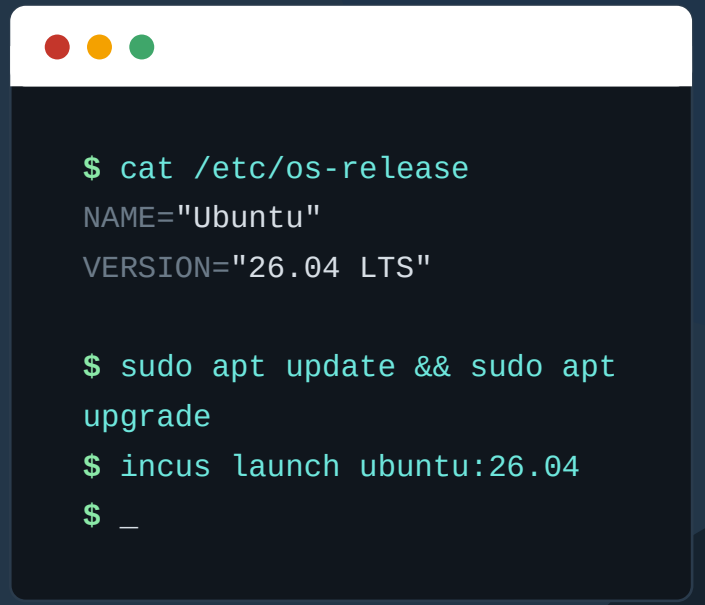
COMPLETE VIDEO COURSE

Ubuntu 26.04 LTS “Resolute Raccoon”

Complete System Administration

apt & snap · netplan · TPM Encryption · AppArmor · Docker · LXD/Incus · Compilers



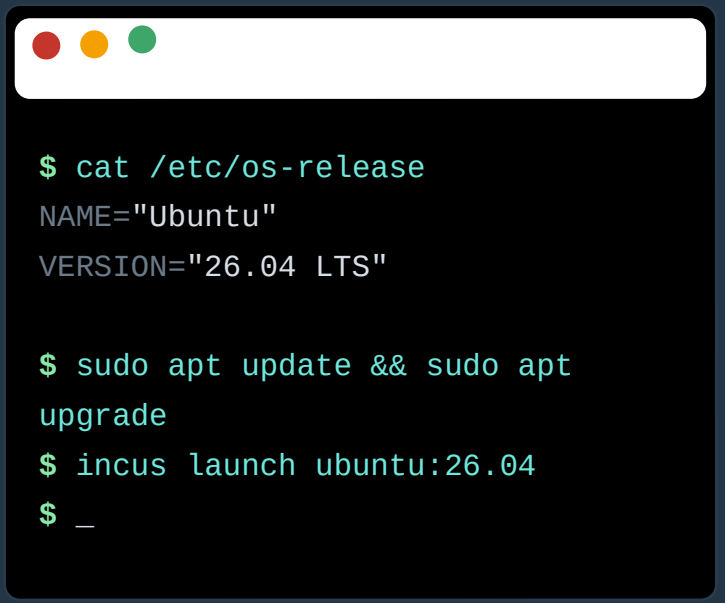
60 SLIDES | 7 MODULES | JUNIOR → SENIOR | HANDS-ON CLI

A COMPLETE VIDEO COURSE

Ubuntu 26.04 LTS Complete System Administration

From apt and netplan to TPM-backed encryption, AppArmor, Docker, LXD/Incus virtualization, and multi-language container builds — for junior, intermediate, and senior administrators.

- 60 Slides · 7 Modules
- Hands-on CLI Examples Throughout
- Ubuntu 26.04 LTS "Resolute Raccoon" (Kernel 7.0)



COURSE ROADMAP

What You Will Learn



01 Foundations & Architecture

Containerization, Ubuntu architecture, release model, repositories

02 Package & System Management

apt, dpkg, snap, PPAs, Ubuntu Pro, tmux, timedatectl, systemd

03 Networking

Netplan, DHCP, ufw & nftables

04 Security & Cryptography

AppArmor, LUKS/TPM, SSL/TLS, USG, FIPS, post-quantum, 2FA

05 Containers & Virtualization

Docker, Podman, LXD/Incus, QEMU/KVM & libvirt

06 Development Toolchains

GCC, Node.js, Java, Python, Rust — host & container

+ BONUS MODULE 07 AI-assisted troubleshooting tips, plus free chapters & the complete book

