BITS Pilani presentation

BITS Pilani
Pilani Campus

Paramananda Barik
CS&IS Department

innovate achieve lead

BITS Pilani

Pilani Campus

SEZG586/SSZG586, Edge Computing

Lecture No.8

Agenda

- What is Virtualization?
- Types of Virtualization
- How Virtualization Works
- Benefits of Virtualization
- Challenges of Virtualization
- Use Cases & Applications
- Virtualization Technologies (Hypervisors, Containers)
- Use Cases

What is Virtualization?

What is Virtualization?

Definition:

Virtualization is the process of creating a virtual version of something, such as computer hardware, operating systems, or network resources.

Key concepts:

Virtual Machines (VMs), Hypervisors, Containers

What is Virtualization?

What is Virtualization?

Definition:

Virtualization is the process of creating a virtual version of something, such as computer hardware, operating systems, or network resources.

Types of Virtualization

1. Hardware Virtualization (Virtual Machines)
2. OS Virtualization (Containers)
3. Network Virtualization
4. Storage Virtualization
5. Desktop Virtualization

Types of Virtualization

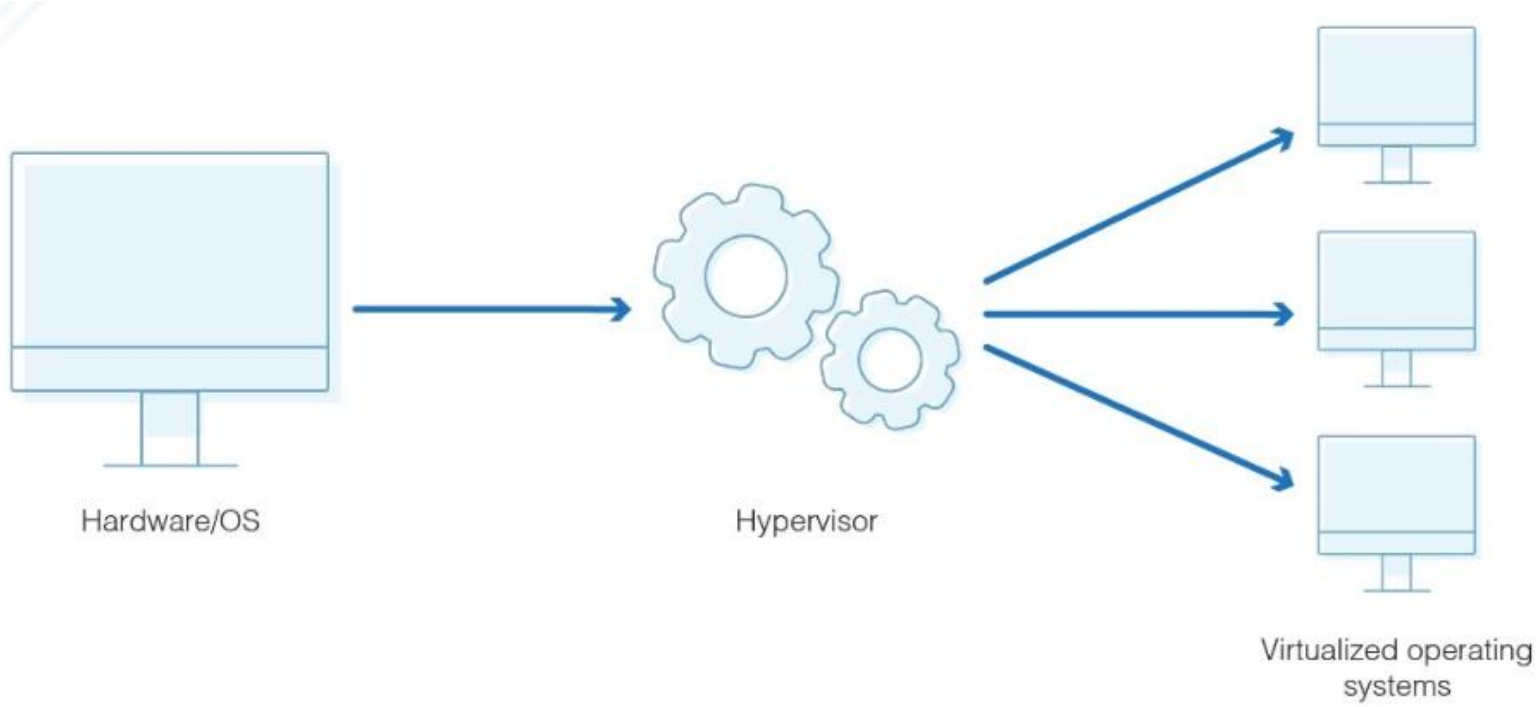
1. **Hardware Virtualization (Virtual Machines)**
   - Hardware virtualization is the process of running multiple virtual machines (VMs) on a single physical machine by abstracting and emulating the underlying hardware.
   - This is achieved through a software layer called a hypervisor, which manages the interaction between the physical hardware and the virtual machines.
   - Each VM operates as if it has its own dedicated hardware, such as CPU, memory, and storage, even though these resources are shared across multiple VMs.

