🐧 What Makes Linux Different

- **Free** and open-source: no licensing fees, easy to install anywhere.
- **Live boot** capability: run a full OS from a USB stick to recover damaged systems.
- **Multi-user** design: multiple users can log in concurrently with isolated permissions.
- **UNIX heritage**: stable, secure kernel and mature command-line tools.
- **Package management**: repositories provide thousands of free applications with automatic dependency handling.

> **Definition – Distribution (distro)**: a customized stack of software packaged with the Linux kernel, delivering a complete operating system tailored for specific purposes (desktop, server, security, IoT, etc.).

Common distro families

Purpose	Example Distros
Security / anti-hacking	**Kali Linux**, **Parrot**
Consumer desktop	**Linux Mint**, **Elementary OS**
Lightweight (old hardware)	**Puppy Linux**, **LXLE**
IoT & embedded	**Snappy Ubuntu Core**
Enterprise server	**CentOS**, **OpenSUSE**, **Ubuntu Server**

🛠️ Basic Survival Skills

Navigation tools

Command	Purpose
ls	List directory contents
ls -l	Long format with permissions, owner, size, timestamp
ls -lh	Human-readable sizes (KB, MB, GB)
pwd	Show present working directory
cd	Change directory (relative or absolute)
cd ..	Move up one level
cd (no args)	Return to home directory

```
$ ls -lh /var/log
$ pwd
$ cd /etc
```

File-management tools

Command	Action
cat file	Print file contents (no editing)
less file	View file page-by-page, scroll with arrows/PGUP/PGDN
touch file	Create empty file or update timestamp
mkdir dir	Create a new directory
rmdir dir	Remove empty directory
rm file	Delete a file

rm -r dir	Recursively delete a directory tree
cp src dest	Copy file or directory
mv src dest	Move or rename file/directory

Permissions & sudo

- Regular users operate with limited rights.
- Prefix a command with **sudo** to execute it as the **root** user after password authentication.

```
$ sudo cat /etc/shadow
```

Getting Help

Tool	How to use
man cmd	Show the manual page for **cmd**
info cmd	Interactive info system, often more detailed
Internet	Search forums (Server Fault, LinuxQuestions) and official docs

> **Tip**: Inside man, type /pattern to search, n to go to next match.

📂 The Linux File System

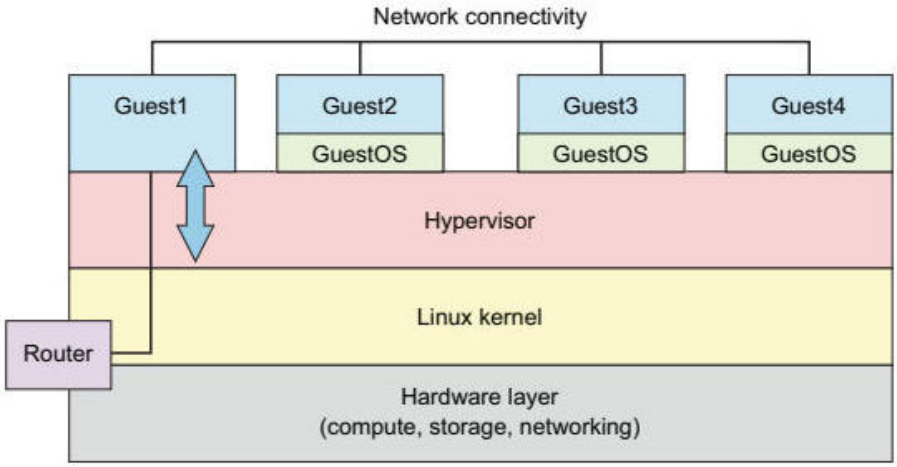
- A **file system** maps logical filenames to physical storage blocks, providing a hierarchical directory structure.
- The root of all directories is **/** (forward slash).
- **Inodes** store metadata (owner, permissions, timestamps) for each file/directory.

Top-level directories (UNIX Filesystem Hierarchy Standard)

Directory	Typical contents
/bin	Essential user binaries
/sbin	System binaries (admin tools)
/lib	Shared libraries
/usr	Secondary hierarchy (applications, docs)
/etc	Host-specific configuration files
/var	Variable data (logs, caches)
/home	User home directories
/tmp	Temporary files (cleared on reboot)
/dev	Device files
/proc	Pseudo-filesystem exposing kernel info
/sys	Pseudo-filesystem for hardware/device tree

🔧 Virtualization Overview

Virtualization lets a single physical host present multiple isolated compute environments (VMs or containers) to run independent workloads.



The diagram shows a hardware layer (compute, storage, networking) under a Linux kernel, a hypervisor, and several guest OS instances, each with its own network connectivity. This visualizes how resources are shared while keeping guests isolated.

> **Definition – Hypervisor**: software that abstracts physical hardware and allocates virtual resources to guest operating systems.
> **Definition – Container**: a lightweight virtualization method where multiple isolated user spaces share the host's kernel, providing near-native performance.

📦 Working with VirtualBox

Installing VirtualBox

- **Debian/Ubuntu** (APT):

```
sudo apt update
sudo apt install virtualbox
```

- **Fedora/CentOS** (DNF/YUM):

```
sudo dnf install dkms kernel-devel patch
sudo dnf install VirtualBox-5.1   # version may differ
```

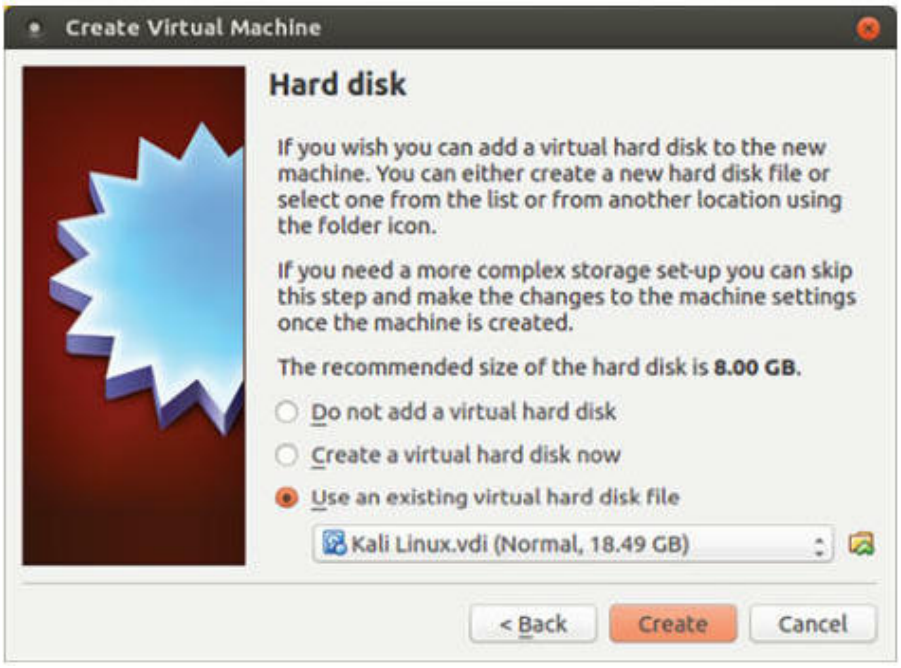
Package-manager vs distro table

Package Manager	Used By
apt	Debian, Ubuntu, Mint, Kali
yum / dnf	Red Hat, CentOS, Fedora, openSUSE (via zypper)

Defining a Virtual Machine (VM) – GUI Steps

1️⃣ Click **New** → give the VM a descriptive **name**.
2️⃣ Choose **Type** (Linux) and **Version** (e.g., Ubuntu 64-bit).
3️⃣ Allocate RAM (default 768 MiB; adjust based on host capacity).
4️⃣ Choose a virtual hard-disk option (create new or use existing).

Hard-disk options (image)



The screenshot shows the "Hard Disk" dialog where you can select a new VDI file, choose between Dynamically Allocated (grows with usage) or Fixed Size (pre-allocates full space). Selecting Dynamically Allocated conserves host storage for most test environments.

Installing an OS on the VM

1. Download an **ISO** image of the desired Linux distribution.
2. In VirtualBox, open **Settings → Storage**, attach the ISO to the virtual DVD drive.
3. Start the VM; the ISO boots and begins the OS installer.
4. Follow the installer prompts (partitioning, user creation, etc.).
5. After installation, detach the ISO and reboot.

Network configuration

- **Bridged Adapter**: VM appears as a separate host on the same LAN (good for inter-VM communication).
- **NAT**: Simpler, provides internet access but isolates VM from LAN peers.

📂 Managing VMs from the Command Line

VirtualBox ships **vboxmanage** for headless operations.

Command	Description
vboxmanage list vms	List all registered VMs
vboxmanage clonevm "BaseVM" --name "CloneVM" --register	Clone a VM (full or linked)
vboxmanage export "VMName" -o vm.ova	Export VM to **OVA** (portable appliance)
vboxmanage import vm.ova	Import an OVA file
vboxmanage startvm "VMName" --type headless	Power on a VM without GUI

Example: Clone a Kali Linux template

```
sudo vboxmanage clonevm --register Kali-Linux-template --name newkali
```

📦 Linux Containers (LXC)

Containers provide fast, lightweight isolation by sharing the host kernel.

Installing LXC

- **Ubuntu**

```
sudo apt update
sudo apt install lxc
```

- **CentOS / RHEL**

```
sudo yum install epel-release
sudo yum install lxc lxc-templates libcap-devel libcgroup busybox bridge-utils
```

Creating Your First Container

```
sudo lxc-create -n myContainer -t ubuntu
```

- -n = container name
- -t = template (Ubuntu, CentOS, etc.)

Managing Containers

Command	Action
lxc-start -n myContainer -d	Start container in background
lxc-attach -n myContainer	Open a shell inside the running container
lxc-stop -n myContainer	Gracefully stop
lxc-destroy -n myContainer	Delete container
lxc-ls --fancy	List containers with state, IP, etc.

