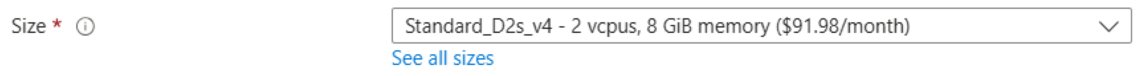
Assignment 7 - Virtualization

Machine specs on Azure:

1. **For use of virtualization in Linux its better to use a VM with higher performance.**
2. **So we create a VM with high performance,**



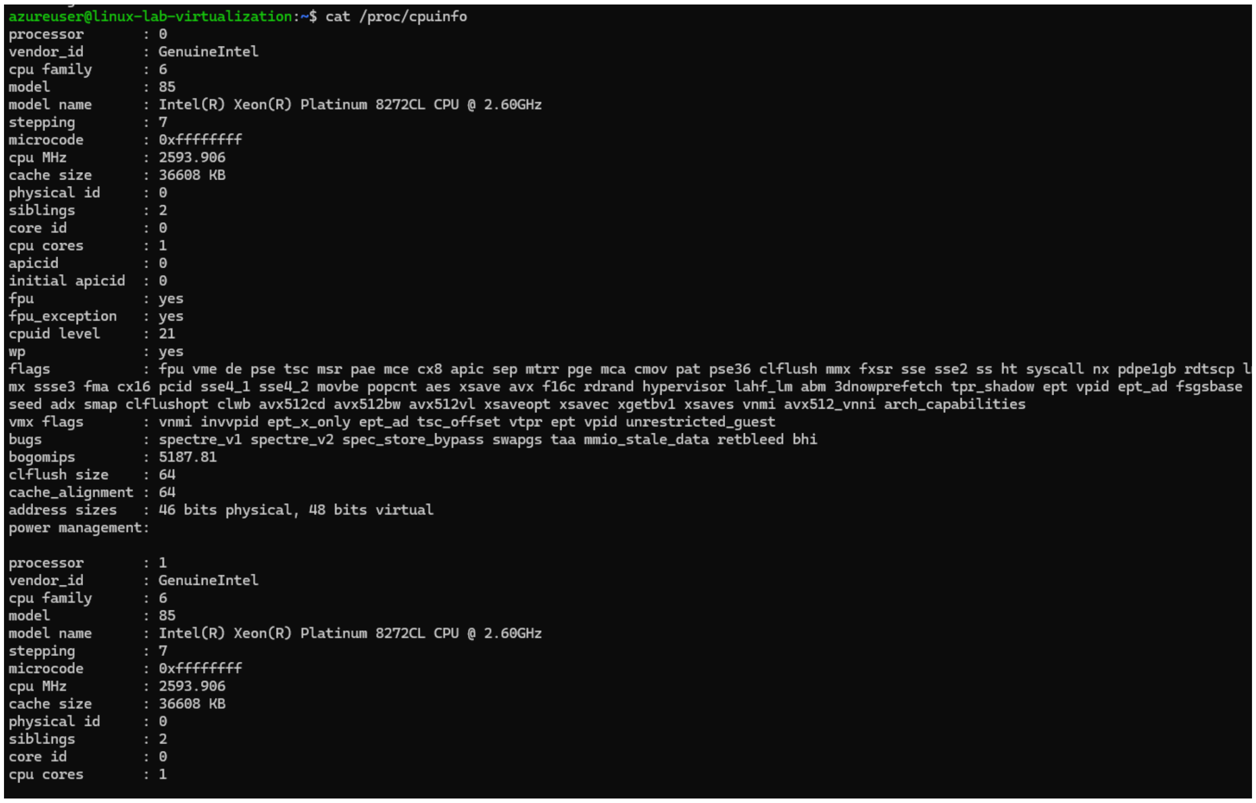
Size * ⓘ

Standard_D2s_v4 - 2 vcpus, 8 GiB memory ($91.98/month)

See all sizes

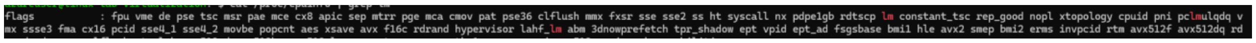
Checking virtualization support on machine:

1. **Once the VM is created we can check the virtualization support by cpuinfo, cat /proc/cpuinfo**



```
azureuser@linux-lab-virtualization:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel(R) Xeon(R) Platinum 8272CL CPU @ 2.60GHz
stepping        : 7
microcode       : 0xffffffff
cpu MHz         : 2593.906
cache size      : 36608 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 21
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp l
mx ssse3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow ept vpid ept_ad fsgsbase
seed adx smap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves vnmi avx512_vnni arch_capabilities
vmx flags       : vnmi invvpid ept_x_only ept_ad tsc_offset vtpr ept vpid unrestricted_guest
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs taa mmio_stale_data retbleed bhi
bogomips        : 5187.81
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel(R) Xeon(R) Platinum 8272CL CPU @ 2.60GHz
stepping        : 7
microcode       : 0xffffffff
cpu MHz         : 2593.906
cache size      : 36608 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
```

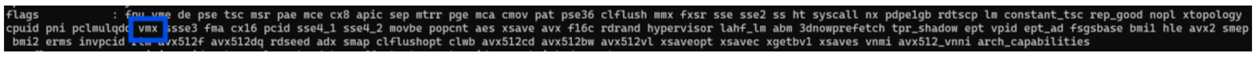
2. **Am I using 64 bit CPU/system [x86_64/AMD64/Intel64]? IN flags check for lm**



```
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology cpuid pni pclmulqdq v
mx ssse3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow ept vpid ept_ad fsgsbase bmi1 hle avx2 smep bmi2 erms invpcid rtm avx512f avx512dq rd
```

3. **Do I have hardware virtualization support?**
   - **vmx – Intel VT-x, virtualization support enabled in BIOS.**
   - **svm – AMD SVM, virtualization enabled in BIOS.**

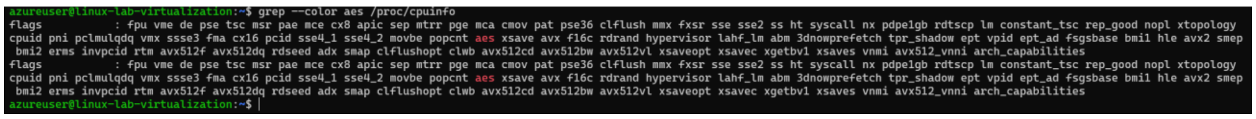
- Since we are using intel based system we only have VMX



```
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology
cpuid pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow ept vpid ept_ad fsgsbase bmi1 hle avx2 smep
 bmi2 erms invpcid rtm avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves vnmi avx512_vnni arch_capabilities
```

4.Do I have hardware AES/AES-NI advanced encryption support?

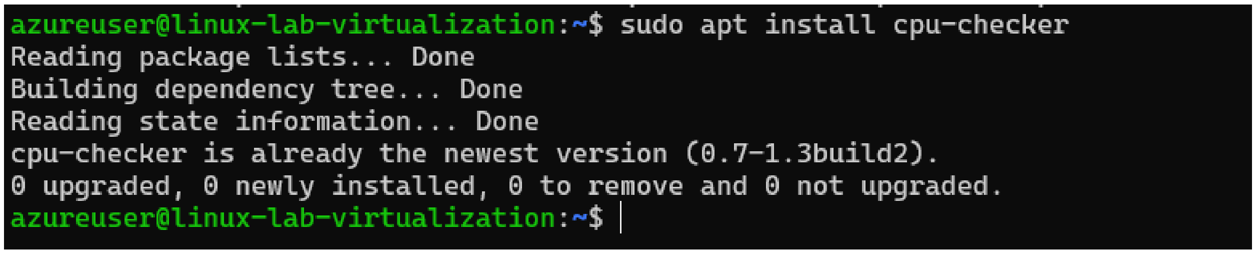
aes – Applications performing encryption and decryption using the Advanced Encryption Standard on Intel and AMD cpus.



```
azureuser@linux-lab-virtualization:~$ grep --color aes /proc/cpuinfo
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology
cpuid pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow ept vpid ept_ad fsgsbase bmi1 hle avx2 smep
 bmi2 erms invpcid rtm avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves vnmi avx512_vnni arch_capabilities
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology
cpuid pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow ept vpid ept_ad fsgsbase bmi1 hle avx2 smep
 bmi2 erms invpcid rtm avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves vnmi avx512_vnni arch_capabilities
azureuser@linux-lab-virtualization:~$
```

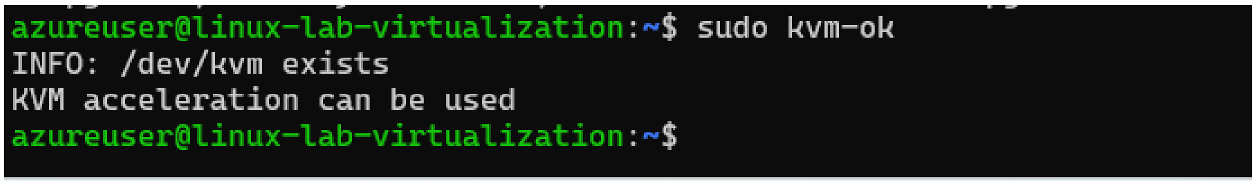
# Install kvm-ok on a Debian/Ubuntu

1. **sudo apt install cpu-checker**



```
azureuser@linux-lab-virtualization:~$ sudo apt install cpu-checker
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
cpu-checker is already the newest version (0.7-1.3build2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
azureuser@linux-lab-virtualization:~$
```

2. **sudo kvm-ok**



```
azureuser@linux-lab-virtualization:~$ sudo kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
azureuser@linux-lab-virtualization:~$
```

Linux Virtualization Exercise

All exercises should be attempted on new Ubuntu 24.04 Linux installation, that supports nested virtualization.

Part 1: Introduction to virtualization concepts (30 minutes)

Research: Using sources (Sources linked at the end of the task), learn about the following concepts:

Virtualization

Virtualization is a technology that allows multiple operating systems or applications to run concurrently on a single physical machine by abstracting the underlying

hardware resources. This is achieved through the use of hypervisors, which are software layers that create and manage virtual machines (VMs).

Hypervisor

A hypervisor is software or firmware that creates and manages virtual machines (VMs) on a physical host. It allows multiple VMs to run on a single physical machine by abstracting and allocating hardware resources like CPU, memory, and storage.

Virtual machines (VM)

A virtual machine (VM) is a software-based emulation of a physical computer that runs an operating system (OS) and applications just like a real machine. VMs are created and managed by a hypervisor, which allocates CPU, memory, storage, and other resources from the physical hardware to the VM.

Containers

A container is a lightweight, portable software unit that packages an application and its dependencies together so it can run consistently across different environments. Unlike virtual machines (VMs), containers share the host operating system's kernel, making them more efficient and faster.

The main differences between VMs and tanks

Summary: Summary in a text file briefly summarize the core differences between VMs and containers. Focus on their architecture, resource utilization, and insulation levels.

Feature Virtual	Machines (VMs)	Containers
Architecture	Each VM has a full OS, including a guest OS and virtualized hardware.	Containers share the host OS kernel and run as isolated processes.
Resource Usage	More resource-intensive due to separate OS instances.	Lightweight, as they share the OS kernel.
Startup Time	Slower, since the OS needs to boot.	Faster, since they don't require OS booting.

Feature Virtual	Machines (VMs)	Containers
Isolation	Stronger isolation (separate OS for each VM).	Weaker isolation (process-level separation).
Portability	Less portable, depends on the hypervisor.	Highly portable across different environments.
Scalability	Requires more resources to scale.	Easily scalable due to lightweight nature.
Use Case	Best for running multiple OS environments or legacy applications.	Best for microservices, cloud applications, and fast deployments.

Part 2: Working with Multipass (1-2 hours)

Multipass is a command-line tool that allows you to quickly create and manage Ubuntu virtual machines.

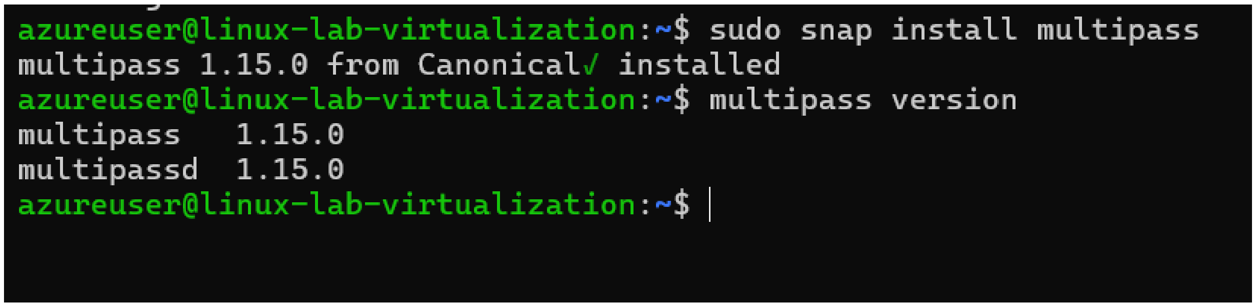
Installation: Follow the instructions on the Multipass website to install Multipass on your system. More information can be found in the source.

1. Start VM and install Multipass.

sudo snap install multipass

2. Verify the installation

multipass version



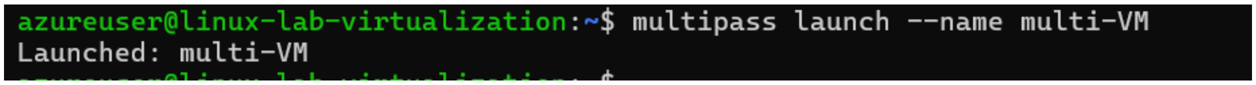
```
azureuser@linux-lab-virtualization:~$ sudo snap install multipass
multipass 1.15.0 from Canonical✓ installed
azureuser@linux-lab-virtualization:~$ multipass version
multipass   1.15.0
multipassd  1.15.0
azureuser@linux-lab-virtualization:~$
```

Basic commands: Use the multipass command-line interface and practice the following:

1. **multipass boot: Launch the default Ubuntu instance.**

multipass launch --name my-instance

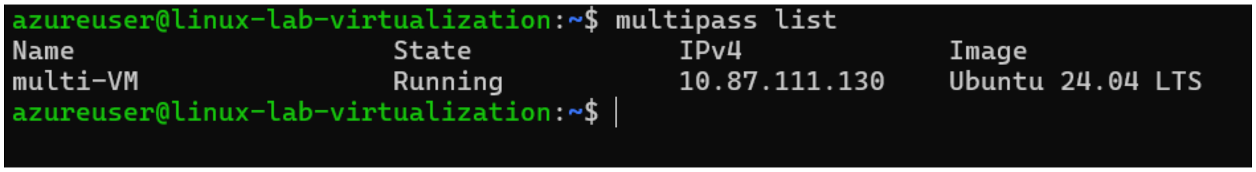
multipass launch --name multi-VM (Name of the instance is multi-VM)



```
azureuser@linux-lab-virtualization:~$ multipass launch --name multi-VM
Launched: multi-VM
```

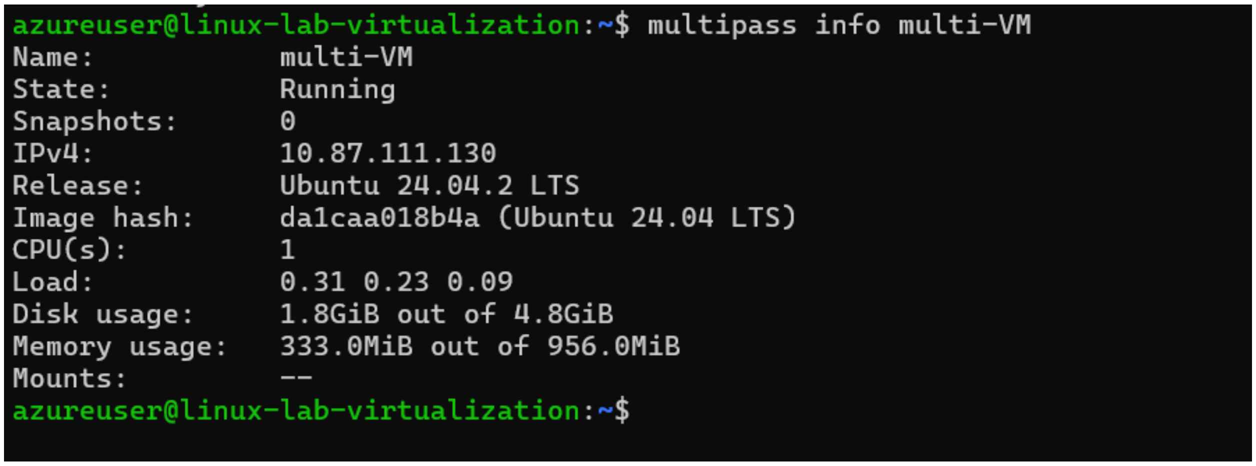
2. **multipass list: Lists all running instances.**

multipass list



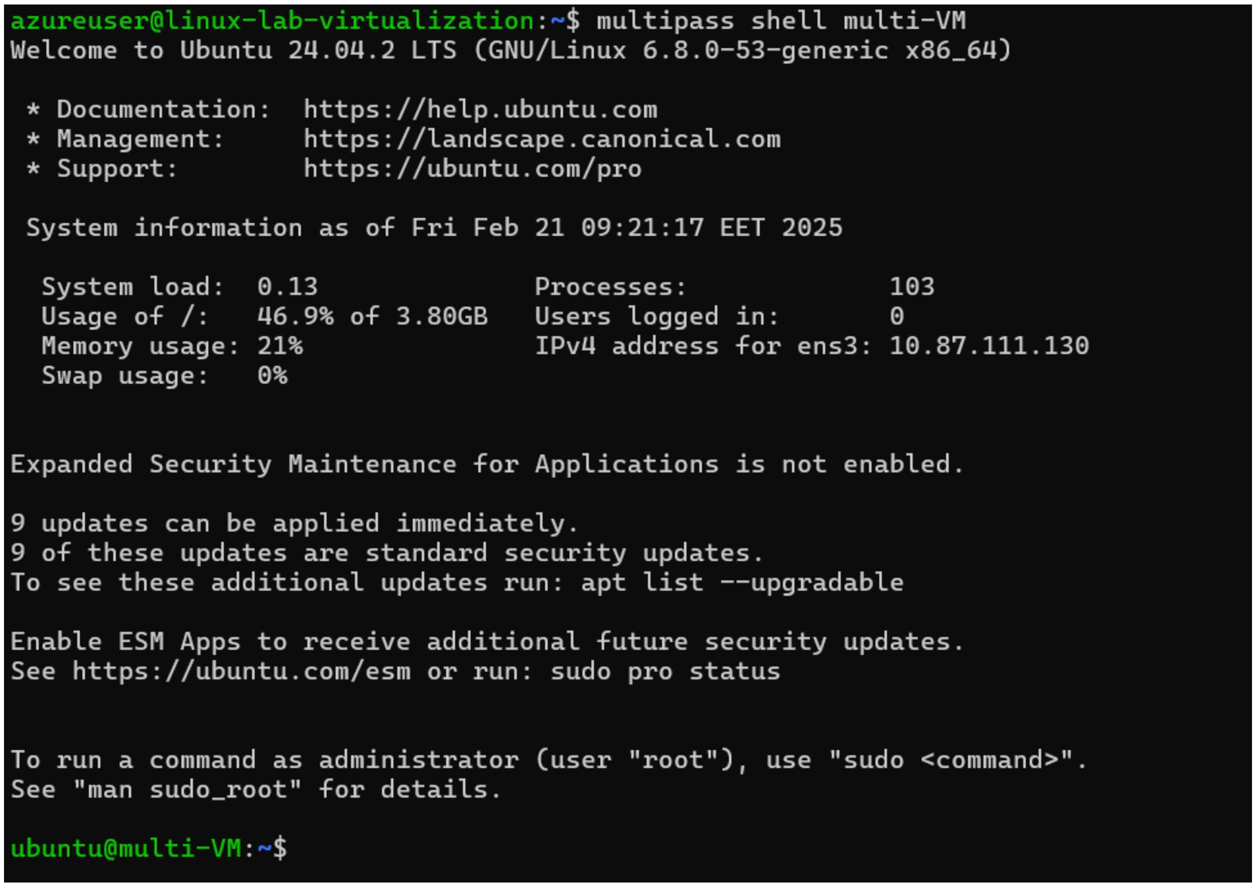
```
azureuser@linux-lab-virtualization:~$ multipass list
Name                    State             IPv4             Image
multi-VM                Running           10.87.111.130    Ubuntu 24.04 LTS
azureuser@linux-lab-virtualization:~$
```

3. **multipass info: View details about a specific instance.**



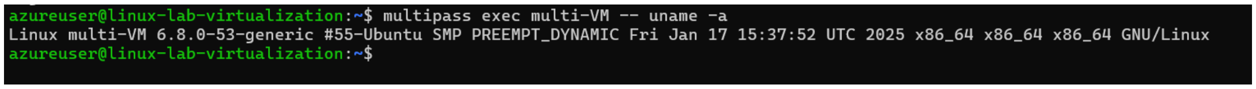
```
azureuser@linux-lab-virtualization:~$ multipass info multi-VM
Name:           multi-VM
State:          Running
Snapshots:      0
IPv4:           10.87.111.130
Release:        Ubuntu 24.04.2 LTS
Image hash:     da1caa018b4a (Ubuntu 24.04 LTS)
CPU(s):         1
Load:           0.31 0.23 0.09
Disk usage:     1.8GiB out of 4.8GiB
Memory usage:   333.0MiB out of 956.0MiB
Mounts:         --
azureuser@linux-lab-virtualization:~$
```

4. **multipass shell: Access to the shell of a running instance.**



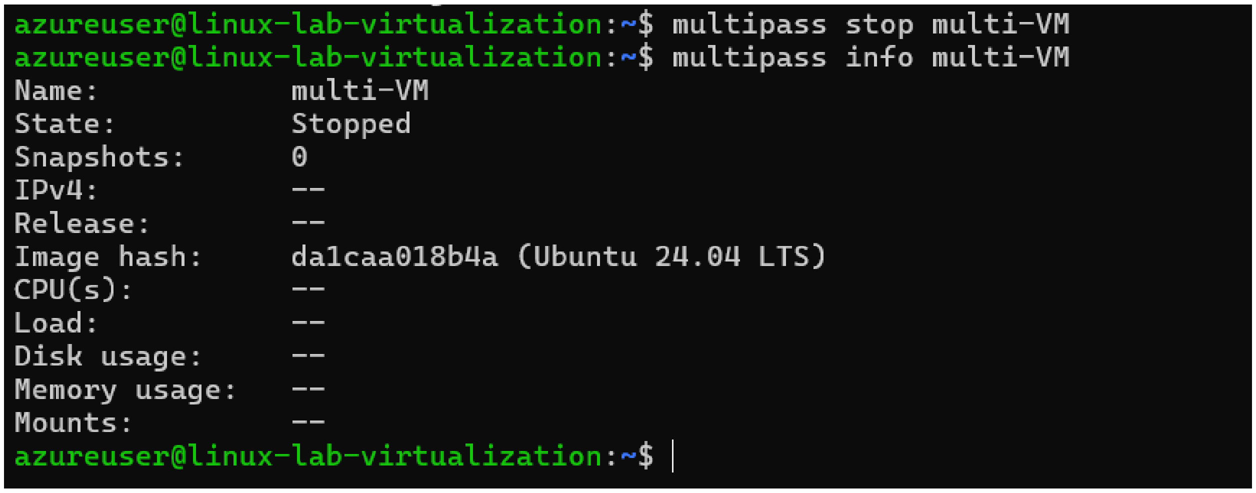
```
azureuser@linux-lab-virtualization:~$ multipass shell multi-VM
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-53-generic x86_64)

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

 System information as of Fri Feb 21 09:21:17 EET 2025

  System load:  0.13               Processes:             103
  Usage of /:   46.9% of 3.80GB    Users logged in:       0
  Memory usage: 21%                IPv4 address for ens3: 10.87.111.130
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

9 updates can be applied immediately.
9 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@multi-VM:~$
```

5. **multipass exec: Run the command on the instance.**



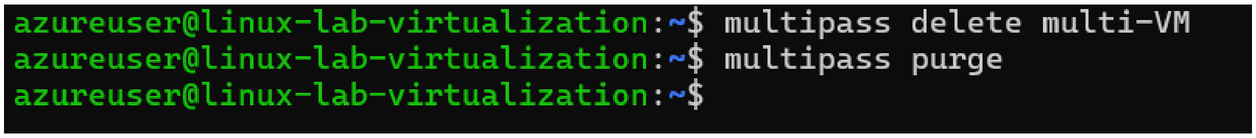
```
azureuser@linux-lab-virtualization:~$ multipass exec multi-VM -- uname -a
Linux multi-VM 6.8.0-53-generic #55-Ubuntu SMP PREEMPT_DYNAMIC Fri Jan 17 15:37:52 UTC 2025 x86_64 x86_64 x86_64 GNU/Linux
azureuser@linux-lab-virtualization:~$
```

6. **multipass stop: Stop the running instance.**



```
azureuser@linux-lab-virtualization:~$ multipass stop multi-VM
azureuser@linux-lab-virtualization:~$ multipass info multi-VM
Name:           multi-VM
State:          Stopped
Snapshots:      0
IPv4:           --
Release:        --
Image hash:     da1caa018b4a (Ubuntu 24.04 LTS)
CPU(s):         --
Load:           --
Disk usage:     --
Memory usage:   --
Mounts:         --
azureuser@linux-lab-virtualization:~$
```

7. **multipass delete: Delete the instance.**

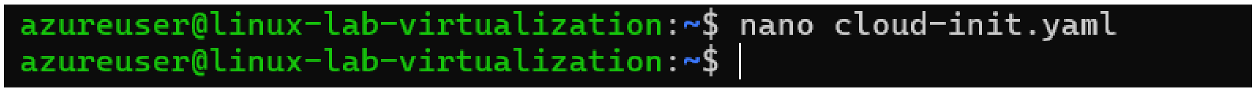


```
azureuser@linux-lab-virtualization:~$ multipass delete multi-VM
azureuser@linux-lab-virtualization:~$ multipass purge
azureuser@linux-lab-virtualization:~$
```

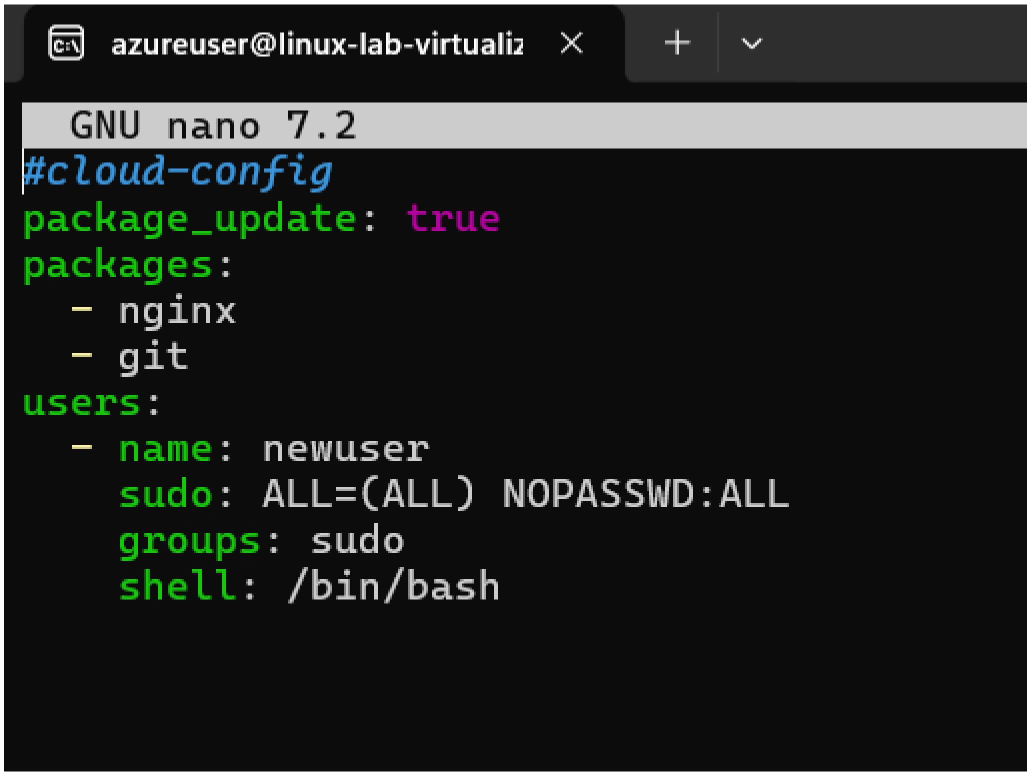
Cloud-init:Study: Learn about cloud-initi and how it can be used to configure virtual machines

Experiment: Create a cloud-init configuration file to customize the installation of a new instance.