Key Components of Hardware Virtualization:

1. Hypervisor :
2. Virtual Machine
3. Virtual Hardware

Key Components of Hardware Virtualization:

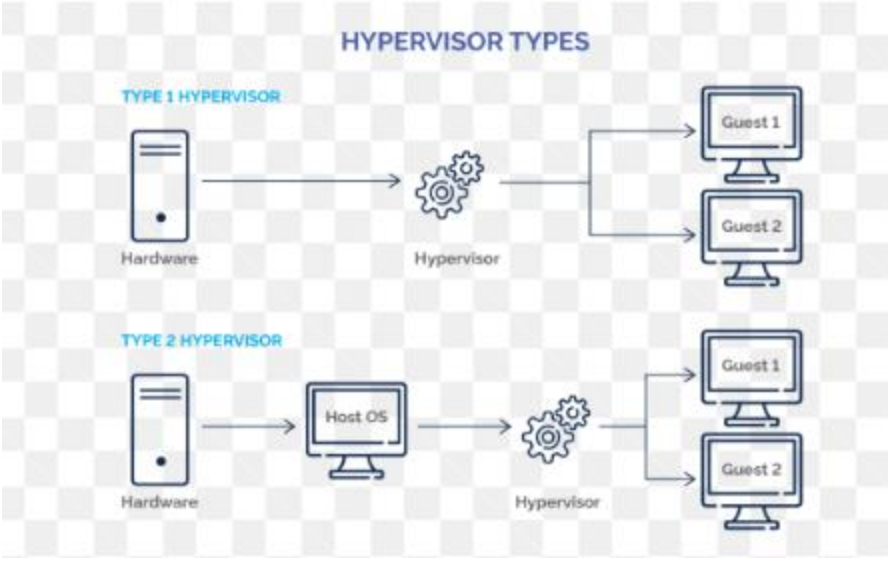
1. Hypervisor :



Key Components of Hardware Virtualization:

innovate achieve lead

1. **Hypervisor :**
   - **Type 1 (Bare-metal):** Runs directly on the physical hardware and controls the hardware resources. Examples include VMware ESXi, Microsoft Hyper-V, and Xen.
   - **Type 2 (Hosted):** Runs on a host operating system, which in turn runs on physical hardware. Examples include Oracle VirtualBox and VMware Workstation.



Key Components of Hardware Virtualization:

1. **Virtual Machine**
   - A software emulation of a physical machine.
   - Each VM runs its own operating system and applications independently.
2. **Virtual Hardware**
   - The hypervisor presents virtualized hardware components (like virtual CPUs, memory, and storage) to the VM, which interacts with these resources as if they were physical.

Benefits of Hardware Virtualization:

Resource Efficiency : Multiple VMs can share the same physical server, maximizing resource utilization.

Isolation: VMs are isolated from one another, ensuring that issues in one VM don't affect others.

Cost Reduction: Reduces the need for physical hardware, leading to lower infrastructure and energy costs.

Flexibility: VMs can be easily created, copied, moved, or deleted, making it easier to manage workloads.

Scalability: It is easier to scale out by adding more VMs rather than purchasing additional hardware.

OS Virtualization (Containers)

OS Virtualization :

- OS virtualization refers to the process of abstracting and isolating different applications on the same operating system, allowing them to run independently within their own isolated environments, called **containers**.
- Unlike hypervisor-based virtualization, where each virtual machine has its own OS, containers share the host OS kernel but operate in isolated user spaces.

OS Virtualization Key Concepts:

1. Containerization:

- Containers are lightweight, executable units that package application code along with its dependencies (libraries, configuration files, etc.) in a way that ensures the application runs consistently in different computing environments.
- Each container is isolated from others, but they all share the same host OS kernel.
- Popular container technologies include Docker and orchestration platforms like Kubernetes.