Inspecting the container's filesystem

```
sudo su
cd /var/lib/lxc/myContainer
ls
# you'll see "config" and "rootfs" directories
cd rootfs
ls
# standard Linux hierarchy appears here
```

> **Definition – LXC rootfs**: the container's root filesystem, stored on the host under **/var/lib/lxc/**, allowing direct editing even when the container is stopped.

📚 Summary of Key Concepts

- **Linux** offers free, open-source, multi-user OS with robust package management.
- **Basic CLI skills** (navigation, file ops, sudo) are essential for any admin task.
- **VirtualBox** (type-2 hypervisor) lets you create full VMs; use **vboxmanage** for scripting.
- **LXC containers** share the host kernel, start in seconds, and consume minimal resources.
- **Package managers** (APT, DNF/YUM) keep software up-to-date and resolve dependencies automatically.

- **Virtualization architectures** (hypervisor vs container) determine isolation level and performance trade-offs.

📖 Key Terms

- **Virtualization** – logical sharing of compute, storage, and networking resources among multiple isolated environments.
- **Hypervisor** – software layer that abstracts hardware for guest OSes (type 1 runs directly on hardware; type 2 runs on a host OS).
- **Container** – lightweight, kernel-shared environment that behaves like a full OS instance.
- **Inode** – metadata structure representing a file or directory on disk.
- **Dynamic allocation** – virtual disk grows only as data is written; saves host space.
- **Fixed-size disk** – pre-allocates the full declared size; may improve VM I/O performance.

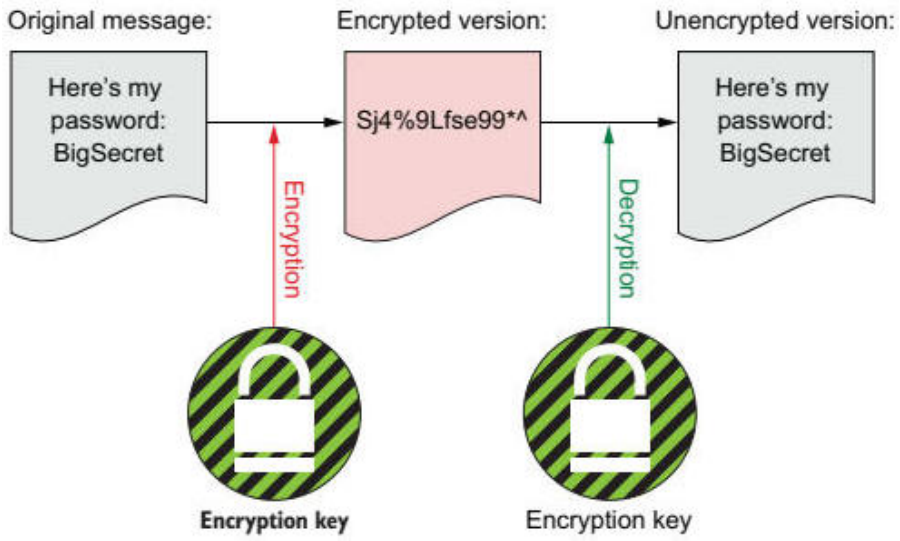
📌 Security Best Practices (quick reference)

- Use **sudo** instead of logging in as root.
- Install software through the distro's **package manager** to benefit from signed repositories.
- Verify **checksums** (shasum or sha256sum) of downloaded ISO files before installation.
- Keep the host system and all VMs/containers **patched** via regular apt update && apt upgrade (or dnf update). ## 🔐 Encryption Basics

> **Encryption key** – a secret value used to transform plaintext into ciphertext and back again. Only holders of the key can decrypt the data.

- Plain-text → **encryption algorithm** → **ciphertext** (gibberish)
- Ciphertext + same **key** (applied in reverse) → original plaintext

Symmetric encryption (same key for encrypt & decrypt) is illustrated in Figure 3.1.



The diagram shows a message being encrypted to "Sj4%9Lfse99^" and then decrypted back using a single padlock-icon key.*

🖥️ SSH & OpenSSH Overview

- **SSH (Secure Shell)** encrypts each data packet before transmission and decrypts it on arrival, making remote logins indistinguishable from Telnet to the user.
- Designed in the 1990s for UNIX-like systems; OpenSSH is the de-facto implementation, now native on Windows.

Installing & Verifying OpenSSH

Command	Purpose
apt install openssh-server	Installs server side (host) package
dpkg -s openssh-client	Checks client package status
systemctl status ssh	Shows if the SSH service is active
systemctl enable ssh	Starts SSH automatically on boot
systemctl start ssh	Starts the service now
systemctl stop ssh	Stops the service

Typical layout (Figure 3.2): client PC runs openssh-client; remote host runs openssh-server.

```
df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            3.5G     0  3.5G   0% /dev
tmpfs           724M  1.5M  722M   1% /run
/dev/sda2       910G  178G  686G  21% /
tmpfs           3.6G  549M  3.0G  16% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           3.6G     0  3.6G   0% /sys/fs/cgroup
/dev/sda1       511M  3.4M  508M   1% /boot/efi
tmpfs           724M   92K  724M   1% /run/user/1000
/dev/sdb1       1.5G  1.5G     0 100% /mnt/UB-16
```

This partition was created during installation to enable UEFI booting.

sdb1 is a USB thumb drive containing an Ubuntu live boot image.

Shows client and server packages involved in an encrypted SSH session.

Logging into a Remote Server

```
ssh ubuntu@10.0.3.144
```

- First connection prompts to confirm the host's fingerprint.
- After acceptance, password is requested (unless password-less is set up).

Testing connectivity

```
ping 10.0.3.144
```

Successful ping shows ttl=64 and low latency; failure shows "Destination Host Unreachable".

🔑 Password-Free SSH Access

What is a Passphrase?

> A passphrase is a longer, often spaced, secret used to protect a private SSH key.

Generating a Key Pair

```
ssh-keygen           # defaults to RSA, stores in ~/.ssh/id_rsa & id_rsa.pub
```

- Press Enter to accept the default file name (id_rsa).
- Optionally add a passphrase for extra security.

```
ubuntu@base:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
    /home/ubuntu/.ssh/id_rsa.
Your public key has been saved in
    /home/ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:1wQzEnybUSOFpYEvmbxVPZjy1uGAV6Tnn5m1w0qD5T8 ubuntu@base
The key's randomart image is:
+---[RSA 2048]----+
|        .oo**=*o  |
|         o.*B*o+  |
|        . =.Oo+.o |
|         = =oooo  |
|         S+..... .|
|         ..  + ..*|
|            . + B.|
|             . +E.|
|              . ..|
+----[SHA256]-----+
```

Displays the name of my local client, base

Helps to prevent man-in-the-middle attacks

The location and name of your new key pair

The key's randomart, a visual cue, also helps to prevent man-in-the-middle attacks.

Shows prompts for file location, passphrase, and the resulting key fingerprint & randomart.

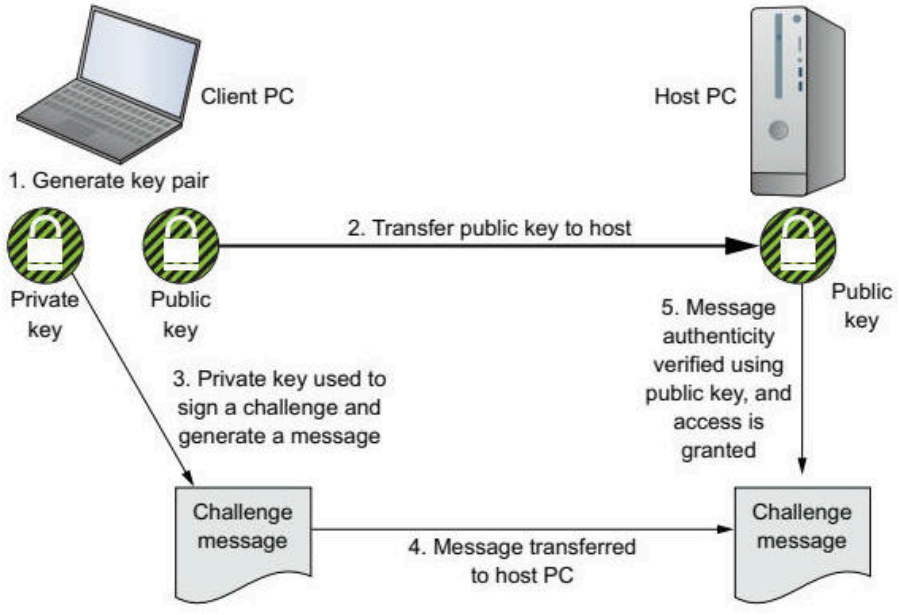
Copying the Public Key to the Host

```
# Create .ssh directory on remote host (if needed)
ssh ubuntu@10.0.3.142 "mkdir -p .ssh"

# Append public key to authorized_keys in one piped command
cat ~/.ssh/id_rsa.pub | ssh ubuntu@10.0.3.142 "cat >> .ssh/authorized_keys"
```

After this, subsequent ssh ubuntu@10.0.3.142 logs in **without a password**.

Visual of the flow (Figure 3.3)



Shows client generating a key pair, sending the public key, and the host verifying a signed challenge.

Using Multiple Keys

```
ssh -i ~/.ssh/mykey.pem ubuntu@10.0.3.142
```

- -i specifies the private key file (often with .pem extension for cloud VMs).

📁 Secure File Transfer with SCP

```
scp .ssh/id_rsa.pub ubuntu@10.0.3.142:/home/ubuntu/.ssh/authorized_keys
```

- Overwrites authorized_keys if it exists (use with caution).

