**LAB- running basic commands**

**STEP 1 – Running basic docker commands**

To pull a docker image use **docker pull ubuntu**

To run the container, run **docker run ubuntu**

You should observe ubuntu container start and also stops immediately

Now run **docker ps** to see all the running containers. You should not see any running containers. To see all the containers Running and Not Running, execute **docker ps –a**

Now we want to run a nginx container. Execute **docker run nginx.**  Nginx container will start in foreground. You should see all the logs. If you want to get command prompt, press CTRL+C.

Is the container stopped once u press CTRL+C ?

Run docker ps and observe that this nginx container is still up.

Now stop the nginx container using **docker stop <<containerid>>** . To get container id, run docker ps and observe.

Run **docker ps** and see that nginx container is not listed. If u want to see all conainers (running and not running) use **docker ps –a**

If u want to run the stopped container again, **docker start <<containerid>>**

If u want to remove a container completely, use **docker rm <<containerid>>**

use docker rm –f <<containerid>> to forcefully remove container even if it is running

if we run ubuntu container using **docker run ubuntu,** it will terminate immediately.

After starting container, if we want to run a command like sleep 20, use **“docker run ubuntu sleep 20”**

execute **docker run –it ubuntu bash** to get the bash shell prompt in the container.

Once u get the bash shell of the container, u can execute and commands inside the container.

I want to see the ubuntu release version in the container. So, execute cat /etc/\*release\*

You should observe the ubuntu version in the output

To come out of bash shell, type exit

If you want to give a name to the container, use a command like below:

**docker run --name mynginx –d nginx**

Can you tell what is the meaning of –d option in the above command?

To remove an image, we can use docker rmi <<imagename>

Now try to remove ubuntu image forcefully. (Hint : use –f option to remove forcefully)

If we want to run a command on existing container, use

docker exec <<containerid>> command

Try the following **docker exec –it <<containerid>> bash**

**STEP 3 – Create your first Image**

We want a custom nginx image which contains tools like ping, ip so that we can see the ip of container and ping one container from another.

Open **nginx-dockerfile-v1** inside **01-ubuntu-with-ping-ip** folder and observe that it contains following lines:

**from nginx**

**RUN apt-get update && apt-get install iputils-ping -y && apt-get install iproute2 -y**

**COPY ./staticfiles-v1/hello.html /usr/share/nginx/html/hello.html**

Now cd to **01-ubuntu-with-ping-ip** folder

Run the following command to build image

**docker build -t mynginx . -f nginx-dockerfile-v1**

Now you should be able to see your images listed if u execute **docker images**

Now we want to run a container of this image and find its ip.

Execute following commads to start 2 containers of mynginx:

**docker run –d --name nginx1 mynginx**

**docker run –d --name nginx2 mynginx**

Now attach to bash shell of nginx1 using

**docker exec –it nginx1 bash**

Once u are in bash shell execute **ip addr show** and see the ip.

Do the same with nginx2 and get its ip.

Now from bash terminal of nginx2, try to ping ip of nginx1 and observe that ping should be successful because all the docker containers created in this host will be part of a bridge Network

Now delete nginx2 container using docker rm –f nginx2

Create another nginx2 container now with link option as shown below:

**docker run –d --name nginx2 –link nginx1:nginx1 mynginx**

Now execute the following to get the bash terminal of nginx2

**docker exec –it nginx2 bash**

Now execute cat /etc/hosts at bash terminal and observe that ip of nginx1 is mapped with name nginx1

So, you will be able to ping nginx1 from nginx2 using name with below command:

**ping nginx1**

**STEP 3 – Using Port Mapping, inspecting container and getting logs**

As we are using Docker toolbox, you containers are running inside VirtualBox in a Linux VM.

How to get ip of the machine?

Execute following command:

**docker-machine ip**

Nginx will be listening for requests at port 80 inside the container. We need to configure port forwarding to forward requests from host to containers

Use the below command **docker run –p 8080:80 -d --name mynginx nginx**

Now you will be able to see index page of nginx by hitting <http://192.168.99.100:8080>

Here 192.168.99.100 is assumed to be ip of linux vm in which our containers are running

Now i want to see all the details of my container. So, use the following command:

**docker inspect <<containerid/name>>**

Now start a mysql container.

See the documentation at <https://hub.docker.com/_/mysql>

Accoring to documentation, this mysql image is configured to store data at **/var/lib/mysql** inside the container.

If we want to mount a directory on host machine to this /var/lib/mysql inside container, use the –v option as shown below:

**docker run --name srini-mysql -e MYSQL\_ROOT\_PASSWORD=root –d –p 3306:3306 –v mysqlvolume:/var/lib/mysql mysql**

**Observe how we are passing environment variables to containers**

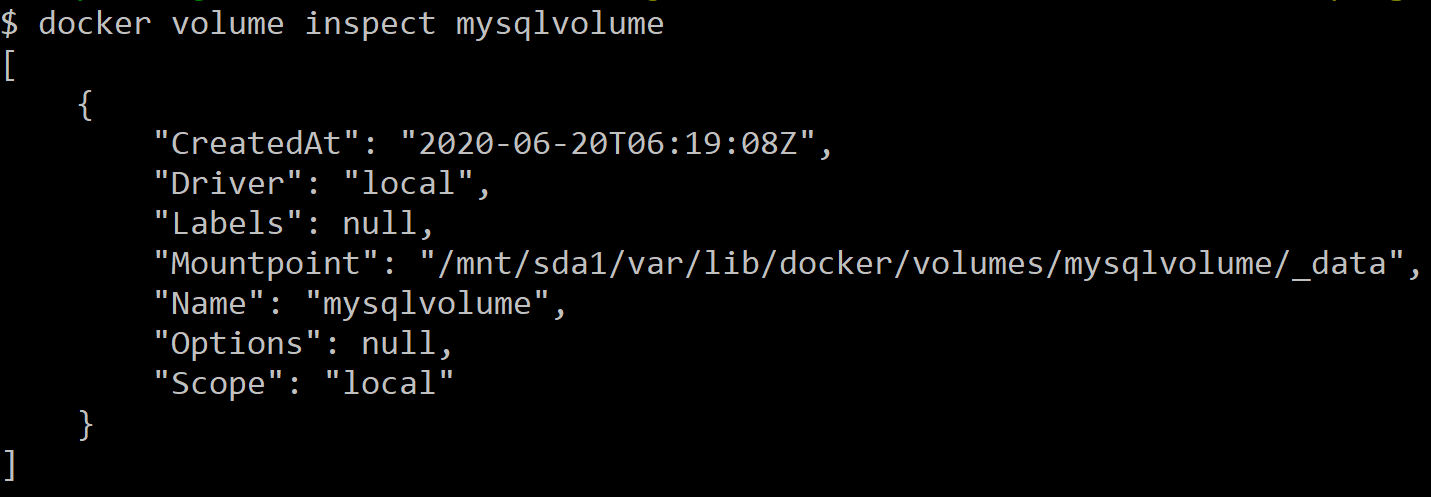
Above command will automatically create a volume with name mysqlvolume on the host machine and mount it to /var/lib/mysql of the container

Now, execute following command to list all the volumes

**docker volume ls**

To inspect a volume, use the below command:

**docker volume inspect mysqlvolume**



We can ssh in to the linux machine where our containers are running using below command:

**docker-machine ssh**

Once you do ssh into linux vm,

execute **sudo ls /var/lib/docker/volumes** and view that there is a folder with name mysqlvolume

To see logs of a container use

docker logs <<containerid/name>>