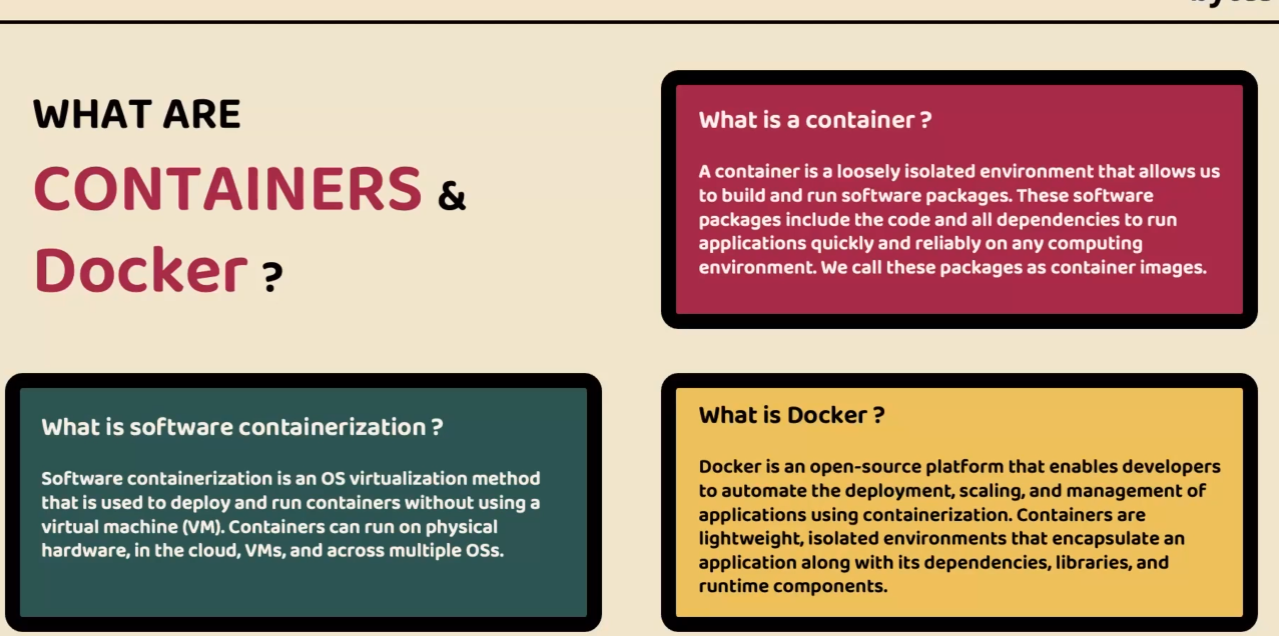
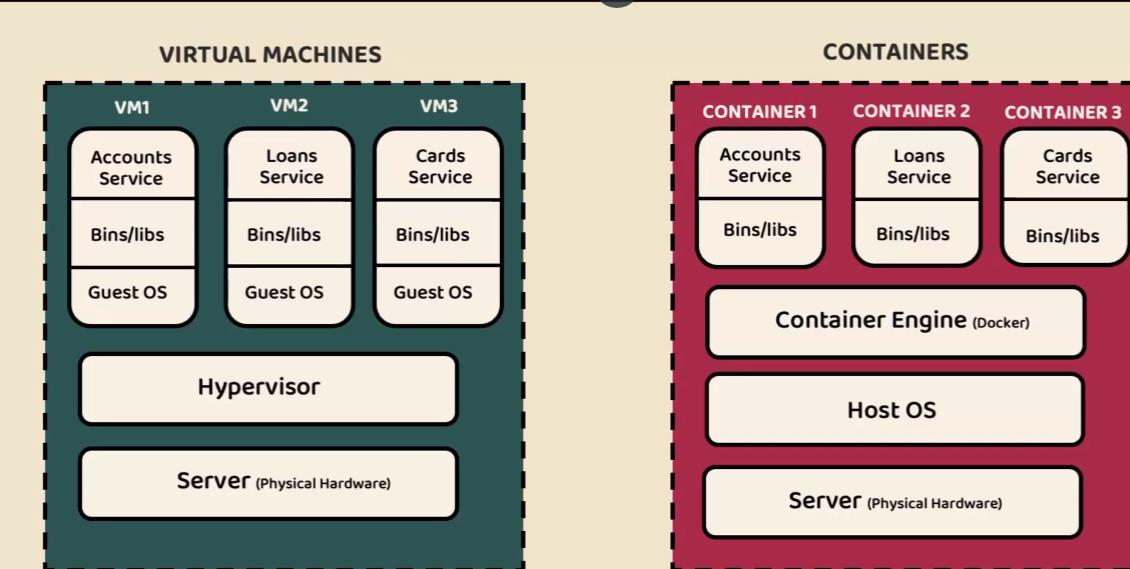
What is Docker, container and containerization ?

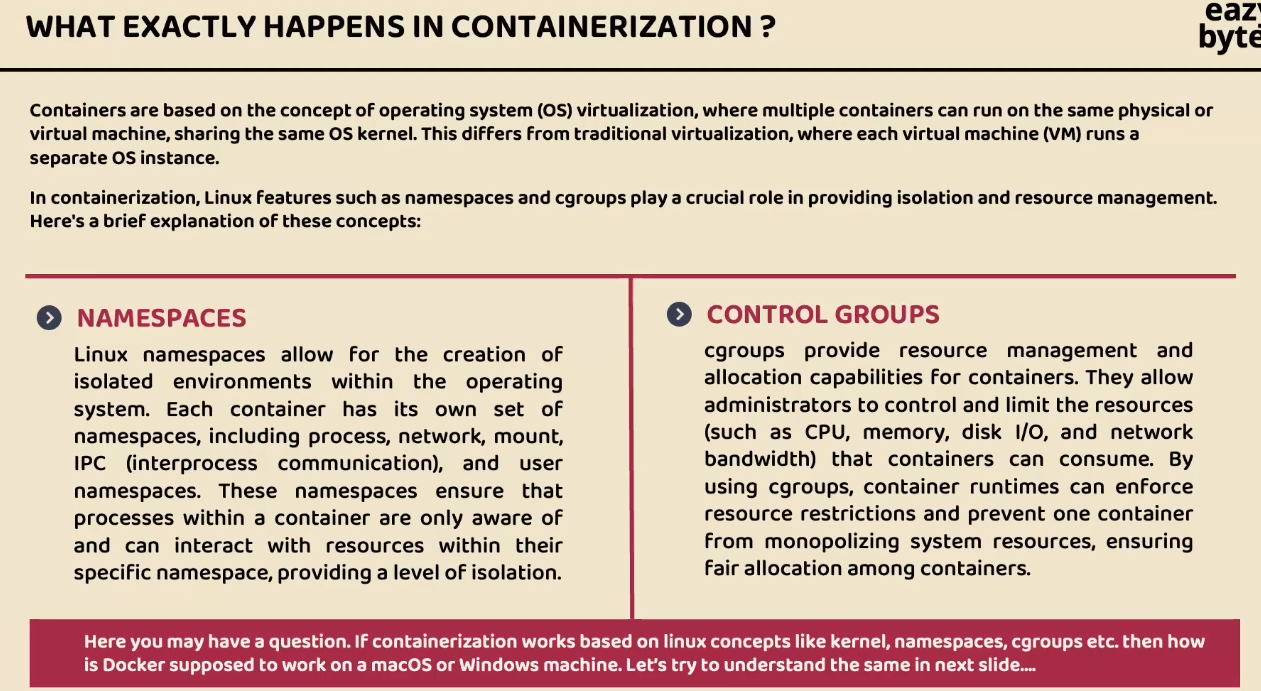
A container is an executable package with all the dependencies and other softwares such as jdk so that it can run in its own isolated environment. And it is independent of the OS and hardware configurations. And these container runs in an environment provided by the docker called Containerization.

