Docker

general commands

- get docker info

```
> docker info
```

commands for managing images

- get the list of images

```
> docker image ls
```

- pull an image from docker hub

```
> docker image pull <image name>

# pull hello world from docker hub
> docker image pull hello-world
```

- get information about an image

```
> docker image inspect <image name>
```

- remove an image from computer

```
> docker image rm <image name>
```

- create a new image from dockerfile

```
# -t: tag (image name)
# . here means the Dockerfile exists in the current directory
> docker image build -t myimage .
```

- delete empty/unreferenced images

```
> docker image prune
```

container commands

- get the list of running containers

```
> docker container ls
```

- get the list of all containers (running and stopped)

```
> docker container ls -a
```

- run a new container

```
> docker container run  <image name>

# run a container from hello-world image
> docker container run hello-world

# starts the container in detached mode
> docker container run -d httpd

# --name sets the name to the container
> docker container run --name mycontainer httpd

# run the container in interactive mode
# you can now interact with the container
> docker container run -i httpd

# get the container attached to the terminal to
# send the commands which will run inside the container
> docker container run -t httpd

# forward the port
# 8080: source port (machine)
# 80  : destination port (container)
> docker container run -p 8080:80 <container id/name>
```

- stop a running container

```
> docker container stop <container id/name>
```

- start a new container

```
> docker container start <container id/name>
```

- remove a container

```
# remove a stopped container
> docker container rm <container id/name>

# remove a running container
> docker container rm --force <container id/name>
```

- get the details of a container

```
> docker container inspect <container id/name>
```

- execute any command inside a container

```
> docker container exec <container id/name> <command>

# run date command inside a container (with name myubuntu)
> docker container exec myubuntu date

# get the terminal of running container
# container start a new process for bash
# when this new process exits the container does not exit
> docker container exec -it myubuntu bash
```

- attach to the container's main process

```
# attaches to the main process of the container
# if the main process exits, the container also gets exited
> docker container attach <container id/name>
```

- check the logs from container

```
# get recent logs
> docker container logs <container id/name>

# get the logs as generated (continuous)
> docker container logs -f <container id/name>
```

exercises

- execute date command inside a httpd container

```
# create a httpd container
> docker container run -itd --name myhttpd httpd

# execute the date command
> docker container exec myhttpd date

# remove the container
> docker container rm --force myhttpd
```

- start a website container from a custom image

```
> docker container rm --force mywebsite

> docker image build -t myimage .

> docker container run -itd -p 8080:80 --name mywebsite myimage
```

- start a mysql container

```
> docker container run -itd -p 3306:3306 --name mydb -e
MYSQL_ROOT_PASSWORD=root mysql
```

- start a backend server (running express) in a container

```
# create your image from node
FROM node

# copy all the contents within this directory to the container
COPY . .

# expose the the port 4000
EXPOSE 4000

# start the server when container gets created
CMD node server.js
```

```
> docker container rm --force myserver

> docker image build -t myserver .

> docker container run -itd -p 4000:4000 --name myserver myserver
```

docker swarm

- create a new swarm

```
> docker swarm init --advertise-addr <ip address of manager>
```

- add a new node in swarm

```
> docker swarm join --token <token> <ip address:port>
```

- get the token to add new manager/worker

```
# this token will be used to add a manager to the swarm
> docker swarm join-token manager

# this token will be used to add a worker to the swarm
> docker swarm join-token worker
```

- to remove a node from swarm

```
# execute this command from the machine that you want to remove
> docker swarm leave
```

- to take the swarm down

```
# execute this command from manager node
> docker swarm leave --force
```

docker node commands

- get the list of nodes

```
> docker node ls
```

- remove a node

```
> docker node rm <node name>
```

- get all the details of a selected node

```
> docker node inspect <node name>
```

docker service commands

- create a new service

```
> docker service create <image name>

# create a service instance with name
> docker service create --name <service name> <image name>

# create no of replicas while creating the service
> docker service create --replicas <no> <image name>

# expose the port number for outside world to connect
> docker service create -p <source port>:<destination port> <no>
<image name>
```

- get the list of services

```
> docker service ls
```

- remove a service

```
> docker service rm <service name>
```

- get service details

```
> docker service inspect <service name>
```

- get list of containers / replicas created by service

```
> docker service ps <service name>
```

- scale (up or down) the service

```
> docker service scale <service name> = <replicas>
```