Linux Virtualization

Part 1: Introduction to Virtualization Concepts

Virtualization– The process of creating virtual instances of hardware or software resources.

Hypervisor– Software that enables virtualization by managing virtual machines (VMs).

Virtual Machines (VMs) – Fully isolated computing environments that run on virtualized hardware.

Containers – Lightweight virtualization at the OS level that allows multiple applications to run in isolated spaces.

Main differences between VMs and Containers

Virtual Machines	Containers
Includes full OS, virtual hardware	Shares host OS kernel
Requires more resources	Lightweight
Strong isolation, each VM runs separately	Process-level isolation
Slightly slower due to full OS overhead	Faster startup

Part 2: Working with Multipass:

Following the Source Guide **Source Used:** Canonical Install Guide

Steps:

To install multipass:

```
snap install multipass
```

Launching the default Mulitpass ubuntu instance:

```
multipass launch
```

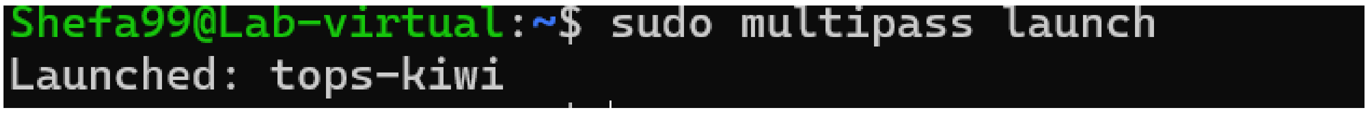
I get a `launch failed` error and telling me to run `multipass authenticate` command first.

But I used:

```
sudo multipass launch
```

Reason: When running Multipass commands, authentication happens automatically in the background.

Result:

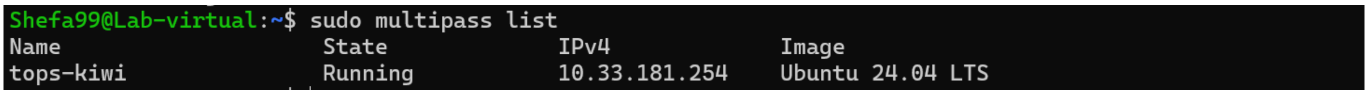


```
Shefa99@Lab-virtual:~$ sudo multipass launch
Launched: tops-kiwi
```

Listing instances: Command:

```
sudo multipass list
```

Result:

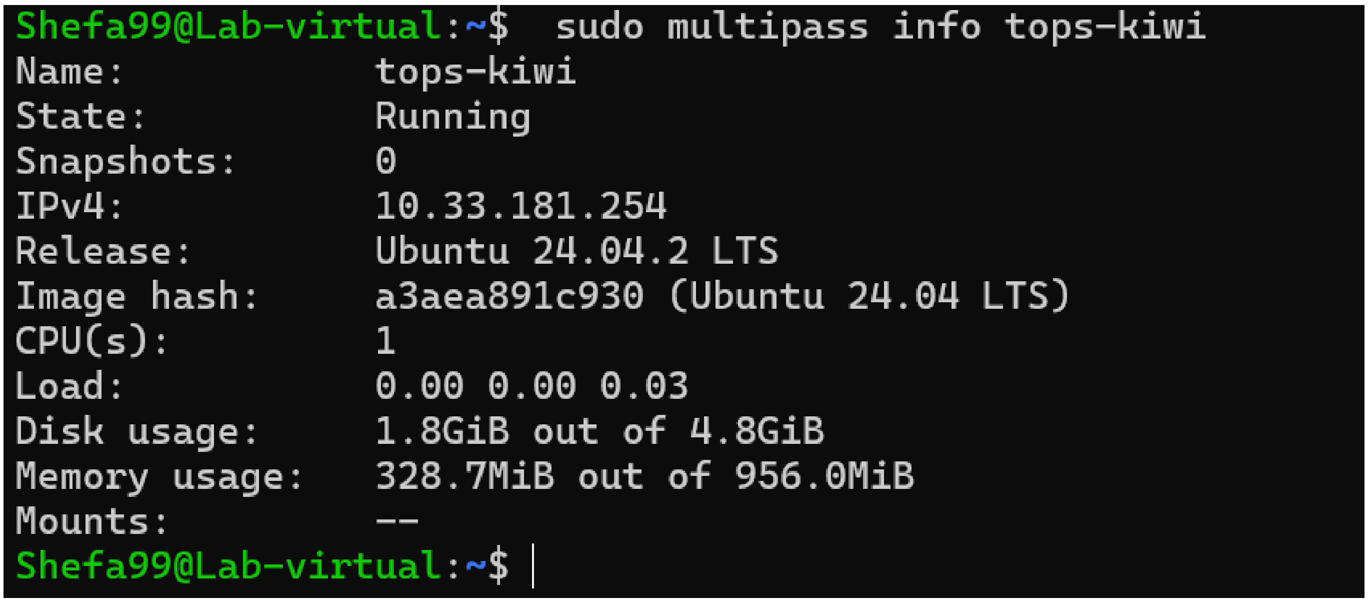


```
Shefa99@Lab-virtual:~$ sudo multipass list
Name                    State             IPv4             Image
tops-kiwi               Running           10.33.181.254    Ubuntu 24.04 LTS
```

