Docker

Author: Amol Shete

What is Docker ?

Docker is a platform that allows developers to create, deploy, and run applications in a containerized environment. A container is a lightweight and standalone executable package that includes everything needed to run the application, such as the code, runtime, system tools, libraries, and settings. Containers provide an isolated and consistent environment, ensuring that the application runs the same way on different systems and infrastructure.

Docker consists of three main components: the Docker engine, the Docker hub, and the Docker CLI. The Docker engine is the core component that runs and manages the containers. The Docker hub is a cloud-based service that allows users to store and share container images with others. The Docker CLI is a command-line interface that allows users to interact with the Docker engine and manage containers.

What can I use Docker for?

- **Fast, consistent delivery of your applications**
- **Responsive deployment and scaling**
- **Running more workloads on the same hardware**

Docker Concepts -

- Docker Image

A Docker image is a packaged and portable software bundle that includes an application's code, runtime environment, libraries, and system tools. It is used to create and run containers, which are isolated and lightweight virtual environments that can be easily moved between different systems and infrastructure. Docker images can be easily shared, downloaded, and deployed, which makes it easier to build, test, and deploy applications in a consistent and reproducible way.

- Docker Container

A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.

By default, a container is relatively well isolated from other containers and its host machine. You can control how isolated a container's network, storage, or other underlying subsystems are from other containers or from the host machine.

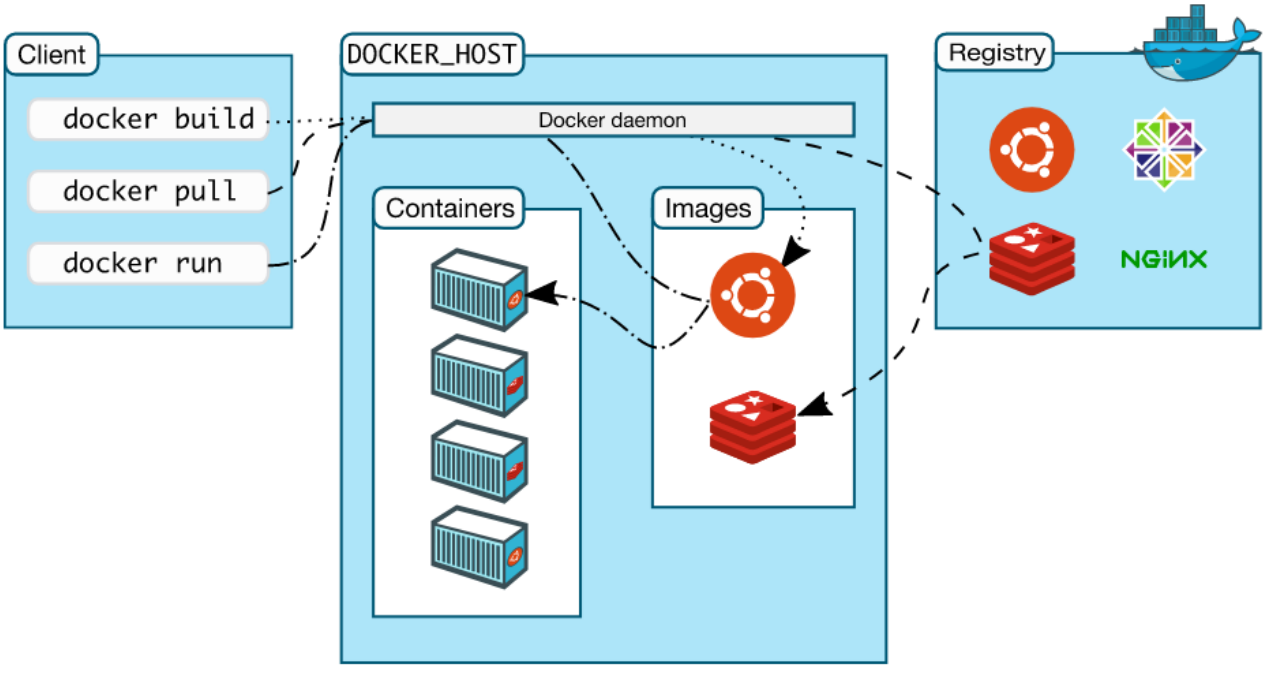
A container is defined by its image as well as any configuration options you provide to it when you create or start it. When a container is removed, any changes to its state that are not stored in persistent storage disappear.

