Lecture-1

Monday, October 5, 2020
13:50

1. Provisioning

It's a method to providing VMs with some application installation. It's a requirement of field company.
How we can optimize the server provisioning cost?
In present we have one VM for one application.

1. Why we are not putting multiple application on single OS?
2. How we can reduce overall OS cost in companies?
3. How we can reduce licensing cost of OS?

Below are the dependencies to run multiple App of single OS.

1. Library conflict
2. Env conflict
3. Soft dependency

Provisioning Method:

1) VM/Instance Based: This is example of hardware virtualization.
2) Container Based: This is example of OS base virtualization.

Docker: Docker is a community and It is Open source container technology.

2. Virtualization

Virtualization is a technology by which we can run multiple machines on a hardware simultaneously.

3. OS Virtualization

Docker Architecture

1. Docker Host/Engine: A machine/OS that will run multiple container machines.

OS: docker 1.13

Two release available in market:

Docker ce: Community Edition
Docker ee: Enterprise Edition (One month subscription is free with your ID)

Redhat also provide their customize docker solution.

1. Atomic host>>rhel7
2. podman>>rhel7.4 Later

2. Docker Client

Client machine from where we can provision containers. This is just like your client workstation and generally its not recommended to take docker client.

3. Docker Images

Its light weight image for particular application. To spin the container you should have at least available. It has minimal configuration to deploy any container.

Two types of images available in market.

1. Community based

There have some communities available in market and they are providing free of cost images.
For example: docker hub
Free available: CE

Red hat also provide their docker images.
but need subscription: registry.access.redhat.com

2. Custom Image

We can configure our offline Docker registry and we can import some ready to use images.

4. Docker Registry

From where we can pull the container images.

5. Docker Container

It's a micro machine running over the docker host.
its have own process/storage/IP/NIC.

Note: We can deploy multiple container using same docker image.

Lecture-2

Monday, October 5, 2020
13:52

Lab-1: How to Install Docker community addition

Docker CE:

two versions available for docker-ce.

1. Edge: 2 months (Testing)
2. Stable: Production

Step1: Create one centos machine in AWS Cloud.

Login to machine by using the public IP.

Step2:

```
# yum install -y yum-utils
# yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
# ls -lrt /etc/yum.repos.d/
# yum install docker-ce -y [docker-ce will install docker client/command also ]
# docker version
# systemctl start docker [to show docker server version]
# systemctl enable docker
# docker version
# docker info\
][o
```

Step3: Docker Image commands

```
# docker images
# docker ps
# docker search mysql
# docker pull mysql:latest
# docker pull mysql:5.6
# docker images
# docker pull nginx:latest
# ls -ld /var/lib/docker
# docker pull centos:7
# docker images
# docker history b5b4d78bc90c [to check the history of docker image]
```

We have two types of images here.

1. With Service
2. Without service

Step4: Create first container by using centos image

```
# hostnamectl set-hostname docker-host
# bash
# docker run -it --name=test centos:7
/]# touch abc
```

/]# exit
#docker ps **[you will not see any container listed here]**
#docker ps -a **[to list all containers]**

Note: Open a duplicate session

docker ps
docker top test **[to check the all process of specific container]**
docker start test
docker attach test **[to access shell of the container]**

ctrl+pq [quit from container without stopping it]

docker exec -it test /bin/bash **[this will start the new process]**
docker ps **[on docker host]**

docker stop test
docker stats test **[to show resource utilization of container]**
docker system df **[Docker hosts resource utilization]**
docker start test
docker attach test
dd if=/dev/zero of=abc1 bs=1M count=2048
docker system df
docker rm test **[to delete container]**
docker inspect test | grep -i ipaddress **[to check IP address details for container]**
docker inspect network bridge | grep -i subnet

How to run service packed container node

docker images
docker run -d --name=webserver nginx:latest **[we can also use image ID]**
docker top webserver
docker exec -it webserver /bin/bash
docker inspect webserver | grep -i ipaddress
curl 172.17.0.2

docker pull mysql:5.6
docker pull centos:7
docker pull nginx:latest
docker images

Note: Here centos is raw image because no service is installed inside this image.
nginx and mysql are service images.

