Assignment week 4

Task 1: Introduction to containerization and Docker fundamentals, Basic Commands

1. **Need of Docker**
   We face issues when we are replicating 1 App from machine A with different versions, into Machine B with different versions.
   - Dependency Installation
   - Specific part of App depends on specific version of SW
   - Commands of Linux doesn't work on Mac or Windows

   These issues are also present in Production.

2. **What is Docker?**
   It is a tool which helps us to run our containers.

3. **What is a Container?**
   It is a single unit which combines our app + dependencies.
   Process of replication becomes easy from machine A to machine B.
   Irrespective of Machine OS, container run there.
   Containers help in replicating entire local environment in a standard way across a large team.

   Properties-
   -Portable
   -Light Weight
   -Different containers can have different environment

4. **Docker Images**
   It is an executable file, which contains instructions to build a container. (As same as the Class-Objects concept).
   Image- Static screenshot of local env
   Container- Actually running instance

5. **Example command to Create Ubuntu Env in Windows using Docker Container**
   docker run -it ubuntu
   -it= interactive mode (We can access ubuntu terminal)

6. **Some Docker commands**
    a. docker pull img-name
       Ex. docker pull hello-world
    b. docker images (show all images)
    c. docker run -it img-name or docker run img-name (creates and executes img container)
       Ex. docker run -it ubuntu
    d. docker ps -a (Tells how many containers exist)
    e. docker ps (Tells which containers are running)
    f. docker start cont_name or docker start cont_id (Starts existing containers).
    g. docker stop cont_name or docker stop cont_id (Stop running container)
    h. docker rmi img-name (remove image)
    i. docker rm cont-name (remove cont)
    j. docker pull img-name:version
       Ex. docker pull mysql:8.0
    k. docker run -d img-name
       -d= detach mode (Runs in Background)
    l. docker run –name my-name cont-name -d img-name (Renaming a cont

7. Docker Image Layers

```
   Cont
     |
 Layer 2
     |
 Layer 1
     |
Base Layer
```

8. Port Binding

Process of binding the host machine port to the container port is called port binding.
-phost-port:cnt-port
Ex. docker run -d -e MYSQL_ROOT_PASSWORD=secret –name mysql-latest
-p8080:3306 mysql

9. Troubleshoot Commands

- docker logs cont_id (can access logs)
- docker exec -it cont_id/bin/bash or sh (Allows us to run additional commands on running cont) ex. Env exit

10. Difference between docker and virtual machine

- Each machine has 3 Layers

```
                Application Layer   (Virtualized by Docker)
                        |
                Host of Kernel      (Virtualized by VM)
                        |
                   Hardware
```

Advantage- Lightweight

Disadvantages-

1. Docker only virtualize Application Layer (Docker was initially built forlinux based OS, Docker wants linux kernel, To use docker on Mac or Windows we use docker desktop)
   So Docker desktop contains a small linux based VM which helps us to run our containers.

11. Networks in Docker

In general when we want 2 different containers to contact each other we need to access their ports.
But if we want that, the above process should happen without any port binding. We use concepts of network.
So docker networks are isolated part where 2 containers can interact with each other without any port.
docker network ls (Display all networks)
docker network create network-name (Creates a new network)

12. Developing with Docker

We will pull mongo and mongo-express images container.

- Mongo Image
  docker run -d -p27017:27017 –name mongo –network my-network -e MONGO_INITDB_ROOT_USERNAME=admin -e MONGO_INITDB_ROOT_PASSWORD=admin mongo

- docker ps
- docker run -d -p8081:8081 –name mongo-express –network my-network -e ME_CONFIG_MONGODB_ADMINUSERNAME=admin -e ME_CONFIG_MONGODB_ADMINPASSWORD=admin -e ME_CONFIG_MONGODB_URL="mongodb://admin:admin@mongo:27017" mongo-express

13. Docker Compose

- It is a simpler way (method) for defining and running multi container applications
- We use .yaml(yet another markup language) for this
- Two Important compose commands are-

docker compose -f fileName.yaml up -d (start the service)
docker compose -f fileName.yaml down (Stop service and delete containers)

14. Dockerizing our App

- It is converting our application to docker-image and then an executable container.
- Generally Jenkins is also used for this
- Here we will use DockerFile which contains all instructions for how to dockerize our app.
- docker build -t appname:1.0 .
- docker run -it appname:1.0 bash

15. Common DockerFile Instructions

- `FROM <image>` - this specifies the base image that the build will extend.
- `WORKDIR <path>` - this instruction specifies the "working directory" or the path in the image where files will be copied and commands will be executed.
- `COPY <host-path> <image-path>` - this instruction tells the builder to copy files from the host and put them into the container image.
- `RUN <command>` - this instruction tells the builder to run the specified command.
- `ENV <name> <value>` - this instruction sets an environment variable that a running container will use.
- `EXPOSE <port-number>` - this instruction sets configuration on the image that indicates a port the image would like to expose.
- `USER <user-or-uid>` - this instruction sets the default user for all subsequent instructions.
- `CMD ["<command>", "<arg1>"]` - this instruction sets the default command a container using this image will run.

16. Publishing our image on Docker

- Create account on hub.docker.com
- Create a new repo
- Build the app with same name as repo name
- Login in your terminal with your credentials
  Commands
  docker login -u username
- docker push image-name