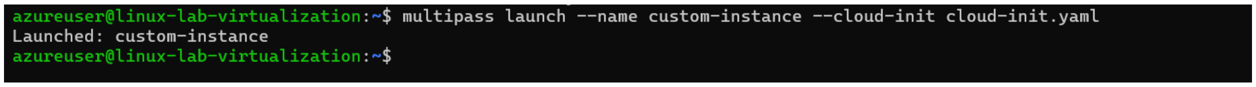
- **For example, you can install specific packages or assign users.**
- **Start a new instance of Multipass using this configuration.**



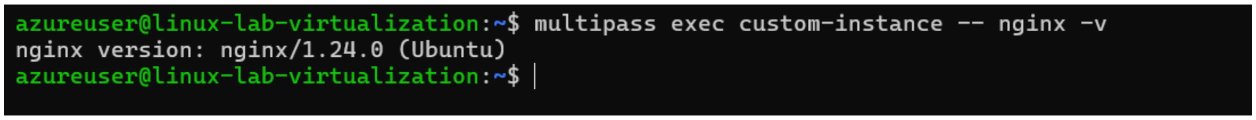
```
azureuser@linux-lab-virtualization:~$ nano cloud-init.yaml
azureuser@linux-lab-virtualization:~$
```



```
  GNU nano 7.2
#cloud-config
package_update: true
packages:
  - nginx
  - git
users:
  - name: newuser
    sudo: ALL=(ALL) NOPASSWD:ALL
    groups: sudo
    shell: /bin/bash
```



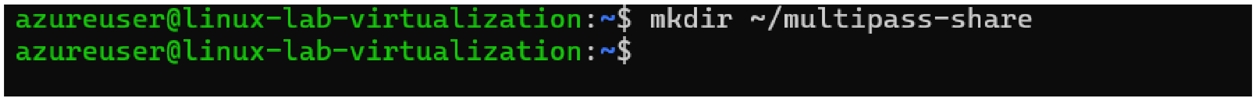
```
azureuser@linux-lab-virtualization:~$ multipass launch --name custom-instance --cloud-init cloud-init.yaml
Launched: custom-instance
azureuser@linux-lab-virtualization:~$
```



```
azureuser@linux-lab-virtualization:~$ multipass exec custom-instance -- nginx -v
nginx version: nginx/1.24.0 (Ubuntu)
azureuser@linux-lab-virtualization:~$
```

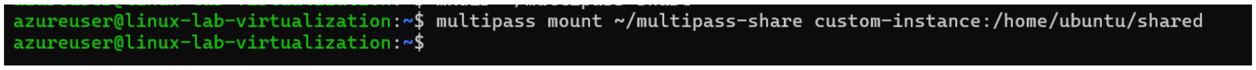
File sharing: Find out: Explore how to share files and folders between your host computer and Multipass instances

1. **Create a shared folder on your Linux VM**



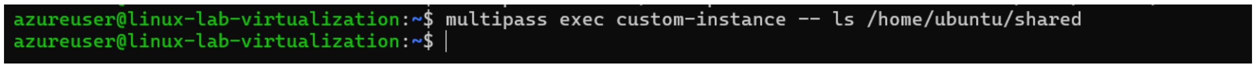
```
azureuser@linux-lab-virtualization:~$ mkdir ~/multipass-share
azureuser@linux-lab-virtualization:~$
```

2. **Mount the folder inside the Multipass instance**



```
azureuser@linux-lab-virtualization:~$ multipass mount ~/multipass-share custom-instance:/home/ubuntu/shared
azureuser@linux-lab-virtualization:~$
```

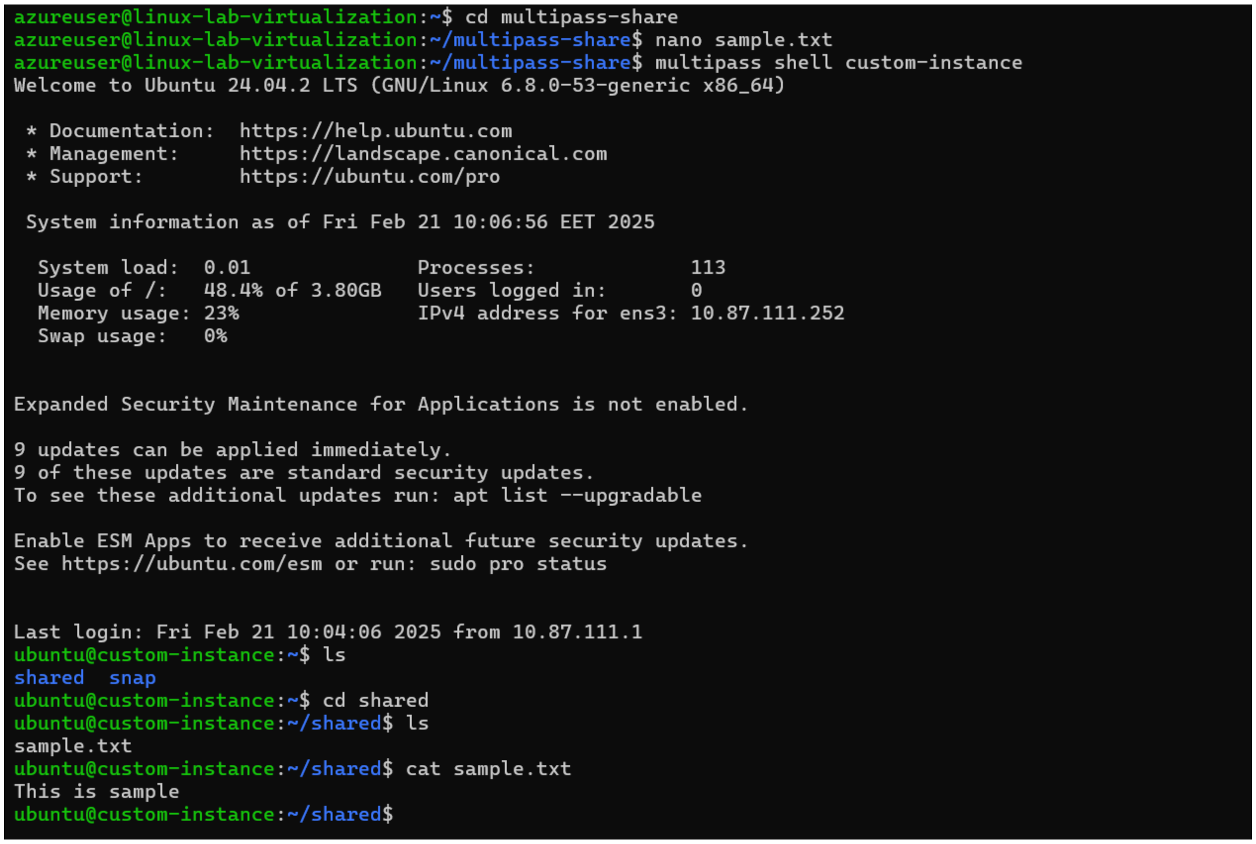
3. **Check the shared folder inside the instance**



```
azureuser@linux-lab-virtualization:~$ multipass exec custom-instance -- ls /home/ubuntu/shared
azureuser@linux-lab-virtualization:~$ |
```

Policy: Create a shared folder and access it from both your host and your Multipass instance.

1. **Create sample.txt file inside VM and check availability inside custom-instance VM,**



```
azureuser@linux-lab-virtualization:~$ cd multipass-share
azureuser@linux-lab-virtualization:~/multipass-share$ nano sample.txt
azureuser@linux-lab-virtualization:~/multipass-share$ multipass shell custom-instance
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-53-generic x86_64)

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

 System information as of Fri Feb 21 10:06:56 EET 2025

  System load:  0.01              Processes:             113
  Usage of /:   48.4% of 3.80GB   Users logged in:       0
  Memory usage: 23%               IPv4 address for ens3: 10.87.111.252
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

9 updates can be applied immediately.
9 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


Last login: Fri Feb 21 10:04:06 2025 from 10.87.111.1
ubuntu@custom-instance:~$ ls
shared  snap
ubuntu@custom-instance:~$ cd shared
ubuntu@custom-instance:~/shared$ ls
sample.txt
ubuntu@custom-instance:~/shared$ cat sample.txt
This is sample
ubuntu@custom-instance:~/shared$
```

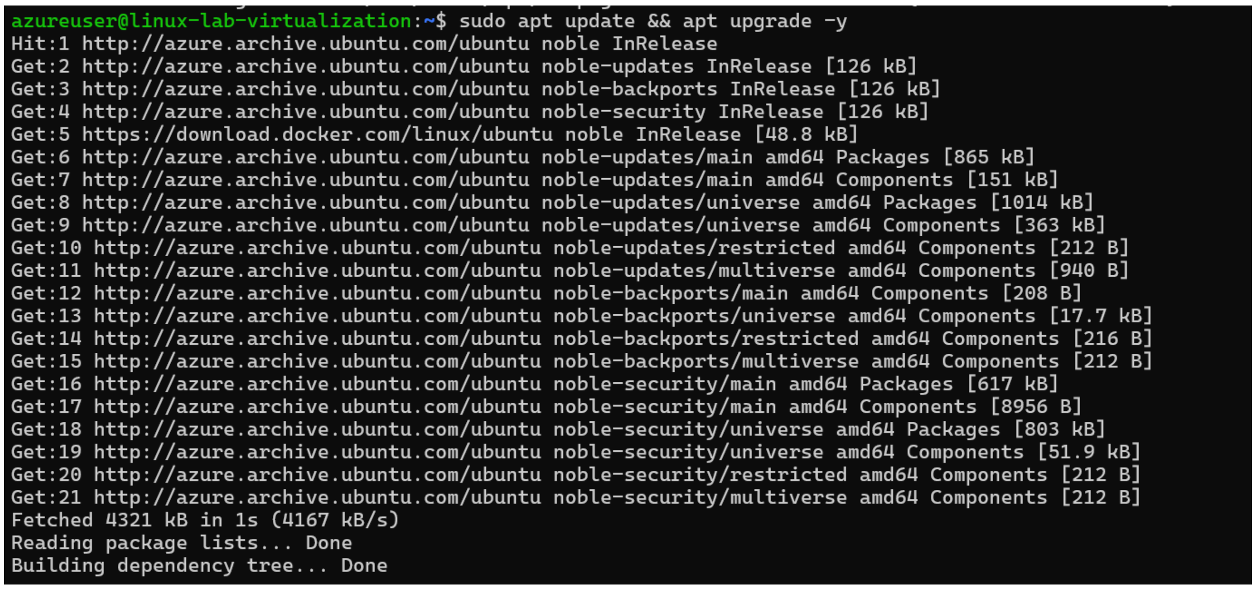
Part 3: Exploring LXD (1 hour)

LXD is another container and virtual machine management built on LXC, the so-called "Virtual Machine Management" of the Linux container. Runtime.

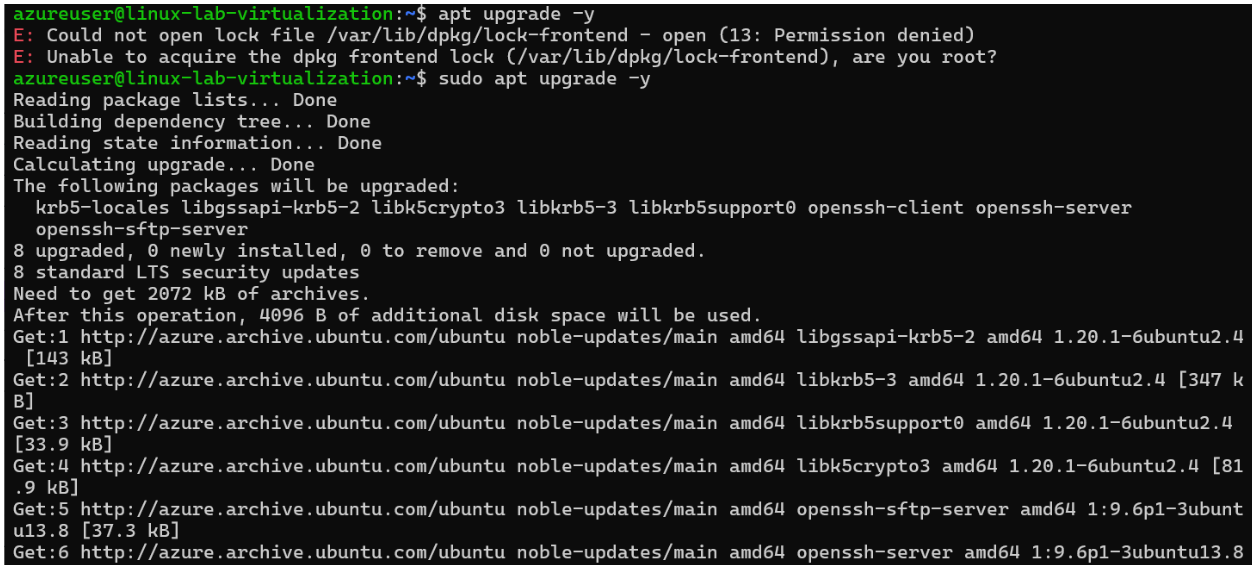
Study: Read about LXD and its features on LinuxContainers website Setup: Install LXD on your system, enable LXD Basic commands: Experiment with basic LXD commands to create, manage, and interact with containers. You can find the starting point in the source

Installing and managing LXD / LXC system

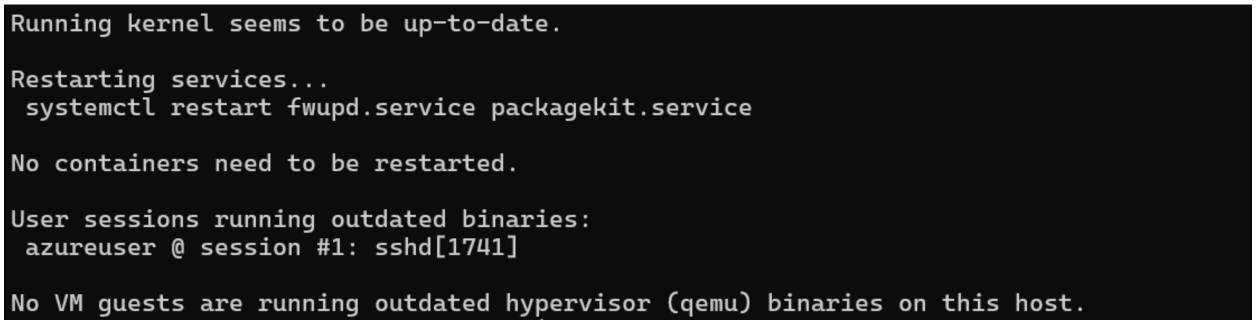
1. **update your system by using apt: apt update && apt upgrade -y**



```
azureuser@linux-lab-virtualization:~$ sudo apt update && apt upgrade -y
Hit:1 http://azure.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [865 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1014 kB]
Get:9 http://azure.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [363 kB]
Get:10 http://azure.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:11 http://azure.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:12 http://azure.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:13 http://azure.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [17.7 kB]
Get:14 http://azure.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:15 http://azure.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:16 http://azure.archive.ubuntu.com/ubuntu noble-security/main amd64 Packages [617 kB]
Get:17 http://azure.archive.ubuntu.com/ubuntu noble-security/main amd64 Components [8956 B]
Get:18 http://azure.archive.ubuntu.com/ubuntu noble-security/universe amd64 Packages [803 kB]
Get:19 http://azure.archive.ubuntu.com/ubuntu noble-security/universe amd64 Components [51.9 kB]
Get:20 http://azure.archive.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:21 http://azure.archive.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Fetched 4321 kB in 1s (4167 kB/s)
Reading package lists... Done
Building dependency tree... Done
```

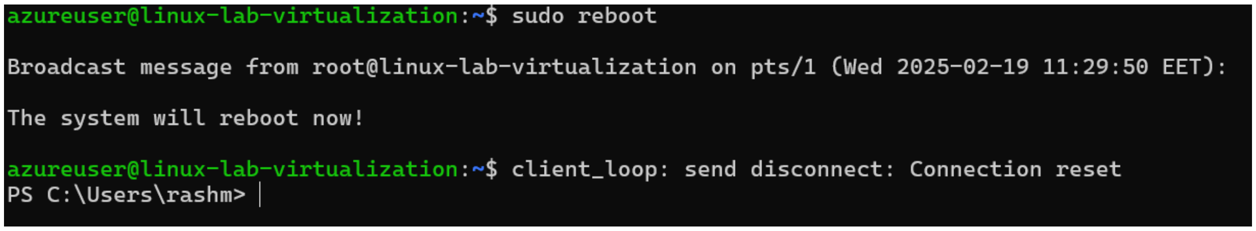


```
azureuser@linux-lab-virtualization:~$ apt upgrade -y
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?
azureuser@linux-lab-virtualization:~$ sudo apt upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  krb5-locales libgssapi-krb5-2 libk5crypto3 libkrb5-3 libkrb5support0 openssh-client openssh-server
  openssh-sftp-server
8 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
8 standard LTS security updates
Need to get 2072 kB of archives.
After this operation, 4096 B of additional disk space will be used.
Get:1 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 libgssapi-krb5-2 amd64 1.20.1-6ubuntu2.4
 [143 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 libkrb5-3 amd64 1.20.1-6ubuntu2.4 [347 k
B]
Get:3 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 libkrb5support0 amd64 1.20.1-6ubuntu2.4
[33.9 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 libk5crypto3 amd64 1.20.1-6ubuntu2.4 [81
.9 kB]
Get:5 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 openssh-sftp-server amd64 1:9.6p1-3ubunt
u13.8 [37.3 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 openssh-server amd64 1:9.6p1-3ubuntu13.8
```



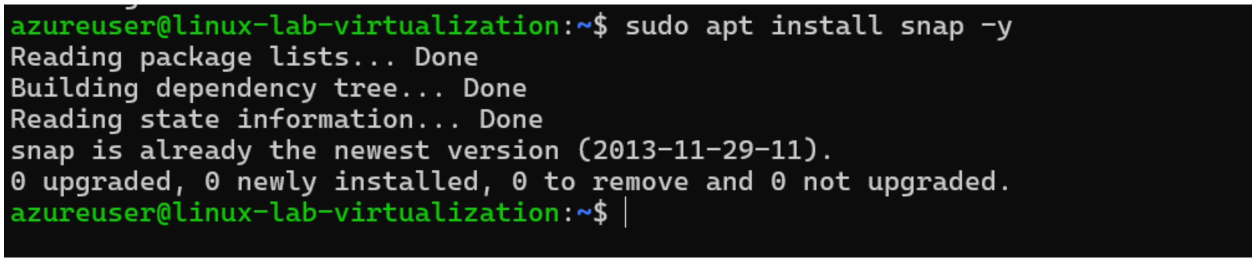
```
Running kernel seems to be up-to-date.

Restarting services...
 systemctl restart fwupd.service packagekit.service

No containers need to be restarted.

User sessions running outdated binaries:
 azureuser @ session #1: sshd[1741]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

2. **check if you just updated your kernel or other systems needing a complete system reboot, and if so, reboot: sudo reboot**



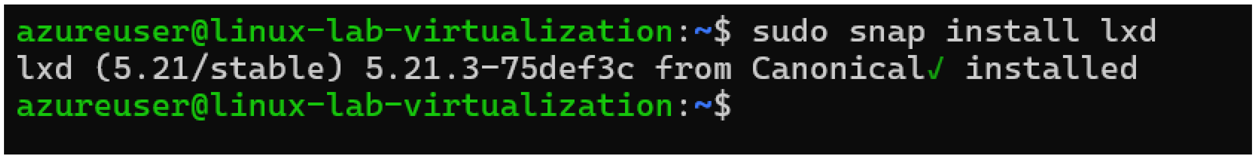
```
azureuser@linux-lab-virtualization:~$ sudo reboot

Broadcast message from root@linux-lab-virtualization on pts/1 (Wed 2025-02-19 11:29:50 EET):

The system will reboot now!

azureuser@linux-lab-virtualization:~$ client_loop: send disconnect: Connection reset
PS C:\Users\rashm>
```

3. **Install Snap: sudo apt install snap -y**

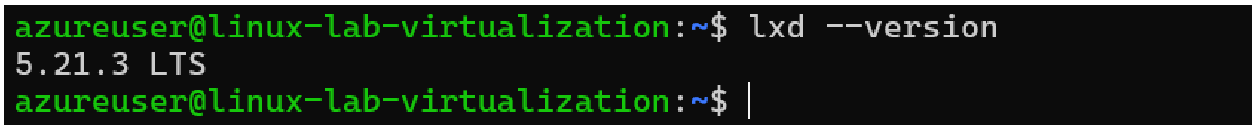


```
azureuser@linux-lab-virtualization:~$ sudo apt install snap -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
snap is already the newest version (2013-11-29-11).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
azureuser@linux-lab-virtualization:~$
```

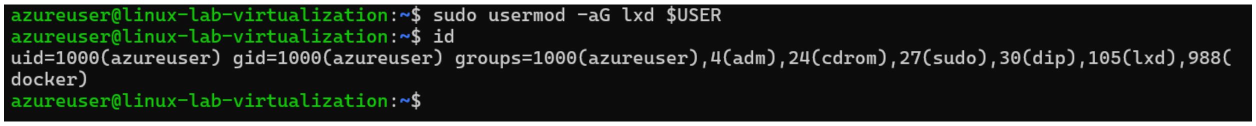
4. **install lxd using snap: sudo snap install lxd**


```
azureuser@linux-lab-virtualization:~$ sudo snap install lxd
lxd (5.21/stable) 5.21.3-75def3c from Canonical✓ installed
azureuser@linux-lab-virtualization:~$
```

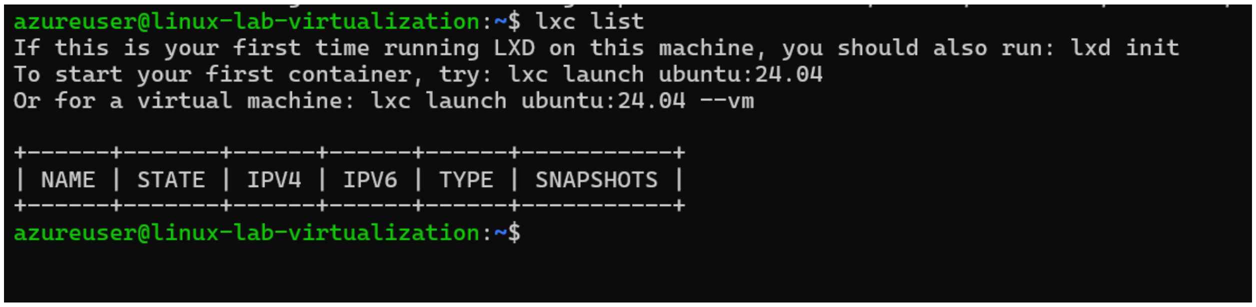
5. **check lxd version and installation: lxd --version**


```
azureuser@linux-lab-virtualization:~$ lxd --version
5.21.3 LTS
azureuser@linux-lab-virtualization:~$ |
```

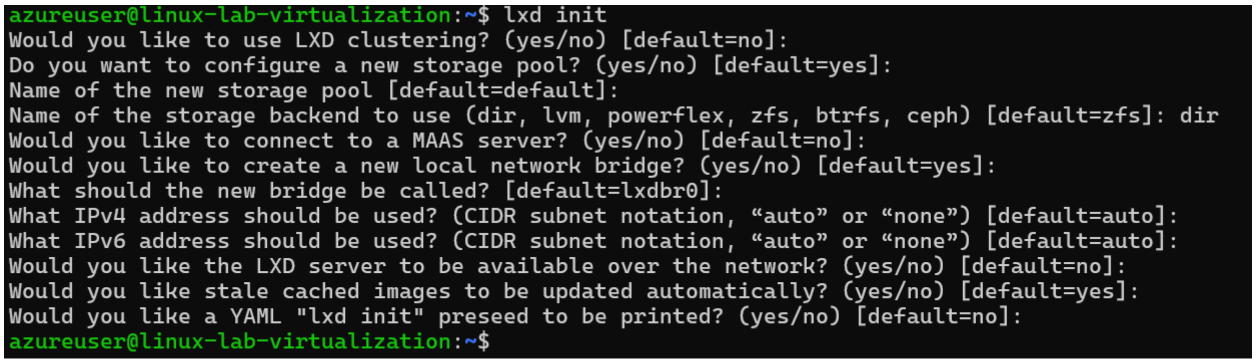
6. **check that your user belongs to LXD group: id, and look for LXD. If you do not find lxd group, add user to it: sudo usermod -aG lxd $USER**


```
azureuser@linux-lab-virtualization:~$ sudo usermod -aG lxd $USER
azureuser@linux-lab-virtualization:~$ id
uid=1000(azureuser) gid=1000(azureuser) groups=1000(azureuser),4(adm),24(cdrom),27(sudo),30(dip),105(lxd),988(
docker)
azureuser@linux-lab-virtualization:~$
```

7. **check lxc system for listing of machines and containers: lxc list**


```
azureuser@linux-lab-virtualization:~$ lxc list
If this is your first time running LXD on this machine, you should also run: lxd init
To start your first container, try: lxc launch ubuntu:24.04
Or for a virtual machine: lxc launch ubuntu:24.04 --vm

