28/03/22

DOCKER :- creating light weighted machines.

With linux.

create an instance (Amazon).

Connect to the instance

Switch to the root

* Yum install docker.
* service docker status.
* service docker start.
* service docker status.

→ first to create a docker container we need a docker image

→ To search images from 'CLI' side we have command

ex:- docker search hello-world

→ To search image from 'GUI' side we have 'docker hub'

→ To take the image into the machine (docker host).

* docker pull hello-world

→ To see the image in the docker host.

* docker images.

→ NOW, after the creation of image we can create the containers.

* docker run hello-world (Default).

if we have a version then.

* docker run hello-world : V1

⇒ Now, container is created (with Pull and Run command).

⇒ When executed a 'Run' command it always creates a New container

⇒ To see the newly created container which are running.

* docker ps

⇒ To see both running and stoped containers

* docker ps -a

∴ container created from already created image with commands (pull and run).

⇒ The other way "direct container creating".

* docker run (image name). Ex:- docker run Ubuntu.

The number of time we run the 'Run' command the number of container going to created in background

But, here the container exited.

* docker --help (for help from docker command)

To, keep the container up and running. (or) to connect

* docker run -it ubuntu

  -i - interactive mode. (ubuntu = apt).

  -t - terminal.

Now, we are in ubuntu instance.

  ls

  exit.

Now, back to EC2-instance.

* docker ps.

⇒ we don't have the machine in running.

* docker ps -a.

⇒ container is Exited.

* docker run -it ubuntu. (A New container).

Here, if we want container not to exited then press.

  control + P + q.

* container will be active

[If we exit container will be gone.

  If we run control + p + q then still active].

  docker ps.

⇒ To

  docker exec -it (Container Id) ls

⇒ With exectuk command we can also connect to the container

* docker exec -it [container id] /bin/bash.

⇒ create the container with "run" command

⇒ keep the container up and running with "-it" option

⇒ once connected to container to keep the container alive control +p+q.

⇒ TO connect to old containers with are running then "exec" command.

⇒ "exec" command purpose is to execute the command in the container (or) we can use shell commands also.

There's another way to keep container alive "ditach or demond" mode

* docker run -dit (image name).

* docker ps

Now to connect

docker exec -it (container id) /bin/bash.

exit.

docker ps

[If we use -d "demon mode we don't have to use control+p+q) to keep the container alive because "-d" will keep it alive. even though we exit)

df -h (to check the size of instance).

To clean-up unwanted containers and images.

docker rm (Container id).

first stop the container and the remove the container.

docker stop (Container id).

docker ps

docker ps -a

docker rm (container id)

docker ps -a

(or) Forcefull removal

docker rm -f (container id)

[best practice is to stop container and then remove it]

In single shot we can remove all containers.

docker rm -f $(docker ps -a)

To delete the image.

docker rmi (image name).

ex:- docker rmi ubuntu

[always best practice is to remove the containers then remove images].

→ Forcepull removal of images

* docker rmi -f ubuntu.

Single shot removal of all images.

* docker rmi -f $(docker images)

Now, TO work with servers like, Apache, nginx, Tomcat.

* docker run -dit nginx
* docker ps
  (nginx is running).

Now, to access nginx, open the ports

Now, nginx is running in a container in the docker.

⇒ Go to ports and open for all traffic and Ipv4 in ~~security~~ security wizard.

⇒ Whenever working with container ~~if~~ if creating any services with the container then the whatever the port there for container should be exposed, then only we can access it.

docker run -dit -P 80:80

-P - port expose external : internal nginx.

external → for access

internal → container servia port

* docker ps

If we want one more nginx machine.

* docker run -dit -p 81:80 nginx.

↓

different port number.

29/03/22

→ we want to know more information of container. (like ip address)

* docker inspect (container id)

→ To check the logs (or) to trouble shoot the containers.

* docker logs (container id)

→ To know current working directory. (CCN = container name)

* docker exec it (CCN id) pwd

→ To change name of current working directory.