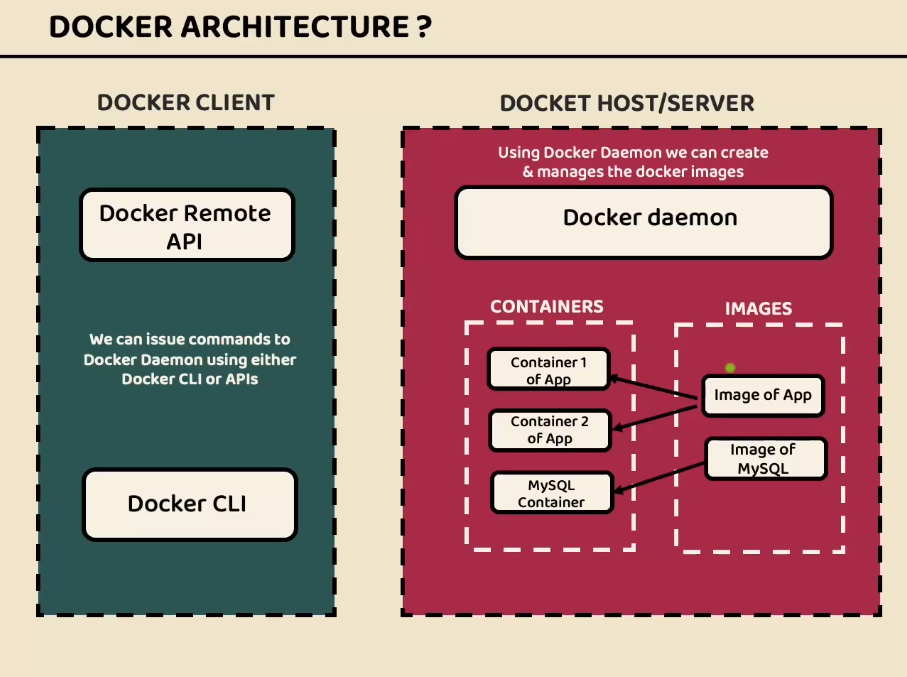
While this is not possible in the monolithic architecture bcoz such application require separate environment which contains JDK and need to be run in separate server and depends on OS and hardware.

Docker is responsible to create containers of our webapp which is called docker images. It not only create docker images for webapps but also for existing applications like mysql and many other . Docker contains Docker server / Host which is responsible for creating the images and also maintaining them. The docker generates the image through CLI command line interface provided by the docker. Where these commands are passed to Docker server to generate images



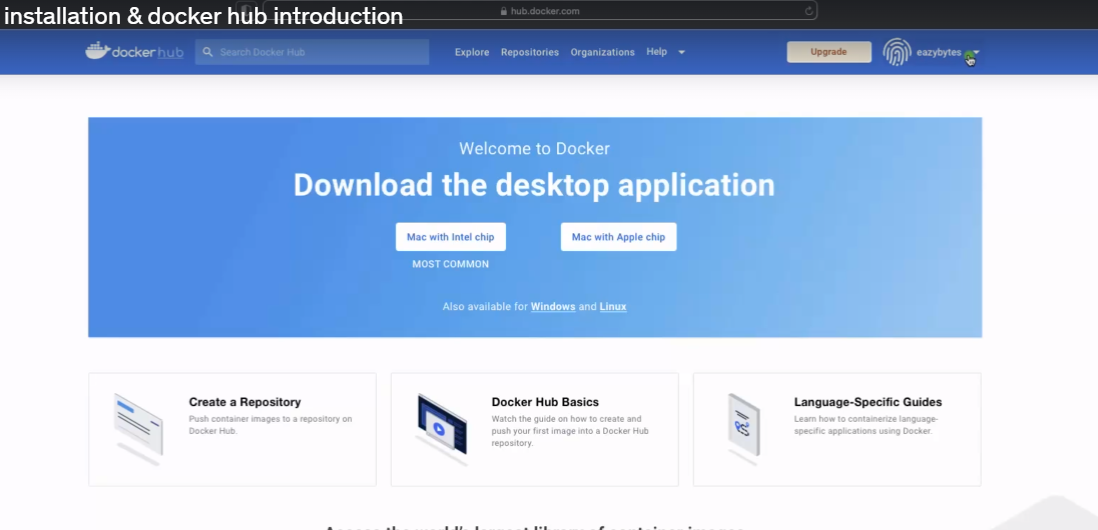






Docker also provide with its own Docker hub/ registry so where we can push the new images and download the same on different machines in this way portability issue is resolved.

