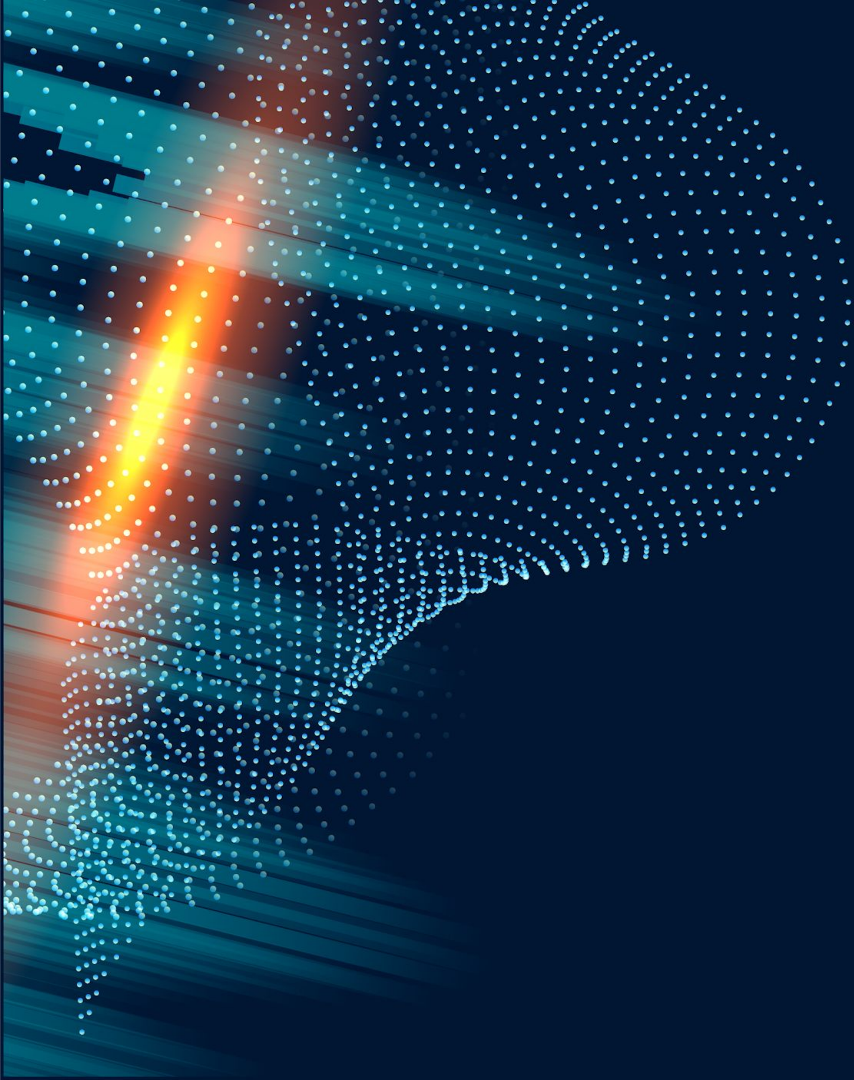


DOCKER

By Vignesh S

WHAT?

Docker is a popular platform for containerization of Applications

WHAT IS

CONTAINERIZATION?

CONTAINERIZATION = VIRTUALIZATION ?