+------+-------+------+------+------+-----------+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+------+-------+------+------+------+-----------+
azureuser@linux-lab-virtualization:~$
```

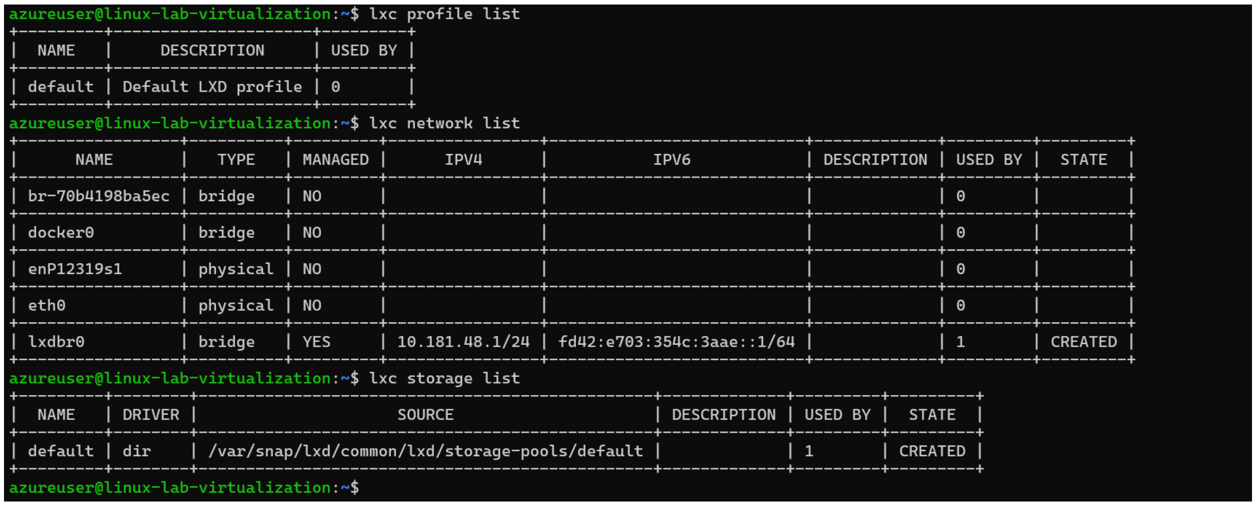
8. **initialize xld, to configure system to your environment: lxd init. Make sure you run this as basic user, without root / sudo privileges it will ask series of questions, so based on your requirement answer them.**
    - **When it asks about clustering, choose 'no' (unless you're setting up a cluster)**
    - **For storage, I recommend saying 'yes' to a new storage pool**
    - **The 'dir' backend is fine for beginners**
    - **Say 'yes' to a network bridge**
    - **Choose 'no' to make LXD sever available over the network.**



```
azureuser@linux-lab-virtualization:~$ lxd init
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (dir, lvm, powerflex, zfs, btrfs, ceph) [default=zfs]: dir
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
Would you like the LXD server to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
azureuser@linux-lab-virtualization:~$
```

9. **Once lxd is initialized successfully, we can verify the information using following set of commands:**

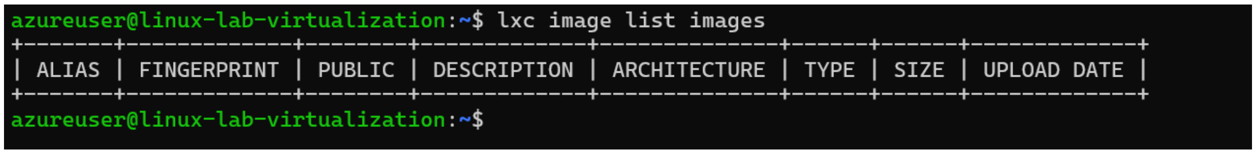
$lxc profile list $lxc network list $lxc storage list



```
azureuser@linux-lab-virtualization:~$ lxc profile list
+---------+---------------------+---------+
|  NAME   |     DESCRIPTION     | USED BY |
+---------+---------------------+---------+
| default | Default LXD profile | 0       |
+---------+---------------------+---------+
azureuser@linux-lab-virtualization:~$ lxc network list
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
|      NAME       |   TYPE   | MANAGED |      IPV4      |           IPV6            | DESCRIPTION | USED BY |  STATE  |
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
| br-70b4198ba5ec | bridge   | NO      |                |                           |             | 0       |         |
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
| docker0         | bridge   | NO      |                |                           |             | 0       |         |
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
| enP12319s1      | physical | NO      |                |                           |             | 0       |         |
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
| eth0            | physical | NO      |                |                           |             | 0       |         |
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
| lxdbr0          | bridge   | YES     | 10.181.48.1/24 | fd42:e703:354c:3aae::1/64 |             | 1       | CREATED |
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
azureuser@linux-lab-virtualization:~$ lxc storage list
+---------+--------+------------------------------------------------+-------------+---------+---------+
|  NAME   | DRIVER |                     SOURCE                     | DESCRIPTION | USED BY |  STATE  |
+---------+--------+------------------------------------------------+-------------+---------+---------+
| default | dir    | /var/snap/lxd/common/lxd/storage-pools/default |             | 1       | CREATED |
+---------+--------+------------------------------------------------+-------------+---------+---------+
azureuser@linux-lab-virtualization:~$
```

10. **In order to list all available images, run:**

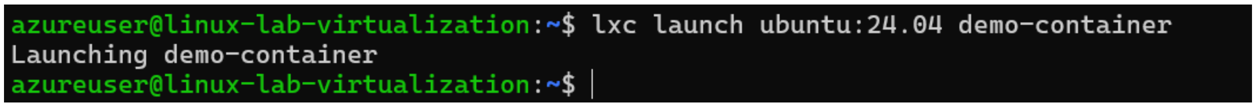
$lxc image list images:



```
azureuser@linux-lab-virtualization:~$ lxc image list images
+-------+-------------+--------+-------------+--------------+------+------+-------------+
| ALIAS | FINGERPRINT | PUBLIC | DESCRIPTION | ARCHITECTURE | TYPE | SIZE | UPLOAD DATE |
+-------+-------------+--------+-------------+--------------+------+------+-------------+
azureuser@linux-lab-virtualization:~$
```

11. **Create your first container:**

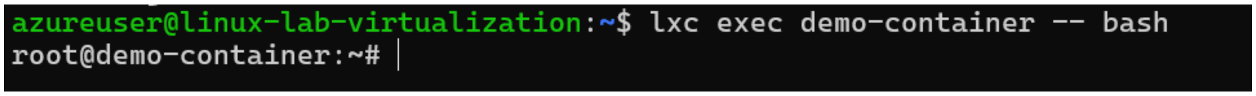
$lxc launch ubuntu:24.04 demo-container



```
azureuser@linux-lab-virtualization:~$ lxc launch ubuntu:24.04 demo-container
Launching demo-container
azureuser@linux-lab-virtualization:~$
```

12. **Access the console of container. Run:**

$ lxc exec demo-container -- bash



```
azureuser@linux-lab-virtualization:~$ lxc exec demo-container -- bash
root@demo-container:~#
```

Part 4: How to Stick Apps with Docker (1 hour)

Docker is a platform for building, deploying and managing container applications

Installation: Install Docker Desktop on your system if you want to familiarize yourself with this section, You can find installation information for Docker Engine in the sources.

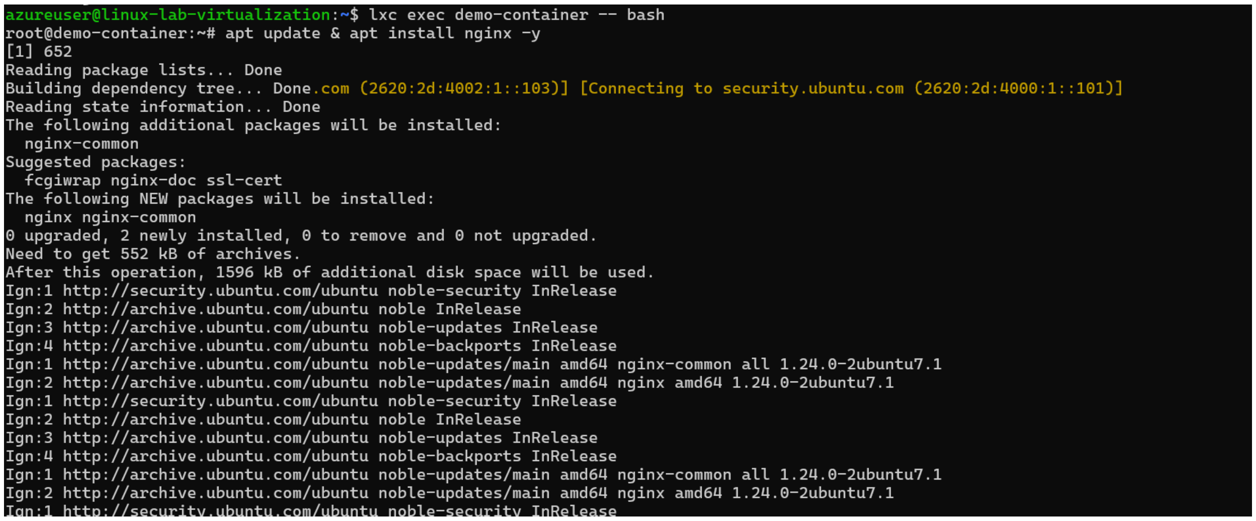
Basic concepts: Research: Explore core Docker concepts such as images, containers, and Dockerfiles. Source.

Experiment: Follow the "Docker Workshop" to get hands-on experience with Docker.

Installing and managing Docker engine based system

Basic steps:

Follow good instructions from Docker, at https://docs.docker.com/



```
azureuser@linux-lab-virtualization:~$ lxc exec demo-container -- bash
root@demo-container:~# apt update & apt install nginx -y
[1] 652
Reading package lists... Done
Building dependency tree... Done.com (2620:2d:4002:1::103)] [Connecting to security.ubuntu.com (2620:2d:4000:1::101)]
Reading state information... Done
The following additional packages will be installed:
  nginx-common
Suggested packages:
  fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
  nginx nginx-common
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 552 kB of archives.
After this operation, 1596 kB of additional disk space will be used.
Ign:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:2 http://archive.ubuntu.com/ubuntu noble InRelease
Ign:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Ign:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 nginx-common all 1.24.0-2ubuntu7.1
Ign:2 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 nginx amd64 1.24.0-2ubuntu7.1
Ign:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:2 http://archive.ubuntu.com/ubuntu noble InRelease
Ign:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Ign:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 nginx-common all 1.24.0-2ubuntu7.1
Ign:2 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 nginx amd64 1.24.0-2ubuntu7.1
Ign:1 http://security.ubuntu.com/ubuntu noble-security InRelease
```

More specifically from Ubuntu engine install instructions for Ubuntu system, and using convenience script:

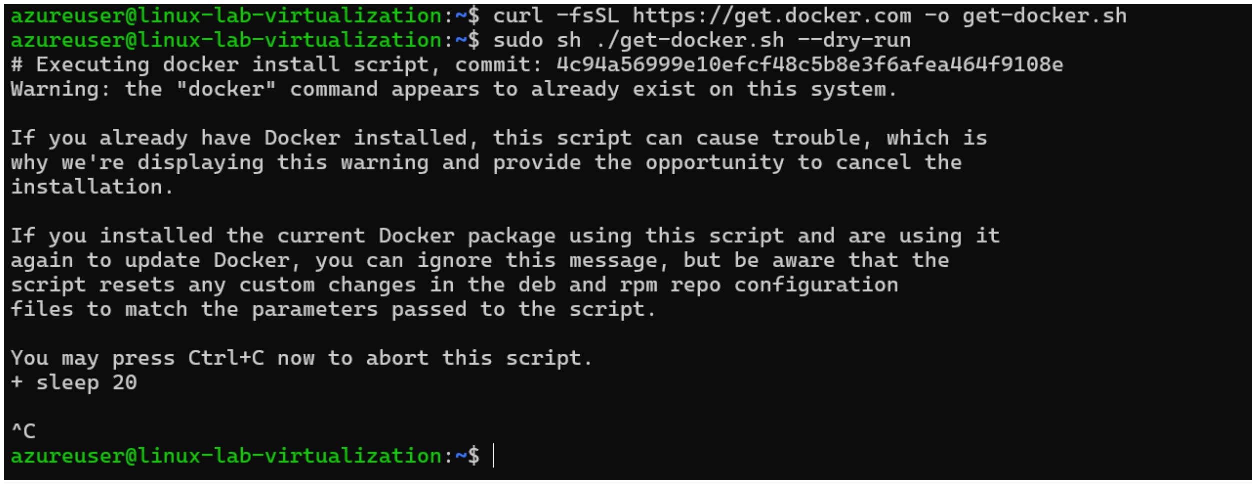
https://docs.docker.com/engine/install/ubuntu/#install-using-the-convenience-script

curl -fsSL https://get.docker.com -o get-docker.sh

sudo sh get-docker.sh

Executing docker install script, commit: 7cae5f8b0decc17d6571f9f52eb840fbc13b2737

<...>



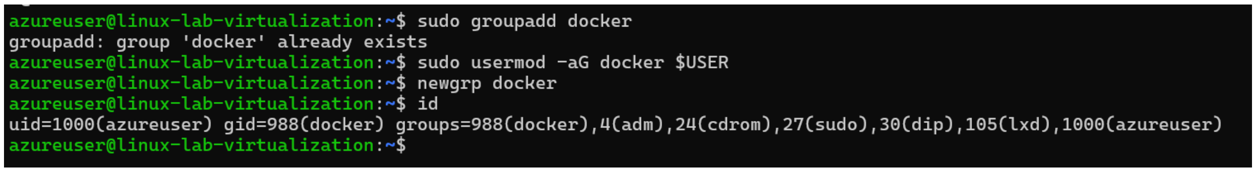
```
azureuser@linux-lab-virtualization:~$ curl -fsSL https://get.docker.com -o get-docker.sh
azureuser@linux-lab-virtualization:~$ sudo sh ./get-docker.sh --dry-run
# Executing docker install script, commit: 4c94a56999e10efcf48c5b8e3f6afea464f9108e
Warning: the "docker" command appears to already exist on this system.

If you already have Docker installed, this script can cause trouble, which is
why we're displaying this warning and provide the opportunity to cancel the
installation.

If you installed the current Docker package using this script and are using it
again to update Docker, you can ignore this message, but be aware that the
script resets any custom changes in the deb and rpm repo configuration
files to match the parameters passed to the script.

You may press Ctrl+C now to abort this script.
+ sleep 20

^C
azureuser@linux-lab-virtualization:~$
```

post installation guide tells us to make user part of Docker group on Linux machine:

sudo groupadd docker

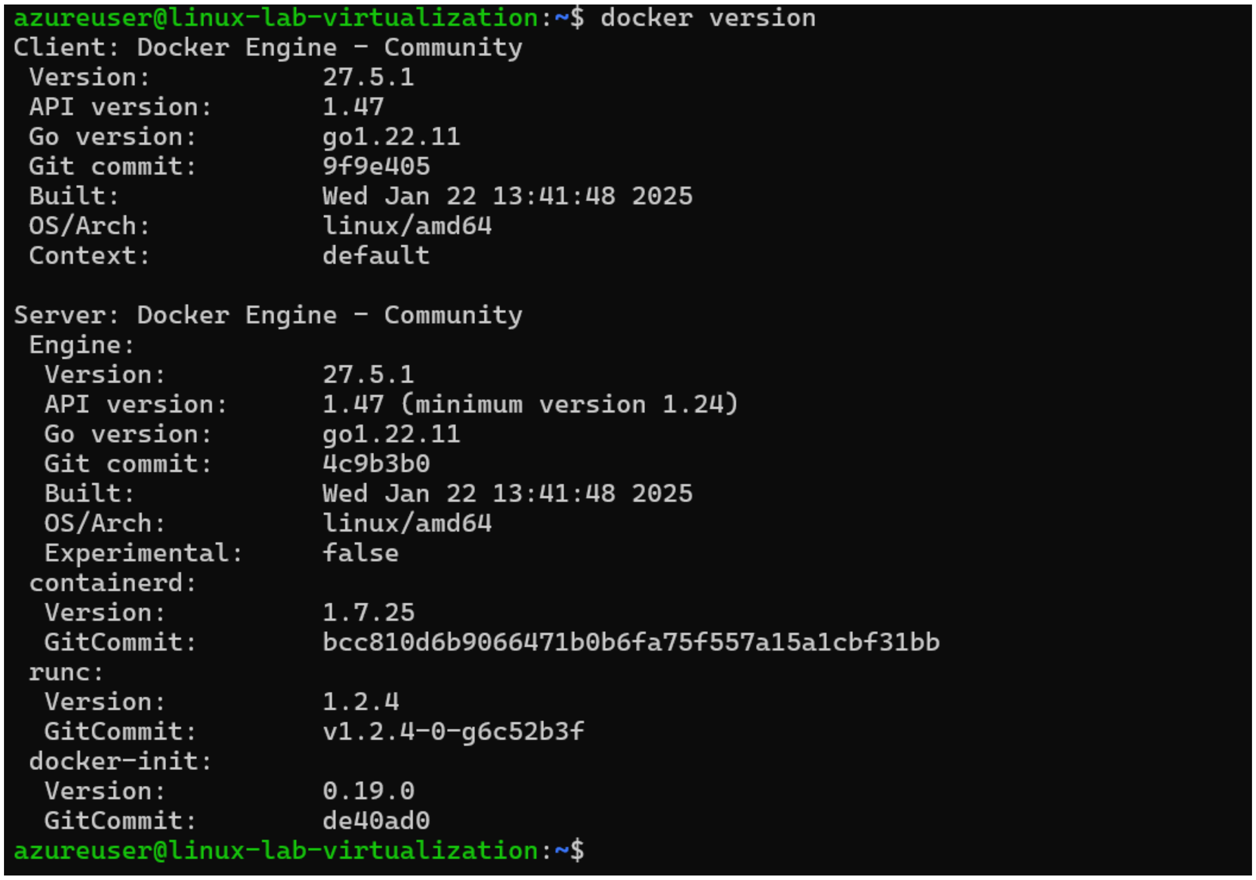
sudo usermod -aG docker $USER

newgrp docker



```
azureuser@linux-lab-virtualization:~$ sudo groupadd docker
groupadd: group 'docker' already exists
azureuser@linux-lab-virtualization:~$ sudo usermod -aG docker $USER
azureuser@linux-lab-virtualization:~$ newgrp docker
azureuser@linux-lab-virtualization:~$ id
uid=1000(azureuser) gid=988(docker) groups=988(docker),4(adm),24(cdrom),27(sudo),30(dip),105(lxd),1000(azureuser)
azureuser@linux-lab-virtualization:~$
```

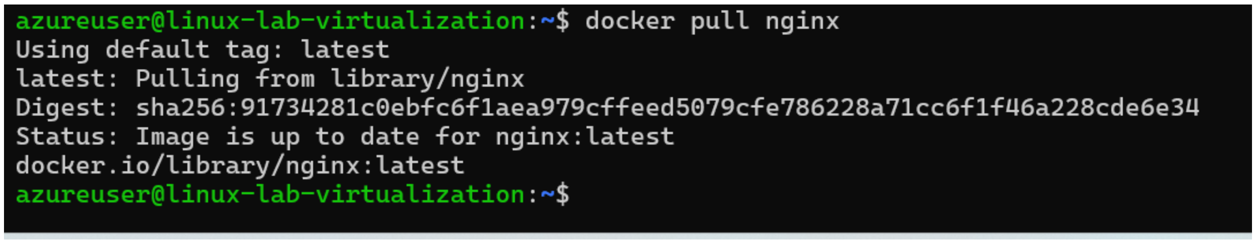
check docker version



```
azureuser@linux-lab-virtualization:~$ docker version
Client: Docker Engine - Community
 Version:           27.5.1
 API version:       1.47
 Go version:        go1.22.11
 Git commit:        9f9e405
 Built:             Wed Jan 22 13:41:48 2025
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          27.5.1
  API version:      1.47 (minimum version 1.24)
  Go version:       go1.22.11
  Git commit:       4c9b3b0
  Built:            Wed Jan 22 13:41:48 2025
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.7.25
  GitCommit:        bcc810d6b9066471b0b6fa75f557a15a1cbf31bb
 runc:
  Version:          1.2.4
  GitCommit:        v1.2.4-0-g6c52b3f
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
azureuser@linux-lab-virtualization:~$
```

Run Nginx on docker:

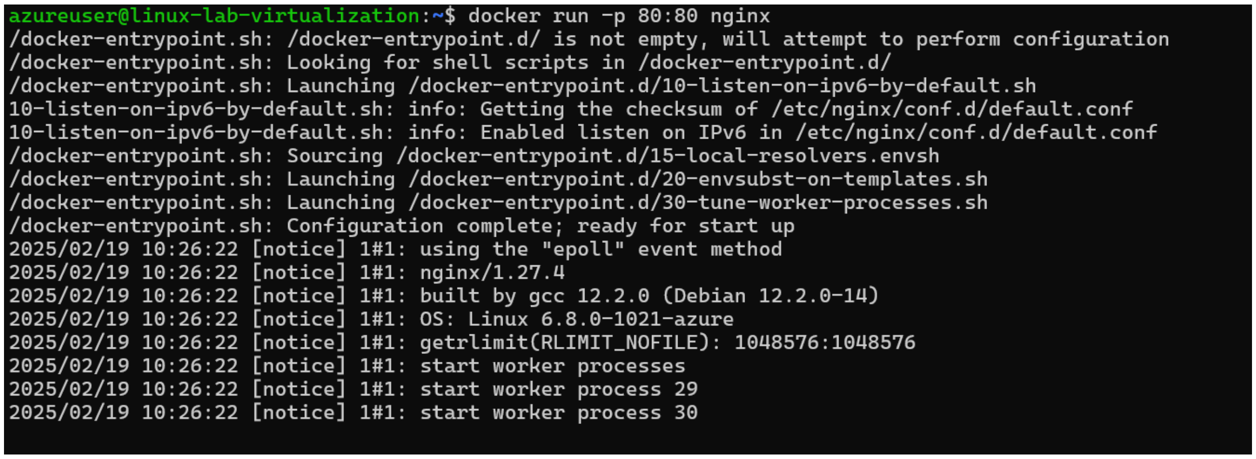
1. **Get the latest Nginx**



```
azureuser@linux-lab-virtualization:~$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
Digest: sha256:91734281c0ebfc6f1aea979cffeed5079cfe786228a71cc6f1f46a228cde6e34
Status: Image is up to date for nginx:latest
docker.io/library/nginx:latest
azureuser@linux-lab-virtualization:~$
```

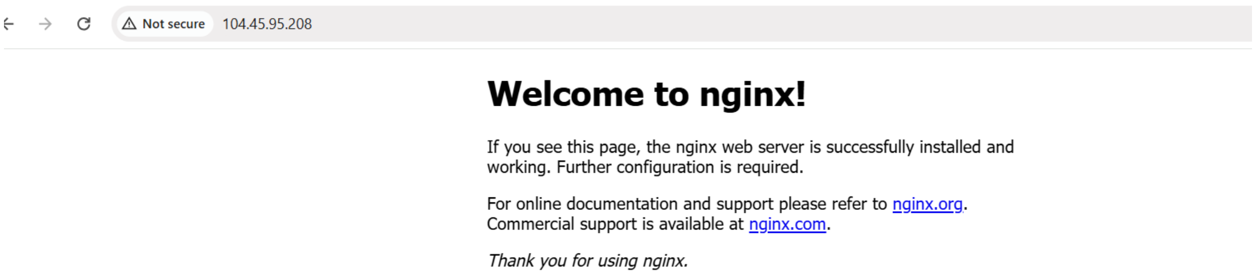
2. **Start docker nginx image**

    - **docker run -p 80:80 nginx**
    - **docker run -d -p 80:80 nginx (To run on background use -d tag)**