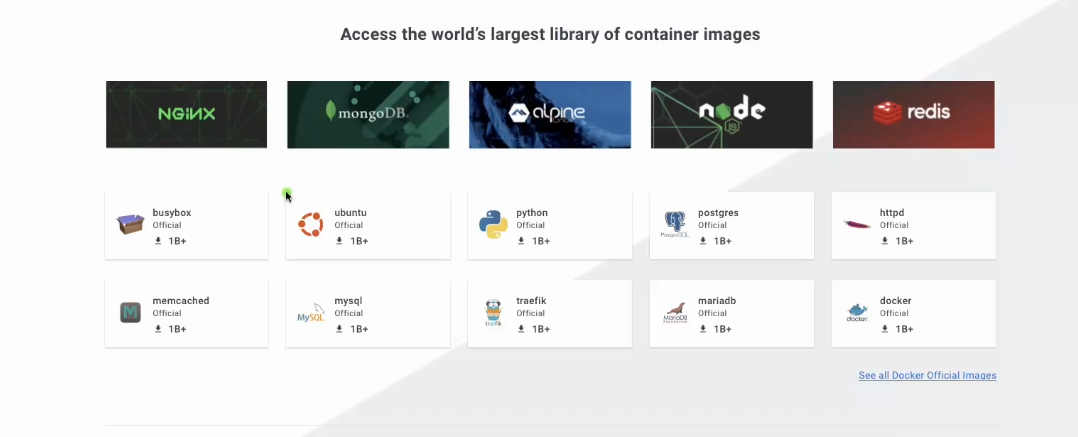
Download docker desktop from official website. On the official web site itself we have login option where it will take you the Docker hub/ registry



Where you can create your own repository and store all the docker images

In this course the username is eazybytes

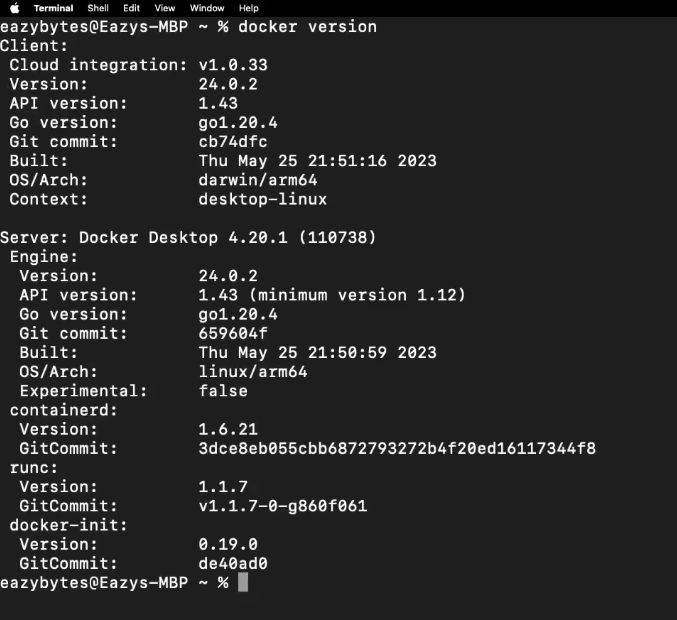
Docker hub has many third party app images as below



After docker installation on local system a docker icon will start appearing click on icon you will options like dashboard. Then click on dashboard then you can see below application also called as Docker desktop which contains all the images and containers as below



Open command prompt and check for the docker version



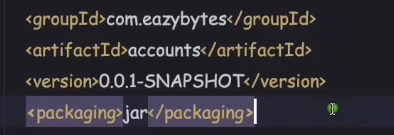
Now once everything is set we can start generating our microservices images this can be generated by using 3 ways

1. Docker file 2) Build Packs 3) Google Jib

Docker file :

First lets add a packaging

tag to generate as jar file in pom.xml



We can create jar using maven commands in terminal first install maven and setup in environment variables

Mvn clean install

Jar generated is called fat jar bcoz it contains all the dependencies required for the application. Except jdk that will have to specify in docker file

Now you can run this jar using command java -jar packagename.jar.

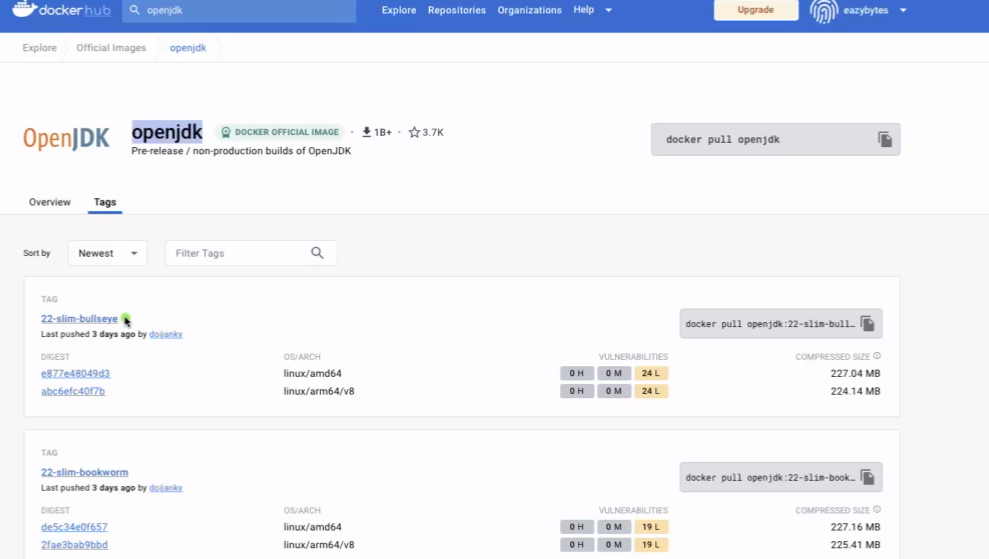
This knowledge is required for creating a docker file.

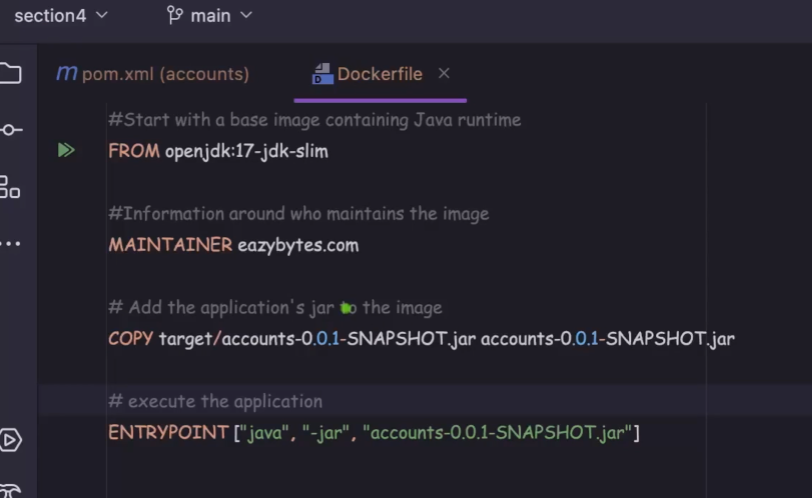
Now lets create a docker file. File name has to be “Dockerfile” without any extension

Docker file contains instructions to generate docker image.

