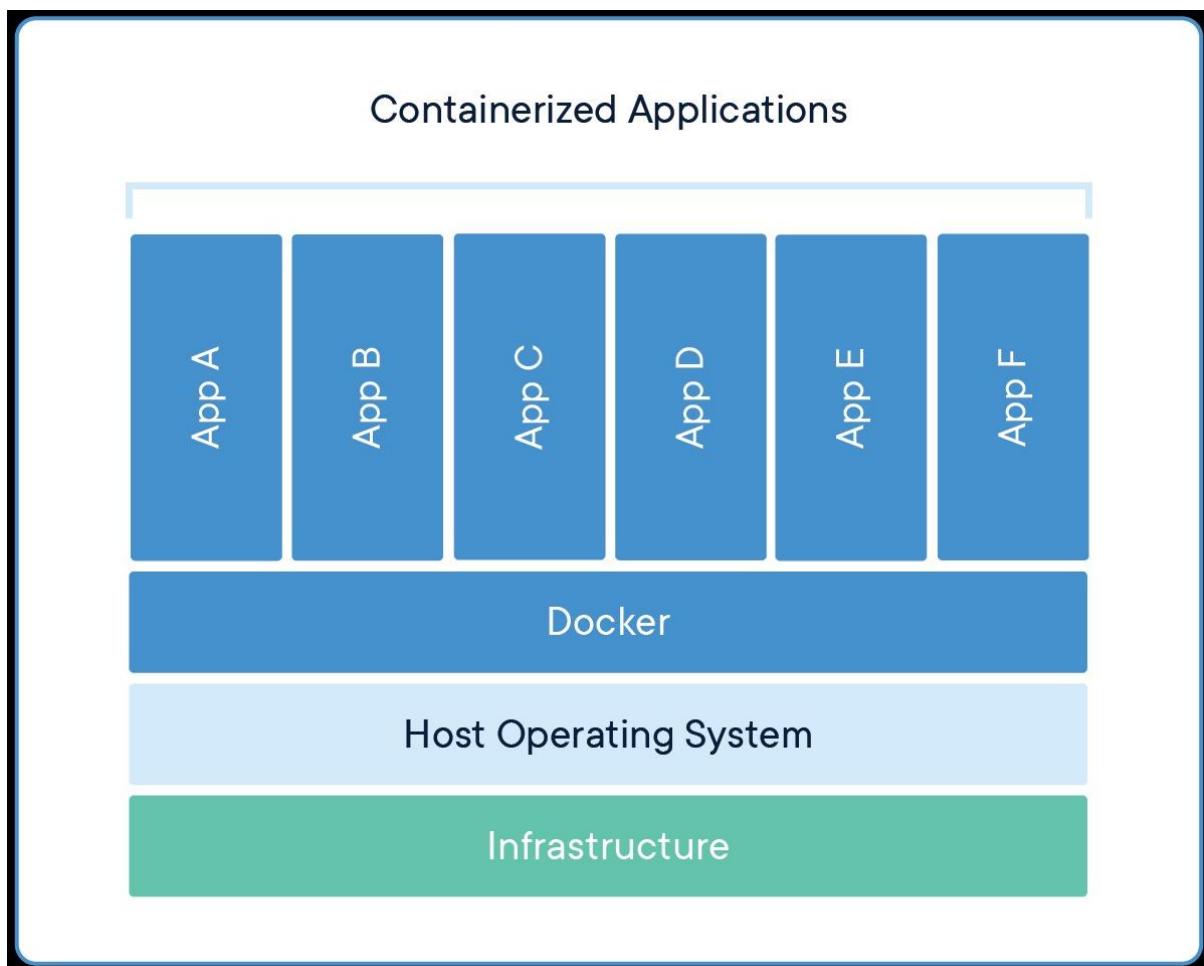


Task 11

What is Docker:

Docker is a containerization platform that allows you to package an application along with all its required libraries, dependencies, and configurations into a lightweight, portable unit called a container. These containers run consistently across different environments whether it's a developer's laptop, a testing server, or a cloud platform eliminating the "works on my machine" problem. Unlike virtual machines, Docker containers share the host operating system, making them faster, more efficient, and lightweight. With Docker, applications can be deployed quickly, scaled easily, and maintained with less complexity, making it a widely used tool in modern DevOps and cloud environments.



This image explains **how Docker works in a system** by showing the layers involved in running containerized applications.

- **Explanation of Each Layer (Top to Bottom)**

1. Containerized Applications (App A, App B, App C, App D, App E, App F)

- These represent different applications running in **separate containers**.
- Each app has its own isolated environment, so they don't conflict with each other.
- Even if they use different versions of libraries or frameworks, they run smoothly because each container is independent.

2. Docker

- This is the **container engine**.
- Docker is responsible for:
 - Creating containers
 - Running containers
 - Managing the container lifecycle
- It provides the platform that allows these applications to run in isolated environments.

3. Host Operating System

- This is the OS installed on the machine that runs Docker (e.g., Linux, Windows, macOS).
- Docker uses the underlying OS features (like namespaces, c groups) to create lightweight containers **without needing a full VM**.