Multipass instance info: Command:

```
sudo multipass info tops-kiwi
```

Result:

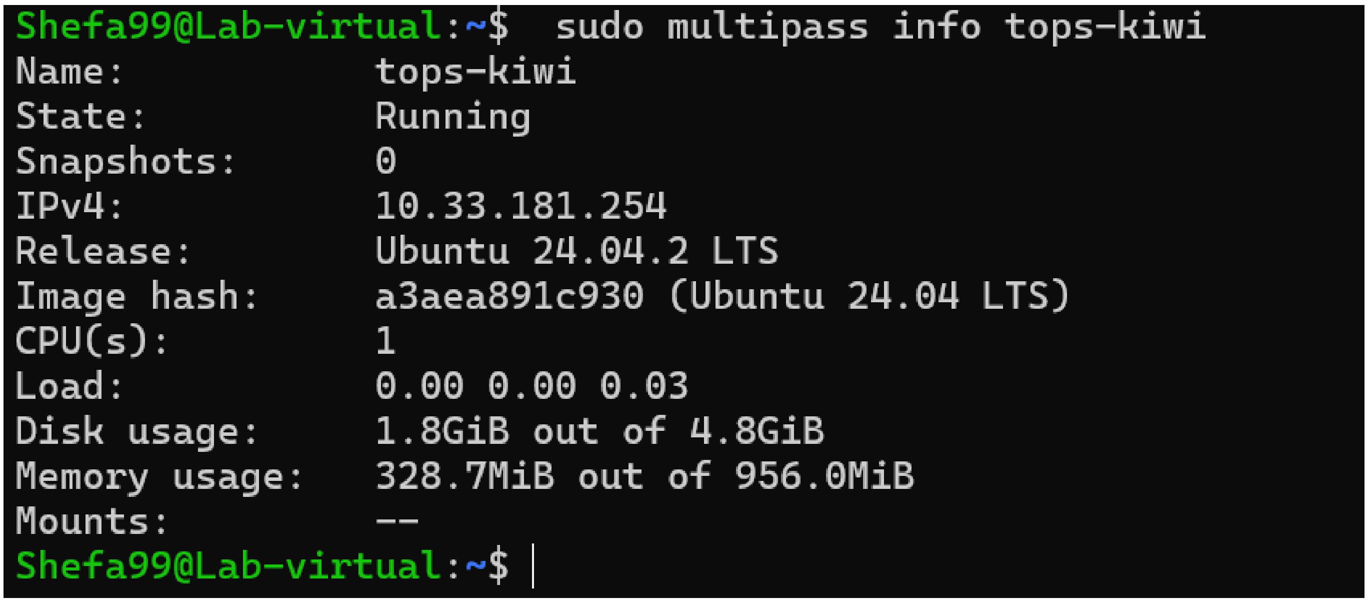


```
Shefa99@Lab-virtual:~$  sudo multipass info tops-kiwi
Name:           tops-kiwi
State:          Running
Snapshots:      0
IPv4:           10.33.181.254
Release:        Ubuntu 24.04.2 LTS
Image hash:     a3aea891c930 (Ubuntu 24.04 LTS)
CPU(s):         1
Load:           0.00 0.00 0.03
Disk usage:     1.8GiB out of 4.8GiB
Memory usage:   328.7MiB out of 956.0MiB
Mounts:         --
Shefa99@Lab-virtual:~$
```