01

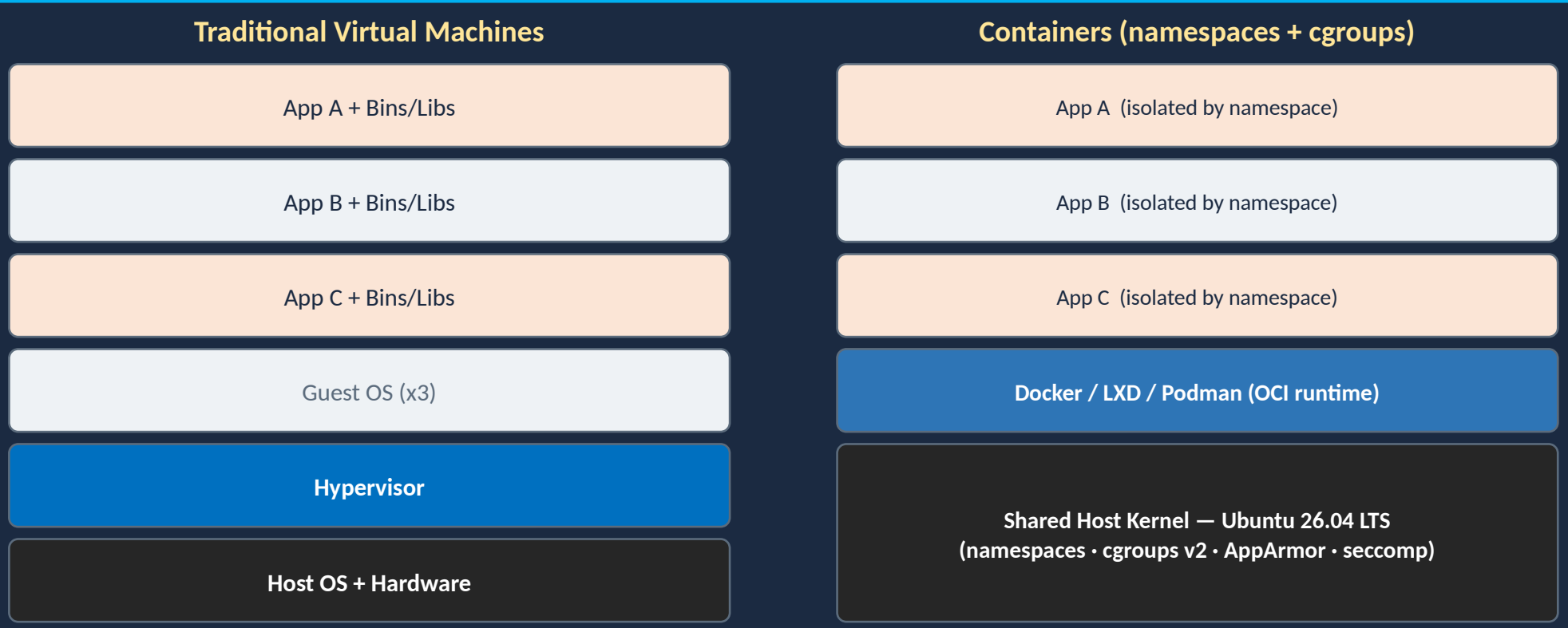
MODULE 1

Foundations & Architecture

Understand what containerization actually is at the kernel level, how Ubuntu's release model and Ubuntu Pro support work, and how the package repository model is organized.

Containerization | Namespaces & cgroups | LTS vs Interim | Main | Universe | PPAs

What Is Containerization?



No guest OS or hypervisor — containers share one kernel and start in milliseconds, not minutes.

How Ubuntu Implements Containers



Applications & Images	OCI images from Docker Hub, the Snap Store, or Canonical's LXD/Incus image server
Docker 29 · Podman · LXD/Incus	Container & system-container tooling — LXD/Incus is unique to Ubuntu for OS-like containers and VMs
containerd / runc	Creates the container process from namespaces & cgroups per the OCI spec
Kernel Isolation Primitives	PID / net / mount / UTS / IPC / user namespaces · cgroups v2 resource limits
Mandatory Access Control & Seccomp	AppArmor profiles confine each container + seccomp filters restrict syscalls
Ubuntu 26.04 LTS Kernel 7.0	Same upstream kernel base whether on a laptop, a bare-metal server, or the cloud

Ubuntu Core takes this stack further — an immutable, snap-based OS image built for IoT and edge devices.

Ubuntu's Release Model: LTS vs. Interim

LTS Releases (e.g. 26.04)

- Released every 2 years in April — the release to run in production
- 5 years of free security updates from Canonical
- Up to 10 more years of Expanded Security Maintenance (ESM) via Ubuntu Pro
- Ubuntu Pro is free for personal use on up to 5 machines
- This course targets 26.04 LTS "Resolute Raccoon"

Interim Releases (e.g. 25.10)

- Released every 6 months (April and October)
- 9 months of support — for testing new features early
- A proving ground for what eventually lands in the next LTS
- Not recommended for production servers
- Great for developers who want the newest packages

Direct upgrades from one LTS to the next open once the first point release ships (26.04.1 for 24.04 -> 26.04 upgrades).

The Ubuntu 26.04 Platform at a Glance

7.0

Upstream Linux kernel

GNOME 50

Wayland-only desktop

6 GB

Minimum desktop RAM

2031

Supported until (free)

Repository Landscape: Main, Universe, Restricted & PPAs

Main

Canonical-supported free & open-source software — the core of Ubuntu, security-patched for the full LTS lifetime.

Universe

Community-maintained open-source software — huge selection, best-effort security support (or full support via Ubuntu Pro).

Restricted / Multiverse

Restricted: proprietary drivers Canonical supports. Multiverse: software restricted by copyright or legal issues.

