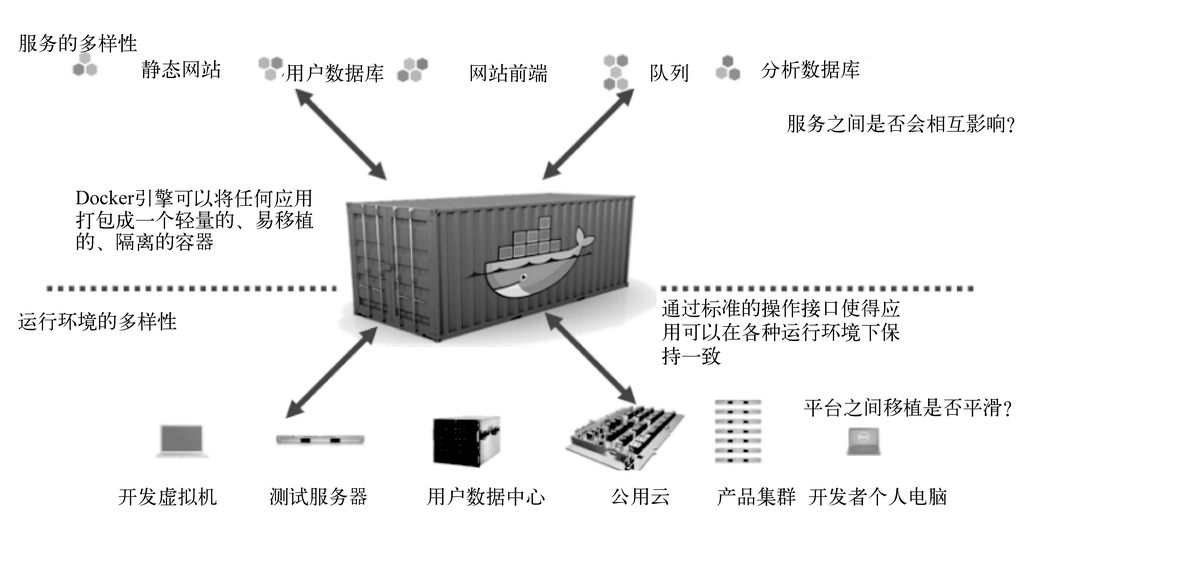
微服务的docker化

王国庆 2020-08-03

1. 简介：

Docker是一个开源的应用容器引擎，让开发者可以打包自己的应用及依赖包到一个可移植的容器中，然后发布到任何的Linux机器上，也可以实现虚拟化，容器完全是使用沙箱机制，相互之间不会有任何的接口。



一个完整的Docker有以下几个部分组成：

1. Docker client 客户端
2. Docker Daemon 守护进程
3. Docker image镜像
4. Docker Container 容器
5. 在linux 服务器中（比如：centos7）中的安装
6. 卸载旧版本（如果安装过旧的docker）yum remove docker
7. Yum -y update
8. Yum –y install docker
9. Vi /etc/sysconfig/docker
10. --selinux-enabled =false
11. /etc/sysconfig/docker
12. 修改配置
13. #重新启动docker
14. Systemctl restart docker

3、采用docker方式打包到服务器

（1）在pom.xml 文件中加入对应的插件(docker.image.prefix是指仓库名称，建议使用dockerhub的名称，否则无法进行提交push到自己的仓库)

<!---docker 远程服务地址---🡪

<dockerHost> http://xx.xx.xx.xx.8888</dockerHost>

采用docker方式打包服务器。

（2）在各模块的src/main 下面新建docker包，在src/main/docker 下面新建DockerFile

FORM open jdk:8-jdk-alpine

VOLUME /tmp/

ADD wm-user –center.jar app.jar

Run sh –c ‘touch /app.jar’

ENV JAVA\_OPTS= “ ”

ENTRYPOINT[“sh “, ”-c”, “java $JAVA\_OPTS –Djava.security.egd= file:/dev/ . /urandom –jar /app.jar”]