```
azureuser@linux-lab-virtualization:~$ docker run -p 80:80 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/02/19 10:26:22 [notice] 1#1: using the "epoll" event method
2025/02/19 10:26:22 [notice] 1#1: nginx/1.27.4
2025/02/19 10:26:22 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/02/19 10:26:22 [notice] 1#1: OS: Linux 6.8.0-1021-azure
2025/02/19 10:26:22 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/02/19 10:26:22 [notice] 1#1: start worker processes
2025/02/19 10:26:22 [notice] 1#1: start worker process 29
2025/02/19 10:26:22 [notice] 1#1: start worker process 30
```

3. **If we open the IP of our virtual machine we can see Nginx is running,**



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Part 5: Snaps for Self-Contained Applications (30 minutes)

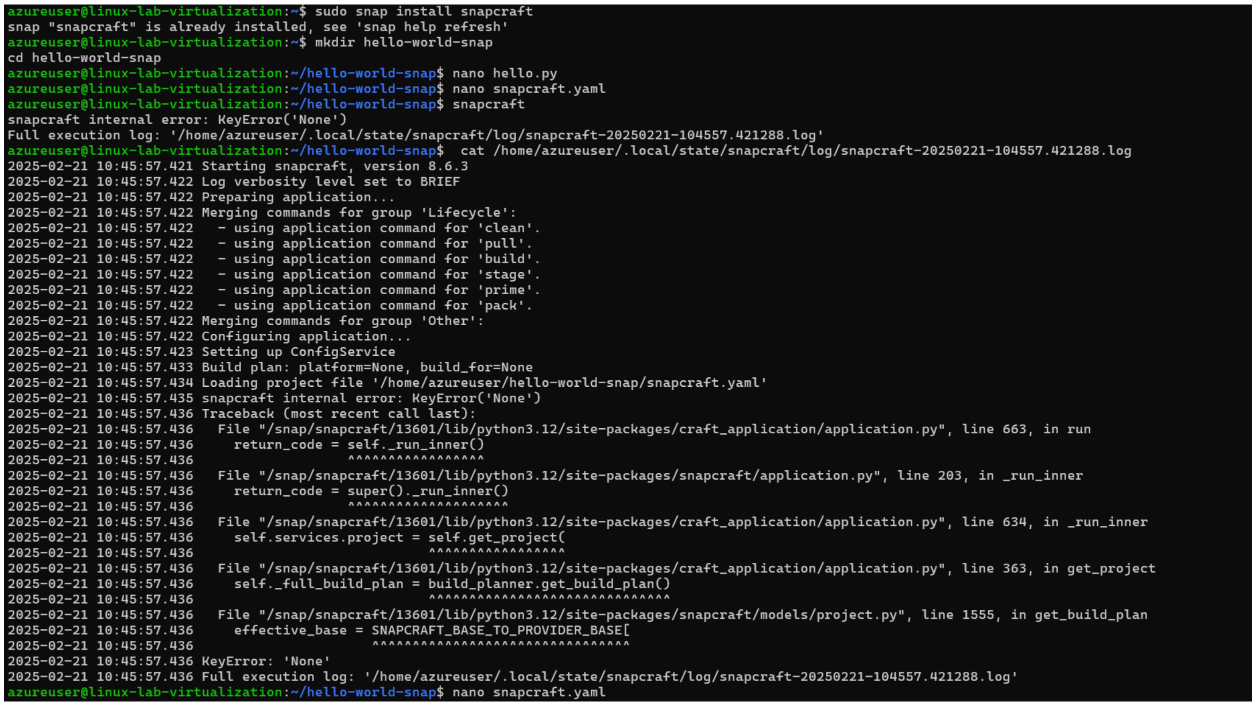
Snaps provides a way to package applications with their dependencies for consistent execution on Linux distributions.

- **Research: Using sources, explore Snapcraft and the concept of Snaps.**
- **Experiment: Choose a simple app and compress it into Snap using Snapcraft. For more information, you can find "Snapcraft" - from the source.**

Snapcraft is a tool and ecosystem that helps developers package their applications into snaps, which are self-contained software packages that include the application and all of its dependencies. This makes it easy to distribute and run applications across different Linux distributions without worrying about compatibility issues or missing dependencies. Snaps are designed to work on most modern Linux distributions, and they are confined, meaning they run in isolated environments to ensure security.

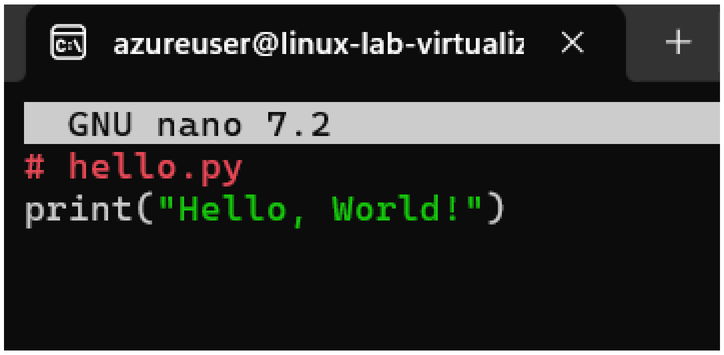
1. **Install snapcraft**

sudo apt update sudo apt install snapcraft



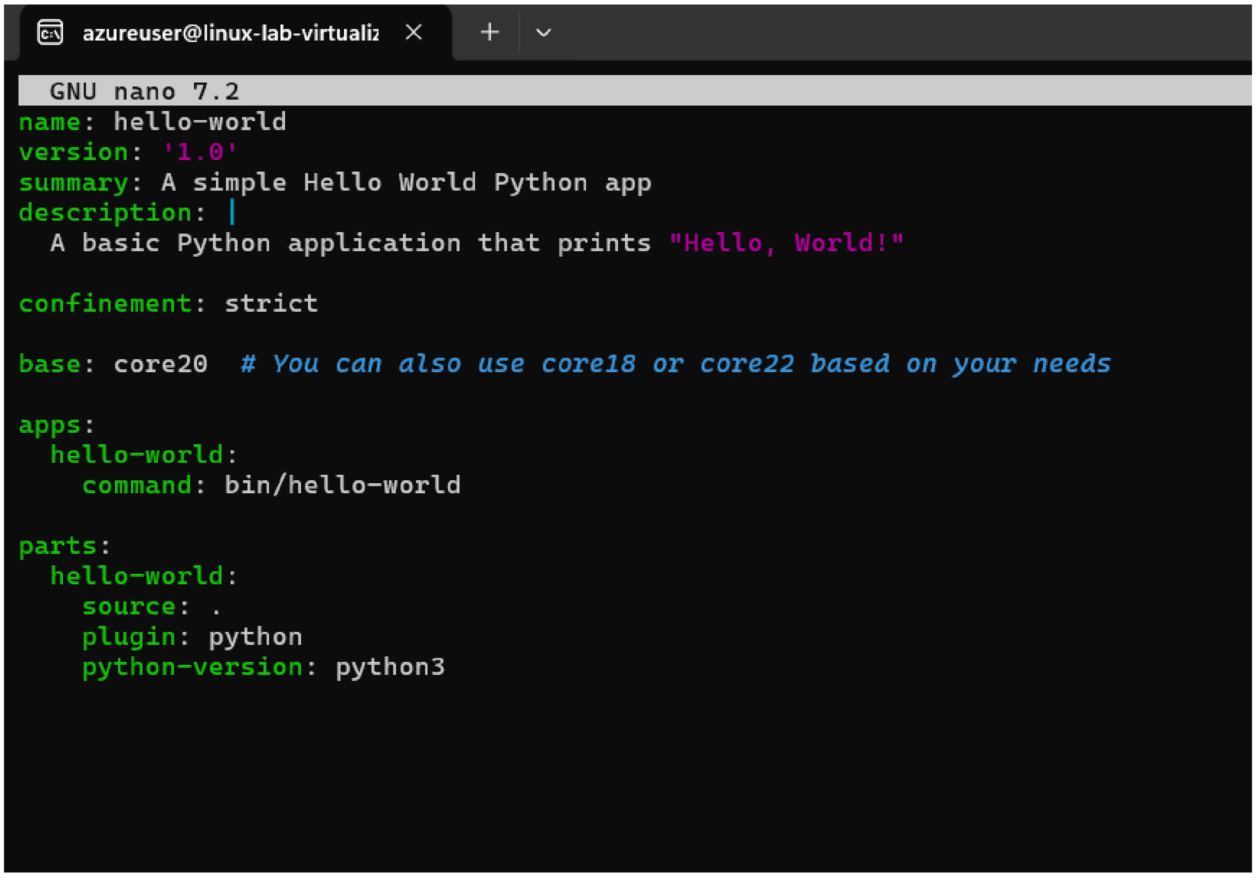
```
azureuser@linux-lab-virtualization:~$ sudo snap install snapcraft
snap "snapcraft" is already installed, see 'snap help refresh'
azureuser@linux-lab-virtualization:~$ mkdir hello-world-snap
cd hello-world-snap
azureuser@linux-lab-virtualization:~/hello-world-snap$ nano hello.py
azureuser@linux-lab-virtualization:~/hello-world-snap$ nano snapcraft.yaml
azureuser@linux-lab-virtualization:~/hello-world-snap$ snapcraft
snapcraft internal error: KeyError('None')
Full execution log: '/home/azureuser/.local/state/snapcraft/log/snapcraft-20250221-104557.421288.log'
azureuser@linux-lab-virtualization:~/hello-world-snap$  cat /home/azureuser/.local/state/snapcraft/log/snapcraft-20250221-104557.421288.log
2025-02-21 10:45:57.421 Starting snapcraft, version 8.6.3
2025-02-21 10:45:57.422 Log verbosity level set to BRIEF
2025-02-21 10:45:57.422 Preparing application...
2025-02-21 10:45:57.422 Merging commands for group 'Lifecycle':
2025-02-21 10:45:57.422   - using application command for 'clean'.
2025-02-21 10:45:57.422   - using application command for 'pull'.
2025-02-21 10:45:57.422   - using application command for 'build'.
2025-02-21 10:45:57.422   - using application command for 'stage'.
2025-02-21 10:45:57.422   - using application command for 'prime'.
2025-02-21 10:45:57.422   - using application command for 'pack'.
2025-02-21 10:45:57.422 Merging commands for group 'Other':
2025-02-21 10:45:57.422 Configuring application...
2025-02-21 10:45:57.423 Setting up ConfigService
2025-02-21 10:45:57.433 Build plan: platform=None, build_for=None
2025-02-21 10:45:57.434 Loading project file '/home/azureuser/hello-world-snap/snapcraft.yaml'
2025-02-21 10:45:57.435 snapcraft internal error: KeyError('None')
2025-02-21 10:45:57.436 Traceback (most recent call last):
2025-02-21 10:45:57.436   File "/snap/snapcraft/13601/lib/python3.12/site-packages/craft_application/application.py", line 663, in run
2025-02-21 10:45:57.436     return_code = self._run_inner()
2025-02-21 10:45:57.436                   ^^^^^^^^^^^^^^^^^
2025-02-21 10:45:57.436   File "/snap/snapcraft/13601/lib/python3.12/site-packages/snapcraft/application.py", line 203, in _run_inner
2025-02-21 10:45:57.436     return_code = super()._run_inner()
2025-02-21 10:45:57.436                   ^^^^^^^^^^^^^^^^^^^^
2025-02-21 10:45:57.436   File "/snap/snapcraft/13601/lib/python3.12/site-packages/craft_application/application.py", line 634, in _run_inner
2025-02-21 10:45:57.436     self.services.project = self.get_project(
2025-02-21 10:45:57.436                             ^^^^^^^^^^^^^^^^^
2025-02-21 10:45:57.436   File "/snap/snapcraft/13601/lib/python3.12/site-packages/craft_application/application.py", line 363, in get_project
2025-02-21 10:45:57.436     self._full_build_plan = build_planner.get_build_plan()
2025-02-21 10:45:57.436                             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2025-02-21 10:45:57.436   File "/snap/snapcraft/13601/lib/python3.12/site-packages/snapcraft/models/project.py", line 1555, in get_build_plan
2025-02-21 10:45:57.436     effective_base = SNAPCRAFT_BASE_TO_PROVIDER_BASE[
2025-02-21 10:45:57.436                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2025-02-21 10:45:57.436 KeyError: 'None'
2025-02-21 10:45:57.436 Full execution log: '/home/azureuser/.local/state/snapcraft/log/snapcraft-20250221-104557.421288.log'
azureuser@linux-lab-virtualization:~/hello-world-snap$ nano snapcraft.yaml
```

2. **Create the python file**



```
  GNU nano 7.2
# hello.py
print("Hello, World!")
```

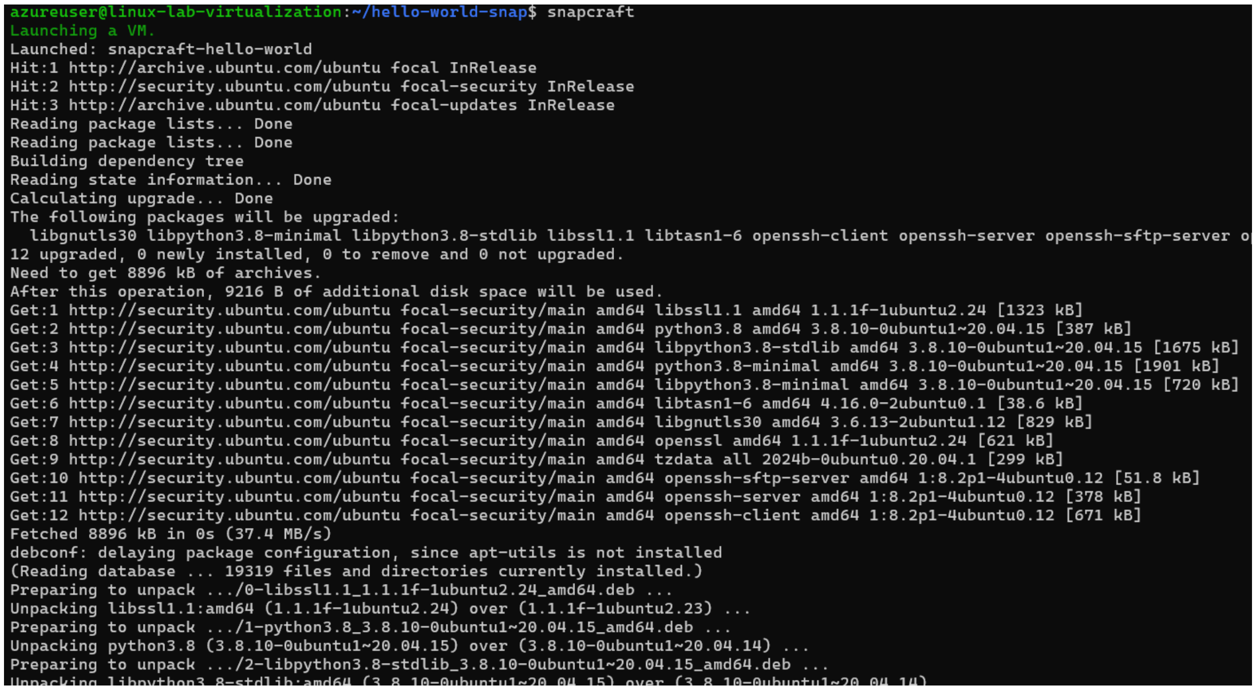
3. **Create the Snapcraft Configuration**



```
  GNU nano 7.2
name: hello-world
version: '1.0'
summary: A simple Hello World Python app
description: |
  A basic Python application that prints "Hello, World!"

confinement: strict

base: core20  # You can also use core18 or core22 based on your needs

apps:
  hello-world:
    command: bin/hello-world

parts:
  hello-world:
    source: .
    plugin: python
    python-version: python3
```

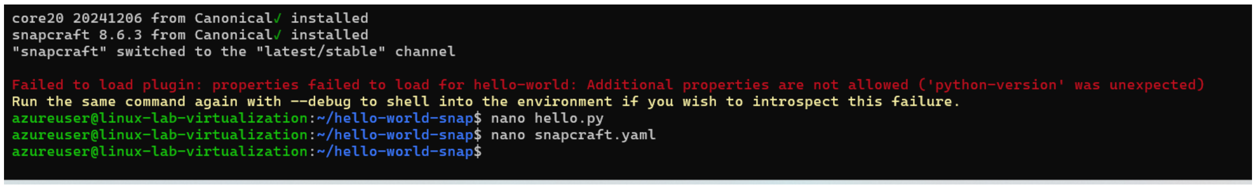
4. **Fix the error by adding base: core20**

5. **Create the snapcraft**

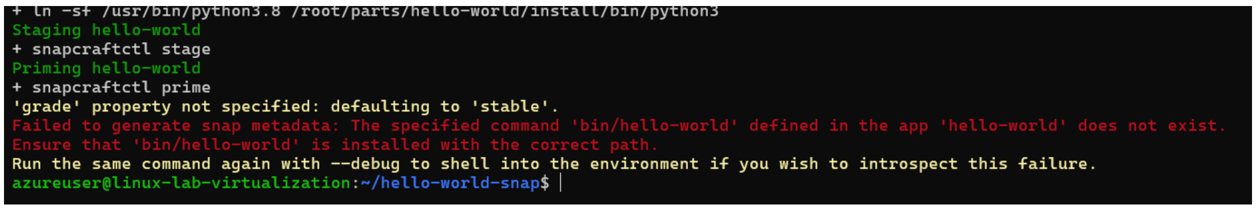


```
azureuser@linux-lab-virtualization:~/hello-world-snap$ snapcraft
Launching a VM.
Launched: snapcraft-hello-world
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  libgnutls30 libpython3.8-minimal libpython3.8-stdlib libssl1.1 libtasn1-6 openssh-client openssh-server openssh-sftp-server o
12 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 8896 kB of archives.
After this operation, 9216 B of additional disk space will be used.
Get:1 http://security.ubuntu.com/ubuntu focal-security/main amd64 libssl1.1 amd64 1.1.1f-1ubuntu2.24 [1323 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security/main amd64 python3.8 amd64 3.8.10-0ubuntu1~20.04.15 [387 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/main amd64 libpython3.8-stdlib amd64 3.8.10-0ubuntu1~20.04.15 [1675 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 python3.8-minimal amd64 3.8.10-0ubuntu1~20.04.15 [1901 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main amd64 libpython3.8-minimal amd64 3.8.10-0ubuntu1~20.04.15 [720 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 libtasn1-6 amd64 4.16.0-2ubuntu0.1 [38.6 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main amd64 libgnutls30 amd64 3.6.13-2ubuntu1.12 [829 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/main amd64 openssl amd64 1.1.1f-1ubuntu2.24 [621 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/main amd64 tzdata all 2024b-0ubuntu0.20.04.1 [299 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/main amd64 openssh-sftp-server amd64 1:8.2p1-4ubuntu0.12 [51.8 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/main amd64 openssh-server amd64 1:8.2p1-4ubuntu0.12 [378 kB]
Get:12 http://security.ubuntu.com/ubuntu focal-security/main amd64 openssh-client amd64 1:8.2p1-4ubuntu0.12 [671 kB]
Fetched 8896 kB in 0s (37.4 MB/s)
debconf: delaying package configuration, since apt-utils is not installed
(Reading database ... 19319 files and directories currently installed.)
Preparing to unpack .../0-libssl1.1_1.1.1f-1ubuntu2.24_amd64.deb ...
Unpacking libssl1.1:amd64 (1.1.1f-1ubuntu2.24) over (1.1.1f-1ubuntu2.23) ...
Preparing to unpack .../1-python3.8_3.8.10-0ubuntu1~20.04.15_amd64.deb ...
Unpacking python3.8 (3.8.10-0ubuntu1~20.04.15) over (3.8.10-0ubuntu1~20.04.14) ...
Preparing to unpack .../2-libpython3.8-stdlib_3.8.10-0ubuntu1~20.04.15_amd64.deb ...
Unpacking libpython3.8-stdlib:amd64 (3.8.10-0ubuntu1~20.04.15) over (3.8.10-0ubuntu1~20.04.14)
```

6. **Remove python-version to fix the error,**

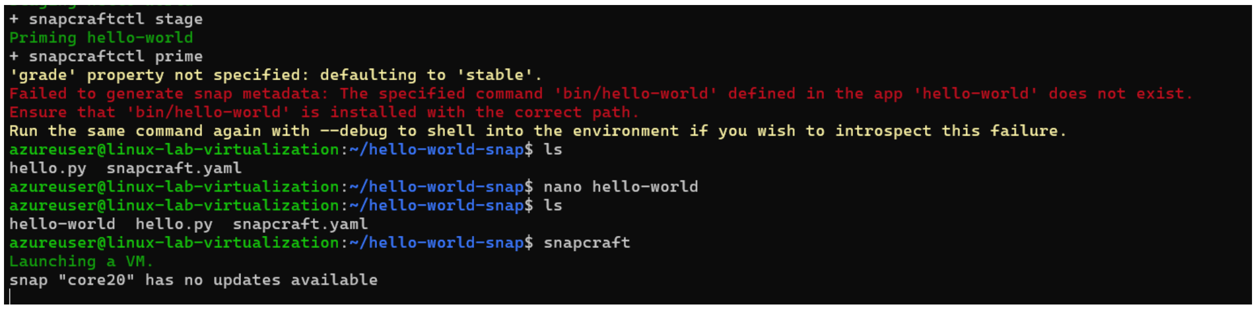


```
core20 20241206 from Canonical✓ installed
snapcraft 8.6.3 from Canonical✓ installed
"snapcraft" switched to the "latest/stable" channel

Failed to load plugin: properties failed to load for hello-world: Additional properties are not allowed ('python-version' was unexpected)
Run the same command again with --debug to shell into the environment if you wish to introspect this failure.
azureuser@linux-lab-virtualization:~/hello-world-snap$ nano hello.py
azureuser@linux-lab-virtualization:~/hello-world-snap$ nano snapcraft.yaml
azureuser@linux-lab-virtualization:~/hello-world-snap$
```



```
+ ln -sf /usr/bin/python3.8 /root/parts/hello-world/install/bin/python3
Staging hello-world
+ snapcraftctl stage
Priming hello-world
+ snapcraftctl prime
'grade' property not specified: defaulting to 'stable'.
Failed to generate snap metadata: The specified command 'bin/hello-world' defined in the app 'hello-world' does not exist.
Ensure that 'bin/hello-world' is installed with the correct path.
Run the same command again with --debug to shell into the environment if you wish to introspect this failure.
azureuser@linux-lab-virtualization:~/hello-world-snap$ |
```

7. **Check for the file hello-world and create the missing file. Re run the snap**

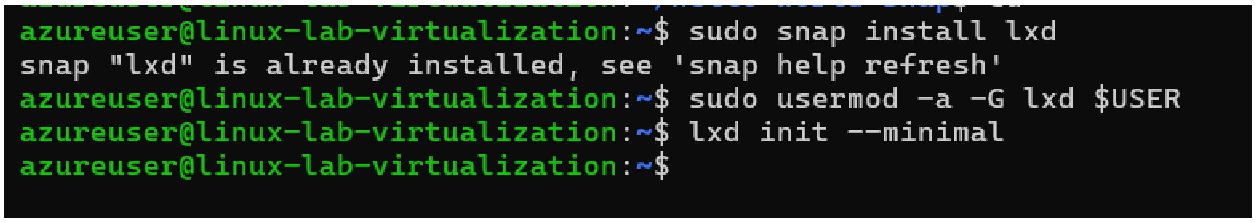


```
+ snapcraftctl stage
Priming hello-world
+ snapcraftctl prime
'grade' property not specified: defaulting to 'stable'.
Failed to generate snap metadata: The specified command 'bin/hello-world' defined in the app 'hello-world' does not exist.
Ensure that 'bin/hello-world' is installed with the correct path.
Run the same command again with --debug to shell into the environment if you wish to introspect this failure.
azureuser@linux-lab-virtualization:~/hello-world-snap$ ls
hello.py  snapcraft.yaml
azureuser@linux-lab-virtualization:~/hello-world-snap$ nano hello-world
azureuser@linux-lab-virtualization:~/hello-world-snap$ ls
hello-world  hello.py  snapcraft.yaml
azureuser@linux-lab-virtualization:~/hello-world-snap$ snapcraft
Launching a VM.
snap "core20" has no updates available
```

Method 2

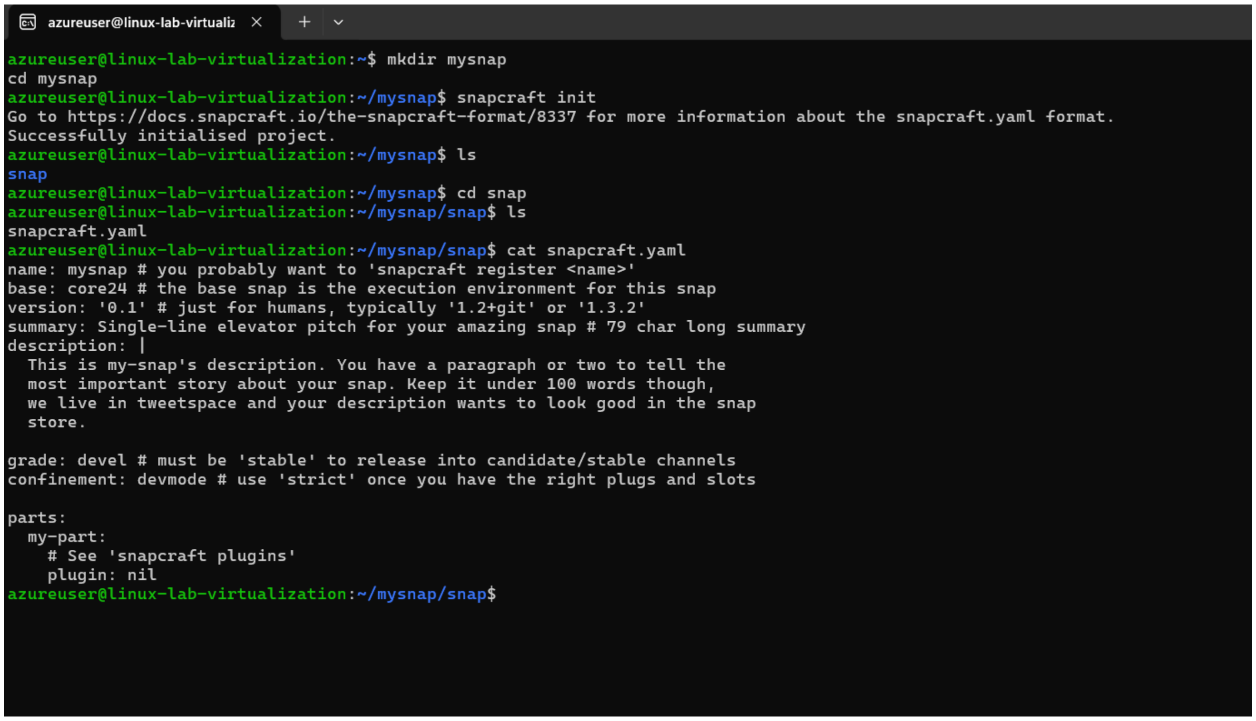
1. **Build snap ENV.**

sudo snap install lxd



```
azureuser@linux-lab-virtualization:~$ sudo snap install lxd
snap "lxd" is already installed, see 'snap help refresh'
azureuser@linux-lab-virtualization:~$ sudo usermod -a -G lxd $USER
azureuser@linux-lab-virtualization:~$ lxd init --minimal
azureuser@linux-lab-virtualization:~$
```

2. **Create YAML template**



```
azureuser@linux-lab-virtualization:~$ mkdir mysnap
cd mysnap
azureuser@linux-lab-virtualization:~/mysnap$ snapcraft init
Go to https://docs.snapcraft.io/the-snapcraft-format/8337 for more information about the snapcraft.yaml format.
Successfully initialised project.
azureuser@linux-lab-virtualization:~/mysnap$ ls
snap
azureuser@linux-lab-virtualization:~/mysnap$ cd snap
azureuser@linux-lab-virtualization:~/mysnap/snap$ ls
snapcraft.yaml
azureuser@linux-lab-virtualization:~/mysnap/snap$ cat snapcraft.yaml
name: mysnap # you probably want to 'snapcraft register <name>'
base: core24 # the base snap is the execution environment for this snap
version: '0.1' # just for humans, typically '1.2+git' or '1.3.2'
summary: Single-line elevator pitch for your amazing snap # 79 char long summary
description: |
  This is my-snap's description. You have a paragraph or two to tell the
  most important story about your snap. Keep it under 100 words though,
  we live in tweetspace and your description wants to look good in the snap
  store.

grade: devel # must be 'stable' to release into candidate/stable channels
confinement: devmode # use 'strict' once you have the right plugs and slots

parts:
  my-part:
    # See 'snapcraft plugins'
    plugin: nil
azureuser@linux-lab-virtualization:~/mysnap/snap$
```

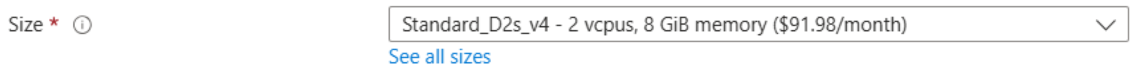
3. **Build template**

Assignment 7 - Virtualization

Machine specs on Azure:

1. For use of virtualization in Linux its better to use a VM with higher performance.

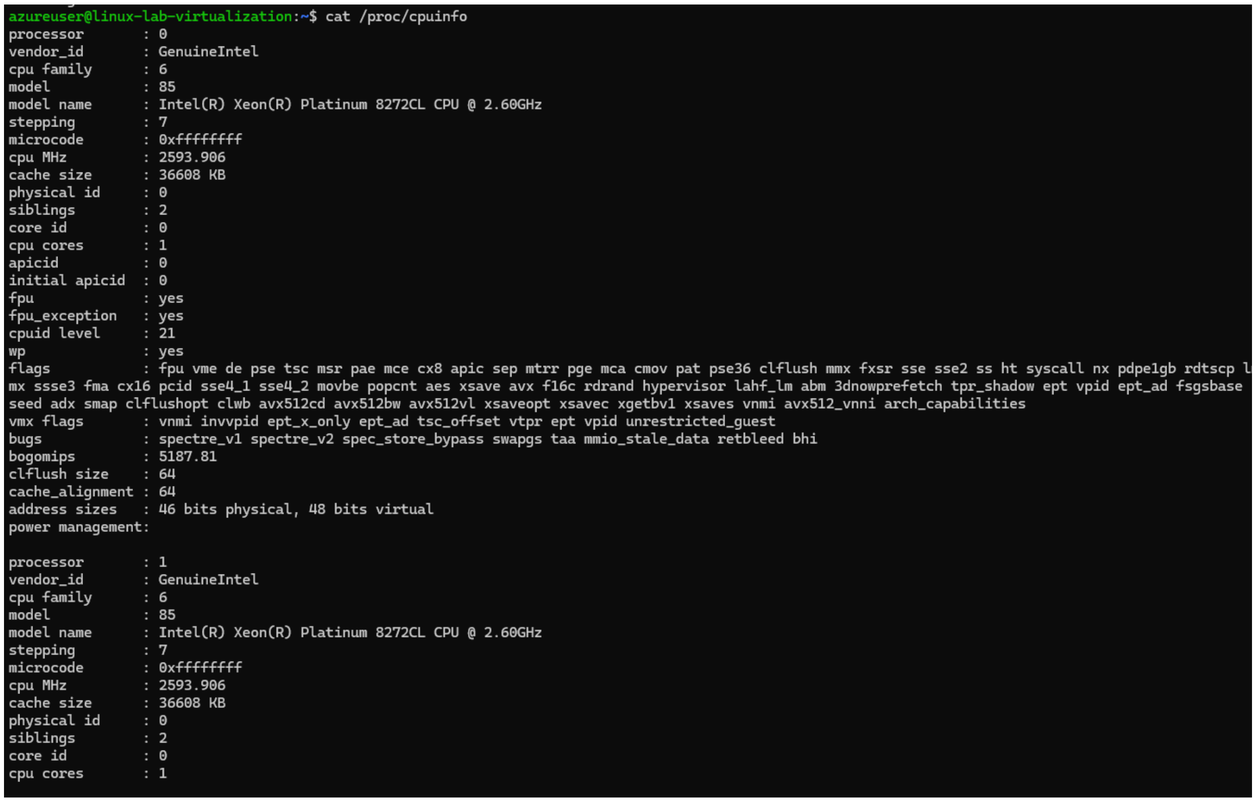
2. So we create a VM with high performance,



Size * : Standard_D2s_v4 - 2 vcpus, 8 GiB memory ($91.98/month)

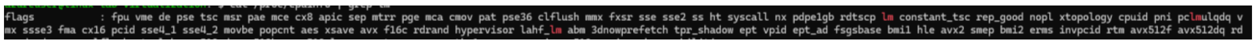
Checking virtualization support on machine:

1. Once the VM is created we can check the virtualization support by cpuinfo, **cat /proc/cpuinfo**



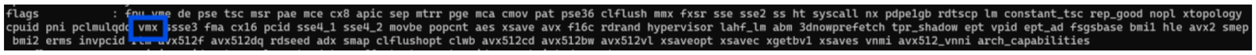
```
azureuser@linux-lab-virtualization:~$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel(R) Xeon(R) Platinum 8272CL CPU @ 2.60GHz
stepping        : 7
microcode       : 0xffffffff
cpu MHz         : 2593.906
cache size      : 36608 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 21
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp l
mx ssse3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow ept vpid ept_ad fsgsbase
seed adx smap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves vnmi avx512_vnni arch_capabilities
vmx flags       : vnmi invvpid ept_x_only ept_ad tsc_offset vtpr ept vpid unrestricted_guest
bugs            : spectre_v1 spectre_v2 spec_store_bypass swapgs taa mmio_stale_data retbleed bhi
bogomips        : 5187.81
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id       : GenuineIntel
cpu family      : 6
model           : 85
model name      : Intel(R) Xeon(R) Platinum 8272CL CPU @ 2.60GHz
stepping        : 7
microcode       : 0xffffffff
cpu MHz         : 2593.906
cache size      : 36608 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
```

2. Am I using 64 bit CPU/system [x86_64/AMD64/Intel64]? IN flags check for **lm**



