1.安装jdk1.8(centos7.5)

yum安装

1.搜索jdk安装包

```
yum search java|grep jdk
```

2.下载jdk1.8，下载之后默认的目录为： /usr/lib/jvm/

```
yum install java-1.8.0-openjdk
```

3配置环境变量及验证安装

```
java -version
```

2.安装docker

1.初始化环境，卸载旧的版本

```
yum remove docker \
                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
                  docker-logrotate \
                  docker-engine
```

2.需要的安装包

```
yum install -y yum-utils
```

3.设置镜像仓库

```
 yum-config-manager \
    --add-repo \
    http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo
```

4.安装docker

```

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

5.启动docker

```
systemctl start docker
```

6.是否安装成功

```
docker version
```

卸载

```
yum remove docker-ce docker-ce-cli containerd.io


rm -rf /var/lib/docker
```

7.配置阿里镜像加速

```
sudo mkdir -p /etc/docker


sudo tee /etc/docker/daemon.json <<-'EOF'
```

```
{
  "registry-mirrors": ["https://ui6jarkn.mirror.aliyuncs.com"]
}
EOF


sudo systemctl daemon-reload


sudo systemctl restart docker
```

3.docker命令

3.1docker帮助命令

```
docker version

docker info

docker 命令 --help  比如docker images --help
```

3.2docker 镜像命令

```
docker images -a 查看所有镜像

docker search [] 搜索镜像 : docker search mysql

docker pull  [] 下载镜像

docker rmi [] 删除镜像

docker rmi -f $(docker images -aq) 删除所有镜像
```

3.3docker 容器命令

```
docker run [可选参数] images

# 参数说明

--name=      容器的名称
-d           后台方式运行
-it          使用交互式方式运行，进如容器查看
-p           指定主机端口与容器端口映射 8080: 8080


docker stats 查看容器运行状态
# 测试
```

```
docker run -it mysql /bin/bash 启动并进如容器
```

```
#退出容器，容器停止
exit

ctrl + q +p 退出容器，容器不停止

#查看容器
docker ps 查看运行中的容器
          -a 查看所有容器
#删除容器
docker rm [容器id] 删除容器，不能删除运行中的容器
        -f [容器id] 强制删除运行中的容器
#启动和停止rq
docker start [容器id] 启动容器

docker restart [容器id] 重启容器

docker stop [容器id] 停止容器

docker kill [容器id] 强制停止容器
```

3.4其他命令

3.4.1后台启动命令

后台启动容器

```
# 命令 docker run -d 镜像名!
[root@kuangshen /]# docker run -d centos

# 问题docker ps，发现 centos 停止了

# 常见的坑：docker 容器使用后台运行，就必须要有要一个前台进程，docker发现没有应用，就会自动停止
# nginx，容器启动后，发现自己没有提供服务，就会立刻停止，就是没有程序了
```

3.4.2查看日志

查看日志

```
docker logs -f -t --tail 容器，没有日志

# 自己编写一段shell脚本
[root@kuangshen /]# docker run -d centos /bin/sh -c "while true;do echo kuangshen;sleep 1;done"

# [root@kuangshen /]# docker ps
CONTAINER ID        IMAGE
dce7b86171bf        centos

# 显示日志
 -tf            # 显示日志
 --tail number  # 要显示日志条数
[root@kuangshen /]# docker logs -tf --tail 10 dce7b86171bf
```

3.4.3查看容器进程信息

查看容器中进程信息 ps

```
# 命令 docker top 容器id
[root@kuangshen /]# docker top dce7b86171bf
UID                 PID                 PPID                C                   STIME               TTY
root                22891               22875               0                   21:21               ?
root                23351               22891               0                   21:25               ?
```

3.4.5查看镜像元数据

查看镜像的元数据

```
# 命令
docker inspect 容器id

# 测试
[root@kuangshen /]# docker inspect dce7b86171bf
[
    {
        "Id": "dce7b86171bf3f60843e0665a93f81c87e95235fa1278dba43df1e7b33e78b9e",
        "Created": "2020-05-14T13:21:56.603556735Z",
```

3.4.6进如当前正在 运行的容器（重要）

进入当前正在运行的容器