Alternative: ssh-copy-id (preferred)

```
ssh-copy-id -i ~/.ssh/id_rsa.pub ubuntu@10.0.3.142
```

🖼️ Remote Graphic Programs (X11 Forwarding)

- Enable on server: edit /etc/ssh/sshd_config → X11Forwarding yes
- Enable on client: edit /etc/ssh/ssh_config → ForwardX11 yes
- Restart SSH on both sides: systemctl restart ssh
- Connect with -X flag:

```
ssh -X ubuntu@10.0.3.142
gnome-mines   # launches Minesweeper locally but runs on remote host
```

🛠️ Linux Process Management

Viewing Processes (ps)

```
ps -ef | grep init
```

Shows all processes; the first line usually contains PID 1 (/sbin/init or /lib/systemd/systemd).

Process Tree (pstree)

```
pstree -p
```

Displays hierarchical relationships; the root is systemd(1) on modern distros.

systemd & systemctl

- systemd replaces the classic init process.
- systemctl is the user-level tool to start/stop/enable services.

Figure 3.4 illustrates systemd's role as a process manager.

📂 File Permissions & Ownership

Permission Bits Table

Symbol	Meaning	Numeric Value
r	Read	4
w	Write	2
x	Execute	1
-	No permission	0

- Permissions are expressed as **owner / group / others** (e.g., 755 = rwx r-x r-x).

Changing Permissions

```
chmod 771 /bin/zcat          # owner rwx, group rwx, others x
chmod o-r /bin/zcat          # remove read for others
chmod g+w /bin/zcat          # add write for group
```

Changing Ownership

```
chown otheruser:otheruser newerfile
```

- user:group format; requires root privileges.

📦 Archiving & Compression

Creating a Simple Archive

```
tar cvf archivename.tar *
```

- c = create, v = verbose, f = file name.

Adding gzip Compression

```
tar czvf archivename.tar.gz *.mp4
```

- z invokes gzip; output extension .tar.gz.

Splitting Large Archives

```
split -b 1G archivename.tar.gz "archivename.tar.gz.part"
# To reassemble:
cat archivename.tar.gz.part* > archivename.tar.gz
```

Streaming an Archive Directly to Remote Host

```
tar czvf - importantstuff/ | ssh user@10.0.3.141 "cat > /home/user/myfiles.tar.gz"
```

- - tells tar to write to **stdout**, which is piped over SSH.

Figure 4.4 (not shown) would depict this “no-local-copy” workflow.

Finding Files for Archiving

```
find /var/www/html/ -iname "*.mp4" -exec tar -rvf videos.tar {} \;
```

- -exec runs tar -r (append) for each matched file.

💿 Disk Imaging with dd

Command	Description
dd if=/dev/sda of=~/sdadisk.img	Image entire disk sda to a file
dd if=/dev/sda2 of=~/partition2.img bs=4096	Image a single partition with block size 4096
dd if=sdadisk.img of=/dev/sdb	Restore image to another disk
dd if=/dev/zero of=/dev/sda1	Overwrite a partition with zeros (secure erase)
dd if=/dev/urandom of=/dev/sda1	Overwrite with random data (harder to recover)

> Warning: dd is unforgiving; a typo can wipe the wrong drive.

🔄 Synchronizing Archives with rsync

```
rsync -av * user@10.0.3.141:syncdirectory
```

- -a = archive mode (preserves permissions, timestamps, recursion)
- -v = verbose

Running the same command later copies only **new or changed** files, greatly reducing bandwidth.

📅 Automated Backups

Bash Script Example (system file backup)

```
#!/bin/sh
cd /var/backups || exit 0
for FILE in passwd group shadow gshadow; do
    test -f /etc/$FILE || continue
    cmp -s $FILE.bak /etc/$FILE && continue
    cp -p /etc/$FILE $FILE.bak && chmod 600 $FILE.bak
done
```

- Checks existence, compares with existing backup, copies if different, tightens permissions.

Scheduling with cron

- Place executable scripts in /etc/cron.daily/, /etc/cron.hourly/, etc.
- For custom timing, add a file in /etc/cron.d/ like:

```
21 5 * * 1 root apt update && apt upgrade -y
```

(Runs every Monday at 05:21.)

Using anacron for Inconsistent Power

/etc/anacrontab entries run after system boot:

```
1 5 cron.daily run-parts --report /etc/cron.daily
```

- Runs daily, 5 minutes after boot.

Systemd Timers (advanced)

Service file (site-backup.service)

```
[Unit]
Description=Backup Apache website

[Service]
Type=simple
ExecStart=/home/username/site-backup.sh
```

Timer file (site-backup.timer)

```
[Unit]
Description=Backup Apache website - daily

[Timer]
OnCalendar=*-*-* 5:51:00
Unit=site-backup.service

[Install]
WantedBy=multi-user.target
```

Activate:

```
systemctl start site-backup.timer
systemctl enable site-backup.timer
```

- systemctl is-enabled and systemctl is-active confirm status.

Figure 5.4 (service/timer listing) illustrates timer metadata.

☁️ Backing Up to AWS S3

Installing AWS CLI

```
pip3 install --upgrade --user awscli
```

- Requires Python 3 and pip.

Configuring Credentials

```
aws configure
# Enter Access Key ID, Secret Access Key, region, output format
```

Creating a Bucket & Syncing

```
aws s3 mb s3://linux-bucket3040
aws s3 sync /home/username/dir2backup s3://linux-bucket3040
```

- First sync uploads everything; subsequent runs transfer only changes.

Automating S3 Sync with Cron

Add to /etc/cron.d/ (run daily at 05:47):

```
47 5 * * * username /usr/local/bin/aws s3 sync /home/username/dir2backup s3://linux-bucket3040
```

📋 Summary of Key Concepts

- **Encryption** turns readable data into ciphertext; the same key decrypts it (symmetric).
- **SSH** provides encrypted remote command execution; OpenSSH is the standard implementation.
- **Password-less SSH** relies on RSA/ECDSA/ED25519 key pairs; the public key lives on the host, the private key stays on the client.
- **systemd** (via systemctl) manages services; ps, pstree inspect processes.
- **Permissions** (rwx) map to numeric codes (4/2/1); use chmod and chown to modify.
- **tar**, **gzip**, **split**, **dd**, **rsync** are core tools for archiving, compressing, imaging, and synchronizing data.
- **Automation**: Bash scripts + cron/anacron/systemd timers ensure regular backups.
- **Cloud backup**: AWS CLI + S3 bucket + aws s3 sync provide off-site storage.

📚 Quick Reference Commands

Task	Command
Check OpenSSH client	dpkg -s openssh-client
Start SSH service	systemctl start ssh
Enable SSH on boot	systemctl enable ssh
Generate RSA key pair	ssh-keygen
Copy public key (manual)	cat ~/.ssh/id_rsa.pub | ssh user@host "cat >> .ssh/authorized_keys"
Password-less login test	ssh user@host
Secure copy file	scp localfile user@host:/remote/path
Remote graphical app	ssh -X user@host
List processes	ps -ef
Show process tree	pstree -p
Change file mode	chmod 755 filename
Change ownership	chown user:group filename
Create tar.gz archive	tar czvf archive.tar.gz /path/*
Split large file	split -b 1G largefile part_

Create disk image	dd if=/dev/sda of=~/disk.img bs=4096
Sync with rsync	rsync -av /src/ user@host:/dest/
Schedule with cron	crontab -e (add entry)
Run anacron job	edit /etc/anacrontab
Systemd timer enable	systemctl enable mytimer.timer
AWS S3 sync	aws s3 sync /local/dir s3://bucket

---## 🛡️ Data Reliability & Security

- **Never expose** AWS access keys, passwords, or encryption key pairs publicly.
- Protect all credentials and keep them out of version-controlled scripts.

🖥️ Command-line Review

Symbol / Command	Meaning / Use	Example
#!/bin/bash	**Shebang line** – tells Linux which interpreter to use for a script.	#!/bin/bash
`		`
&&	**AND condition** – execute right-hand command only if left succeeds.	cmd1 && cmd2
test -f /etc/filename	Checks if a file exists.	test -f /etc/passwd && echo "Exists"
chmod +x script.sh	Makes a script executable.	chmod +x upgrade.sh
pip3 install --upgrade --user awscli	Installs/updates AWS CLI via pip.	pip3 install --upgrade --user awscli
aws s3 sync <src> <dest>	Synchronises a local directory with an S3 bucket.	aws s3 sync /home/user/dir s3://my-bucket
cron: 21 5 * * 1 root apt update && apt upgrade	Runs two apt commands at **05:21 AM every Monday**.	–
NOW=$(date +"%m_%d_%Y")	Assigns current date to variable NOW.	–
systemctl start site-backup.timer	Starts a **systemd timer**.	–

```
#!/bin/bash
# Example: backup script
if test -f /etc/important.conf; then
    cp /etc/important.conf /backup/
fi
```

❓ Test Yourself – Linux Scripting & AWS

> **Q1:** Character used to start a comment in a Linux script? **Answer:** #
> **Q2:** Purpose of ||? **Answer: OR** condition.
> **Q3:** Data type stored in /etc/shadow? **Answer:** Encrypted account passwords.
> **Q4:** Command to create a new AWS S3 bucket? **Answer:** aws s3 mb s3://mybucket
> **Q5:** Command to edit a personal crontab? **Answer:** crontab -e
> **Q6:** Cron line that runs each Monday morning? **Answer:** 21 5 * * 1 root apt update && apt upgrade
> **Q7:** Tool unsuitable for computers that aren't always on? **Answer:** crontab (use anacron or systemd timer).
> **Q8:** Purpose of systemctl enable site-backup.timer? **Answer:** Configures the timer to start on boot.

Answer Key: 1-c, 2-a, 3-c, 4-b, 5-d, 6-b, 7-a, 8-b

🛠️ Emergency Tools – System Recovery

6.1 Working in Recovery/Rescue Mode

- **GRUB** (GNU GRand Unified Bootloader) presents a menu even if Linux fails to boot fully.
- Press **Right Shift** during power-on to force the GRUB menu.