Very first instruction is JRE required to run the image as images will run in its own isolated environment. Now this JRE is already a docker image in docker hub so we can just use it by specifying it in docker file. We can instruct docker by adding From command

Openjdk is image name and 17-jdk-slim is a tag name. its available in docker hub as below





Now copy command will copy the app.jar from target folder and creates an docker image you can give same name or some othr name

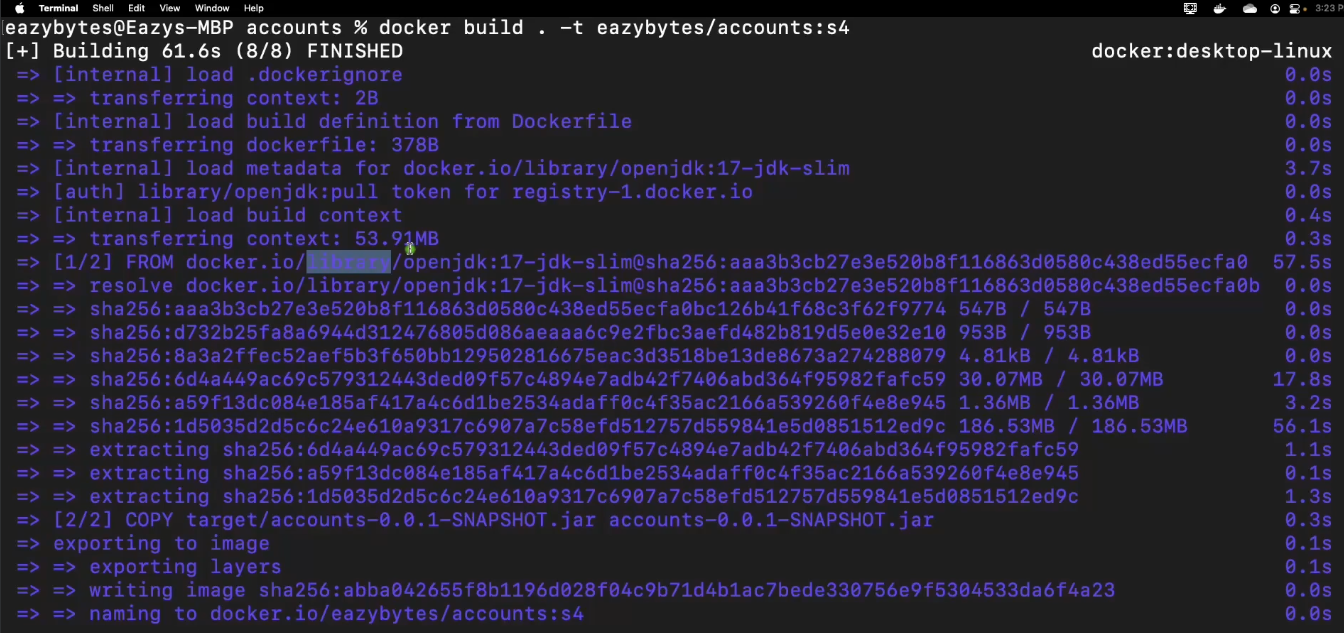
To generate a container we need to have Entrypoint command. It takes an array of string usually an command to run this jar

Now to generate a docker image we need to go to location where we have created Dockerfile and then in command prompt docker build command needs to be executed.

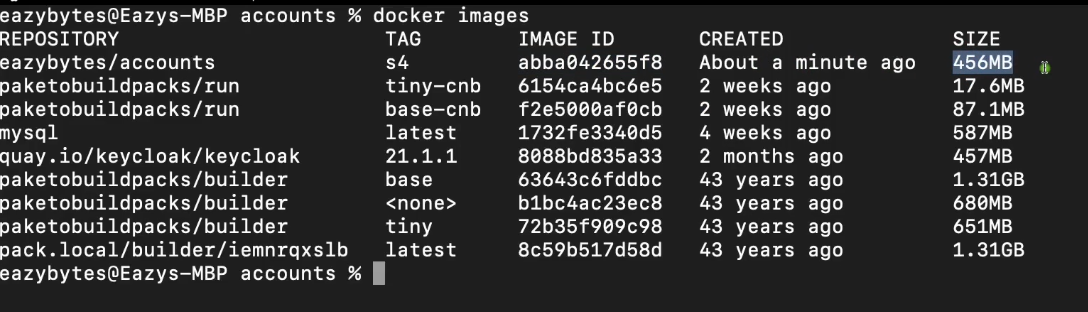
docker build . -t eazybytes/accounts:s4

-t stands for tag name that we are going to provide for the image

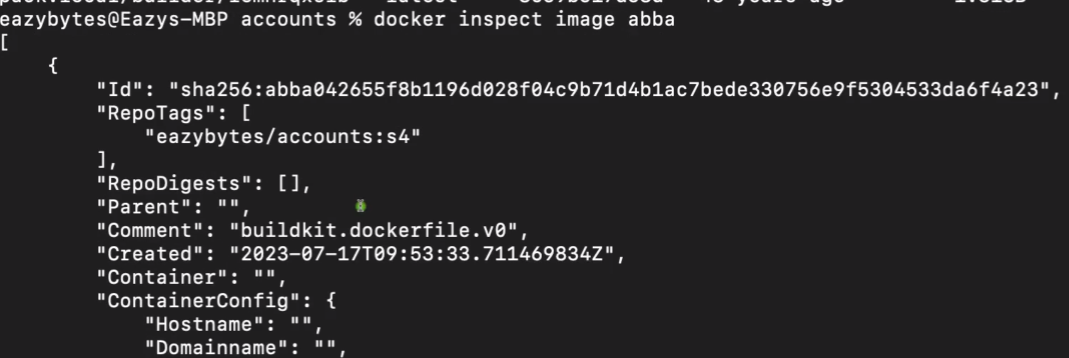
Tag name we need to follow like “username/aapname:version”, username is docker hub username



You can check docker images by command docker images



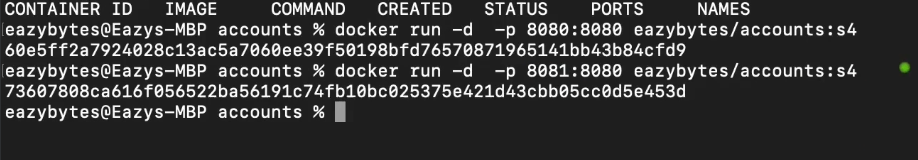
Then you inspect this image whether all the provided 3rd party dependencies are proper or no like jdk 17 etc