Multipass Shell access: Command:

```
sudo multipass shell tops-kiwi
```

Result:



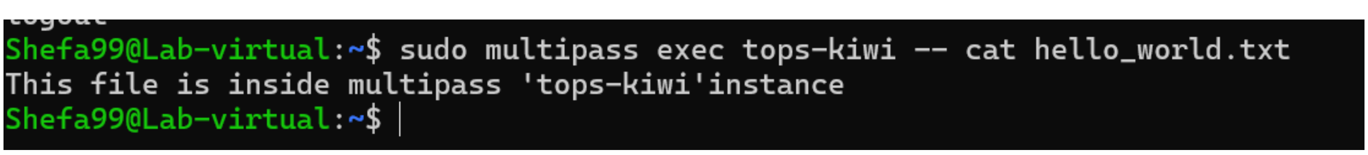
```
Shefa99@Lab-virtual:~$  sudo multipass info tops-kiwi
Name:           tops-kiwi
State:          Running
Snapshots:      0
IPv4:           10.33.181.254
Release:        Ubuntu 24.04.2 LTS
Image hash:     a3aea891c930 (Ubuntu 24.04 LTS)
CPU(s):         1
Load:           0.00 0.00 0.03
Disk usage:     1.8GiB out of 4.8GiB
Memory usage:   328.7MiB out of 956.0MiB
Mounts:         --
Shefa99@Lab-virtual:~$
```

I created a file in the instance home directory and named it 'hello_world.txt'. Now trying to read the file outside the instance using the cat command.

Command:

```
sudo multipass exec tops-kiwi -- cat hello_world.txt
```

Result:



```
Shefa99@Lab-virtual:~$ sudo multipass exec tops-kiwi -- cat hello_world.txt
This file is inside multipass 'tops-kiwi'instance
Shefa99@Lab-virtual:~$
```

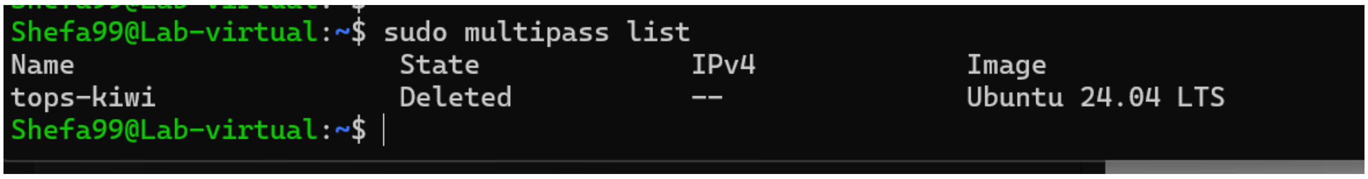
Stopping the instace:

```
sudo multipass stop tops-kiwi
```

