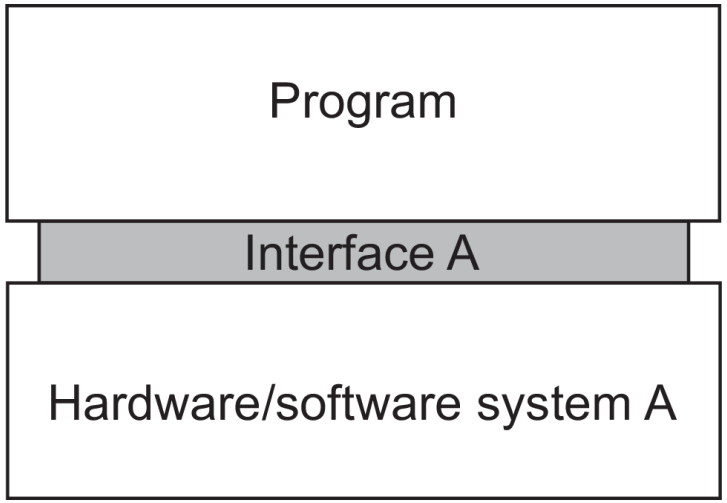
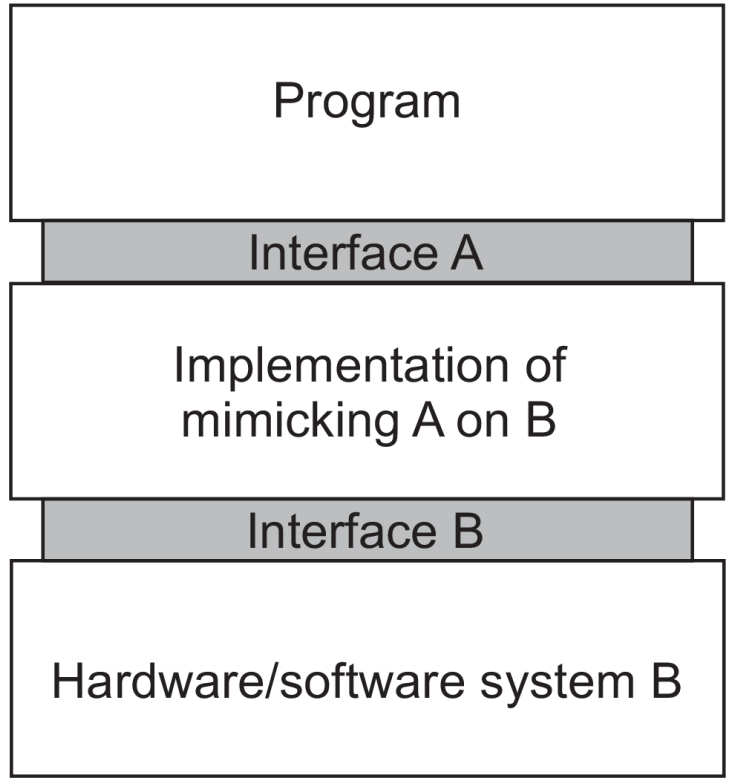
# CSE 601: Distributed Systems

Toukir Ahammed

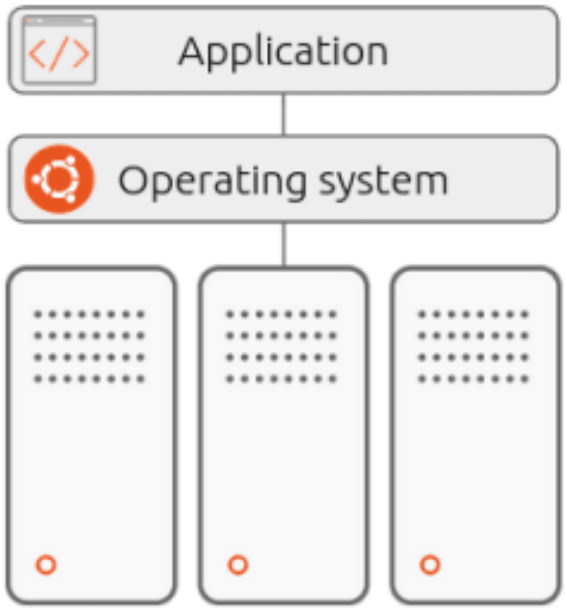
# Virtualization

- Virtualization deals with extending or replacing an existing interface so as to mimic the behavior of another system,