注：

需要注意的是：（app.jar是指该工程打包的jar）  
FROM 指向需要创建的容器名称，open jdk:8-jdk-alpine （除了注释外，必须是第一行写，否则报错）

Tag 不写默认取latest，可以当做版本号。

Add (向容器中添加文件，主机文件、网络文件、文件夹)

VOLUME(指定挂载点、主机的其他目录)

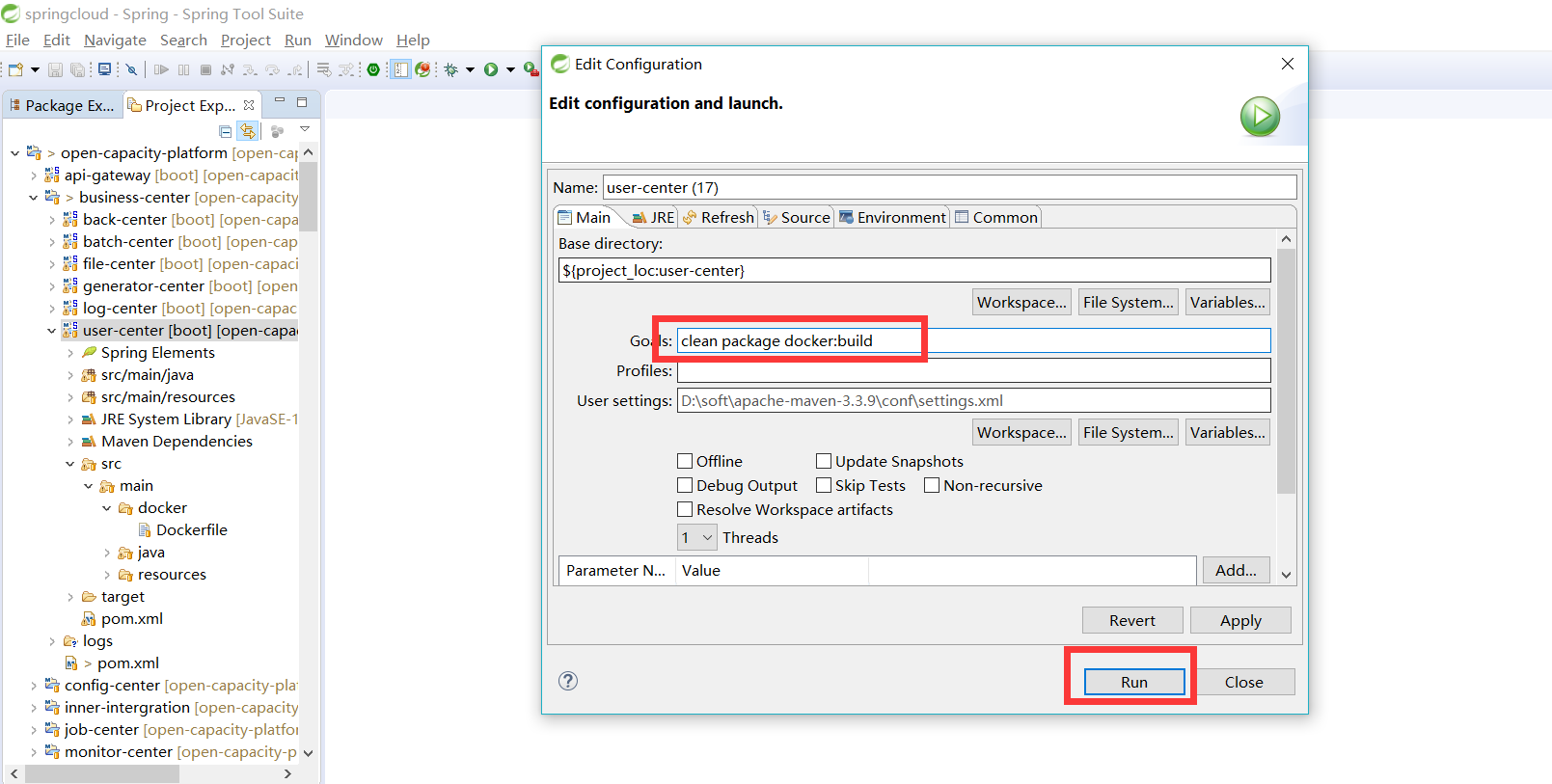
Run(用于修改镜像的命令，常用的安装库，程序以及配置程序)