PPAs & Snap Store

Personal Package Archives add third-party .debs; the Snap Store distributes sandboxed, auto-updating snap packages.

```
$ sudo add-apt-repository universe && sudo apt update
$ sudo add-apt-repository ppa:example/ppa
```

02

MODULE 2

Package & System Management

Master apt and dpkg, understand snap packages and PPAs, manage Ubuntu Pro entitlements, and get comfortable with tmux, timedatectl, and the systemd service model.

apt & dpkg | snap | PPAs | Ubuntu Pro | tmux | timedatectl | systemd

APT Fundamentals

apt-basics.sh

```
$
sudo apt update
$
sudo apt install nginx
$
sudo apt remove nginx
$
apt search "web server"
$
apt show nginx
$
apt list --installed | grep linux-image
$
sudo apt full-upgrade
$
sudo apt autoremove
$
```

NOTE apt full-upgrade may remove packages to resolve dependencies — review the plan before confirming on production systems.

dpkg & Package Internals



dpkg-internals.sh

```
$
sudo dpkg -i custom-app_1.2_amd64.deb
$
dpkg -L nginx
$
dpkg -S /usr/sbin/nginx
$
sudo apt-mark hold linux-image-generic
$
apt-mark showhold
$
sudo dpkg-reconfigure tzdata
$
dpkg -l | grep ^ii | wc -l
```

Snap vs. .deb/APT Packages

Traditional .deb via APT

- Shares system libraries — smaller install footprint
- Updated on your schedule via apt update / upgrade
- No sandboxing beyond AppArmor policy defaults
- The right choice for core OS packages and servers

Snap Packages

- Bundles its own dependencies — works identically across distros
- Auto-updates in the background on its own channel/track
- Strictly confined by default via its own AppArmor sandbox
- Great for desktop apps and fast-moving tools (e.g. code editors)

snap list / snap refresh / snap info <name> manage snaps; both systems coexist on every Ubuntu install.

PPAs: Personal Package Archives



ppa-mgmt.sh

```
$
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
$
sudo apt update
$
apt policy gcc-13
$
sudo add-apt-repository --remove ppa:ubuntu-
toolchain-r/test
$
ls /etc/apt/sources.list.d/
$
apt-cache policy
```

Ubuntu Pro & Expanded Security Maintenance

ubuntu-pro.sh

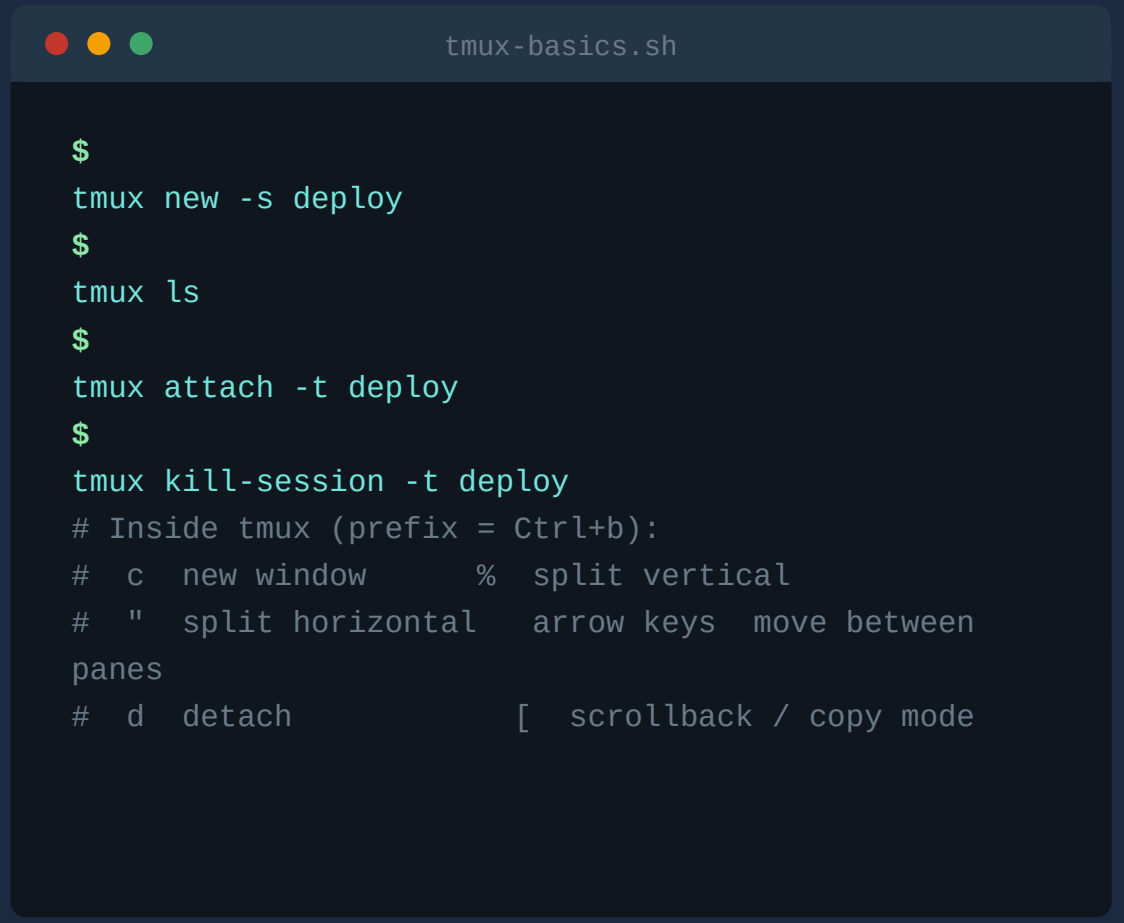
```
$
sudo pro attach <your-token>
$
pro status
$
sudo pro enable esm-infra
$
sudo pro enable esm-apps
$
sudo pro enable livepatch
$
sudo pro detach
```

