# Introduction

# to

# Containerization

## npNOG 10

November 25 - 28, 2024

# What is Containerization?

- Containerization is a lightweight form of virtualization that allows applications and their dependencies to be packaged into isolated containers.
- Unlike traditional virtualization, which relies on hypervisors to create multiple virtual machines (VMs) on a single physical server, containerization shares the host operating system kernel and only isolates the application's runtime environment.
- Everything an application needs to run - its binaries, libaries, configuration files and dependencies - is encapsulated and isolated in its container.
- These containers can run on any system with a container runtime installed, making them highly portable and efficient.

# Evolution of Container Technology

- The concept of containerization traces back to the early 2000s with technologies like FreeBSD jails and Solaris Zones. However, Docker revolutionized containerization in 2013 by providing an easy-to-use platform for creating, deploying, and managing containers.

# Why Containerization is Important

- Containerization offers several benefits, including improved consistency across environments, faster application deployment, efficient resource utilization, and simplified scaling.
- Most important, containerization allows applications to be “written once and run anywhere.”

# What is container?

- a lightweight OS-level virtualization method
- sand-alone piece of executable software
- NOT a virtual machine
- process with isolation, shared resources and layered filesystems

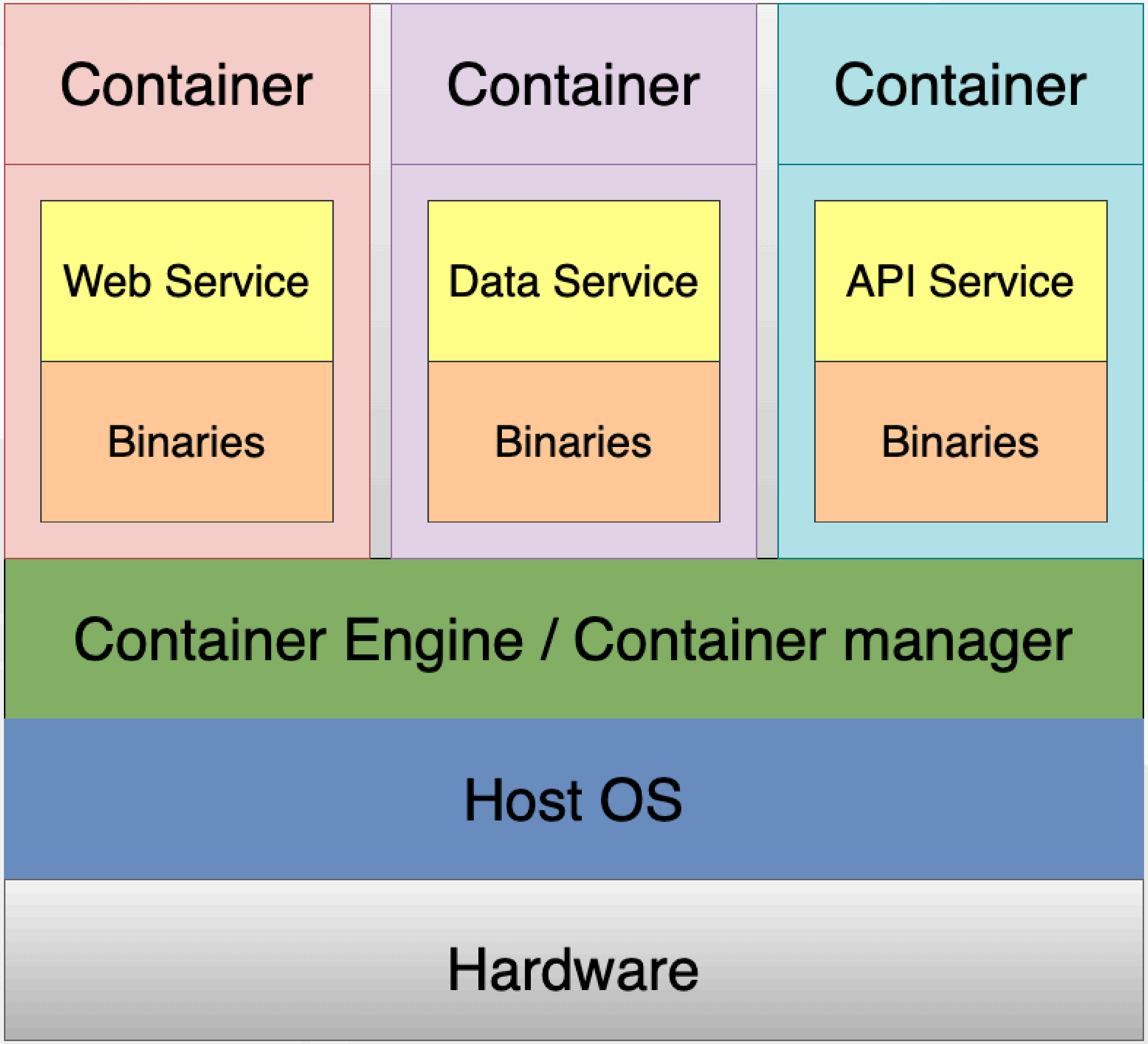
# Container Terms

- **namespace**: linux kernel feature that isolates and virtualizes system resources for a collection of processes and their children
    - **PID**: gives process own view of subset of system processes
    - **MNT**: gives process mount table and allows process to have own filesystem
    - **NET**: gives process own network stack (container can have virtual ethernet pairs to link to host or other containers)
    - **UTS**: gives process own view of system hostname and domain name
    - **IPC**: isolates inter-process communications (i.e. message queues)
    - **USER**: newest namespace that maps process UIDs to different set of UIDs on host (can map containers root uid to unprivileged UID on host)
- **cgroups**: control groups collect set of process tasks IDS together and apply limits, such as for resource utilization
    - enforce fair/unfair resource sharing between processes
    - exposed by kernel as special file system to mount
