## 1.什么是Docker？

Java：一次编译，到处运行在JVM虚拟机上;

Docker：真正实现了一次编译,到处运行;

Docker是一个开源的应用容器引擎,基于GO语言;

Docker可以让开发者打包他们的应用到一个轻量级、可移植的容器中,然后发布到任何流行的Linux机器上,也可以实现虚拟化;

容器完全使用沙箱机制，互相之间不会有任何接口(类似手机中的App),更重要的是容器性能开销极低;

## 2.为什么要用Docker？

一个项目开发、上线需要经历多套环境的配置,每一套机器都需要一套环境，而Docker 就可以把环境一起打包，安装，部署；

Docker支持将软件编译成一个镜像;然后在镜像中各种软件做好配置，将镜像发布出去， 其他使用者可以直接使用这个镜像。运行中的这个镜像成为容器,容器的启动速度是非常快速的；

1.作为一种新兴的虚拟化方式,Docker和传统的虚拟化方式相比有众多的优势;

### a.更高效利用系统资源;

由于容器不需要进行硬件虚拟以及运行操作系统等额外开销，Docker对系统资源的利用率更高。无论是应用执行速度、内存损耗或者文件存储速度，都要比传统虚拟机技术更高效。因此，相比虚拟机技术，一个相同配置的主机,往往可以运行更多数量的应用。

### b.更快的启动时间;

传统的虚拟机技术启动应用服务往往需要数分钟，而Docker容器应用，由于直接运行在宿主内核，无需启动完整的操作系统,因此可以做到秒级、甚至毫秒级的启动时间。大大节约了开发、测试、部署时间;

### c.一致的运行环境;

开发中一个常见的问题是环境一致性问题。由于开发环境、测试环境、生产环境不一致，导致有些bug并未在开发过程中被发现。而Docker的镜像提供了除内核外完整的运行时环境，确保了应用环境一致性。

### d.持续交付和部署;

对开发和维护人员来说，最希望的是一次创建或配置，可以在任意地方正常运行，使用Docker可以通过定制应用镜像来实现持续集成、持续交付、部署。开发任意可以通过Dockerfile来进行镜像构建，并结合集成系统进行集成测试，而运维人员可以直接在生产环境中快速部署改镜像，甚至结合持续部署系统进行自动部署。而且使用Dockerfile使镜像构建透明化，不仅仅开发团队可以理解应用运行环境，也方便运维团队理解应用所运行所需要的条件，帮助更好的在生产环境中部署该镜像。

### e.更轻松的迁移;

由于Docker确保了执行环境的一致性，使得应用的迁移更加容易。Docker可以在很果断平台上运行，无论是物理机、虚拟机、公有云、私有云，甚至笔记本，其运行结果是一致的，因此用户可以轻易的将一个平台上的应用,迁移到另外一个平台上，而不用担心运行环境的变化导致应用无法正常运行。

### f.更加轻松的维护和扩展;

Docker使用了分层存储以及镜像技术,使得应用重复部分的复用更为容易，也使得应用的维护更新更加的简单,基于基础镜像进一步扩展镜像也非常简单。此外，Docker团队和各个开源项目团队维护了一大批高质量的官方镜像,既可以直接在生产环境中使用,也可以作为基础进一步定制,大大降低了应用服务镜像制作成本;

## 3.Docker的应用场景

### 1.应用程序的打包和发布

### 2.应用程序隔离

### 3.自动化测试、持久集成、发布

### 4.部署微服务

### 5.快速搭建测试环境

### 6.提供PaaS平台级产品

## 4.Docker和虚拟机的区别

**1.物理机模型:**

相当于一栋楼一户人家，独立地基，独立花园;

**2.虚拟机模型：**

相当于一栋楼包含了多套房，一套房一户人家，共享基地，共享花园，独立卫生间，厨房、宽带;

**3.Docker 模型:**

一套房隔成多个隔间,每个隔间一个租户,共享基地，共享花园，共享卫生间，厨房、宽带;

## 5. Docker的核心概念

a.**Docker主机(host)**:安装了docker程序的机器(Docker直接安装在操作系统之上)  
b**.Docker客户端（client）**：连接docker主机进行操作;  
c.**Docker仓库(registry)**：用来保存各种打包好的软件镜像;  
d.**Docker镜像(Images)** :软件打包好的镜像,放在docker仓库;  
e.**Docker容器(container)**：镜像启动后的实例称为一个容器,容器是独立运行的一个或一组应用;

## 6.使用Docker的步骤

1.安装Docker;  
2.去Docker仓库找到需要安装的软件对应的镜像;  
3.使用Docker运行该软件的镜像,这个镜像镜像会产生一个Docker容器,容器就代表这个当前正在运行的应用;  
4.对容器的启动停止就是对软件的启动停止;