OS Virtualization Key Concepts:

Benefits of OS Virtualization (Containers):

- Efficiency: Containers share the host OS kernel, meaning they don't require the overhead of running multiple operating systems, making them more resource-efficient than traditional VMs.
- Faster Startup: Containers can be started or stopped quickly, making them ideal for microservices architectures and applications that need rapid scaling.
- Portability: Containers package everything an application needs to run, ensuring consistent behavior across development, testing, and production environments.
- Scalability: Because containers are lightweight, they can easily be deployed in large numbers to meet scaling demands.

OS Virtualization Key Concepts:

innovate achieve lead

How Containers Work:

- Containers use a shared kernel but isolate the user space (i.e., file system, network, process tree) to create a secure, isolated environment for applications.
- They are managed by container runtimes such as Docker or containerd, which manage the lifecycle of containers (starting, stopping, and monitoring).
- Kubernetes is commonly used for orchestrating large-scale container deployments, automating tasks like scaling, load balancing, and monitoring across clusters.

Network Virtualization

Network Virtualization

- Network Virtualization is the process of abstracting physical network resources (such as routers, switches, firewalls, and bandwidth) into virtual networks.
- These virtual networks are treated as logical entities, separate from the physical infrastructure, allowing for more flexibility, scalability, and efficient use of network resources.

Key Concepts of Network Virtualization

Virtual Networks:

- A virtual network is a logical overlay built on top of physical network hardware. It behaves like a traditional network but operates independently of the physical devices.
- Each virtual network can have its own IP addressing, routing, and network policies.

Key Concepts of Network Virtualization

Network Abstraction:

- In network virtualization, the underlying physical hardware is abstracted so that network traffic, devices, and connections are treated as software-defined entities.
- This allows for more flexible management of network traffic and resources, decoupling the network from the hardware.

Key Concepts of Network Virtualization

Software-Defined Networking (SDN):

- **SDN** is a core technology in network virtualization, where the network control plane (which decides where traffic is sent) is separated from the data plane (which forwards traffic).
- A centralized SDN controller manages traffic routing and policies across the network, enabling dynamic reconfiguration of virtual networks in real-time.

Key Concepts of Network Virtualization

Network Functions Virtualization (NFV):

- **NFV** virtualizes traditional network services like firewalls, load balancers, and intrusion detection systems, running them as software rather than hardware appliances.
- This makes it easier to deploy and scale network functions as needed, using general-purpose servers rather than specialized hardware.

Types of Network Virtualization

Types of Network Virtualization:

- **Internal Network Virtualization:**
  - This involves segmenting and managing traffic within a single physical network.
  - **VLANs (Virtual LANs)** and **VXLANs (Virtual Extensible LANs)** are examples, where parts of the network are isolated for security or performance purposes

- **External Network Virtualization:**
  - This abstracts multiple physical networks into a unified virtual network. It is typically used in data centers and cloud environments.
  - By pooling network resources, external network virtualization allows for flexible and scalable networking across multiple physical infrastructures.

Use Cases

Use Cases:

1. **Cloud Computing**: Network virtualization is essential in cloud platforms like AWS, Azure, and Google Cloud to provide isolated, scalable networking for multiple tenants or applications.
2. **Data Centers**: Virtualized networks enable more efficient use of resources by supporting dynamic workload balancing, improved disaster recovery, and easier scaling.
3. **Software-Defined Wide Area Networks (SD-WAN)**: SD-WAN uses network virtualization to optimize the performance of wide-area networks (WANs), dynamically routing traffic based on current conditions and policy rules.

Storage Virtualization

Storage Virtualization:

- The process of abstracting physical storage resources to create a unified, centralized virtual storage system that can be managed and optimized more efficiently.
- It decouples storage hardware from the software that controls it, making it easier to pool and allocate resources without regard to the underlying physical infrastructure.

Storage Virtualization

Examples of Storage Virtualization:

1. Data Centers:

1. Virtualized storage helps data centers optimize space, reduce costs, and simplify management by allowing more flexible use of physical storage resources.

2. Disaster Recovery:

1. Virtualized storage systems can replicate data across multiple locations, making disaster recovery faster and more efficient.

3. Cloud Storage:

1. Cloud providers use storage virtualization to provide scalable, flexible, and cost-efficient storage solutions to users.
2. Users can access storage resources without needing to know the underlying infrastructure, which is managed and abstracted by the cloud provider.

