

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.

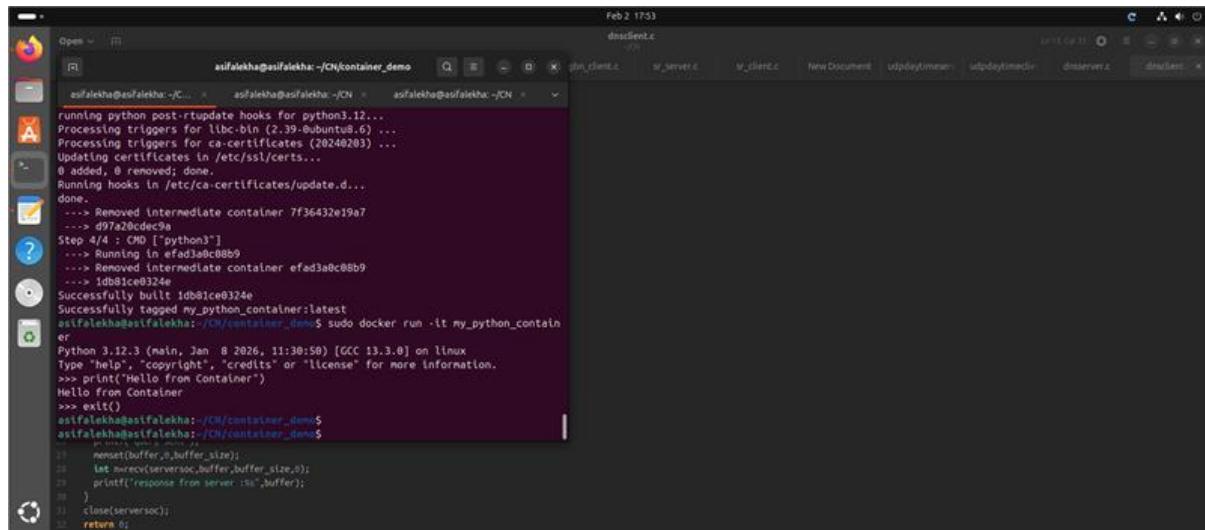
Step 2: Create a New Virtual Machine

- 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



The screenshot shows a terminal window with several tabs open. The active tab displays the command `sudo docker run -it my_python_container` followed by a Python script. The script prints "Hello from Container" and exits. The terminal also shows the output of `apt-get update` and `apt-get upgrade` commands, indicating the system was updated before the container was built.

```
Feb 2 17:53
asifalekha@asifalekha:~/CN/container_demo$ sudo docker run -it my_python_container
Python 3.12.3 (main, Jan  8 2026, 11:30:58) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello from Container')
Hello from Container
>>> exit()
asifalekha@asifalekha:~/CN/container_demo$ asifalekha@asifalekha:~/CN/container_demo$ python3
Python 3.12.3 (main, Jan  8 2026, 11:30:58) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello from Container')
Hello from Container
>>> exit()
asifalekha@asifalekha:~/CN/container_demo$ asifalekha@asifalekha:~/CN/container_demo$
```

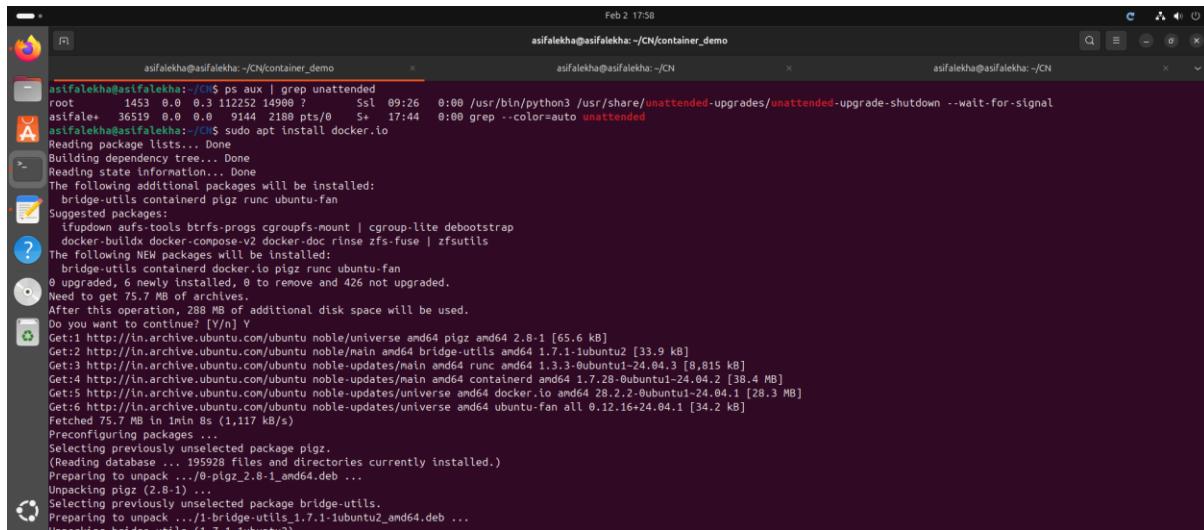
4.2 Creation of OS-Level Virtual Machine using Docker

Docker was installed inside the Ubuntu system virtual machine.

