Introduction to Docker and Containers

Dr. JASMEET SINGH
ASSISTANT PROFESSOR,
CSED, TIET

What is a Docker?

- Docker is a software platform (a toolkit) that allows us to build, test, and deploy applications quickly.
- Using Docker, we can quickly deploy and scale applications into any environment and know our code will run.
- Docker is an open source **containerization** platform i.e. Docker packages software into standardized units called **containers** that have everything the software needs to run including libraries, system tools, code, and runtime.
- It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

Docker Terminology

- Some of the terminology we will encounter when using Docker include:

1. Docker Images
2. Docker Containers
3. Docker File
4. Docker Hub
5. Docker Daemon
6. Docker Registry

Docker Images

- A Docker image is a file used to execute code in a Docker container. **Docker images act as a set of instructions to build a Docker container, like a template.**
- Docker images also act as the starting point when using Docker.
- It's possible to build a Docker image from scratch, but most developers pull them down from common repositories.
- Multiple Docker images can be created from a single base image, and they'll share the commonalities of their stack.
- Docker images are made up of *layers*, and each layer corresponds to a version of the image. Whenever a developer makes changes to the image, a new top layer is created, and this top layer replaces the previous top layer as the current version of the image. Previous layers are saved for rollbacks or to be re-used in other projects.

