DOCKER :-

- Docker is a set of platform-as-a-service products that uses OS-level virtualization to deliver software in packages called **Containers**
The software that hosts the containers is called **Docker Engine.**

- A **container** packages code and all its dependencies into a single unit, thus letting an application run quickly and reliably from one computing environment to another. This makes such applications portable between machines.

- Docker is installed as Docker Desktop on windows, and it runs using WSL2.

- For various tasks to do with docker, we have GUI, as well as commands :-

- To check version and other details about the docker software installed on a system :-
  `docker version`

It will show details about 2 components of Docker — Docker Client & Docker Server (Docker Daemon)
Docker Daemon runs in a Linux virtual machine.

When we say Docker, we are talking about the entire ecosystem of Docker. It includes :-

i) Docker Client,
ii) Docker Server (Daemon)
iii) Docker Machine,
iv) Docker Hub,
v) Docker Compose.

Docker client is a tool to enter commands and reach out to Docker Daemon

Docker Daemon is a service that runs on our host OS. It is responsible for running containers (& allocating resources to them) & managing them (starting or stopping containers)

Command to execute/run any container :-

[docker run <image_name>

Ex:- [docker run hello-world

This command first searches if the image to be executed exists locally. If it does not exist, it downloads the image, saves it into cache, and creates container from that image.

If it is already present locally in the cache, then directly container is being created from the image.

IMAGE & CONTAINER :-

Image is the blueprint of a container. It is simply a set of files (ex:- bin, etc) that is used to create a container. It does not consume resources.

⇒ Container is the instance of an Image. Container consumes resources. It is in the running state (exists in the memory). It is an isolated environment.

19:00

⇒ [docker run image_name

Ex:- [docker run hello-world

This command tells the docker server to containerise the image (whose name is mentioned) if it is present locally.

If not present locally, the image will be downloaded from docker hub, and then its container will be made. After downloading, it is stored in the cache, for further usage.

⇒ CONTAINER : IN DETAIL :-

35:00

⇒ Two things using which resources of a machine is divided by OS are :-

i) Namespacing , ii) Control Groups.

i) Namespacing is used for isolation of resources b/w different processes.

ii) Control Groups are used to limit resources per process.

Both the above are features of Linux.

⇒ A container is also a process.

⇒ A container will have specific set of resources (CPU, Memory, Hard Disk), which is isolated (using Namespacing) and limited using Control Groups. —— all these is facilited by Docker

38:00

⇒ IMAGE : IN DETAIL :-

⇒ A container Image contains 2 things :-
i) File system snapshot , ii) Startup command

i) File system snapshot :- is a set of files, similar to the ones that are obtained on any software installation. Ex :- bin, lib, exe files, etc...
(these files are necessary for software execution).

ii) Startup Command :- It is the command that will be executed (by default, if not overridden) when we excute docker run... for that image, i.e, when containerising the image.

⇒ When container is made from the image, entire File system snapshot in the image is copied (exactly) to the hard disk resource allocated to the container.

⇒ After a container runs, the 1st thing done is that the Kernel assigns resources to it.

43:50

⇒ Running command inside a container.

⇒ Start up command of an image can be overridden with another command

⇒ When we do docker run ... , the default startup command of the image will get executed, and after the completion of the startup command it brings us back to our original main terminal.

Suppose, we want any command to be run on the container, we can write the command after I with run command as follows :-

[docker run busybox echo hello debanjan,

command that we want to execute upon the container that we are running/making.

⇒ The above command means — containerise busybox and instead of running its default startup command, run echo hello debanjan.

This is called overriding startup command.

Ex:

=> docker run busybox ping google.com

=> docker run busybox ls

This command will all the directories present in the busybox container (& not windows ones), proving that command is run on the container.

Command to get details of all the running containers :-

To know which all containers running currently, the command is :-

```
docker ps
```

It states the container ids, image names (of containers) command run, time of containerisation, & the ports they are using.

To get details of all the containers that are currently running, and were run & closed in past (in same session), the command is :-

```
docker ps --all
```

states the closed containers too.

50:15

Command to manually stop a running container :-

```
docker stop 8cab51e4df
```

container id of the respective container which is to be stopped. Can be obtained using `docker ps` command.

To force stop a container immediately, command is :-

```
docker kill <docker_id>
```

docker stop v/s docker kill :-

⇒ docker stop puts a request to the running container, to stop normally after finishing all the task it has pending / ongoing.

But, docker kill forces the container to exit immediately, by terminating all the pending / ongoing tasks unfinished.

After running docker stop, system waits till 10 sec. for the container to finish all its tasks & end normally, else, docker kill is run implicitly.

⇒ docker run command is actually combination of 2 commands :-

[docker run ≈ docker create + docker start

⇒ First, a container needs to be created, as follow :-

[docker start hello-world

If image is present in cache (else will get downloaded) then its container will be created, by this command, and a string will be returned.

The container still now hasn't started.
To start it, the command is :-

hash string of a started container obtained after any container being started

`docker start -a 7b28efc294df248d16fmp...`

This flag is used so that we can see the output, that comes in the container, when it is started.
If this flag is not provider, container will start, but we cannot see any output.
It means :- attach to the container and give containers output to my terminal.

54:20

To check the logs of any running container :-

`docker logs <docker_id>`

58:10

To execute commands inside a running container :-

Command is :- container id of the container inside which we want to execute our custom command. | command that we want to execute

`docker exec -i -t c1f21885266c sh`

(c1f21885266c : id of redis)

"sh" command opens a shell

We can then execute & see commands like: pwd, cd../, ls

To exit from this shell → ctrl + D

-i flag use :- It is used such that the next things that we gonna type in the docker CLI, those must be directed to STDIN (standard input)

of the docker process, that is the container whose id is being mentioned.
Also, the STDOUT (standard output) of the docker process to our docker CLI, using the same flag.

⇒ -t flag use :- The output shown will be pretty formatted, such that it is easy to understand.

1:00:30