* docker run -dit -w /(name) nginx.

ex:- docker run -dit -w /sankar nginx.

→ will create new container with /sankar pwd.

→ To check the environment variables in container

docker exect -it (CN id) env.

***

→ To create environment variables. (-e, --env, (or)
--env-file)

docker run -dit --env JAVA_HOME=/usr/share/Java/jdk-1.8

--env JRE_HOME=/usr/share/java/jre

nginx.

Now, variable created in a New container

To check env

docker exec -it (CN id) env.

We can also create list and can place the variables in containers.

* vi env.list

→ place the variables.

ls.

* docker run -dit --env-file (file name) (nginx).

* docker run -dit --env-file (env.list) nginx.

docker ps.

Now, see

docker exec -it (CN) env.

docker ps.

→ To Give name to container

Instead of taking container id we can take name for container

(option) (Name given) (image)

* docker run -dit --name nginx-ctr nginx

[--name is the option to create name.
nginx-ctr is the example provided name].

* docker log nginx-ctr

To connect to the container

docker exec -it (container id) /bin/bash.

ls
cd usr/
cd share/
cd nginx/
pwd
ls
cd html/
ls
cat index.html/

⟹ Here, we can see the launch page of the nginx container

Now, we can overwrite it with our content

echo "welcome to Docker" > index.html

cat index.html

⇒ /usr/share/nginx/html is the deployment location on nginx side

exit

docker ps

Now, remove the container

docker rm -f (CN id) (container is removed)

docker ps

docker ps -a

Now, if we want nginx back

Sy:- docker run -dit -p 80:80 ~~nginx~~ --name nginx-ctr1 nginx.

* docker run -dit -p 80:80 --name (name) (image).

we got nginx server again, but the data which is overwritten nginx ("welocm to docker") is gone.

So, to manage the data we have volumes.

⇒ To manage the Data we have volumes. this volumes are hosted by ec2 instance (docker host).

⇒ docker volume ls.

⇒ To see the list of volumes

⇒ To creati volume

docker volume creati (volume Name).

ex:- docker volume (my-vol).

⇒ To know the more information about the volume where it mounted (placed).

docker inspect my-vol

docker inspect (VN).

To attach this volume to another new container.

docker run -dit -p 81:80 -v (VN): (location) (image).

docker run -dit -p 81:80 -v my-vol:/usr/share/nginx/html/ nginx.

Now, go to location and see

cd /var/lib/docker/volumes/my-vol/_data

ls

cat index.html

pwd.

ls

docker ps

Now,

Change the (welcome page)

docker exec it (CN id) /bin/bash (connect to container)

cd /usr/share/nginx/html/

ls

echo "welcome to Docker" > index.html

cat index.html

exit

prod

ls

cat index.html

Now, we can change from ec2-instance has well

prod

vi index.html

→ change the file

Now clean up all container

docker rm -f $(docker ps -a)

docker ps

docker run -dit -p 80:80 -v my-vol:/usr/share/nginx/html/ nginx.

docker ps.

⇒ To remove the volumes:

docker volume rm (volume name).

docker volume rm my-vol.

→ To remove volume, first remove containers.

docker ps -a.

docker ps rm -f $(docker ps -a)

docker volume ls.

docker ps -a.

docker volume rm my-vol (volume removed).

⇒ To remove dangled and unused space in container.

* docker system prune.

⇒ To remove unused volumes.

* docker system prune --volumes.

⇒ To know the disk space utilized by docker.

** docker system df

docker system df

If you also want to delete unused containers & images.

* docker system prune -a

Commit container :- (creating a custom image) (same as Tomcat image)

To create custome image (commit)

* docker commit (container id) ubuntu-git-apache

* docker image

Now, whenever we want this custom image we can use this

* docker run -dit ubuntu-git-apache

* docker ps

To check history

* docker history ubuntu-git-apache

→ To share this custom images to team

Export container

* docker export (CN id) > ubuntu-git-apache.tar

first save as tar file

ls

after recieving the tar file now import this file to use it (Container name)

docker import new-uga < Ubuntu-git-apache.tar

Export image

* docker save -o ubuntu-git.tar ubuntu-git-apache

ls

* docker load < ubuntu-ag.tar

Container side :- Export & import

Image side :- save & load.

