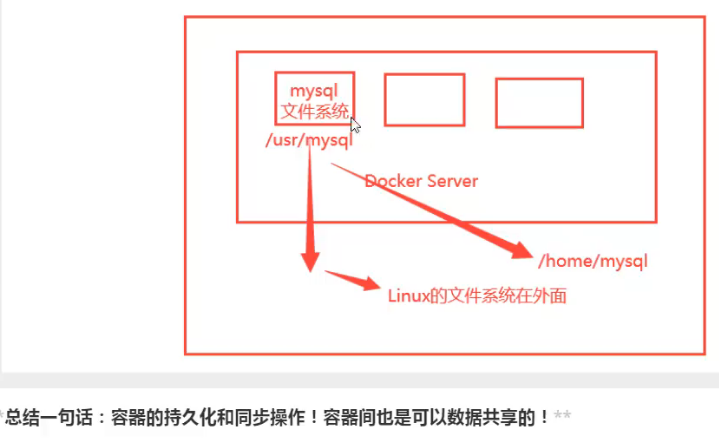
## 容器数据卷：

Docker的理念回顾，将应用和环境打包成一个镜像；

数据？数据存在容器中，那么我们容器删除，数据也会删除，这不是我们想要的、需求：数据持久化.

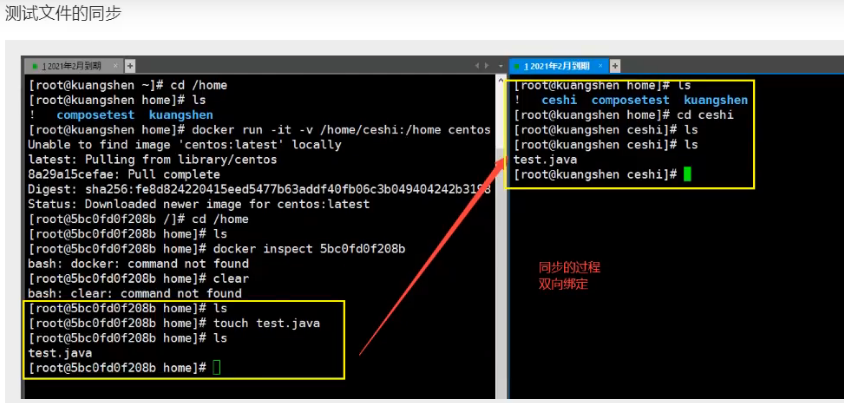
容器之间可以有一个数据共享的技术,docker容器中产生的数据,同步到本地.

这就是卷技术,!目录的挂载,将我们的容器目录,挂载到liunx上面.