Desktop Virtualization

Desktop Virtualization:

- The process of creating and running virtual desktops, where the desktop environment (including the operating system, applications, and user data) is decoupled from the physical hardware, such as a traditional desktop PC or laptop.
- This allows users to access their desktop environments remotely from any device while administrators can centrally manage and secure these virtual desktops.

Example of Desktop Virtualization

Virtual Desktop Infrastructure (VDI):

- VDI is a technology that hosts desktop environments on virtual machines (VMs) running on centralized servers. Users can access these virtual desktops from their personal devices over the network, typically through a thin client, web browser, or remote desktop protocol (RDP).
- Virtual desktops in a VDI environment are stored and managed centrally in a data center or the cloud, providing IT departments with centralized control over the entire desktop environment.

How Virtualization Works

- **Type 1 Hypervisor (Bare-metal):**
  - A Type 1 hypervisor runs directly on the physical hardware without an underlying operating system.
  - It manages hardware resources (CPU, memory, storage, etc.) and allocates them to the VMs.
  - Type 1 hypervisors are often used in enterprise environments and data centers.
  - Examples: VMware ESXi, Microsoft Hyper-V, Xen.
- **Process**
  - The physical hardware (e.g., a server) boots the hypervisor.
  - The hypervisor creates multiple VMs, each with its own virtual CPU, memory, and storage.
  - The VMs run their own operating systems, such as Linux or Windows, independently of each other.

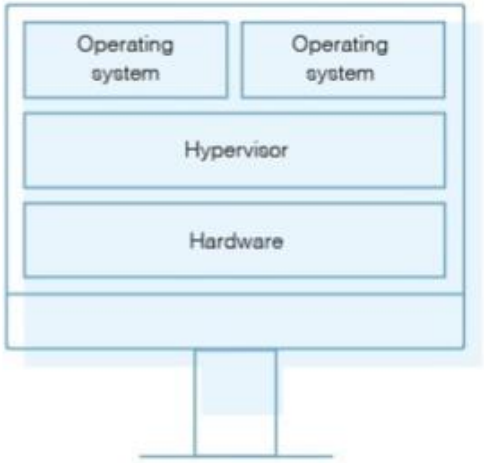
How Virtualization Works

- **Type 2 Hypervisor (Hosted):**
  - A Type 2 hypervisor runs on top of a host operating system (like Windows or Linux), which in turn runs on the physical hardware.
  - This type of hypervisor is typically used for development, testing, or personal use where users want to run multiple OS instances on their desktop or laptop.
  - Examples: Oracle VirtualBox, VMware Workstation.
- **Process**
  - The physical machine runs a host operating system.
  - The Type 2 hypervisor operates as an application within the host OS.
  - The hypervisor then creates VMs that are managed like separate applications running on the host OS. These VMs also have their own OS and applications

How Virtualization Works

innovate achieve lead

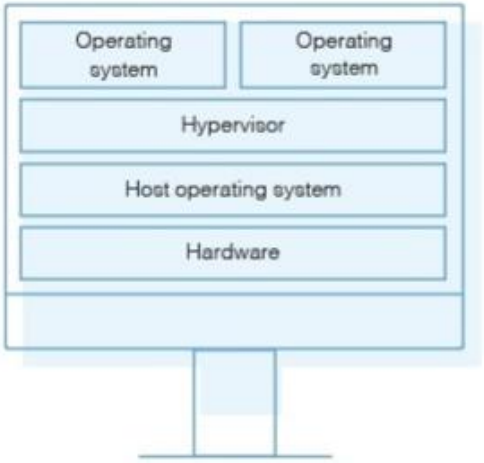
1. **Relationship between hardware, hypervisor, VMs, and operating systems:**



Native or Bare Metal

Runs directly on the hardware

- Isolates partitions
- Security features



Hosted or Embedded

Runs within and uses the host OS

- Ease to install and use
- Low cost

Benefits of Virtualization

innovate achieve lead

1. Efficient resource utilization
2. Reduced hardware costs
3. Easier management and scalability
4. Enhanced disaster recovery
5. Isolation and security benefits

Challenges of Virtualization

innovate achieve lead

1. Performance overhead
2. Security concerns (VM escape, vulnerabilities)
3. Licensing costs
4. Complexity in management

Use Cases of Virtualization

innovate achieve lead

1. Data centers and cloud computing
2. Testing and development environments
3. Server consolidation
4. Running legacy applications