- Docker registries

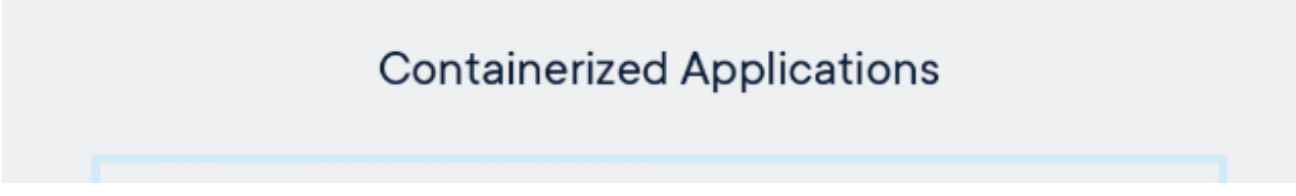
A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

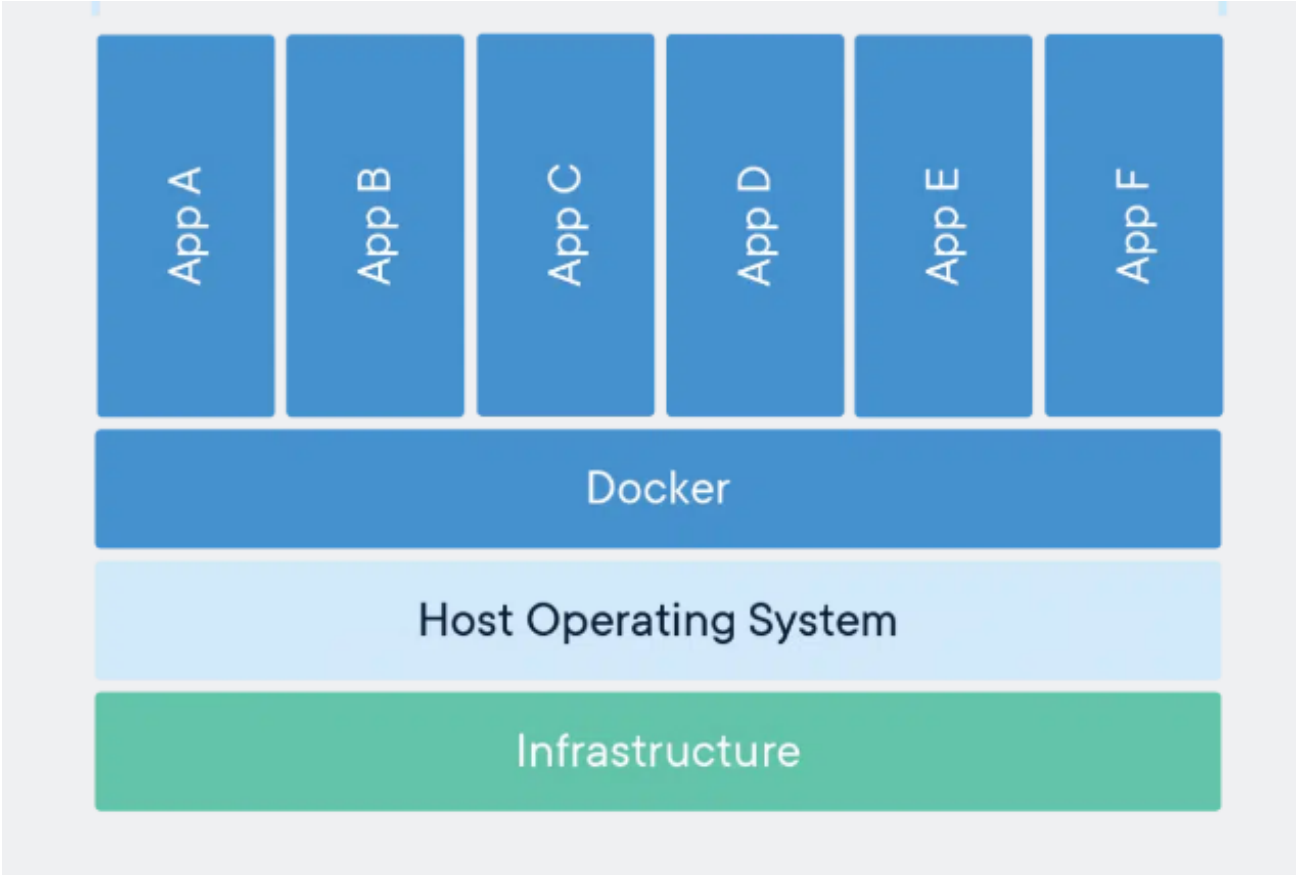
When you use the docker pull or docker run commands, the required images are pulled from your configured registry. When you use the docker push command, your image is pushed to your configured registry.