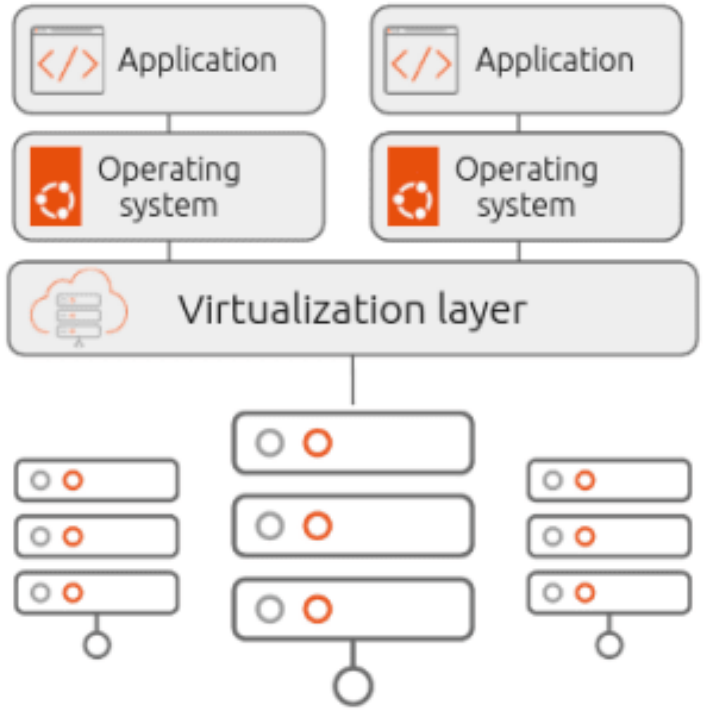


# Traditional and Virtualized Environment




Traditional environment



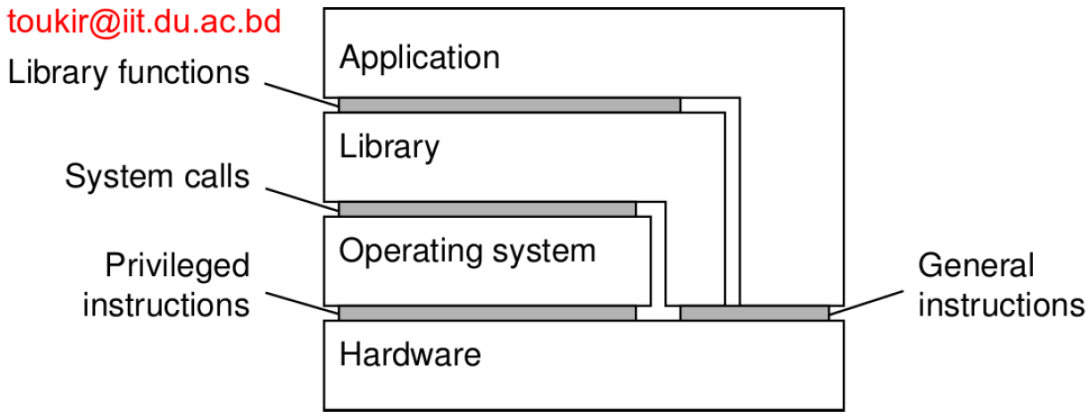

Virtualized environment

# Virtualization

Four different types of interfaces, at four different levels:

1. An interface between the hardware and software, consisting of **machine instructions** that can be invoked by any program.
2. An interface between the hardware and software, consisting of machine instructions that can be invoked only by **privileged programs**, such as an operating system.
3. An interface consisting of **system calls** as offered by an operating system.
4. An interface consisting of library calls, generally forming what is known as an **application programming interface** (**API**).





# Virtualization

- Virtualization is a process of creating an abstraction layer over hardware, allowing a single computer to be divided into multiple virtual computers.
- Each of those virtual computers (known as “guests”) uses part of the hardware resources of the main computer (known as a “host”).
- The software used to achieve this is a hypervisor. Hypervisors run on a host operating system and enable multiple guest operating systems to run on top of it, sharing the same physical computing resources managed by the host operating system.