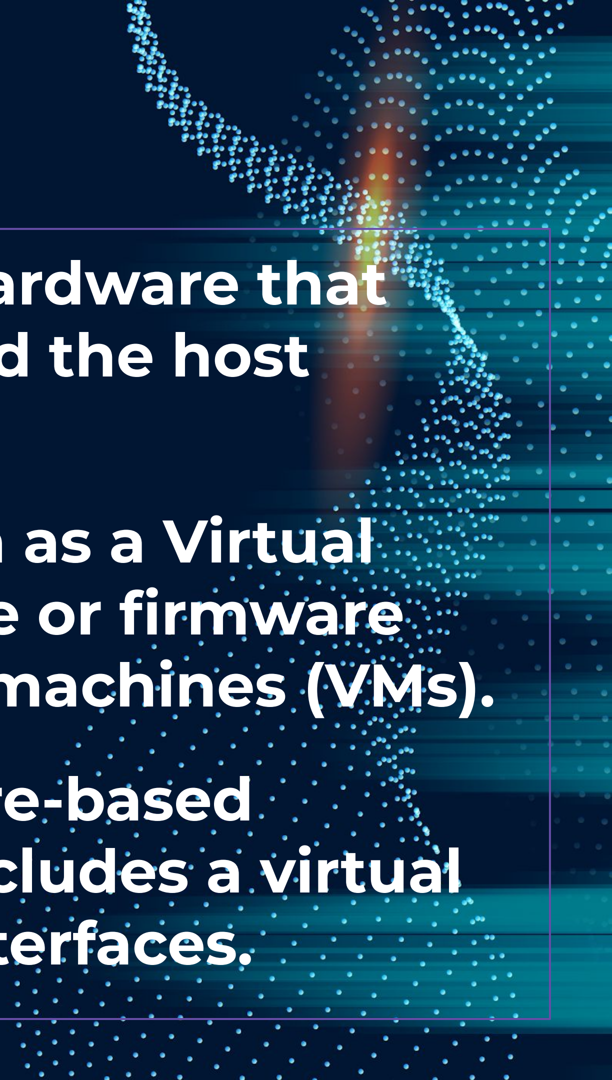
VIRTUALIZATION

Virtualization is a technology that allows you to create multiple virtual instances or environments on a single physical server or host.

It enables the efficient utilization of hardware resources and provides isolation between different virtual environments.

Virtualization has revolutionized data centers, cloud computing, and software development by making it easier to manage, scale, and optimize IT infrastructure.

COMPONENTS



Host Machine: The physical server or hardware that runs the virtualization software is called the host machine.