[root@ip-172-31-13-80 ~]# docker run -it --name=con1 centos:7
[root@7960a29d6eaf /]#

[root@ip-172-31-13-80 ~]# hostname docker
[root@ip-172-31-13-80 ~]# bash
[root@docker ~]# docker ps

[root@7960a29d6eaf /]# touch abc
[root@7960a29d6eaf /]# exit
exit
[root@ip-172-31-13-80 ~]#

[root@docker ~]# docker ps -a

```
[root@ip-172-31-13-80 ~]# docker start con1
con1
[root@ip-172-31-13-80 ~]# docker attach con1
[root@7960a29d6eaf /]#

[root@ip-172-31-13-80 ~]# docker stop con1    [To stop container]
con1
[root@ip-172-31-13-80 ~]# docker rm con1    [To delete container]
con1

[root@ip-172-31-13-80 ~]# docker run -d --name=test nginx:latest    [Without interaction mode]
[root@ip-172-31-13-80 ~]# docker exec -it test  /bin/bash
root@d49f3d2c782a:/# exit
exit

[root@docker ~]# docker ps
```

####Logs file of docker####

```
[root@ip-172-31-13-80 ~]# docker run -d --name=db mysql:5.6
[root@ip-172-31-13-80 ~]# docker ps
[root@ip-172-31-13-80 ~]# docker ps -a

[root@ip-172-31-13-80 ~]# docker logs db
```

Note: How to pass variables while launching container.

```
[root@ip-172-31-13-80 ~]# docker run -d --name=db -e MYSQL_ROOT_PASSWORD=redhat mysql:5.6
6baaabed1a265fe52f4c013da9b2c6134d2c54bc21d448be9756c33480b93556
[root@ip-172-31-13-80 ~]# docker ps

[root@ip-172-31-13-80 ~]# docker inspect db | grep -i ipaddress
root@6baaabed1a26:/# mysql -u root -p
[root@ip-172-31-13-80 ~]# docker stats db

[root@ip-172-31-13-80 ~]# docker run -d --name=db -e MYSQL_ROOT_PASSWORD=redhat --cpu=0.5  mysql:5.6
```

###How to Connect two containers###

```
[root@ip-172-31-13-80 ~]# docker run -d --name=database  -e MYSQL_ROOT_PASSWORD=redhat  mysql:5.6
#docker ps
#docker inspect database | grep -I ipaddress
#docker pull wordpress
#docker images
#docker run -d --name=application -e WORDPRESS_DB_HOST=172.17.0.2:3306 -e WORDPRESS_DB_PASSWORD=redhat
#docker ps
#docker exec -it database /bin/bash
/]# mysql -u root -p
:redhat
>show databases;
>exit;

#docker run -it --name=demo --restart=always centos:7
```

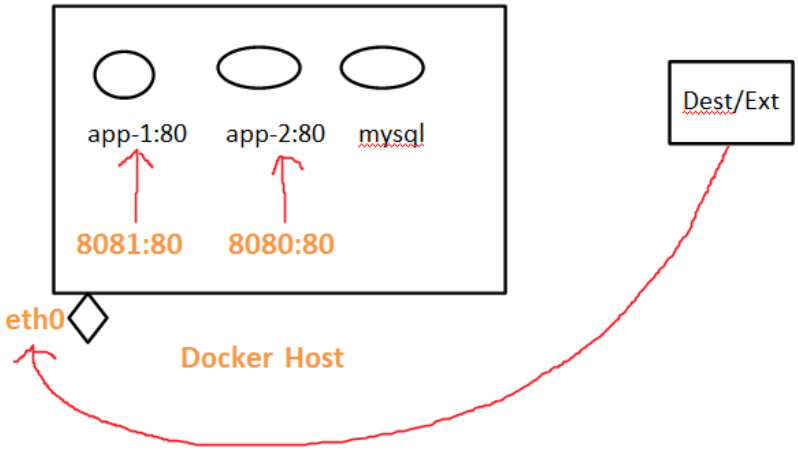
Note: Whenever we give restart to docker server on docker hosts all container will go shutdown if we will not set this option .