```
# 我们通常容器都是使用后台方式运行的，需要进入容器，修改一些配置

# 命令
docker exec -it 容器id bashShell

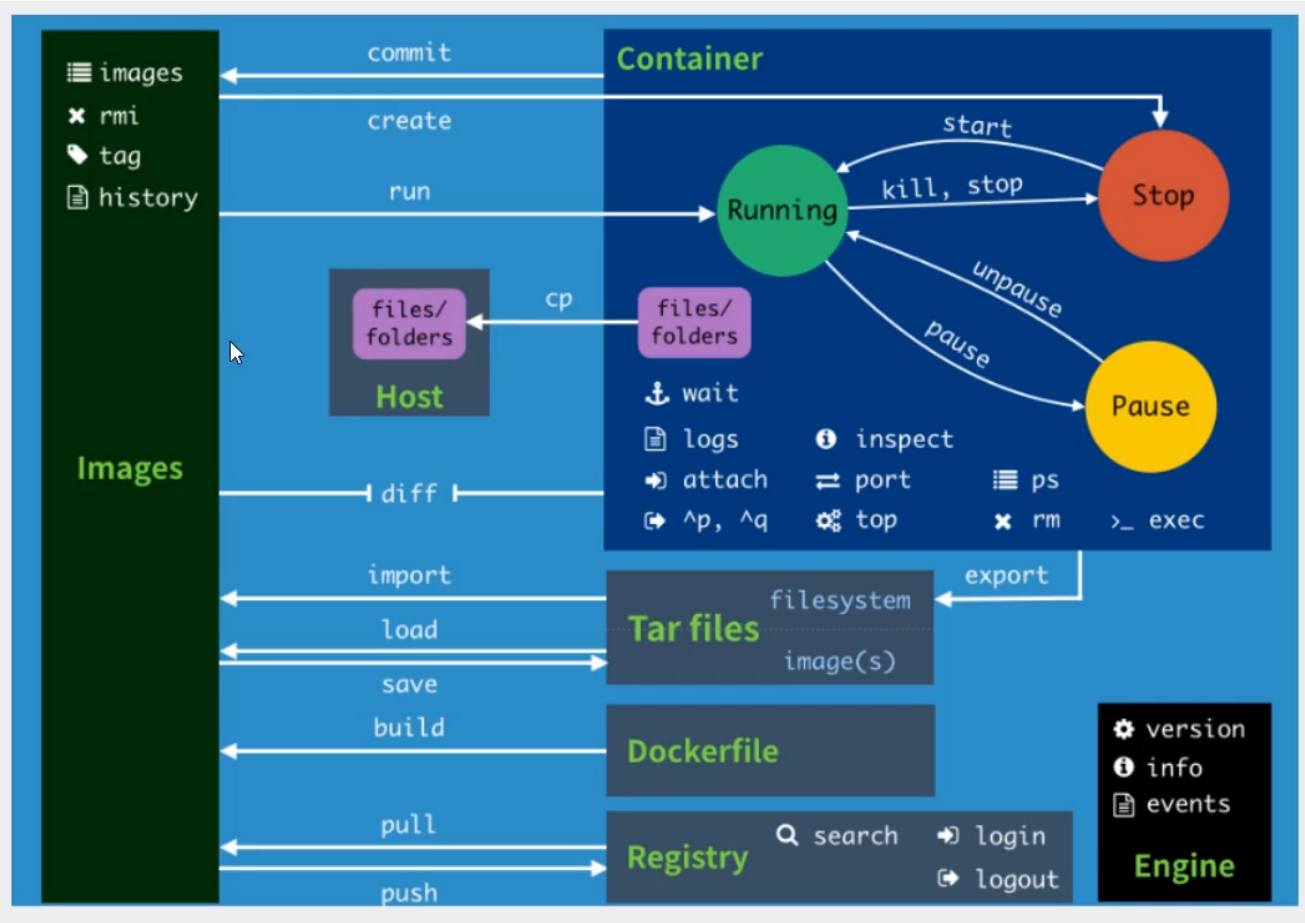
# 测试
[root@kuangshen /]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS               NAMES
dce7b86171bf        centos              "/bin/sh -c 'while t…"   8 minutes ago       Up 8 minutes
                     determined_proskuriakova
[root@kuangshen /]# docker exec -it dce7b86171bf /bin/bash
[root@dce7b86171bf /]# ls
bin  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr
var
```

3.4.7从容器内拷贝文件到主机

从容器内拷贝文件到主机上

```
docker cp 容器id:容器内路径   目的的主机路径
```

3.4.8命令图



4.可视化

portainer

什么 portainer ？

Docker图形化界面管理工具！提供一个后台面板供我们操作！

```
docker run -d -p 8088:9000 \
--restart=always -v /var/run/docker.sock:/var/run/docker.sock --privileged=true portainer/portainer
```

访问测试：http://ip:8088/

