**SRE TRAINING (DAY 11) - CONTAINERIZATION**

**CONTAINERIZATION**

Containerization is a technology that allows you to package an application along with

all its dependencies (such as libraries, configuration files, and environment variables)

into a single unit called a container. This ensures that the application runs

consistently across different computing environments, whether it’s on a developer's

local machine, on a testing server, or in production.

**DOCKER**

Docker is an open-source platform that automates the deployment, scaling, and

management of applications using containers.

**Key Docker Components:**

1. **Docker Engine:** The core part of Docker that runs on the host machine and

allows you to build, run, and manage containers.

2. **Docker Images:** Read-only templates that define the container’s contents.

3. **Docker Containers:** Running instances of Docker images, providing an

isolated environment for applications.

4. **Dockerfile:** A script containing instructions to build a Docker image.



5. **Docker Hub:** A cloud-based registry where Docker images can be stored and

shared publicly or privately.6. **Docker Compose:** A tool for defining and running multi-container Docker

applications using a docker-compose.yml file.

7. **Docker CLI -** command-line interface used to interact with the Docker Engine.

It allows you to build, run, manage, and monitor Docker containers, images,

networks, and volumes using simple commands.

**Basic Docker Workflow**

1. Write a **Dockerfile.**

2. **Build** an image from a Dockerfile.

3. **Run** a container from the image.

4. **Manage** images, containers, volumes, and networks.

5. **Push/Pull** images to/from Docker Hub.

**Common Docker CLI Commands**

❖ **docker --version** Displays the installed Docker version**.**

❖ **docker images** Lists all Docker images on the local machine.

❖ **docker build -t <image-name> .** Builds an image from a Dockerfile in

the current directory.

❖ **docker pull <image-name>** Pulls an image from Docker Hub.

❖ **docker run -d -p 8080:80 <image-name>** Runs a container in

detached mode with port mapping.

❖ **docker ps** List running containers❖ **docker stop <container-id>** Stops a running container.

❖ **docker start <container-id>** Starts a stopped container.

❖ **docker push <username>/<image-name>** Pushes an image to

Docker Hub.

❖ **docker system prune** Cleans up unused containers, images,

networks, and volumes.

❖ **docker-compose up -d** Starts all services defined in

docker-compose.yml in detached mode.

**AUTHENTICATION** - **docker login**

The **docker login** command is used to authenticate your Docker CLI session with Docker

Hub or any Docker registry. You will need to connect your device with an access token for

the first time.

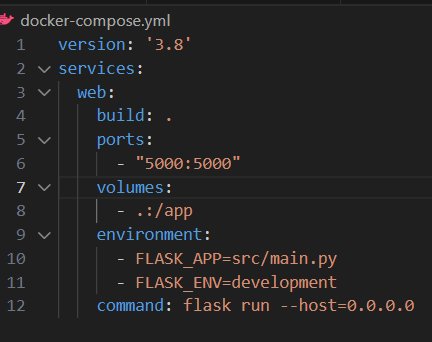
🐳 **DOCKER COMPOSE**

**Docker Compose** is a tool that allows you to define and manage **multi-container Docker**

**applications** using a simple YAML configuration file. With Docker Compose, you can define

services, networks, and volumes in a single file (docker-compose.yml) and manage them

Easily.



**DOCKER IMAGE TAGS**

In Docker, **tags** are used to identify and differentiate images. Tags typically represent

different versions, environments, or configurations of an image.

**DOCKER BUILDS**

**Docker build** is the process of creating a Docker image from a **Dockerfile** and application

source code. The build process packages the application and its dependencies into an

image that can run on any machine with Docker installed. **Build once, run anywhere**.