```
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology cpuid pni pclmulqdq v
mx ssse3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow ept vpid ept_ad fsgsbase bmi1 hle avx2 smep bmi2 erms invpcid rtm avx512f avx512dq rd
```

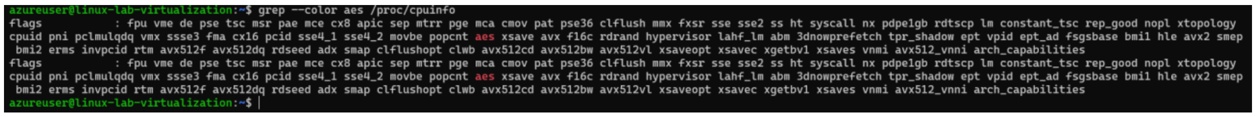
3. Do I have hardware virtualization support?
    - vmx – Intel VT-x, virtualization support enabled in BIOS.
    - svm – AMD SVM, virtualization enabled in BIOS.
    - Since we are using intel based system we only have VMX



```
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology
cpuid pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow ept vpid ept_ad fsgsbase bmi1 hle avx2 smep
 bmi2 erms invpcid rtm avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves vnmi avx512_vnni arch_capabilities
```

4.Do I have hardware AES/AES-NI advanced encryption support?

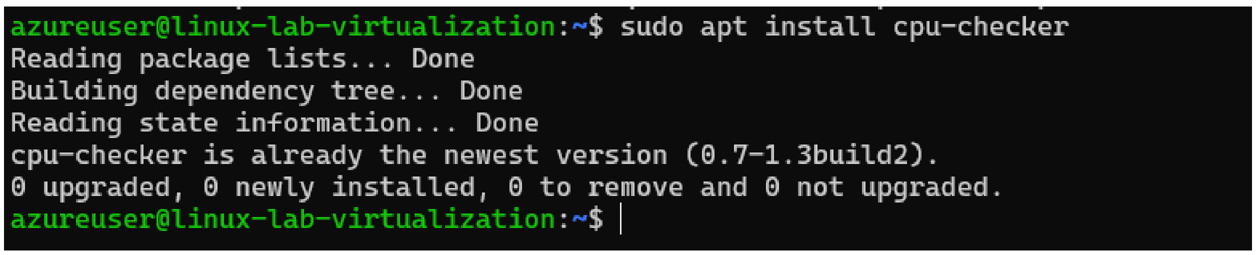
aes – Applications performing encryption and decryption using the Advanced Encryption Standard on Intel and AMD cpus.



```
azureuser@linux-lab-virtualization:~$ grep --color aes /proc/cpuinfo
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology
cpuid pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow ept vpid ept_ad fsgsbase bmi1 hle avx2 smep
 bmi2 erms invpcid rtm avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves vnmi avx512_vnni arch_capabilities
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology
cpuid pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow ept vpid ept_ad fsgsbase bmi1 hle avx2 smep
 bmi2 erms invpcid rtm avx512f avx512dq rdseed adx smap clflushopt clwb avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves vnmi avx512_vnni arch_capabilities
azureuser@linux-lab-virtualization:~$ |
```

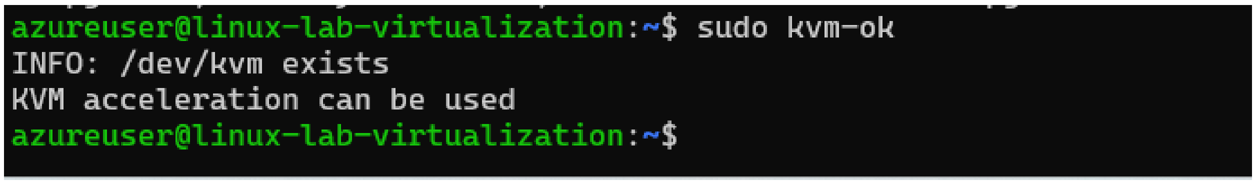
# Install kvm-ok on a Debian/Ubuntu

1. sudo apt install cpu-checker



```
azureuser@linux-lab-virtualization:~$ sudo apt install cpu-checker
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
cpu-checker is already the newest version (0.7-1.3build2).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
azureuser@linux-lab-virtualization:~$ |
```

2. sudo kvm-ok



```
azureuser@linux-lab-virtualization:~$ sudo kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
azureuser@linux-lab-virtualization:~$
```

Linux Virtualization Exercise

All exercises should be attempted on new Ubuntu 24.04 Linux installation, that supports nested virtualization.

Part 1: Introduction to virtualization concepts (30 minutes)

Research: Using sources (Sources linked at the end of the task), learn about the following concepts:

Virtualization

Virtualization is a technology that allows multiple operating systems or applications to run concurrently on a single physical machine by abstracting the underlying hardware resources. This is achieved through the use of hypervisors, which are software layers that create and manage virtual machines (VMs).

Hypervisor

A hypervisor is software or firmware that creates and manages virtual machines (VMs) on a physical host. It allows multiple VMs to run on a single physical machine by abstracting and allocating hardware resources like CPU, memory, and storage.

Virtual machines (VM)

A virtual machine (VM) is a software-based emulation of a physical computer that runs an operating system (OS) and applications just like a real machine. VMs are created and managed by a hypervisor, which allocates CPU, memory, storage, and other resources from the physical hardware to the VM.

Containers

A container is a lightweight, portable software unit that packages an application and its dependencies together so it can run consistently across different environments. Unlike virtual machines (VMs), containers share the host operating system's kernel, making them more efficient and faster.

The main differences between VMs and tanks

Summary: Summary in a text file briefly summarize the core differences between VMs and containers. Focus on their architecture, resource utilization, and insulation levels.

Feature Virtual	**Machines (VMs)**	**Containers**
Architecture	Each VM has a full OS, including a guest OS and virtualized hardware.	Containers share the host OS kernel and run as isolated processes.
Resource Usage	More resource-intensive due to separate OS instances.	Lightweight, as they share the OS kernel.
Startup Time	Slower, since the OS needs to boot.	Faster, since they don't require OS booting.
Isolation	Stronger isolation (separate OS for each VM).	Weaker isolation (process-level separation).
Portability	Less portable, depends on the hypervisor.	Highly portable across different environments.
Scalability	Requires more resources to scale.	Easily scalable due to lightweight nature.
Use Case	Best for running multiple OS environments or legacy applications.	Best for microservices, cloud applications, and fast deployments.

Part 2: Working with Multipass (1-2 hours)

Multipass is a command-line tool that allows you to quickly create and manage Ubuntu virtual machines.

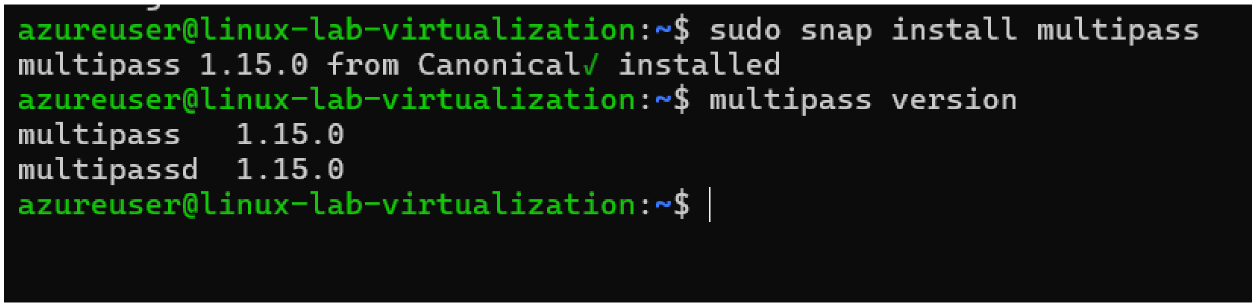
Installation: Follow the instructions on the Multipass website to install Multipass on your system. More information can be found in the source.

1. Start VM and install Multipass.

sudo snap install multipass

2. Verify the installation

multipass version



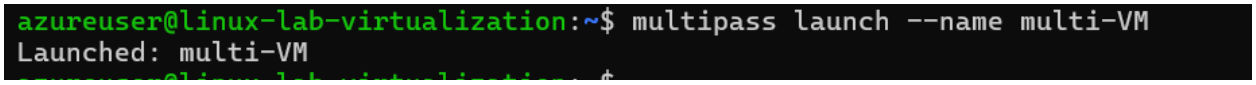
```
azureuser@linux-lab-virtualization:~$ sudo snap install multipass
multipass 1.15.0 from Canonical✓ installed
azureuser@linux-lab-virtualization:~$ multipass version
multipass   1.15.0
multipassd  1.15.0
azureuser@linux-lab-virtualization:~$ 
```

Basic commands: Use the multipass command-line interface and practice the following:

1. **multipass boot: Launch the default Ubuntu instance.**

multipass launch --name my-instance

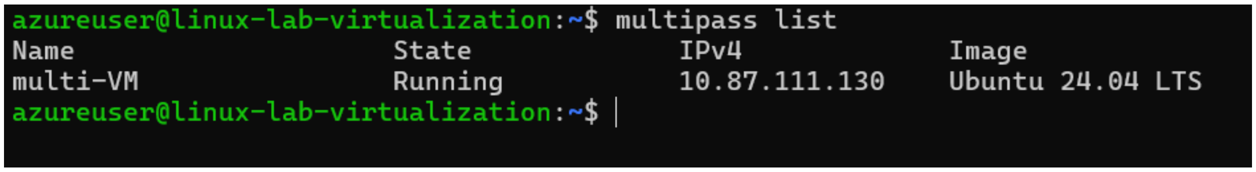
multipass launch --name multi-VM (Name of the instance is multi-VM)



```
azureuser@linux-lab-virtualization:~$ multipass launch --name multi-VM
Launched: multi-VM
```

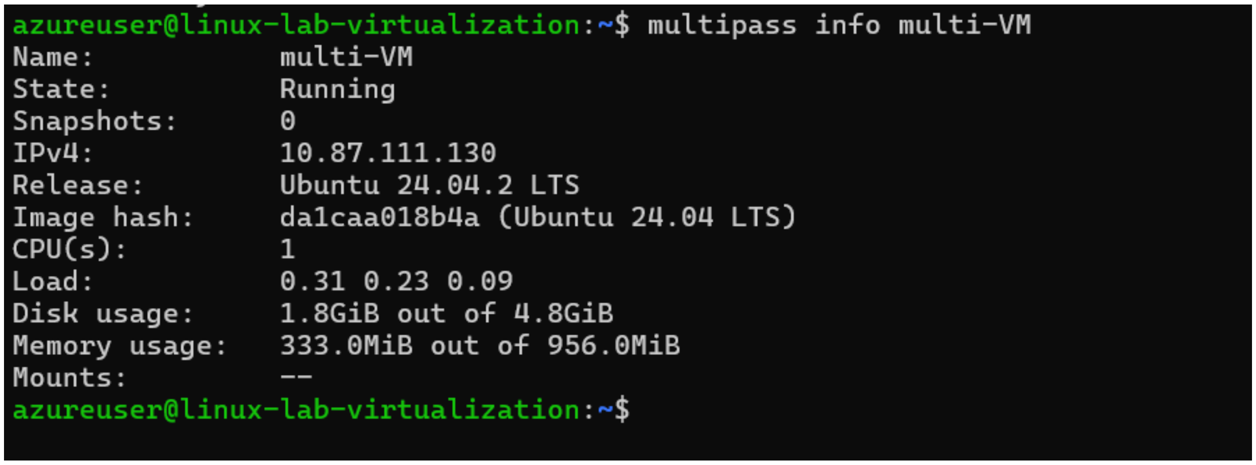
2. **multipass list: Lists all running instances.**

multipass list



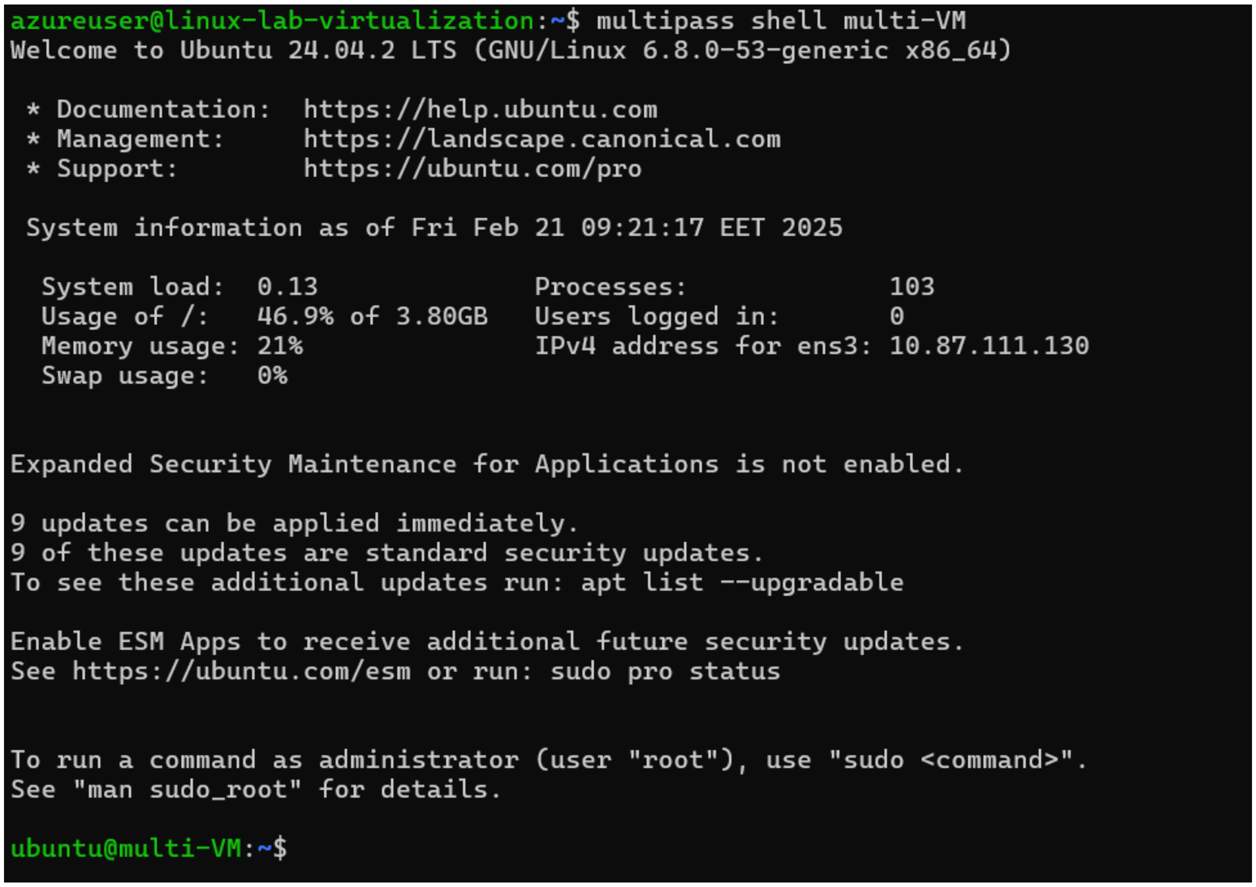
```
azureuser@linux-lab-virtualization:~$ multipass list
Name                    State             IPv4             Image
multi-VM                Running           10.87.111.130    Ubuntu 24.04 LTS
azureuser@linux-lab-virtualization:~$ 
```

3. **multipass info: View details about a specific instance.**



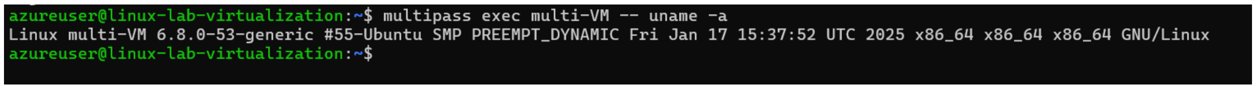
```
azureuser@linux-lab-virtualization:~$ multipass info multi-VM
Name:           multi-VM
State:          Running
Snapshots:      0
IPv4:           10.87.111.130
Release:        Ubuntu 24.04.2 LTS
Image hash:     da1caa018b4a (Ubuntu 24.04 LTS)
CPU(s):         1
Load:           0.31 0.23 0.09
Disk usage:     1.8GiB out of 4.8GiB
Memory usage:   333.0MiB out of 956.0MiB
Mounts:         --
azureuser@linux-lab-virtualization:~$
```

4. **multipass shell: Access to the shell of a running instance.**



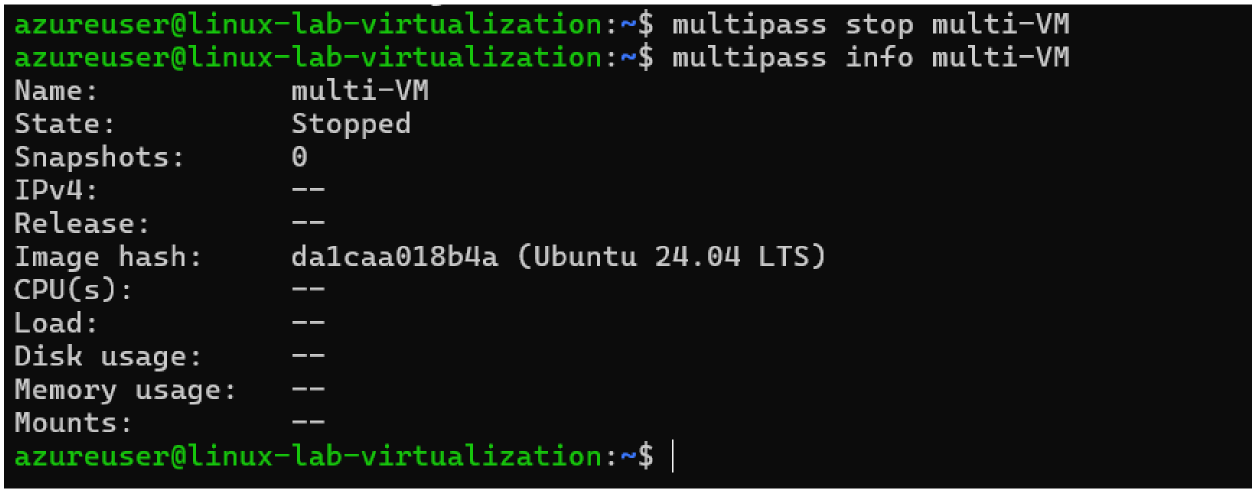
```
azureuser@linux-lab-virtualization:~$ multipass shell multi-VM
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-53-generic x86_64)

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

 System information as of Fri Feb 21 09:21:17 EET 2025

  System load:  0.13              Processes:             103
  Usage of /:   46.9% of 3.80GB   Users logged in:       0
  Memory usage: 21%               IPv4 address for ens3: 10.87.111.130
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

9 updates can be applied immediately.
9 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@multi-VM:~$
```

5. **multipass exec: Run the command on the instance.**



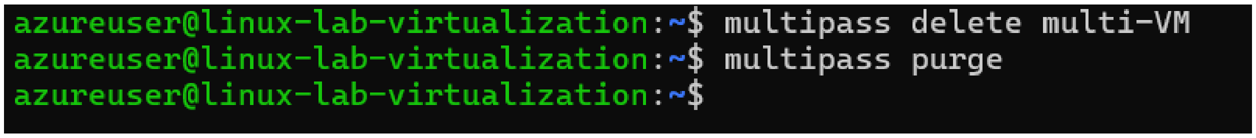
```
azureuser@linux-lab-virtualization:~$ multipass exec multi-VM -- uname -a
Linux multi-VM 6.8.0-53-generic #55-Ubuntu SMP PREEMPT_DYNAMIC Fri Jan 17 15:37:52 UTC 2025 x86_64 x86_64 x86_64 GNU/Linux
azureuser@linux-lab-virtualization:~$
```

6. **multipass stop: Stop the running instance.**



```
azureuser@linux-lab-virtualization:~$ multipass stop multi-VM
azureuser@linux-lab-virtualization:~$ multipass info multi-VM
Name:           multi-VM
State:          Stopped
Snapshots:      0
IPv4:           --
Release:        --
Image hash:     da1caa018b4a (Ubuntu 24.04 LTS)
CPU(s):         --
Load:           --
Disk usage:     --
Memory usage:   --
Mounts:         --
azureuser@linux-lab-virtualization:~$
```

7. **multipass delete: Delete the instance.**

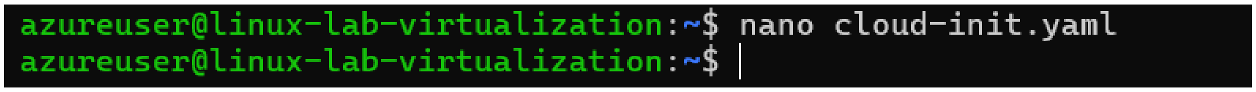


```
azureuser@linux-lab-virtualization:~$ multipass delete multi-VM
azureuser@linux-lab-virtualization:~$ multipass purge
azureuser@linux-lab-virtualization:~$
```

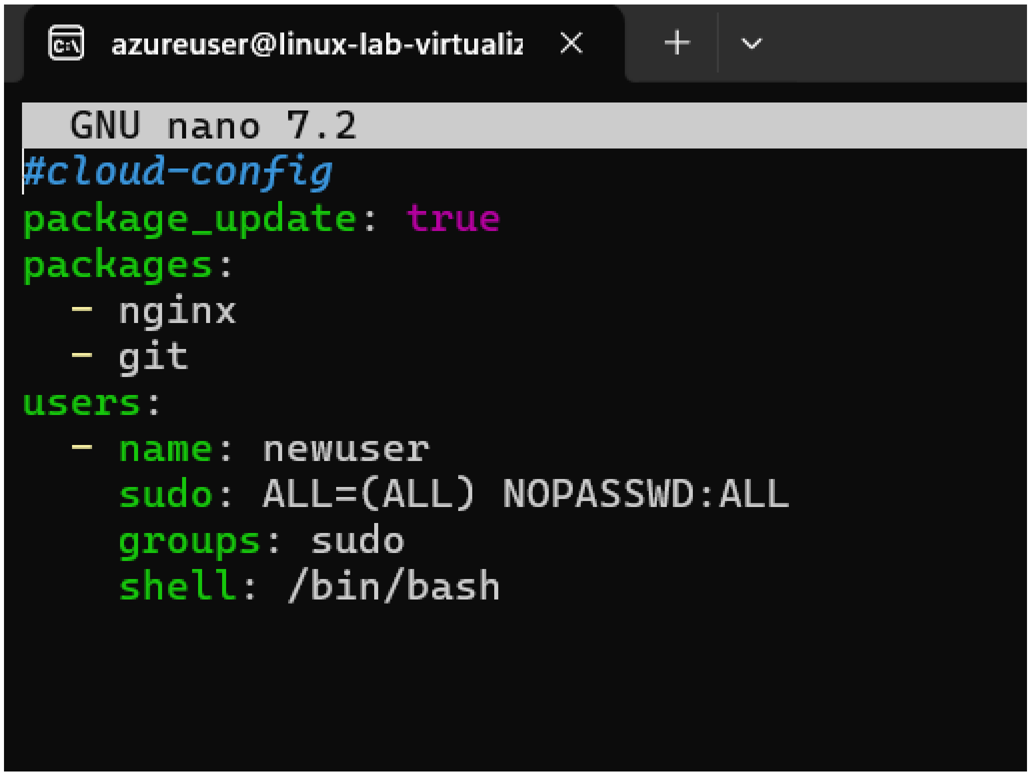
Cloud-init:Study: Learn about cloud-initi and how it can be used to configure virtual machines

Experiment: Create a cloud-init configuration file to customize the installation of a new instance.

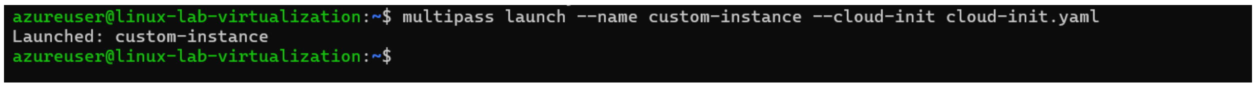
- For example, you can install specific packages or assign users.
- Start a new instance of Multipass using this configuration.



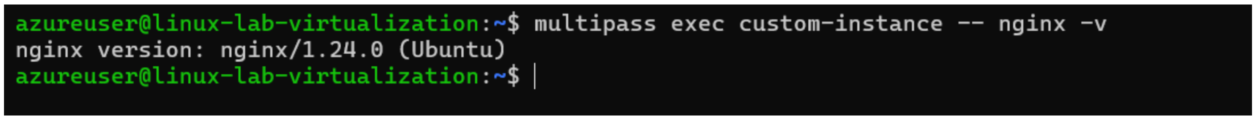
```
azureuser@linux-lab-virtualization:~$ nano cloud-init.yaml
azureuser@linux-lab-virtualization:~$
```