6.1.1 The GRUB Bootloader

- Firmware (BIOS/UEFI) loads the **MBR** → GRUB → selected kernel image → root filesystem → init/systemd.

6.1.2 Ubuntu Recovery Mode

- Offers tools: **clean**, **dpkg**, **fsck**, **root shell**, etc.
- The **root** option provides a minimal Bash shell for manual repairs.

6.1.3 CentOS Rescue Mode

- Boots a **rescue kernel** that drops directly to a single-user root shell.

6.1.4 Finding Rescue Scripts (Ubuntu)

```
$ locate recovery-mode
/lib/recovery-mode
/lib/recovery-mode/l10n.sh
/lib/recovery-mode/options
/lib/recovery-mode/recovery-menu
/lib/recovery-mode/options/apt-snapshots
/lib/recovery-mode/options/clean
/lib/recovery-mode/options/dpkg
/lib/recovery-mode/options/failsafeX
/lib/recovery-mode/options/fsck
/lib/recovery-mode/options/grub
/lib/recovery-mode/options/network
/lib/recovery-mode/options/root
/lib/recovery-mode/options/system-summary
```

The 110n.sh script sets appropriate environment variables for the menu.

The recovery-menu script file

The script for reducing disk usage

The screenshot shows /lib/recovery-mode/ containing the main recovery-menu script and individual option scripts (e.g., fsck). Inspecting these scripts reveals how each menu item works.

💾 Building a Live-Boot Recovery Drive

6.2 System Rescue Images

- **Boot-Repair** – fixes GRUB, repairs boot issues.
- **GParted Live** – partition editor for corrupted tables.
- **SystemRescueCd** – lightweight, packed with rescue tools.

6.2.2 Writing ISO to USB

```
# Verify download integrity
sha256sum systemrescuecd-x86-5.0.2.iso

# Add MBR for USB (Ubuntu host)
sudo apt install syslinux-utils
cd ~/Downloads
isohybrid systemrescuecd-x86-5.0.2.iso

# Identify target device (e.g., /dev/sdb)
```

```
sudo umount /dev/sdb
sudo dd bs=4M if=systemrescuecd-x86-5.0.2.iso of=/dev/sdb && sync
```

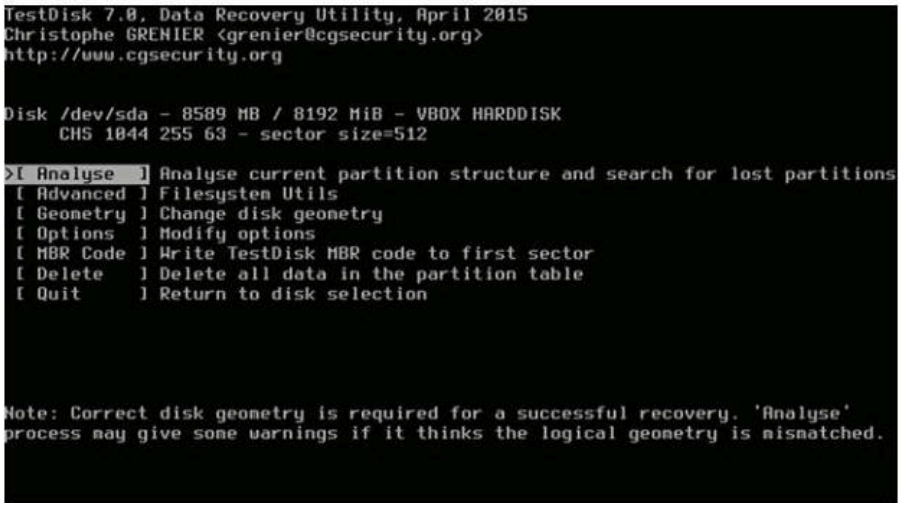
> **Tip:** df -h shows mounted filesystems; lsblk lists block devices.

🔧 Putting the Live-Boot Drive to Work

6.3.1 Testing System Memory

- Select **Test Memory** from the Ubuntu live-boot menu → runs **Memtest86+**.

6.3.2 Damaged Partitions – TestDisk



TestDisk provides a text UI for analyzing, repairing, and rebuilding partition tables.

6.3.3 File Recovery – ddrescue & PhotoRec

Tool	Purpose	Typical Command
ddrescue	Copies data while retrying bad sectors; creates a log for resume.	ddrescue -d /dev/sda1 /mnt/backup.img /mnt/ddrescue.log
photorec	Recovers files by file-type signatures, ignoring filesystem metadata.	photorec (interactive)

🔐 Password Recovery – chroot

1. Boot from a live-boot USB.
2. Identify root partition (lsblk).
3. Mount it:

```
sudo mkdir /mnt/root
sudo mount /dev/sdb1 /mnt/root
```

4. **Chroot** into it:

```
sudo chroot /mnt/root /bin/bash
passwd <username>   # set a new password
```

```
exit
sudo reboot
```

> **Note:** chroot changes the apparent root directory, letting you operate as if the mounted system were the live system.

📋 Summary & Key Terms

- **GRUB** – bootloader that loads kernel images.
- **Live-boot USB** – portable Linux environment for diagnostics and repairs.
- **TestDisk** – partition recovery utility.
- **ddrescue** – robust disk cloning with error handling.
- **PhotoRec** – file-type based data recovery.
- **chroot** – changes the root directory for a process, enabling offline system repair.

Security Best Practices

- Verify ISO hashes (sha256sum) before writing to media.
- Prefer **systemd timers** (systemctl enable …timer) over cron for machines that may be powered off.
- Keep **password vaults** and **single sign-on** (e.g., Kerberos) for credential management.

🌐 Building a LAMP Server (Linux, Apache, MySQL/MariaDB, PHP)

7.1 Installing Apache (Ubuntu)

```
sudo apt update
sudo apt install apache2
```

- Default **DocumentRoot**: /var/www/html.

7.2 Configuring DocumentRoot

```
# Verify current setting
grep DocumentRoot /etc/apache2/sites-available/000-default.conf
# Example output:
# DocumentRoot "/var/www/html"
```

- To serve a custom site, edit or add a new .conf file in /etc/apache2/sites-available/ and enable it with a2ensite.

7.3 Installing MariaDB

```
sudo apt install mariadb-server
sudo systemctl start mariadb
sudo systemctl enable mariadb
```

- Secure installation: sudo mysql_secure_installation.

7.4 Creating a Database & User

```
CREATE DATABASE wikidb;
CREATE USER 'mw-admin'@'localhost' IDENTIFIED BY 'StrongPass!';
GRANT ALL PRIVILEGES ON wikidb.* TO 'mw-admin'@'localhost';
FLUSH PRIVILEGES;
```

7.5 Installing PHP & Extensions

```
sudo apt install php libapache2-mod-php php-mysql php-mbstring php-xml php-apcu php-imagick
sudo systemctl restart apache2
```

- Verify with phpinfo() (create /var/www/html/info.php containing <?php phpinfo(); ?>).

7.6 Deploying MediaWiki

1. Download & extract:

```
wget https://releases.wikimedia.org/mediawiki/1.30/mediawiki-1.30.0.tar.gz
tar xzf mediawiki-1.30.0.tar.gz
sudo cp -r mediawiki-1.30.0/* /var/www/html/
```

2. Adjust ownership: sudo chown -R www-data:www-data /var/www/html/.
3. Complete web-based installer (set database credentials, admin user).

☁️ Nextcloud Installation (Snap vs Manual)

8.1 Snap Installation (Ubuntu)

```
sudo snap install nextcloud
```

- Provides a fully functional Nextcloud instance with minimal configuration.

8.2 Manual Installation (Ubuntu)

1. Install LAMP stack (see Section 7).
2. Download Nextcloud tarball:

```
wget https://download.nextcloud.com/server/releases/nextcloud-12.0.0.tar.bz2
tar xjf nextcloud-12.0.0.tar.bz2
sudo cp -r nextcloud /var/www/
sudo chown -R www-data:www-data /var/www/nextcloud
```

3. Create Apache site configuration:

```
# /etc/apache2/sites-available/nextcloud.conf
Alias /nextcloud "/var/www/nextcloud/"

<Directory /var/www/nextcloud/>
  Options +FollowSymlinks
  AllowOverride All
  <IfModule mod_dav.c>
    Dav off
  </IfModule>
  SetEnv HOME /var/www/nextcloud
  SetEnv HTTP_HOME /var/www/nextcloud
</Directory>
```

4. Enable site & required modules:

```
sudo a2enmod rewrite headers
sudo a2ensite nextcloud
```

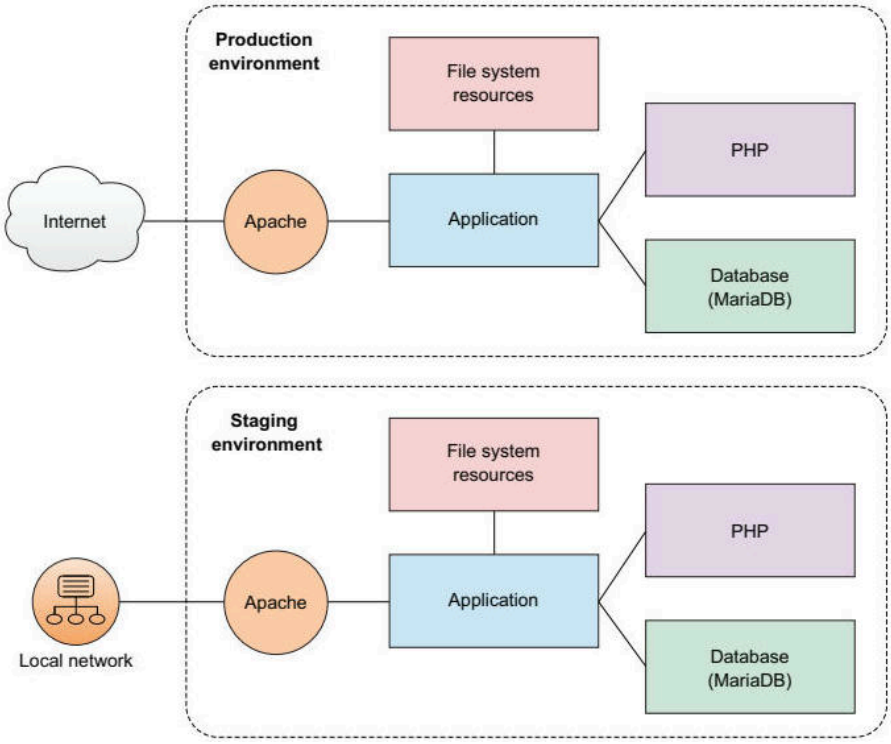
```
sudo systemctl reload apache2
```

5. Complete web installer (create admin account, connect to MariaDB).

8.3 External Storage – Amazon S3

```
aws s3 mb s3://my-nextcloud-bucket
# In Nextcloud Admin → External Storage, select "Amazon S3"
# Provide bucket name, Access Key, Secret Key
```

- Successful connection shows a green check mark.