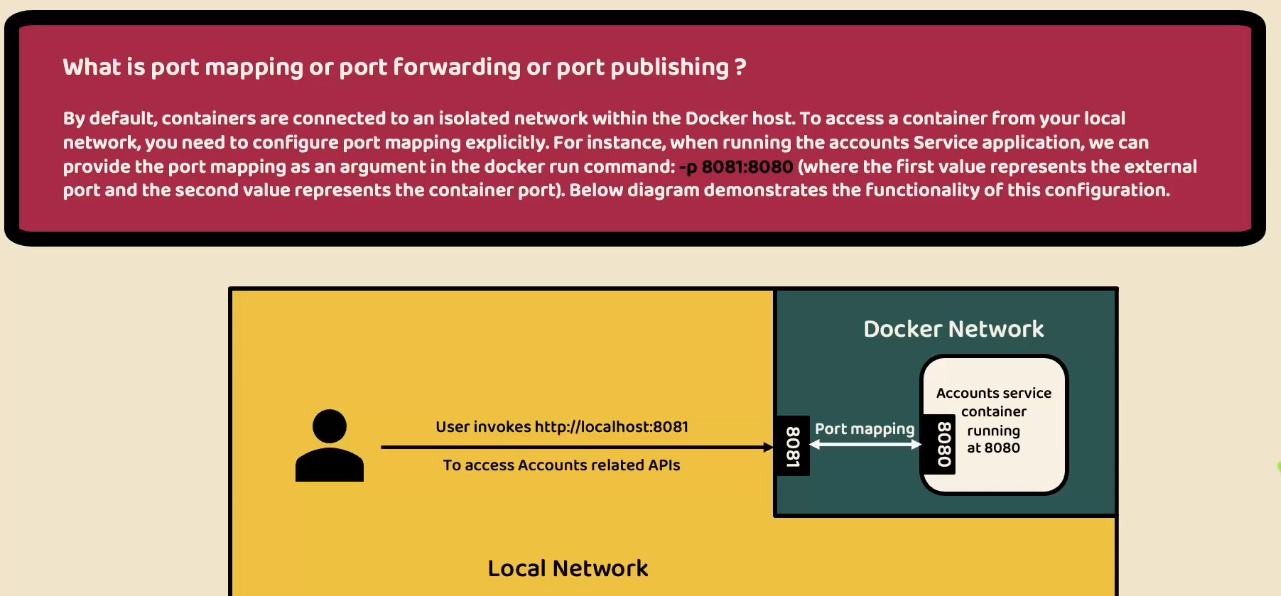
abba is first 4 character of image id

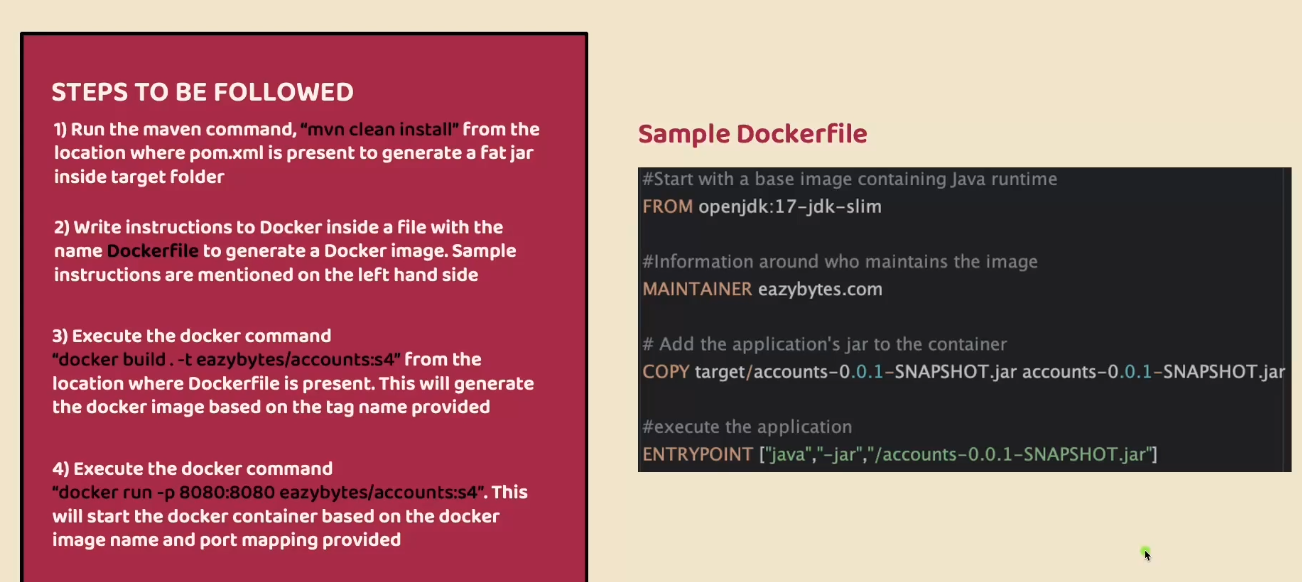
You can also check in Docker desktop search for name

Now to create the containers of the image we need to run the below command



Two containers are created and running at port 8080 and 8081 respectively





Disadvantages of docker file

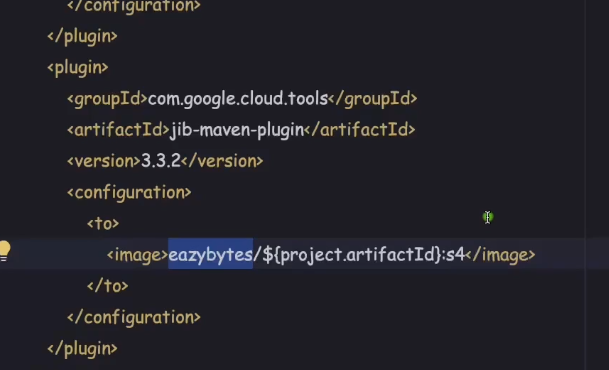
To write a docker file we need to be expert in Docker concepts and commands as Devops are involved

We need to created a docker file for every individual microservice

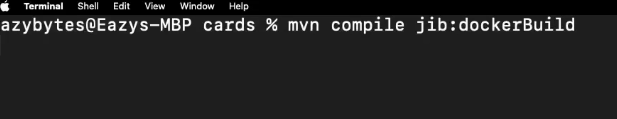
Hence there are readily available tools to generate the docker images one such tool is google jib

