**DOCKER**

* Frist off all we need to create “dockerhub account”

**My dockerhub details:**

Id: adapaladocker

Email: [adapalapavan5@gmail.com](mailto:adapalapavan5@gmail.com)

* DockerHub is A Repository like GitHub. Here we can pack application into image with decencies and configurations. Any one can use this application with any OS system. These are called images in docker terminology.
* DockerHub is a Public resource.
* **Docker Installation:**

1. **Create EC2 instance.**
2. **Sudo su**
3. **sudo yum update -y**
4. **install docker engine “sudo amazon-linux-extras install docker” ===> for Amazon linux2**
5. **install docker engine “sudo yum install docker“ ===> for amazon linux**
6. **start “sudo service docker start“**
7. **(optional)** Add the ec2-user to the docker group ===> **sudo usermod -a -G docker ec2-user**
8. **docker info**

* After installation check any images are presented or not **“docker images”**
* **docker ps ===>** check any containers are running or not
* now I want to run images from DockerHub for this use below commands

1. **docker run <image name> ===>** it will pull and run the image

docker run httpd:2.4.48-alpine

1. **docker pull <image name: version>**

docker pull httpd: 2.4.48-alpine

* **docker commands:**

1. **docker ps ===>** to know about how many containers running
2. **docker images ===>** to know how many images available
3. **docker run <image name> ===>** it pulls and runs the images
4. **docker pull <image name: version> ===>** it pulls the images from DockerHub
5. **docker ps -a ===>** it gives the Container history (how many containers are running and stopped)
6. **docker rm <CONTAINER ID/container name> ===>** it removes the containers
7. **docker rmi < image name/IMAGE ID>** ===> it removes the images
8. **docker run -d -p <host port:docker port> <docker image>**
9. **docker run -d -p <host port:docker port> <docker image:tags>**
10. **docker run -dit -p <host port:docker port> <docker image>**
11. **docker run -it --rm -p <host port:docker port> <docker image:tags> ===>** if we stopped it will not store in “docker ps -a”
12. **docker inspect <container id / container name>** ===> it gives the container details
13. **docker inspect <container id / container name> | grep ipaddress** ===> container ip address
14. **docker ps -qa ===>** displays the all container ID’s
15. **docker rm $(docker ps -qa) ===>** removes all containers
16. **docker exec -it** **<container id / container name> /bin/bash** ===> login into container
17. **docker run -d -p <host port:docker port> -v <pwd(or)(path):/usr/share/nginx/html> --name <like “mynginx”> <imagename>** ===> this command is used when you want to host your html file in the docker container
18. **docker logs <container id / container name>**

**Create Docker Image**

1. install Prerequisites in EC2
2. Python3 “sudo yum install python3”
3. Pip3 “pip3 –version”
4. Install docker
5. Sudo su
6. cd app
7. touch requirements.txt
8. mkdir src
9. cd src
10. touch prog.py
11. vi prog.py

go to this link <https://pypi.org/project/Flask/>

copy the code

# save this as app.py

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route("/")

def hello():

return "Hello, World!"

1. cd ..
2. vi requirements.txt

enter flask==2.0.1

1. vi Dockerfile

# set base image

FROM python:3.7.9

# set working directory in the container

WORKDIR /code

# copy the dependencies file to working directory

COPY requirements.txt .

# install dependencies

RUN pip3 install -r requirements.txt

#copy the content of the local src directory to the working directory

COPY src/ .

# command to run on container start

CMD [ "python3", "prog.py" ]

1. docker build -t my-testapp .
2. docker run -dit -p 8000:5000 my-testapp
3. docker login
4. docker tag <image\_id> username/imagename:V1.1
5. check docker images
6. docker push username/imagename (or) docker push username/imagename:V1.1

(Above is the Flask, python webpage )

**How to create Docker image documentation provided by AWS:**

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/docker-basics.html>

**Management Commands:**

builder Manage builds

config Manage Docker configs

container Manage containers

context Manage contexts

image Manage images

manifest Manage Docker image manifests and manifest lists

network Manage networks

node Manage Swarm nodes

plugin Manage plugins

secret Manage Docker secrets

service Manage services

stack Manage Docker stacks

swarm Manage Swarm

system Manage Docker

trust Manage trust on Docker images

volume Manage volumes

**Commands:**

attach Attach local standard input, output, and error streams to a running container

build Build an image from a Dockerfile

commit Create a new image from a container's changes

cp Copy files/folders between a container and the local filesystem

create Create a new container

diff Inspect changes to files or directories on a container's filesystem

events Get real time events from the server

exec Run a command in a running container

export Export a container's filesystem as a tar archive

history Show the history of an image

images List images

import Import the contents from a tarball to create a filesystem image

info Display system-wide information

inspect Return low-level information on Docker objects

kill Kill one or more running containers

load Load an image from a tar archive or STDIN

login Log in to a Docker registry

logout Log out from a Docker registry

logs Fetch the logs of a container

pause Pause all processes within one or more containers

port List port mappings or a specific mapping for the container

ps List containers

pull Pull an image or a repository from a registry

push Push an image or a repository to a registry

rename Rename a container

restart Restart one or more containers

rm Remove one or more containers

rmi Remove one or more images

run Run a command in a new container

save Save one or more images to a tar archive (streamed to STDOUT by default)

search Search the Docker Hub for images

start Start one or more stopped containers

stats Display a live stream of container(s) resource usage statistics

stop Stop one or more running containers

tag Create a tag TARGET\_IMAGE that refers to SOURCE\_IMAGE

top Display the running processes of a container

unpause Unpause all processes within one or more containers

update Update configuration of one or more containers

version Show the Docker version information

wait Block until one or more containers stop, then print their exit codes