```
docker run -d -p 9000:9000 --restart always -v /var/run/docker.sock:/var/run/docker.sock --
privileged=true portainer/portainer
```

访问www.allweing.cn:9000

portainer.io

Please create the initial administrator user.

Username admin

Password

Confirm password

✖ The password must be at least 8 characters long

Create user

Connect Portainer to the Docker environment you want to manage.

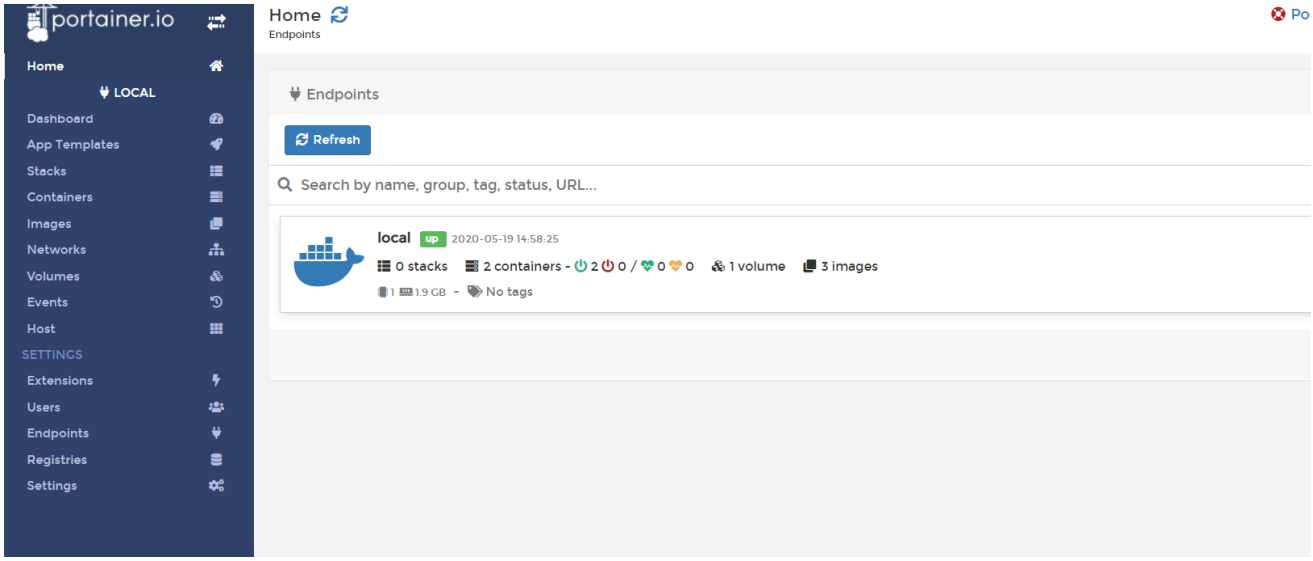
Local
Manage the Docker environment where Portainer is running

Remote
Manage a remote Docker environment

⚠ This feature is not yet available for native Docker Windows containers.

Please ensure that you have started the Portainer container with the following Docker flag `-v "/var/run/docker.sock:/var/run/docker.sock"` in order to connect to the local Docker environment.

Connect



5,提交一个自己的镜像

如何提交一个自己的镜像

commit镜像

```
docker commit 提交容器成为一个新的副本

# 命令和git原理类似
docker commit -m="提交的描述信息" -a="作者" 容器id  目标镜像名：[TAG]
```

实战测试

```
# 1、启动一个默认的tomcat

# 2、发现这个默认的tomcat 是没有webapps应用， 镜像的原因，官方的镜像默认 webapps下面是没有文件的！

# 3、我自己拷贝进去了基本的文件
```

提交一个容器为镜像

```
docker commit -m="my first tomcat" -a="zzzgy" 55c4d304adfc mytomcat:1.0
              #信息                 #作者      #容器id       镜像名：版本号
```

```
[root@VM_0_3_centos ~]# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mytomcat            1.0                 55b1b563e600        7 seconds ago       652MB
tomcat              9.0                 1b6b1fe7261e        2 days ago          647MB
```

6.容器数据卷

容器的持久化和同步操作，容器间数据共享

```
docker run -d --name zhutomcat -p 8081:8080 -v /home/webapps:/usr/local/tomcat/webapps
mytomcat:1.0
```

通过文件挂载，就可以在主机上修改文件内容同步到容器中

具名和匿名挂载

```
# 匿名挂载
-v 容器内路径!
docker run -d -P --name nginx01 -v /ect/nginx nginx

# 查看所有的 volume 的情况
[root@kuangshen home]# docker volume ls
local               9f38292179faa178afcce54d80be99d4ddd68c91d2a68870bcece72d2b7ed061