    - add a process or thread by adding process IDs to task file and read/configure values by editing subdirectory files
- **layered filesystems**: optimal way to make a copy of root filesystem for each container
    - one of the reasons why it is easy to move containers around
    - can "copy on write" (btrFS)
    - can use "union mounts" (aufs, OverlayFS) - way of combining multiple directories

# How Containers work?

- Each container is an executable packege of software, running on top of a host OS. A host(s) may support many containters (tens, hundreds or even thousands)
- At the bottom, there is the hardware including its CPU, disk storage and network interfaces.
- Above that, there is the host OS and its kernel - the latter servers as a bridge between the software of the OS and the hardware of the underlying system
- The container engine and its minimal guest OS, which are particular to the containerization technology being used, sit atop the host OS
- At the very top are the binaries and libraries (bins/libs) for each application and the apps themselves, runnning in their isolated user spaces (containers)

# Architecture



# Tools used in Containerization

- Docker
- Linux Containers (LXC)
- Kubernetes
- AWS ECS
- Azure Container Service

# Docker

Docker, or Docker Engine, is the most popular containerization platform. It simplifies the process of building, shipping, and running containers. For example, you can use Docker to package a web application and its dependencies into a container image, which can then be deployed on any Docker-enabled host.

# Linux Containers (LXC)

Commonly known as LXC, these are the original Linux container technology. LXC is a linux operating system-level virtualization method for running multiple isolated Linux systems (containers) on a single Linux host. It uses Linux kernel features such as namespaces and control groups (cgroups) to create isolated environments for applications. LXC provides a user-friendly interface for managing containers and is often used for development, testing, and deployment of applications.

# Kubernetes

Kubernetes is an open-source container-orchestration system for automating computer application deployment, scaling, and management. It was originally designed by Google and is now maintained by the Cloud Native Computing Foundation. It aims to provide a "platform for automating deployment, sacling and operations of application conteriners across cluster of hossts"

# AWS ECS

Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast container management service that makes it easy to run, stop and manage containers on a cluster.

