Docker是一个轻量级的容器技术。Docker支持将软件编译为一个镜像（image），在这个镜像里做好对各个软件的各种配置，然后发布这个镜像，使用者可以运行这个镜像。运行中的镜像称之为容器container。容器的启动速度是非常快的。

Dock的使用：

使用oracle为数据库，安装redis作为缓存和NoSql数据库，安装mongoDb作为NoSQL数据库演示

安装Active MQ和RabbitMQ进行异步消息演示，基于Docker的spring boot部署

在linux安装Docker

在centos安装命令

Sudo yum update

Sudo yum install docker

启动docker

sudo systemctl start docker

设置docker开机自启

Systemctl enable docker

验证docker是否安装成功

docker run hello-world

1. Docker 镜像命令

Docker镜像检索

Docker search 镜像 检索redis docker search redis

镜像下载

Docker pull 镜像 下载redis docker pull redis

查看本地镜像列表

Docker images



REPOSTORY是镜像的名字；tag是软件版本，lastest是最新版本；IMAGE\_ID是当前镜像的唯一ID；Created 是当前镜像的创建时间；size是当前镜像的大小。

删除镜像

Docker rmi image-id

删除所有镜像

Docker rmi $(docker images –q)

1. Docker容器命令

运行镜像为容器的命令

Docker run –-name container-name –d image-name

运行一个容器只需通过运行docker run命令即可。其中—name的参数是为容器起的名字；-d表示detached，意味着执行完这个命令后控制台将不会被阻碍，可继续输入命令操作；最后一个image-name是镜像的名称，运行哪一个镜像作为容器。

比如说运行一个redis的镜像作为容器 docker run –-name redis-test –d redis

容器列表

Docker ps

CONTAINER ID 启动时生成的ID，image时该容器使用的镜像；comand时容器启动时调用的命令；created时容器创建的时间；status时容器当前的状态；ports是容器系统所使用的端口，redis默认使用6379端口；name时刚才定义的容器的名称。

查看运行和停止状态的容器

Docker ps -a

停止和运行容器

Docker stop container-name/container-id

例如docker stop redis-test

启动容器

Docker start container-name/container-id

例如 docker start redis-test

端口映射

映射容器的6379端口到本机的6378端口

Docker run –d –p 6378:6379 –name port-redis redis

删除容器

删除单个容器

Docker rm container-id

删除所有容器

Docker rm $(docker ps –a -p)

容器日志

查看当前容器的日志

Docker logs container-name/container-id

Docker logs port-redis

登陆容器

Docker exec –it container-id/container-name bash

查看容器端口信息 docker port container-name/container-id

云部署-----基于docker的部署

编译自己的docker镜像，并运行镜像的容器

Docker编译容器是使用dockerfile文件来编译自己的镜像的

Dockerfile的主要命令：

* 1. FROM 指令：FROM指令指明了当前镜像继承的基镜像，编译时当前镜像会自动下载基镜像，比如项目使用到java 8的镜像

FROM java:8

* 1. MAINTAINER指令：MAINTAINER指令指明了当前镜像的作者

MAINTAINER qinry

* 1. RUN指令：RUN指令可以在当前镜像上执行Linux命令并形成一个新的层，RUN是编译时（Bulid）的动作。

RUN /bin/bash –c “echo helloworld”或者Run {“/bin/bash”,”-c”,”echo hello”}

* 1. CMD指令：CMD命令指明了启动镜像是容器的默认行为。一个Dockerfile里只能有一个CMD指令。CMD指令里设定的命令可以在运行镜像时使用参数覆盖。CMD时运行时（run）的动作。

示例：CMD echo “this is a test”可被docker –run –d imag\_name echo “hello world”