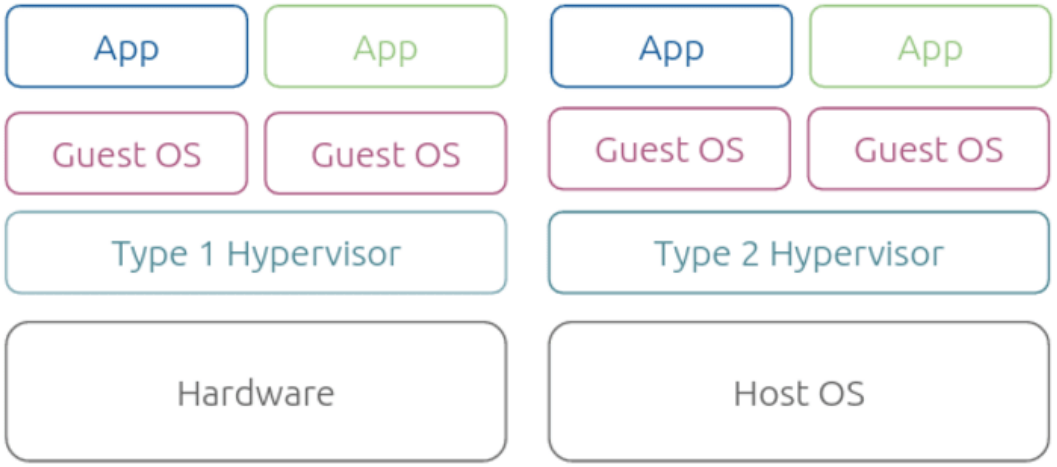
# Virtualization

Virtualization is important:

- Hardware changes faster than software
- Ease of portability and code migration
- Isolation of failing or attacked components

# The role of a Hypervisor

- Virtualization wouldn't be possible without a hypervisor (also known as a virtual machine monitor) – a software layer enabling multiple operating systems to co-exist while sharing the resources of a single hardware host.
- The hypervisor acts as an intermediary between virtual machines and the underlying hardware, allocating host resources such as memory, CPU, and storage.



# The role of a hypervisor

- **Type 1 hypervisors**, also known as bare-metal hypervisors, run directly on the host's hardware and are responsible for managing the hardware resources and running the virtual machines.
- Examples of Type 1 hypervisors include VMware ESXi, Microsoft Hyper-V, and Citrix XenServer.
- **Type 2 hypervisors**, also known as hosted hypervisors, run on top of a host operating system and rely on it to provide the necessary hardware resources and support.
- Examples of Type 2 hypervisors include VMware Workstation and Oracle VirtualBox.

# Types of Virtualization

- Server Virtualization
- Storage Virtualization
- Network Virtualization
- Application Virtualization