Automatic Updates & Canonical Livepatch

updates.sh

```
$
sudo apt install unattended-upgrades
$
sudo dpkg-reconfigure --priority=low unattended-
upgrades
$
cat /etc/apt/apt.conf.d/50unattended-upgrades
$
sudo pro enable livepatch
$
canonical-livepatch status --verbose
$
sudo apt list --upgradable
```

tmux: Terminal Multiplexing for Admins



timedatectl & chrony: Accurate Time

time-mgmt.sh

```
$
timedatectl status
$
sudo timedatectl set-timezone America/Los_Angeles
$
sudo timedatectl set-ntp true
$
timedatectl list-timezones | grep -i angeles
$
chronyc sources -v
$
chronyc tracking
```

systemd Fundamentals

systemd is PID 1 on Ubuntu — the init system, service manager, and dependency graph.

U

.service, .socket, .timer, .mount, .target files under /usr/lib/systemd/system

T

Group units into boot stages (multi-user.target, graphical.target)

J

Centralized binary logging — query with journalctl -u <service> -f

D

Wants=, Requires=, After=, Before= define the startup ordering graph

Managing systemd Services

systemd-ops.sh

```
$
sudo systemctl enable --now nginx
$
systemctl status nginx
$
sudo systemctl restart nginx
$
sudo systemctl mask sendmail
$
sudo systemctl edit nginx.service
$
sudo systemctl daemon-reload
$
journalctl -u nginx --since "1 hour ago"
$
systemctl list-units --failed
```

03

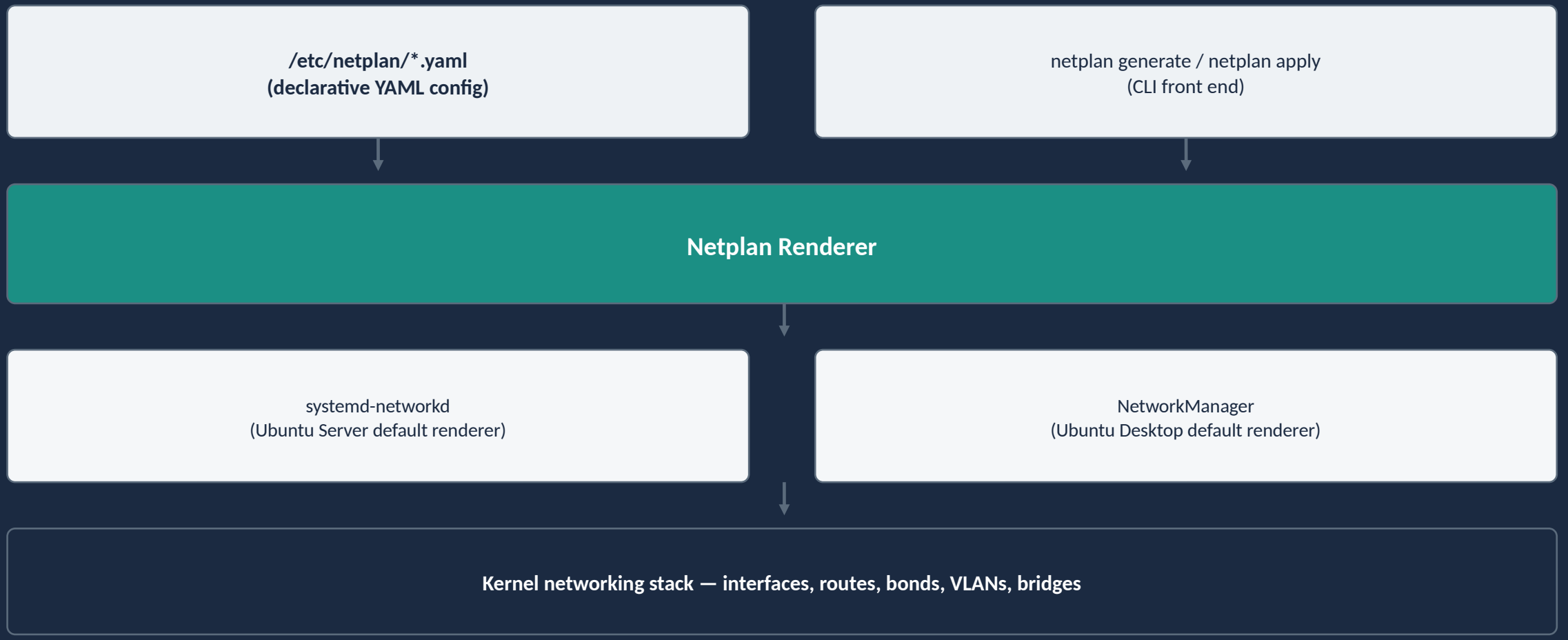
MODULE 3

Networking

Configure interfaces the modern, declarative way with Netplan, hand out and request addresses with DHCP, and protect the host with ufw and nftables.

