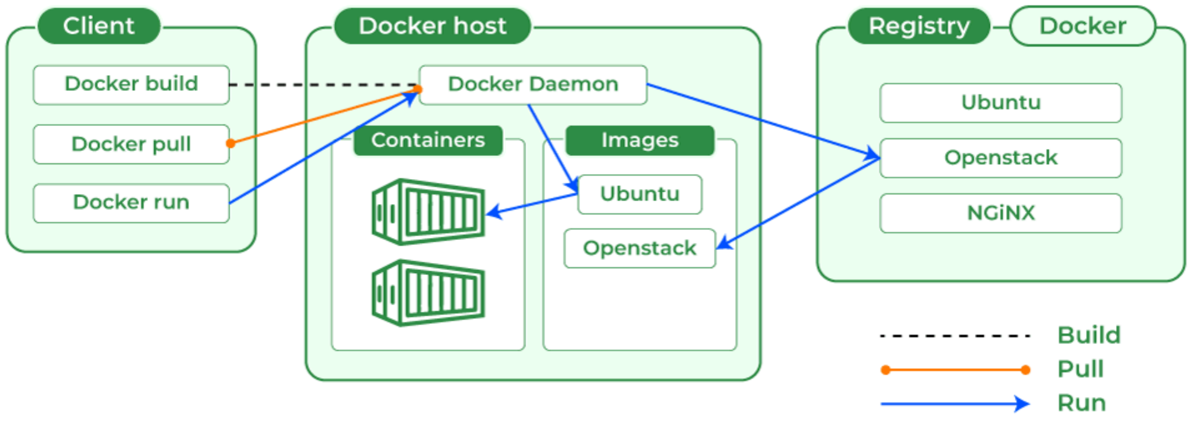
DevOPS training:

DOCKER:

Docker Architecture-

In order to administer containerized applications, the Docker client interfaces with the Docker daemon (server) through a client-server architecture. Containers are small, isolated runtime environments that are created using read-only templates called Docker images. These images are kept in Docker registries, and communication between containers is facilitated via Docker networks. Data is stored on volumes, and configuration files are used by Docker Compose to streamline the management of multi-container applications. Applications may be deployed consistently and effectively across many environments thanks to this configuration.



Importance of Docker:

Docker is essential because it packages software into lightweight, homogeneous containers that function consistently in a variety of contexts, making application deployment and management easier. This guarantees that programs operate consistently across all systems—from development to production and expedites the process of updating and growing programs while lowering compatibility problems and conflicts.

Docker engine:
The background program that oversees Docker containers on a system is called Docker Daemon. In addition to handling container lifecycle tasks like constructing, executing, and maintaining containers, it also responds to Docker API requests. To pull and push images, the daemon also communicates with the Docker registry.

Docker CLI (docker): To communicate with the Docker daemon, utilize the Docker Command Line Interface (CLI). You can use it to execute instructions for managing volumes, networks, pictures, and containers. The CLI includes commands such as docker run, docker ps, and docker build, among others.

Docker API: This is the REST API that the Docker daemon makes available, enabling communication between Docker and other programs and utilities. It offers interfaces for managing and building containers.

Images- Templates of project
Container: Running instance of Image
Base image is the parent image

Docker components:
1- Image
2- Container
3- Networking
4- Storage

```
                build                              run
Dockerfile——-------------->Docker image——----------->Docker container
```

Docker command notes:

docker run-it ubantu /bin/bash
ctrl+pq :-exit a container without stopping it
docker ps :- to check all running containers
docker ps -a :- to check all runningg and exited/stoped containers
docker attach <container name or container ID> :- connect to a running container
docker stop <container name or container ID> :- Stop a container
docker rmi -f <image name or image ID> :- kill/permanentley remove a image
docker run -d <docker image> :- pulls and run containers in detached mode
docker run -d -p <VM port>:<container port> <container image> :- Maps the VM port with the container port
docker build -t <tag(custom name)> . :-build a image from the dockerfile. (.) means docker file present in current working directory

```
        ex:- dockerfile=dock
                docker build -t newdocker .
```

docker login -u <user_name of docker hub> -p <password of docker hub>
docker tag <imageid> <dockerhub_userid>/<docker_hub_reprositery_name> :-to tag your docker image with docker hub
docker push <docker hub user name>/<docker hub repository> :- to push the docker image to the hub

docker pull <docker hub user name>/<docker hub repository name> :- to pull the docker image
Listing container- docker container ls
Making an container to run in same shell- docker container exec -it imageid bash
Removing container- docker rm container_id
Deleting stopped container- docker container prune -f
Knowing layers of image: docker inspect image_id
Go inside the container- docker run -it image_id /bin/bash
Kill container- docker kill container_id
Docker pull from docker hub:
Docker pull priyamchowdhury23/dockerpy:latest

Docker push to Docker hub:
docker login -u priyamchowdhury23
pass-*****
docker tag imagename username/reponame
docker push username/reponame

Docker volume:
docker volume create myvol1
docker volume ls (for listing the volumes in docker)
docker run -it -v myvol1:/home ubuntu

Binding mount:
docker run -v /home/priyam/Desktop:/home -it image_id

Logs of container:
Docker log my_container
Seeing logs since last 10 minutes- docker logs --since 10m <container_id>