**What is docker?**

Docker is an open-source project that automates the deployment of software applications inside containers by providing an additional layer of abstraction and automation of OS-level virtualization on Linux.

**Docker Images**

* Images are read only templates used to create containers.
* Images are created with the docker build command,either by us or by other docker users.
* Images are composed of layers of other images.
* Images are stored in a Docker registry.

**Docker Container**

A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

**Registries and Repositories**

• A registry is where we store our images.

• You can host your own registry, or you can use Docker’s public registry which is called DockerHub.

• Inside a registry, images are stored in repositories.

• Docker repository is a collection of different docker images with the same name, that have different tags,each tag usually represents a different version of the image

**Docker Port Mapping**

Port mapping is used to access the services running inside a Docker container. We open a host port to give us access to a corresponding open port inside the Docker container. Then, all the requests that are made to the host port can be redirected into the Docker container

**Docker Log**

The docker logs command shows information logged by a running container. The docker service logs command shows information logged by all containers participating in a service. The information that is logged and the format of the log depends almost entirely on the container's endpoint command.

**Build Docker Image**

1.committing changes made in a container

* Spin up a container from a base image.
* Install Git package in the container.
* Commit changes made in the container.

**Docker Commit**

Docker commit command would save the changes we made to the Docker container’s file system to a new image.

docker commit container\_ID repository\_name:tag

2. Writing a Dockerfile

* A Dockerfile is a text document that contains all the instructions users provide to assemble an image.
* Each instruction will create a new image layer to the image.
* Instructions specify what to do when building the image.

**Docker container link**

Docker also has a linking system that allows you to link multiple containers together and send connection information from one to another. When containers are linked, information about a source container can be sent to a recipient container.

**Benefits of Docker Container Links**

* The main use for docker container links is when we build an application with a microservice architecture, we are able to run many independent components in different containers.
* Docker creates a secure tunnel between the containers that doesn’t need to expose any ports externally on the container.

**Docker Compose**

* Docker compose is a very handy tool to quickly get docker environment up and running.
* Docker compose uses yaml files to store the configuration of all the containers, which removes the burden to maintain our scripts for docker orchestration.

**Docker Compose command**

* **docker compose up**- starts up all the containers.
* **docker compose ps** -checks the status of the containers managed by docker compose.
* **docker compose logs**- outputs colored and aggregated logs for the compose-managed containers.
* **docker compose logs** -with dash f option outputs appended log when the log grows.
* **docker compose logs** -with the container name in the end outputs the logs of a specific container.
* **docker compose stop**- stops all the running containers without removing them.
* **docker compose rm** -removes all the containers.
* **docker compose** **build**- rebuilds all the images.

**Docker Network Type**

* Closed Network / None Network
* Bridge Network
* Host Network
* Overlay Network

**Continuous Integration**

* Continuous integration is a software engineering practice in which isolated changes are immediately tested and reported when they are added to a larger code base.
* The goal of Continuous integration is to provide rapid feedback so that if a defect is introduced into the code base, it can be identified and corrected as soon as possible.

**Swarm cluster**

A docker swarm is comprised of a group of physical or virtual machines operating in a cluster. When a machine joins the cluster, it becomes a node in that swarm.

**How Swarm cluster works**

* To deploy your application to a swarm, you submit your service to a manager node.
* The manager node dispatches units of work called tasks to worker nodes.
* Manager nodes also perform the orchestration and cluster management functions required to maintain the desired state of the swarm.
* Worker nodes receive and execute tasks dispatched from manager nodes.
* An agent runs on each worker node and reports on the tasks assigned to it. The worker node notifies the manager node of the current state of its assigned tasks so that the manager can maintain the desired state of each worker

**Docker Stack**

A docker stack is a group of interrelated services that share dependencies, and can be orchestrated and scaled together.

**Docker Swarm commands**

**docker swarm init** – Initialize a swarm. The docker engine targeted by this command becomes a manager in the newly created single-node swarm.

**docker swarm join** – Join a swarm as a Swarm node.

**docker swarm leave** – Leave the swarm.