# 这里发现，这种就是匿名挂载，我们在 -v 只写了容器内的路径，没有写容器外的路径!

# 具名挂载
[root@kuangshen home]# docker run -d -P --name nginx02 -v juming-nginx:/etc/nginx nginx
95b809564484c8ac87d65c69643e7e67447f1c77ff9a91b93edec7003692e3a9
[root@kuangshen home]# docker volume ls
DRIVER              VOLUME NAME
local               juming-nginx

# 通过 -v  卷名:容器内路径
# 查看一下这个卷
```

所有的docker容器内的卷，没有指定目录的情况下都是在 `/var/lib/docker/volumes/xxxx/_data`

我们通过具名挂载可以方便的找到我们的 一个卷，大多数情况在使用的 `具名挂载`

```
# 如何确定是具名挂载还是匿名挂载，还是指定路径挂载!
-v 容器内路径        # 匿名挂载
-v 卷名:容器内路径    # 具名挂载
-v /宿主机路径::容器内路径  # 指定路径挂载!
```

拓展：

```
# 通过 -v 容器内路径：ro  rw 改变读写权限
ro   readonly  # 只读
rw   readwrite  # 可读可写

# 一旦这个了设置了容器权限，容器对我们挂载出来的内容就有限定了!
docker run -d -P --name nginx02 -v juming-nginx:/etc/nginx:ro  nginx
docker run -d -P --name nginx02 -v juming-nginx:/etc/nginx:rw  nginx

# ro 只要看到ro就说明这个路径只能通过宿主机来操作，容器内部是无法操作!
```

shell

7.dockerfile(命令参数脚本)

7.1dockerfile构建过程

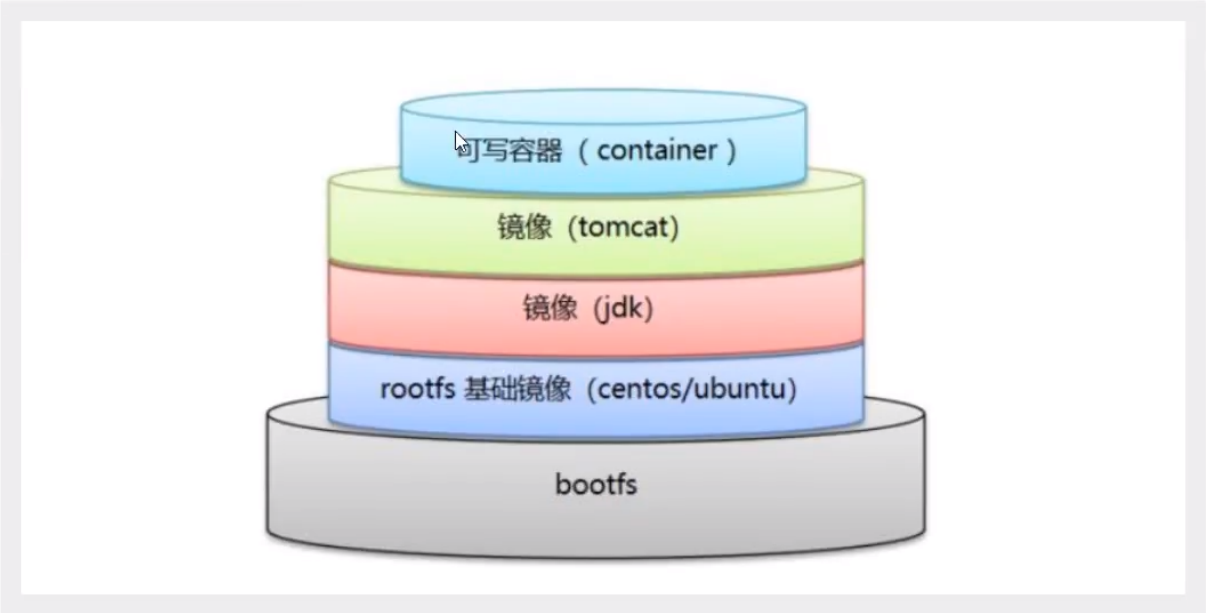
基础知识：

1、每个保留关键字（指令）都是必须是大写字母

2、执行从上到下顺序执行

3、# 表示注释

4、每一个指令都会创建提交一个新的镜像层，并提交！



7.2dockerfile 命令

以前的话我们就是使用别人的，现在我们知道了这些指令后，我们来练习自己写一个镜像！

