What is Docker?

Docker is a containerisation tool that simplifies the process of developing, deploying, and running applications by packaging them with all dependencies into a standardized unit called a container. These containers can be easily moved between different environments, making it easier to ensure that the application works consistently across various systems.

Instructions in Docker:

1. FROM:

Docker files must start with a from instruction.

From is used for pulling an image from docker registry.

Eg: FROM ubuntu

2. MAINTAINER:

Maintainer is used to set the author field for images to be generated.

Eg: MAINTAINER rajith<rajithbhargav036@gmail.com>

3. LABEL:

The label instruction adds metadata to an image.

A label is a key value pair .

Eg: LABEL env=dev tag=release ver=1.0

4. RUN:

Executes commands in a new layer on top of the current image and commits the results. Used for installing packages, updating, or making configuration changes.

Eg:- RUN apt-get update \

&& apt-get install git -y \

5. WORKDIR:

Sets the working directory for other instructions. It affects RUN, CMD, ENTRYPOINT, COPY, and ADD instructions.

Eg:- WORKDIR /app

6. COPY:

Copies files or directories from a source on the host to the container's filesystem.

Eg:- COPY index.html /app

7. ADD:

Similar to COPY, but can also fetch files from URLs or extract tarballs.

Eg: ADD https://example.com/myfile.txt /app/

8. EXPOSE:

Informs Docker that the container exposes to specified network ports at runtime.

Eg:- EXPOSE 80

9. CMD:

it will run linux commands. It allows you to specify a command that will be run when the container starts. It can be overridden when running the container.

Eg:- CMD ["ls", "-l"]

10. ENTRYPOINT:

It allows you to specify a command that will be run when the container starts. It cannot be overridden while running the container.

Eg:- ENTRYPOINT ["ls", "-l"]

11. ENV:

In a Dockerfile, the ENV instruction is used to set environment variables within the image. These variables are accessible to any instructions in the Dockerfile and are also available during container runtime.

Eg:- APP_VERSION=1.0

12. ARG:

The ARG instruction in a Dockerfile is used to define build-time variables. These variables can be set to specific values when you build an image using the docker build command. ARG is particularly useful for passing dynamic information to the Dockerfile during the build process.

Eg:- ARG pkg

ENV pkg=${pkg}

13. Volume:

In a Dockerfile, the VOLUME instruction is used to create a mount point for external volumes.
It allows the container to store data outside the container's filesystem, and it is available for use even after the container is stopped or removed.
Eg:- VOLUME /var/dir1

14. USER:
The USER instruction in a Dockerfile is used to set the user or UID (User Identifier) that the image should use when running the container. It helps to define under which user the subsequent instructions (such as CMD, RUN, and ENTRYPOINT) will be executed.
Eg:- USER <user>[:<group>] or USER <UID>[:<GID>]
15. SHELL:
The SHELL instruction in a Dockerfile is used to set the default shell variables for RUN instructions. It allows you to customize the shell environment variables used by the Dockerfile.
Eg:- SHELL ["/bin/bash", "-c"]
What is Docker Volumes, How is it mounted?
Ans:-
Docker volumes are used to persist the data of containers and it is also backup for containers and if a container is crashed or deleted even though we can access the container data.
It can be mounted in 2 ways:
1->During the runtime of a container by using run command with option -v
Ex: docker run -it - - name c1 -v <path> image_name
2-> while writing a dockerfile, we can directly declare in dockerfile
FROM ubuntu
MAINTAINER rajith<rajithbhargav036@gmail.com>
WORKDIR /app
VOLUME /app

CMD ["/bin/bash"]

How to use volumes of an already mounted container to another?

Ans:- docker run -it --name container2 --volumes-from container1 image-name

What is the difference between volumes and bind mounts in docker?

Ans:-

Volumes are used to store a backup data of a container and it has a default path in the host system.

Bind mounts are also one type of docker volumes, which are stored in user defined paths in the host system.

What is docker compose?

Ans:-

Docker-compose is a tool which is used to create 2 containers at a time with a single command by using docker-compose file.

How to create two containers at a time?

Ans:-

By using docker-compose.

docker-compose up -d

What is docker swarm?

Ans:- docker swarm is a tool that helps to manage clusters of servers, by using this tool we can maintain high availability of containers.

Docker Swarm	Kubernetes
Easier installation	Complex installation
More lightweight and easier to use but limited functionality.	More complex with a high learning curve but more powerful.
Manual scaling	Supports auto-scaling
Needs third party tools for monitoring	Built in monitoring

Auto load balancing	Manual setup of load balancer
Integrates docker CLI	Need for a separate CLI tool

How to create multiple containers in multiple servers?

Ans:-

With the help of docker-compose.yml

How to write docker compose files in docker swarm?

Ans:-

Docker-compose is a tool which is used to create 2 containers at a time with a single command by using docker-compose file.

```
version: "3.6"
services:
  ng:
    image: nginx
    ports:
    - "8086:80"
  ht:
    image: httpd
    ports:
    - "8085:80"
```

docker-compose up -d

Docker-compose using docker swarm

```
version: "3.6"
services:
  ng:
    image: nginx
    deploy:
      replicas: 3
    ports:
    - "8086:80"
  ht:
    image: httpd
    deploy:
      replicas: 3
    ports:
    - "8085:80"
```

How to run docker swarm?

Ans:-

Docker Swarm is a container orchestration tool for managing multiple Docker containers across multiple machines. It allows you to easily deploy, scale, and manage containerized applications. Swarm enables high availability and load balancing for distributed applications.

docker stack deploy -c docker-compose.yml myapp

To scale up

docker service scale myapp=5