# Containers

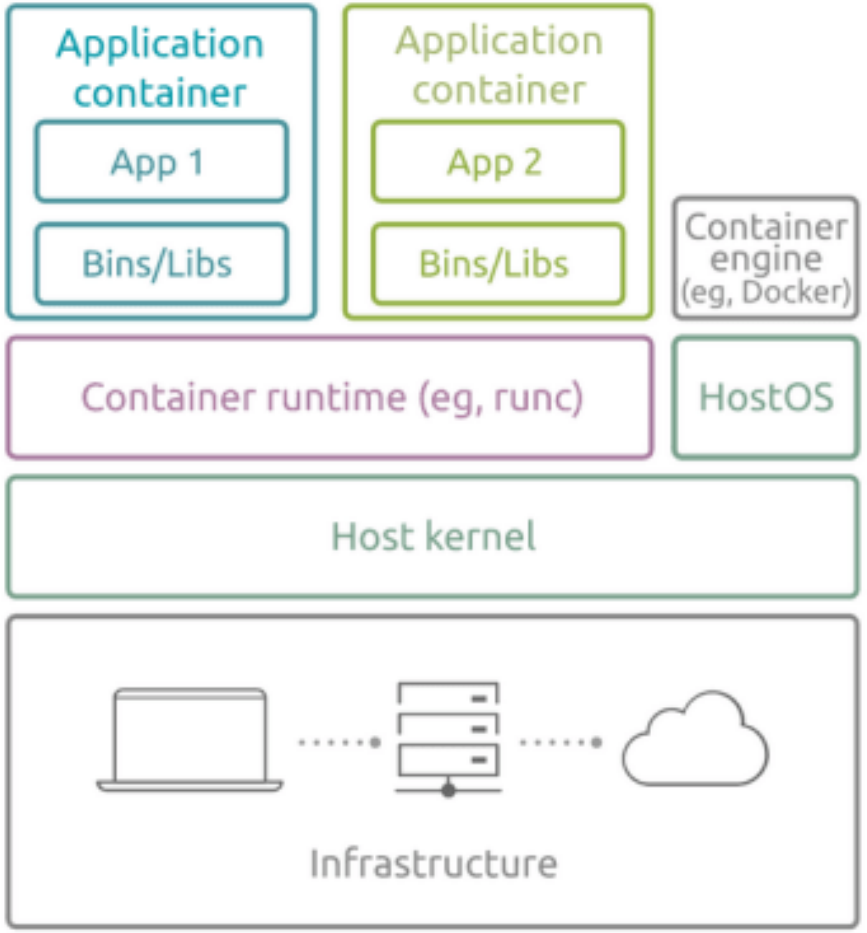
- A container can be thought of a collection of binaries (also called images) that jointly constitute the software environment for running applications.
- A container is a lightweight alternative to full machine virtualization, which involves encapsulating an application in an isolated operating environment, with all the files and libraries it needs to operate.
- Every containerized application can share the host system's user space, while still maintaining its individual system processes, environment variables, and libraries.

# What is containerization?

- Containerization also allows users to run many instances on a single physical host, but it does so without needing the hypervisor to act as an intermediary.
- Instead, the functionality of the host system kernel is used to isolate multiple independent instances (containers).
- By sharing the host kernel and operating system, containers avoid the overhead of virtualization, as there's no need to provide a separate virtual kernel and OS for each instance.

