Implementation of System Virtual Machine and OS-Level Virtual Machine using VMware and Docker

---

1. Aim

To study and implement different types of virtual machines by creating

1. a **System Virtual Machine** using VMware, and
2. an **OS-Level Virtual Machine** using Docker containers,
   and to demonstrate multiple container instances.

---

2. Software and Hardware Requirements

Software Requirements

- Host Operating System: Windows / Linux
- VMware Workstation
- Ubuntu Linux (Guest OS)
- Docker Engine
- Terminal

Hardware Requirements

- Minimum 8 GB RAM
- Intel/AMD processor with virtualization support

---

3. Theory

Virtualization is the process of creating a virtual version of computing resources such as hardware, operating systems, or storage. Virtual machines are classified into different types based on the level of abstraction they provide.

3.1 System Virtual Machine

A **System Virtual Machine** emulates a complete physical machine and allows multiple operating systems to run independently on the same hardware. Each system VM has its own kernel and operating system.

Examples: VMware, VirtualBox, Hyper-V

3.2 OS-Level Virtual Machine (Container)

An **OS-Level Virtual Machine**, also known as a container, provides an isolated execution environment that shares the host operating system kernel. Containers are lightweight, fast, and efficient compared to system virtual machines.

Examples: Docker, Podman, LXC

---

4. Implementation

4.1 Creation of System Virtual Machine using VMware

Step 1: Launch VMware Workstation

VMware Workstation was opened on the host machine.

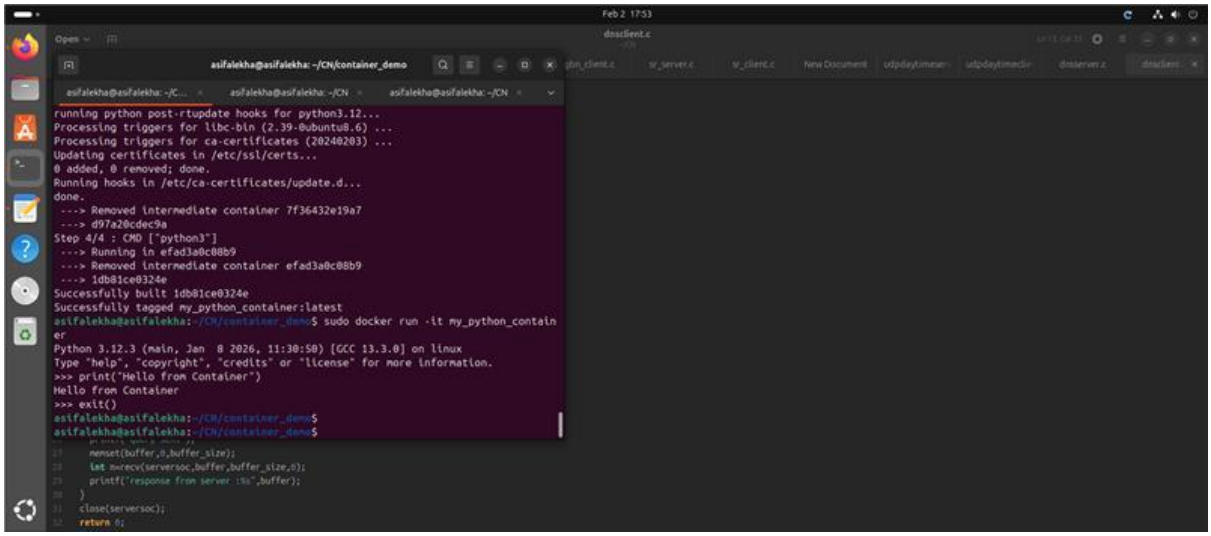
Step 2: Create a New Virtual Machine

- Selected **Create a New Virtual Machine**
- Chose **Typical configuration**
- Selected Ubuntu ISO file
- Allocated memory and disk space

Step 3: Install Ubuntu OS

Ubuntu Linux was installed inside VMware, creating a **System Virtual Machine** with its own kernel and operating system.