```
FROM          # 基础镜镜像，一切从这里开始构建
MAINTAINER    # 镜像是谁写的，姓名+邮箱
RUN           # 镜像构建的时候需要运行的命令
ADD           # 步骤：tomcat镜像，这个tomcat压缩包！添加内容
WORKDIR       # 镜像的工作目录
VOLUME        # 挂载的目录
EXPOSE        # 保留端口配置
CMD           # 指定这个容器启动的时候要运行的命令,只有最后一个会生效，可被替代
ENTRYPOINT    # 指定这个容器启动的时候要运行的命令,可以追加命令
ONBUILD       # 当构建一个被继承 DockerFile 这个时候就会运行 ONBUILD  的指令。触发指令。
COPY          # 类似ADD ，将我们文件拷贝到镜像中
ENV           # 构建的时候设置环境变量！
```

`docker history [镜像id]` 查看构建历史

实战：自己做一个centos镜像

1. cd /home
2. mkdir dockerfile
3. cd dockerfile
4. vim mydockerfile

5.
```
FROM centos
MAINTAINER zzzgy<1589953744@qq.com>

ENV MYPATH /usr/local
WORKDIR $MYPATH

RUN yum -y install vim
RUN yum -y install net-tools

EXPOSE 80

CMD echo $MYPATH
CMD echo "------------end--------------"
CMD  /bin/bash
```

6.
```
docker build -f mydockerfile -t mycentos:0.1 .
```

7. ```
   Successfully built 557aca4492da
   Successfully tagged mycentos:0.1
   #成功
```

```
docker history [镜像id] 查看构建历史
```

7.3cmd 与ENTORYPOINT区别
```

测试cmd

```shell
# 编写 dockerfile 文件
[root@kuangshen dockerfile]# vim dockerfile-cmd-test
FROM centos
CMD ["ls","-a"]

# 构件镜像
[root@kuangshen dockerfile]# docker build -f dockerfile-cmd-test -t cmdtest .

# run运行 ，发现我们的ls -a 命令生效
[root@kuangshen dockerfile]# docker run dd8e4401d72f
.
..
.dockerenv
bin
dev
etc
home
lib
lib64

# 想追加一个命令 -l   ls -al
[root@kuangshen dockerfile]# docker run dd8e4401d72f -l
docker: Error response from daemon: OCI runtime create failed: container_linux.go:349: starting
container process caused "exec: \"-l\": executable file not found in $PATH": unknown.

# cmd的清理下 -l 替换了CMD ["ls","-a"] 命令，-l 不是命令所以保存！
```

ENTORYPOINT 可以直接追加命令

实战：构建一个tomcat镜像

```
FROM centos
MAINTAINET kuangshen<24736743@qq.com>

COPY readme.txt /usr/local/readme.txt

ADD jdk-8u11-linux-x64.tar.gz /usr/local/
ADD apache-tomcat-9.0.22.tar.gz /usr/local/

RUN yum -y install vim

ENV MYPATH /usr/local
WORKDIR $MYPATH

ENV JAVA_HOME /usr/local/jdk1.8.0_11
ENV CLASSPATH $JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
ENV CATALINA_HOME /usr/local/apache-tomcat-9.0.22
ENV CATALINA_BASH /usr/local/apache-tomcat-9.0.22
ENV PATH $PATH:$JAVA_HOME/bin:$CATALINA_HOME/lib:$CATALINA_HOME/bin

EXPOSE 8080

CMD /usr/local/apache-tomcat-9.0.22/bin/startup.sh && tail -F /url/local/apache-tomcat-9.0.22/bin/logs/catalina.out
~
```

```
docker build -t mytomcat .
#如果文件名是dockerfile 就不用指定文件
docker build -f mydockerfile -t mycentos .
```

7.4发布镜像

> DockerHub

1、地址 https://hub.docker.com/ 注册自己的账号！

2、确定这个这账号可以登录

3、在我们服务器上提交自己的镜像

```
[root@kuangshen tomcat]# docker login --help

Usage:  docker login [OPTIONS] [SERVER]

Log in to a Docker registry.
If no server is specified, the default is defined by the daemon.