* 1. EXPOSE指令

EXPOSE指明了镜像运行时的容器必须监听指定的端口

示例：EXPOSE 8080

* 1. ENV指令：ENV指令可用设置环境变量

示例：ENV myname=myname 或者 ENV myname myname

* 1. ADD指令：ADD指令是从当前工作目录复制文件到镜像目录中去。

示例：ADD test.txt /mydir/

* 1. ENTRYPOINT指令：ENTRYPOINT指令可以让容器像一个可执行程序一样运行，这样镜像运行时可以像软件一样接受参数执行，ENTRYPOINT是运行时（RUN）动作。

示例：ENTRYPOINT [“/bin/echo”]

我们可以向镜像传递参数运行：docker run –d image\_name “this is not a test”

1. 使用项目来演示docker项目部署
   1. 使用spring boot新建一个hello world项目
   2. 打包build生成dockertest-0.0.1-SNAPSHOT.jar
   3. 将jar放到指定目录/var/app目录下
   4. 当前目录下新建文件Dockerfile

FROM java:8

MAINTAINER qinry

ADD dockertest-0.0.1-SNAPSHOT.jar app.jar

EXPOSE 8080

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

输入以上信息。 上面信息解释

基镜像为java，版本为8

作者为qinry

我们将dockertest-0.0.1-SNAPSHOT.jar添加到镜像中，并重命名为app.jar

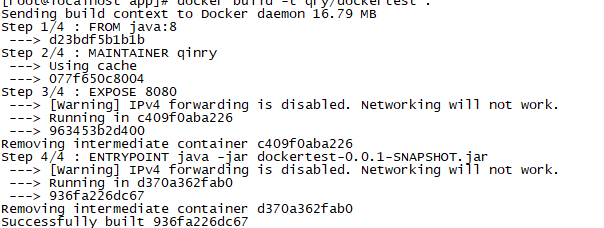
运行镜像的容器，监听8080端口

启动是运行java –java app.jar

* 1. 编译镜像

在当前目录下执行：docker build –t qry/dockertest .

其中，qry/dockertest为镜像名称，设置qry为前缀，也是docker的一种命名习惯。注意最后还有一个“.”，这是用来指明dockerfile路径的，“.”表示dockeflie在当前路径下。



出现IPv4 forwarding is disabled. Networking will not work.警告。

解决办法：

# vi /etc/sysctl.conf

或者

# vi /usr/lib/sysctl.d/00-system.conf

添加如下代码： net.ipv4.ip\_forward=1

重启network服务

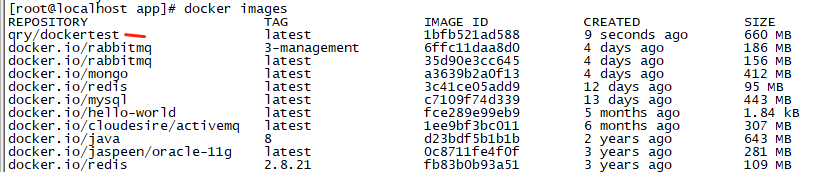
# systemctl restart network

查看是否修改成功

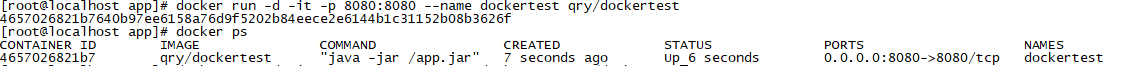
# sysctl net.ipv4.ip\_forward

如果返回为“net.ipv4.ip\_forward = 1”则表示成功了

* 1. 编译完成查看本地镜像：docker images



* 1. 运行镜像：docker run -d -it -p 8080:8080 --name dockertest qry/dockertest



启动成功，端口映射成功。访问<http://192.168.213.110:8080/>可以成功的访问。

1. 在使用redis或者mysql这些软件的时候可以登陆容器进行操作

使用如下命令登陆当前容器，登陆后可以进行常规的linux操作，可以使用exit推出登陆。

Docker exec –it container-id/container-name bash

如下是登陆redis的容器进行的操作。

