Name: Sownthari R P
Date: 07.08.2024

1.Mention the actions of following comments:

- **git remote add origin "http://github/a.git"**: This command adds a new remote repository with the name "origin" and the URL "http://github/a.git". The remote repository is typically where your code is stored, and "origin" is the default name for a remote.
- **git pull origin master**: This command fetches and merges changes from the "master" branch of the remote repository named "origin" into the current branch. It combines the git fetch and git merge commands.
- **git push origin dev**: This command pushes the commits from the local "dev" branch to the remote repository named "origin" and updates the "dev" branch there.

2. What are the functions of following Docker objects and key components:

- **Dockerd**: Dockerd is the Docker daemon, which runs on a host machine. It listens for Docker API requests and manages Docker objects like images, containers, networks, and volumes. It handles the creation, running, and stopping of containers.
- **Dockerfile**: A Dockerfile is a script containing a series of instructions on how to build a Docker image. It includes commands to set up the environment, copy files, install dependencies, and define default behaviors like running a specific command when a container starts.
- **docker-compose.yaml**: This file is used with Docker Compose to define and run multi-container Docker applications. It specifies the services, networks, and volumes required for the application, allowing you to manage the entire application stack with a single command (docker-compose up).
- **Docker Registries**: Docker registries are repositories where Docker images are stored. The most common registry is Docker Hub, but there are others like

Amazon ECR, Google Container Registry, and private registries. They allow you to share, store, and distribute Docker images.

- **DockerHost**: The DockerHost refers to the physical or virtual machine on which Docker is installed and running. It can be a local machine, a remote server, or a cloud-based environment where Docker manages containers.

3. What's the isolation in Docker container?

Isolation in Docker containers refers to the separation of applications and their dependencies into independent units. Each container runs in its own isolated environment with its own file system, network interfaces, and process space. Key aspects of Docker's isolation include:

- **Filesystem Isolation**: Each container has its own filesystem, provided by Docker images, ensuring that changes in one container do not affect others.
- **Process Isolation**: Containers run their own processes independently. The processes inside a container are isolated from those running on the host and in other containers.
- **Network Isolation**: Containers can have their own network interfaces and IP addresses, allowing for isolated network environments. Docker provides bridge networks, overlay networks, and other options for container networking.
- **Resource Limiting**: Docker can limit the amount of CPU, memory, and other resources that containers can use, preventing a single container from consuming all the host’s resources.