Container will come up automatically.

#docker start $(docker ps -q -a) **[To start all container in one go]**

###Expose containers for world Network###

Note: By default container can not be access from external machine. It is accessible from docker host only. You need to create port forwarding for container from docker host.



#docker images
#docker run -d -p 8080:80 --name webserver1 nginx:latest
#docker ps
#netstat -tunlp | grep -w 8080

Note: Now try to open public IP of EC2 server in browser. <public ip:8080>

#docker run -d -p 8081:80 --name=webserver2 nginx:latest
#docker ps
#ifconfig eth0
#docker run -d -p <ip>:8082:80 --name=webserver3 nginx:latest
#curl 127.0.0.1:8082

Note: To persist port for specific container or IP binding

#docker ps
#docker run -d -p 8083:80 -p 8084:80 --name=webserver4 nginx:latest
#docker ps
#docker stop $(docker ps -q -a)
#docker rm $(docker ps -q -a)
#docker run -d -P --name=webserver5 nginx:latest **[To allocate automatic port]**

###Build Docker Image###

#docker image

Note: Without provision container we can not modify image.

#docker run -it --name=test centos:7
/]#echo "helloworld" > run.sh
/]#ls [ctrl+pq]

Note: Launch new container using the same image and check run.sh file inside new container, you will not get file inside the container because image is totally isolate.

#find /var/lib/docker --name run.sh **[Container writing layer]**
#docker ps
#docker commit test centos:v1
#docker images
#docker run -it --name=demo centos:v1
/]#ls -l run.sh
#docker stop $(docker ps -q -a)
and
rm
Note: Now launch one raw container by using centos image and install the http service inside this and create your custom and try this image launching new container.

#docker commit -m "apache installed" apache centos:apache
#docker history <image ID>
#docker ps
#docker rmi <centos> **[Base image ID]**

Note: New image has dependency on base image that's why can't delete.

#docker save centos:apache > apache.tar
#docker rmi -f $(docker image -q)
#docker images
#docker load -i apache.tar
#docker images
#docker pull centos:7 **[This pull will not work]**
#docker images
#docker history <image ID>

###Docker Image/Container File System###

Docker uses overlay2 filesystem by default.

#docker info

Storage driver: overlay2

#ls /var/lib/docker/overlay2
#docker pull centos:7
#ls /var/lib/docker/overlay2
#docker images inspect docker.io/centos:7 | less
#docker run -it --name=test centos:7
/]#ls

Note: Create one file on image overlay path and check the file inside container.

/]#touch xyz
"OverlayFS"
#df -TH /

Note: Inside overlay2 we have two layer path one for image and one for container.
Create one file inside container and check on container layer path inside diff folder and file should be available. But will not available inside the image layer path.

Lecture-3

Monday, October 5, 2020
13:52

###Docker Volume###

We need volume to store the application/user data.

```
#docker pull mysql:5.6
#docker images
#docker run -d --name=test -e MYSQL_ROOT_PASSWORD=redhat mysql:5.6
#docker exec -it test /bin/bash
/]#mysql -u root -p
:redhat
>create database abc;
>show databases;
>exit;
```

Note: We writing data to root file system of container.

```
#docker rm -f test
#mkdir /volume
```