Netplan | systemd-networkd | Bonding & VLANs | DHCP | ufw | nftables

Netplan Architecture



Netplan Basics



01-netcfg.yaml

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      addresses: [192.168.1.50/24]
      routes:
        - to: default
          via: 192.168.1.1
      nameservers:
        addresses: [1.1.1.1, 1.0.0.1]
$
sudo netplan try
$
sudo netplan apply
```

Netplan Advanced: Bonding, VLANs & Bridges

02-advanced.yaml

```
network:
  version: 2
  bonds:
    bond0:
      interfaces: [eth0, eth1]
      parameters:
        mode: active-backup
  vlans:
    vlan10:
      id: 10
      link: eth2
  bridges:
    br0:
      interfaces: [eth3]
```

DHCP: Client & Server

dhcp.sh

```
# netplan: dhcp4: true
$
sudo apt install kea-dhcp4-server
# /etc/kea/kea-dhcp4.conf (excerpt)
"subnet4": [{
  "subnet": "192.168.1.0/24",
  "pools": [{"pool": "192.168.1.100-192.168.1.200"}],
  "option-data": [{"name": "routers", "data":
"192.168.1.1"}]
}]
$
sudo systemctl enable --now kea-dhcp4-server
```

ufw & nftables



ufw.sh

```
$
sudo ufw allow OpenSSH
$
sudo ufw allow 443/tcp
$
sudo ufw allow from 10.0.0.0/8 to any port 5432
$
sudo ufw enable
$
sudo ufw status numbered
$
sudo ufw delete 3
$
sudo nft list ruleset | less
```

04

MODULE 4

Security & Cryptography

Go deep on AppArmor, disk and transport encryption, the Ubuntu Security Guide, FIPS 140-3 certified modules, post-quantum readiness, and two-factor authentication.

AppArmor | TPM & LUKS | SSL/TLS | USG | FIPS 140-3 | Post-Quantum | 2FA | AD Integration

AppArmor Fundamentals

AppArmor enforces Mandatory Access Control by path — confining what each program can read, write, or execute, regardless of file permissions.

P

Path-based rules per binary, stored under /etc/apparmor.d/ — easier to read than SELinux labels

E

Enforce blocks and logs violations; complain mode only logs — used while tuning a new profile

S

Every strictly-confined snap gets its own auto-generated AppArmor profile

A

Denials land in the kernel audit log, readable with aa-status and journalctl

AppArmor Troubleshooting

apparmor-troubleshoot.sh

```
$
sudo aa-status
$
sudo aa-complain /usr/sbin/nginx
$
sudo aa-enforce /usr/sbin/nginx
$
sudo aa-logprof
$
journalctl -k | grep -i apparmor
$
sudo apparmor_parser -r
/etc/apparmor.d/usr.sbin.nginx
```

Encryption in Ubuntu 26.04: The Big Picture

Data at Rest

- TPM-backed full-disk encryption (new default flow in 26.04)
- Manual LUKS2 volume encryption via cryptsetup
- Recovery-key and re-encryption support built into the installer

Data in Transit

- TLS/SSL via OpenSSL 3.x
- OpenSSH 10.2 with post-quantum hybrid key exchange
- WireGuard / IPsec tunnels

Compliance & Trust

- Ubuntu Security Guide (USG) for CIS / DISA-STIG hardening
- FIPS 140-3 certified crypto modules via Ubuntu Pro
- AppArmor confinement on every default service

Every layer works the same on a laptop, a bare-metal server, or an Ubuntu cloud image.

TPM-Backed Full-Disk Encryption

tpm-fde.sh

```
# During install: choose "Encrypt the new Ubuntu
installation"
# with "Use TPM to unlock automatically"
$
sudo apt install tpm2-tools
$
sudo tpm2_pcrread sha256:0,7
$
lsblk -f   # confirm the crypto_LUKS volume
$
sudo journalctl -u systemd-cryptsetup
```

NOTE Keep the recovery key somewhere other than the encrypted machine itself.

Manual LUKS Disk Encryption

luks.sh

```
$
sudo cryptsetup luksFormat /dev/sdb1
$
sudo cryptsetup luksOpen /dev/sdb1 secure_data
$
sudo mkfs.ext4 /dev/mapper/secure_data
$
sudo cryptsetup luksAddKey /dev/sdb1
$
sudo cryptsetup luksHeaderBackup /dev/sdb1 \
    --header-backup-file /root/sdb1-header.img
$
sudo cryptsetup luksClose secure_data
```

SSL/TLS with OpenSSL

openssl.sh

