Docker:-

Docker is an open platform where we can develop, test, share my application. We can say that it is kind of VM(Virtual Machine). There are two main concepts in Docker,and entire workflow is revolving around them

1. Images
2. Containerization

Images:-

* Lightweight
* Standalone
* Executable package

It includes everything required to run our application including code, tools, liabraries, runtime, system tools, even Operating system. We can say that it’s a recipie of any dish. To run this project we need an environment or place where we can run our app. So for this containers come into place

Containers:-

Docker Container is a runnable instance of docker image. It provides an environment where our Docker image can run. We can create multiple instance or containers of a single Docker image. It is basically provides an execution environment for docker image

Volumes:-

It is an data persistence storage mechanism that shares the data between docker container and the host machine or in between multiple container instances

Docker Network:-

Its basically a communication channel that connects different containers together

DOCKER WORKFLOW:-

1. Docker Client

This is a place where we execute commands to give instructions to docker to perform any task

1. Docker Host

This is a place in which all our instruction that was given from the Docker client, executed and run i-e managing containers and all the docker related task are done in this Docker Host

1. Docker Registery(Docker Hub):-

It is a centralized repo for docker images. In which we store, share docker images

Note:-

Download and install Docker Desktop on your machine.

Commands:-

1. docker pull image\_name

this is for cloning any image from docker hub to our machine .

1. docker run -it image\_name

this is for creating a docker container and run the image in this container and move within the Operating system inside our machine. After running this step you will see that we will use the completely different Operating system in our machine. For eg if we have an ubuntu docker image so after running this command we will see ubuntu terminal to work . experiment this in visual studio code.

1. docker run image\_name

for running images inside a container only

1. docker build -t folder\_name .

this command is used for building your image. Here folder name means image name and you give that folder name where Dockerfile is present. The . denotes the docker that you casn find the Dockerfile in this directory .

1. docker run -p hostMachinePort:containerPort image\_name

this command is used for port mapping between host machine.

and the running docker container.

1. docker ps

this command is used for check all the running containers only

1. docker ps -a

to show all the containers

1. docker stop containerId

this is used to stop abny specific running container by just passing the starting 3 digits of container id

1. docker container prune

this will remove all the containers except the running ones

1. docker rm continerId –force

this will use to remove any specific container

1. -v “$(pwd):/app”

this telling docker that mount/link the entire current working directory to docker image files which present in the app directory in files tab of docker image. So the latest changes in our local code immediately reflect in the docker image files in real time. Here -v stands for volume

1. -v /app/node\_modules

Here we are creating a new volume for node modules. So that every time we run a container it picks the node\_modules from the volume. If there will be a new dependency so they will auto update in the docker image and wont require to re-install the node modules from the local code

1. docker login

to authenticate with docker

PUBLISHING YOUR IMAGE:-

1. docker tag image\_name user\_name/image\_name
2. docker push user\_name/image\_name

DOCKER COMPOSE

Another cool feature of docker is docker compose. It reduced all the manual work we did in the past to create Dockerfile . It automate the entire process. It automatically create docker image and run into a container by himself by just running a “docker compose up” command .

Commands:

1. docker init

this command is used to init docker cli and create Docker and yaml file .

1. docker compose up

this command automatically create an image and run in a container.

But in this there is an issue that if we make any changes in files it wont be auto update in the container files even after setting volumes in the **compose.yaml** file. Every time we need to re-start the container. To solve this we can use a new feature docker compose watch

DOCKER COMPOSE WATCH

It listen all the changes immediately we make in the host machine and update in the container . It has 3 actions

1. sync

when we need immediate code reflection in our running container during active development of an app . sync is better option

1. rebuild:-

rebuild is best when we are working with configuration files like package.json, environment variables etc.

1. sync + restart

**compose.yaml file:-**

this file is used to enable compose watch and we define all the settings and configuration of our services including port, environment, web, api, db services , watch settings and action etc.

1. docker compose up

this command is used to build and run the image in containers , even if we have multiple services (for eg web, api, db), this one single command and one file create three images and run three containers simultaneously . but this also wont sync changes in real time. To enable real time changes in web or api or db we ll use the below command

1. docker compose watch

this command is used to see live changes we made in frontend, backend or any where at real time . it will watch all the files we mentioned in every service’s watch block in compose.yaml file. we can see and test the changes by running two terminals side by side. In one terminal we ll run **docker compose up** and in second terminal we ll run docker compose watch to see the live frontend and backend changes

DOCKER SCOUT:-

Basically it’s a new docker feature that checks the vulnarabilities in your image to prevent cyber attacks . We can just analyze our image via this feature .

GENERAL NOTES:-

* docker containers run in isolated environments and by default they didn’t expose their port to host machine. It means even if we expose a port in Dockerfile so that Docker container listen this port but the port is only accessible inside the container not outside so our host machine cant access this port
* To aware our host machine about the port that docker container is listening, we need to use the concept name port mapping. Port mapping is a method to share the port between Docker container and host machine.
* **RUN** is used to execute commands during the build process of a Docker image, while **CMD** is used to specify the default command to run when a Docker container is started from the image.