Note: /var/lib/mysql **[container mount point]**

```
#getenforce
#setenforce 0
#docker run -d -v /volume<docker host path>:/var/lib/mysql<container path> --name=db -e MYSQL_ROOT_PASSWORD=redhat
mysql:5.6
#docker ps
#docker exec -it db /bin/bash
#mysql -u root -p
:redhat
>create database test;
>exit;
#ls /volume
#docker rm -f db
#ls /volume
#docker run -d -v /volume:/var/lib/mysql --name=db -e MYSQL_ROOT_PASSWORD=redhat mysql:5.6
#docker exec -it db /bin/bash
/]#mysql -u root -p
:redhat
>show databases;
>exit;
#setenforce 1
#getenforce
#mkdir /data
#ls -ld /data
#docker run -it --name=con1 centos:7
/]#ls -ldZ  /opt
/]#exit
#docker rm -f con1
#ls -ldZ  /data
#chcon -Rt container_share_t /data
#docker run -it --name=con1 -v /data:/opt centos:7
```

/]#touch /opt/abc
/]#exit
#ls /data **[context should be match]**

###Another Way to provision storage###

#docker rm -f $(docker ps -q -a)
#docker system prune
[y]
#docker images
#docker volume ls
#docker volume create mysqlvol
#docker volume ls
#ls -ld /var/lib/docker/volumes
#docker volume inspect mysqlvol
#docker run -d -v mysqlvol:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=redhat mysql:5.6
#ls /var/lib/docker/volumes/mysqlvol/_data
#docker stop $(docker ps -q -a)
and
rm
#docker images
#docker volume ls
#docker volume rm mysqlvol
#ls /var/lib/docker/volumes

#docker run -d --name=db -e MYSQL_ROOT_PASSWORD=redhat mysql:5.6
#docker volume ls
#docker pull nginx
#docker run -d --name=webserver nginx
#docker volume ls **[search on google mysql docker images]**

#ls /var/lib/docker/volumes
#docker rm -f db
#docker volume ls
#docker system df
#docker volume create vol1
#docker volume ls
#lsblk

Note: Mount a new 2GB ebs volume inside EC2 server.

#setenforce 0
#docker run -it -v /vol1:/opt --name=testcon centos:7
/]#cd /opt
opt]#dd if=/dev/zero of=abc1 bs=1m count=4000

Note: No space left on device.

opt]#exit

#docker volume ls
#docker system df
#docker system prune

[y]

#docker stop $(docker ps -q -a)
and
rm

Lecture-4

Monday, October 5, 2020
13:53

###Docker Image File###

Types of images build process.

1. Manual
2. Docker File

#docker images

Note: On google search "nginx docker image" and copy the GIT URL of the image and download it over docker host.

#yum install git -y
#git clone https://github.com/nginx/docker-nginx.git
#ls
#cd docker-nginx
docker-nginx]#ls
#cd mainline/stable
#cat DockerFile
docker build -t nginx:latest
#docker images

###How to write Docker File###

Below are the docker parameters.

1. FROM: It always contains base image reference to building new images.
2. RUN: What command you want to execute to provision new image.
3. ADD: You want to copy some files into image. Can download content from the internet.
4. COPY: Copy also do same thing like ADD Parameter. Can't download from internet, Only add from localhost.
5. EXPOSE: To set default container port to access your application.
6. WORKDIR: To change the present working directory when login to container first time.
7. ENV: To store the variable inside the docker file and its limited to docker file only.
8. CMD: Any process/service daemon which you want to start automatically when you provision the container. we can't use multiple time in a single file. If defining multiple time last CMD command will execute.
9. ENTRYPOINT: It is basically to execute shell file on creation/booting time of container. Can't call multiple time.
10. MAINTAINER: Optional parameter (Email, author name and Department)
11. USER: Default conatiner namespace ownership user and login user also when access container.
12. VOLUME: To creating one volume automatically for the container and mounting inside the container.
Note: If using CMD and Entrypoint both, entrypoint will have priority.