Delete the instance:

```
sudo multipass delete tops-kiwi
```

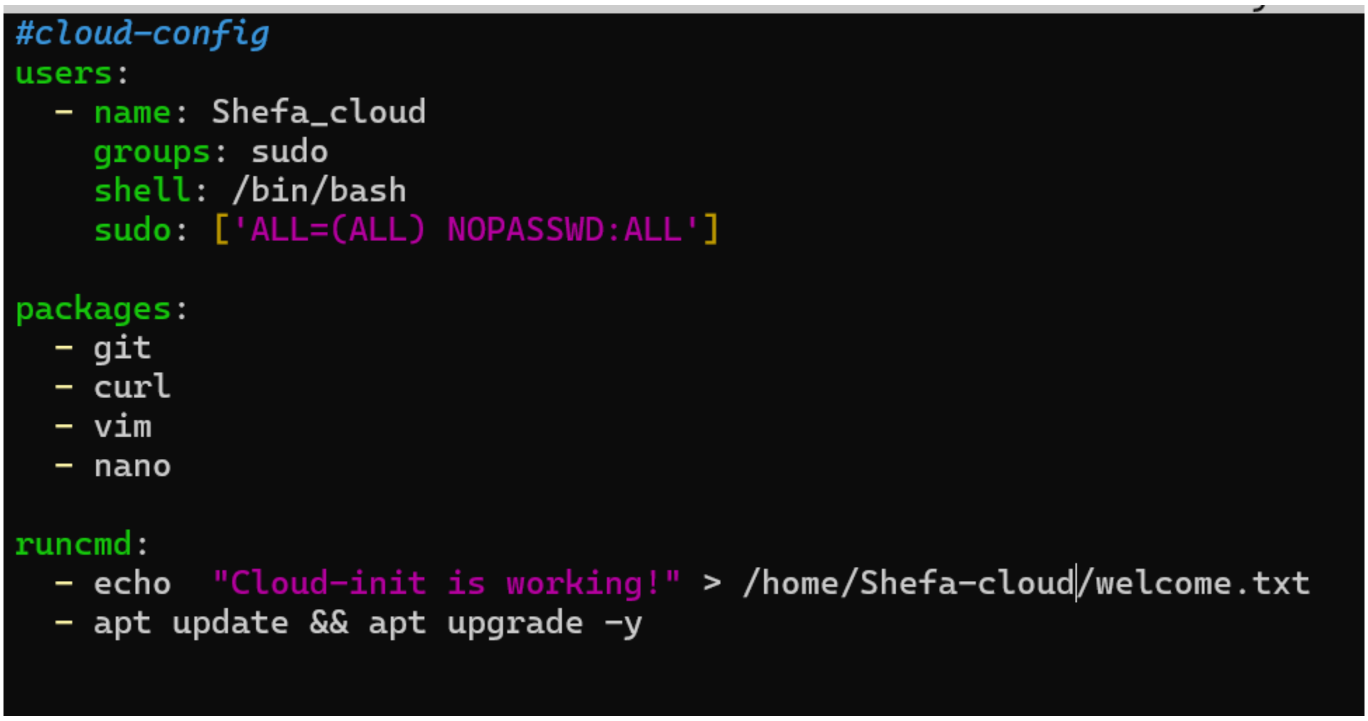
```
sudo multipass list
```



```
Shefa99@Lab-virtual:~$ sudo multipass list
Name                    State             IPv4              Image
tops-kiwi               Deleted           --                Ubuntu 24.04 LTS
Shefa99@Lab-virtual:~$
```

Cloud-init Experiment:

created the **"cloud-init.yml"** file.

