**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.

**Differences between Docker and Virtual Machine:**

| **Feature** | **Docker (Containers)** | **Virtual Machine (VM)** |
| --- | --- | --- |
| **Architecture** | Shares **host OS kernel** | Each VM has a **separate OS** |
| **Performance** | Lightweight, fast startup | Heavier, slower startup |
| **Resource Usage** | Low (uses host OS resources) | High (allocates dedicated resources) |
| **Isolation** | Process-level isolation | Full OS-level isolation |
| **Portability** | Highly portable across environments | Less portable, OS-dependent |
| **Boot Time** | Seconds | Minutes |
| **Use Case** | Best for microservices, CI/CD, and cloud apps | Best for running different OS environments |

**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)**

**Difference between Docker Image and Docker Container:**

| **Feature** | **Docker Image** | **Docker Container** |
| --- | --- | --- |
| **Definition** | A read-only blueprint for creating containers | A running instance of a Docker image |
| **State** | Static (does not change) | Dynamic (can be started, stopped, modified) |
| **Persistence** | Remains unchanged | Changes are lost unless committed to a new image |
| **Storage** | Stored in Docker registry (e.g., Docker Hub) | Exists in runtime and removed when stopped (unless persistent storage is used) |
| **Creation** | Built using a Dockerfile (docker build) | Created from an image (docker run) |

**DockerFile:**

* A Dockerfile is a text file that contains a set of instructions to automate the creation of a Docker image.
* It defines how an image is built, including the base image, dependencies, environment variables, and commands.

**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.

**service 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**

**-----------------------------------------------------------------------------------------------------**

**20-03-2025**

**Pulling Images from Online:**

🡪First connect the aws account.

🡪Login to the docker using docker credentials.

**docker login**

🡪It is used to login the docker.

🡪Now change to the root user using:

**sudo su**

🡪After changing into the root user, now change the directory using:

**cd /var/lib/docker**

🡪Now pull the docker images from online.

**docker pull <image\_name>**

🡪Add tag to the image.

**docker tag <image\_name>:latest username:<tag\_name>**

🡪Push the image to the docker.

**docker push username:<tag\_name>**

🡪To see the docker images.

**docker images**

🡪To see the running containers.

**docker ps**

🡪To see all the running and not running containers.

**docker ps -a**

🡪To create container.

**docker run -itd –name <container\_name> -p port\_number <image\_id>**

🡪To start the container.

**docker start <container\_id>**

1. **Nginx:**

🡪Pull the image from the online using:

**docker pull nginx**

🡪To know the complete information of the image.

**docker inspect <image\_id>**

🡪Add the tag to the image.

**docker tag nginx:latest deepthi7981/qa:jen**

🡪Push the image to the docker.

**docker push deepthi7981/qa:jen**

🡪Create the container:

**docker run -itd –name ngi -p 80:80 nginx**

🡪To start the container:

**docker start 7653bcd0c582**

🡪To stop the container:

**docker stop 7653bcd0c582**

🡪To remove the container:

**docker rm 7653bcd0c582**

1. **Ubuntu**

🡪Pull the image from the online using:

**docker pull ubuntu**

🡪To know the complete information of the image.

**docker inspect <image\_id>**

🡪Add the tag to the image.

**docker tag ubuntu:latest deepthi7981/dev:ubu**

🡪Push the image to the docker.

**docker push deepthi7981/dev:ubu**

🡪Create the container:

**docker run -it –name my--ubuntu -p 8080:80 a04dc4851cbc**

**apt update && apt install -y apache2**

**service apache2 start**

**ctrl+p+q**

**docker ps**

🡪To start the container:

**docker start 5391259161c5**

🡪To stop the container:

**docker stop 5391259161c5**

🡪To remove the container:

**docker rm 5391259161c5**

🡪To run the ubuntu again:

**docker exec -it my-ubuntu bash**

**service apache2 start**

**ctrl+p,ctrl+q**

**docker ps**

1. **Httpd:**

🡪Pull the image from the online using:

**docker pull httpd**

🡪To know the complete information of the image.

**docker inspect <image\_id>**

🡪Add the tag to the image.

**docker tag httpd:latest deepthi7981/dev:http**

🡪Push the image to the docker.

