Docker Theory and Commands

1. Why Docker is Needed:
Docker helps developers build, package, and deploy applications consistently across environments.
It solves the 'it works on my machine' problem by packaging code, dependencies, and runtime together.

2. Benefits of Docker:
- Portability: Runs consistently across development, testing, and production.
- Isolation: Containers do not interfere with each other.
- Scalability: Easily scale applications by running multiple containers.

3. Docker vs Virtual Machine:
Docker containers share the host OS and are lightweight and fast.
Virtual Machines run a full OS and are heavier and slower.

4. Docker Engine:
- Docker Daemon: Manages containers and images.
- REST API: Communication between CLI and daemon.
- Docker CLI: User interface to run Docker commands.

5. Docker Image:
A read-only blueprint containing application code, dependencies, base image, and metadata.

6. Docker Image Lifecycle:
Build → Store → Push → Pull → Run

7. Dockerfile:
Defines instructions to build a Docker image using commands like FROM, RUN, COPY, CMD, EXPOSE.

8. Docker Container:
A running instance of a Docker image. Lightweight, isolated, and portable.

9. Docker Registry:
Stores Docker images (Docker Hub, AWS ECR, GCP GCR, Azure ACR).

Docker Commands Workflow

1. docker login
2. docker build -t mlops_docker .
3. docker run -p 5000:5000 mlops_docker (Run locally)
4. docker tag mlops_docker:latest nithin1123/mlops_docker:latest
5. docker push nithin1123/mlops_docker:latest
6. docker pull nithin1123/mlops_docker:latest
7. docker run -p 5000:5000 nithin1123/mlops_docker:latest (Run from remote)
8. docker images