**Docker**

Install Docker desktop, enable Hyper-V in windows 10 and install Linux kernel as per the docker pop window and Restart the computer.

Once sign in back to computer, go to command line and type

***docker –version***

Note: Hyper-V is a Windows 10 virtualization technology and is used to let us create multiple virtual machines and run different OSes in one single Windows 10 OS.

**Docker Registry:**

When we run the below command, we are downloading docker image from some remote location called *docker registry (*[*https://hub.docker.com*](https://hub.docker.com)*)*

***docker run in28min/todo-rest-api-h2:1.0.0.RELEASE***

Docker Registry contains lot of repositories and lot applications under each repository with different versions.

*What does the image contains?*

Images contains all the dependencies and software that are needed for our application to run. Like for example Java, all the dependent libraries.

An image is static and is just a set of bytes when it is present in the registry and even when it is downloaded to a local computer it is just an image but when it is running then it is called container.

When we run the below command and try to access the application in browser we will have issues.

***docker run in28min/todo-rest-api-h2:1.0.0.RELEASE***

So we are adding ***-p {Host Port}:{Container Port}***

***docker run -p 5000:5000 in28min/todo-rest-api-h2:1.0.0.RELEASE***

This is because the container that we run is part of docker bridge network (kind of docker’s internal network) and is not accessible directly until you expose this to the system where the container is running. Hence we adding above attribute to map internal port to host port.

Another handy attribute is detach, -d

When we put -d in the docker run command, we are actually detaching it from the terminal. So it will be running in the background. Once we add this and run, then we will get a container id.

***docker run -d -p 5000:5000 in28min/todo-rest-api-h2:1.0.0.RELEASE***

In order to see the logs, we use

***docker logs containerID***

here the containerId for logs can be part of full id. It doesn’t have to be full id. We can also add -f to follow the tail of logs.

***docker logs -f containerID***

In order to see what all the containers that are running, we can use below command

***docker container ls***

If we want to see images that are downloaded to our local computer

***docker images***

If we want to see all the containers that either running or stopped

***docker container ls -a***

In order to stop a container

***docker container stop containerID***

**Docker Architecture**



Docker Client

Docker demon

Containers

Local images

Image Registry

Mysql

Eurek

Your-app



When we install docker in our computer we will get both docker client and docker demon.



Docker client is where we type all the commands and these commands are sent to docker demon, and docker demon is the one that runs the command. This is also a client server architecture. Docker demon manages all the locally run containers, local images and pulling images from image registry or pushing locally build image to image registry.

**Why is Docker popular?**

**Playing with Docker**

We already know how to see the list of images in our local computer using docker command