# 使用v命令把容器内的目录和linux目录挂载一起

Docker run –it –v /home/ceshi:/home centos /bin/bash



可以把容器内的配置同步到服务器外。方便修改配置。还有日志之类的

### Mysql数据同步

Docker pull mysql5.7 # 拉取镜像

Docker images

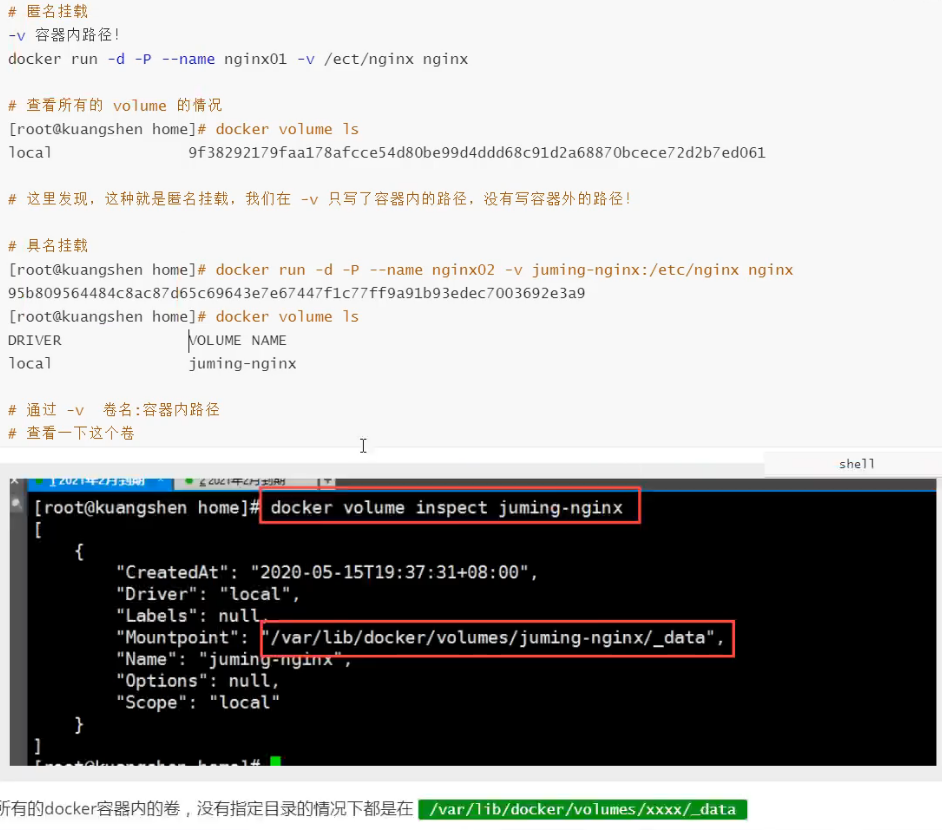


把mysql配置和数据挂载到外面。

假设我们删除了MySQL容器，我们挂载到本地的数据卷依然有丢失，这就实现了容器数据持久功能。

### 匿名挂载和具名挂载：

docker 容器内 vim 命令没有下载



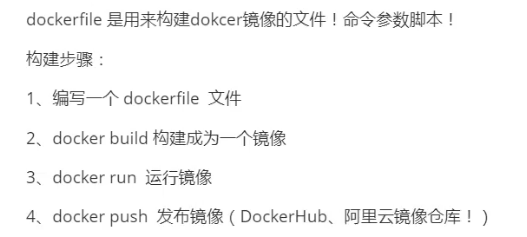
一般使用具名挂载，指定名称（不指定路径，存在docker var/lib/docker/volumes/下面）

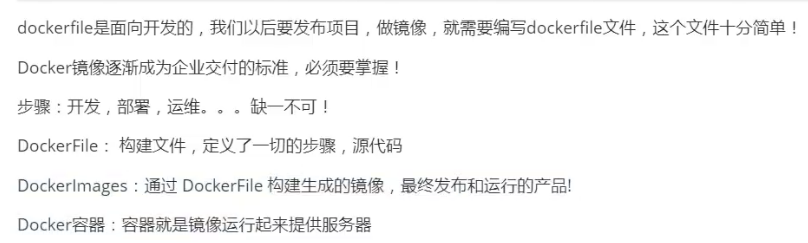
-d 后台运行，-p端口映射 – name 名称 –v 挂载 tomcat-volume 为linux目录 ，etc/tomcat 为容器目录，经过测试发现，如果容器没有这个/etc/tomcat这个目录，不会报错，挂载的目录为空

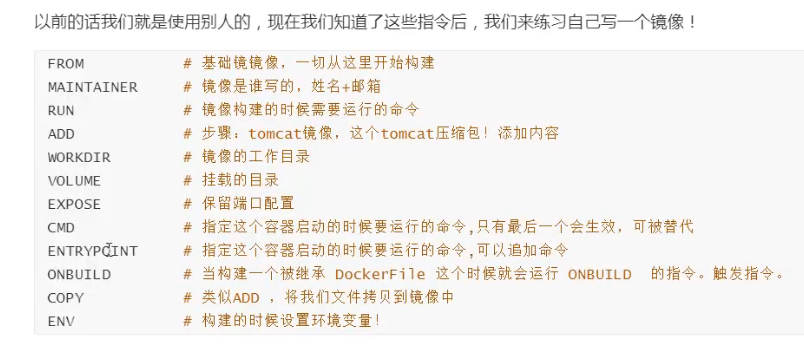
docker run -d -p 8080:8080 --name tomcat01 -v tomcat-volume:/etc/tomcat tomcat