Docker Architecture



Container runs on top of the Docker engine. Docker engine is nothing but a software which is available for most of the operating system. So when we have to run isolated application there is no need to install guest operating system like VM solution. We can simply run the app in container form on top of the docker engine.





Installation of docker:

We can follow the official documents to install docker on required OS. Following is the link for the same : "https://docs.docker.com/engine/install/" .

Docker Commands :

Here are some commonly used Docker commands with a one-line explanation of their purpose:

docker run	Run a container from a Docker image
docker build	Build a Docker image from a Dockerfile
docker ps	List running containers

docker images	List Docker images on the system
docker stop	Stop a running container
docker start	Start a stopped container
docker rm	Remove a container
docker rmi	Remove a Docker image
docker exec	Execute a command in a running container
docker logs	View the logs of a running container
docker pull	Pull a Docker image from a registry
docker push	Push a Docker image to a registry
docker-compose up	Start a group of containers defined in a Docker Compose file
docker-compose down	Stop and remove containers defined in a Docker Compose file

Note that these commands are just a small subset of the many available Docker commands, and they can be used with various options and arguments to achieve different results.

Below are few examples :

- To download a Docker image:

  docker pull <image_name>

- To list all the running Docker containers:

  docker ps

- To stop a running Docker container:

  docker stop <container_id>

- To remove a Docker container:

  docker rm <container_id>

- To build a Docker image from a Dockerfile:

docker build -t <image_name> <path_to_Dockerfile>

- To start a new Docker container from an image with detached mode:

  docker run -d <image_name>

- To execute a command inside a running Docker container:

  docker exec <container_id> <command>

- To view the logs of a running Docker container:

  docker logs <container_id>

- To inspect a Docker image or container:

  docker inspect <image_or_container_id>

- To push a Docker image to a registry:

  docker push <image_name>

Docker Volume:

A Docker container volume is a mechanism for persisting and sharing data between a Docker container and the host system or between multiple containers. It provides a way to store and manage data separately from the container itself.

When a Docker container is started, it runs in an isolated environment with its own file system. Any data that is created or modified inside the container is typically lost when the container is stopped or deleted. To preserve this data, you can use a Docker volume to map a directory on the host system or another container to a directory in the container.

Using a volume has several benefits:-

- It allows data to be shared between containers.
- It enables data to persist even if the container is deleted or recreated.
- It provides a way to manage data separately from the container, making it easier to back up or migrate.

To create a volume, you can use the docker volume create command. For example:

```
docker volume create my_volume
```

You can then use the volume in a container by specifying it with the -v option when you start the container. For example:

```
docker run -v my_volume:/app/data my_image
```

This creates a container that uses the my_volume volume and mounts it at the /app/data directory inside the container. Any data that is written to this directory will be persisted in the my_volume volume and can be accessed by other containers that use the same volume.

Practical example for docker volume could be, lets say we want to run the mysql container. Now we can create first docker volume by command:

docker volume create mydbdata

And then we can create mount point while running the docker container:

docker run --name myapp -d -e MYSQL_ROOT_PASSWORD=secretpass -p 3306:3306 -v

Thank you!