What is Containerization?

Containerization is a form of operating system virtualization, through which it runs applications in secluded user spaces called containers, all using the same shared operating system (OS).

What is Docker?

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and deploy it as one package. Initially, built for Linux, Docker now runs on Windows and macOS. To understand how Docker works, let's look at some components you would use to create Docker-containerized applications.

Docker Terms and Tool

Some of the tools and terminology you'll encounter when using Docker include the following:

> **DockerFile:** A DockerFile is a text file that contains instructions on how to build a docker image. A Dockerfile specifies the operating system that will underlie the container, along with the languages, environmental variables, file locations, network ports, and other components it needs—and what the container will do once we run it.

Docker Images: Docker images contain executable application source code as well as all the tools, libraries, and dependencies that the application code needs to run as a container. When you run the Docker image, it becomes one instance (or multiple instances) of the container.

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

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. All Docker Hub users can share their images at will. They can also download predefined base images to use as a starting point for any containerization project.

Docker Daemon: Docker Daemon is the background service running on the host that manages the building, running, and

distributing Docker containers. The daemon is the process that runs in the operating system in which clients speak.

Docker Engine: Docker Engine is a client-server application that supports the tasks and workflows involved to build, ship, and run container-based applications. The *engine* creates a server-side daemon process that hosts images, containers, networks, and storage volumes.

Docker Registry: The Docker Registry is where the Docker Images are stored. The Registry can be either a user's local repository or a public repository like a Docker Hub allowing multiple users to collaborate in building an application. Even with several teams within the same organization can exchange or share containers by uploading them to the Docker Hub, which is a cloud repository similar to GitHub.

Docker Compose: Docker-compose is for running multiple containers as a single service. It does so by running each container in isolation but allowing the containers to interact with one another.

Docker Swarm: Docker swarm is a service for containers that allows IT administrators and developers to create and manage a cluster of swarm nodes within the Docker platform. Each node of Docker swarm is a Docker daemon, and all Docker daemons interact using the Docker API. A swarm consists of

two types of nodes: a manager node and a worker node. A manager node maintains cluster management tasks. Worker nodes receive and execute tasks from the manager node.

Applications of Docker

Docker is an excellent tool for developers and system administrators. We can use it in multiple stages of the DevOps cycle and for the rapid deployment of applications. It allows the developers to build an application and package an application with all its dependencies into a Docker run container that can run in any environment.

Docker allows us to develop an application and its supporting components efficiently using the containers. These containers are lightweight and can run directly within the host machine’s kernel. Thus, it allows running more containers on a single hardware.

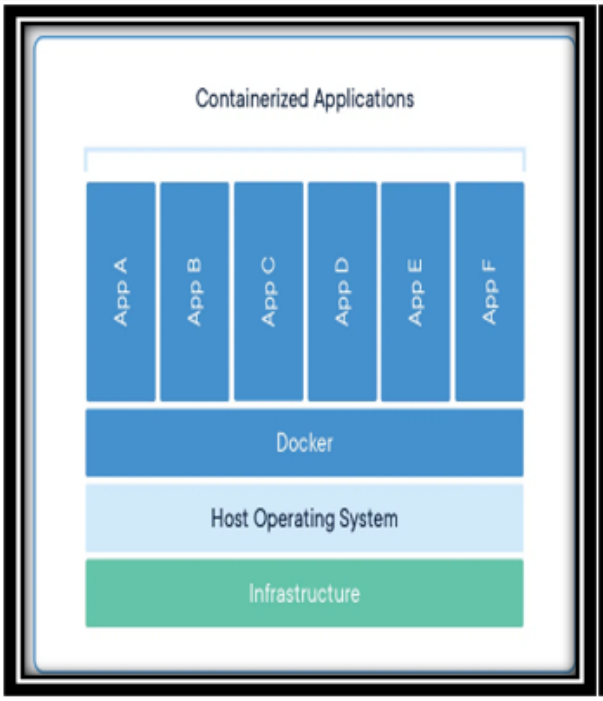
It provides a loosely isolated environment that is secure enough to run multiple containers simultaneously on a particular host.

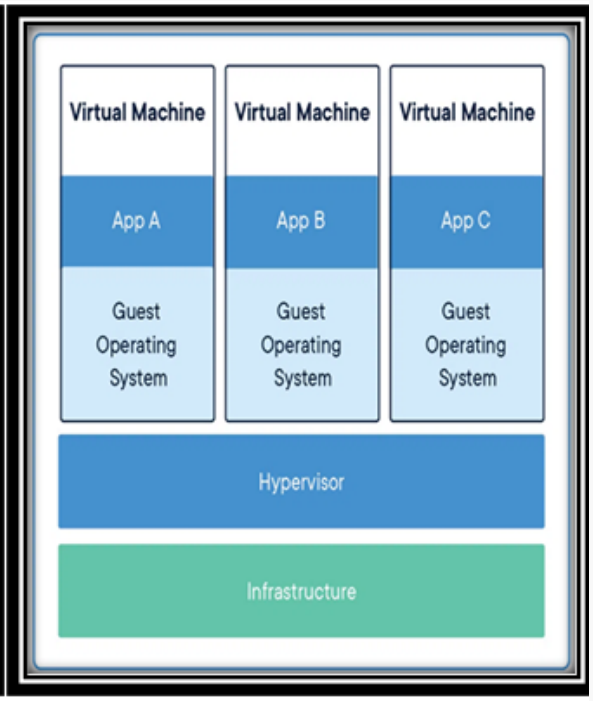
Any unforeseen condition or situation can halt the software development lifecycle and affect the business organization significantly. But, with Docker, it can be mitigated. Docker allows the functionality to easily replicate the file or Docker image to new hardware and retrieve it later in case of any issues. In case of rollback of any particular feature or version, Docker can be useful to revert to the last version of the Docker image quickly.

Docker allows developing, testing, and deploying applications faster. A software development life cycle is long, as it includes testing, making necessary changes, finding bugs, and deploying it to see the final results. Docker allows the developers to find bugs in the initial stages of development so that they can be fixed in the development environment and can be redeployed for testing and validation.

Comparing Containers and Virtual Machines

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

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.

VIRTUAL MACHINES

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.