docker pull node

Pulls the image from docker hub to local machine

docker run -it node

Starts interactive session with the node. This command basically starts a container from the base image

docker ps -a

it will show that the container is running from the node image we downloaded

in the above case what we have done is we have built from the base image. However in practice we want to custom the base image and build container from the custom image

docker build <path to docker file>

what it does is, it will build the custom image from the docker file

if it is in the same path then the command will be like

docker build .

docker stop <container name>

docker run -p 3000:80 6ab5a906119d

Need to mention the local port which will map to exposed image port

It is not necessary to mention EXPOSE 80 in docker file

We can expose the port using above command

###############################################################################################################

Lets say we want to make change to our node server source code.

We make the change and run the container. We will observe that the chnge is not reflected

It is because we need to rebuilt the image. Images are locked and are read only.

So, we run below command again and it creates a new image

docker build .

-Docker images are layer based. Every instruction is layered. When we rebuild any image, it fetches data from cache if no changes are made.

If one layer changes all subsequent layers will change

npm install depends on package.json and not on server.js. if we make changes to server.js, it has to run np install again

FROM node

WORKDIR /app

COPY . /app

RUN npm install

EXPOSE 80

CMD ["node", "server.js"]

so we can optimize below code like this

FROM node

WORKDIR /app

COPY package.json /app

RUN npm install

COPY . /app

EXPOSE 80

CMD ["node", "server.js"]

Now, even if we make change to source code, npm install will run before that

#######################################################################

docker run -p 3000:80 <image>

docker run -p 8000:80 <image>

It will run two containers on the same image.

Also, when we run the above command, we cannot interact with the terminal. We have to open a new terminal. But we don’t face the same issue with docker start command

So, docker run -> attach mode

docker start - > detach mode

In attach mode, whatever commands we run in browser, we can view in the terminal

docker run -p 8000:80 -d <image> # it will run container in detach mode

we can attach the container again using

docker attach <container name> (or)

docker logs <container name>

To keep on listening use follow mode

docker logs -f <container name>

docker -i -t <container name>

docker -it <container name>

If we have an interactive docker we can always restart using attach mode

However, in attach mode as well, we can’t interact. So, we have to use interactive mode also

docker -i -a <container name>

docker rm <container name> - > it will not remove a running container

so we first stopt the running container

docker stop <container name>

docker rm <container1> <container 2> <container 3>

docker images -> lists all images

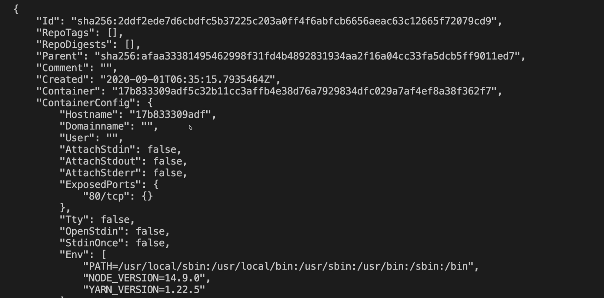
docker rmi <imageid> - > removes images

We can remove images only if they are not used on any container. We first need to remove container and then we can remove images

docker run –p 3000:80 –d –rm <image id>

it will start container in detach mode and once container stops, it removes it automatically without having to remove container manually

docker image inspect <image id>



docker cp .\hello.txt wizardly\_gates:/

docker cp wizardly\_gates:/hello.txt test

Assigning name to a container

docker run --name hellocontainer f86edf1db28c

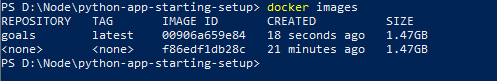
Tags – Lets say we have a node image. There are multiple tags associated with it. Lets say we want to download node image with tag 14. So in the rocket file, we can mention

From node:14



While building image with a tag

docker build -t goals:latest .



Images can be shared in two ways.

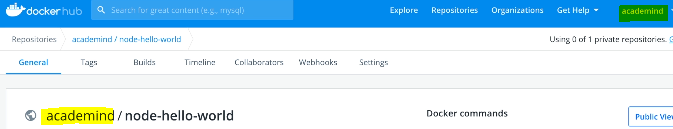
1. Share a docker file
   1. run docker build
   2. docker file instructions may need surrounding files/folders like source code
2. Share the built image
   1. No built step required

docker push <image name> -> upload image to docker registry like docker hub

docker pull <image name> -> download image from docker registry to local

Dockerhub> repository> create repository> there is a account name

we have to push image using that name



1. docker build -t myimage .
2. docker run --name mycontainer myimage
3. docker stop mycontainer

Image and container have a custom name.

1. docker build -t myimage:latest .
2. docker run --name mycontainer --rm myimage
3. docker stop mycontainer

Creates an image with tag, create container with name and it will automatically remove container once the container has stopped

Tags – An image can have a name and multiple versions of that image attaches on the same name