✅ **System Virtual Machine successfully created**



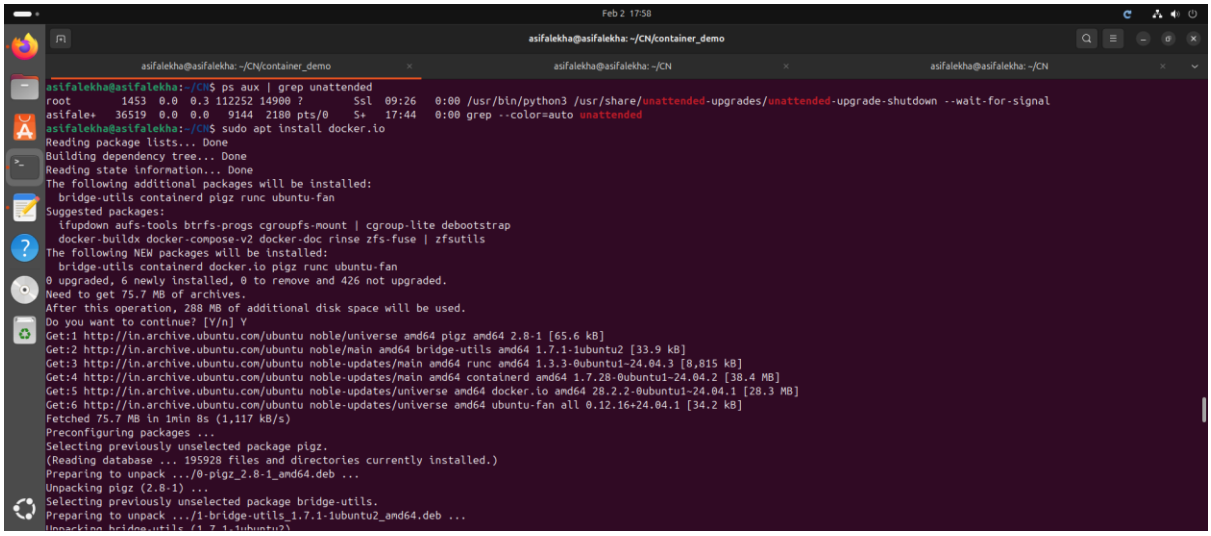
4.2 Creation of OS-Level Virtual Machine using Docker

Docker was installed inside the Ubuntu system virtual machine.