```
azureuser@linux-lab-virtualiz
  GNU nano 7.2
#cloud-config
package_update: true
packages:
  - nginx
  - git
users:
  - name: newuser
    sudo: ALL=(ALL) NOPASSWD:ALL
    groups: sudo
    shell: /bin/bash
```



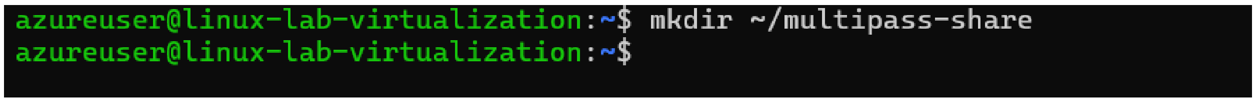
```
azureuser@linux-lab-virtualization:~$ multipass launch --name custom-instance --cloud-init cloud-init.yaml
Launched: custom-instance
azureuser@linux-lab-virtualization:~$
```



```
azureuser@linux-lab-virtualization:~$ multipass exec custom-instance -- nginx -v
nginx version: nginx/1.24.0 (Ubuntu)
azureuser@linux-lab-virtualization:~$
```

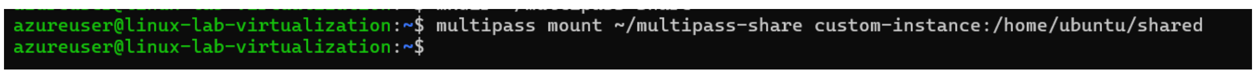
File sharing: Find out: Explore how to share files and folders between your host computer and Multipass instances

1. Create a shared folder on your Linux VM



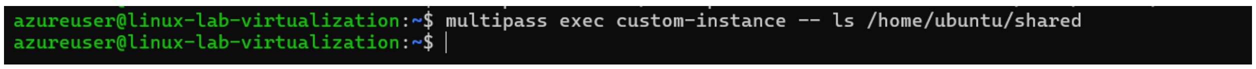
```
azureuser@linux-lab-virtualization:~$ mkdir ~/multipass-share
azureuser@linux-lab-virtualization:~$
```

2. Mount the folder inside the Multipass instance



```
azureuser@linux-lab-virtualization:~$ multipass mount ~/multipass-share custom-instance:/home/ubuntu/shared
azureuser@linux-lab-virtualization:~$
```

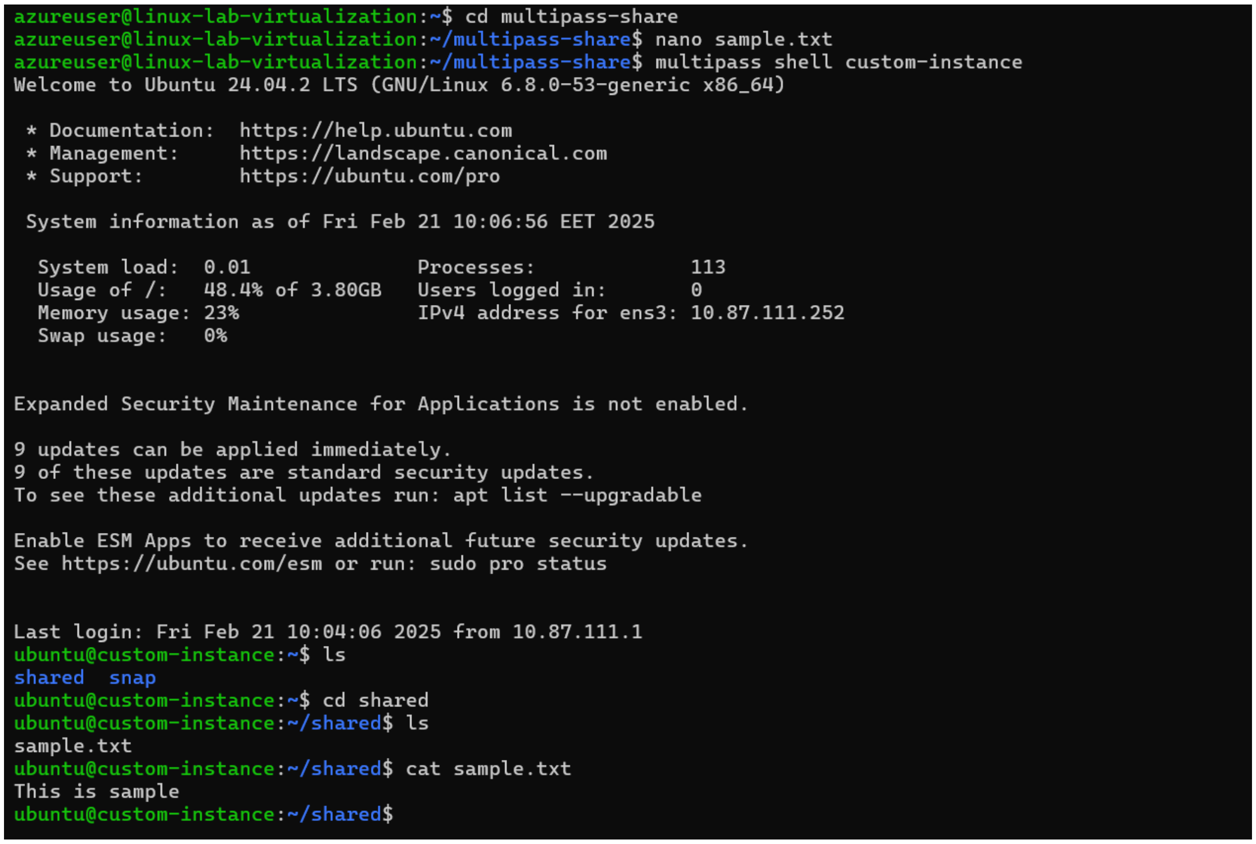
3. Check the shared folder inside the instance



```
azureuser@linux-lab-virtualization:~$ multipass exec custom-instance -- ls /home/ubuntu/shared
azureuser@linux-lab-virtualization:~$
```

Policy: Create a shared folder and access it from both your host and your Multipass instance.

1. Create sample.txt file inside VM and check availability inside custom-instance VM,



```
azureuser@linux-lab-virtualization:~$ cd multipass-share
azureuser@linux-lab-virtualization:~/multipass-share$ nano sample.txt
azureuser@linux-lab-virtualization:~/multipass-share$ multipass shell custom-instance
Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-53-generic x86_64)

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

 System information as of Fri Feb 21 10:06:56 EET 2025

  System load:  0.01              Processes:             113
  Usage of /:   48.4% of 3.80GB   Users logged in:       0
  Memory usage: 23%               IPv4 address for ens3: 10.87.111.252
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

9 updates can be applied immediately.
9 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


Last login: Fri Feb 21 10:04:06 2025 from 10.87.111.1
ubuntu@custom-instance:~$ ls
shared  snap
ubuntu@custom-instance:~$ cd shared
ubuntu@custom-instance:~/shared$ ls
sample.txt
ubuntu@custom-instance:~/shared$ cat sample.txt
This is sample
ubuntu@custom-instance:~/shared$
```

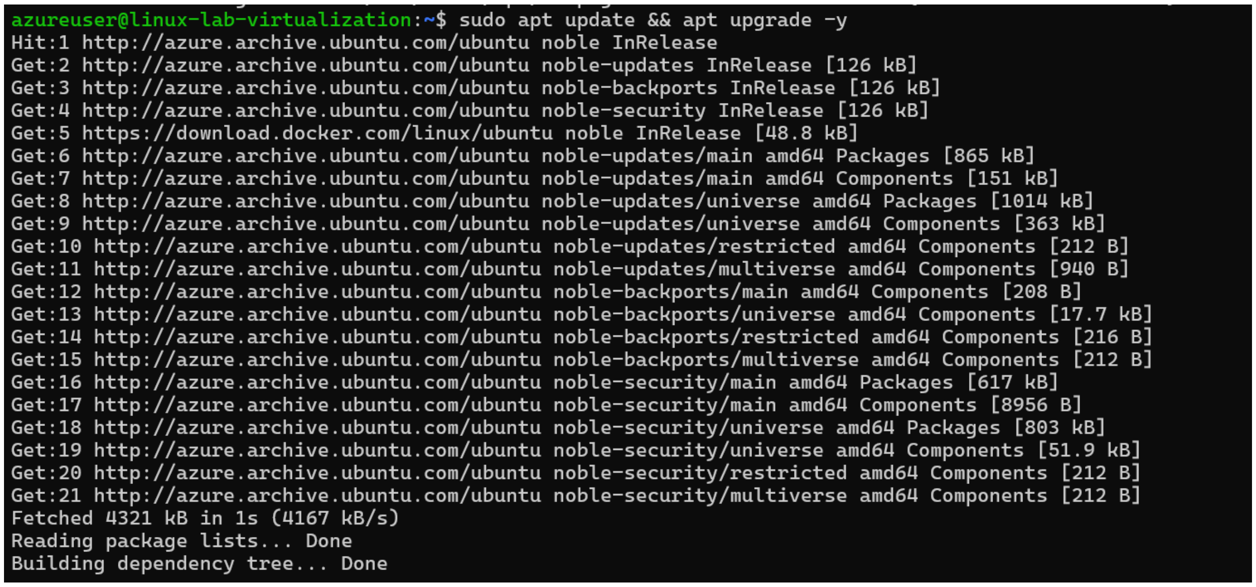
Part 3: Exploring LXD (1 hour)

LXD is another container and virtual machine management built on LXC, the so-called "Virtual Machine Management" of the Linux container. Runtime.

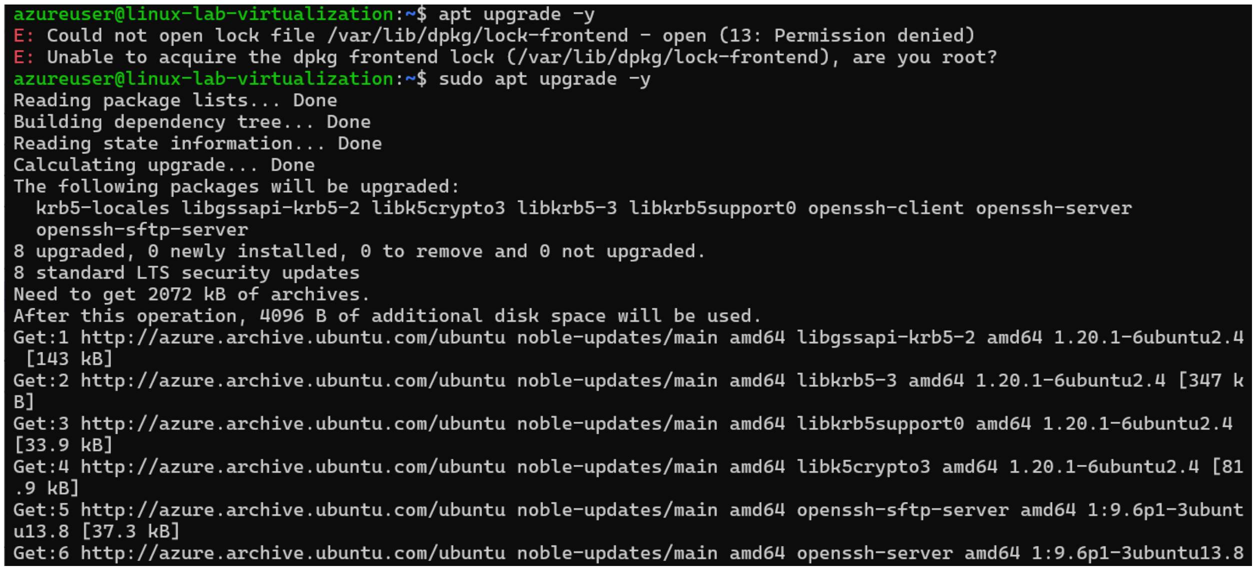
Study: Read about LXD and its features on LinuxContainers website Setup: Install LXD on your system, enable LXD Basic commands: Experiment with basic LXD commands to create, manage, and interact with containers. You can find the starting point in the source

Installing and managing LXD / LXC system

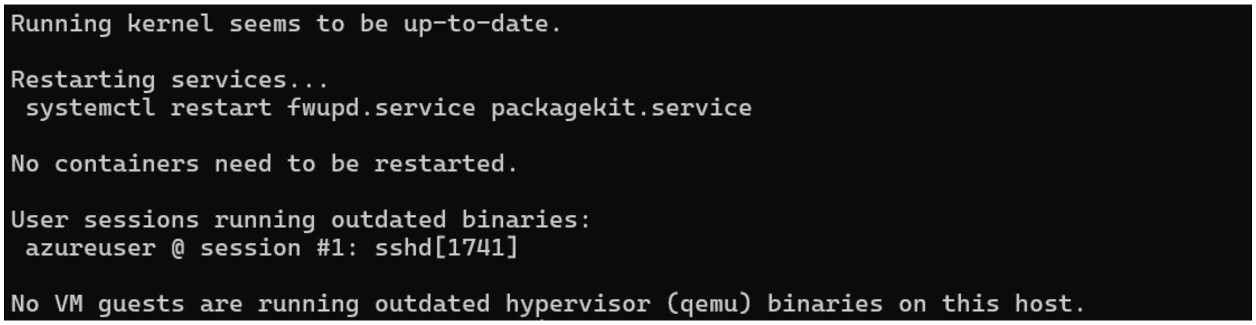
1. update your system by using apt: apt update && apt upgrade -y



```
azureuser@linux-lab-virtualization:~$ sudo apt update && apt upgrade -y
Hit:1 http://azure.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [865 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [151 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1014 kB]
Get:9 http://azure.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [363 kB]
Get:10 http://azure.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:11 http://azure.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:12 http://azure.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:13 http://azure.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [17.7 kB]
Get:14 http://azure.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:15 http://azure.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:16 http://azure.archive.ubuntu.com/ubuntu noble-security/main amd64 Packages [617 kB]
Get:17 http://azure.archive.ubuntu.com/ubuntu noble-security/main amd64 Components [8956 B]
Get:18 http://azure.archive.ubuntu.com/ubuntu noble-security/universe amd64 Packages [803 kB]
Get:19 http://azure.archive.ubuntu.com/ubuntu noble-security/universe amd64 Components [51.9 kB]
Get:20 http://azure.archive.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:21 http://azure.archive.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Fetched 4321 kB in 1s (4167 kB/s)
Reading package lists... Done
Building dependency tree... Done
```

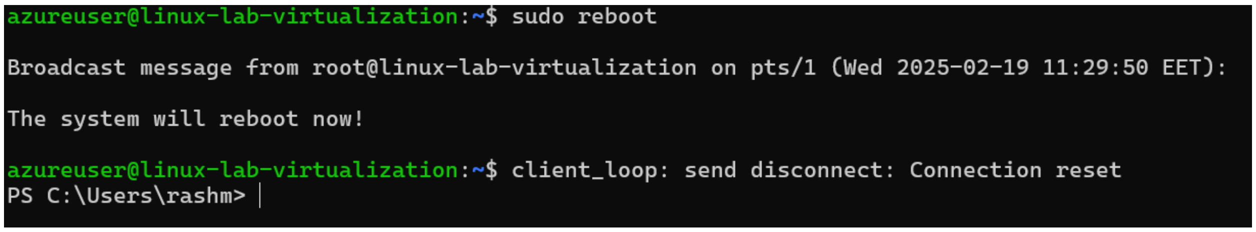


```
azureuser@linux-lab-virtualization:~$ apt upgrade -y
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?
azureuser@linux-lab-virtualization:~$ sudo apt upgrade -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  krb5-locales libgssapi-krb5-2 libk5crypto3 libkrb5-3 libkrb5support0 openssh-client openssh-server
  openssh-sftp-server
8 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
8 standard LTS security updates
Need to get 2072 kB of archives.
After this operation, 4096 B of additional disk space will be used.
Get:1 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 libgssapi-krb5-2 amd64 1.20.1-6ubuntu2.4
 [143 kB]
Get:2 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 libkrb5-3 amd64 1.20.1-6ubuntu2.4 [347 k
B]
Get:3 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 libkrb5support0 amd64 1.20.1-6ubuntu2.4
[33.9 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 libk5crypto3 amd64 1.20.1-6ubuntu2.4 [81
.9 kB]
Get:5 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 openssh-sftp-server amd64 1:9.6p1-3ubunt
u13.8 [37.3 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu noble-updates/main amd64 openssh-server amd64 1:9.6p1-3ubuntu13.8
```

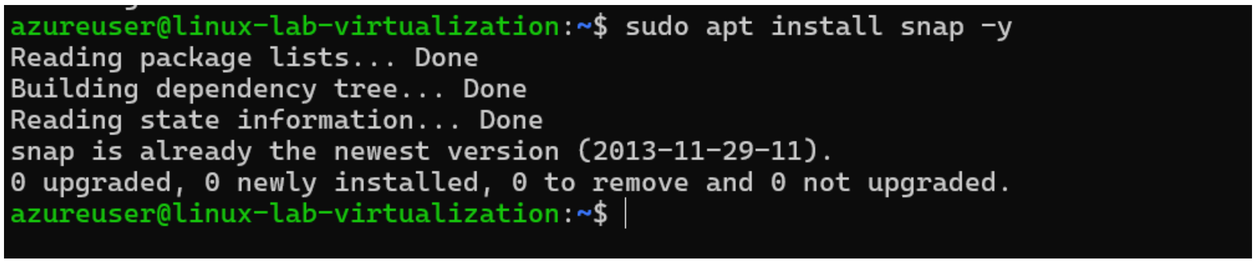


```
Running kernel seems to be up-to-date.

Restarting services...
 systemctl restart fwupd.service packagekit.service

No containers need to be restarted.

User sessions running outdated binaries:
 azureuser @ session #1: sshd[1741]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
```

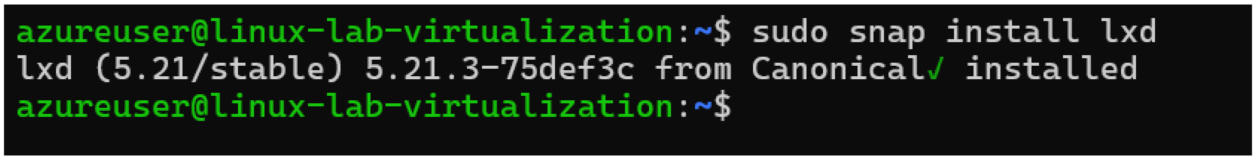
2. check if you just updated your kernel or other systems needing a complete system reboot, and if so, reboot: sudo reboot


```
azureuser@linux-lab-virtualization:~$ sudo reboot

Broadcast message from root@linux-lab-virtualization on pts/1 (Wed 2025-02-19 11:29:50 EET):

The system will reboot now!

azureuser@linux-lab-virtualization:~$ client_loop: send disconnect: Connection reset
PS C:\Users\rashm>
```

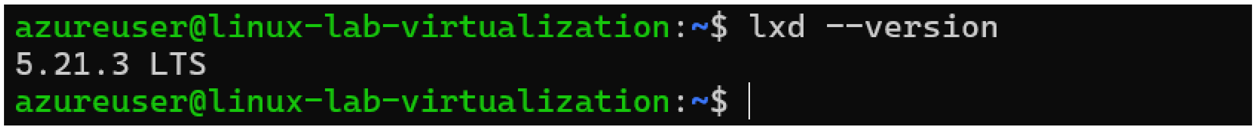
3. Install Snap: sudo apt install snap -y


```
azureuser@linux-lab-virtualization:~$ sudo apt install snap -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
snap is already the newest version (2013-11-29-11).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
azureuser@linux-lab-virtualization:~$
```

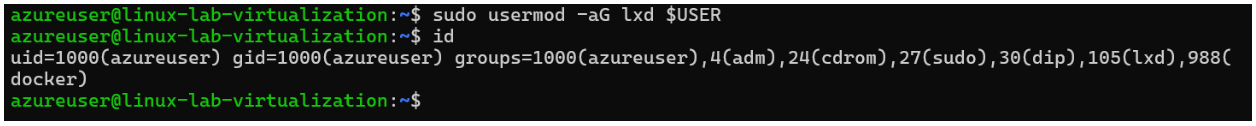
4. install lxd using snap: sudo snap install lxd


```
azureuser@linux-lab-virtualization:~$ sudo snap install lxd
lxd (5.21/stable) 5.21.3-75def3c from Canonical✓ installed
azureuser@linux-lab-virtualization:~$
```

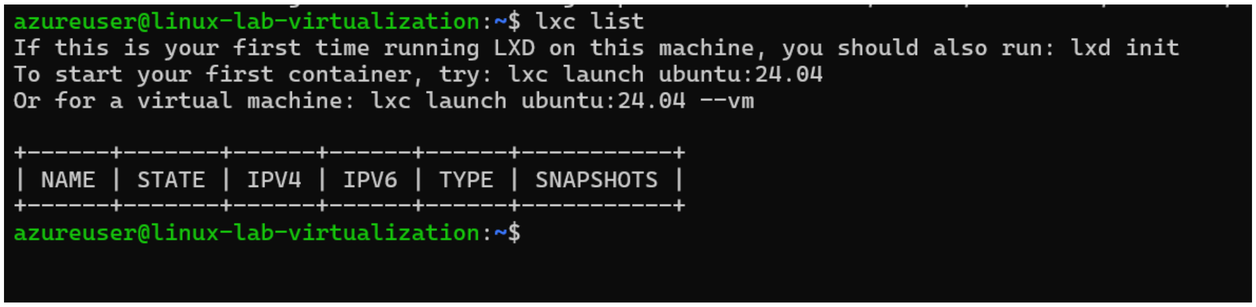
5. check lxd version and installation: lxd --version


```
azureuser@linux-lab-virtualization:~$ lxd --version
5.21.3 LTS
azureuser@linux-lab-virtualization:~$
```

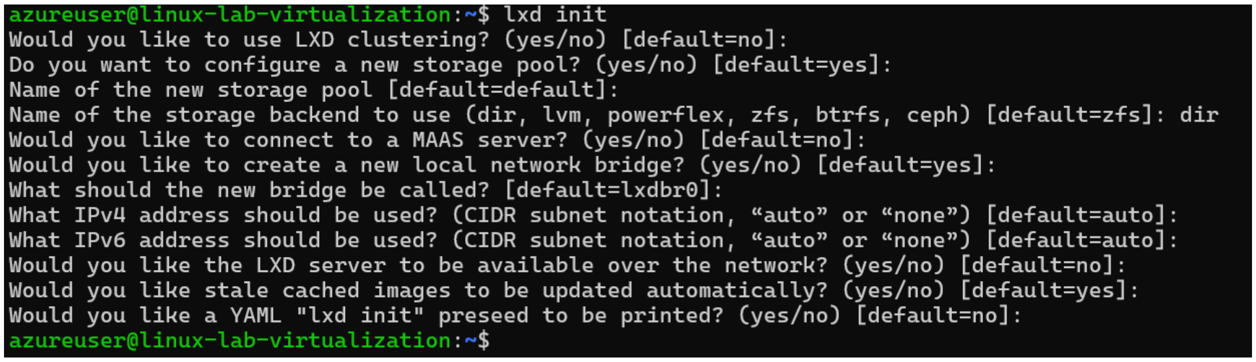
6. check that your user belongs to LXD group: id, and look for LXD. If you do not find lxd group, add user to it: sudo usermod -aG lxd $USER


```
azureuser@linux-lab-virtualization:~$ sudo usermod -aG lxd $USER
azureuser@linux-lab-virtualization:~$ id
uid=1000(azureuser) gid=1000(azureuser) groups=1000(azureuser),4(adm),24(cdrom),27(sudo),30(dip),105(lxd),988(
docker)
azureuser@linux-lab-virtualization:~$
```

7. check lxc system for listing of machines and containers: lxc list


```
azureuser@linux-lab-virtualization:~$ lxc list
If this is your first time running LXD on this machine, you should also run: lxd init
To start your first container, try: lxc launch ubuntu:24.04
Or for a virtual machine: lxc launch ubuntu:24.04 --vm

+------+-------+------+------+------+-----------+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+------+-------+------+------+------+-----------+
azureuser@linux-lab-virtualization:~$
```

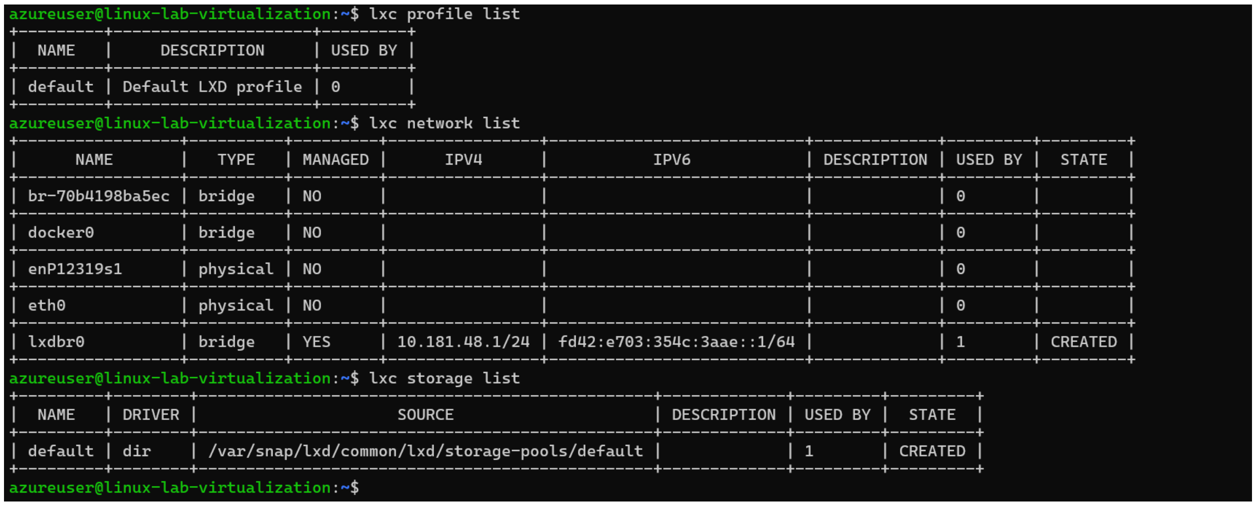
8. initialize xld, to configure system to your environment: lxd init. Make sure you run this as basic user, without root / sudo privileges it will ask series of questions, so based on your requirement answer them.
    - When it asks about clustering, choose 'no' (unless you're setting up a cluster)
    - For storage, I recommend saying 'yes' to a new storage pool
    - The 'dir' backend is fine for beginners
    - Say 'yes' to a network bridge
    - Choose 'no' to make LXD sever available over the network.