## 7.在Linux（cetos7）上安装Docker

1.查看Linux系统的内核版本

必须是3.10及以上

**uname -r**

2.升级软件包及内核;(选做)

**yum update**

3.安装 docker

**yum install docker**

查看docker版本号

**docker -v**

4.启动docker

**systemctl start docker**

5.将docker服务设为开机启动

**systemctl enable docker**

## 7.2 [使用 DockerHub 加速器](https://cloud.tencent.com/document/product/457/9113)

1. 腾讯云为您提供如下 DockerHub 加速器

https://mirror.ccs.tencentyun.com

2. Linux实例配置

1. 创建或修改 /etc/docker/daemon.json 文件，并写入以下内容：

{

"registry-mirrors": [

"https://mirror.ccs.tencentyun.com"

]

}

2. 依次执行以下命令，重新启动 Docker 服务。

$ sudo systemctl daemon-reload

$ sudo systemctl restart docker

# Ubuntu16.04 请执行 sudo systemctl restart dockerd 命令

3.检查加速器是否生效

执行 docker info 命令，返回结果中包含以下内容，则说明配置成功。

Registry Mirrors:

https://mirror.ccs.tencentyun.com

容器服务相关文档：<https://cloud.tencent.com/document/product/457/9113>

## 8.Docker相关命令

1.停止Docker命令

**systemctl stop docker**

## 9. Docker镜像操作常用命令

**1.检索**

命令：docker search 关键字

eg:docker search redis 说明：我们经常去docker hub上检索镜像的详细信息,如镜像的Tag;

默认去 docker hub 仓库搜索镜像; <https://hub.docker.com/>

**2.拉取**

命令：docker pull 镜像名:tag

说明：:tag是可选的,tag表示标签，多为软件的版本,默认为latest

**3.列表**

命令：docker images

说明：查看所有的本地镜像

**4.删除**

命令：docker rmi image-id

说明：删除指定的本地镜像

## 10. Docker的容器操作

软件镜像–运行镜像—产生一个容器（正在运行的软件）

**a.搜索镜像**

docker search tomcat

**b.拉取镜像**

docker pull tomcat

**c.根据镜像启动容器**

docker run --name mytomacat -d tomcat:latest

**1.运行**

命令：docker run --name container-name -d image-name

eg: docker run --name myredis -d redis

说明：–name:自定义容器名;

-d:后台运行;

image-name:指定镜像模板;

**2.列表**

命令：docker ps(查看运行中的容器) 说明：加上-a 可以查看所有容器

**3.停止：**

命令：docker stop container-name/container-id 说明：停止当前运动的容器

**4：启动**

命令： docker start container-name/container-id

说明：启动容器

**5 删除**

命令：docker rm container-id

说明：删除指定容器

**6端口映射:**

命令：-p 6379:6379 主机端口：容器内部端口

eg: docker run -d -p 6379:6379 -name myredis docker.io/redis

说明：-p:主机端口(映射到)容器内部的端口

**7 容器日志：**

命名： docker logs -f container-name/container-id

-f动态查询日志

**8.进入容器：**

docker exec -it container-id /bin/bash

## 11.容器操作相关操作说明

**docker run**

-i:表示运行容器

-t:表示容器启动后会进入命令行,加上这个参数后，容器创建就能登录进云，及分配了一个伪终端.

交互式容器：docker run -it --name -mycentos centos:7 /bin/bash

该类型容器退出容器之后容器就停止.

守护式容器：docker run -di --name -mycentos2 centos:7

登录守护式容器:cocker exec -it 容器名称 /bin/bash

该类型的容器退出后不会停止容器；

-d:创建一个守护式容器在后台一直运行

-p:端口映射 指定容器暴露的端口 9000:9000 （前者是宿主机的端口:后者是容器的端口）

--restart:指定容器重启后的策略

no:容器退出不重启;

no-failure:容器故障退出（返回值非零）时重启；

always:容器退出总是重启；

-v：给容器挂载存储卷，挂载到容器的某个目录 宿主机目录：容器目录

--name:指定容器名字,后续可以通过名字进行容器管理，links特性需要使用名字

--rm=false,指定容器停止后自动删除容器(不支持以docker run -d启动的容器)

为什么要做目录映射：因为容器里的系统不能直接去上传文件;