### DockerFile技术和commit一样，可以构建自己的新的镜像









### 发布自己的镜像

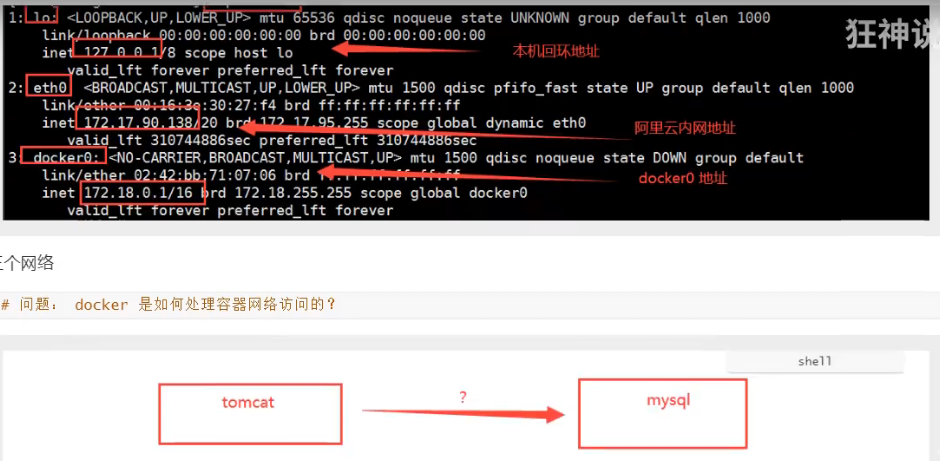
#发布到dockerhub

Docker login –u 用户名

Docker push 镜像名版本号

#发布到阿里云

## Docker网络：（待学习）



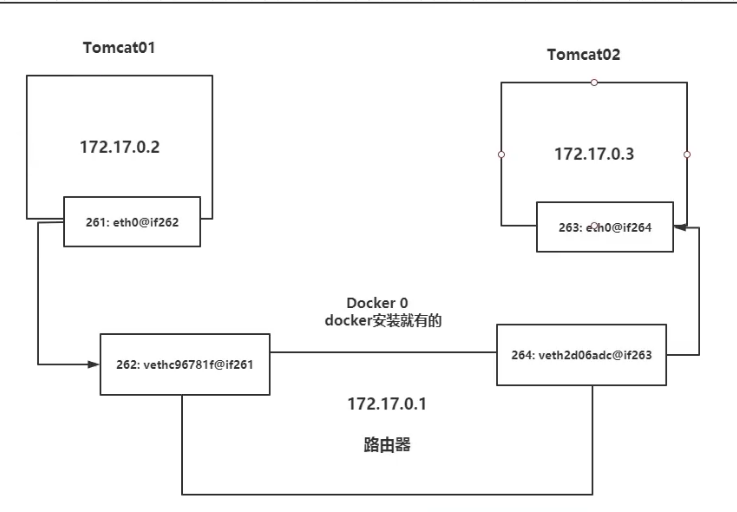
查看容器ip地址ip addr

在linux里可以ping通容器ip，

原理：evth-pair 技术

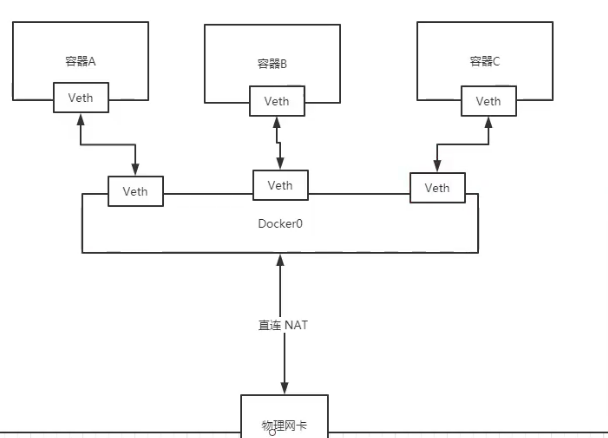
1. 我们每启动一个docker容器，docker就会给docker容器分配一个ip，（这个ip会使用ecth-pair技术和linux外进行绑定，）我们只要安装了docker，就会有一个网卡docker0





结论：