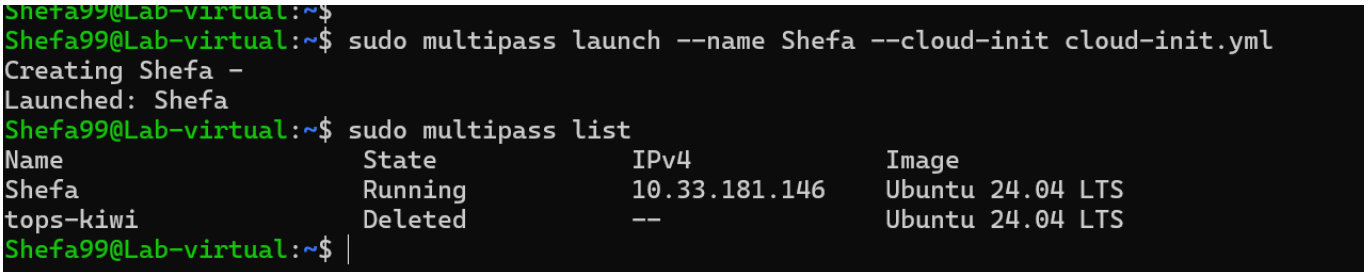


```
#cloud-config
users:
  - name: Shefa_cloud
    groups: sudo
    shell: /bin/bash
    sudo: ['ALL=(ALL) NOPASSWD:ALL']

packages:
  - git
  - curl
  - vim
  - nano

runcmd:
  - echo  "Cloud-init is working!" > /home/Shefa-cloud/welcome.txt
  - apt update && apt upgrade -y
```

The config file will install some basic packeges and after successfully starting the instance it will return "Cloud-init is working!" from the welcome.txt file which will create after running the first time.

Commnad:

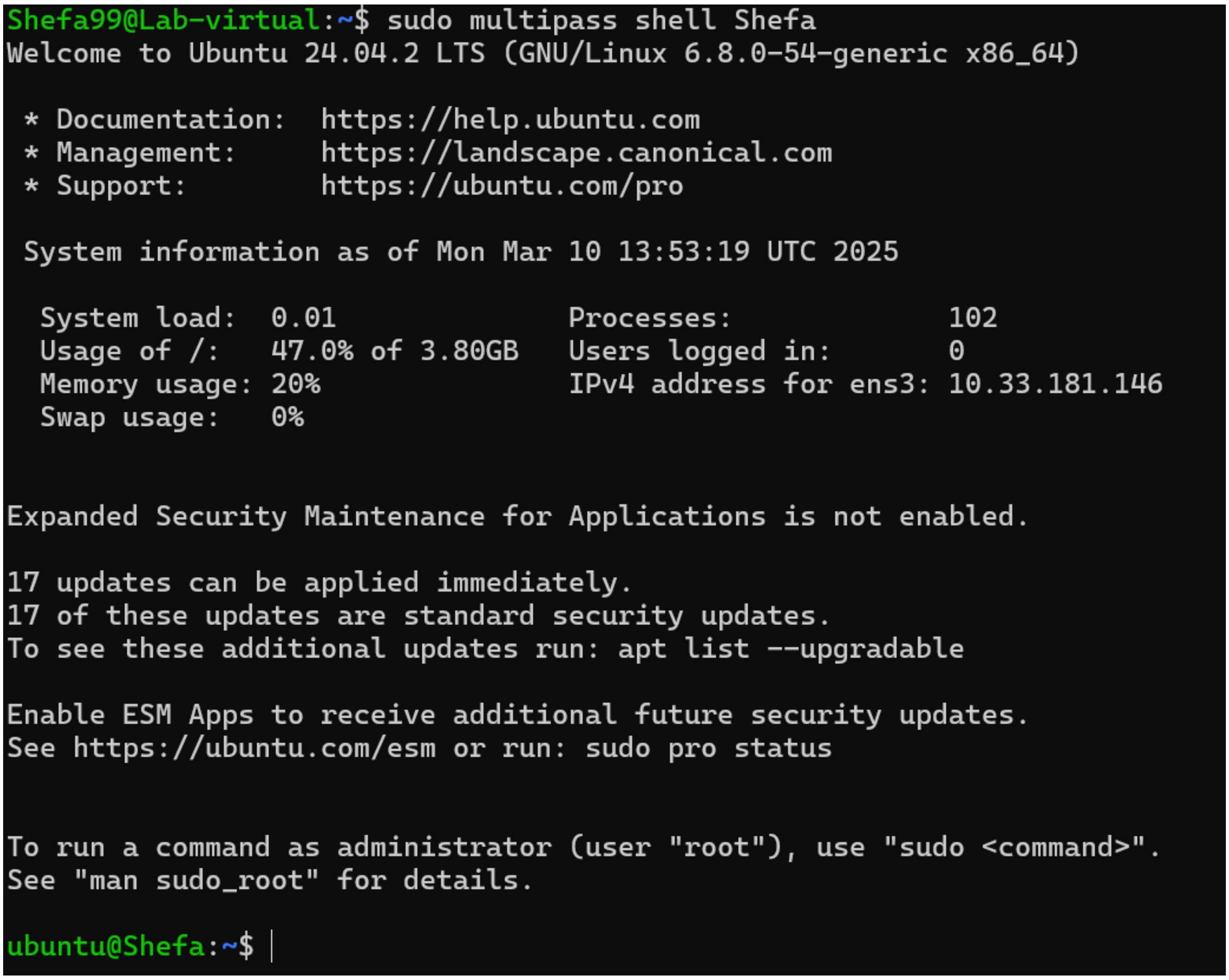
```
sudo multipass launch --name cloud-init --cloud-init cloud-init.ym
```



```
Shefa99@Lab-virtual:~$
Shefa99@Lab-virtual:~$ sudo multipass launch --name Shefa --cloud-init cloud-init.yml
Creating Shefa -
Launched: Shefa
Shefa99@Lab-virtual:~$ sudo multipass list
Name                    State             IPv4              Image
Shefa                   Running           10.33.181.146     Ubuntu 24.04 LTS
tops-kiwi               Deleted           --                Ubuntu 24.04 LTS
Shefa99@Lab-virtual:~$
```