相关文档：<https://docs.docker.com/engine/reference/commandline/run/>

## 11.1 使用docker安装时可能会用到的Docker相关命令

**1.容器与主机之间的数据拷贝**

**docker cp [options] container : src\_path dest\_path**

**docker cp 容器名称或id ：容器目录或文件 宿主机目录或文件**

docker cp c10d0b833743:/etc/nginx/nginx.conf ./

复制c10d0b833743容器/etc/nginx/nginx.conf目录下的文件当当前目录

## 12.安装DockerUI：portainer

1.查看DockerUI:portainer镜像;

docker search portainer

2.拉取portainer镜像

docker pull docker.io/portainer/portainer

3.启动portainer

docker run -d -p 9000:9000 --restart=always -v /var/run/docker.sock:/var/run/docker.sock --name ben-portainer portainer/portainer

## 12.1安装Nginx

1.创建挂载目录(不一定要在etc目录下可以在自己想挂载的目录下)

1.在主机/etc目录下执行 mkdir -p ./nginx/{html,log,conf.d}创建挂载目录;

2.运行nginx

docker run -d --name nginx -p 8081:80 nginx

缺点:修改配置文件需要进入容器修改;

3. 将nginx容器内部配置文件挂载到主机

1.进入主机/etc/nginx挂载目录复制容器配置文件(不一定是在/etc目录下)

**docker cp 容器名称或id:容器路径 宿主机路径**

docker cp c10d0b833743:/etc/nginx/nginx.conf ./

docker cp c10d0b833743:/etc/nginx/conf.d/default.conf ./conf.d/

c10d0b833743：nginx的容器ID

4.移除运行中的容器(如果需要相同端口)

docker rm c10d0b833743

4.运行容器

docker run -d --name test-nginx -p 8082:80

-v /etc/nginx/nginx.conf:/etc/nginx/nginx.conf

-v /etc/nginx/log:/var/log/nginx

-v /etc/nginx/html:/usr/share/nginx/html

-v /etc/nginx/conf.d:/etc/nginx/conf.d --privileged=true nginx

**将nginx容器内部配置文件挂载到主机后,运行镜像是提示：container init exited prematurely ；**

因为文件挂载时，需要提前准备，所以要先把文件放在指定挂载位置，才能正常启动

相关资料：

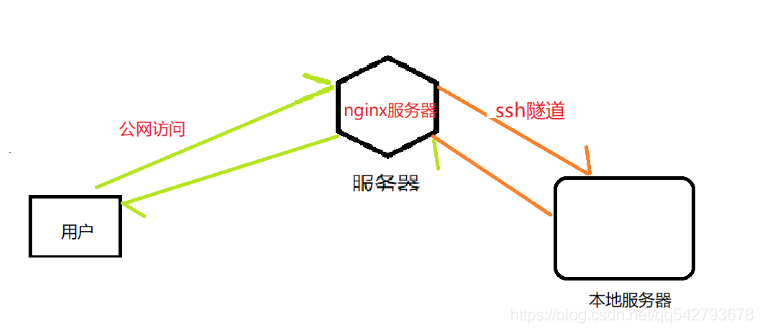
<https://www.cnblogs.com/qiqiloved/p/13470064.html>

<https://blog.csdn.net/weixin_45784642/article/details/102932905>

**配置云服务器centeros7，nginx反向代理+ssh实现内网穿透**

内网穿透:

使用ip地址或域名通过公网访问服务器服务器反向代理转发到指向到本地的服务器，这样就可以实现通过公网ip或域名访问本地的应用了;



**内网穿透相关资料:**

[**http://service.oray.com/question/5571.html**](http://service.oray.com/question/5571.html)

[**https://blog.csdn.net/qq542793678/article/details/89635824**](https://blog.csdn.net/qq542793678/article/details/89635824)

**实现：**

1. 需要1台公网服务器;
2. 最好有一个域名，这样方便一点，没有域名也行;
3. 安装Nginx
4. 配置Nginx
5. 搭建SSH通道

SSH是一种网络协议，用于计算机之间的加密登录。如果一个用户从本地计算机，使用SSH协议登录另一台远程计算机，我们就可以认为，这种登录是安全的，即使被中途截获，密码也不会泄露。

## 13.Docker安装软件及使用中遇到的问题

### 1. 腾讯云CentOS安装软件, 外部无法访问;

解决方案:

a.腾讯云默认情况下只允许ssh的22端口通过，其余端口一律封锁,解决方法就是进到腾讯云控制台：选择安全组：开放端口;

相关文档: <https://blog.csdn.net/weixin_38423249/article/details/80554583>

b.放开端口

1.添加指定需要开放的端口：

**firewall-cmd --add-port=9000/tcp --permanent**

2.重载入添加的端口：

**firewall-cmd --reload**

3.查询指定端口是否开启成功：

**firewall-cmd --query-port=123/tcp**

相关文档:

[https://blog.csdn.net/lazycheerup/article/details/90238615](%20%20https://blog.csdn.net/lazycheerup/article/details/90238615)