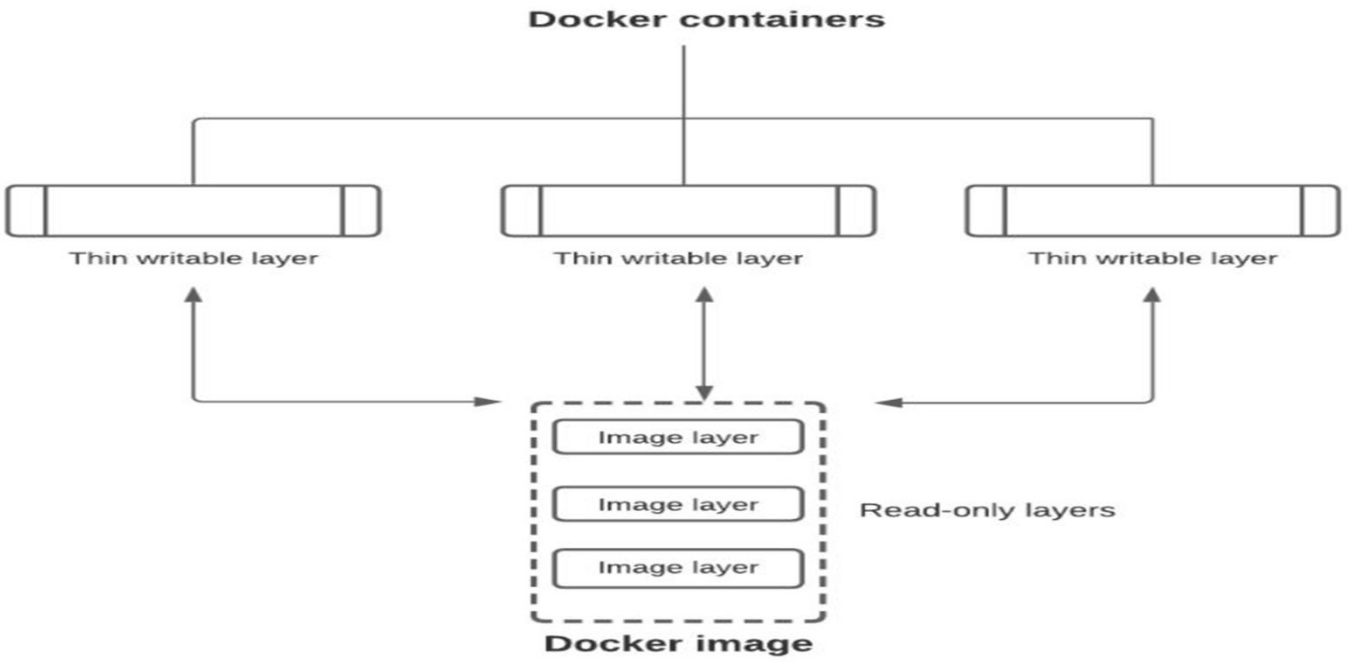
Docker Containers

- Docker containers are the live, running instances of Docker images.
- While Docker images are read-only files, containers are live and executable content. Users can interact with them, and administrators can adjust their settings and conditions using docker commands.
- Developers can create containers without Docker, but the platform makes it easier, simpler, and safer to build, deploy and manage containers.
- Docker is essentially a toolkit that enables developers to build, deploy, run, update, and stop containers using simple commands and work-saving automation through a single API.

Docker Containers (Contd….)

- Each time a container is created from a Docker image, yet another new layer called the container layer is created.
- Changes made to the container—such as the addition or deletion of files—are saved to the container layer only and exist only while the container is running.
- Multiple live container instances can run from just a single base image, and when they do so, they leverage a common stack.

Docker Images vs. Docker Containers



Docker Images vs. Docker Containers

Docker Image	Docker Container
It's a container blueprint	It's an image instance
It's immutable	It's writable
It can exist without a container	A container must run an image to exist
Does not need computing resources to operate	Need computing resources to run—containers run as Docker virtual machines
It can be shared via a public or private registry platform	No need to share an already running entity
Created only once	Multiple containers can be created from the same image

Docker File

- *DockerFile* automates the process of Docker image creation.
- It's essentially a list of command-line interface (CLI) instructions that Docker Engine will run in order to assemble the image.
- These instructions are given in GO Programming language.
- Every Docker container starts with a simple text file containing instructions for how to build the Docker container image.

Docker Hub

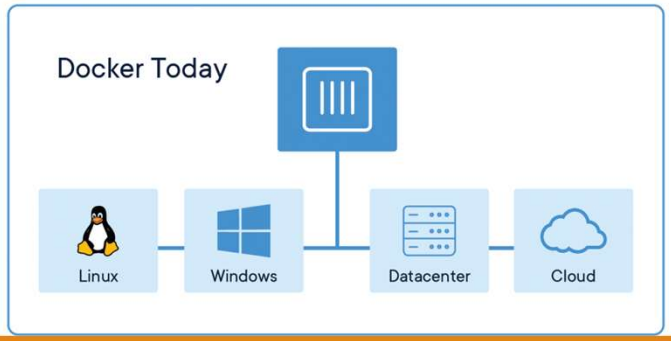
- *Docker Hub* is the public repository of Docker images that calls itself the "world's largest library and community for container images."
- It holds over 100,000 container images sourced from commercial software vendors, open-source projects, and individual developers.
- It includes images that have been produced by Docker, Inc., certified images belonging to the Docker Trusted Registry, and many thousands of other images.

Docker Daemon & Docker Registry

- **Docker daemon** is a service running on our operating system, such as Microsoft Windows or Apple MacOS or iOS.
- This service creates and manages our Docker images for you using the commands from the client, acting as the control center of your Docker implementation.
- A **Docker registry** is a scalable open-source storage and distribution system for docker images. The registry enables you to track image versions in repositories, using tagging for identification.