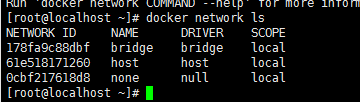
Tomcat01和Tomcat02是公用的路由器docker0，所有的容器在不指定网络的情况下都是docker路由的吗，docker会给我们容器分配一个默认的可用ip



### --link

通过服务名ping,万一有个服务挂了，重启的话，ip地址可能会变，通过服务名访问就不用担心这个问题。（现在不建议使用）

### 自定义网络



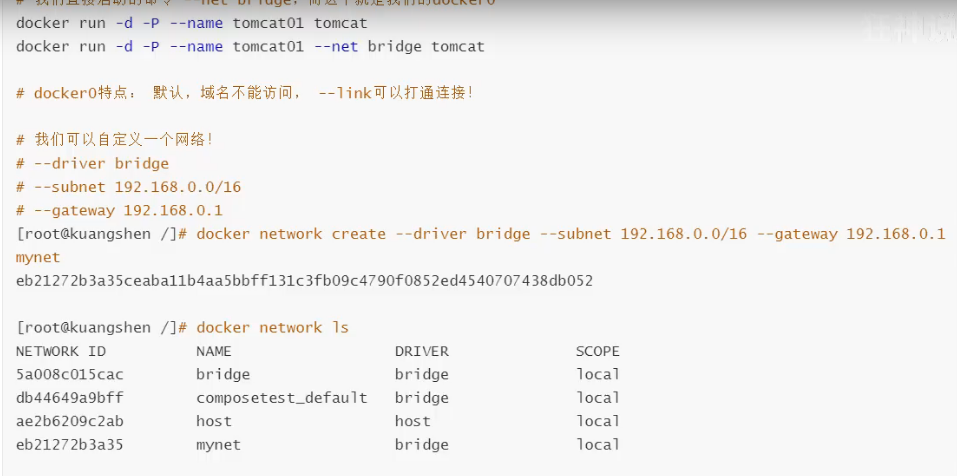
网络模式：

Bridge 桥接（默认），比如Tomcat01 和Tomcat02 通过docker桥接，可以互相访问

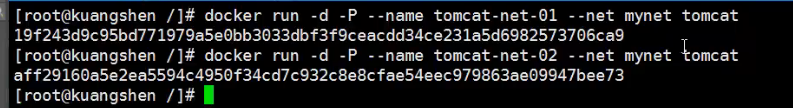
None:不配置网络

Host:和宿主机共享网络

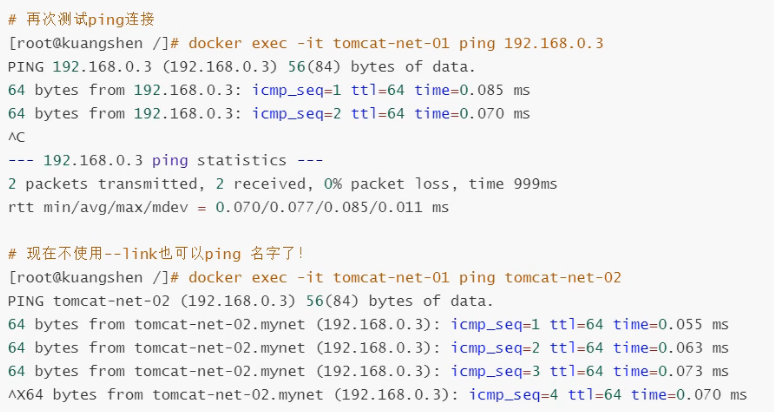
创建一个自定义网络mynet:



通过 –-net指定网络，走我们自定义的mynet网络,tomcat-net-01 和Tomcat-net-02 在mynet网络下



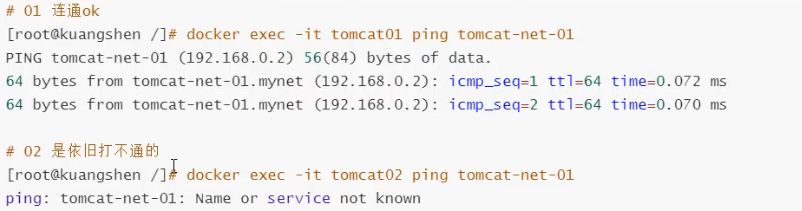
接下来可以通过名称来访问了



### 网络连通：



将Tomcat01容器和mynet网络打通



结论：假设要跨网络操作别人，就需要使用docker network connect 连通

### Redis集群部署

1，创建6个redis容器，运行容器

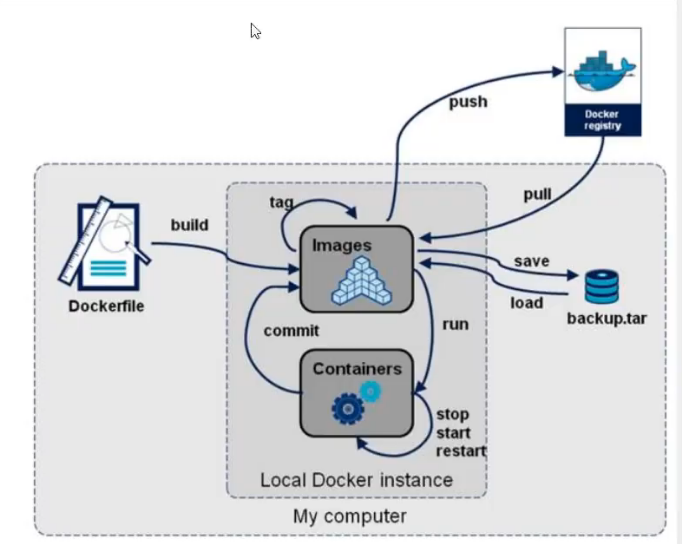
2，创建集群



主节点挂掉，从节点成为主节点



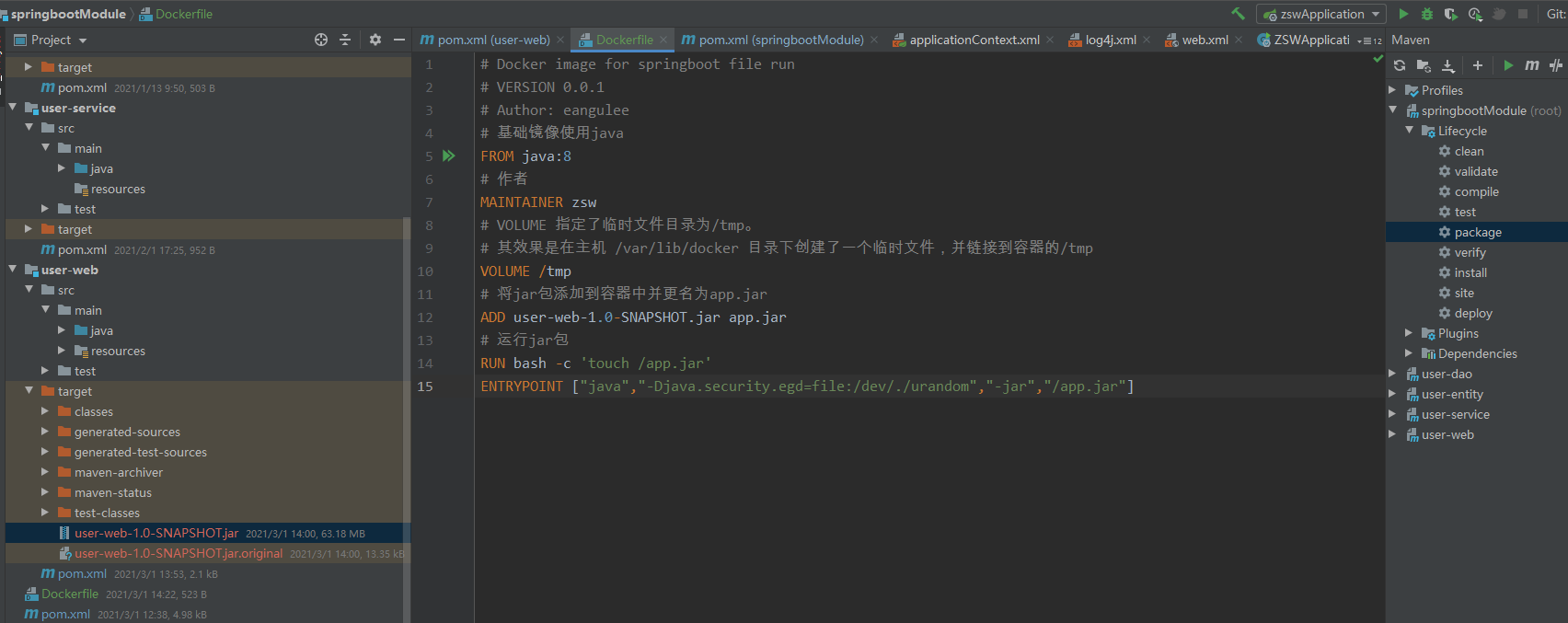
## 小结：



## Springboot 微服务项目打包docker镜像

1. 构建springboot微服务项目
2. 打包项目项目
3. 直接在idea上打包docker 文件 （这种报错111）
4. 在idea上 以 maven 打包 , 在去linux上docker 运行（这种可行）