Hypervisor: The hypervisor, also known as a Virtual Machine Monitor (VMM), is the software or firmware that manages and controls the virtual machines (VMs).

Virtual Machine (VM): A VM is a software-based emulation of a physical computer. It includes a virtual CPU, memory, storage, and network interfaces.

Virtual Machines (VMs)

Virtual machines are the fundamental building blocks of virtualization.

They are complete, self-contained environments that run operating systems and applications as if they were running on a physical computer.

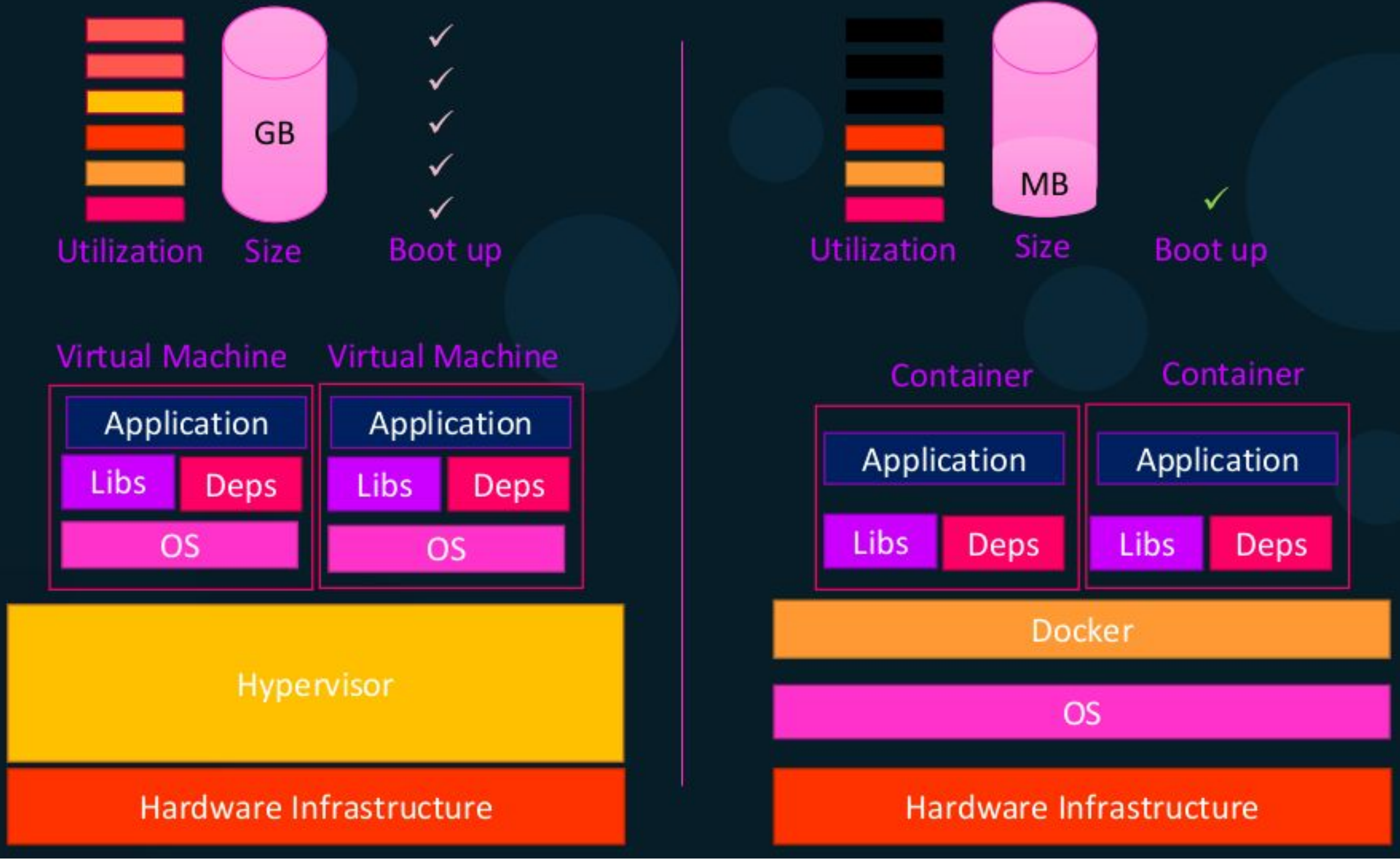
VM’s used for running multiple OS and applications on a single physical machine.

