**Docker**

🡪Docker is a containerization tool.

🡪Docker is a client-server architecture (one server-many clients).

🡪Developer will write the docker file.

🡪Docker is used for developing, running and delivering the applications.

🡪Container is a lightweight, portable, and isolated environment that runs an application and its dependencies.

🡪Container is a way to package application with all the necessary dependencies and configuration.

🡪We can create n number of containers in the system.

🡪The package is in the form of Portable Artifact, easily shared and moved around development team and operations team or in between development team.

🡪It makes development and deployment team more efficient.

**Where do containers live?**

🡪Containers live in container repository.

🡪Many companies hosts there private repositories to store the containers.

🡪There is a public repository for Docker, i.e., DockerHub.

**VMWare:**

🡪A VM is a software-based simulation of a physical computer. It runs on full OS on a hypervisor.

🡪It can run any OS (Linux, Windows, MacOS).

🡪High availability and load balancing.

🡪High resource consumption.

🡪Slower boot and scaling time.

🡪Licensing costs for VMWare products.

**Why use Docker?**

🡪**Consistency**: Works same way in development, testing and production.

🡪**Isolation**: Applications run in isolated environments, avoiding dependency conflicts.

🡪**Portability**: Containers can run on any system with docker installed.

🡪**Efficiency**: Containers share the host OS kernel, making them lightweight compared to virtual machines (VMs).

🡪**Scalability**: Easily scale applications horizontally by running multiple containers.

**Core components of Docker:**

1. **Docker Engine:**

🡪The core of Docker, responsible for creating and managing containers.

🡪Consists of:

* **Docker Daemon(dockerd):** A background service that manages Docker objects like images, containers, networks and volumes.
* **Docker CLI:** A command-line tool to interact with the docker daemon.
* **Restful API:** Allows programs to interact with the docker daemon.

1. **Docker Images:**

🡪A read-only template with instructions for creating a container.

🡪Docker image is an executable package of software that includes everything needed to run an application.

🡪Docker images are built using the Dockerfile which consists of set of instructions.

🡪A docker image is a platform-independent image.

🡪Docker image is lightweight so can be portable to different platforms very easily.

**Components of Docker Image:**

* **Layers:** Images are made up of multiple layers.
* **Base Image:** The foundational layer, often a minimal OS or runtime environment.
* **Dockerfile:** A text file containing instructions to build a Docker Image.
* **Image ID:** A unique identifier for each Docker image.
* **Tags:** Labels used to manage and version Docker images.

**Docker Image Commands:**

**🡪**Listing all docker images.

**docker images**

🡪If you want to display just image IDs.

**Docker images ls -q**

🡪Pulling docker images.

**docker pull ubuntu:20.04**

🡪Building docker images from the dockerfile.

**docker build -t myapp:latest .**

🡪Tagging docker images.

**docker tag myapp:latest myrepo/myapp:v1.0**

🡪Pushing the docker images.

**docker push myrepo/myapp:v1.0**

🡪Removing docker images.

**docker rmi myapp:latest**

**🡪Pruning docker images:**  The docker image prune command removes the unused docker images from your local machine.

**docker image prune -a**

🡪Viewing docker image history.

**docker image history myimage:latest**

🡪Inspecting docker images.

**docker image inspect myimage:latest**

🡪To delete all the docker images at once.

**docker rmi $(docker image ls -q)**

**Docker Installation on Windows:**

1. **Create a Docker hub account:**

🡪Go to Docker website and create a docker account.

1. **Create 2 repositories :**

🡪Create 2 repositories in the docker account.

1.dev

2.qa

**3. AWS console:**

🡪Go to aws account and create an instance with the name docker and connect with it.

1. **Switch to root user:**

🡪By using the sudo command switch to the root user.

**sudo su**

1. **Display all network interfaces:**

**🡪**Displays information about all network interfaces, whether they are active or inactive.

**ifconfig -a**

1. **Install Docker:**

🡪Install Docker using the following command.

**yum install -y docker**

1. **Check version:**

🡪To check the docker version use the following command.

**docker –version**

1. **Again display all network interfaces:**

**🡪**Displays information about all network interfaces, whether they are active or inactive.

**ifconfig -a**

1. **Start Docker:**

🡪Start the docker using the following command.

**servive docker start**

1. **Default home directory:**

🡪The default home directory of the docker is :

**/var/lib/docker**

🡪Operating System: Amazon Linux 2023.6.20250303

🡪OSType: linux

🡪Architecture: x86\_64

🡪CPUs: 1

🡪Total Memory: 949.5MiB

🡪Name: ip-172-31-7-215.ap-south-1.compute.internal

🡪ID: 0345fd2c-a14f-415e-abae-39ecea8a998d

1. **Docker Images:**

🡪To see how many images are created.

**docker images**

1. **Docker ps:**

**🡪**To see the containers that are currently running.

**docker ps**

1. **Docker ps -a:**

🡪To see all the containers (running and not running).

**docker ps -a**

1. **Image pulling:**

🡪There are 2 ways to pull the image:

1.Pulling docker image from online(official website)

2.Writing the docker file.

**15. Pulling the image from online:**

🡪I am pulling the docker image from online:

🡪nginx(web server) is used to pull the image.

🡪To pull the image from online.

**docker pull iginx**

🡪To check images pulled or not.

**docker images**

🡪To know the complete information of the image.

**docker inspect <image\_id>**

**To create Container:**

🡪**docker run -itd –name deepthi -p 40:80 <image\_id>**

**🡪docker container ps**

**🡪docker inspect <container\_id>**

**🡪docker exec -it <container\_id> bin/bash**