Copying the data :-

* docker exec -it (CN id) ls /opt (first be in the location you want to copy to)
* docker cp (source) (destination): location
* docker cp Env.list (CN id):/opt

Interview Q/A

docker top (CN id) - TO know process of container CN. name.

docker stats - TO check stats (resource).

docker system ps - space checking.

docker log - TO check logs.

To share images and container there's another way

create docker.hub account.

Now, connect the EC2-instance to docker.hub.

* docker login.

⇒ To connect to docker.hub

docker tag (image name):latest (account details)/: version
image name.

docker image

⇒ there, new image is created.

Now, push the image.

* docker push docker.io/(account name)/my image:latest

30/03/22

Word press :-

It is a combination of linux, mysql, apache

docker run --name wordpresssql -e MYSQL_ROOT_PASSWORD=password
-d mysql:5.5

docker ps

docker run --name mywp --link wordpresssql:mysql -P -d wordpress

→ one container linking to another container "--link"

docker ps

--link to connect one container to another

Dockerfile:- (Maintained by developers).

When creating a Dockerfile "From" is mandatory

ii, Maintainer

iii, Run

iv, User

v, Volume

vi, Env

vii, Entrypoint:- executed when the container is running.

viii, Expose :- exposing the ports.

ix, Add/copy :- Both add and copy will copy the content but, add will not only copy but will download the content.

x, WORKDIR :- default is /

xi, CMD :-

```
mkdir hello-world
ls
cd hello-world/
ls
vi Dockerfile
```

⇒ paste data from pdf

```
docker run hello-world
```

ls

cat Dockerfile

vi Hello.Java

⇒ Paste code from Pdf.

ls

⇒ whenever you build dockerimage alongs with docker file we need code aswell [ex:- Dockerfile Hello.java].

⇒ When you want create an image, you should be in Dockerimage location.

⇒ To create an image using dockerfile.

* docker build -t (image name) (file path).

ex:- docker build -t javaapp-image .

docker image

Similarly, service apache.

mkdir apache

cd apache/

ls

vi Dockerfile.

⇒ Paste the code from pdf

vi index.html

⇒ paste from pdf

ls

⇒ Dockfile index.html

docker build -t apache-image .

⇒ failed ←

vi Dockefile

(update with another data from pdf)

* docker build -t my-apache2 .

* docker run -dit -p 89:80 my-apache2:latest

* Difference b/w entrypoint & CMD

This will be executed at the time of container creation

Entrypoint won't accept any overwrite

CMD will allow the overwrite

31/03/22

* What is the concern you recently faced?

⇒ We got a concern build and deployment are taking time so asking to reduce deployment and build time there we have gone through all steps, there we found that Docker image building is taking lot of time, so for every change it's taking time for downloading the dependency freshely, so, we updated the Dockerfile until unless there's change in dependency it should take from cache not from community. So, now it's not going to download dependency for every change; if there's change in dependency it will download otherwise it will build the code only. Hence we reduced time for build and deploy.

(even when there's no change in dependency)

Docker echo system :-

Docker compose :-

Install docker compose :-

* Sudo amazon-linux-extras install epel
* Yum install gcc python-devel krb5-devel krb5-workstation
* Yum install python-pip -y
* Sudo pip install docker-compose
* Sudo yum upgrade python*

Adding user

usermod -ag docker $(whoami)

Create a New machine :-

Yum install docker docker-registry
usermod -ag docker $(whoami)
systemctl enable docker
systemctl start docker
systemctl status docker

Docker-compose setup :-

sudo amazon-linux-extras install epel.