***docker images***

Now if we want to change the tag/version of the image we can do that using

***docker tag in28min/todo-rest-api-h2:1.0.0.RELEASE in28min/todo-rest-api-h2:1.0.0.latest***

In order to pull any images then use the below command

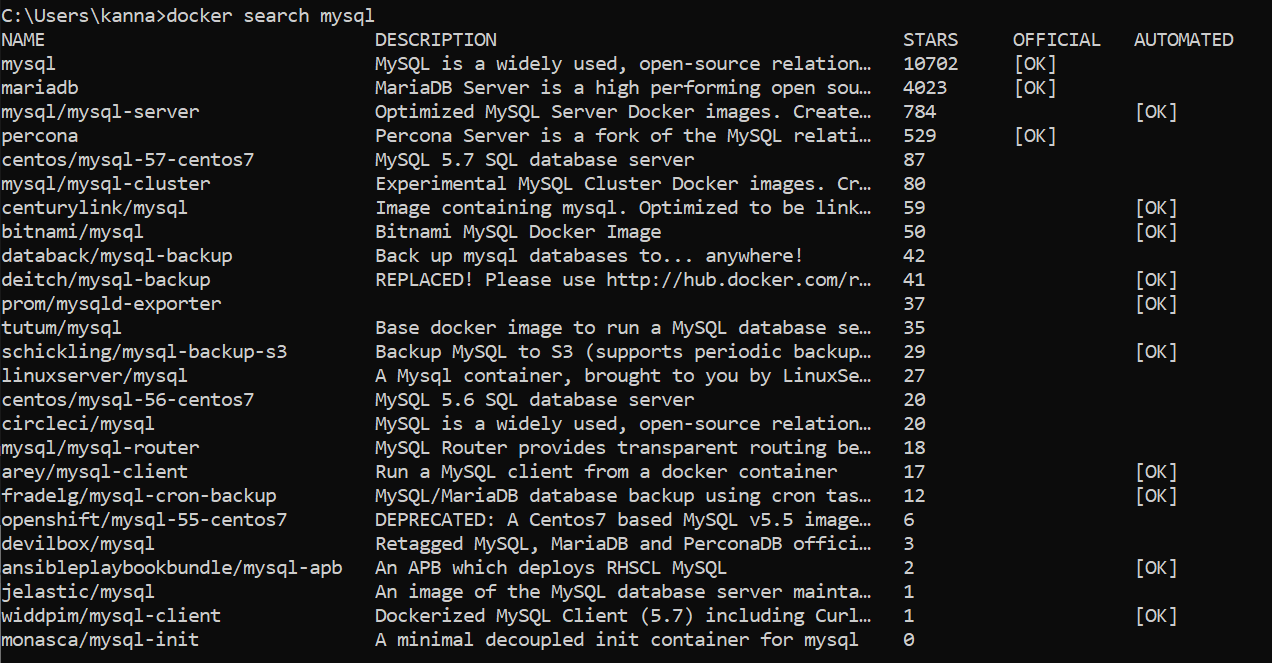
***docker pull imageName***

eg: docker pull mysql

here mysql is an official image

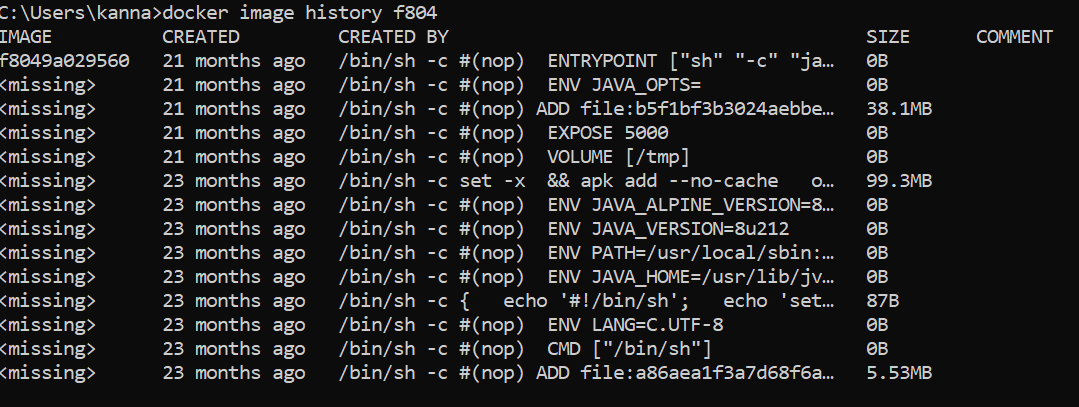
now if we want to search images we can use below command and make sure if it is official.

***docker search mysql***



If we need to get all the steps involved in creating an image then use the below command

***docker image history imageId***



If we want to see the configuration related to the image then following command

***docker image inspect imageID***

In order to remove the image from local memory, then use the below command

***docker image remove imageId***

**Playing with docker container**

Previously we were using the below command format to run a container..

***docker run …***

but we can also run a container by including or specifying container in the command

***docker container run……….***

We can **pause** and **unpause** the docker container

***docker container pause containerId***

***docker container unpause containerId***

We can inspect the docker container using below command.

***docker container inspect containerID***

Another interesting command is prune. This is used to remove all the stopped containers.

***docker container prune***

Earlier we have see docker stop command which gracefully shutsdown the docker container. This means demon will take sometime to gracefully shutdown container.

***docker container stop containerID***

We can use docker container kill command to kill the command immediately without giving it anytime. We generally won’t use this most of the times.

***docker container kill containerID***

Now there is a docker policy for restarting a container. If we have a container in docker before restarting a docker desktop, then based on the --restart policy, we can tell docker demon whether to bring specific containers up after it restarts.

***docker run -p 5000:5000 -d --restart=always in28min/todo-rest-api-h2:1.0.0.RELEASE***

Here restart can have two values **always** and **no.** The default value is no.

Now there is a command called ***docker events***, to see what is happening with the containers.

There is another interesting command called docker top, this one checks what is the top process which is running in a specific container.

***docker top containerID***

Another interesting command is docker stats. It give memory, cpu usage percentage of running containers.

***docker stats***

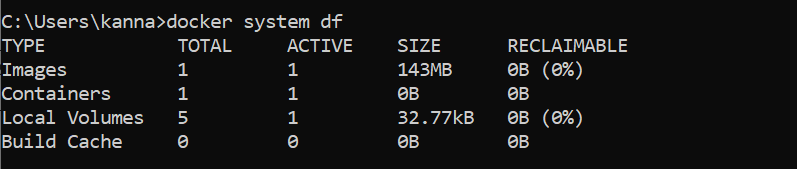
Now for a specific container if we want to allocate cpu and memory limits, we can do that using below command

***docker run -p 5000:5000 -m 512m --cpu-quota 5000 -d in28min/todo-rest-api-h2:1.0.0.RELEASE***

**Note**: 100,000 -100% CPU, 5000 – 5% CPU

Another interesting command is ***docker system df***

This command gives details of all the things docker demon manages.



Until now we have been playing with images that were already created. Now lets focus on creating our own images.