Google Jib website - <https://github.com/GoogleContainerTools/jib>

Google jib works only on Java application for documentation goto maven

We need to add plugin related to google jib in pom.xml we can get this plugin from maven documentation 

Now command required to create a image is



**We can generate the docker image even if we don’t have the docker installed in our system**

Using below command

Mvn compile jib : build

But since docker is not installed on localsystem hence it will look for remote repository that where image will be stored. We need to update configuration in pom.xml file as below for docker hub repository

**<configuration>**

**<to>**

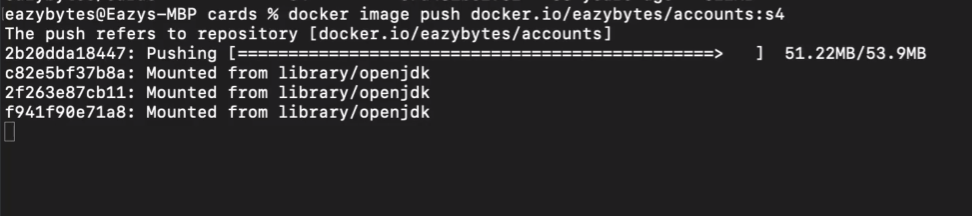
**<image>docker.io/my-docker-id/my-app</image>**

**</to>**

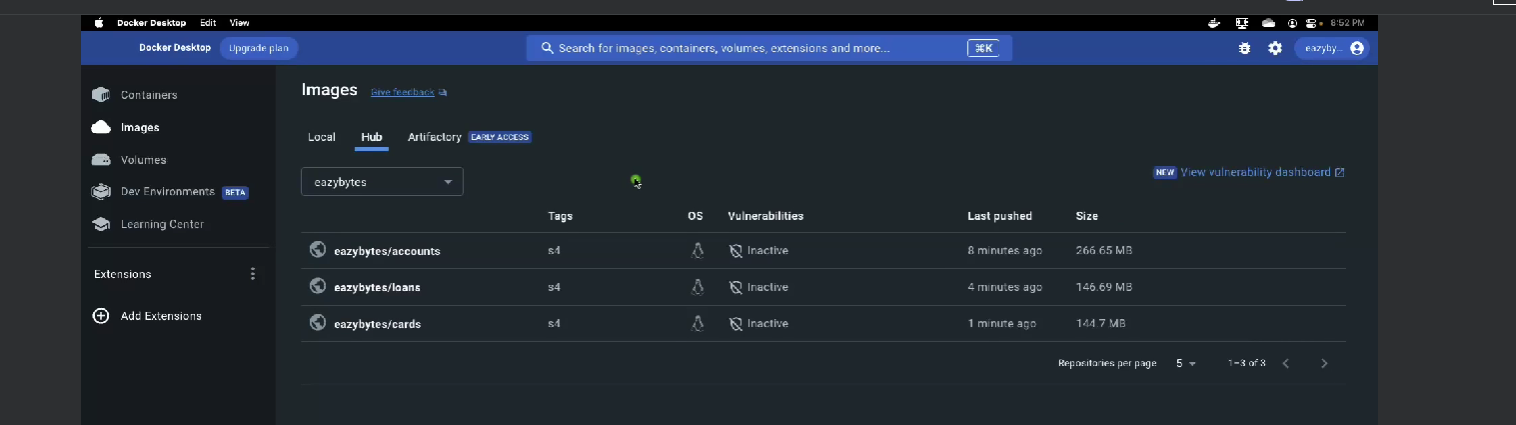
**</configuration>**

My-docker-id is the username of docker hub and my-app is the image name

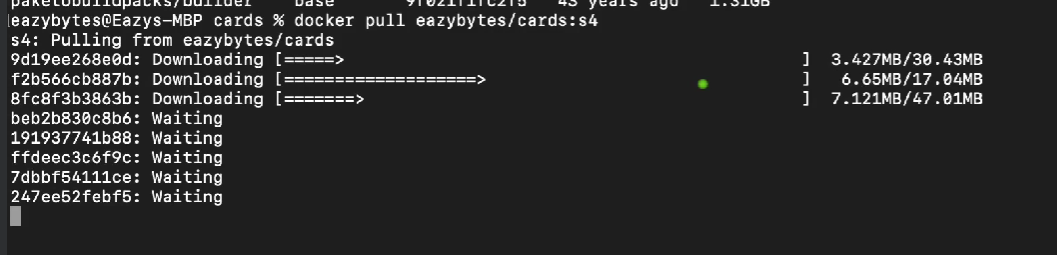
**Pushing docker images to docker hub**



Now after pushing the images to hub you can check on docker desktop in hub tab as below