#mkdir test
#cd test
#echo helloword > run.sh
#vim DockerFile

FROM centos:7
MAINTAINER shyam@gmail.com "it's a practice image"
RUN yum install httpd -y
ADD https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm /tmp
COPY run.sh /tmp

:wq!

#docker build -t centos:apache
#docker images
#docker run -it --name=demo centos:apache
/]#rpm -qa httpd
/]#cat /tmp/run.sh
/]#exit
#docker rm -f demo

Note: RUN Parameter format
RUN command1
RUN command2
RUN command3

To decrease command history , use below format for RUN parameter.

#mkdir src
#echo website > src/index.html
#vim DockerFile

FROM centos:7
ENV a 10
RUN echo $a > /var/tmp/abc
ENTRYPOINT "abc.sh"
MAINTAINER shyam@gmail.com "it's a practice image"
RUN yum install httpd -y && echo "Helloworld" > /tmp/report \
&& date && cal \
WORKDIR /var/www/html
COPY ./src/ .
RUN 'sed -i "s/Listen 8080/Listen 8080/g" /etc/httpd/conf/httpd.conf' \
&& chown -R apache:root /var/log/httpd /var/run/httpd \
&& chmod -R 777 /var/log/httpd /var/run/httpd
EXPOSE 8080
USER apache
ADD https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm /tmp
COPY run.sh /tmp
CMD ["/usr/sbin/httpd", "-D", "FORGROUND"] **[we can not provision container with -d option without this parameter you need to -it option]**
VOLUME /volume [container mounting point]
:wq!

#docker build -t apache:latest .
#docker run -d -p --name=web apache:lastest
#docker exec -it web /bin/bash
bash-4.2$whoami
bash-4.2$pwd
bash-4.2$cat /var/tmp/abc
bash-4.2$ls
bash-4.2$curl 127.0.0.1:8080
bash-4.2$exit
#docker ps
#docker volume ls
#ls /var/lib/docker/volume

#docker run -d -p --name=webserver nginx
#docker ps

Note: You will get default port 80 which is already defined inside the docker file.

#docker run -it --name=test centos:7
/]#whoami
root **[Default username]**
#docker search ramprakashupadhyay123
#docker run -d -p --name=myweb ramprakashupadhyay123/openshift
#docker ps

Note: Default port is 8080 here.
#docker exec -it myweb /bin/bash
bash-4.2$whoami
bash-4.2$id apache
bash-4.2$exit
#ps -aef

Note: You will get user ID here.

Container Process Mode:

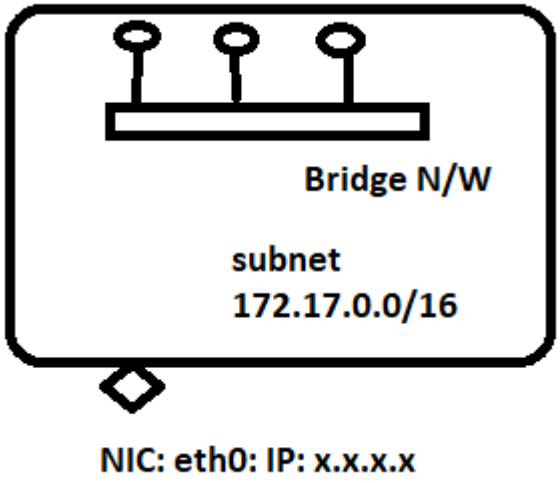
1. root mode: Privileged mode
2. normal mode: Non-priviledge mode

Lecture-5

Monday, October 19, 2020
23:05

###Docker Network###

- How containers communicate internally.
- Docker containers by default works on bridge network concept.
- **Bridge:** It is separate network for container.
- Whenever we start docker daemon on docker host. It create bridge by default.



docker network ls

Note: But all container communicate via bridge within docker host.

docker network inspect bridge | grep -i subnet

Ext Port: docker0

ifconfig docker0
docker run -it --name=con1 centos:7
/]# ctrl+pq
docker inspect con1 | grep -i ipaddress
ping 172.17.0.2
docker run -it --name=con2 cento:7
/]# ping 172.17.0.2
/]# ping google.com

Note: By default containers uses NAT feature to reach internet via docker host network.