<https://blog.csdn.net/weixin_43257494/article/details/87927020>

### 2.安装tomcat访问主页404;

a.解决方案：

访问不到主页的原因是因为存放主页的文件夹webapps是空的,docker的tomcat将这些文件放在了 webapps.dist文件夹中,我们只需要将空的webapps文件夹删除,然后将ebapps.dist重命名为webapps就可以了;

b.相关文档:

<https://www.cnblogs.com/chzushl/p/13385580.html>

### 3.Docker插件连接不到Docker

1.检查Docker引擎已开启远程访问;

2.检查腾讯云安全组开启了2375端口;

3.检查已开放2375 端口

firewall-cmd --add-port=2375/tcp –permanent

firewall-cmd –reload //重启防火墙

防火墙相关操作：

<https://blog.csdn.net/cool_summer_moon/article/details/78744009>

### 4.运行容器时提示：Error response from daemon: driver failed programming external connectivity on endpoint …….

**Why:**

docker服务启动时定义的自定义链DOCKER由于某种原因被清掉  
重启docker服务及可重新生成自定义链DOCKER;

解决方案：重启docker

Systemctl restart docker

相关资料:

<https://blog.csdn.net/qq_29393273/article/details/83625889>

<https://blog.csdn.net/qq_29393273/article/details/119294243>

2．

## 14.安装MYSQL

1.下载镜像

docker pull mysql

2.运行镜像

docker run -d --restart=always --name ben-test-mysql -e MYSQL\_ROOT\_PASSWORD=mysql@958958 -p 3307:3306 -v /var/test/mysqldata:/var/lib/mysql -v /var/test/mysql/conf:/etc/mysql/conf.d mysql --lower-case-table-names=1

## 15.安装Redis

## 16.Docker部署Springboot项目

1.0 在Linux上只安装Docker;

1.切换到宿主机 :/usr/local/docker 如果没有docker就创建一个新的目录mkdir docker;

2.创建一个存放项目的文件夹 mkdir itour;

3.进入文件夹 cd /usr/local/docker/itour/

4.使用rz命令 上传springboot项目的jar;

如果找不到该命令，安装rz命令：yum install lrzsz

5.新建Dockerfile文件

Dockerfile文件,类似于window中bat文件，可以执行一些命令;

#Docker image for springboot file run

#基础镜像使用java

FROM java:8

#作者

MAINTAINER wangtao <wwangtaoc11@163.com>

#指定临时文件目录为/tmp

VOLUME /tmp

#将jar包添加到容器中并更名为itour.jar

ADD itour-online-web.jar itour.app

#运行jar

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/itour.jar"]

6.构建容器：

docker build -t itour .

7.启动容器

docker run -name itour -p 8899:8899 itour

8.运行成功及可以访问

相关文档:

<https://www.jianshu.com/p/397929dbc27d>

<https://www.imooc.com/article/44582>

Dockerfile相关文档：

<https://www.runoob.com/docker/docker-dockerfile.html>

<https://blog.csdn.net/cxzhq2002/article/details/91371225>

<https://www.cnblogs.com/edisonchou/p/dockerfile_inside_introduction.html>

<https://www.cnblogs.com/panwenbin-logs/p/8007348.html>

## 17.安装Tomcat

1.拉取tomcat镜像

docker pull tomcat

2.运行镜像(创建一个守护式容器（如果对一个需要长期运行的容器来说，我们可以创建一个守护式容器,容器的名称不能重复）)

docker run -d --name tomcat-8080 -p 8080:8080 -v /usr/local/webapps/:/usr/local/tomcat/webapps/ tomcat

3.进入守护式容器内部;

docker exec -it tomcat-8080 bash

3.1切换到/usr/local/tomcat/webapps/

cd /usr/local/tomcat/webapps/

3.2创建一个目录(test或root目录(tomcat的默认目录))

mkdir test

3.3将index.html 复制到容器中

rz 选择index.html 页面;

4.4安装rz上传命令

yum install lrzsz

5.5直接访问

http://服务器地址:端口+目录/index.html

4.关于Docker中安装Tomcat后不能访问及访问主页404的解决办法

https://blog.csdn.net/weixin\_38423249/article/details/80554583

https://www.cnblogs.com/chzushl/p/13385580.html

## 18.Docker部署war项目

1.Linux安装docker

1. 运行tomcat

docker run -d --name tomcat-8080 -p 8080:8080 -v /usr/local/webapps/:/usr/local/tomcat/webapps/ tomcat

2.切入到/usr/local/webapps/

3.使用rz命令选择war包;