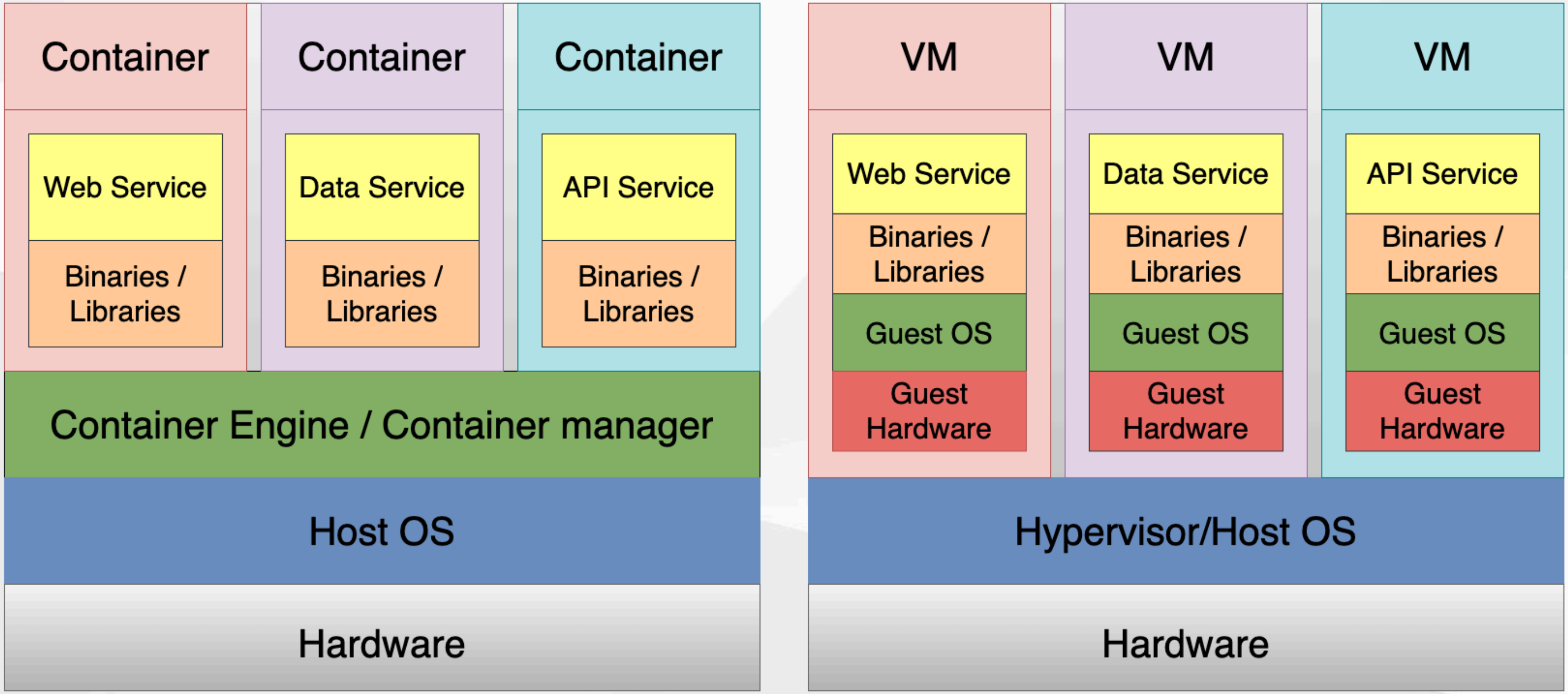
# Azure Container Service

ACS is a cloud-based container deployment and management sercie that supports popular open-source tools and technologies for container and container orchestration.

# Comparison with Virtualization

- While both containerization and virtualization provide isolation and resource management, containerization offers several advantages over traditional virtualization:
    - **Efficiency**: Containers are more lightweight than VMs since they share the host OS kernel, resulting in faster startup times and lower resource overhead.
    - **Portability**: Containers can run on any system with a compatible container runtime, making them highly portable across different environments.
    - **Scalability**: Containers can be rapidly instantiated and scaled horizontally to meet fluctuating workload demands.
    - **Consistency**: Containers encapsulate the application and its dependencies, ensuring consistency across development, testing, and production environments.

# Containers vs VMS



Container | Container | Container

Web Service | Data Service | API Service

Binaries / Libraries | Binaries / Libraries | Binaries / Libraries

Container Engine / Container manager

Host OS

Hardware

VM | VM | VM

Web Service | Data Service | API Service

Binaries / Libraries | Binaries / Libraries | Binaries / Libraries

Guest OS | Guest OS | Guest OS

Guest Hardware | Guest Hardware | Guest Hardware

Hypervisor/Host OS

Hardware

# VMS vs Containers

| VMS | Containers |
|---|---|
| - hypervisors run software on physial server to emulate a particualr hardware system (aka a virtual machine) | - run isolated process on a **single server or host operating system (OS)** |
| - VM runs a **full copy of the operating system (OS)** | - can migrate only to server with compatible OS kernels |
| - can run multiple applications | - best for a single application |

# Disadvantage of Containerization

- **Security**: One can not ignore the security issues with the container and associated containers. In reality, hackers can penetrate its OS-level virtualization. Yes, they fo have this flaw, but it is not that easy to breach conttainers' security. While in the case of Virtual Machines, you have a hypervisor that proviedes a petite point of the breach, which is more secure than that of the container's surface.
- **Monitoring**: Sometimes, there could be a chance that many containers are working on the same server, which is a good thing. But when you look at tahe maintenance side, you will find it very hard to manage it all. This could lead to many mishaps around the system.

# Conclusion

- Various containerization tools are available in the market that could best suit our requirements. The power of containers helps get the desired output, irrespective of the OS at the client-side or other dependencies.
- In the lifecycle of software, there could be multiple rolling updates necessary for creating the best software. Now, you get to choose which containtgers are the best fit and capable of updating very easily and quickly on the go for you.

?