Yum install gcc python-devel krb5-devel krb5-workstation

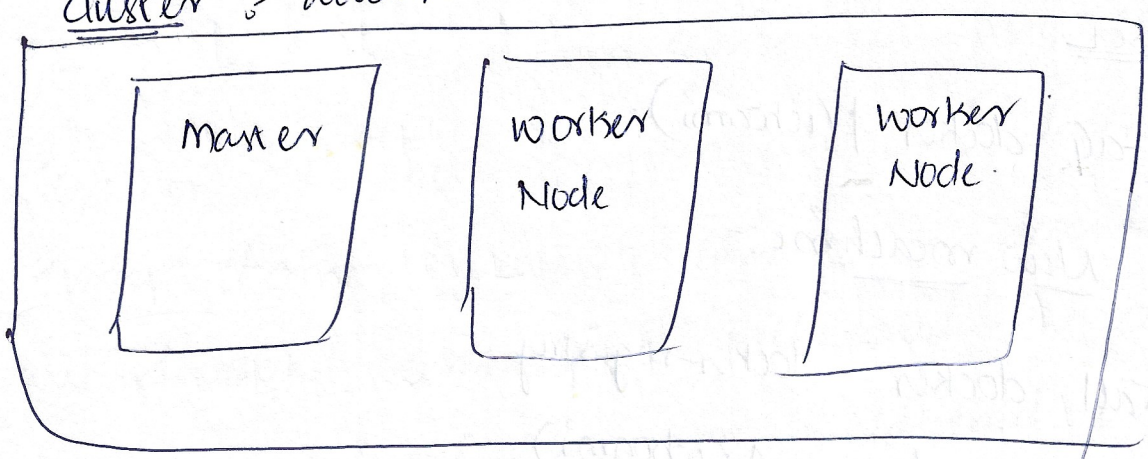
Yum install python-pip -y

sudo pip install docker-compose

sudo yum upgrade python*

Docker swarm :-

cluster :- atleast two machines



master :- Master manages the cluster.

worker :- TO manage the load

Launch instance :-

ubuntu 18.04 (select)

Servers (3)

Launch

Connect to the machines

Give names to the machines instead of ip address.

vi /etc/hosts

⇒ paste the (Ip address) manager
" " " worker01
" " " worker 02

Similarly do it on other machines also.

vi /etc/hosts

⇒ Add hosts names.

Now, open the ports.

Now, from manager
ping worker 01
ping worker 02
} Similarly check other machines too.

Now, set up the docker.

apt update -y (on all machines).

Now, docker install

Sudo apt install apt-transport-https software-properties-common
ca-certificates -y

(on all machines).

Add the Dockey key & Docker-ce repository

* curl -fsSL https://donwload.docker.com/linux/debian/gpg |
sudo apt-key add-

* sudo echo "deb[arch=amd64] https://donwload.docker.com/
linux/ubuntu xenial stable" > /etc/apt/sources.
list.d/docker-ce.list

sudo apt updati.

sudo apt install docker-ce -y

systemctl start docker.

systemctl enable docker

Now, create the user

useradd -m -s /bin/bash (username)

sudo usermod -aG docker (username).

su (username).

cd.

Now, creati the swarm cluster

docker swarm init --advertise-addr (master IP).

Run the join command in the worker01 and worker02 Nodes (from master node).

On master

* docker node ls.

To create micro serviceses in the clusters:-

* docker service creati --name my-web --publish 8080:80 nginx:1.13-alpine

docker service ls

where the service is running (on which node)

* docker service ps (service name).

ex:- docker service ps my-web.

[Here service is running on only master node].

Scale up the service

* docker service scale my-web=10.

[Here service is equally distributed and running on all the serviceses].

If one node completely gone:-

* docker node updati --availability drain ip-ipaddress.

[run this command on master].

docker node ls

* docker ps

to bring the drained server.

* docker node updati --availability active ip-Ipaddress.

01/04/22

Docker Swarm (compose) :-

Install docker.