The External Storages page confirms a working S3 backend.

🔐 Securing Your Web Server

- **Firewall**: allow only necessary ports (e.g., HTTP 80, HTTPS 443, SSH 22).

```
# Ubuntu (ufw)
sudo ufw allow http
sudo ufw allow https
sudo ufw allow ssh
sudo ufw enable
```

- **TLS/SSL**: obtain certificates (e.g., via Let's Encrypt) and configure Apache with SSLEngine on.
- **File Permissions**: ensure web-accessible files are owned by www-data, not root.
- **Regular Updates**:

```
sudo apt update && sudo apt upgrade -y
```

- **Backup Strategy**: automate backups, store off-site, test restores regularly.

📚 Quick Reference Tables

Bash Operators

Operator	Meaning
&&	Execute right command **only if** left succeeds.
`	
#	Comment line.
#!/bin/bash	Shebang – specifies interpreter.

Common Recovery Tools

Tool	Primary Use
GRUB	Bootloader; selects kernels & recovery modes.
TestDisk	Repair partition tables, recover lost partitions.
ddrescue	Clone disks while handling read errors; log-resume.
PhotoRec	File-type based recovery from damaged media.
chroot	Change root directory to repair offline system.
SystemRescueCd	Live-boot environment with extensive utilities.
Boot-Repair	Fix GRUB and boot configuration issues.

🧭 Further Learning

- Review **systemd timers** vs **cron** for scheduled tasks.
- Explore **SELinux** or **AppArmor** for mandatory access control.
- Practice creating **staging environments** (see Figure 9.1) to test updates safely.

---## 🔥 Firewalls: firewalld & ufw

firewalld (systemd-based) is the default on Red Hat/CentOS and can be installed on Debian/Ubuntu.

```
# apt update
# apt install firewalld
# firewall-cmd --state
running
```

- --add-port=80/tcp and --add-port=443/tcp open HTTP/HTTPS.
- --add-service=http and --add-service=https use names from */etc/services*.
- --permanent makes the rule survive reboots; --reload applies changes immediately.

UFW (Uncomplicated Firewall) is disabled by default; enable it after allowing SSH:

```
# apt install ufw
# ufw allow ssh          # allow port 22 before enabling
# ufw enable
```

Command	Effect
ufw allow 80	Open HTTP (port 80)
ufw allow 443	Open HTTPS (port 443)
ufw allow from 10.0.3.1 to any port 22	Permit SSH **only** from 10.0.3.1
ufw delete 2	Remove the second rule listed by ufw status

> **Definition** – *Firewall*: A network security device (software or hardware) that filters inbound/outbound traffic based on **protocol**, **port**, and **source/destination** criteria.

🔐 Restricting SSH Access (firewalld)

```
# firewall-cmd --permanent --remove-service=ssh
# firewall-cmd --reload               # SSH blocked
# firewall-cmd --add-rich-rule='rule family="ipv4" source address="192.168.1.5" port protocol="tcp" port="22"
# firewall-cmd --reload               # Allow only 192.168.1.5
```

> **Note** – Manipulating firewall rules over an active SSH session can lock you out. Use a disposable VM/LXC container when experimenting.

📶 Non-Standard Ports

Changing the default service port adds **security through obscurity**. Example: move SSH from 22 → 53987.

```
# edit /etc/ssh/sshd_config
Port 53987
# systemctl restart ssh
# firewall-cmd --permanent --add-port=53987/tcp   # firewalld
# ufw allow 53987/tcp                              # ufw
```

Port ranges

Range	Description
1-1023	**Well-known** (reserved for standard services)
1024-49151	**Registered** (assigned to specific applications)
49152-65535	**Dynamic/Private** (safe for custom services)

🔐 Encrypting Data in Transit (TLS/HTTPS)

How TLS Handshake Works

1. Client requests server identity.
2. Server sends its **certificate** (signed by a CA).
3. Browser verifies the cert against its trusted root store.
4. Browser encrypts a **symmetric session key** with the server's public key and sends it.
5. All subsequent traffic uses the session key.

The image shows the exchange of certificates and session keys during a TLS-encrypted browser session.

Let's Encrypt & Certbot (Ubuntu Apache)

```
# apt install software-properties-common
# add-apt-repository ppa:certbot/certbot
# apt update
# apt install certbot python3-certbot-apache
# certbot --apache
```

During certbot --apache you'll be prompted to select domains (e.g., bootstrap-it.com). Certbot obtains a free 90-day certificate, configures Apache, and sets up automatic renewal.

The Certbot home page guides users through selecting their web server and OS, then shows installation commands.

🔐 Hardening SSH Authentication

```
# Edit /etc/ssh/sshd_config
PermitRootLogin no          # disallow root logins
```

```
PasswordAuthentication no   # enforce key-pair auth only
# systemctl restart sshd
```

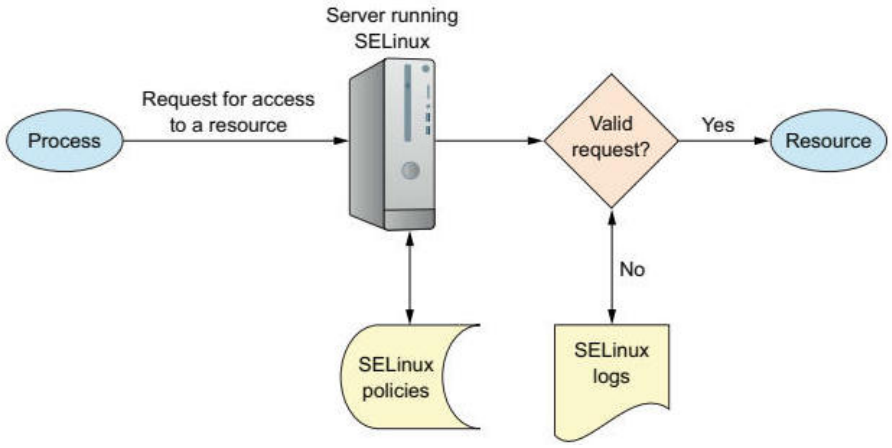
> **Principle** – *Least Privilege*: Users receive only the permissions essential for their tasks.

SELinux Overview

When enabled, SELinux enforces **mandatory access control (MAC)**, overriding traditional DAC file permissions.

```
# apt install setools policycoreutils selinux
# reboot
# sestatus
# setenforce 0      # switch to permissive (logs only)
# setenforce 1      # enforce policy
```

Config file (/etc/selinux/config)	Setting	Meaning
SELINUX=	disabled	SELinux off
	enforcing	Enforce policies
	permissive	Log violations only
SELINUXTYPE=	targeted	Restricts only selected domains
	minimum	Minimal restrictions
	mls	Multi-Level Security



The flowchart shows how a resource request is checked against SELinux policies before being granted.

Simple SELinux Policy Example

```
# ls -Z /var/www/html/
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 index.html
# chcon -t samba_share_t /var/www/html/index.html
# ls -Z /var/www/html/
-rw-r--r--. root root unconfined_u:object_r:samba_share_t:s0 index.html
# wget localhost              # → 403 Forbidden (Apache blocked by SELinux)
```

System Groups & Least Privilege

```
# groupadd app-data-group
# chown ubuntu:app-data-group datafile.txt
# chmod 770 datafile.txt
# usermod -aG app-data-group otheruser
# su - otheruser
$ cat datafile.txt   # succeeds because of group membership
```

> **Blockquote** – *Discretionary Access Control (DAC)* lets owners set permissions; *Mandatory Access Control (MAC)* (e.g., SELinux) enforces system-wide policies regardless of ownership.

📦 Container Isolation

- **Docker** containers provide lightweight process isolation.
- **Microservices** architecture runs each service (web, DB, media) in its own container, limiting blast radius of a compromise.