Then accessed shell using

```
sudo multipass shell cloud-init
```

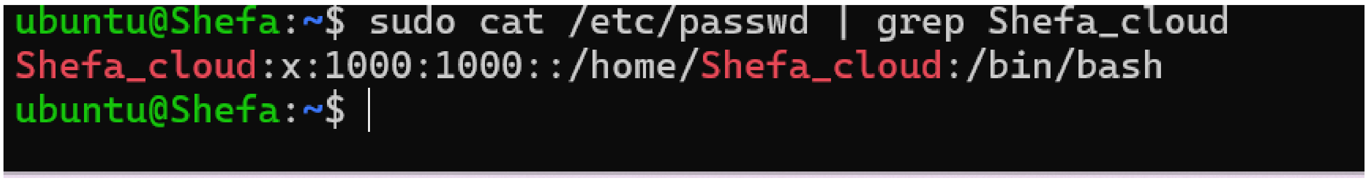


```
Shefa99@Lab-virtual:~$ sudo multipass shell Shefa
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

 System information as of Mon Mar 10 13:53:19 UTC 2025

  System load:  0.01              Processes:             102
  Usage of /:   47.0% of 3.80GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for ens3: 10.33.181.146
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

17 updates can be applied immediately.
17 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@Shefa:~$
```

Now checking if the user Shefa_cloud exist. command:

```
cat /etc/passwd | grep Shefa_cloud
```

Result:



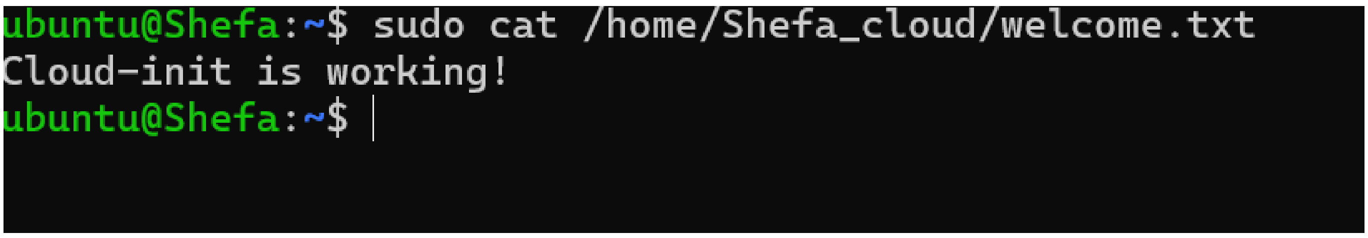
```
ubuntu@Shefa:~$ sudo cat /etc/passwd | grep Shefa_cloud
Shefa_cloud:x:1000:1000::/home/Shefa_cloud:/bin/bash
ubuntu@Shefa:~$
```