```
azureuser@linux-lab-virtualization:~$ lxd init
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (dir, lvm, powerflex, zfs, btrfs, ceph) [default=zfs]: dir
Would you like to connect to a MAAS server? (yes/no) [default=no]:
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, "auto" or "none") [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, "auto" or "none") [default=auto]:
Would you like the LXD server to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]:
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
azureuser@linux-lab-virtualization:~$
```

9. Once lxd is initialized successfully, we can verify the information using following set of commands:

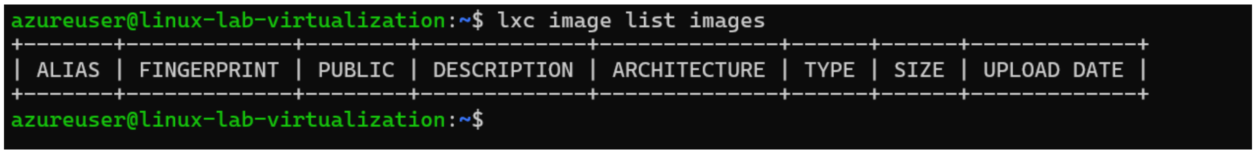
$lxc profile list $lxc network list $lxc storage list



```
azureuser@linux-lab-virtualization:~$ lxc profile list
+---------+---------------------+---------+
|  NAME   |     DESCRIPTION     | USED BY |
+---------+---------------------+---------+
| default | Default LXD profile | 0       |
+---------+---------------------+---------+
azureuser@linux-lab-virtualization:~$ lxc network list
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
|      NAME       |   TYPE   | MANAGED |      IPV4      |           IPV6            | DESCRIPTION | USED BY |  STATE  |
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
| br-70b4198ba5ec | bridge   | NO      |                |                           |             | 0       |         |
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
| docker0         | bridge   | NO      |                |                           |             | 0       |         |
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
| enP12319s1      | physical | NO      |                |                           |             | 0       |         |
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
| eth0            | physical | NO      |                |                           |             | 0       |         |
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
| lxdbr0          | bridge   | YES     | 10.181.48.1/24 | fd42:e703:354c:3aae::1/64 |             | 1       | CREATED |
+-----------------+----------+---------+----------------+---------------------------+-------------+---------+---------+
azureuser@linux-lab-virtualization:~$ lxc storage list
+---------+--------+------------------------------------------------+-------------+---------+---------+
|  NAME   | DRIVER |                     SOURCE                     | DESCRIPTION | USED BY |  STATE  |
+---------+--------+------------------------------------------------+-------------+---------+---------+
| default | dir    | /var/snap/lxd/common/lxd/storage-pools/default |             | 1       | CREATED |
+---------+--------+------------------------------------------------+-------------+---------+---------+
azureuser@linux-lab-virtualization:~$
```

10. In order to list all available images, run:

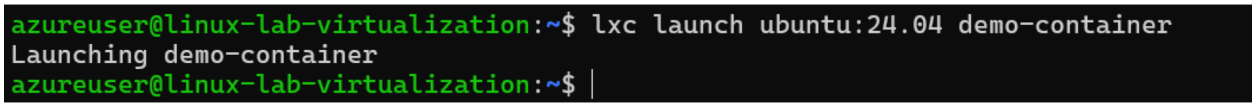
$lxc image list images:



```
azureuser@linux-lab-virtualization:~$ lxc image list images
+-------+-------------+--------+-------------+--------------+------+------+-------------+
| ALIAS | FINGERPRINT | PUBLIC | DESCRIPTION | ARCHITECTURE | TYPE | SIZE | UPLOAD DATE |
+-------+-------------+--------+-------------+--------------+------+------+-------------+
azureuser@linux-lab-virtualization:~$
```

11. Create your first container:

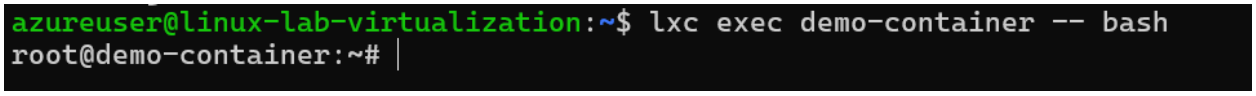
$lxc launch ubuntu:24.04 demo-container



```
azureuser@linux-lab-virtualization:~$ lxc launch ubuntu:24.04 demo-container
Launching demo-container
azureuser@linux-lab-virtualization:~$
```

12. Access the console of container. Run:

$ lxc exec demo-container -- bash



```
azureuser@linux-lab-virtualization:~$ lxc exec demo-container -- bash
root@demo-container:~#
```

Part 4: How to Stick Apps with Docker (1 hour)

Docker is a platform for building, deploying and managing container applications

Installation: Install Docker Desktop on your system if you want to familiarize yourself with this section, You can find installation information for Docker Engine in the sources.

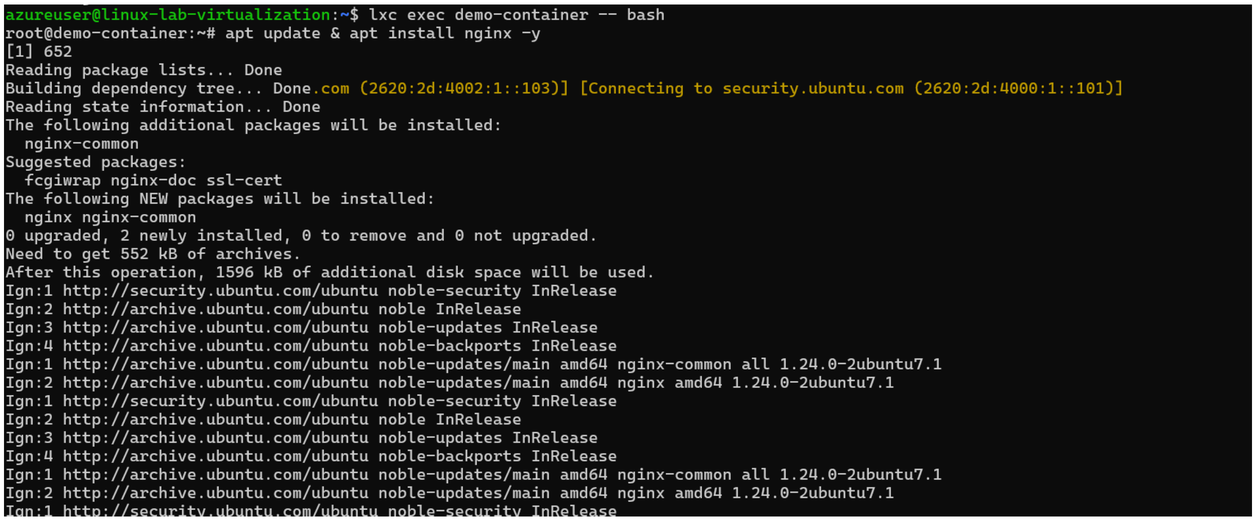
Basic concepts: Research: Explore core Docker concepts such as images, containers, and Dockerfiles. Source.

Experiment: Follow the "Docker Workshop" to get hands-on experience with Docker.

Installing and managing Docker engine based system

Basic steps:

Follow good instructions from Docker, at https://docs.docker.com/



```
azureuser@linux-lab-virtualization:~$ lxc exec demo-container -- bash
root@demo-container:~# apt update & apt install nginx -y
[1] 652
Reading package lists... Done
Building dependency tree... Done.com (2620:2d:4002:1::103)] [Connecting to security.ubuntu.com (2620:2d:4000:1::101)]
Reading state information... Done
The following additional packages will be installed:
  nginx-common
Suggested packages:
  fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
  nginx nginx-common
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 552 kB of archives.
After this operation, 1596 kB of additional disk space will be used.
Ign:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:2 http://archive.ubuntu.com/ubuntu noble InRelease
Ign:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Ign:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 nginx-common all 1.24.0-2ubuntu7.1
Ign:2 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 nginx amd64 1.24.0-2ubuntu7.1
Ign:1 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:2 http://archive.ubuntu.com/ubuntu noble InRelease
Ign:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Ign:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 nginx-common all 1.24.0-2ubuntu7.1
Ign:2 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 nginx amd64 1.24.0-2ubuntu7.1
Ign:1 http://security.ubuntu.com/ubuntu noble-security InRelease
```

More specifically from Ubuntu engine install instructions for Ubuntu system, and using convenience script:

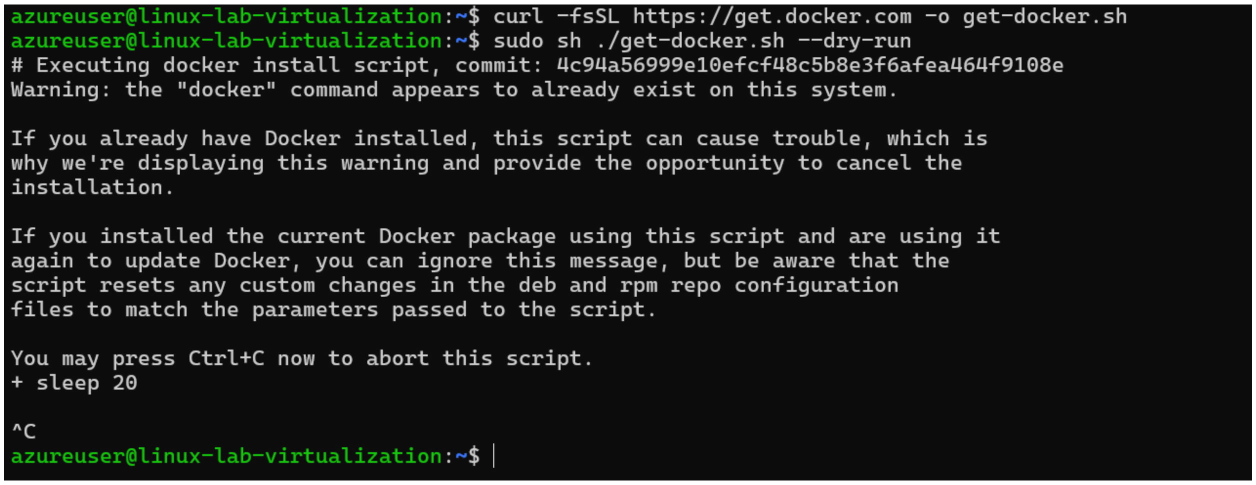
https://docs.docker.com/engine/install/ubuntu/#install-using-the-convenience-script

curl -fsSL https://get.docker.com -o get-docker.sh

sudo sh get-docker.sh

Executing docker install script, commit: 7cae5f8b0decc17d6571f9f52eb840fbc13b2737

<…>



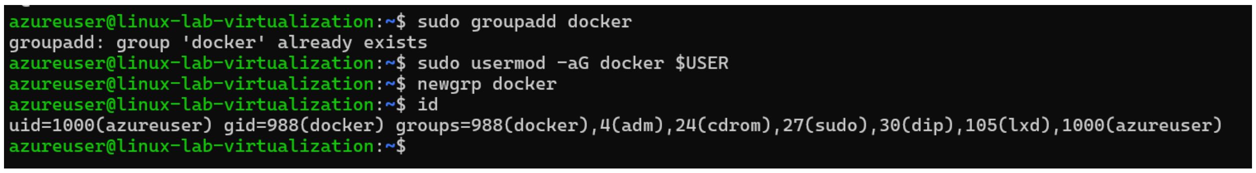
```
azureuser@linux-lab-virtualization:~$ curl -fsSL https://get.docker.com -o get-docker.sh
azureuser@linux-lab-virtualization:~$ sudo sh ./get-docker.sh --dry-run
# Executing docker install script, commit: 4c94a56999e10efcf48c5b8e3f6afea464f9108e
Warning: the "docker" command appears to already exist on this system.

If you already have Docker installed, this script can cause trouble, which is
why we're displaying this warning and provide the opportunity to cancel the
installation.

If you installed the current Docker package using this script and are using it
again to update Docker, you can ignore this message, but be aware that the
script resets any custom changes in the deb and rpm repo configuration
files to match the parameters passed to the script.

You may press Ctrl+C now to abort this script.
+ sleep 20

^C
azureuser@linux-lab-virtualization:~$
```

post installation guide tells us to make user part of Docker group on Linux machine:

sudo groupadd docker

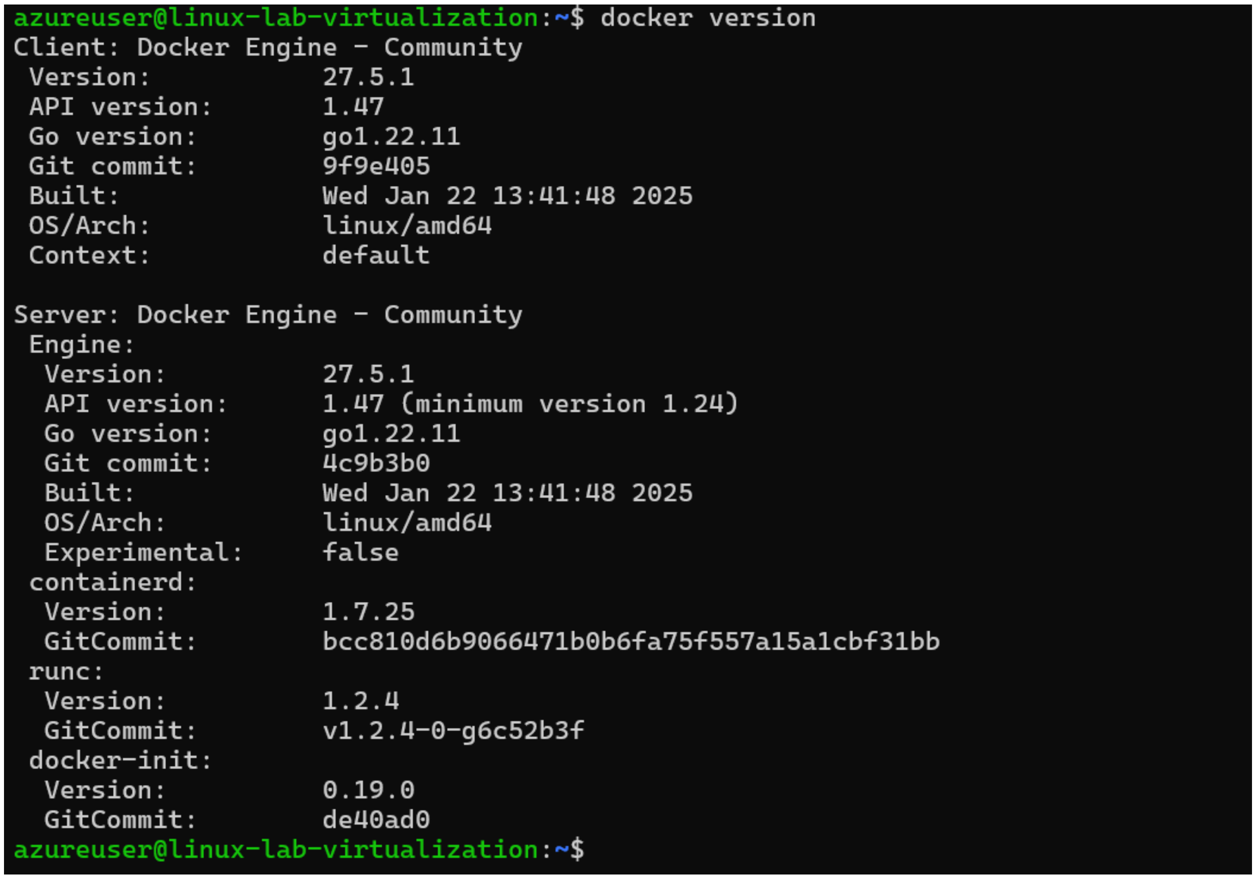
sudo usermod -aG docker $USER

newgrp docker



```
azureuser@linux-lab-virtualization:~$ sudo groupadd docker
groupadd: group 'docker' already exists
azureuser@linux-lab-virtualization:~$ sudo usermod -aG docker $USER
azureuser@linux-lab-virtualization:~$ newgrp docker
azureuser@linux-lab-virtualization:~$ id
uid=1000(azureuser) gid=988(docker) groups=988(docker),4(adm),24(cdrom),27(sudo),30(dip),105(lxd),1000(azureuser)
azureuser@linux-lab-virtualization:~$
```

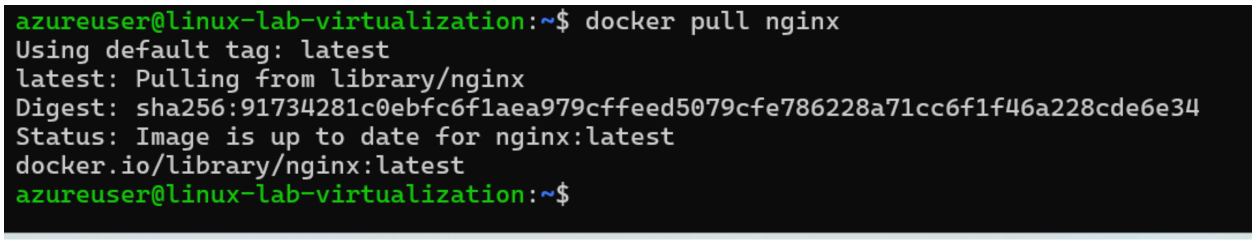
check docker version



```
azureuser@linux-lab-virtualization:~$ docker version
Client: Docker Engine - Community
 Version:           27.5.1
 API version:       1.47
 Go version:        go1.22.11
 Git commit:        9f9e405
 Built:             Wed Jan 22 13:41:48 2025
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          27.5.1
  API version:      1.47 (minimum version 1.24)
  Go version:       go1.22.11
  Git commit:       4c9b3b0
  Built:            Wed Jan 22 13:41:48 2025
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.7.25
  GitCommit:        bcc810d6b9066471b0b6fa75f557a15a1cbf31bb
 runc:
  Version:          1.2.4
  GitCommit:        v1.2.4-0-g6c52b3f
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
azureuser@linux-lab-virtualization:~$
```

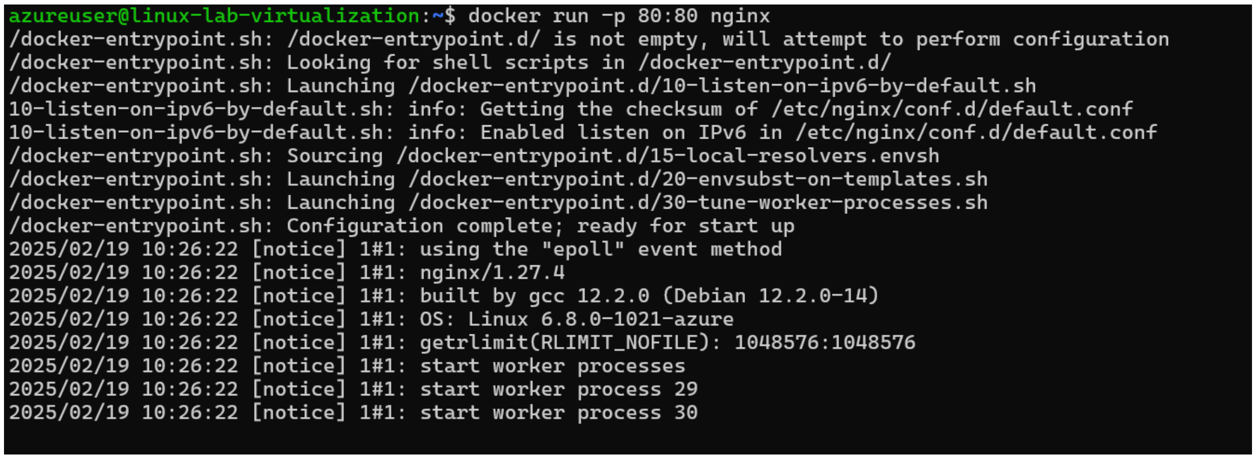
Run Nginx on docker:

1. Get the latest Nginx



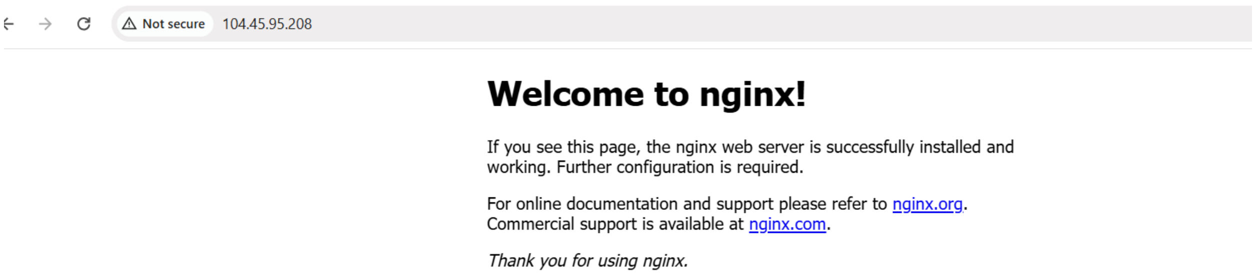
```
azureuser@linux-lab-virtualization:~$ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
Digest: sha256:91734281c0ebfc6f1aea979cffeed5079cfe786228a71cc6f1f46a228cde6e34
Status: Image is up to date for nginx:latest
docker.io/library/nginx:latest
azureuser@linux-lab-virtualization:~$
```

2. Start docker nginx image
   - docker run -p 80:80 nginx
   - docker run -d -p 80:80 nginx (To run on background use -d tag)



```
azureuser@linux-lab-virtualization:~$ docker run -p 80:80 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/02/19 10:26:22 [notice] 1#1: using the "epoll" event method
2025/02/19 10:26:22 [notice] 1#1: nginx/1.27.4
2025/02/19 10:26:22 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/02/19 10:26:22 [notice] 1#1: OS: Linux 6.8.0-1021-azure
2025/02/19 10:26:22 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/02/19 10:26:22 [notice] 1#1: start worker processes
2025/02/19 10:26:22 [notice] 1#1: start worker process 29
2025/02/19 10:26:22 [notice] 1#1: start worker process 30
```

3. If we open the IP of our virtual machine we can see Nginx is running,



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Part 5: Snaps for Self-Contained Applications (30 minutes)

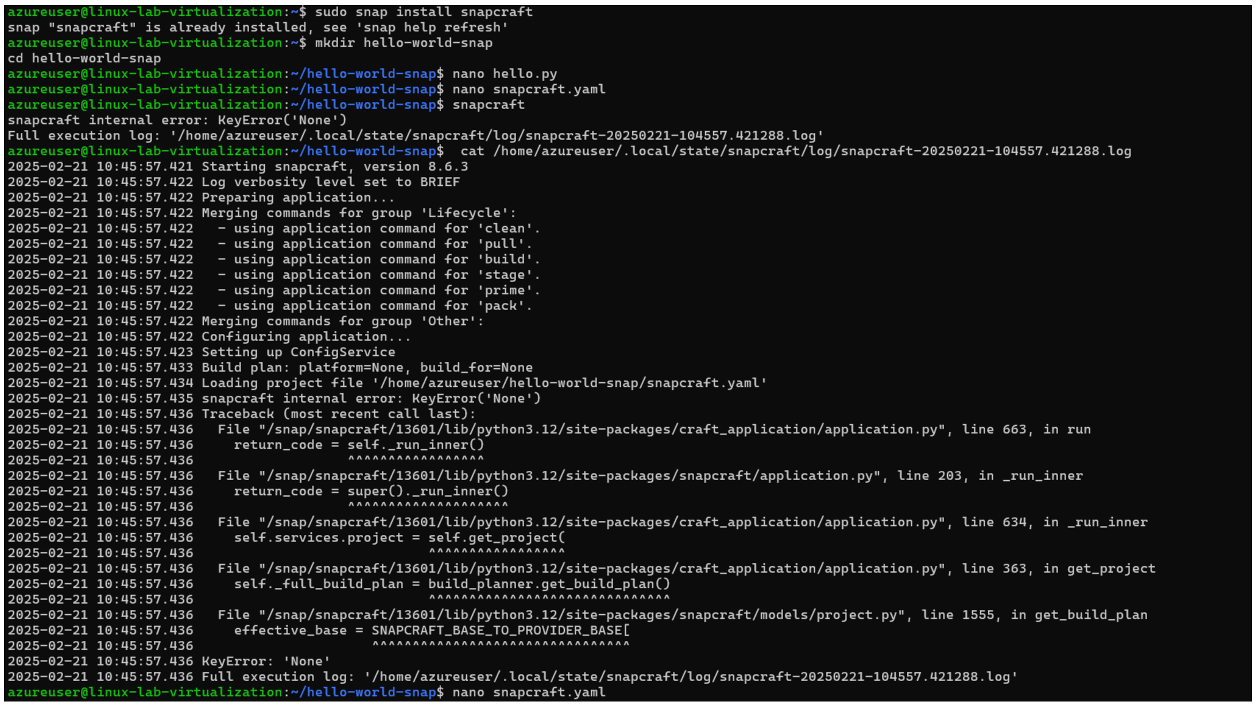
Snaps provides a way to package applications with their dependencies for consistent execution on Linux distributions.

- Research: Using sources, explore Snapcraft and the concept of Snaps.
- Experiment: Choose a simple app and compress it into Snap using Snapcraft. For more information, you can find "Snapcraft" - from the source.

Snapcraft is a tool and ecosystem that helps developers package their applications into snaps, which are self-contained software packages that include the application and all of its dependencies. This makes it easy to distribute and run applications across different Linux distributions without worrying about compatibility issues or missing dependencies. Snaps are designed to work on most modern Linux distributions, and they are confined, meaning they run in isolated environments to ensure security.

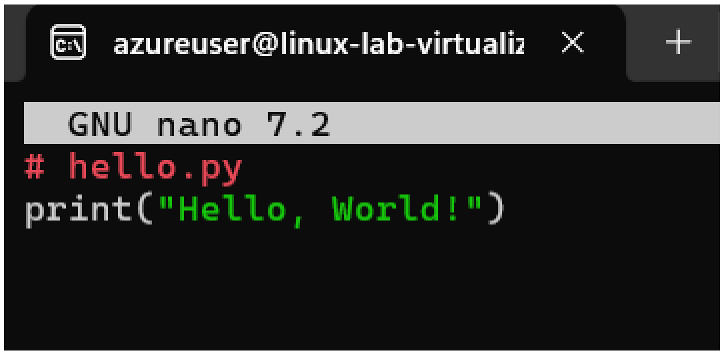
1. Install snapcraft

sudo apt update sudo apt install snapcraft



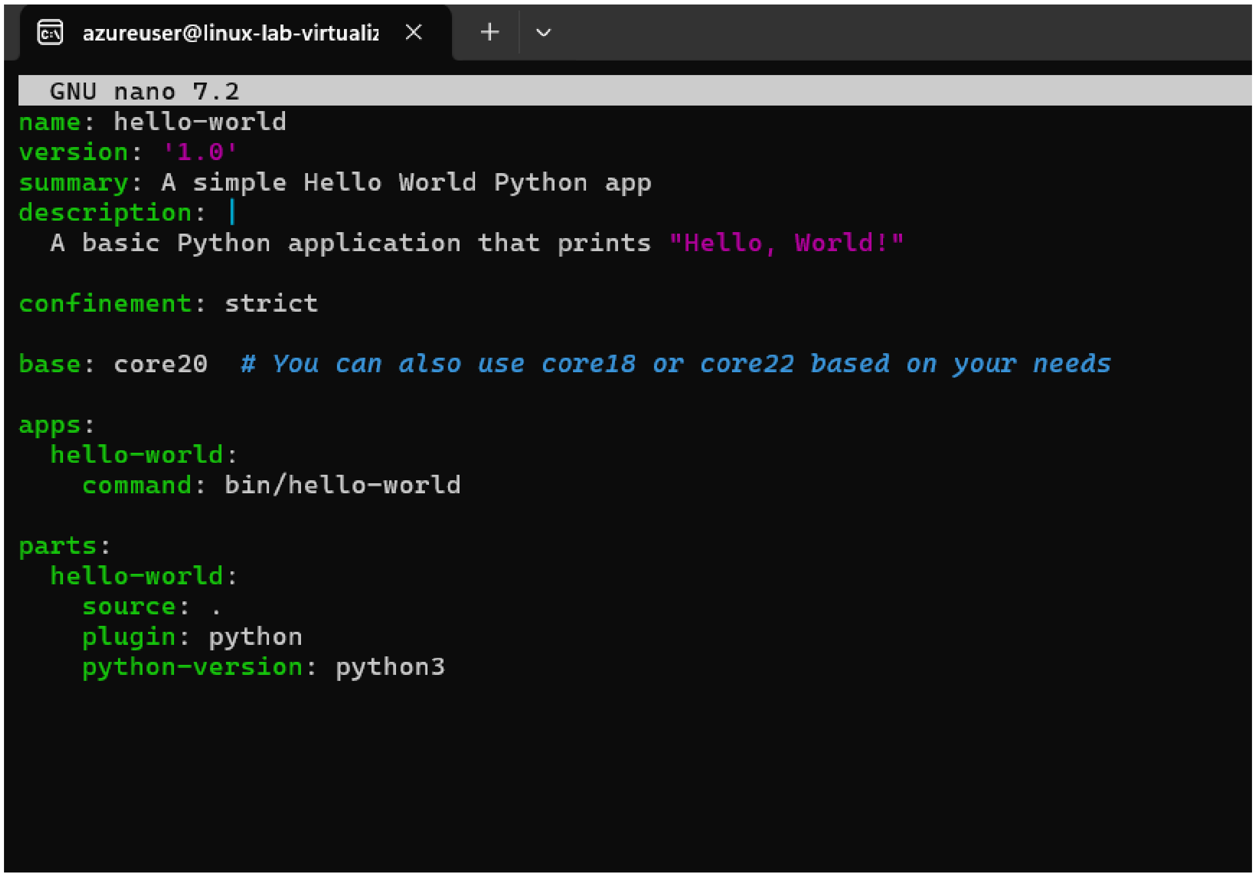
```
azureuser@linux-lab-virtualization:~$ sudo snap install snapcraft
snap "snapcraft" is already installed, see 'snap help refresh'
azureuser@linux-lab-virtualization:~$ mkdir hello-world-snap
cd hello-world-snap
azureuser@linux-lab-virtualization:~/hello-world-snap$ nano hello.py
azureuser@linux-lab-virtualization:~/hello-world-snap$ nano snapcraft.yaml
azureuser@linux-lab-virtualization:~/hello-world-snap$ snapcraft
snapcraft internal error: KeyError('None')
Full execution log: '/home/azureuser/.local/state/snapcraft/log/snapcraft-20250221-104557.421288.log'
azureuser@linux-lab-virtualization:~/hello-world-snap$  cat /home/azureuser/.local/state/snapcraft/log/snapcraft-20250221-104557.421288.log
2025-02-21 10:45:57.421 Starting snapcraft, version 8.6.3
2025-02-21 10:45:57.422 Log verbosity level set to BRIEF
2025-02-21 10:45:57.422 Preparing application...
2025-02-21 10:45:57.422 Merging commands for group 'Lifecycle':
2025-02-21 10:45:57.422   - using application command for 'clean'.
2025-02-21 10:45:57.422   - using application command for 'pull'.
2025-02-21 10:45:57.422   - using application command for 'build'.
2025-02-21 10:45:57.422   - using application command for 'stage'.
2025-02-21 10:45:57.422   - using application command for 'prime'.
2025-02-21 10:45:57.422   - using application command for 'pack'.
2025-02-21 10:45:57.422 Merging commands for group 'Other':
2025-02-21 10:45:57.422 Configuring application...
2025-02-21 10:45:57.423 Setting up ConfigService
2025-02-21 10:45:57.433 Build plan: platform=None, build_for=None
2025-02-21 10:45:57.434 Loading project file '/home/azureuser/hello-world-snap/snapcraft.yaml'
2025-02-21 10:45:57.435 snapcraft internal error: KeyError('None')
2025-02-21 10:45:57.436 Traceback (most recent call last):
2025-02-21 10:45:57.436   File "/snap/snapcraft/13601/lib/python3.12/site-packages/craft_application/application.py", line 663, in run
2025-02-21 10:45:57.436     return_code = self._run_inner()
2025-02-21 10:45:57.436                   ^^^^^^^^^^^^^^^^^
2025-02-21 10:45:57.436   File "/snap/snapcraft/13601/lib/python3.12/site-packages/snapcraft/application.py", line 203, in _run_inner
2025-02-21 10:45:57.436     return_code = super()._run_inner()
2025-02-21 10:45:57.436                   ^^^^^^^^^^^^^^^^^^^^
2025-02-21 10:45:57.436   File "/snap/snapcraft/13601/lib/python3.12/site-packages/craft_application/application.py", line 634, in _run_inner
2025-02-21 10:45:57.436     self.services.project = self.get_project(
2025-02-21 10:45:57.436                             ^^^^^^^^^^^^^^^^^
2025-02-21 10:45:57.436   File "/snap/snapcraft/13601/lib/python3.12/site-packages/craft_application/application.py", line 363, in get_project
2025-02-21 10:45:57.436     self._full_build_plan = build_planner.get_build_plan()
2025-02-21 10:45:57.436                             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2025-02-21 10:45:57.436   File "/snap/snapcraft/13601/lib/python3.12/site-packages/snapcraft/models/project.py", line 1555, in get_build_plan
2025-02-21 10:45:57.436     effective_base = SNAPCRAFT_BASE_TO_PROVIDER_BASE[
2025-02-21 10:45:57.436                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
2025-02-21 10:45:57.436 KeyError: 'None'
2025-02-21 10:45:57.436 Full execution log: '/home/azureuser/.local/state/snapcraft/log/snapcraft-20250221-104557.421288.log'
azureuser@linux-lab-virtualization:~/hello-world-snap$ nano snapcraft.yaml
```

2. Create the python file



