**docker version** : **verify cli can talk to engine**. **docker info** : **most config values of engine** . **docker run**

\*Image is application we want to run \*A container is an instance of that image running as a process \*i can have many containers of same image.For example: Image is nginx web server. **80:80 = host port: port in container**

**docker run -p 80:80 -d --name webhost nginx** //named the server

**docker stop [container id]** //stop docker

as now logs not visible to check it's running write:**docker container top webhost**

to check all status: **docker ps -a**

If we want to **remove container** we use : **docker rm [1st 3 number of container id,multiple can be place with space]**

if we want to **remove running container** too : **docker rm -f [1st 3 number of container id,multiple can be place with space]**

**\*\*\*next\*\*\***

**docker run --name mongo -d mongo** //-d is dispatch,last mongo is image

**docker stop mongo**

**docker start mongo**

**ps aux | grep mongo** to find all process of mongo.

**docker top mongo**// show processes running in a single container

**\*\*\*\*run 3 diff container with 3 diff images \*\*\*\***

**docker run -d --name db -p 3036:3036 -e MYSQL\_RANDOM\_ROOT\_PASSWORD=yes mysql**

**docker logs db** //to find generated password.

**docker run -d --name webserver -p 8080:80 httpd**

**docker ps** //to see all active containers

**docker run -d --name proxy -p 80:80 nginx**

//now check if they are working correctly in each port.

**curl localhost**

**curl localhost: 8080**

**docker stop [press tab to get suggestions of container ids]**

**docker ps -a** //check all run or stopped containers.

**docker rm [press tab to get suggestions of container ids]**

**docker image ls** //to see all images.doesn't matter if containers removed.

**\*\*\*\*\*\*\*What's going on in containers\*\*\*\*\*\*\*\*\*\*\***

**docker run -d --name proxy nginx**

**docker run -d --name db -e MYSQL\_RANDOM\_ROOT\_PASSWORD=yes mysql**

**docker ps** //lists containers

**docker top mysql** //processes of that container

**docker inspect mysql** //json of all data of how the container started.

**docker stats** //to monitor mem usage and limit.ctrl+c to come out of it.

**\*\*\*\*\*\*\*\*Getting a shell inside containers\*\*\*\*\*\*\*\*\*\***

**docker run -it --name proxy nginx bash** //it gives terminal inside the running container,bash is one of the available terminal in container now inside container.

**exit**: tocome out. the container stops. As we change it from normal one to bash , so when we exit bash the container stop.

**docker container run -it --name ubuntu ubuntu** //it's default cmd is bash

**apt-get install -y curl**

**curl google.com**

**exit** //ubuntu container stop

**docker container ls -a** //to verify it existed

**docker container start -ai ubuntu** //to start the stopped container that was exited again and the package curl will be present.

**exit**

**\*\*\*\*\*\*\*\*\*Running additional process in a running container\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**docker container exec -it mysql bash**// now i am in the container with mysql

**ps aux** //it will show the processes in the container [ps is no longer included in mysql img by default,use apt-get update && apt-get install -y procps]

**exit**

**docker container ls** //it will show mysql still tunning even though bash it stopped

**\*\*\*\*\*\*\*Docker Networks create,connect disconnect\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**docker container run -p 80:80 -d --name webhost nginx**

**docker container port webhost**

**docker container inspect --format "{{.NetworkSettings.IPAddress}}" webhost** //docker ip will appear which is not same as my laptop's ip using "ifconfig en0"

''

-p means exposing port to the physical network .so we can communicate using that port to container

But 2 container under same virtual net can communicate without port

docker network create --driver to create virtual net.

Now go to bash again from where we left before.

''

**docker network ls**

**docker network inspect bridge**

**docker network ls** // here 3 name will appear

//host skip virtual network but sacrifices security of container model

//none remove eth0 and leave with localhost

**docker network create my\_app\_net**

**docker network ls**

**docker container run -p 80:80 -d --name new\_nginx --network my\_app\_net nginx**

**docker inspect my\_app\_net** //here we will see that new container

**docker network connect [use tab to access id of my\_app\_net] [place the id of container i want to add in that network]**

**docker network disconnect [use tab to access id of my\_app\_net] [place the id of container i want to add in that network]** //to disconnect containers

**\*\*\*\*\*\*\*\*\*DNS\*\*\*\*\*\*\*\*\*\*\***

now in my app net new nginx is present.If we create a new container and run in that network

**docker run -d --name my\_nginx --network my\_app\_net nginx**

**docker container inspect [id of my\_app\_net]** // to check the containers in it

**docker container exec -it my\_nginx ping new\_nginx**

//"docker container exec -it" can be used to jump in and run command without creating additional process

//it will check they can communicate

**\*\*\*\*\*\*\*\*\*\*Excercise:Using containers for network cli app testing\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*learn about --rm at first\*\*\*\*\*\*\*\*\*\*\*\*\***

**docker container run --rm -it centos:7 bash**

**yum update curl**

**curl version**

ctrl+shift+d to open a new window inside

**docker container run --rm -it ubuntu:14.04 bash**

**apt-get update && apt-get install -y curl**

**curl --version**

now in both window run exit. then check using **docker container ls -a**