Options:
  -p, --password string   Password
      --password-stdin    Take the password from stdin
  -u, --username string   Username
```

4、登录完毕后就可以提交镜像了，就是一步 docker push

```
[root@kuangshen tomcat]# docker login -u kuangshen
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

4、登录完毕后就可以提交镜像了，就是一步 docker push

```
# push自己的镜像到服务器上！
[root@kuangshen tomcat]# docker push diytomcat
The push refers to repository [docker.io/library/diytomcat]
fcc7fccb8e04: Preparing
b5577f344233: Preparing
bdcb94365850: Preparing
1c5bd81521f5: Preparing
0683de282177: Preparing
denied: requested access to the resource is denied # 拒绝

# push镜像的问题？
[root@kuangshen tomcat]# docker push kuangshen/diytomcat:1.0
The push refers to repository [docker.io/kuangshen/diytomcat2]
An image does not exist locally with the tag: kuangshen/diytomcat2

# 解决，增加一个 tag
[root@kuangshen tomcat]# docker tag f8559daf1fc2 kuangshen/tomcat:1.0

# docker psuh上去即可！
[root@kuangshen tomcat]# docker push kuangshen/tomcat:1.0
The push refers to repository [docker.io/kuangshen/tomcat]
fcc7fccb8e04: Preparing
b5577f344233: Preparing
bdcb94365850: Preparing
1c5bd81521f5: Preparing
0683de282177: Preparing
```