```
 GNU nano 7.2
# hello.py
print("Hello, World!")
```

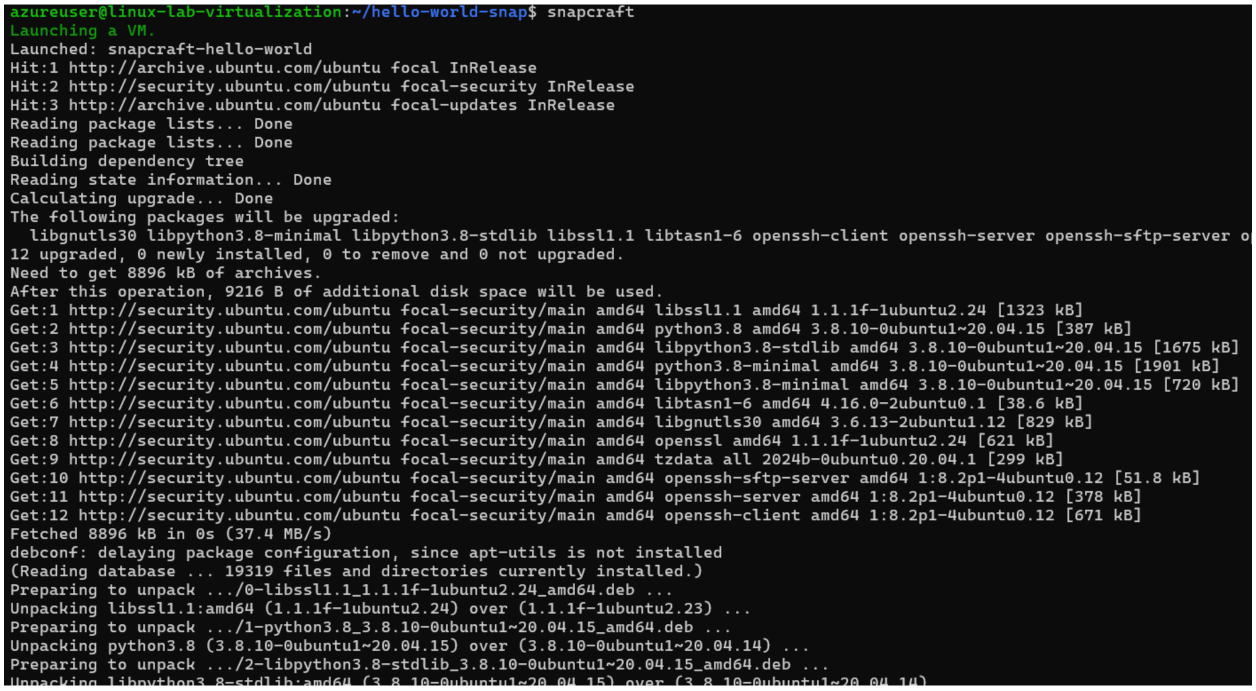
3. Create the Snapcraft Configuration



```
  GNU nano 7.2
name: hello-world
version: '1.0'
summary: A simple Hello World Python app
description: |
  A basic Python application that prints "Hello, World!"

confinement: strict

base: core20  # You can also use core18 or core22 based on your needs

apps:
  hello-world:
    command: bin/hello-world

parts:
  hello-world:
    source: .
    plugin: python
    python-version: python3
```

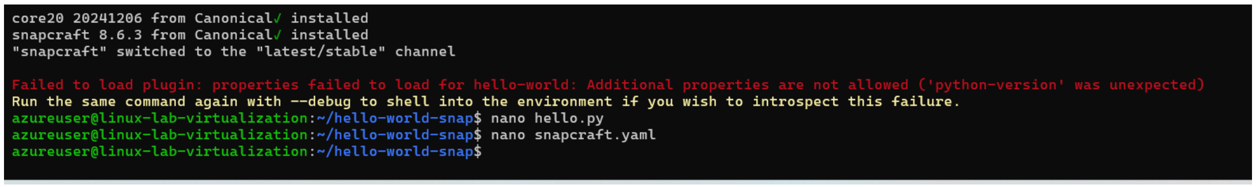
4. Fix the error by adding base: core20

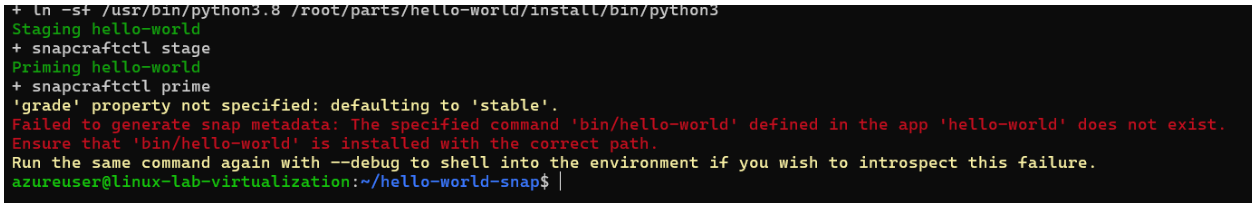
5. Create the snapcraft



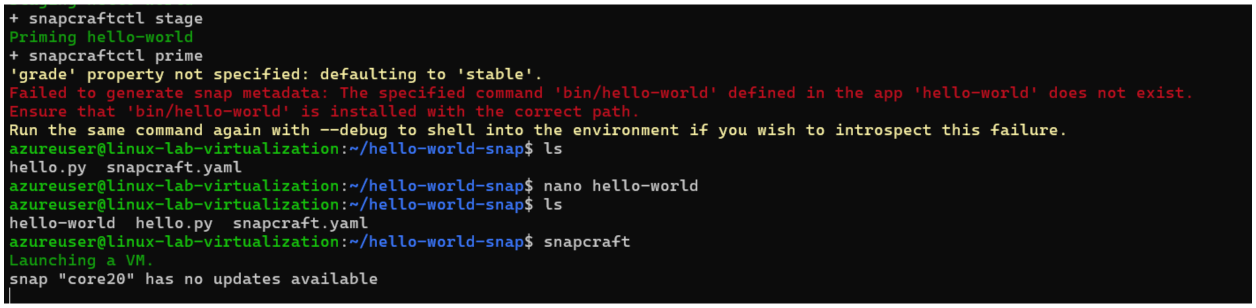
```
azureuser@linux-lab-virtualization:~/hello-world-snap$ snapcraft
Launching a VM.
Launched: snapcraft-hello-world
Hit:1 http://archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://security.ubuntu.com/ubuntu focal-security InRelease
Hit:3 http://archive.ubuntu.com/ubuntu focal-updates InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  libgnutls30 libpython3.8-minimal libpython3.8-stdlib libssl1.1 libtasn1-6 openssh-client openssh-server openssh-sftp-server o
12 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 8896 kB of archives.
After this operation, 9216 B of additional disk space will be used.
Get:1 http://security.ubuntu.com/ubuntu focal-security/main amd64 libssl1.1 amd64 1.1.1f-1ubuntu2.24 [1323 kB]
Get:2 http://security.ubuntu.com/ubuntu focal-security/main amd64 python3.8 amd64 3.8.10-0ubuntu1~20.04.15 [387 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security/main amd64 libpython3.8-stdlib amd64 3.8.10-0ubuntu1~20.04.15 [1675 kB]
Get:4 http://security.ubuntu.com/ubuntu focal-security/main amd64 python3.8-minimal amd64 3.8.10-0ubuntu1~20.04.15 [1901 kB]
Get:5 http://security.ubuntu.com/ubuntu focal-security/main amd64 libpython3.8-minimal amd64 3.8.10-0ubuntu1~20.04.15 [720 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 libtasn1-6 amd64 4.16.0-2ubuntu0.1 [38.6 kB]
Get:7 http://security.ubuntu.com/ubuntu focal-security/main amd64 libgnutls30 amd64 3.6.13-2ubuntu1.12 [829 kB]
Get:8 http://security.ubuntu.com/ubuntu focal-security/main amd64 openssl amd64 1.1.1f-1ubuntu2.24 [621 kB]
Get:9 http://security.ubuntu.com/ubuntu focal-security/main amd64 tzdata all 2024b-0ubuntu0.20.04.1 [299 kB]
Get:10 http://security.ubuntu.com/ubuntu focal-security/main amd64 openssh-sftp-server amd64 1:8.2p1-4ubuntu0.12 [51.8 kB]
Get:11 http://security.ubuntu.com/ubuntu focal-security/main amd64 openssh-server amd64 1:8.2p1-4ubuntu0.12 [378 kB]
Get:12 http://security.ubuntu.com/ubuntu focal-security/main amd64 openssh-client amd64 1:8.2p1-4ubuntu0.12 [671 kB]
Fetched 8896 kB in 0s (37.4 MB/s)
debconf: delaying package configuration, since apt-utils is not installed
(Reading database ... 19319 files and directories currently installed.)
Preparing to unpack .../0-libssl1.1_1.1.1f-1ubuntu2.24_amd64.deb ...
Unpacking libssl1.1:amd64 (1.1.1f-1ubuntu2.24) over (1.1.1f-1ubuntu2.23) ...
Preparing to unpack .../1-python3.8_3.8.10-0ubuntu1~20.04.15_amd64.deb ...
Unpacking python3.8 (3.8.10-0ubuntu1~20.04.15) over (3.8.10-0ubuntu1~20.04.14) ...
Preparing to unpack .../2-libpython3.8-stdlib_3.8.10-0ubuntu1~20.04.15_amd64.deb ...
Unpacking libpython3.8-stdlib:amd64 (3.8.10-0ubuntu1~20.04.15) over (3.8.10-0ubuntu1~20.04.14)
```

6. Remove python-version to fix the error,


```
core20 20241206 from Canonical✓ installed
snapcraft 8.6.3 from Canonical✓ installed
"snapcraft" switched to the "latest/stable" channel

Failed to load plugin: properties failed to load for hello-world: Additional properties are not allowed ('python-version' was unexpected)
Run the same command again with --debug to shell into the environment if you wish to introspect this failure.
azureuser@linux-lab-virtualization:~/hello-world-snap$ nano hello.py
azureuser@linux-lab-virtualization:~/hello-world-snap$ nano snapcraft.yaml
azureuser@linux-lab-virtualization:~/hello-world-snap$
```


```
+ ln -sf /usr/bin/python3.8 /root/parts/hello-world/install/bin/python3
Staging hello-world
+ snapcraftctl stage
Priming hello-world
+ snapcraftctl prime
'grade' property not specified: defaulting to 'stable'.
Failed to generate snap metadata: The specified command 'bin/hello-world' defined in the app 'hello-world' does not exist.
Ensure that 'bin/hello-world' is installed with the correct path.
Run the same command again with --debug to shell into the environment if you wish to introspect this failure.
azureuser@linux-lab-virtualization:~/hello-world-snap$ |
```

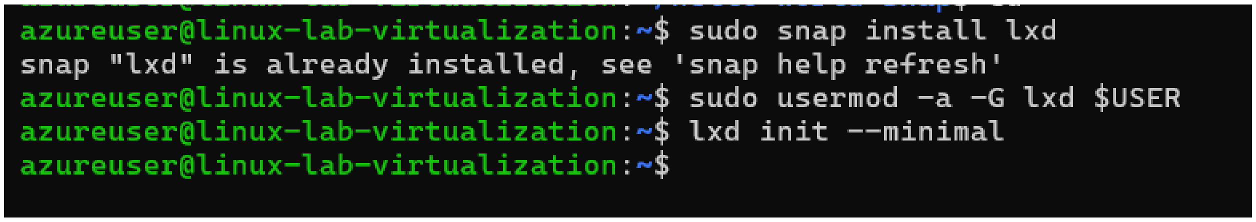
7. Check for the file hello-world and create the missing file. Re run the snap


```
+ snapcraftctl stage
Priming hello-world
+ snapcraftctl prime
'grade' property not specified: defaulting to 'stable'.
Failed to generate snap metadata: The specified command 'bin/hello-world' defined in the app 'hello-world' does not exist.
Ensure that 'bin/hello-world' is installed with the correct path.
Run the same command again with --debug to shell into the environment if you wish to introspect this failure.
azureuser@linux-lab-virtualization:~/hello-world-snap$ ls
hello.py  snapcraft.yaml
azureuser@linux-lab-virtualization:~/hello-world-snap$ nano hello-world
azureuser@linux-lab-virtualization:~/hello-world-snap$ ls
hello-world  hello.py  snapcraft.yaml
azureuser@linux-lab-virtualization:~/hello-world-snap$ snapcraft
Launching a VM.
snap "core20" has no updates available
```

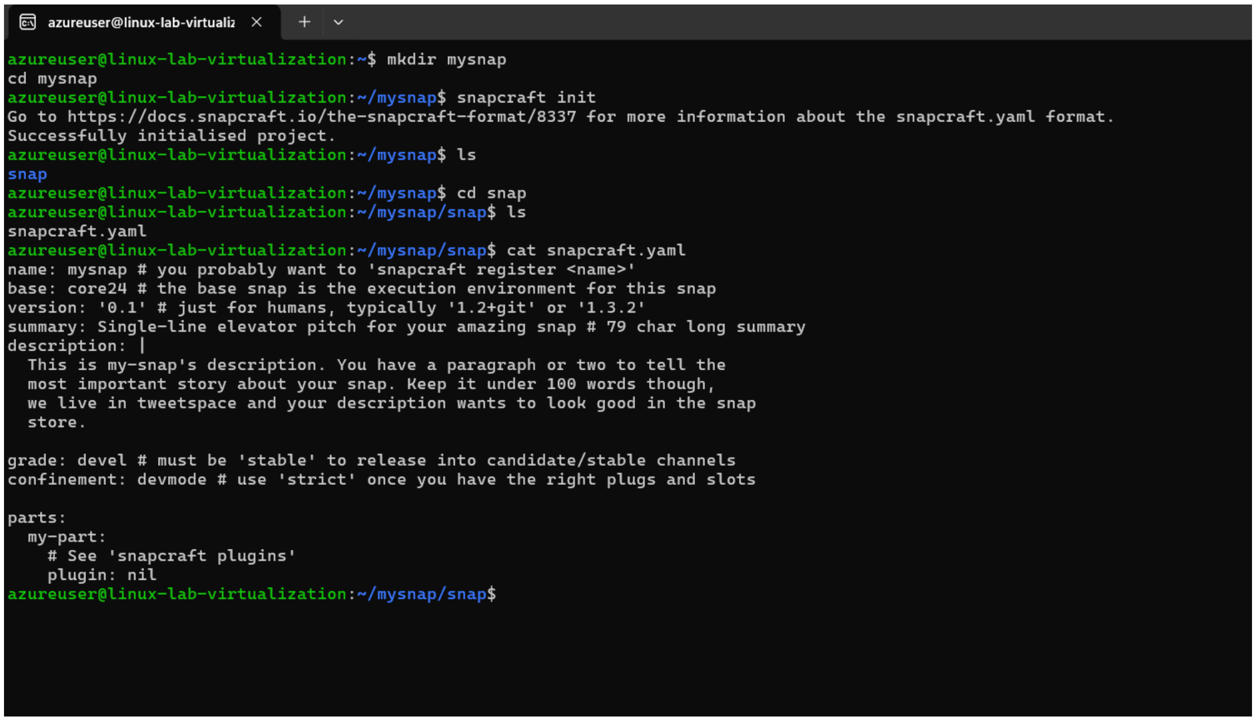
Method 2

1. Build snap ENV.

sudo snap install lxd

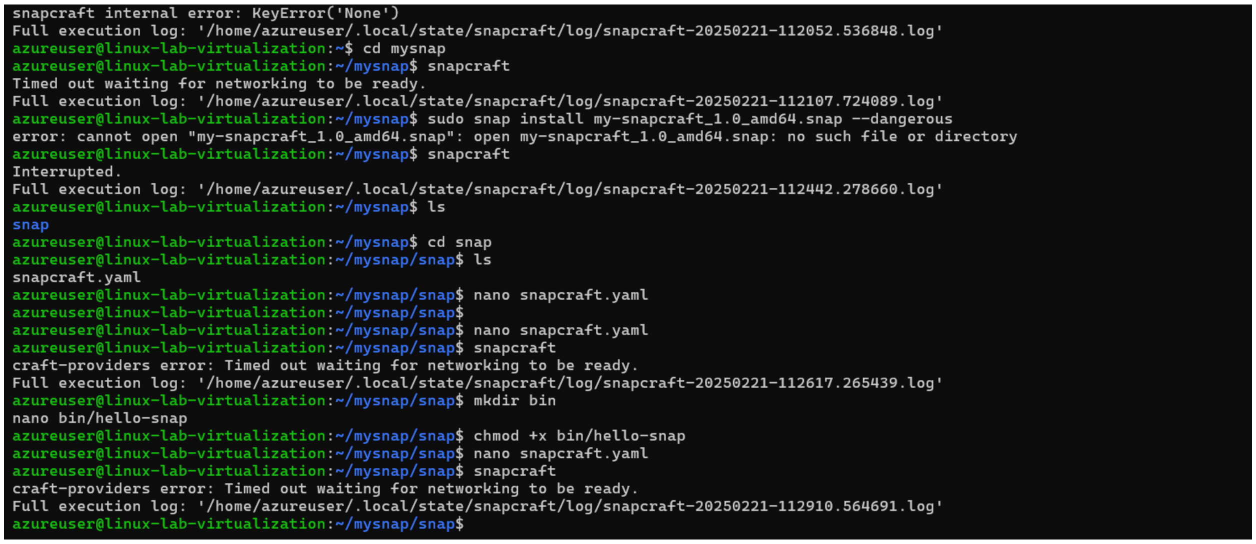

```
azureuser@linux-lab-virtualization:~$ sudo snap install lxd
snap "lxd" is already installed, see 'snap help refresh'
azureuser@linux-lab-virtualization:~$ sudo usermod -a -G lxd $USER
azureuser@linux-lab-virtualization:~$ lxd init --minimal
azureuser@linux-lab-virtualization:~$
```

2. Create YAML template



```
azureuser@linux-lab-virtualization:~$ mkdir mysnap
cd mysnap
azureuser@linux-lab-virtualization:~/mysnap$ snapcraft init
Go to https://docs.snapcraft.io/the-snapcraft-format/8337 for more information about the snapcraft.yaml format.
Successfully initialised project.
azureuser@linux-lab-virtualization:~/mysnap$ ls
snap
azureuser@linux-lab-virtualization:~/mysnap$ cd snap
azureuser@linux-lab-virtualization:~/mysnap/snap$ ls
snapcraft.yaml
azureuser@linux-lab-virtualization:~/mysnap/snap$ cat snapcraft.yaml
name: mysnap # you probably want to 'snapcraft register <name>'
base: core24 # the base snap is the execution environment for this snap
version: '0.1' # just for humans, typically '1.2+git' or '1.3.2'
summary: Single-line elevator pitch for your amazing snap # 79 char long summary
description: |
  This is my-snap's description. You have a paragraph or two to tell the
  most important story about your snap. Keep it under 100 words though,
  we live in tweetspace and your description wants to look good in the snap
  store.

grade: devel # must be 'stable' to release into candidate/stable channels
confinement: devmode # use 'strict' once you have the right plugs and slots

parts:
  my-part:
    # See 'snapcraft plugins'
    plugin: nil
azureuser@linux-lab-virtualization:~/mysnap/snap$
```

3. Build template



```
snapcraft internal error: KeyError('None')
Full execution log: '/home/azureuser/.local/state/snapcraft/log/snapcraft-20250221-112052.536848.log'
azureuser@linux-lab-virtualization:~$ cd mysnap
azureuser@linux-lab-virtualization:~/mysnap$ snapcraft
Timed out waiting for networking to be ready.
Full execution log: '/home/azureuser/.local/state/snapcraft/log/snapcraft-20250221-112107.724089.log'
azureuser@linux-lab-virtualization:~/mysnap$ sudo snap install my-snapcraft_1.0_amd64.snap --dangerous
error: cannot open "my-snapcraft_1.0_amd64.snap": open my-snapcraft_1.0_amd64.snap: no such file or directory
azureuser@linux-lab-virtualization:~/mysnap$ snapcraft
Interrupted.
Full execution log: '/home/azureuser/.local/state/snapcraft/log/snapcraft-20250221-112442.278660.log'
azureuser@linux-lab-virtualization:~/mysnap$ ls
snap
azureuser@linux-lab-virtualization:~/mysnap$ cd snap
azureuser@linux-lab-virtualization:~/mysnap/snap$ ls
snapcraft.yaml
azureuser@linux-lab-virtualization:~/mysnap/snap$ nano snapcraft.yaml
azureuser@linux-lab-virtualization:~/mysnap/snap$
azureuser@linux-lab-virtualization:~/mysnap/snap$ nano snapcraft.yaml
azureuser@linux-lab-virtualization:~/mysnap/snap$ snapcraft
craft-providers error: Timed out waiting for networking to be ready.
Full execution log: '/home/azureuser/.local/state/snapcraft/log/snapcraft-20250221-112617.265439.log'
azureuser@linux-lab-virtualization:~/mysnap/snap$ mkdir bin
nano bin/hello-snap
azureuser@linux-lab-virtualization:~/mysnap/snap$ chmod +x bin/hello-snap
azureuser@linux-lab-virtualization:~/mysnap/snap$ nano snapcraft.yaml
azureuser@linux-lab-virtualization:~/mysnap/snap$ snapcraft
craft-providers error: Timed out waiting for networking to be ready.
Full execution log: '/home/azureuser/.local/state/snapcraft/log/snapcraft-20250221-112910.564691.log'
azureuser@linux-lab-virtualization:~/mysnap/snap$
```