Installation of Docker and Docker-compose

Update the system

```
$ sudo apt update -y
```

Install docker and docker-compose packages

```
$ sudo apt install docker.io docker-compose -y
```

start docker service

```
$ sudo sytemctl start docker
```

enable the docker service for persistent across reboots

```
$ sudo systemctl enable docker
```

check the status of docker service

```
$ sudo systemctl status docker
```

if this user is sudoer then,

```
$ sudo usermod -aG docker $USER
```

check for docker version

```
$ docker --version
$ docker version
```

check for docker-compose version

```
$ docker-compose --version
```

To view docker commands

```
$ docker --help
```

To view docker-compose commands

```
$ docker-compose --help
```

To pull different linux images from docker public repository

```
$ docker pull ubuntu
$ docker pull debian
$ docker pull centos
$ docker pull fedora
$ docker pull alpine
```

To view the images in docker host

```
$ docker images
```

Dockerfile Commands

FROM - specifies the base image

RUN - runs a Linux command. Used to install packages into container, create folders, etc

ENV - sets environment variable. We can have multiple variables in a single dockerfile.

COPY - copies files and directories to the container.

EXPOSE - expose ports

CMD - provides a command and arguments for an executing container. There can be only one CMD

creating ubuntu apache2 webserver container from docker file

creating apache webserver image using docker file

```
$ mkdir apache && cd apache
$ vim Dockerfile
FROM ubuntu
ENV DEBIAN_FRONTEND=noninteractive
RUN apt-get update -y
RUN apt-get install apache2 -y
RUN apt-get install apache2-utils -y
RUN apt-get clean
COPY index.html /var/www/html/
EXPOSE 80
CMD ["apache2ctl","-D","FOREGROUND"]
:x

$ vim index.html
<h1> Hi this is Apache Docker Container</h1>
:x

$ docker build -t myapache .
$ docker run -d --name apache_demo -p 8085:80 myapache

$ docker ps
access in browser
http://ip:8085
or
localhost:8085
```

```
$ docker container ls
$ docker images
```

ubuntu apache2 and nginx webserver containers out of images from docker hub

creating ubuntu apache webserver container out of image ubuntu/apache2 from docker hub

```
$ docker pull ubuntu/apache2
$ docker images
$ docker run -d --name apache_demo2 -p 8086:80 ubuntu/apache2
$ docker ps
```

access in browser

http://ip:8086

or

localhost:8086

creating ubuntu nginx webserver container out of image ubuntu/nginx from docker hub

```
$ docker pull ubuntu/nginx
$ docker images
$ docker run -d --name nginx_demo -p 8087:80 ubuntu/nginx
$ docker ps
```

access in browser

http://ip:8087

or

localhost:8087

Flask app

```
$ mkdir flask_app && cd flask_app

$ vim requirements.txt
flask
redis
:x

$ vim app.py
# compose_flask/app.py
from flask import Flask
from redis import Redis

app = Flask(__name__)
redis = Redis(host='redis', port=6379)

@app.route('/')
def hello():
    redis.incr('hits')
    return 'This Compose/Flask demo has been viewed %s
time(s).' % redis.get('hits')


if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True)
:x
```

```
$ vim Dockerfile
FROM python:3.9
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
CMD python app.py
:x

$ vim docker-compose.yml
version: '2'
services:
    web:
        build: .
        ports:
            - "5000:5000"
        volumes:
            - .:/code
        depends_on:
            - redis
    redis:
        image: redis
:x

To start the flask application container
$ docker-compose up -d

To stop the flask application container
$ docker-compose down
```

access in browser

http://ip:5000

or

localhost:5000

wordpress

```
$ mkdir wordpress && cd wordpress
```

```
$ vim docker-compose.yml
version: '3.3'
services:
   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     volumes:
       - wordpress_files:/var/www/html
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: my_wordpress_db_password

   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: my_db_root_password
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: my_wordpress_db_password
volumes:
    wordpress_files:
```

```
    db_data:
:x
```

To validate the docker-compose.yml

```
$ docker-compose -f docker-compose.yml config
```

To start the wordpress container

```
$ docker-compose up -d
```

To stop the wordpress container

```
$ docker-compose down
```

To view the docker process status

```
$ docker ps -a  (all)
$ docker ps -s  (size)
```

To login to a docker container

```
$ docker exec -it container_name /bin/bash
```

https://hub.docker.com/

To login to dockerhub registry

```
$ docker login
username:
password:
```

To logout from dockerhub registry

```
$ docker logout
```

How to push image from local to dockerhub registry

```
$ docker tag portainer/portainer:latest
dnadna/myrepo:myfirstimagepush
```

then

```
$ docker push dnadna/myrepo:myfirstimagepush
```

How to pull image from dockerhub registry to local

```
$ docker dnadna/myrepo:flask_app
```