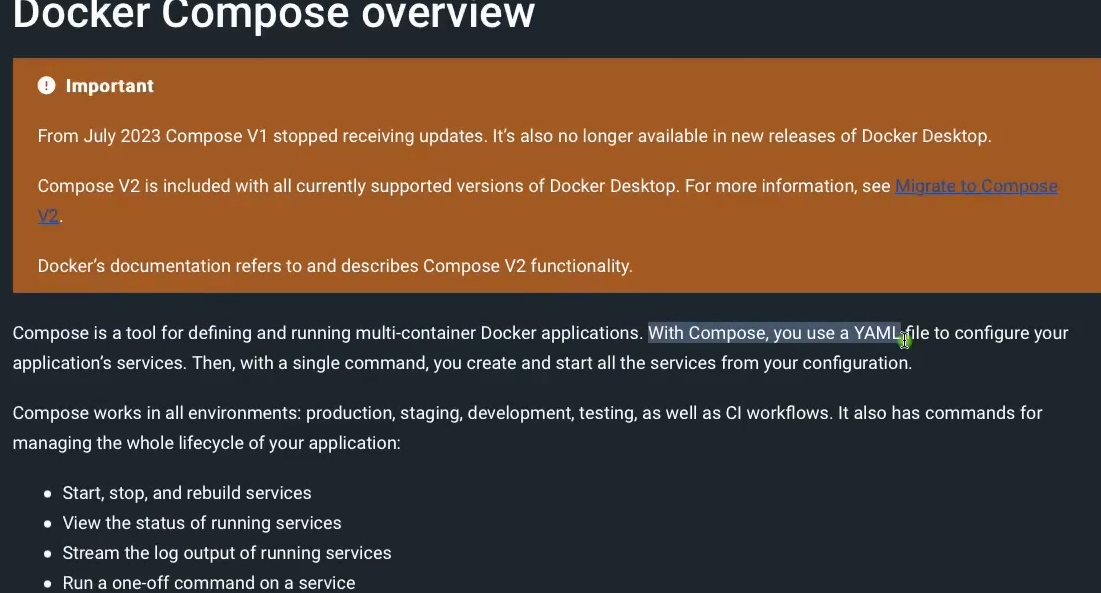


To pull docker images in our current system

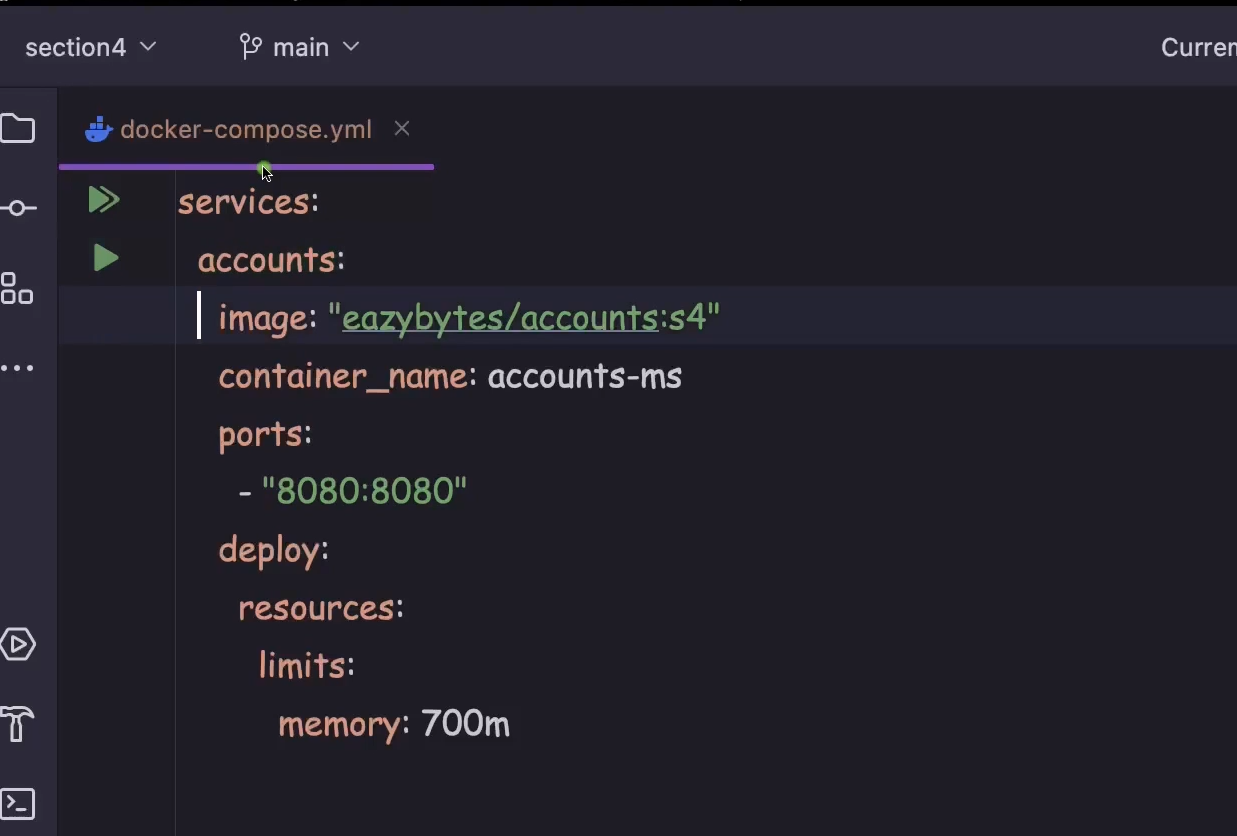


**Docker Compose tool**

When there are multiple microservices then executing run command for each and every microservices is cumbersome process so instead we can define these in single yaml file and can be run by Docker compose tool



We need to create compose file with the name docker-compose.yml we should follow the same name.

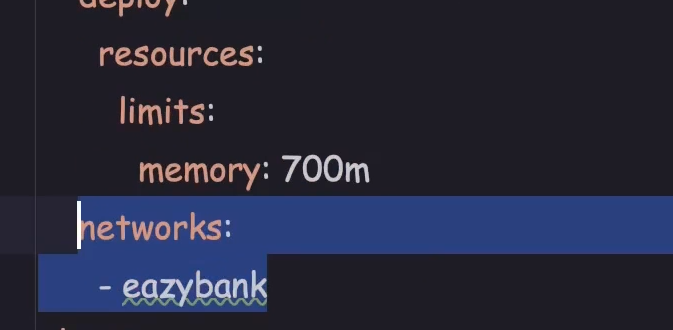


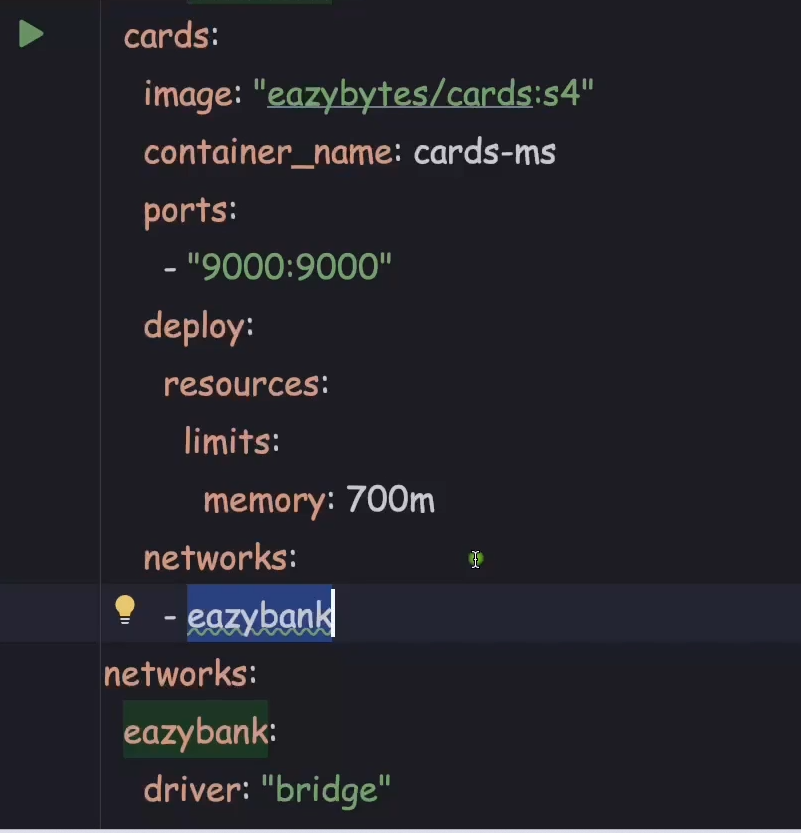
Sample docker-compose.yml file

It should start with services then followed by service name accounts then image name that you have created and also you can specify container\_name which is optional else random name is assigned. Port mapping is assigned we can have this multiple if we want to run multiple instances then just below the first mapping add same with different port mapping “-8081:8080”.