Virtualization Technologies

innovate achieve lead

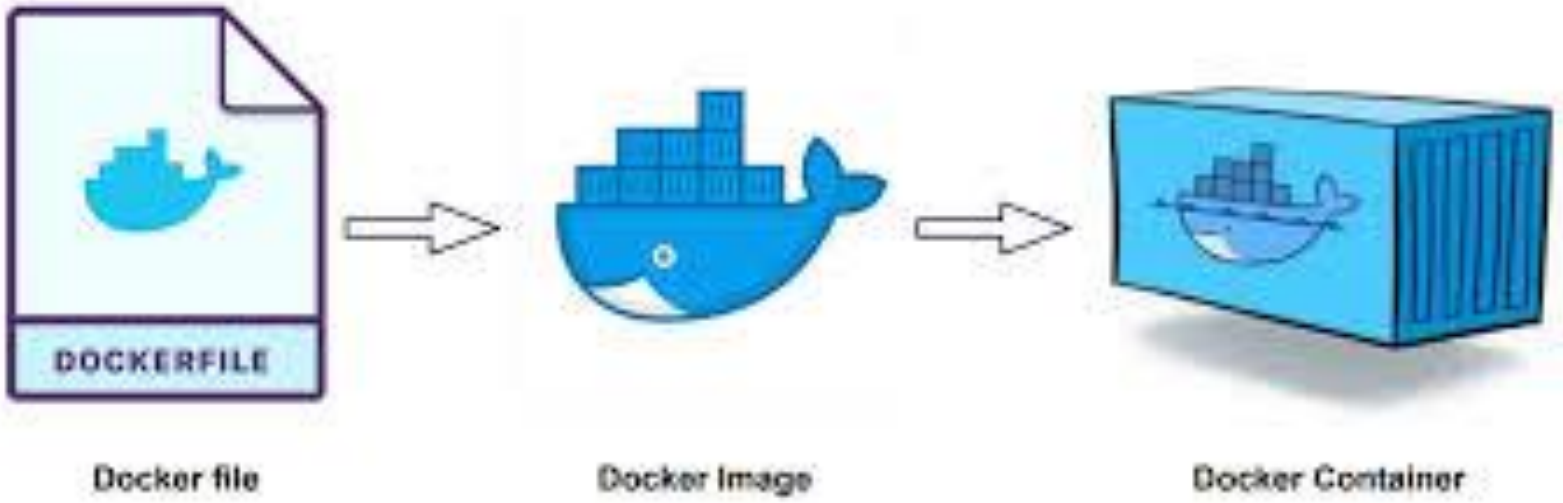
Containers

1. **Docker**
2. **Kubernetes**

Docker :

- Docker is an open-source platform that automates the deployment, scaling, and management of applications by packaging them into lightweight, portable containers.
- Containers are a form of operating system (OS) virtualization, allowing developers to bundle an application along with its dependencies, libraries, and configuration files into a single package that can run consistently across various environments.

Virtualization Technologies



Virtualization Technologies

innovate achieve lead

Docker Concepts:

Containers:

- A Docker container is a lightweight, standalone, and executable package that includes everything needed to run a piece of software, including the code, runtime, libraries, and system tools.
- Containers are isolated from each other and from the host system, ensuring that they run the same regardless of where they are deployed (e.g., on a developer's laptop, in a data center, or in the cloud).
- Unlike traditional virtual machines (VMs), containers share the host OS kernel, making them more efficient in terms of system resources.

Virtualization Technologies

Docker Concepts:

Docker Images:

- A Docker image is a read-only template that defines a container's behavior and dependencies. It contains the application code, runtime, libraries, and configurations.
- Docker images are used to create containers. Once an image is created, it can be deployed across multiple environments without modification.
- Images are built using Dockerfiles, which are simple text files that specify the steps and dependencies required to create the image.

Virtualization Technologies

innovate achieve lead

Docker Concepts:

Dockerfile:

- A **Dockerfile** is a script that contains a series of commands and instructions to build a Docker image. It typically specifies the base image (e.g., a specific version of an OS), application dependencies, and commands to run inside the container.
- Example of a Dockerfile

```
# Use an official Python runtime as a base image
FROM python:3.9-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any necessary dependencies
RUN pip install --no-cache-dir -r requirements.txt

# Run the application
CMD ["python", "app.py"]
```

Copy code

Virtualization Technologies

innovate achieve lead

Some of the docker use cases:

- **Hybrid Cloud and Multi-Cloud**:
  - Docker makes it easy to deploy applications across multiple environments, such as on-premises, public cloud, or hybrid setups. This allows for greater flexibility and better resource optimization.
- **Edge Computing:**
  - Containers are also used in edge computing to deploy lightweight applications to edge devices (e.g., IoT devices) with limited resources.