```
$
openssl genpkey -algorithm RSA -out server.key -
pkeyopt rsa_keygen_bits:4096
$
openssl req -new -key server.key -out server.csr
$
openssl x509 -in server.crt -noout -text
$
openssl s_client -connect example.com:443 -tls1_3
$
openssl verify -CAfile ca.crt server.crt
```

Ubuntu Security Guide (USG) & Compliance Profiles



De

Default

Secure, broadly-compatible baseline shipped out of the box on Ubuntu 26.04.

CI

CIS Level 1

Baseline CIS Benchmark hardening — sensible for most servers, minimal compatibility risk.

CI

CIS Level 2

Stricter CIS profile for high-security environments — may reduce usability.

DI

DISA-STIG

U.S. DoD Security Technical Implementation Guide profile, available via Ubuntu Pro.

```
$ sudo pro enable usg && sudo usg audit cis_level1_server
$ sudo usg generate-tailoring cis_level1_server tailoring.xml
```

FIPS 140-3 Certified Crypto on Ubuntu Pro

140-3

Current FIPS validation standard

1

Command to enable

10 yr

ESM coverage via Ubuntu Pro

Post-Quantum Cryptography Readiness

NEW IN UBUNTU 26.04

Classical ECDH
(nistp256 / nistp384)
secure until large quantum computers arrive

+

ML-KEM
(Module-Lattice KEM)
NIST-standardized, quantum-resistant

=

Hybrid Key Exchange in OpenSSH 10.2 & OpenSSL 3.x on Ubuntu 26.04

Defends today's captured traffic against “harvest now, decrypt later” attacks from future quantum computers

Rust-based core utilities (sudo-rs, coreutils) reduce memory-safety attack surface · TPM-backed FDE with recovery-key support · Livepatch now available for Arm-based servers.

Two-Factor Authentication: TOTP with PAM

totp-2fa.sh

```
$
sudo apt install libpam-google-authenticator
$
google-authenticator   # per-user setup, generates QR
code
# /etc/pam.d/sshd
auth required pam_google_authenticator.so
# /etc/ssh/sshd_config
KbdInteractiveAuthentication yes
AuthenticationMethods publickey,keyboard-interactive
$
sudo systemctl restart ssh
```

Two-Factor Authentication: FIDO2 & Smart Cards



fido2-smartcard.sh

```
$
sudo apt install libpam-u2f
$
pamu2fcfg > ~/.config/Yubico/u2f_keys
# /etc/pam.d/sudo
auth required pam_u2f.so
$
sudo apt install sssd sssd-tools opensc
$
sudo pam-auth-update   # enable smart-card profile
```

Joining Active Directory with SSSD & realmd

Ubuntu servers commonly need to authenticate against an existing Windows Active Directory domain — realmd makes that a one-command operation.

R

Auto-discovers the domain and drives the join with realm join ad.example.com

S

Caches credentials, resolves AD users/groups, and enforces access control locally

K

Single sign-on tickets once joined, shared with other domain-joined Linux and Windows hosts

G

adcli and sssd.conf control which AD groups may log in and with what sudo rights

05

MODULE 5

Containers & Virtualization

Go hands-on with Docker and Podman, explore Ubuntu's unique LXD/Incus system containers, and stand up full virtual machines with the QEMU/KVM and libvirt stack.

Docker | Podman | LXD / Incus | QEMU/KVM | libvirt

Container Internals: Namespaces, cgroups & OCI

PID | NET | MNT | UTS | IPC | USER

Linux namespaces isolate what a process can SEE (its own PIDs, network stack, filesystem mounts, hostname, IPC, and UIDs)

cgroups v2
limits what a process can USE:
CPU shares, memory, I/O, PIDs count

AppArmor + seccomp
restricts what a process is ALLOWED
to do at the kernel/syscall level

OCI (Open Container Initiative) Spec — the contract that Docker, Podman, containerd, and LXD/Incus all build against:
Image Spec (how a container image is packaged) + Runtime Spec (how containerd/runc starts it) + Distribution Spec (how a registry serves it)

Docker 29 Basics



docker-basics.sh