To stop a running container

```
$ docker stop <container_id>
```

To start a container

```
$ docker start <container_id>
```

To restart a container

```
$ docker restart <container_id>
```

To kill the container by stopping its execution immediately

```
$ docker kill <container_id>
```

To commit changes to docker image

```
$ docker commit [CONTAINER_ID or Name] [new_image_name]
```

or

```
$ docker container commit [CONTAINER_ID or Name] [new_image_name]
```

To delete a stopped container

```
$ docker rm <container_id>
```

To directly remove the container without stopping it

```
$ docker rm -f <container_name or container_id>
```

To delete an image from local storage

```
$ docker rmi <image_id>
```

To force delete a Docker Image

```
$ docker rmi -f <image_id or image_name>
```

To list the details of all the network

```
$ docker network ls
```

To get information about docker installed

```
$ docker info
```

To Copy file from a docker container to the local system

```
$ docker cp <containerId>:/tmp/sample.txt
/home/dhana/Documents/
```

To copy file from local system to docker container

```
$ docker cp sample.txt container_id:/tmp
```

To show the history of a docker image

```
$ docker history <image_id>
```

To show the logs of the docker container

```
$ docker logs <container_id>
```

To search for a docker image on dockerhub

```
$ docker search <image_name>
```

To create a volume

```
$ docker volume create
```

or

```
$ docker volume create --name volume-name
```

To list the volumes

```
$ docker volume ls
```

To get Details about a Docker Volume

```
$ docker volume inspect <volume_name>
```

To delete a volume

first stop the container

```
$ docker volume rm <volume_name>
```

To change repository name or rename image

```
$ docker tag repository:tag new_image_name:tag
```

or

```
$ docker tag image_id new_image_name:tag
```

To list networks

```
$ docker network ls
```

To list all the Running Containers with the File Size

```
$ docker container ls -s
```

To List the IDs of the Running Containers

```
$ docker ps -q
```

or

```
$ docker container ls -q
```

List the IDs of all the Containers (irrespective of the state)

```
$ docker ps -a -q
```

or

```
$ docker ps -aq
```

To Pause a running Container

```
$ docker pause <container_id or container_name>
```

or

```
$ docker container pause <container_id or container_name>
```

To Unpause a paused Container

```
$ docker unpause <container_id or container_name>
```

or

```
$ docker container unpause <container_id or
container_name>
```

Listing Processes running in a Docker Container

```
$ docker top <container_name or container_id>
```

or

```
$ docker container top <container_name or container_id>
```

Rename a Docker Container

```
$ docker rename <old_name> <new_name>
```

or

```
$ docker container rename <old_name> <new_name>
```

pass any command that we want to execute inside the container

```
$ docker exec -it <container_name> echo "Hello, from
container"
```

To list the Docker Image Ids

```
$ docker images -q
```

To list all the Docker Images (including dangling images)

```
$ docker images -a
```

To list Dangling Docker Images

```
$ docker images -f dangling=true
```

To get the logs of the Docker container

```
$ docker container logs <container_id or container_name>
```

To display the last few lines of the container logs

```
$ docker container logs -f <container_id or
container_name>
```

To Get the last 3 lines of the Container Logs

```
$ docker container logs --tail 3 <container_id or
container_name>
```

To get Docker Stats of the running Container

```
$ docker stats
```

To get Docker stats of all containers

```
$ docker stats --all
```

To get Detailed Info about an Object (Container, Image, Volume, etc)

```
$ docker inspect <name or id>
```

To get the Summary of Docker Usage

```
$ docker system df
```

To Show all mapped ports

```
$ docker port container_name
```

Exporting a container

```
$ docker export container_name > container_name.tar
```

or

```
$ docker export container_name | gzip > container_name.gz
```

Create a backup

```
$ docker save image_name > image_name.tar
```

To delete all the Stopped Containers

```
$ docker container prune
```

To remove all the Dangling Docker Images

```
$ docker image prune
```

To remove all the Dangling and Unused Docker Images

```
$ docker image prune -a
```

To Clean your Docker system

```
$ docker system prune
```

ctop tool to monitor docker health status

```
$ sudo wget
https://github.com/bcicen/ctop/releases/download/v0.7.1/c
top-0.7.1-linux-amd64  -O /usr/local/bin/ctop

$ sudo chmod +x /usr/local/bin/ctop

$ ctop
```

To display active containers

```
$ ctop -a
```

To display CPU as % of system total

```
$ ctop -scale-cpu
```

Portainer GUI Management Tool

```
$ docker volume create portainer_data

$ docker run -d -p 8000:8000 -p 9000:9000 --name
portainer --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v
pt_data:/data portainer/portainer-ce:latest

$ docker ps
```

access the GUI management tool

http://ip:9000

or

http://localhost:9000

default user is admin

create password