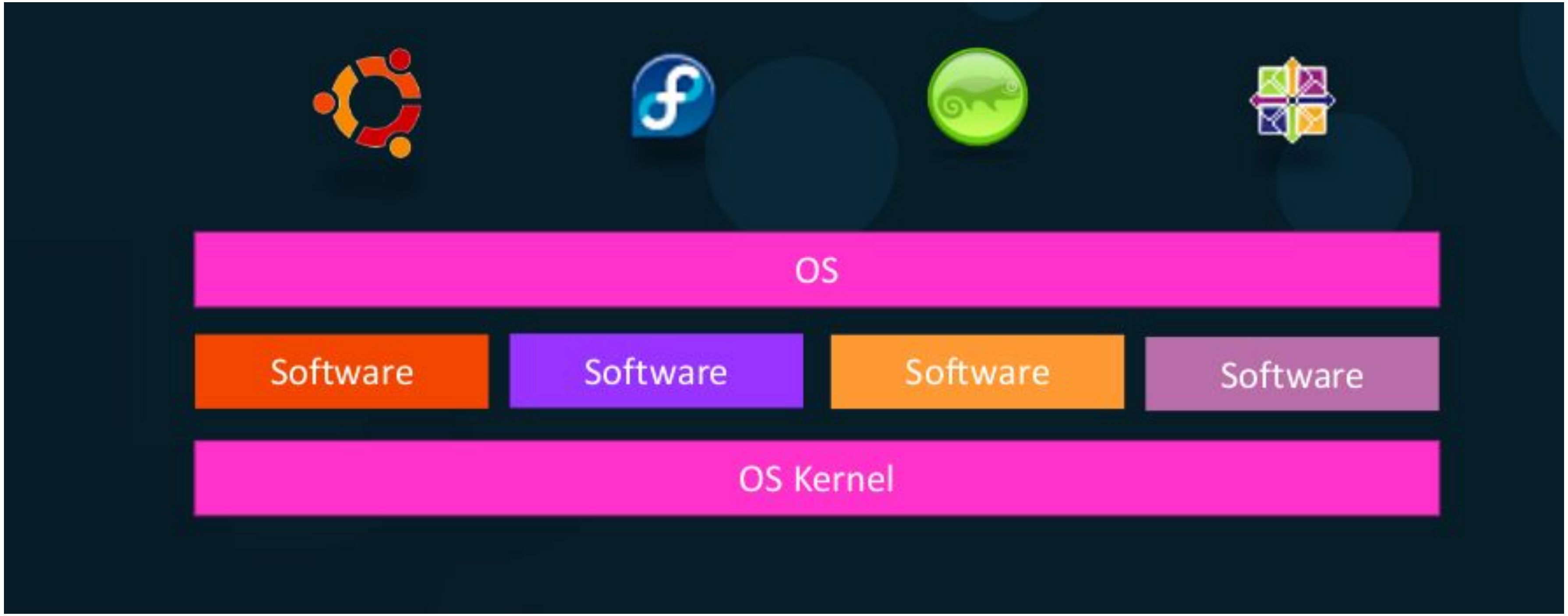
Containerization

Lightweight form of virtualization that allows you to package and run applications and their dependencies in isolated environments called containers.

Containerization is primarily used for application deployment and management.

Containers vs Virtual Machines




OS
Software
Software
Software
Software
OS Kernel

WHY ?

Isolation:

In virtualization, VMs are isolated at the operating system level, which means each VM can run a different OS.

In containerization, containers share the host OS but are isolated at the application level. This makes containers more lightweight since they don't require a separate OS for each instance.

WHY ?

Resource Efficiency:

VMs require separate OS instances, which can consume more memory and storage. Containers share the host OS kernel, making them smaller and faster to start and stop.

Portability:

You can create a container image on one system and run it on another without compatibility issues.

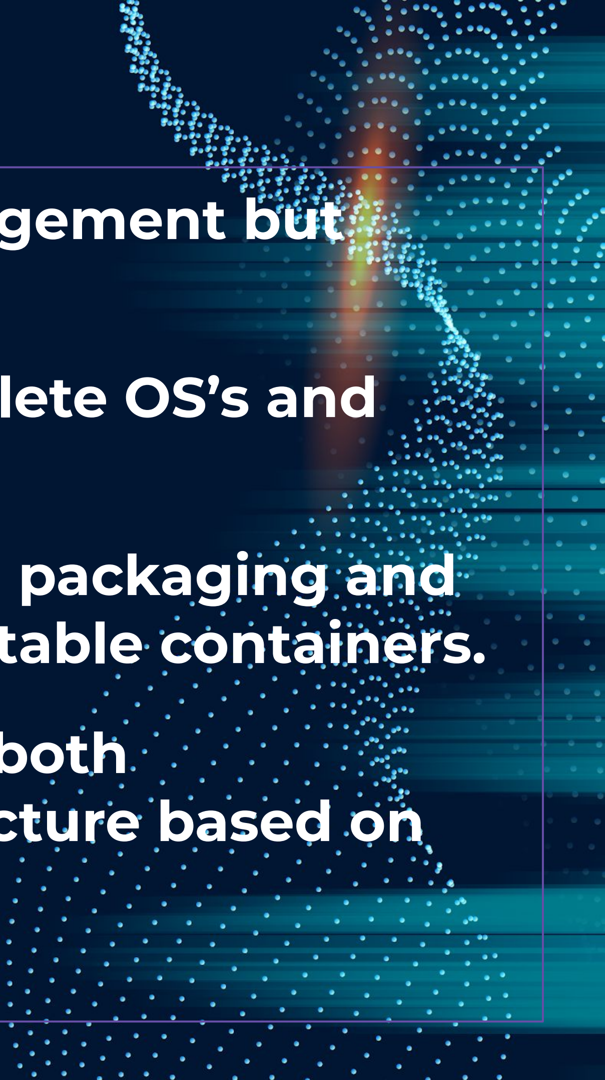
SUMMARY

Both provide isolation and resource management but target different use cases.

Virtualization is focused on running complete OS’s and workloads.

While containerization is centered around packaging and deploying applications in lightweight, portable containers.

Many organizations use a combination of both technologies to optimize their IT infrastructure based on specific requirements



DOCKER

Docker is a popular platform for developing, shipping, and running applications inside containers.