Step 1: Install Docker

```
sudo apt update
```

```
sudo apt install docker.io
```



The screenshot shows a terminal window with three tabs. The left tab runs a command to list processes (ps aux | grep unattended). The middle tab runs 'sudo apt install docker.io'. The right tab runs a command to search for 'unattended' packages. The output of the middle command shows the installation process, including dependency resolution and package download details from the Ubuntu archive.

```
asifalekha@asifalekha:~/CN/container_demo
```

```
root 1453 0.0 0.3 112252 14980 ? Ssl 09:26 0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
```

```
asifalekha+ 36519 0.0 0.0 9144 2180 pts/0 S+ 17:44 0:00 grep --color=auto unattended
```

```
asifalekha@asifalekha:~/CN
```

```
Reading package lists... Done
```

```
Building dependency tree... Done
```

```
Reading state information... Done
```

```
The following additional packages will be installed:
```

```
bridge-utils containerd runc ubuntu-fan
```

```
Suggested packages:
```

```
ifupdown aufs-tools brtrfs-progs cgroups-mount | cgroup-lite debootstrap
```

```
docker-buildx docker-compose-v2 docker-doc rinse zfs-fuse | zfsutils
```

```
The following NEW packages will be installed:
```

```
bridge-utils containerd docker.io runc ubuntu-fan
```

```
0 upgraded, 6 newly installed, 0 to remove and 426 not upgraded.
```

```
Need to get 75.7 MB of archives.
```

```
After this operation, 288 MB of additional disk space will be used.
```

```
Do you want to continue? [Y/n] Y
```

```
Get:1 http://in.archive.ubuntu.com/ubuntu noble/universe amd64 pigz amd64 2.8-1 [65.6 kB]
```

```
Get:2 http://in.archive.ubuntu.com/ubuntu noble/main amd64 bridge-utils amd64 1.7.1-ubuntu2 [33.9 kB]
```

```
Get:3 http://in.archive.ubuntu.com/ubuntu noble-updates/main amd64 runc amd64 1.3.3-0ubuntu1-24.04.3 [8,815 kB]
```

```
Get:4 http://in.archive.ubuntu.com/ubuntu noble-updates/main amd64 containerd amd64 1.7.28-0ubuntu1-24.04.2 [38.4 MB]
```

```
Get:5 http://in.archive.ubuntu.com/ubuntu noble-updates/universe amd64 docker.io amd64 28.2.2-0ubuntu1-24.04.1 [28.3 MB]
```

```
Get:6 http://in.archive.ubuntu.com/ubuntu noble-updates/universe amd64 ubuntu-fan all 0.12.16+24.04.1 [34.2 kB]
```

```
Fetched 75.7 MB in 1min 8s (1,117 kB/s)
```

```
Preconfiguring packages ...
```

```
Selecting previously unselected package pigz.
```

```
(Reading database ... 195928 files and directories currently installed.)
```

```
Preparing to unpack .../0-pigz_2.8-1_amd64.deb ...
```

```
Unpacking pigz (2.8-1) ...
```

```
Selecting previously unselected package bridge-utils.
```

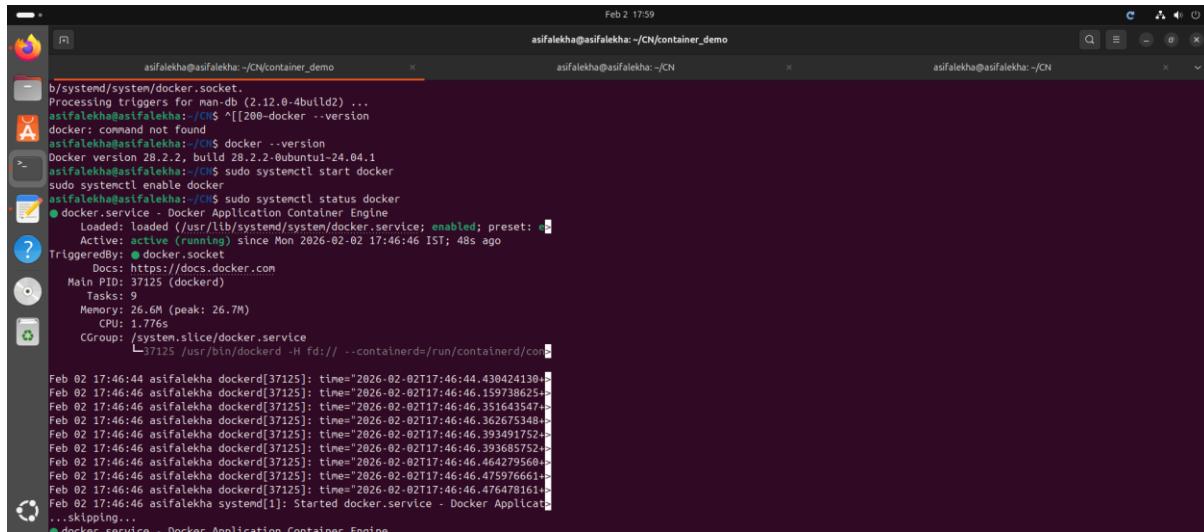
```
Preparing to unpack .../1-bridge-utils_1.7.1-ubuntu2_amd64.deb ...
```

```
Unpacking bridge-utils (1.7.1-ubuntu2) ...
```