/]# yum install net-tools -y
/]# ifconfig eth0

/]# ctrl+pq
docker network inspect bridge

#####How to Setup Custom Bridge#####

docker network create dev --subnet 192.168.10.0/24 --driver bridge dev
docker network ls
docker run -it --name=con3 --network dev centos:7
/]# hostname -i
/]# ping 172.17.0.2 **[default bridge container IP]**
/]# ctrl+pq
docker network inspect bridge | grep -i name
docker network inspect dev | gerp -i name
docker network connect bridge con3
docker attach con3
/]# hostname -i
/]# ctrl+pq
docker network disconnect bridge con3
docker run -it --name=con4 --ip 192.168.10.10 --network dev centos:7
/]# hostname -i
ctrl+pq
docker run -it --name=con5 --hostname web --ip 192.168.10.20 --network dev centos:7
/]# hostname -i

Note: Docker works on two types of networks by default.

1. **Bridge network:** Container used either default or custom bridge for the communication.
2. **Host network:** Container doesn't create network namespace. It directly used docker host adapter.

netstat -tunlp | grep -w 80
docker run -d --name=webserver1 --network=host nginx:latest
netstat -tunlp | grep -w 80
docker exec -it webserver1 /bin/bash
/]# hostame -i
/]# ctrl+pq
hostname -i

docker run -d --name=webserver2 --network=host nginx:latest
docker logs webserver2 **[Container will not provision]**
docker ps -a
cat /etc/docker/daemon.json

Lecture-6

Monday, November 9, 2020
8:04 PM

#####Docker Compose#####

It's a method to automate provisioning of conatiners.

curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /bin/docker-compose
chmod u+x /bin/docker-compose
docker-compose version
mkdir test
cd test
vim docker-compose.yaml

```
version: '3'
services:
      web:
         image: nginx
         ports:
          -  8080:80

:wq!
```

docker-compose up -d
docker ps

vim docker-compose.yaml

```
version: '3'
services:
      web:
         image: nginx
         ports:
          -  8080:80
        apache:
        image:nginx
        ports:
         - 8081:80
:wq!
```

docker-compose up -d
docker ps
docker-compose ps
docker-compose stop **[Need to execute command from the compose file path]**
docker-compose rm
docker-compose ls
#cd ; mkdir wordpress
cd wordpress
vim docker-compose.yaml

```
version: '3.3'
       services:
         db:
          image: mysql:5.7
          volumes:
           - db_data:/var/lib/mysql
          restart: always
```

```
          environment:
            MYSQL_ROOT_PASSWORD: somewordpress
            MYSQL_DATABASE: wordpress
            MYSQL_USER: wordpress
            MYSQL_PASSWORD: wordpress

wordpress:
    depends_on:
     - db
    image: wordpress:latest
    ports:
     - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress

volumes:
   db_data: {}

:wq!

# docker-compose up -d
# docker network ls
# docker-compose ls
# docker-compose volume ls
# docker-compose stop
# docker-compose rm
# docker inspect network wordpress_default | grep -i subnet
# docker-compose kill       [Shutdown and kill]
# docker network ls
# docker network rm wordpress_default
# docker-compose down       [Everything will be deleted]
# vim docker-compose.yaml


version: '3.3'
       services:
         db:
           image: mysql:5.7
           volumes:
             - db_data:/var/lib/mysql
           restart: always
           environment:
             MYSQL_ROOT_PASSWORD: somewordpress
             MYSQL_DATABASE: wordpress
             MYSQL_USER: wordpress
             MYSQL_PASSWORD: wordpress
           networks:
             - abc

wordpress:
    depends_on:
     - db
    image: wordpress:latest
    ports:
     - "8000:80"
    restart: always
```

```
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    networks:
      - abc
volumes:
   db_data: {}
   xyz: {}
networks:
       abc:
         driver: bridge

:wq!

# docker-compose up -d
# docker network ls
# vim Docker-File

FROM centos:latest
RUN yum install  httpd -y
RUN sed -i "s/Listen 80/Listen 8080/g" /etc/httpd/conf/httpd.conf
COPY src/  /var/www/html
RUN chown apache:apache /var/run/httpd /var/log/httpd
RUN chmod -R 777 /var/run/httpd /var/log/httpd
EXPOSE 8080
USER apache
CMD ["/usr/sbin/httpd", "-D", "FOREGROUND"]

:wq!

# mkdir src
# echo helloworld > src/index.html
# ls
# vim docker-compose.yaml

version: '3'
services:
    webserver:
      build: .
      port:
       - 8089:80

:wq!

# docker-compose up -d
# docker ps

# docker-compose up -d --build    [This will again recreate the container]

Note: If you have change the website content then this option is useful.

# curl 127.0.0.1:8089
```