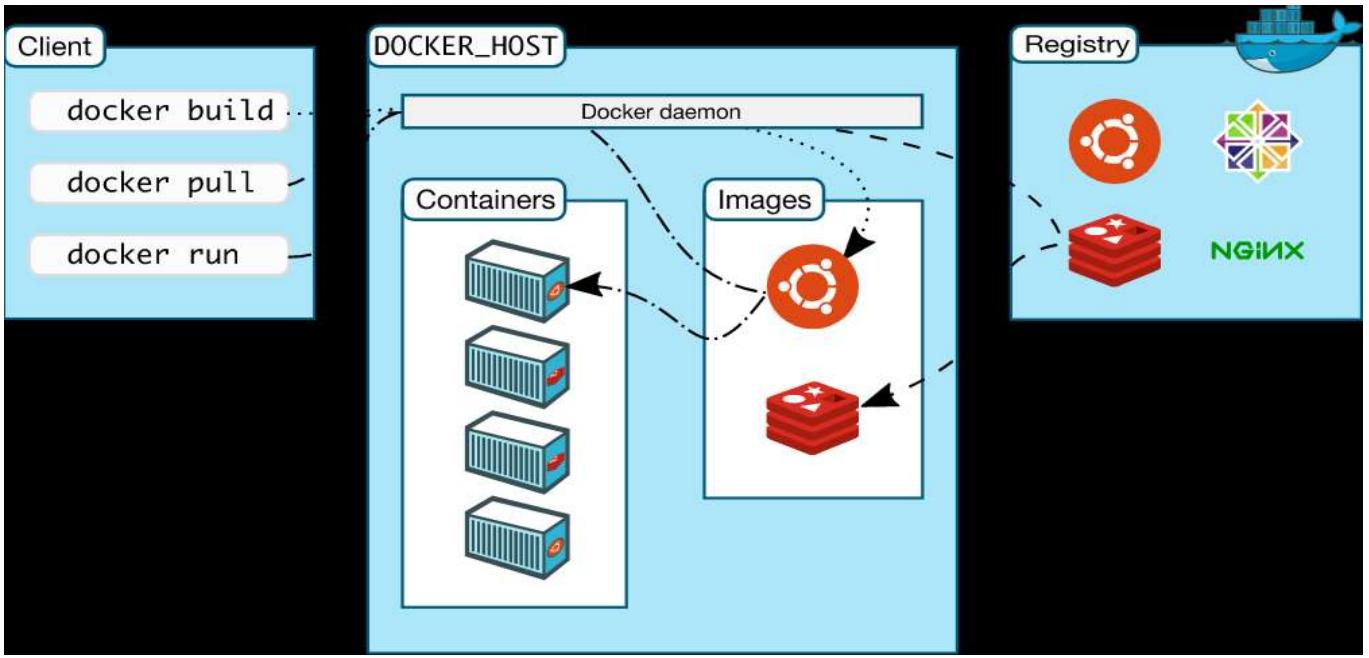
History of Docker Containers

- Docker container technology was launched in 2013 as an open source Docker Engine for Linux operating ysrems.
- Docker's technology is unique because it focuses on the requirements of developers and systems operators to separate application dependencies from infrastructure.
- Success in the Linux world drove a partnership with Microsoft that brought Docker containers and its functionality to Windows Server.
- **Docker Containers Are Everywhere: Linux, Windows, Data center, Cloud, Serverless, etc.**



Docker Architecture

- Docker uses a client server architecture.



The Docker *client* talks to the Docker *daemon*, which does the heavy lifting of building, running, and distributing your Docker containers.
The Docker client and daemon *can* run on the same system, or we can connect a Docker client to a remote Docker daemon.

Docker Architecture

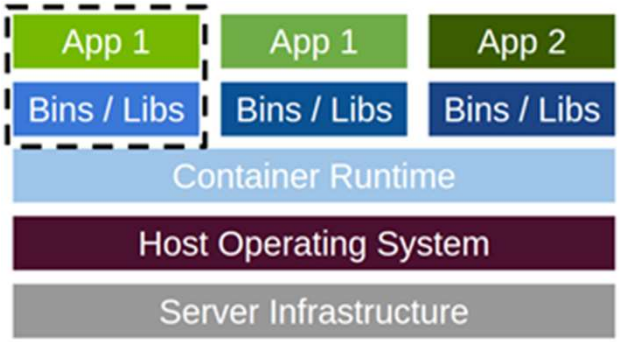
- **The Docker daemon**
  - The Docker daemon (**dockerd**) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.
  - A daemon can also communicate with other daemons to manage Docker services.
- **The Docker Client**
  - The Docker client (**docker**) is the primary way that many Docker users interact with Docker.
  - When we use commands such as docker run, the client sends these commands to dockerd, which carries them out.
  - The docker command uses the Docker API. The Docker client can communicate with more than one daemon.
- **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.

Why use Docker Containers?