**docker push deepthi7981/dev:http**

🡪Create the container:

**docker run -itd –name httpd1 -p 40:80 83d938198316**

🡪To start the container:

**docker start ccdd7af18da7**

🡪To stop the container:

**docker stop ccdd7af18da7**

🡪To remove the container:

**docker rm ccdd7af18da7**

1. **Tomcat:**

🡪Pull the image from the online using:

**docker pull tomcat:8.0.52**

🡪To know the complete information of the image.

**docker inspect b4b762737ed4**

🡪Add the tag to the image.

**docker tag tomcat:8.0.52 deepthi7981/dev:tom**

🡪Push the image to the docker.

**docker push deepthi7981/dev:tom**

🡪Create the container:

**docker run -itd –name tomcat\_server -p 8080:8080 b4b762737ed4**

🡪To start the container:

**docker start 3066eec70ab3**

🡪To stop the container:

**docker stop 3066eec70ab3**

🡪To remove the container:

**docker rm 3066eec70ab3**

1. **Jenkins:**

🡪Pull the image from the online using:

**docker pull Jenkins/jenkins**

🡪To know the complete information of the image.

**docker inspect eff2362b2d14**

🡪Add the tag to the image.

**docker tag Jenkins/Jenkins:latest deepthi7981/dev:jenkins**

🡪Push the image to the docker.

**docker push deepthi7981/dev:jenkins**

🡪Create the container:

**docker run -itd –name jenkins-server -p 8080:8080 -p 50000:50000 jenkins/jenkins**

🡪To start the container:

**docker start 0d0d7a399d62**

🡪To stop the container:

**docker stop 0d0d7a399d62**

🡪To remove the container:

**docker rm 0d0d7a399d62**

1. **Python-Flask:**

🡪Pull the image from the online using:

**docker pull python**

🡪To know the complete information of the image.

**docker inspect 3123f56531af**

🡪Add the tag to the image.

**docker tag python-flask-app:latest deepthi7981/dev:py**

🡪Push the image to the docker.

**docker push deepthi7981/dev:py**

🡪Create a directory:

**mkdir docker\_python**

🡪Change the directory:

**cd docker\_python**

🡪Here we have create 3 files:

**nano app.py**

**nano requirements.txt**

**nano dockerfile**

🡪Build the docker image:

**docker build -t docker\_python**

🡪Create the container:

**docker run -d -p 5000:5000 –name python-flask python**

🡪To start the container:

**docker start fe51dfedc40c**

🡪To stop the container:

**docker stop fe51dfedc40c**

🡪To remove the container:

**docker rm fe51dfedc40c**

1. **Mysql:**

🡪Pull the image from the online using:

**docker pull mysql**

🡪To know the complete information of the image.

**docker inspect fa262c3a6564**

🡪Add the tag to the image.

**docker tag mysql:latest deepthi7981/dev:sql**

🡪Push the image to the docker.

**docker push deepthi7981/dev:sql**

🡪Create the container:

**docker run -d –name mysql-server MYSQL\_ROOT\_PASSWORD=deepthi08 -p 3306:3306 fa262c3a6564**

🡪To see the running containers:

**docker ps**

🡪To start the container:

**docker start d0f4b5a1b940**

🡪To stop the container:

**docker stop d0f4b5a1b940**

🡪To remove the container:

**docker rm d0f4b5a1b940**

🡪To run the mysql:

**docker exec -it mysql-server mysql -u root -p**

🡪To see the logs:

**docker logs mysql\_server**

1. **MongoDB:**

🡪Pull the image from the online using:

**docker pull mongo**

🡪To know the complete information of the image.

**docker inspect b81a621037ef**

🡪Add the tag to the image.

**docker tag mongo:latest deepthi7981/qa:mongo**

🡪Push the image to the docker.

**docker push deepthi7981/qa:mongo**

🡪Create the container:

**docker run -d –name mongodb-container -p 27017:27017 b81a621037ef**

🡪To see the running containers:

**docker ps**

🡪To start the container:

**docker start d0f4b5a1b940**

🡪To stop the container:

**docker stop d0f4b5a1b940**

🡪To remove the container:

**docker rm d0f4b5a1b940**

🡪To run the mongodb:

**docker exec -it mongodb-container mongosh**