4.浏览器访问

## 19.Eclipse或Idea 集成Docker实现镜像打包一键部署

### 1.Linux环境准备Cetos7(腾讯云)

a.安装Docker

b.安装Mysql

c.安装Redis

### 2.windows 环境准备

a.安装开发工具

b.安装docker插件

c.Linux远程连接工具(xshell)

d.项目准备(Springboot+Maven+mysql+Redis+eureka)

e.打包项目推送镜像到服务器

f.运行镜像

d.浏览器访问

### 3.Docker插件的安装

Idea安装Docker插件

a.打开idea,从File-->Settings-->Plugins-->Install jetBrains plugin 进入插件安装界面;

b.在搜索框中输入docker，可以看到Docker integration,点击右边的install按钮进行安装，安装后重启idea;

**Idea配置docker**

a.配置docker，连接到远程的docker容器

从File-->Setttings-->Builde,Excution,Deployment->Docker打开配置界面;

eclipse 安装Docker插件

a.Help-->Eclipse Marketplace-->搜索 Eclipse Docker Tooling

**eclipse配置docker**

1. window-->show view-->other-->选择docker explorer

相关文档：<https://www.e-learn.cn/topic/1883011>

### 4.Docker插件连接到Docker

1.Docker引擎默认不允许外部访问需要开启远程访问;  
#修改Docker服务文件  
vim /usr/lib/systemd/system/docker.service  
#修改ExecStart这行,在后面追加红色部分

ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock  
将文件内的ExecStart注释，新增上行红色字体部分;

#重新加载配置文件  
systemctl daemon-reload  
#重启服务  
systemctl restart docker.service  
#查看端口是否开启  
netstat -nlpt #如果找不到netstat命令，可以进行安装 yum install net-tools  
#检查curl是否生效，注意，必须关闭防火墙或腾讯云安全组开放2375端口，这个端口也可以自定义  
curl http://服务器地址：2375/info  
注意ip地址必须和服务器地址统一;  
相关文档：<https://blog.csdn.net/boling_cavalry/article/details/100049996>

### 5.配置Docker插件

两种方式:

1.直接在pom.xml中配置;

2.在pom中指定Dockfile文件的位置;

直接在pom.xml中配置的方式:

<!-- docker 插件 可以帮助我们在Maven工程中通过简单的配置，自动生成镜像并推送到仓库中-->

<plugin>

<groupId>com.spotify</groupId>

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

<version>1.0.0</version>

<configuration>

<!-- 镜像名称 -->

<imageName>${docker.image.prefix}/${project.artifactId}</imageName>

<!-- 指定标签 -->

<imageTags>

<imageTag>latest</imageTag>

</imageTags>

<!-- 基础镜像java 1.8 -->

<baseImage>java</baseImage>

<!-- 制作者提供本人信息 -->

<maintainer>wangtao wwangtaoc11@163.com</maintainer>

<!-- 切换到/root目录 -->

<workdir>/ROOT</workdir>

<cmd>["java","-version"]</cmd>

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

<!-- 指定远程docker api 地址 -->

<dockerHost>http://106.54.162.159:2375</dockerHost>

<!-- 指定Dockerfile文件的位置

<dockerDirectory>src/main/docker</dockerDirectory> -->

<resources>

<resource>

<targetPath>/ROOT</targetPath>

<!-- 指定需要复制的根目录:${project.build.directory}表示target目录 -->

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

<!-- 指定需要复制的文件 ${project.build.finalName}.jar指的是打包后的文件-->

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

</resource>

</resources>

</configuration>

<!-- 以上还需要通过mvn clean package docker:build 手动构建镜像，在<plugin>添加以下插件即可以

在项目打包Package的时候即可自动构建镜像,只需要run就可以访问了;