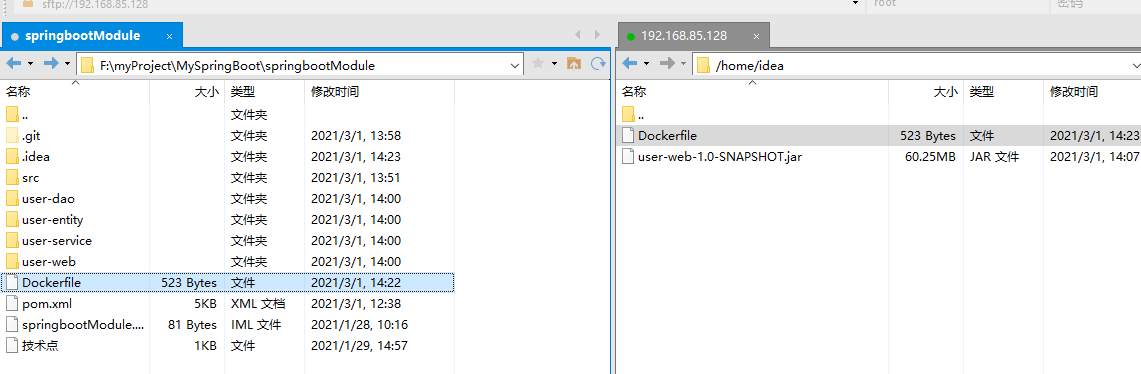
user-web-1.0-SNAPSHOT.jar为项目打包的jar包

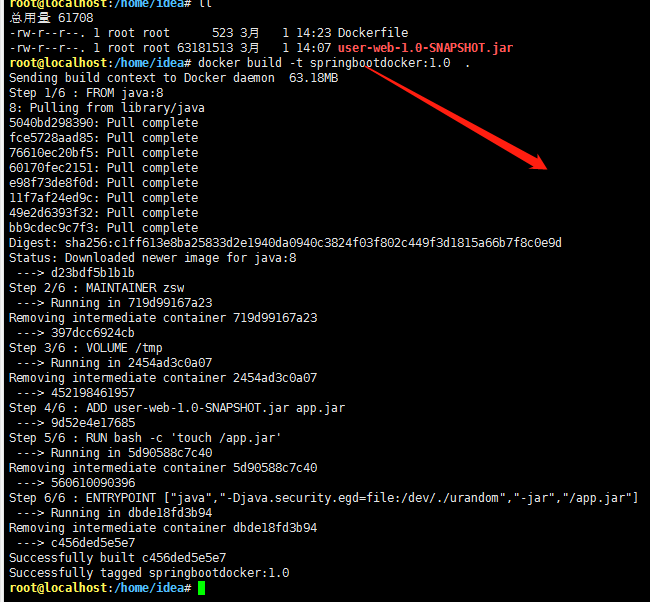


1. 编写docker文件



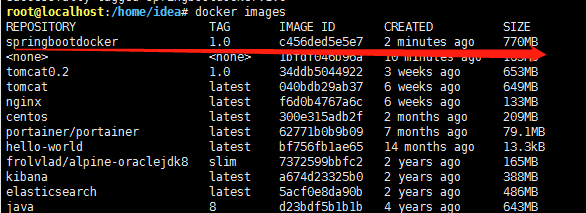
上传到linux:





springbootdocker:1.0 名字只能为小写字母

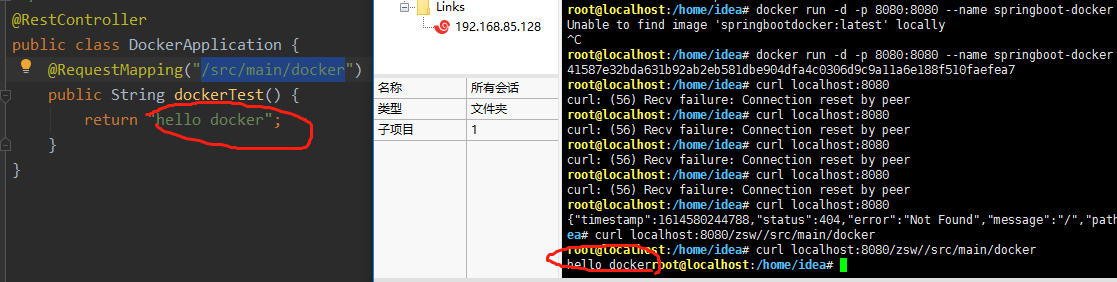
1. 构建镜像



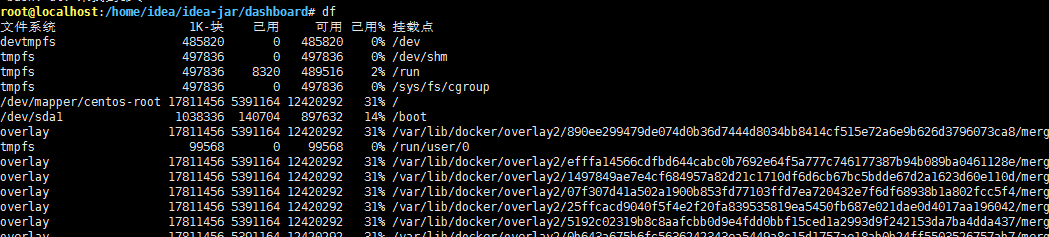


1. 发布运行





1. 查看linux磁盘使用情况：df



1. 查看cpu内存使用 top 或者 free命令