# Application Containers

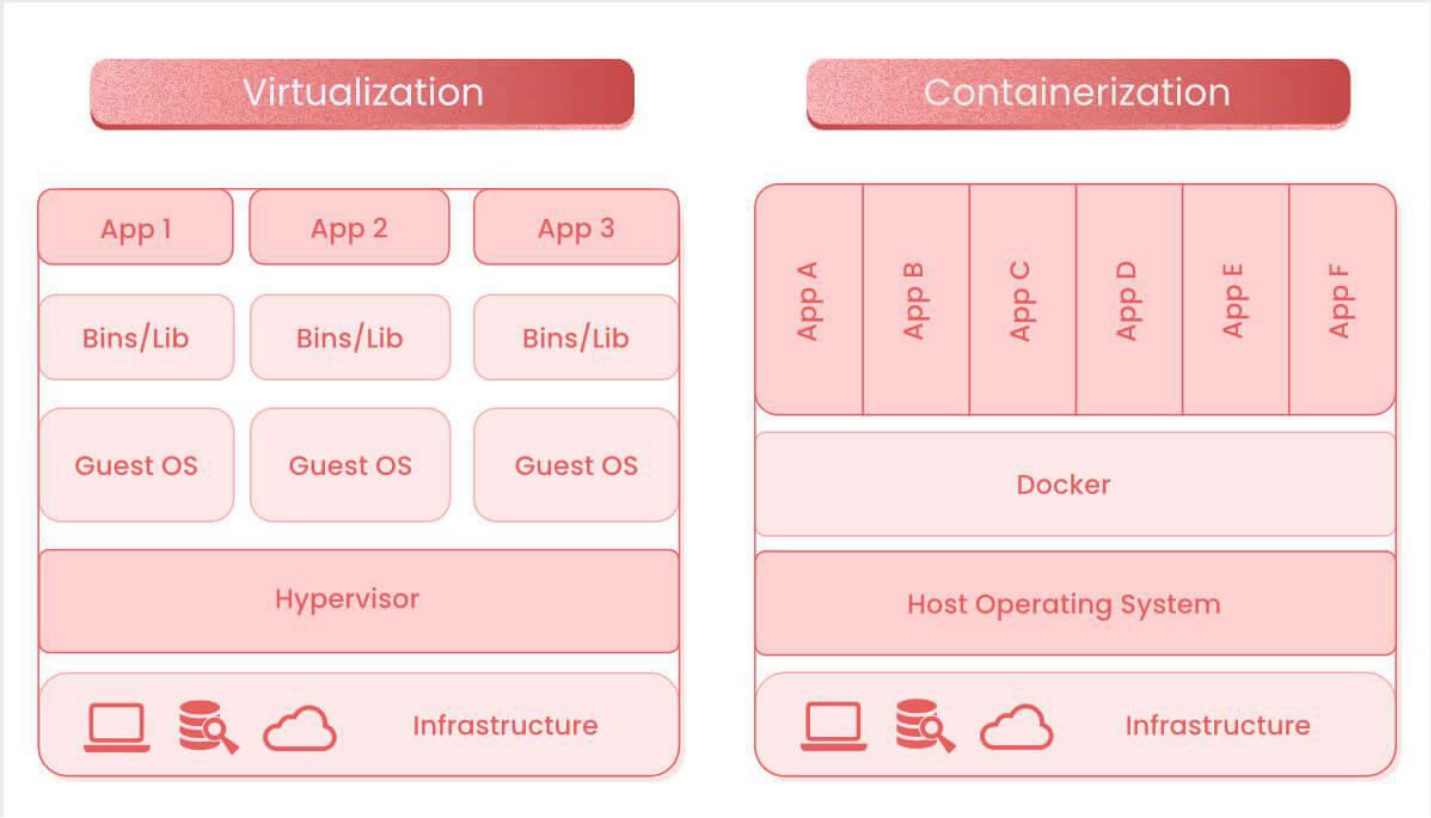
- Application containers (such as Docker), also known as process containers, package and run a single process or a service per container.
- They are packaged along with all the libraries, dependencies, and configuration files they need, allowing them to be run consistently across different environments.




Application containers (eg, Docker)

# Containerization vs. Virtualization

Virtualization and containerization are the two most frequently used mechanisms to host applications in a computer system.



# Containerization vs. Virtualization

**Resource Overhead**

- When comparing containerization vs virtualization in terms of resource overhead, containerization is the clear winner.
- Because containers share the host system's operating system, and do not need to run a full operating system, they are significantly more lightweight and consume fewer resources.
- Virtual machines, on the other hand, each require their own OS, which increases the overhead.

# Containerization vs. Virtualization

**Startup Time**

- In general, containers start up more quickly than VMs, because they don't have to start up an entire operating system.
- Virtual machines take much longer to boot up.
- This means containers are more flexible and can be torn down and restarted whenever needed, supporting immutability, which means that a resource never changes after being deployed.

# Containerization vs. Virtualization

**Portability**

- Both containers and virtual machines offer a high degree of portability.
- However, containers have a slight edge because they package the application and all of its dependencies together into a single unit, which can be run on any system that supports the container platform.
- Virtual machines, while also portable, are more dependent on the underlying hardware.

# Containerization vs. Virtualization

**Security Isolation**

- In terms of security isolation, virtual machines have the advantage.
- Because each VM is completely isolated from the host system and other VMs, a security breach in one VM typically does not affect the others (although it is possible to compromise the hypervisor and take control of all VMs on the device).
- Containers, while isolated from each other, still share the host system's OS, so a breach in one container could possibly leak to other containers.

# Containerization vs. Virtualization

**Scalability and Management**

- The lightweight nature and rapid startup time offered by containers make them ideal for scaling applications quickly and efficiently.
- They also lend themselves well to the microservices architecture, which can simplify the management of complex applications.
- Virtual machines, while also scalable, are more resource-intensive and take longer to start, making them less suitable for microservices and distributed applications.

# Use Cases for Virtualization

- Legacy Applications
- Environments Needing Strong Isolation
- IaaS Scenarios

# Use Cases for Containerization

- Microservices Architectures
- CI/CD
- PaaS Scenarios