当执行mvn package时执行mvn clean package docker:build

-->

<executions>

<execution>

<id>build-image</id>

<phase>package</phase>

<goals>

<goal>build</goal>

</goals>

</execution>

</executions>

</plugin>

### 6.执行 mvn package [-P test]

## 20.Docker部署Springboot项目中遇到的问题

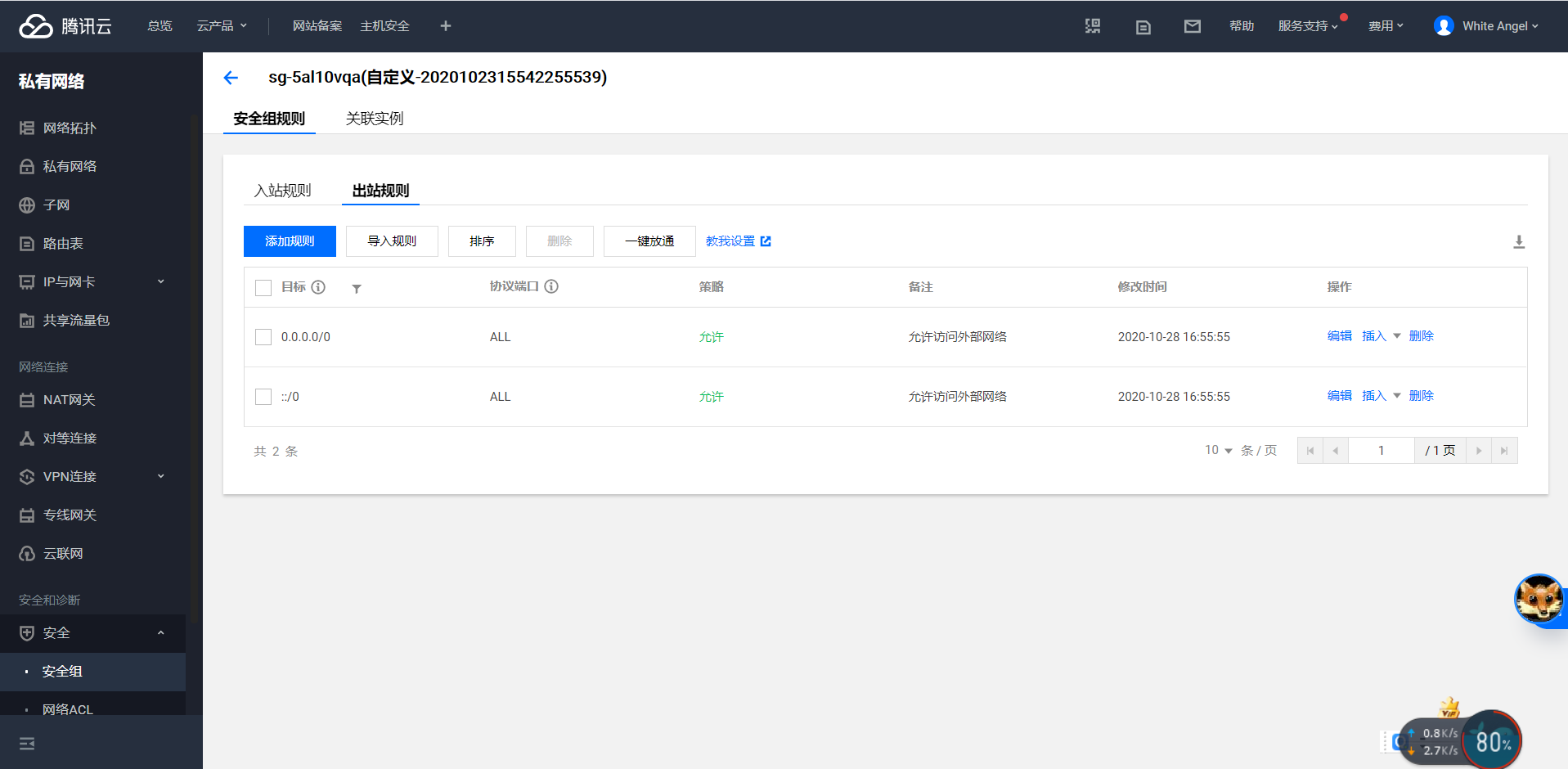
### 本地正常,部署Springboot+eureka项目到Docker时eureka启动正常，但是其他服务不能注册到注册中心(eureka)

使用curl或telnet查看容器是否可以访问;

解决办法：

登录腾讯云控制台--》私有网络—》安全与诊断—》实例端口验通—》发现出站未放通

去腾讯云控制台安全组添加出站规则



### 2. Note: further occurrences of HTTP request parsing errors will be logged at DEBUG level.

### java.lang.IllegalArgumentException: Request header is too large

解决方案:

尝试配置：

springboot部署到服务器(1核2G)内存不够用

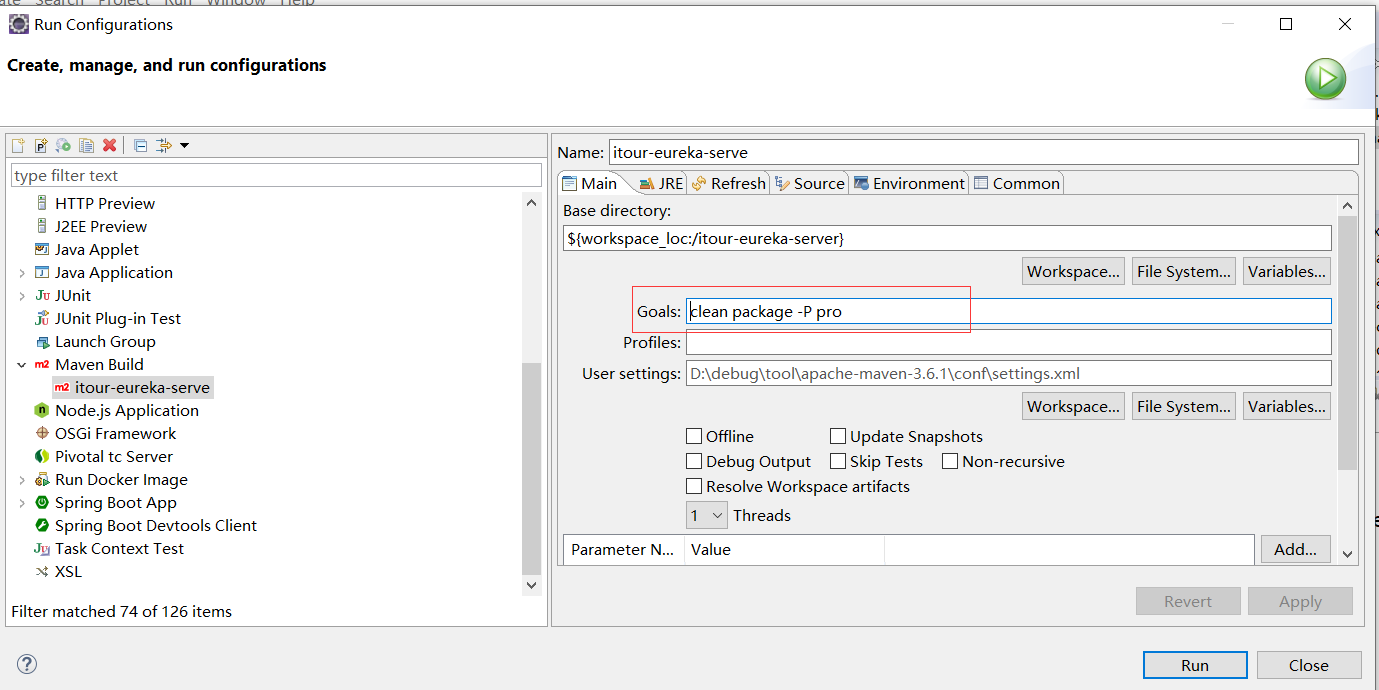
Springboot应用特别吃内存;

Linux 查看内存使用情况：

Free -h

<https://www.cnblogs.com/ultranms/p/9254160.html>

### 执行mvn命令的时候 eclipse 默认添加了mvn前缀，所有不需要添加mvn前缀了



镜像文件很大解决方方案：

Docker相关文档：

<https://yeasy.gitbook.io/docker_practice/image/dockerfile/entrypoint>

部署上去的镜像文件较大;

1. 目前未找到对应的解决办法；

删除内存缓存命令：echo 1 > /proc/sys/vm/drop\_caches

## 21.Dockefile文件

### 1.什么是Dockerfile文件？

Dockerfile 是一个用来构建镜像的文本文件，文本内容包含了一条条构建镜像所需的指令和说明。

Docker通过读取Dockerfile中的指令自动生成映像。docker build命令用于从Dockerfile构建映像。可以在docker build命令中使用-f标志指向文件系统中任何位置的Dockerfile；

**docker build -f /path//Dockerfile**

### 2.Dockerfile的基本结构

Dockerfile 一般分为四部分：基础镜像信息、维护者信息、镜像操作指令和容器启动时执行指令，’#’ 为 Dockerfile 中的注释。

### 3.Dockerfile文件中常用指令

#### 1.From

**FROM：**定制的镜像都是基于 FROM 的镜像（基础镜像）

格式：FROM image

FROM image:tag

FROM image@digest

tag或digest是可选的，如果不使用这两个值时，会使用latest版本的基础镜像;

例如：FROM java

#### 2. MAINTAINER

MAINTAINER: 指定维护者信息

格式：MAINTAINER name

例如：MAINTAINER [wwangtao@c11@163.com](mailto:wwangtao@c11@163.com)

#### 3.RUN

RUN: 构建镜像时执行的命令;

RUN用于在镜像容器中执行命令，其有以下两种命令执行方式：

shell 格式:

RUN **命令行命令**

**命令行命令**等同于，在终端操作的 shell 命令。

exec 格式：

RUN ["executable","param1","param2"]

Executable可执行文件;

param1:参数1;

param2：参数2;

RUN ["./test.php", "dev", "offline"] 等价于 RUN ./test.php dev offline

**注意**：Dockerfile 的指令每执行一次都会在 docker 上新建一层。所以过多无意义的层，会造成镜像膨胀过大。

例如：

FROM centos  
RUN **yum** -y **install** **wget**  
RUN **wget** -O redis.tar.gz "http://download.redis.io/releases/redis-5.0.3.tar.gz"  
RUN **tar** -xvf redis.tar.gz