> 阿里云镜像服务上

1、登录阿里云

2、找到容器镜像服务

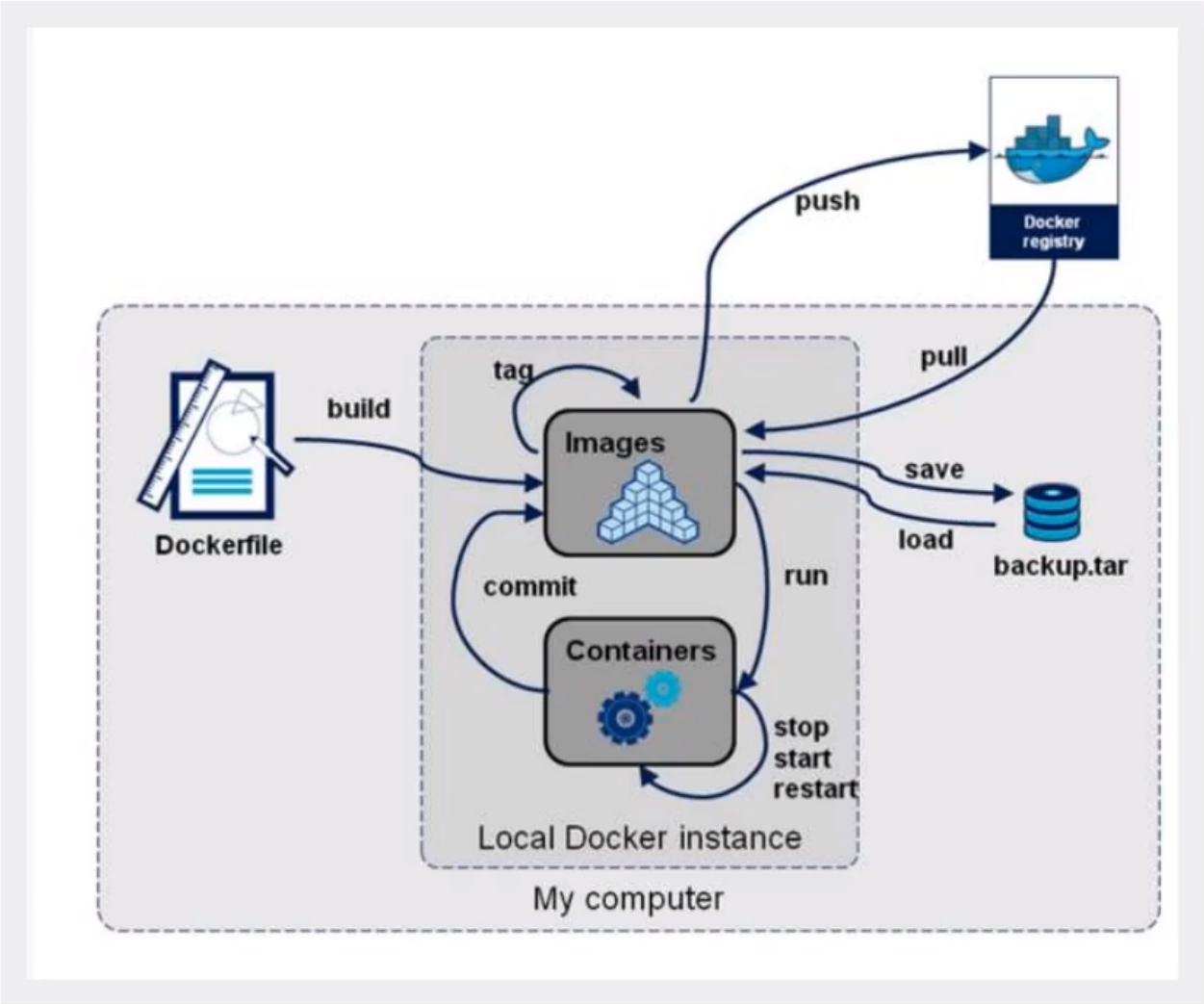
3、创建命名空间

4、创建容器镜像

5、浏览阿里云

参考阿里云文档

小结



8docker网络

```
[root@kuangshen tomcat]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0  <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:16:3e:30:27:f4 brd ff:ff:ff:ff:ff:ff
    inet 172.17.90.138/20 brd 172.17.95.255 scope global dynamic eth0
       valid_lft 310744886sec preferred_lft 310744886sec
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:bb:71:07:06 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global docker0
       valid_lft forever preferred_lft forever
```

本机回环地址

阿里云内网地址

docker0 地址

```
# 问题： docker 是如何处理容器网络访问的？
```

shell

```
# [root@kuangshen /]# docker run -d -P --name tomcat01 tomcat


# 查看容器的内部网络地址  ip addr  ，  发现容器启动的时候会得到一个 eth0@if262 ip地址，docker分配的！
[root@kuangshen /]# docker exec -it tomcat01 ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
261: eth0@if262: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:12:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.18.0.2/16 brd 172.18.255.255 scope global eth0
       valid_lft forever preferred_lft forever

# 思考。liunx 能不能 ping 通容器内部！
[root@kuangshen /]# ping 172.18.0.2
PING 172.18.0.2 (172.18.0.2) 56(84) bytes of data.
64 bytes from 172.18.0.2: icmp_seq=1 ttl=64 time=0.067 ms
64 bytes from 172.18.0.2: icmp_seq=2 ttl=64 time=0.055 ms

# linux 可以ping 通 docker 容
```