It revolutionized the way applications are packaged and deployed by introducing a standardized format for containers and tools to manage them efficiently.

Docker has become a fundamental tool in modern software development and deployment, enabling DevOps practices and facilitating the creation of microservices architectures.

TERMINOLOGIES

<ins>Docker Engine (dockerd):</ins> The core component of Docker that runs as a background service and manages containers on a host system.

<ins>Docker Container:</ins> A runnable instance created from a Docker image. Containers encapsulate the application code, dependencies, and runtime.

<ins>Docker Image:</ins> A template or blueprint for creating containers. Images include all the necessary components to run an application.

TERMINOLOGIES

<u>Dockerfile:</u> A text file that contains instructions for building a Docker image. It specifies the base image, dependencies, and application code.

<u>Docker Hub:</u> A public repository for Docker images, where users can find, share, and distribute container images.

<u>Docker Compose:</u> A tool for defining and running multi-container applications using a YAML configuration file. It simplifies the management of complex application stacks.

TERMINOLOGIES

Docker Swarm: Docker's native container orchestration and clustering tool, allowing the management of containers across a cluster of Docker nodes.

Docker Registry: A repository for storing and sharing Docker images. Docker Hub is a popular public registry, while organizations often set up private registries.

Volume: A Docker feature for persisting data outside of containers, ensuring data durability and sharing data between containers and the host system.

TERMINOLOGIES

Network: Docker provides networking capabilities that allow containers to communicate with each other and external networks. It supports various network modes and custom networks.

Port: Containers can expose ports to allow communication with services running inside the container. Port mapping allows mapping container ports to host ports.

Namespace: A feature that provides process and filesystem isolation for containers, ensuring that containers do not interfere with each other.

TERMINOLOGIES

Cgroups (Control Groups): Linux kernel feature used by Docker to limit and isolate the resource usage of containers, such as CPU, memory, and I/O.

CLI (Command Line Interface): The Docker CLI is a command-line tool for interacting with Docker, allowing users to manage containers and images using commands.

API (Application Programming Interface): Docker provides a RESTful API that allows programmatic interaction with the Docker daemon, enabling automation and integration with other tools.

DOCKER IMAGES

Fundamental building block in containerization.

It serves as a blueprint for creating containers, which are lightweight, isolated environments used to run applications.

Docker images are designed to encapsulate everything needed to run an application, including the application code, dependencies, libraries, and configuration.

CHARACTERISTICS

Immutable:

Docker images are immutable, meaning they cannot be modified once created. Any changes to an image result in the creation of a new image. This immutability ensures consistency and reproducibility in application deployments.

Layered File System:

Docker images are constructed using a layered file system. Layers are stacked on top of each other, with the final layer forming the complete image. This layered approach makes images efficient and allows for reuse of common layers among different images.

CHARACTERISTICS

Versioned:

Docker images can be versioned and tagged, making it easy to manage and track changes to images over time. Common version tags include version numbers, labels like "latest," or custom identifiers.

Dependency Management:

Docker images capture all dependencies required to run an application. This includes not only the application code but also libraries, configurations, and system tools. This eliminates compatibility issues and ensures that the application runs consistently in different environments.

Creating a Docker Image

To create a Docker image, you typically use a text file called a Dockerfile.

A Dockerfile contains instructions for building the image, specifying the base image, installing dependencies, adding application code, and configuring settings.

Docker uses the Dockerfile as a blueprint to build the image.

DOCKER FILE

FROM base_image:tag —---------------------- *(Choose a base image)*

RUN package_manager install package_name —------------------ *(Install packages)*

COPY source_path destination_path —----------------*(Copy from host to the container)*

EXPOSE port_number —------------------*(Expose ports to listen)*

CMD ["command", "arg1", "arg2"] —---------*(Define a command to run when the container starts)*

Working with Docker Images

Pulling Images: You can use the docker pull command to download Docker images from a registry to your local system.

Building Images: To create custom Docker images, you build them from a Dockerfile using the docker build command.

Running Containers: Docker images are used to create and run containers with the docker run command. Containers based on the same image share the same environment and configurations.

Pushing Images: To share custom images with others, you can push them to a Docker registry using the docker push command.

LETS HANDS ON !!