```
$
sudo apt install docker-ce docker-ce-cli
containerd.io
$
sudo usermod -aG docker $USER
$
docker run -d --name web -p 8080:80 nginx:latest
$
docker ps
$
docker logs -f web
$
docker compose up -d
```

Building Images & Rootless Podman

Containerfile

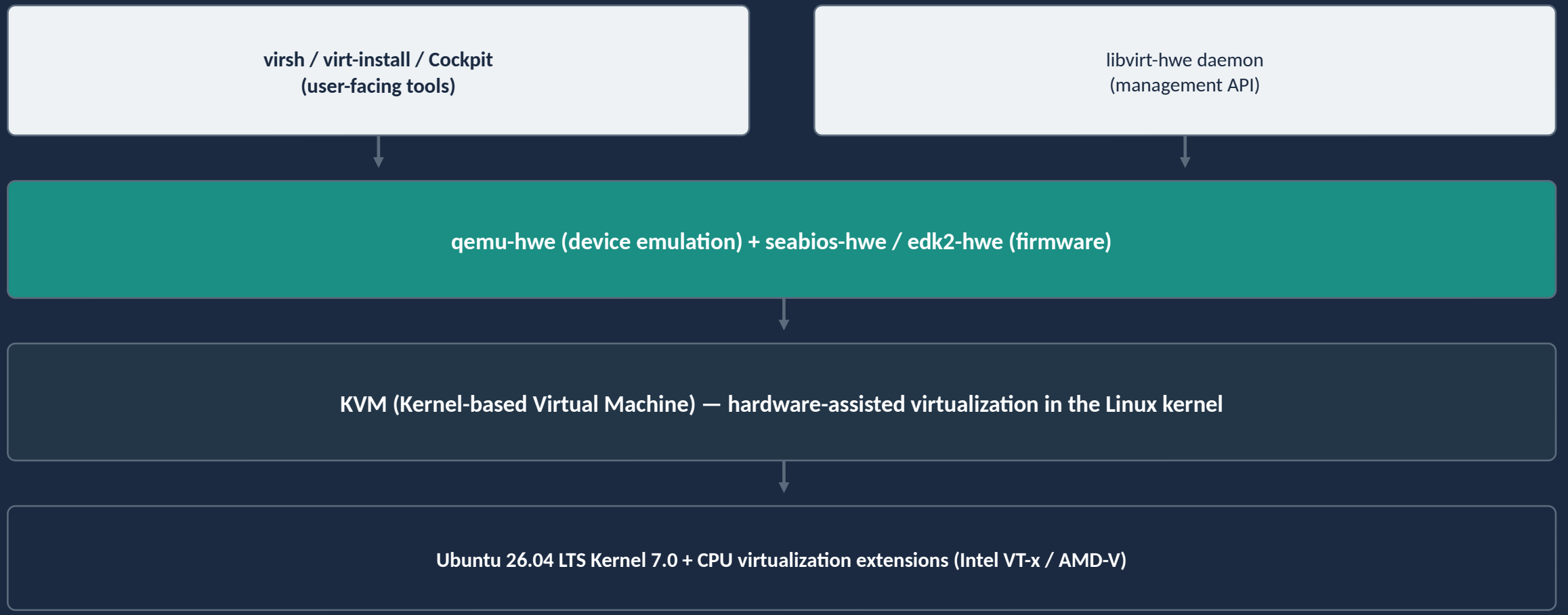
```
FROM ubuntu:26.04 AS build
RUN apt update && apt install -y python3.13
python3.13-venv gcc && \
    python3.13 -m pip install --prefix=/install -r
requirements.txt
FROM ubuntu:26.04-minimal
COPY --from=build /install /usr/local
COPY app.py /app/app.py
CMD ["python3.13", "/app/app.py"]
$
podman build -t myapp:1.0 .
```

LXD/Incus: System Containers & VMs

incus.sh

```
$
sudo apt install incus
$
incus launch ubuntu:26.04 web-container
$
incus launch ubuntu:26.04 test-vm --vm
$
incus list
$
incus exec web-container -- bash
$
incus snapshot create web-container before-upgrade
```

Full Virtualization: The virt-hwe Stack



The virt-hwe (Hardware Enablement) stack ships newer QEMU/libvirt on LTS releases for the latest CPU and GPU support.

KVM & libvirt Hands-On

kvm-libvirt.sh

```
$
sudo apt install qemu-kvm libvirt-daemon-system
virtinst
$
sudo virt-install --name test-vm --memory 4096 --
vcpus 2 \
    --disk size=20 --cdrom ubuntu-26.04-server-
amd64.iso
$
virsh list --all
$
virsh start test-vm
$
virsh console test-vm
```

06

MODULE 6

Development Toolchains

Tour the compiler and language ecosystem shipped in Ubuntu 26.04 — GCC, Node.js, Java, Python, and Rust — both directly on the host and packaged inside containers.

GCC / build-essential | Node.js | Java / OpenJDK | Python 3.13 | Rust

Compiler & Language Toolchains in 26.04

In the Base Repositories

- build-essential pulls in GCC, G++, make, and libc headers in one command
- Default Python 3 and OpenJDK packages, ready via apt install
- PostgreSQL 18, Docker 29, and PHP 8.5 ship as current server-stack versions

Optional / Newer Toolchains

- Toolchain PPAs (e.g. ubuntu-toolchain-r) offer newer GCC releases ahead of the next LTS
- NodeSource's apt repository ships current Node.js LTS releases
- rustup installs and manages Rust toolchains independent of the distro package manager

Rust-based core utilities (sudo-rs, uutils coreutils) also ship in the base system for improved memory safety.

GCC / G++ Demo Across Versions



gcc-demo.sh

```
$
sudo apt install build-essential
$
gcc --version        # distro default
$
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
$
sudo apt install gcc-14 g++-14
$
sudo update-alternatives --install /usr/bin/gcc
gcc /usr/bin/gcc-14 100
$
g++ -std=c++23 -O2 -Wall -o app main.cpp && ./app
```

Node.js & Python: Application Runtimes



node-python.sh