4. Infrastructure

- This is the physical or cloud-based hardware:
 - On-prem servers
 - AWS, Azure, GCP virtual machines

- Any computing environment
- It provides the CPU, memory, storage, and networking that everything runs on.

Containerization:

Containerization in Docker refers to the process of using Docker to package an application and all its dependencies into a lightweight, isolated container. Docker provides the tools and platform to create these containers so the application runs consistently across different environments—whether it's a developer's system, a testing server, or any cloud platform. In Docker, containerization eliminates compatibility issues, reduces resource usage compared to virtual machines, and allows applications to be deployed, updated, and scaled faster. Essentially, Docker makes containerization simpler, more efficient, and highly portable for modern application development and DevOps workflows.

Virtualization:

Virtualization is a technology that allows you to create multiple simulated environments or virtual machines (VMs) on a single physical piece of hardware. Essentially, it uses a software layer called a **hypervisor** to divide the physical machine into multiple virtual machines, each with its own operating system and applications.

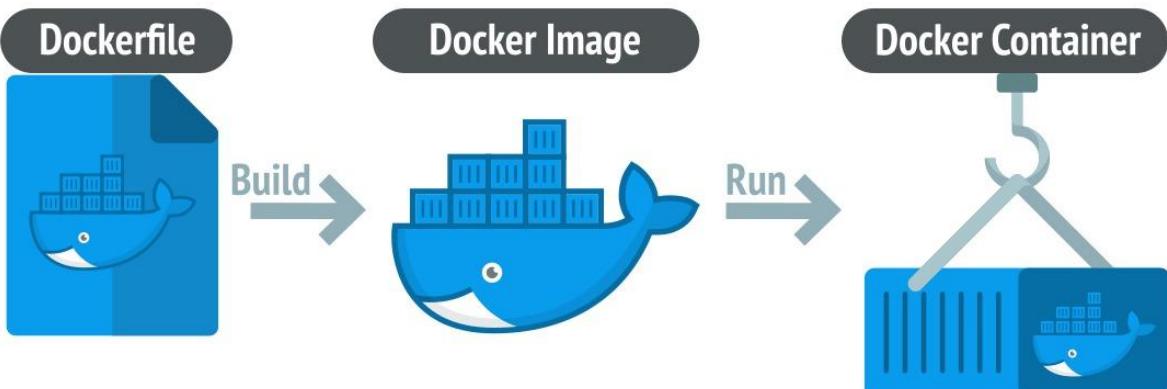
Diff Containers and Virtual Machines (VMs):

Feature	Containerization	Virtualization
Definition	Packages apps into lightweight containers sharing the host OS	Creates full virtual machines with their own OS
OS Usage	Shares host OS kernel	Each VM has its own separate OS
Size	Lightweight (MBs)	Heavy (GBs)
Startup Time	Seconds	Minutes
Resource Usage	Low	High
Performance	Faster (less overhead)	Slower (more overhead)

Feature	Containerization	Virtualization
Isolation Level	Process-level isolation	Full OS-level isolation
Technology Used	Docker, Kubernetes	Hypervisors (VMware, Hyper-V, KVM)
Portability	Highly portable	Less portable
Best Use Cases	Microservices, cloud-native apps, DevOps	Legacy apps, apps requiring full OS

Docker File:

A Docker file is a simple text file that contains a series of instructions that Docker uses to automatically build a Docker image. It acts like a blueprint or recipe, describing everything needed for the application to run—such as the base image, required packages, application files, environment settings, and the command that should run when the container starts. By using a Docker file, developers can create consistent and repeatable container images without manual setup, making application deployment faster, more reliable, and easier to automate in DevOps pipelines.



Docker Image:

A Docker image is a lightweight, read-only package that contains everything needed to run an application, including the code, runtime, libraries, dependencies, and configuration files. It is created from a Dockerfile and acts as a blueprint for launching containers. Once an image is built, Docker can use it to create one or multiple containers that all behave consistently. Because images are portable and versioned, they allow applications to be deployed easily across different environments without compatibility issues.

Docker container:

A Docker container is a lightweight, standalone, and isolated environment that runs an application along with all its required files, libraries, and dependencies. It is created from a Docker image and provides a consistent runtime environment across any system—whether it's a developer's laptop, a server, or a cloud platform. Containers share the host operating system's kernel, which makes them fast, efficient, and much lighter compared to virtual machines. In simple terms, a Docker container is the running instance of a Docker image that ensures your application works the same everywhere.

Docker Hub:

Docker Hub is a cloud-based repository service where developers can store, manage, and share Docker images. It acts as a central hub that provides access to thousands of official and community-contributed images, making it easy to download ready-to-use environments like Nginx, MySQL, Ubuntu, and more. Developers can also upload their own images to private or public repositories, enabling smooth collaboration and faster deployment. In simple terms, Docker Hub is the place where Docker images live, allowing users to pull, push, and distribute container images easily across different systems.