* Yum install docker.
* Sudo curl -L https://github.com/docker/compose/release/download/1.21.0/docker-compose-'uname -s'-'uname -m' | Sudo tee /usr/local/bin/docker-compose > /dev/null.
* Sudo Chmod +x /usr/local/bin/docker-compose
* Sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
* docker-compose -v.

Whenever we want to work with docker-compose, we have to create docker-compose.yml file inside the file our service container is to be placed.

mkdir hello-world

cd hello-world/

→ inside this create

vi docker-compose.yml

→ Paste code from pdf.

→ To execute the docker-compose.yml

docker-compose up

mkdir wordpress

cd word press

ls

vi docker-compose.yml

→ Paste the code from pdf

Now, execute

docker-compose up

→ Page opened but in interactive way.

docker-compose up -d

→ Now, it will work in backed

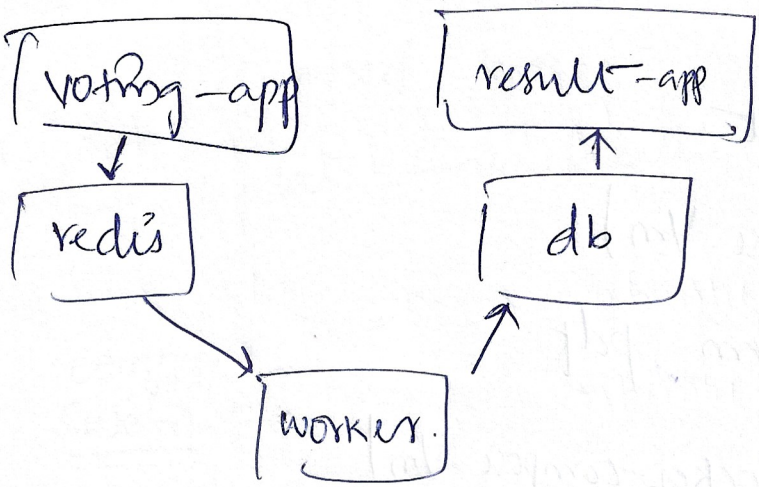
docker-compose stop

→ (To stop the service)

docker-compose start

→ To start the service.

voting App



install git

Yum install git

git clone (Repo from pdf).

cd example-voting-app/

Cat docker-compose.yml

cd vote

ls

Cat app.py

Cat dockerfile

cd ..

ls

cd worker/

ls

cd ..

ls

cd result/

ls

cd ..

is

cat docker-compose.yml

docker-compose up -d

error.

mv docker-compose.yml docker-compose1.yml

vi docker-compose.yml

=> (Paste the code from pdf)

docker-compose up -d

app not working as excepted as db is init so, clear the error for that part the db data from first repo and paste in the second db server we are using.

**

Docker Networking :-

Bridge :- which is by default docker network driver uses the Ip address to connect

Host :- uses port to connect

None :- no network connection

Overlay (Swarms) :- when created a cluster, this cluster are bound with Overlay.

macvlan :- based on macadress

docker run -dit Ubuntu

docker ps

docker rm -f (ip address)

docker run -dit --name ubuntu-ctr1 ubuntu.

" " " " ubuntu-ctr2 ubuntu.

* docker inspect bridge

⇒ whatever the containers running with bridge are displayed.

Now, Go (connect) to one server.

* docker exec -it ubuntu-ctr1 /bin/bash.

* ping 172.17.0.3

⇒ Not found. ping (install ping).

apt-get update

apt-get install iputils-ping.

ping (ip address).

Here, connecting with its names is not possible, but if we want to connect then we have to create 'custom networks' and use with containers.

creating custom Network :-

* docker network creati --driver=bridge custom-network

* docker network ls

* docker run -dit --name=ubuntu-ctr3 --network=custom-network ubuntu

* docker inspect custom Network

⇒ To see the custom networks

To (Switch / Connect) bridge driver to custom network then first disconnect the driver

* docker network / disconnect bridge ubuntu-ctr1

* docker network connect custom-network ubuntu-ctr1

⇒ this new network driver will get new ip address

* docker network rm (network name)

⇒ To delete the network