Day 10

Name: Siddarth S
Date: 08/08/2024

1. Actions of the Following Git Commands:

1. **git remote add origin "https://github.com/siddarthhh/demo1.git"**:
    - This command sets the remote repository for your local Git repository. It informs Git where to push your commits. The URL should point to the repository you wish to connect to.
2. **git pull origin master**:
    - This command fetches and merges changes from the master branch of the remote repository named origin into your current branch.
3. **git push origin dev**:
    - This command pushes your local dev branch to the remote repository named origin.

2. Functions of the Following Docker Objects and Key Components:

- **Dockerd**:
    - Dockerd is the Docker daemon that listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. It can also communicate with other daemons to manage Docker services.
- **Dockerfile**:
    - A Dockerfile is a script containing a series of instructions on how to build a Docker image. Each instruction creates a layer in the image. Common instructions include FROM, RUN, COPY, and CMD.
- **docker-compose.yaml**:
    - The docker-compose.yaml file defines services, networks, and volumes for a multi-container Docker application. It allows you to manage multiple containers as a single service, define how the containers interact, and configure each container's environment.
- **Docker Registries**:
    - Docker registries are repositories for Docker images. They allow you to store and distribute Docker images. Docker Hub is the default public registry, but you can also use private registries.
- **DockerHost**:
    - DockerHost refers to the machine where Docker is installed and running. It can be a local machine, a virtual machine, or a cloud server. It hosts Docker daemons and Docker containers.

3. What is Isolation in Docker Containers?

Isolation in Docker containers refers to the encapsulation of applications and their dependencies within containers, which run as isolated processes in user space on the host operating system. This isolation ensures that containers do not interfere with each other or the host system, providing several benefits:

- **Process Isolation**: Each container runs as a separate process with its own file system, network interface, and process space, isolated from other containers and the host system.
- **Resource Control**: Docker uses control groups (cgroups) to limit and prioritize resources (CPU, memory, disk I/O) for each container.
- **Namespace Isolation**: Docker uses namespaces to provide containers with their own view of the system, including process IDs (PID namespace), network (net namespace), user IDs (user namespace), and file systems (mount namespace).