Virtualization Technologies

innovate achieve lead

Some of the docker use cases:

- **Hybrid Cloud and Multi-Cloud**:
  - Docker makes it easy to deploy applications across multiple environments, such as on-premises, public cloud, or hybrid setups. This allows for greater flexibility and better resource optimization.
- **Edge Computing:**
  - Containers are also used in edge computing to deploy lightweight applications to edge devices (e.g., IoT devices) with limited resources.

Use Cases

How do containerization and virtualization enable efficient deployment and management in edge computing environments?

Containerization

1. **Resource Efficiency**: Containers are lightweight compared to virtual machines (VMs) because they share the host system's OS kernel rather than requiring a full OS instance. This efficient use of resources is crucial for edge environments, where computing resources are often limited.
2. **Rapid Deployment and Scaling**: Containers can be started, stopped, and scaled rapidly. This agility allows for quick adaptation to changing conditions and workloads at the edge. For instance, if an edge device needs to handle a sudden increase in data processing, containers can be scaled up easily.
3. **Consistency and Portability**: Containers encapsulate applications and their dependencies, ensuring that they run consistently across different environments. This consistency is important in edge computing, where devices may vary in hardware and software.
4. **Isolation**: Containers provide process-level isolation, which helps in managing multiple applications on the same device without interference. This is useful in edge environments where devices may need to handle multiple tasks concurrently.
5. **Ease of Management**: Container orchestration platforms like Kubernetes automate the deployment, scaling, and management of containerized applications. This simplifies the operations in edge environments, which may involve numerous distributed devices.

Use Cases

Virtualization

1. **Hardware Abstraction**: Virtualization allows multiple VMs to run on a single physical machine by abstracting the hardware resources. This helps in efficiently utilizing hardware resources in edge environments where physical space and resources might be constrained.
2. **Isolation and Security**: VMs offer strong isolation between different applications and services, providing enhanced security and stability. This isolation is beneficial in edge computing, where multiple applications or tenants might be sharing the same physical hardware.
3. **Legacy Application Support**: Virtualization can run legacy operating systems and applications, which can be useful when integrating older systems with modern edge infrastructure.
4. **Flexible Resource Management**: Virtualization platforms provide tools for managing VMs, allocating resources, and balancing loads. This flexibility is essential in edge computing, where workloads and resource demands can be highly variable.
5. **Snapshot and Recovery**: Virtualization supports features like snapshots and backups, which are useful for maintaining system states and recovering from failures. This can be crucial in edge environments where physical access might be limited.

Use Cases

What are the roles of Virtualization and Containerization technologies in addressing the challenges of edge computing, and how do they enhance cloud models like IaaS, PaaS, and SaaS in edge scenarios?

Virtualization and containerization technologies play pivotal roles in addressing the challenges of edge computing and enhancing cloud models like IaaS, PaaS, and SaaS in edge scenarios.

Roles in Addressing Edge Computing Challenges

1. **Resource Efficiency**:
    1. **Containerization**: Containers are lightweight and share the host OS kernel, which makes them resource-efficient. This efficiency is crucial at the edge, where computational resources are often limited.
    2. **Virtualization**: Virtualization allows multiple VMs to run on a single physical machine, maximizing hardware utilization. This helps in optimizing the use of limited edge computing resources.
2. **Scalability and Flexibility**:
    1. **Containerization**: Containers can be quickly scaled up or down to handle varying workloads, which is important in dynamic edge environments where workloads can fluctuate.
    2. **Virtualization**: VMs can be provisioned and deprovisioned as needed, providing flexibility to adapt to changing demands at the edge.

Use Cases

3. Deployment Agility:

- **Containerization**: Containers facilitate rapid deployment and updates. This is valuable for edge computing, where rapid changes and updates to applications are often required.
- **Virtualization**: Virtual machines can also be quickly deployed, but they have a higher overhead compared to containers. Virtualization is useful when a more isolated environment is needed.

4. Consistency and Portability:

- **Containerization**: Containers package applications and their dependencies, ensuring they run consistently across different environments. This is beneficial in edge computing, where devices may have varied configurations.
- **Virtualization**: Virtualization provides a consistent environment through VMs, but the overhead is higher compared to containers.