Step 1: Install Docker

sudo apt update

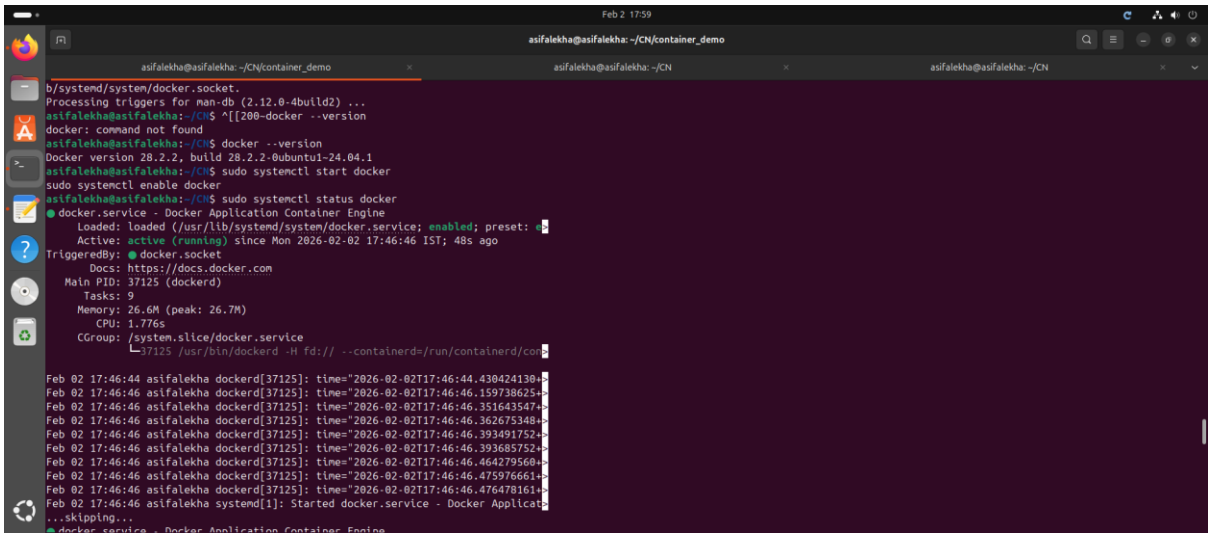
sudo apt install docker.io



Step 2: Start Docker Service

sudo systemctl start docker

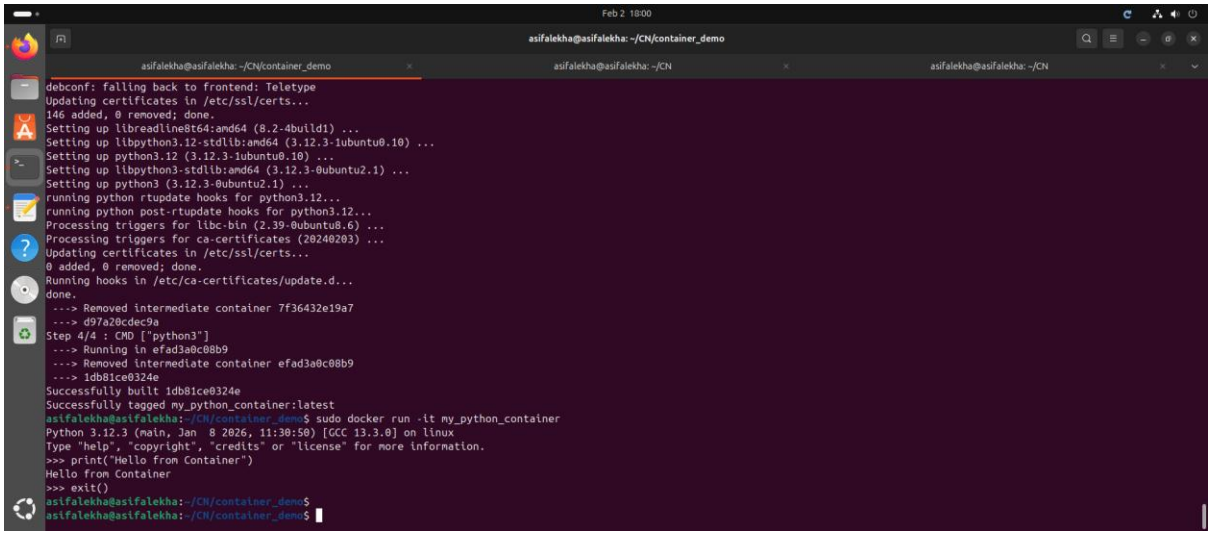
sudo systemctl enable docker



Step 3: Create OS-Level Virtual Machine (Container)

sudo docker run -it ubuntu

This command pulls the Ubuntu image and creates a container, which acts as an **OS-level virtual machine**.



4.3 Creation of Different Types of Containers

Container Type 1: Ubuntu Base Container

sudo docker run -it ubuntu

- Provides a minimal Ubuntu environment
- Used for basic OS-level operations

📸 *Screenshot 7: Ubuntu Base Container*

Container Type 2: Custom Python Container

Dockerfile Creation

FROM ubuntu

RUN apt update

RUN apt install -y python3

CMD ["python3"]

Build Container Image

sudo docker build -t python_container .

Run Container

sudo docker run -it python_container

This container supports Python execution inside an isolated environment.

📸 *Screenshot 8: Python Container Execution*

---

5. Types of Virtual Machines Implemented

Type	Technology Used	Description
System Virtual Machine	VMware	Full OS with separate kernel
OS-Level Virtual Machine	Docker	Lightweight container sharing host kernel

---

6. Observations

- System virtual machines consume more resources but provide complete OS isolation.
- OS-level virtual machines are lightweight and start quickly.
- Docker containers are suitable for distributed systems due to scalability and portability.

---

7. Result

Thus, a **System Virtual Machine** using VMware and an **OS-Level Virtual Machine** using Docker containers were successfully created and executed. Multiple containers were implemented to demonstrate container-based virtualization.