```
$
curl -fsSL https://deb.nodesource.com/setup_lts.x |
sudo -E bash -
$
sudo apt install nodejs
$
node --version && npm --version
$
sudo apt install python3.13 python3.13-venv
$
python3.13 -m venv ~/venvs/app
$
source ~/venvs/app/bin/activate
$
pip install --upgrade pip requests
```

Java (OpenJDK) & Rust Development



java-rust-demo.sh

```
$
sudo apt install openjdk-21-jdk openjdk-17-jdk
$
sudo update-alternatives --config java
$
java -version && javac HelloWorld.java && java
HelloWorld
$
curl --proto '=https' --tlsv1.2 -sSf
https://sh.rustup.rs | sh
$
source "$HOME/.cargo/env"
$
cargo new myapp && cd myapp && cargo build --release
```

Compilers & Runtimes: Host vs. Container



On the Host OS

Inside a Container (Ubuntu base image)

```
FROM ubuntu:26.04 AS build
RUN apt update && apt install -y build-essential && \
    g++ -O2 -o app main.cpp
FROM ubuntu:26.04-minimal
COPY --from=build /app /usr/local/bin/app
```

Building a Multi-Language Container Image



Containerfile.multi

```
FROM ubuntu:26.04 AS gobuild
RUN apt update && apt install -y golang-go && go
build -o /out/svc ./cmd/svc

FROM ubuntu:26.04 AS cppbuild
RUN apt update && apt install -y build-essential &&
g++ -O2 -o /out/worker worker.cpp

FROM ubuntu:26.04-minimal
COPY --from=gobuild /out/svc /usr/local/bin/svc
COPY --from=cppbuild /out/worker
/usr/local/bin/worker
```

Native AI/ML Toolkits & Fleet Observability

C

Native, out-of-the-box support in 26.04 makes Ubuntu a first-class AI/ML development and production platform

L

Apply critical kernel security fixes without rebooting — now available for Arm-based servers too

F

Fleet-wide patch management, monitoring, and compliance reporting across hundreds of Ubuntu machines

P

Standard Linux profiling tools for CPU, memory, and I/O bottleneck analysis

You're Ready to Administer Ubuntu 26.04 LTS

From package management to TPM-backed encryption and full virtualization — the full stack, junior to senior.

Practice	Rebuild every command shown in a lab VM — muscle memory beats memorization.
Go Deeper	Read the official Ubuntu 26.04 LTS release notes for full technical detail.
Stay Current	Ubuntu ships security & feature updates on a predictable LTS cadence.
Thank You	Questions and feedback are always welcome — happy administering!

07

MODULE 7

AI Tools & Free Learning Resources

One more module: honest guidance on using AI tools to troubleshoot Ubuntu, then find out how to keep learning for free — including this course creator's complete 43-chapter Linux book, in English and Dari.

AI-Assisted Troubleshooting

43 Free Chapters

eBook / Paperback / Hardcover

Subscribe & Share

Using AI Tools to Troubleshoot Ubuntu

Unlike RHEL's built-in Lightspeed assistant, Ubuntu doesn't ship a native AI admin tool — but the same AI-assisted workflow works great with any general-purpose assistant.

P

journalctl -u nginx --since -10m | llm "why did this fail?" turns raw logs into plain-English answers

E

Paste an unfamiliar command into any AI chat and ask it to explain each flag before executing it

C

Cross-check AI suggestions against askubuntu.com and the official Ubuntu docs for anything destructive

R

Treat AI-suggested commands the same as a stranger's Stack Overflow answer — read it before running it

A Real AI-Assisted Troubleshooting Example

ai-troubleshoot.sh

```
$
pip install llm
$
llm keys set openai   # or any supported provider
$
journalctl -u nginx --since -10m | llm "why did nginx
fail to start?"
  -> Port 80 is already in use by apache2.service.
  suggested fix:
    sudo systemctl stop apache2 && sudo systemctl
start nginx
# Read it, understand it, THEN run it yourself:
$
sudo systemctl stop apache2 && sudo systemctl start
nginx
```

NOTE No AI tool replaces understanding the fix — use it to learn the reasoning, not just to copy-paste commands.

Read the Complete 43-Chapter Linux Book — Free

43

Chapters free to read online

2

Languages: English & Dari

3

Starter chapters also as PDF

Prefer Print or Kindle? Get the Complete Book

Amazon KDP

- eBook — $9.99
- Paperback — $34.99
- Hardcover — $44.99

Barnes & Noble Press

- eBook — $9.99
- Paperback — $24.99
- Hardcover — $44.99

Available in English, with a full Dari-language edition too — every link is on the MyWebUniversity.com homepage.

Enjoyed This Course? Let's Stay Connected

Free Linux, DevOps, and Enterprise Linux training — new videos every week.

Subscribe	@MyWebUniversity (English) and @MyWebUniversity-Dari (Dari) — 2,600+ free videos and counting
Comment	Tell us which Ubuntu or Linux topic you want covered next
Share	Send this video to a teammate studying for their next Linux certification
Read & Buy the Book	Free chapters, the full 43-chapter book, and every link — all on MyWebUniversity.com