🛡️ OpenVPN Tunnel – Server Setup

1️⃣ Install Packages

```
# apt install openvpn easy-rsa
# ufw allow 22
# ufw allow 1194/tcp   # default OpenVPN port
# sysctl -w net.ipv4.ip_forward=1
# sysctl -p
```

2️⃣ Generate PKI (Easy-RSA)

```
# cp -r /usr/share/easy-rsa/ /etc/openvpn/
# cd /etc/openvpn/easy-rsa
# . ./vars
# ./clean-all
# ./build-ca              # creates ca.crt
# ./build-key-server server
# ./build-dh
# cp keys/{ca.crt,server.crt,server.key,dh2048.pem} /etc/openvpn/
```

Easy-RSA script summary

Script	Purpose
clean-all	Remove old keys
build-ca	Create root CA
build-key-server	Generate server key/cert
build-dh	Diffie-Hellman parameters
pkitool	Generate client keys

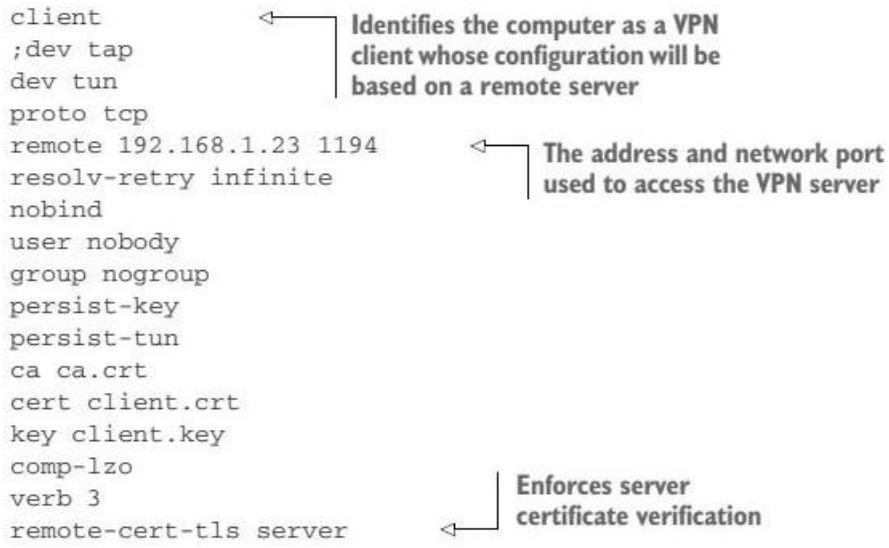
3️⃣ Server Configuration (/etc/openvpn/server.conf)

```
port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key
```

```
dh dh2048.pem
server 10.8.0.0 255.255.255.0
push "route 10.0.3.0 255.255.255.0"
keepalive 10 120
persist-key
persist-tun
status openvpn-status.log
log /var/log/openvpn.log
verb 3
```

4️⃣ Client Configuration

```
# apt install openvpn
# cp /usr/share/doc/openvpn/examples/sample-config-files/client.conf /etc/openvpn/
# edit /etc/openvpn/client.conf
```



The client.conf file lists settings such as remote server IP, protocol, and certificate verification.

Transfer the client keys:

```
# scp user@vpn-server:/etc/openvpn/ca.crt .
# scp user@vpn-server:/etc/openvpn/client.crt .
# scp user@vpn-server:/etc/openvpn/client.key .
```

Start the client:

```
# openvpn --tls-client --config /etc/openvpn/client.conf
```

5️⃣ Verify Tunnel

```
$ curl 192.168.1.23:1194   # should retrieve the web server's index.html
```

🏰 DMZ Architecture

iptables Example

```
# iptables -P INPUT DROP
# iptables -P FORWARD DROP
# iptables -P OUTPUT ACCEPT

# iptables -A FORWARD -i eth1 -o eth2 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
# iptables -A FORWARD -i eth2 -o eth1 -m state --state ESTABLISHED,RELATED -j ACCEPT

# iptables -t nat -A PREROUTING -p tcp -i eth0 -d 54.4.32.10 --dport 80 -j DNAT --to-destination 192.168.1.20
```

Chain	Default Policy
INPUT	DROP
FORWARD	DROP
OUTPUT	ACCEPT

Shorewall Configuration Files

File	Purpose
zones	Declare network zones (net, dmz, loc)
interfaces	Map zones to physical interfaces
policy	Baseline allow/deny rules between zones
rules	Exceptions and fine-grained allowances
masq	NAT (optional)
params	Global variables
conntrack	Exempt traffic from connection tracking

Sample policy file

```
net   all   DROP
loc   net   ACCEPT
fw    all   ACCEPT
*     all   REJECT
```

🖥️ Virtual Network Testing (VirtualBox)

```
# vboxmanage natnetwork add --netname dmz --network "10.0.1.0/24" --enable --dhcp on
# vboxmanage natnetwork add --netname loc --network "10.0.2.0/24" --enable --dhcp on
# vboxmanage natnetwork start --netname dmz
# vboxmanage natnetwork start --netname loc
```

Assign each VM's adapters:

- Adapter 1 → NAT (Internet)
- Adapter 2 → dmz network
- Adapter 3 → loc network

Verify interfaces inside a VM:

```
$ ip addr
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 ...
```

```
    inet 10.0.1.5/24 brd 10.0.1.255 scope global enp0s3
```

📜 System Logging

journald vs. syslogd

- **journald** stores binary logs, queried with journalctl.
- **syslogd** writes plain-text files under /var/log/.

journalctl useful options

Option	Effect
-n 20	Show last 20 entries
-f	Follow new entries (real-time)
-p err	Filter by priority (error)
--since "2024-09-01"	Entries after date
--until "2024-09-10 12:00"	Entries before date

Common syslog files (/var/log/)

File	Content
auth.log	Authentication events
boot.log	Boot-process messages
kern.log	Kernel messages
dpkg.log	Package manager actions
syslog	All-purpose catch-all log
wtmp	User login records (who, last)

syslog priority levels

Level	Description
debug	Detailed debugging info
info	General informational messages
notice	Normal but significant events
warn	Potential problems
err	Errors
crit	Critical conditions
alert	Immediate attention required
emerg	System unusable

Example: directing cron logs

```
cron.*    /var/log/cron.log
```

Log Rotation

logrotate automatically compresses and removes old logs. Typical /var/log/ files appear as:

```
auth.log
auth.log.1
auth.log.2.gz
```

📋 Command-Line Review

Command	Purpose
firewall-cmd --permanent --add-port=80/tcp	Open HTTP permanently (firewalld)
firewall-cmd --list-services	List active firewalld services
ufw allow ssh	Open SSH (port 22) via ufw
ufw delete 2	Remove second rule listed by ufw status
ssh -p53987 user@host	Connect using a non-standard SSH port
certbot --apache	Obtain & install Let's Encrypt cert for Apache
selinux-activate	Enable SELinux on Ubuntu (requires reboot)
setenforce 1	Switch SELinux to enforcing mode
ls -Z /var/www/html/	Show SELinux context of files
usermod -aG app-data-group user	Add user to a group
netstat -npl	List listening ports with PIDs
iptables -A FORWARD -i eth1 -o eth2 …	Allow forwarding between DMZ and internal net
shorewall restart	Apply Shorewall configuration
vboxmanage natnetwork add …	Create VirtualBox NAT network
journalctl -f	Follow live journal logs
logrotate /etc/logrotate.conf	Manually trigger log rotation

Test Yourself

1. Which command lists SELinux violations without enforcing them? setenforce 0
2. How do you permanently open HTTPS on firewalld? firewall-cmd --permanent --add-service=https && firewall-cmd --reload
3. Which port range is safe for custom services? **49152-65535** (dynamic/private)

All LaTeX expressions are wrapped in single $ signs; code blocks are highlighted for Bash where applicable.## 📂 Log Rotation & Management

> **Log rotation** automatically renames, compresses, and removes old log files to keep disk usage under control.

- Active log: auth.log
- First retired version: auth.log.1
- Older versions are gzipped (auth.log.2.gz, auth.log.3.gz, …).

Rotation cycle (default, weekly) is defined in /etc/logrotate.conf:

```
# rotate log files weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# create new (empty) log files after rotating old ones
create
# include package-specific configs
include /etc/logrotate.d
```

Individual services drop their own configs into /etc/logrotate.d/ (e.g., apache2, apt, mysql-server).

Example: apt log rotation (/etc/logrotate.d/apt)

```
/var/log/apt/term.log {
    rotate 12
    monthly
    compress
    missingok
    notifempty
}

/var/log/apt/history.log {
    rotate 12
    monthly
    compress
    missingok
    notifempty
}
```

- **rotate 12** → keep 12 months, then delete.
- **compress** → gzip immediately after rotation.

Tip: Forward logs to a remote log server to offload storage and centralize analysis.

🔍 Text-Processing Tools

1️⃣ grep – Pattern Matching

```
# Find authentication failures and show one line before/after each match
cat /var/log/auth.log | grep -B 1 -A 1 failure
```

- -B 1 → one line **B**efore, -A 1 → one line **A**fter.
- Generate a test entry: logger "Authentication failure"

Recursive search with line numbers

```
grep -nr daemon /path/to/manuscript
```

- -n → include line numbers, -r → recurse directories.

2️⃣ awk – Field-Based Filtering

Log entry example (/var/log/mysql/error.log):

```
2017-09-05T04:42:52.293846Z 0 [Note] Shutting down plugin 'sha256_password'
```

Count warnings correctly

```
cat error.log | awk '$3 ~ /\[Warning\]/' | wc -l
```

- $3 → third field (the priority level).
- Brackets are treated literally inside the regex (/\[Warning\]/).

3️⃣ sed – Stream Editing

Simple substitution

```
echo "hello world" | sed "s/world/fishtank/"
# → hello fishtank
```

Remove leading line numbers from a code snippet

```
sed "s/^ *[0-9]* //g" numbers.txt
```

- ^ anchors to line start, * matches spaces, [0-9]* matches any digits.