ENV (用于设置容器的环境变量)

CMD(启动容器的默认命令)

ENTRYPOINT(与cmd类似，配合CMD使用)

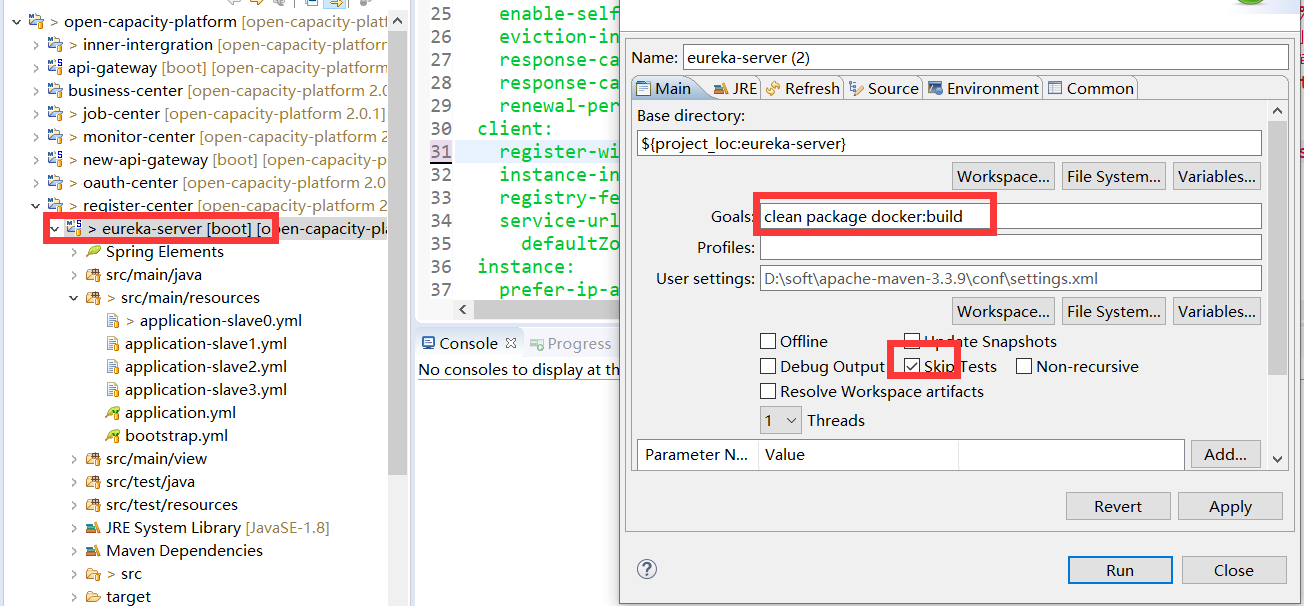
项目打包成镜像



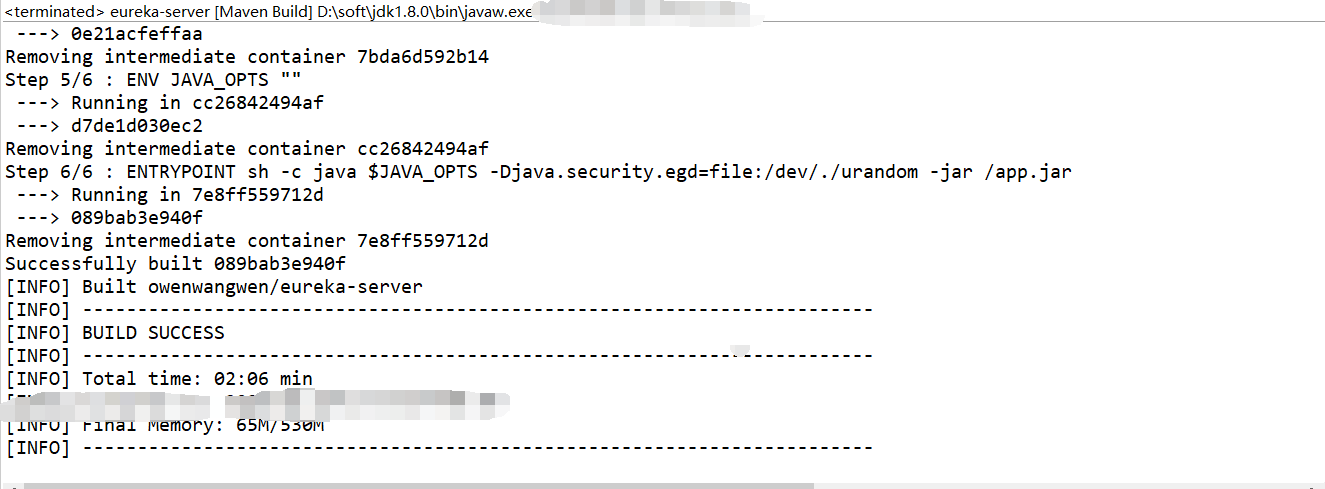
（3）步骤：

a、配置docker主机地址

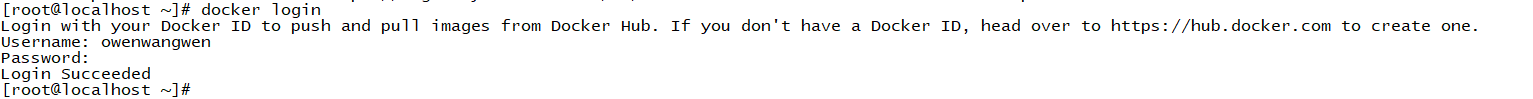
b、docker:build形成镜像上传到docker 主机



C、形成镜像过程



d、登录主机查看镜像

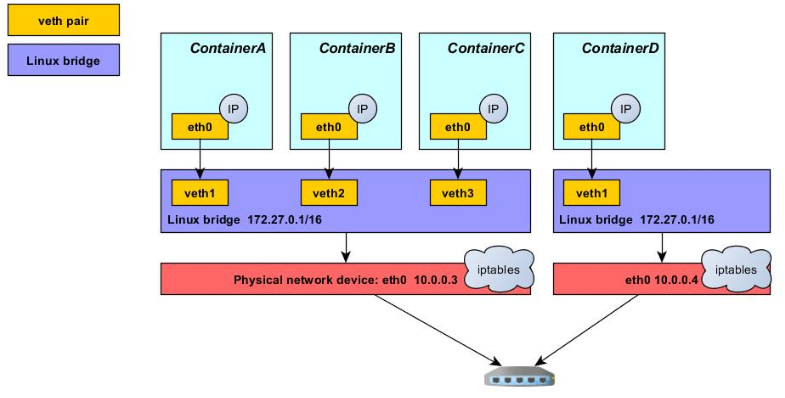


e、上传hub.docker

[root@localhost ~]# docker push owenwanggq/eureka-serve

r

1. docker网络原理



宿主机和docker容器通过veth网卡通讯。

Veth网卡对

Ip link add veth0 type veth peer name veth1

Ip link show

Ip netns add ns1

Ip nets show

放入不同的命名空间

Ip link set veth1 netns ns1

设置veth0

Ip addr add 172.0.0.1/24 dev veth0