1、我们每启动一个docker容器，docker就会给docker容器分配一个ip，我们只要安装了docker，就会有一个网卡 docker0

桥接模式，使用的技术是 evth-pair技术！

9springboot微服务打包成docker

1.构建springboot项目

2.打包应用

3.编写dockerfile

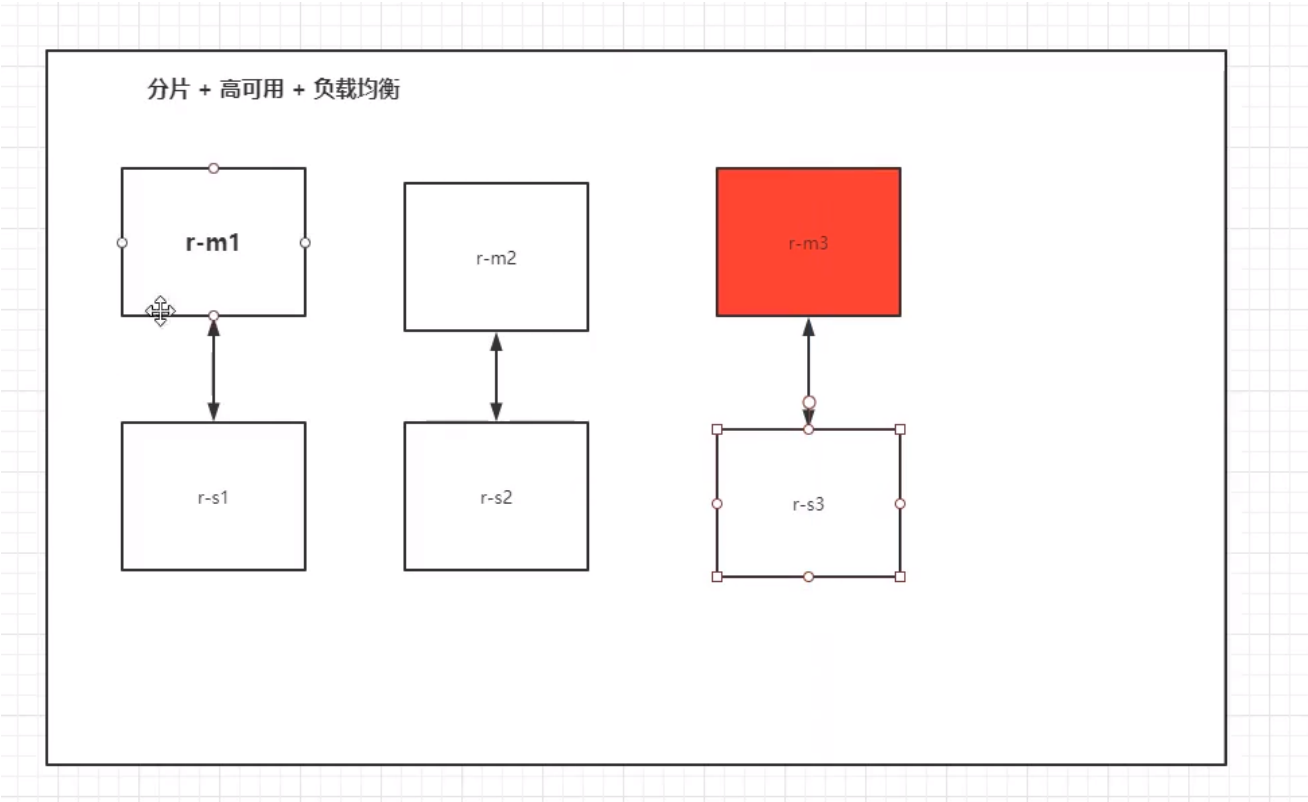
4.构建镜像

5.发布运行

```
FROM java:8

COPY *.jar /app.jar

CMD ["--server.port=8080"]

EXPOSE 8080

ENTRYPOINT ["java","-jar","/app.jar"]
```

实战部署

1.redis集群部署



shell脚本！

```
# 创建网卡
docker network create redis --subnet 172.38.0.0/16

# 通过脚本创建六个redis配置
for port in $(seq 1 6); \
do \
mkdir -p /mydata/redis/node-${port}/conf
touch /mydata/redis/node-${port}/conf/redis.conf
cat << EOF >/mydata/redis/node-${port}/conf/redis.conf
port 6379
bind 0.0.0.0
cluster-enabled yes
cluster-config-file nodes.conf
cluster-node-timeout 5000
cluster-announce-ip 172.38.0.1${port}
cluster-announce-port 6379
cluster-announce-bus-port 16379
appendonly yes
EOF
done

docker run -p 637${port}:6379 -p 1637${port}:16379 --name redis-${port} \
-v /mydata/redis/node-${port}/data:/data \
-v /mydata/redis/node-${port}/conf/redis.conf:/etc/redis/redis.conf \
-d --net redis --ip 172.38.0.1${port} redis:5.0.9-alpine3.11 redis-server /etc/redis/redis.conf; \


docker run -p 6371:6379 -p 16371:16379 --name redis-1 \
-v /mydata/redis/node-1/data:/data \
-v /mydata/redis/node-1/conf/redis.conf:/etc/redis/redis.conf \
-d --net redis --ip 172.38.0.11 redis:5.0.9-alpine3.11 redis-server /etc/redis/redis.conf
```

2.mysql安装

```
docker pull mysql:5.7


docker run --name mysql57 -e MYSQL_ROOT_PASSWORD=root -d -p 3306:3306 mysql:5.7
```

3.tomcat 安装

```
docker pull tomcat:8.0.15


docker run -d -p 8080:8080 --name mytomcat -v /mytomcat/webapps:/usr/local/tomcat/webapps
tomcat:8.0.1
```

```
cp -r webapps.dist/* webapps
```

4.nginx安装

```
#拉取镜像
docker pull nginx
#进入根目录，创建文件夹
cd /
mkdir -r /nginx/html
mkdir -r /nging/path

#创建临时nginx容器
docker run --name tmp-nginx-container -d nginx

#将容器内的nginx.conf复制到主机文件夹来
docker cp tmp-nginx-container:/etc/nginx/nginx.conf /nginx/path/nginx.conf

#删除临时nginx容器
docker rm -f tmp-nginx-container

#创建nginx容器，并挂载文件
docker run -d -p 80:80 --name mynginx -v /nginx/html:/usr/share/nginx/html -v
/nginx/path/nginx.conf:/etc/nginx/nginx.conf nginx

#查看容器
docker ps
```

5.redis 单机安装

```
docker pull redis

docker run -d -p 6379:6379 --name myredis redis
```

6.elasticsearch安装

```
docker pull elasticsearch


docker run -e ES_JAVA_OPTS="-Xms256m -Xmx256m" -d -p 9200:9200 -p 9300:9300 --name
myelasticsearch elasticsearch:5.6
```

7.RabbitMQ安装

```
#指定版本，该版本包含了web控制页面
docker pull rabbitmq:management

#方式一：默认guest 用户，密码也是 guest
docker run -d --hostname my-rabbit --name rabbit -p 15672:15672 -p 5672:5672 rabbitmq:management

#方式二：设置用户名和密码
docker run -d --hostname my-rabbit --name rabbit -e RABBITMQ_DEFAULT_USER=user -e
RABBITMQ_DEFAULT_PASS=password -p 15672:15672 -p 5672:5672 rabbitmq:management
```