Step 2: Start Docker Service

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```



The screenshot shows a terminal window with three tabs. The left tab runs 'ls /system/docker.socket'. The middle tab runs 'docker --version'. The right tab runs 'sudo systemctl status docker'. The output of the right tab shows the Docker service is active and running.

```
asifalekha@asifalekha:~/CN/container_demo
```

```
b/systemd/system/docker.socket
```

```
Processing triggers for man-db (2.12.0-4build2) ...
```

```
asifalekha@asifalekha:~/CN
```

```
docker: command not found
```

```
asifalekha@asifalekha:~/CN
```

```
asifalekha@asifalekha:~/CN$ docker --version
```

```
Docker version 28.2.2, build 28.2.2-0ubuntu1-24.04.1
```

```
asifalekha@asifalekha:~/CN$ sudo systemctl start docker
```

```
sudo systemctl enable docker
```

```
asifalekha@asifalekha:~/CN
```

```
● docker.service - Docker Application Container Engine
```

```
  Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
```

```
  Active: active (running) since Mon 2026-02-02 17:46:46 IST; 48s ago
```

```
    TriggeredBy: ● docker.socket
```

```
      Docs: https://docs.docker.com
```

```
      Main PID: 37125 (dockerd)
```

```
        Tasks: 9
```

```
         Memory: 26.6M (peak: 26.7M)
```

```
          CPU: 1.776s
```

```
          CGroup: /system.slice/docker.service
```

```
              └─37125 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

```
Feb 02 17:46:44 asifalekha dockerd[37125]: time="2026-02-02T17:46:44.430424130Z"
```

```
Feb 02 17:46:46 asifalekha dockerd[37125]: time="2026-02-02T17:46:46.159738625Z"
```

```
Feb 02 17:46:46 asifalekha dockerd[37125]: time="2026-02-02T17:46:46.351643547Z"
```

```
Feb 02 17:46:46 asifalekha dockerd[37125]: time="2026-02-02T17:46:46.362675348Z"
```

```
Feb 02 17:46:46 asifalekha dockerd[37125]: time="2026-02-02T17:46:46.393491752Z"
```

```
Feb 02 17:46:46 asifalekha dockerd[37125]: time="2026-02-02T17:46:46.393685752Z"
```

```
Feb 02 17:46:46 asifalekha dockerd[37125]: time="2026-02-02T17:46:46.464279560Z"
```

```
Feb 02 17:46:46 asifalekha dockerd[37125]: time="2026-02-02T17:46:46.475976661Z"
```

```
Feb 02 17:46:46 asifalekha dockerd[37125]: time="2026-02-02T17:46:46.476478161Z"
```

```
Feb 02 17:46:46 asifalekha systemd[1]: Started docker.service - Docker Application Container Engine
```

```
... skipping...
```

```
● docker.service - Docker Application Container Engine
```

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**.

The screenshot shows a terminal window with three tabs. The left tab shows the output of a Dockerfile build process, including dependency installation and a successful tag. The middle tab shows the container running a Python script that prints "Hello from Container". The right tab shows the command to run the container.

```
debconf: Falling back to frontend: Teletype
Updating certificates in /etc/ssl/certs...
146 added, 0 removed; done.
Setting up libreadline8t64:amd64 (8.2-4build1) ...
Setting up libpython3.12-stdlib:amd64 (3.12.3-1ubuntu0.10) ...
Setting up python3.12 (3.12.3-1ubuntu0.10) ...
Setting up libpython3.12-stdlib:amd64 (3.12.3-0ubuntu2.1) ...
Setting up python3 (3.12.3-0ubuntu2.1) ...
running python rupdate hooks for python3.12...
running python post-rupdate hooks for python3.12...
Processing triggers for libc-bin (2.39-0ubuntu0.6) ...
Processing triggers for ca-certificates (20240203) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
--> Removed intermediate container 7f36432e19a7
--> d97a0cdec9a
Step 4/4 : CMD ["python3"]
--> Running in efad3a0c08b9
--> Removed intermediate container efad3a0c08b9
--> idb81ce0324e
Successfully built idb81ce0324e
Successfully tagged my_python_container:latest
asifalekha@asifalekha:~/CN/container_demo$ sudo docker run -it my_python_container
Python 3.12.3 (main, Jan 8 2026, 11:30:50) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello from Container")
Hello from Container
>>> exit()
asifalekha@asifalekha:~/CN/container_demo$ asifalekha@asifalekha:~/CN/container_demo$
```

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.