Print only directories from ls -l

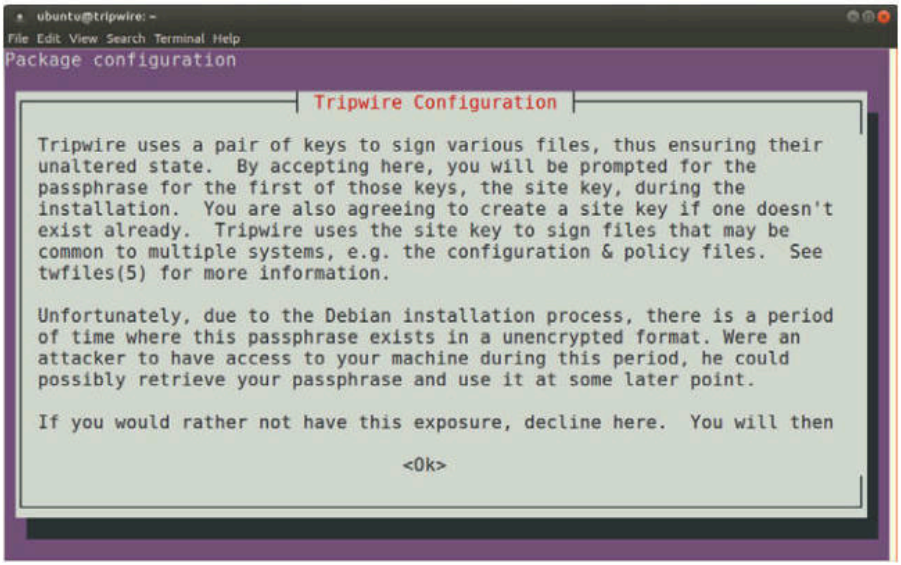
```
ls -l | sed -n '/^d/ p'
```

- -n suppresses automatic printing; /^d/ p prints lines starting with d (directory flag).

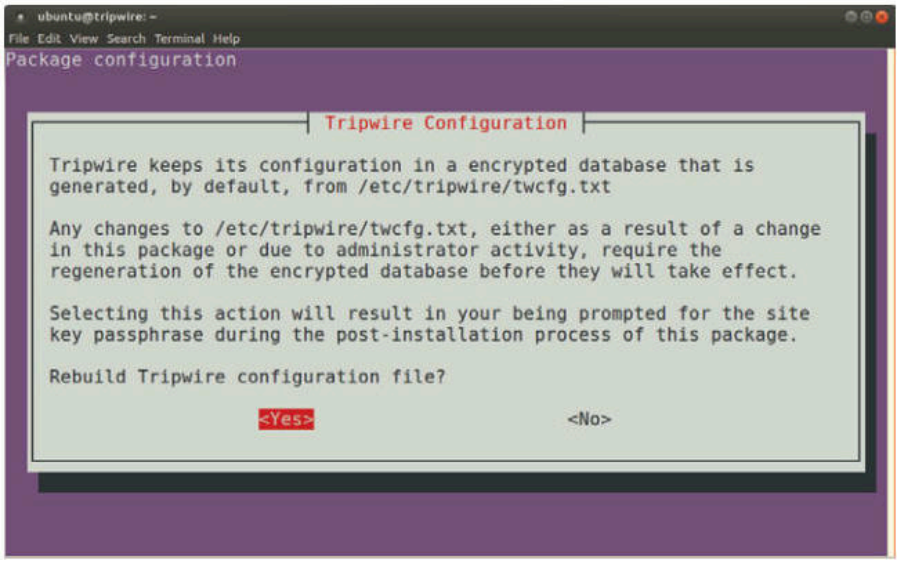
Intrusion Detection with Tripwire

> Tripwire (HIDS) records cryptographic hashes of critical files; any change triggers an alert.

Installation (Ubuntu) – Screenshots



The installer prompts for passphrases for the signing keys; the warning highlights the temporary exposure of the passphrase.



After key creation, Tripwire asks whether to rebuild the encrypted configuration file.

Key files after installation

File	Purpose
/usr/sbin/tripwire	Binary executable
/var/lib/tripwire/tripwire.twd	Database of file hashes
/etc/tripwire/tw.cfg	Encrypted configuration
/etc/tripwire/tw.pol	Encrypted policy

Configuration Workflow

1. Edit plain-text sources: twcfg.txt & twpol.txt.
2. Encrypt them:

```
twadmin --create-cfgfile --site-keyfile site.key twcfg.txt
twadmin --create-polfile twpol.txt
```

3. Secure the plain files:

```
rm /etc/tripwire/twcfg.txt /etc/tripwire/twpol.txt
```

4. Initialize the database:

```
tripwire --init
```

5. Perform a check and email the report:

```
tripwire --check --email-report --email-report-level 1
```

Sample Policy Rule (twpol.txt)

```
(rulename = "Invariant Directories", severity = $(SIG_MED)) {
    / -> $(SEC_INVARIANT) (recurse = 0);
    /home -> $(SEC_INVARIANT) (recurse = 0);
    /etc -> $(SEC_INVARIANT) (recurse = 0);
}
```

- **Invariant Directories** should never change ownership or permissions.

📡 Network File System (NFS)

Server Setup

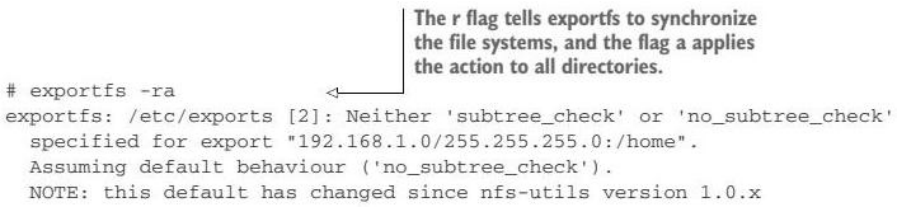
1. Install the server package
    - Ubuntu: apt install nfs-kernel-server
    - CentOS: yum install nfs-utils
2. Edit /etc/exports (example for a single client 192.168.1.11):

```
/home 192.168.1.11(rw,sync)
```

- rw – read/write, sync – write-through.
- Add no_root_squash **only if you trust remote root.

3. Apply changes:

```
exportfs -ra
# Output shows subtree checking status
```



The exportfs -ra command reloads all export definitions.

4. Open firewall (CentOS example):

```
firewall-cmd --add-service=nfs
firewall-cmd --reload
systemctl start nfs-server
```

Client Mount

```
mkdir -p /nfs/home
mount 192.168.1.23:/home /nfs/home
```

- -p creates intermediate directories automatically.

Persist mount via /etc/fstab

```
192.168.1.23:/home  /nfs/home  nfs  defaults  0 0
```

Security Considerations

- Use **least-privilege** exports (ro when possible).
- Restrict client IPs; block unwanted hosts with ufw or firewall-cmd.
- Encrypt traffic (VPN, IPSec, or SSH tunneling) when crossing untrusted networks.

🗂️ Samba – Windows File Sharing

Server Preparation

```
# Ubuntu
apt install samba smbclient

# CentOS
yum install samba samba-client
```

Create Samba User

```
adduser sambauser
smbpasswd -a sambauser
```

Define Share (/etc/samba/smb.conf)

```
[sharehome]
   path = /samba/sharehome
   writable = yes
```

- Set directory permissions (chmod 777 /samba/sharehome) for testing.

Start Service

```
systemctl start smbd   # Ubuntu
systemctl enable smbd
# On CentOS the service name is `smb`
systemctl start smb
```

Test Configuration

```
testparm   # validates smb.conf
smbclient //localhost/sharehome -U sambauser
```

Windows Access

Enter \\192.168.1.23\sharehome in Explorer; authenticate with the Samba credentials.

🔗 Symbolic vs. Hard Links

Feature	Symbolic Link (ln -s)	Hard Link (ln)
Points to	Pathname of target	Direct inode reference
Breaks if target is moved/deleted	**Yes** (dangling)	**No** (still valid)

Can span filesystems	Yes	No
Shows as lrwxrwxrwx	✓	✗
Shares inode number	✗	✓

Create a desktop shortcut to an NFS share

```
ln -s /nfs/home/ /home/username/Desktop/
```

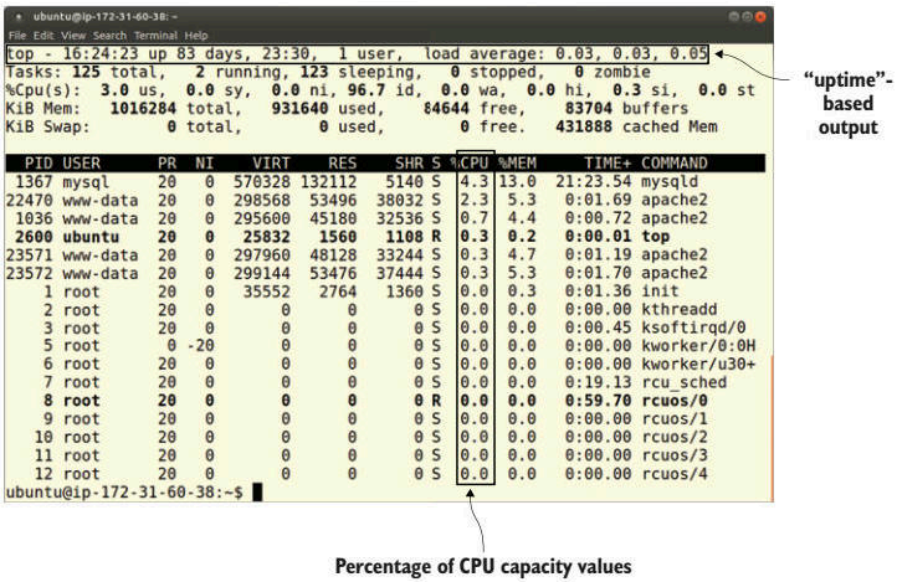
System Monitoring & Performance

CPU

- **Load average** (uptime): three values → 1-, 5-, 15-minute averages.
- Full utilization on an *N*-core CPU is N.

```
uptime
# Example output: load average: 0.12, 0.18, 0.27
```

- **top** shows per-process %CPU.



The top display highlights CPU usage, memory consumption, and process IDs.

Adjust priority with nice / renice

```
nice --15 /var/scripts/mybackup.sh      # lower priority (more “nice”)
renice 15 -p 2145                       # increase niceness of PID 2145
```

Memory

- free -h reports total, used, free, buffers/cache, and swap.