welcome.txt

```
sudo cat /home/Shefa_cloud/welcome.txt
```

Result:

Using sudo because the file was created by root and was accessable by Shefa_cloud.



```
ubuntu@Shefa:~$ sudo cat /home/Shefa_cloud/welcome.txt
Cloud-init is working!
ubuntu@Shefa:~$
```

created a folder in the host machine named "host_machine" and created another folder in instance called "shared_folder". then mounted the folder on the host machine using:

```
sudo multipass mount ~/host_machine cloud-init:/home/ubuntu/shared_folder
```

Creating a test file in the host machine.

```
echo "Hello from the host!" > ~/shared-folder/hostfile.txt
```

Study

LXD is a next-generation system container manager that provides a user-friendly experience for managing Linux containers. It extends LXC, offering a robust API, CLI tools, and the ability to manage both containers and virtual machines.

Key Features of LXD Image-based: LXD uses prebuilt images for various Linux distributions. Security: Containers run under an unprivileged user, increasing isolation. Scalability: LXD supports clustering, making it efficient for managing multiple containers. Live Migration: Containers can be moved between hosts. Setup To install and enable LXD on Ubuntu 24.04, follow these steps:

```
sudo apt update && sudo apt install -y lxd
sudo lxd init
```

During initialization, LXD will prompt for storage, networking, and security configurations.

Below are essential LXD commands to manage containers:

```
lxc launch ubuntu:24.04 my-container
lxc list
lxc exec my-container -- bash
lxc stop my-container
lxc delete my-container
```

Challenges Faced Storage Backend Configuration: The setup required choosing a storage backend (ZFS, LVM, or directory). Opted for ZFS for better performance. Network Bridge Setup: Had to manually create a bridge to allow internet access to containers. How to Stick Apps with Docker

Installation

To install Docker on Ubuntu 24.04:

```
sudo apt update
sudo apt install -y docker.io
sudo systemctl enable --now docker
```

Verify installation:

```
docker --version
```

Basic Concepts

Images: Read-only templates used to create containers. Containers: Instances of Docker images. Dockerfile: A script defining how to build an image. Experiment To test Docker, I created a simple containerized Nginx server:

```
docker run -d -p 8080:80 --name my-nginx nginx
```

Accessing http://localhost:8080 confirmed it was running successfully.

Challenges Faced Permission Issues: Initially, needed sudo for Docker commands. Solved by adding the user to the docker group.

```
sudo usermod -aG docker $USER
```

Port Conflicts: Another service was using port 80, so I mapped to 8080 instead. Snaps for Self-Contained Applications Research Snaps are self-contained application packages that include dependencies, allowing for easy deployment across Linux distributions.

Benefits of Snaps Automatic Updates: Ensures applications stay updated. Isolation: Snaps run in a sandboxed environment. Cross-Distro Compatibility: Works across different Linux distributions. Experiment: Creating a Snap Installed Snapcraft:

```
sudo snap install snapcraft --classic
```

Created a basic snap package:

```
mkdir my-snap
cd my-snap
```

```
snapcraft init
```

Modified snapcraft.yaml to build a simple application. Then, built and installed the snap:

```
snapcraft
sudo snap install my-snap_*.snap --dangerous
```

Verified installation:

```
snap list | grep my-snap
```

Challenges Faced Dependency Issues: Had to install missing dependencies manually. Permissions: Required --dangerous flag to install locally built snaps.