Ip link set veth0 up

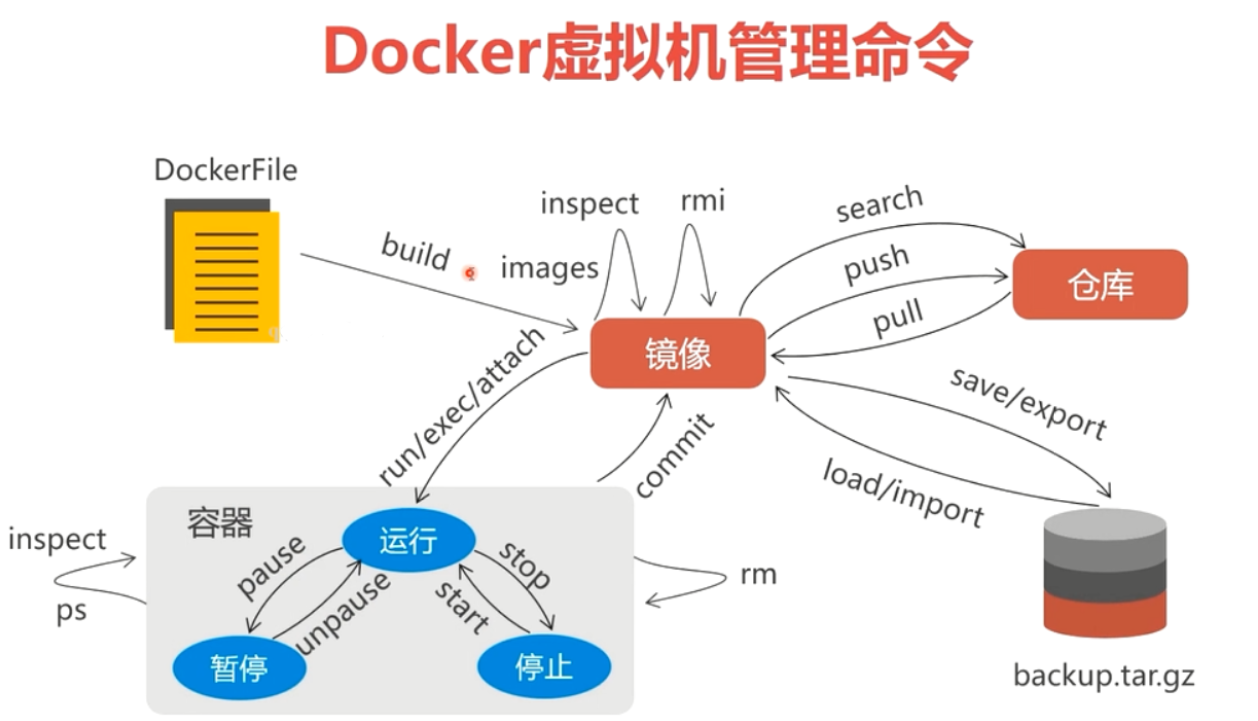
设置veth1

Ip netns exec ns 1 ip addr add 172.0.0.2/24 dev veth1

Ip netns exec ns1 ip link set veth1 up

Ip netns exec ns1 ping 172.0.0.1

5．容器的管理

[root@localhost ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

3401182aa6ea nginx:1.15 "nginx -g 'daemon of鈥 34 minutes ago Up 34 minutes 0.0.0.0:80->80/tcp nginx

[root@localhost ~]# docker port 3401182aa6ea

80/tcp -> 0.0.0.0:80

[root@localhost ~]# docker top 3401182aa6ea

UID PID PPID C STIME TTY TIME CMD

root 25694 25676 0 15:14 ? 00:00:00 nginx: master process nginx -g daemon off;

101 25731 25694 0 15:14 ? 00:00:00 nginx: worker process

[root@localhost ~]# docker stats 3401182aa6ea --no-stream

CONTAINER ID NAME CPU % MEM USAGE / LIMIT MEM % NET I/O BLOCK I/O PIDS

3401182aa6ea nginx 0.00% 1.379MiB / 1.928GiB 0.07% 2.72kB / 2.8kB 0B / 0B 2

[root@localhost ~]#

运行容器：

[root@localhost ~]# docker run -d -p 80:80 -e ABC=123 --name=nginx -h nginxweb nginx:1.15

查看容器：

[root@localhost ~]# docker inspect 965757850fef

进入容器：

[root@localhost ~]# docker exec -it nginx bash

root@nginxweb:/# echo $ABC

123

root@nginxweb:/# hostname

nginxweb

root@nginxweb:/#

删除容器：

[root@localhost ~]# docker ps |awk '{print $1}'|xargs docker rm -f