Then also we can specify the resources for each container with certain limit for eg in the above we have specified 700mb memory for accounts container

As we know that each microservices is going to run in its own network environment in order to enable communication among these microservices we need to specify network which will be common among all the microservices



Now add the root element with networks and driver as bridge and same bridge is defined in each service so that these services use the bridge to communicate with each other. 

Now to run to all the services defined in yaml file goto the location where yaml file is present and need to execute below command

docker compose up -d

to shutdown all the services

docker compose down

if we want to stop the containers

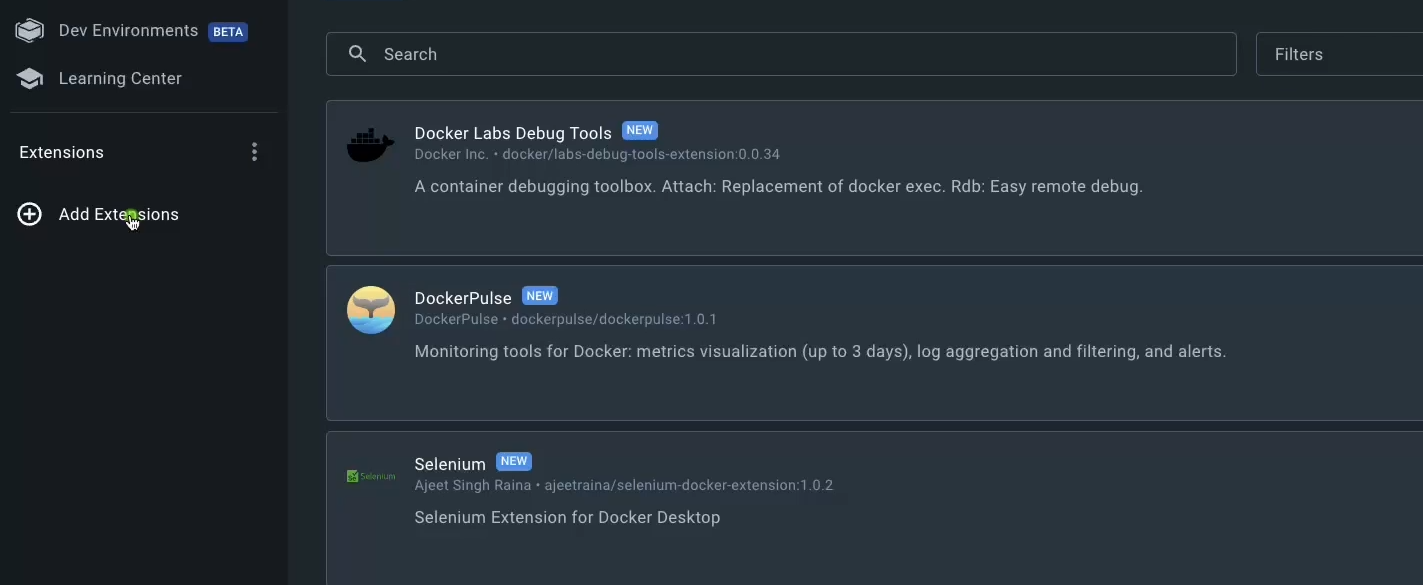
docker compose stop

To restart the containers

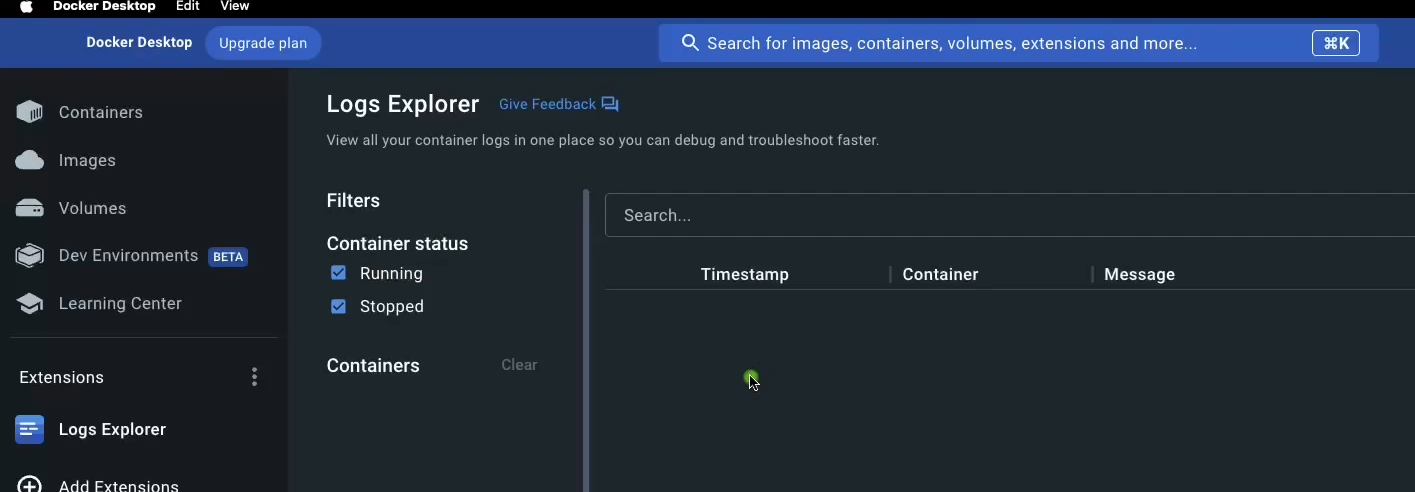
docker compose start

For more docker commands check PDF

We can also add plugins to the docker desktop one such is logger which is used for console for all the services



Then search logger and click install



After starting all the services you can see log as below