以上执行会创建 3 层镜像。可简化为以下格式：

FROM centos  
RUN **yum** -y **install** **wget** \  
    **&&** **wget** -O redis.tar.gz "http://download.redis.io/releases/redis-5.0.3.tar.gz" \  
    **&&** **tar** -xvf redis.tar.gz

如上，以 **&&** 符号连接命令，这样执行后，只会创建 1 层镜像

#### 4.上下文路径

1.上下文路径：

上下文路径是指 docker 在构建镜像，有时候想要使用到本机的文件（比如复制），docker build 命令得知这个路径后，会将路径下的所有内容打包。

**由于 docker 的运行模式是 C/S。我们本机是 C，docker 引擎是 S。实际的构建过程是在 docker 引擎下完成的，所以这个时候无法用到我们本机的文件。这就需要把我们本机的指定目录下的文件一起打包提供给 docker 引擎使用。如果未说明最后一个参数，那么默认上下文路径就是 Dockerfile 所在的位置。**

**上下文路径下不要放无用的文件，因为会一起打包发送给 docker 引擎，如果文件过多会造成过程缓慢。**

#### 5.COPY

复制指令，从上下文目录中复制文件或者目录到容器里指定路径。

格式：COPY --chown=<user>:<group> 源路径1... 目标路径

COPY --chown=<user>:<group> ["源路径1",... "目标路径"]

**--chown=<user>:<group>**：可选参数，用户改变复制到容器内文件的拥有者和属组

**源路径**：源文件或者源目录，这里可以是通配符表达式，其通配符规则要满足 Go 的 filepath.Match 规则。例如：

COPY hom\* /mydir/

COPY hom?.txt /mydir/

**目标路径**：容器内的指定路径，该路径不用事先建好，路径不存在的话，会自动创建;

#### 6.ADD

ADD 指令和 COPY 的使用格类似（同样需求下，官方推荐使用 COPY）

格式：add --chown=<user>:<group> 源路径1... 目标路径

add--chown=<user>:<group> ["源路径1",... "目标路径"]

**ADD 的优点**：在执行 **源文件** 为 tar 压缩文件的话，压缩格式为 gzip, bzip2 以及 xz 的情况下，会自动复制并解压到 **目标路径**。

**ADD 的缺点**：在不解压的前提下，无法复制 tar 压缩文件。会令镜像构建缓存失效，从而可能会令镜像构建变得比较缓慢。具体是否使用，可以根据是否需要自动解压来决定；

#### 7.CMD

类似于 RUN 指令，用于运行程序，但二者运行的时间点不同:

CMD 在docker run 时运行。

RUN 是在 docker build。

CMD <shell 命令>

CMD ["<可执行文件或命令>","<param1>","<param2>",...]

CMD ["<param1>","<param2>",...] # 该写法是为 ENTRYPOINT 指令指定的程序提供默认参数

**作用**：为启动的容器指定默认要运行的程序，程序运行结束，容器也就结束。CMD 指令指定的程序可被 docker run 命令行参数中指定要运行的程序所覆盖。

**注意**：如果 Dockerfile 中如果存在多个 CMD 指令，仅最后一个生效。

推荐使用第二种格式，执行过程比较明确。第一种格式实际上在运行的过程中也会自动转换成第二种格式运行，并且默认可执行文件是 sh;

#### 8.ENTRYPOINT

ENTRYPOINT：类似于 CMD 指令，但其不会被 docker run 的命令行参数指定的指令所覆盖，而且这些命令行参数会被当作参数送给 ENTRYPOINT 指令指定的程序。但是, 如果运行 docker run 时使用了 --entrypoint 选项，将覆盖 CMD 指令指定的程序。

优点：在执行 docker run 的时候可以指定 ENTRYPOINT 运行所需的参数。

注意：如果 Dockerfile 中如果存在多个 ENTRYPOINT 指令，仅最后一个生效。

格式：

ENTRYPOINT ["<executeable>","<param1>","<param2>",...]

可以搭配 CMD 命令使用：一般是变参才会使用 CMD ，这里的 CMD 等于是在给 ENTRYPOINT 传参

#### 9. WORKDIR

指定工作目录。用 WORKDIR 指定的工作目录，会在构建镜像的每一层中都存在。（WORKDIR 指定的工作目录，必须是提前创建好的）。

docker build 构建镜像过程中的，每一个 RUN 命令都是新建的一层。只有通过 WORKDIR 创建的目录才会一直存在。

格式：

WORKDIR <工作目录路径>

更多的相关命令：

<https://www.runoob.com/docker/docker-dockerfile.html>

### 4.Docker build命令

<https://www.runoob.com/docker/docker-build-command.html>