Lecture-7

Monday, November 9, 2020
9:51 PM

Docker-Registry#####

It's the location to store the containers images.

1. By default repository is public so anyone can access your image.
2. To push the images you should have account to docker hub.

Registry Servers.

a. Community based [redhat, docker.io/docker.hub]
b. Custom registry/offline

docker search nginx
docker search ram123
docker pull ram123/openshift

Note: Without password we can pull the image but can not push.

1. **Login to docker hub and create repository.**

docker images
docker tag <image-ID> docker.io/ram123/jboss:latest
docker images
docker push ram123/jboss:latest

Note: You will get the error.

docker login

Username: ram123
Password:

docker push ram123/jboss:latest

Note: Check image inside your repository.

docker save <image-ID> apache.tar
ls
docker load -i apache.tar

docker search ram123

#####How to install offline docker registry server#####

There have two ways to setup offline docker registry server.

1. Complete VM reserve
2. Container based registry server

Step1: Create one centos machine in aws cloud.
Step2: Login to your machine.

```
# hostname registry-server
# bash
# yum install docker-distribution -y
# systemctl start docker-distribution
# systemctl enable docker-distribution
# netstat -tunlp | grep -w 5000
# ls -ld /var/lib/registry
# ifconfig eth0
```

On Docker host

```
# docker tag <image-ID> <ip-registry-server>:5000/apache:latest
# docker push <ip-registry-server>:5000:latest
```

Note: You will get the error because of unsecure registry which is running on by default http based.

```
# vim /etc/docker/daemon.json

{"insecure-registry": ["<ip-registry-server>:5000"]}

:wq!

# systemctl restart docker
# docker push <ip-registry-server>:5000:latest
```

On Registry-Server

```
# ls /var/lib/registry/docker/registry/v2/repositories
# ls
```

On Docker Host

```
# docker pull <ip-registry-server>:5000/apache:latest
```

#####How to install docker registry as a container#####

On Registry-Server

```
# systemctl stop docker-distribution
# systemctl disable docker-distribution
```

```
# yum remove docker-distribution -y
# yum install docker -y
# systemctl start docker
# systemctl enable docker
# docker pull registry:2
# docker images
# docker run -d -p 5000:5000 --name=registry-server registry:2
# netstat -tunlp | grep -w 5000
```

On Docker Host

```
# docker psuh <ip-registry-server>:5000/apache:latest
```

On Registry Server

```
# docker exec -it registry-server /bin/bash
/]# ls /var/lib/registry/docker/v2/repositories
```

Note: This is non-persistent storage if you container will loss, images will also delete.

```
# docker volume create vol1
# docker run -d -v vol1:/var/lib/registry -p 5000:5000 --name=registry-server registry:2
```

Note: This is now persistent

#####How to install https based registry server#####