Useful docker commands

```
there are many docker images available on hub.docker.com, we can just
pull them & start running containers

  docker search <image name> -- to search the docker images from
command line
  docker pull <image name> -- to pull a docker image from docker hub

  docker images -- to display the images on your local machine
  docker history < image name > -- to know the changes done to the image
  docker inspect < image name > -- to view the detaied information in
JSON output
  docker rmi <image name> -- to remove a image from local machine
  docker image prune -- to remove all unused images from local machine

how to run containers from a docker image

  two ways we can run the images
    interative mode ( -it )
    detached mode  ( -d )

  docker run -d nginx ( always creates new container & detaches from
the terminal you are working on ( goes in background process ))
  docker run -it nginx bash ( always creates new container & gets
inside the container )
  docker ps -- to check all running containers
  docker ps -a -- to check all running + exited/stopped containers

  docker stop <container id> -- to stop a container
  docker start <container id> -- to start a stopped container
  docker restart <container id> -- to restart a container
  docker inspect <container id> -- to view detailed information about
the continaer in JSON format
  docker rm <container id> -- to remove a stopped/exited container
  docker rm -f <container id> -- to remove a container forcefully
though it is in running state
  docker container prune -- to remove all stopped/exited containers

how to get inside a running container

  docker run -d nginx ( creates a new container in detached mode )
  docker ps  -- displays the running containers, note down the
container id
  docker exec -it <container id> bash

how to come out of a container
  ctrl pq ( it is mandatory to use this command always )
```

```
how to access the applications running inside the container from
external world ( browser )

  services running inside a container can never be accessed direcltry,
we always need to publish/expose them on to docker host while we create
the containers

  we need to publish/expose the services using -P (capital) OR -p
(small) with in the docker run command

  ex: -P ( capital ) -- docker will publish/expose the port number
dynamically on docker host & maps with port running inside the cont

    docker run -d -P nginx -- which create a new port mapping from
docker host to the service inside the container
    docker ps -- to check the container port
    to access : in the browser http://<docker host IP>:<exposed port>
ex: http://52.14.62.88:32768

    -p ( small ) -- we need to assign a port on docker host & map to
the port inside the container

    docker run -d -p 1234:80 nginx ( always port-on-docker-host :
process-port-inside-container )
    docker ps -- to check the container port
    to access : in the browser http://<docker host IP>:<exposed port>
ex: http://52.14.62.88:1234

=========================================================================
=======================================================

Advanced Container commands

limit container resources
  docker run -d --restart unless-stopped -p 8080:80 --memory 500M --
memory-reservation 256M nginx
  docker run -d -P --cpus=".5" nginx

Run a docker container, overriding the system default logging driver
settings:
  docker run --log-driver json-file --log-opt max-size=50m nginx

updating container network
  docker network disconnect bridge <contid>
  docker network connect myb <contid>

  docker run -itd --net net1 alpine
  docker network connect net2 contid
  ( container will be present in both net1 & net2 networks )
```

```
--restart flag: specify when the container should be automatically
restarted
  1) no(default): Never restart the container
  2) on-failure: only if the container fails (exits with non-zero exit
code)
  3) always: Always restart the container whether it succeeds or fails.
Also start the container automatically on  daemon startup
  4) unless-stopped: Always restart the container whether it succeeds
or fails, and the daemon startup, unless the container was manually
stopeed
```