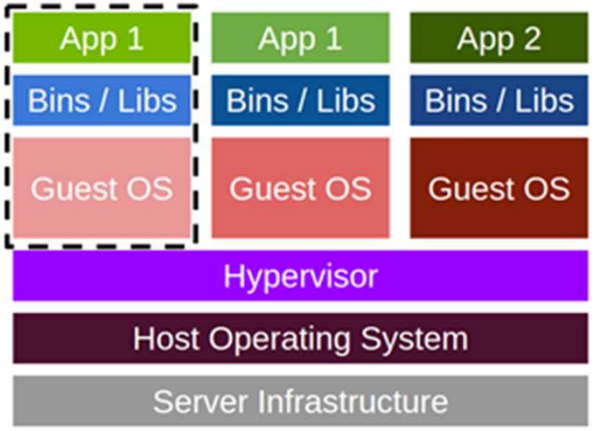
- Using Docker lets you ship code faster, standardize application operations, seamlessly move code, and save money by improving resource utilization.
- **SHIP MORE SOFTWARE FASTER**
  - Docker users on average ship software 7x more frequently than non-Docker users.
  - Docker enables you to ship isolated services as often as needed.
- **STANDARDIZE OPERATIONS**
  - Small containerized applications make it easy to deploy, identify issues, and roll back for remediation.
- **SEAMLESSLY MOVE**
  - Docker-based applications can be seamlessly moved from local development machines to production deployments on Servers
- **SAVE MONEY**
  - Docker containers make it easier to run more code on each server, improving our utilization and saving money.

Docker Containers vs. Virtual Machines (VMs)

- Containers and virtual machines have similar resource isolation and allocation benefits, but function differently because containers virtualize the operating system instead of hardware. Containers are more portable and efficient.

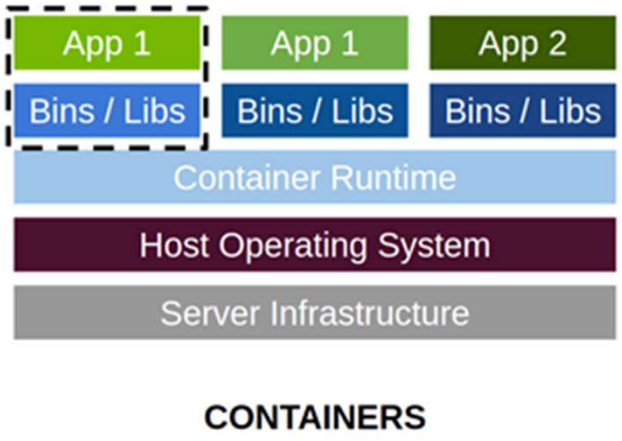


CONTAINERS



VIRTUAL MACHINES

Docker Containers vs. Virtual Machines (VMs)



CONTAINERS

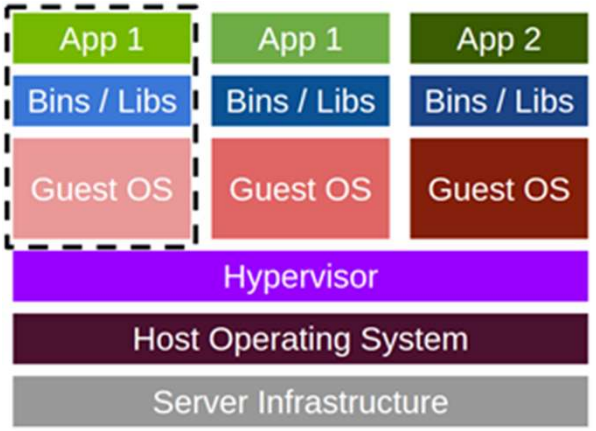
Containers are an abstraction at the app layer that packages code and dependencies together.

Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space.

Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.

Docker Containers vs. Virtual Machines (VMs)

- Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers.
- The hypervisor allows multiple VMs to run on a single machine.
- Each VM includes a full copy of an operating system, the application, necessary binaries and libraries - taking up tens of GBs.
- VMs can also be slow to boot.



VIRTUAL MACHINES