Column	Meaning
Mem:	Physical RAM

Swap:	Virtual swap space
available	Estimate of memory usable without swapping

Storage

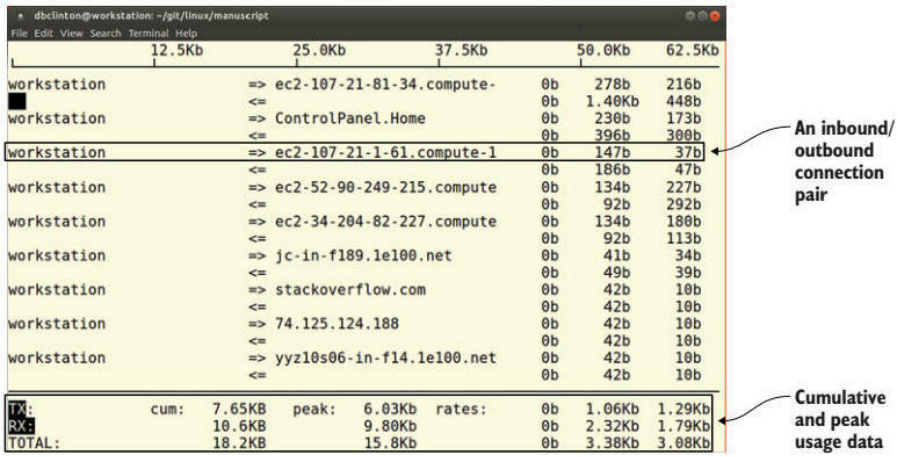
- Disk usage: df -h
- Inode usage: df -i

Find directories with most files

```
cd /
find . -xdev -type f | cut -d "/" -f2 | sort | uniq -c | sort -n
```

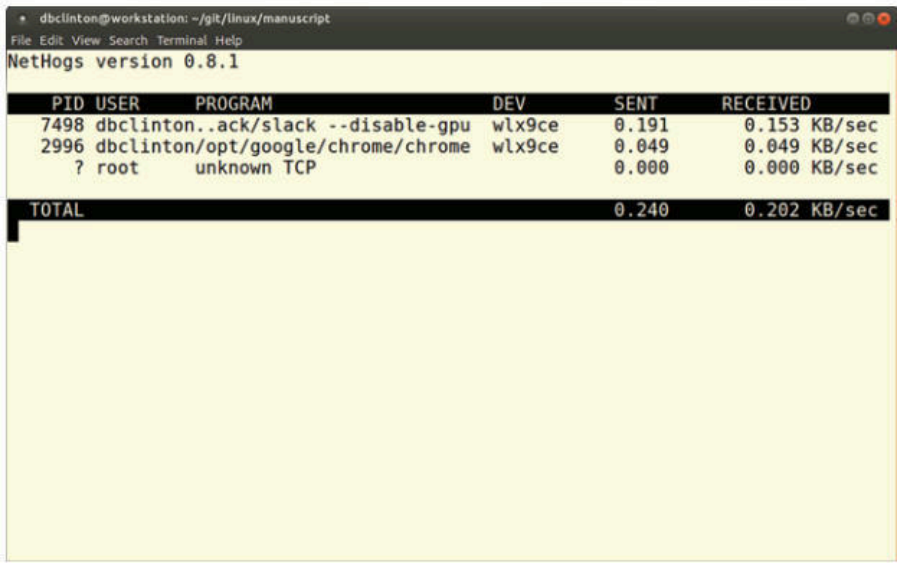
Network

- **iftop -i eth0** – live bandwidth per remote host.



iftop lists connections with their transmit/receive rates.

- **nethogs eth0** – shows bandwidth per PID.



Processes like Chrome and Slack are identified with their traffic.

- Traffic shaping with tc

```
# Add 100 ms delay to all traffic on eth0
tc qdisc add dev eth0 root netem delay 100ms

# Remove the rule
tc qdisc del dev eth0 root
```

Comprehensive Monitoring (nmon)

```
nmon -f -s 30 -c 120   # record every 30 s, 120 samples
```

- Convert to HTML with nmonchart for web-based dashboards.

📋 Command-Line Review (Cheat Sheet)

Command	Purpose
cat /var/log/auth.log | grep -B1 -A1 failure	Show failure lines with context
cat /var/log/mysql/error.log | awk '$3 ~ /\[Warning\]/' | wc -l	Count warning entries
sed "s/^ *[0-9]* //g" numbers.txt	Strip leading line numbers
tripwire --init	Initialize Tripwire database
twadmin --create-cfgfile ...	Build encrypted Tripwire config
exportfs -ra	Reload NFS export table
mount 192.168.1.23:/home /nfs/home	Mount NFS share
systemctl start smbd	Start Samba daemon (Ubuntu)
ln -s /nfs/home/ ~/Desktop/nfs_home	Create symbolic link on desktop
nice --15 script.sh	Run script with low priority
renice 10 -p 1234	Increase niceness of PID 1234
iftop -i eth0	Live bandwidth monitor
nethogs eth0	Bandwidth per process
tc qdisc add dev eth0 root netem delay 100ms	Add network delay
nmon -f -s 30 -c 120	Record system stats for later analysis

🧪 Test Yourself

1. Which command shows the five most recent journal entries?
   - c journalctl -n 5
2. How do you instruct grep to include surrounding lines?
   - c cat /var/log/auth.log | grep -B 1 -A 1 failure
3. Which sed expression removes leading line numbers?
   - a sed "s/^ *[0-9]* //g"
4. What argument prepares the Tripwire database?
   - b tripwire --init
5. Which tc command adds a 100 ms delay?
   - c tc qdisc add dev eth0 root netem delay 100ms

(Answer key: 1-c, 2-c, 3-a, 4-b, 5-c) ## 📡 Requesting a Dynamic IP Address

> **DHCP (Dynamic Host Configuration Protocol)** automatically assigns an IP address, netmask, gateway, and DNS information to a network interface.

```
# dhclient enp0s3
Listening on LPF/enp0s3/08:00:27:9c:1d:67
Sending on LPF/enp0s3/08:00:27:9c:1d:67
DHCPDISCOVER on enp0s3 to 255.255.255.255 port 67 interval 3 (xid=0xf8aa3055)
DHCPREQUEST of 192.168.1.23 on enp0s3 to 255.255.255.255 port 67 (xid=0x5530aaf8)
DHCPOFFER of 192.168.1.23 from 192.168.1.1
DHCPACK of 192.168.1.23 from 192.168.1.1
RTNETLINK answers: File exists
bound to 192.168.1.23 -- renewal in 34443 seconds.
```

- The DHCP server in the example is **192.168.1.1**.
- Lease renewal occurs automatically before the timeout expires.

🏠 Configuring a Static IP Address

1️⃣ One-off static configuration (does **not** survive reboot)

```
# ip addr add 192.168.1.10/24 dev eth0
```

2️⃣ Persistent static configuration

Distribution	Configuration File	Example Section
Ubuntu	/etc/network/interfaces	text\nauto enp0s3\niface enp0s3 inet static\n address 192.168.1.10\n netmask 255.255.255.0\n gateway 192.168.1.1\n
CentOS	/etc/sysconfig/network-scripts/ifcfg-enp0s3	text\nBOOTPROTO=none\nIPADDR=10.0.2

- After editing, restart networking: systemctl restart networking (Ubuntu) or bring the interface up: ip link set dev enp0s3 up.

🧭 DNS Service

> **DNS (Domain Name System)** translates human-readable hostnames (e.g., wikipedia.org) into numeric IP addresses required for routing traffic.

How DNS Works

1. **Local cache** – The OS checks /etc/hosts and a local resolver cache.
2. **Recursive query** – If not cached, the request is sent to a configured **public DNS server** (e.g., Google 8.8.8.8).
3. The DNS server replies with the requested IP address.

Fixing DNS Problems

```
# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=60 time=10.3 ms
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss
```

- If ping by **IP** works but ping by hostname fails, the issue is DNS.

Adding DNS servers

- **Ubuntu** (/etc/network/interfaces)

```
dns-nameserver 8.8.8.8
dns-nameserver 8.8.4.4
```

- **CentOS** (/etc/sysconfig/network-scripts/ifcfg-enp0s3)

```
DNS1=8.8.8.8
DNS2=8.8.4.4
```

📍 Troubleshooting Connectivity

Traceroute – “network drain snake”

```
# traceroute google.com
traceroute to google.com (172.217.0.238), 30 hops max, 60 byte packets
 1  192.168.1.1 (192.168.1.1)  21.173 ms  21.733 ms  23.081 ms
 2  173.206.64.1 (dsl-173-206-64-1.tor.primus.ca)  25.550 ms  27.360 ms  27.865 ms
 …
 7  172.217.0.238 (yyz10s03-in-f14.1e100.net)  12.364 ms  *  6.315 ms
```

- Asterisks (*) indicate lost packets; a complete stop before the final hop suggests a routing or firewall issue.

🛠️ Inbound Connectivity

1️⃣ Scanning local listening sockets with **netstat**

```
# netstat -l | grep http
tcp6       0      0 [::]:http               [::]:*                  LISTEN
tcp6       0      0 [::]:https              [::]:*                  LISTEN
```

- **Network socket** = *IP address + port* (e.g., 192.168.1.23:80).

Interface	RX-OK	RX-ERR	RX-DRP	TX-OK	TX-ERR
enp1s0	0	0	0	0	0
wlx9cefd5fe6a19	1001876	0	0	623247	0

- RX-OK/TX-OK = error-free packets; RX-DRP/TX-DRP = dropped packets.

2️⃣ Scanning remote services with **netcat** (nc)

```
# nc -z -v bootstrap-it.com 443 80
Connection to bootstrap-it.com 443 port [tcp/https] succeeded!
Connection to bootstrap-it.com 80 port [tcp/http] succeeded!
```

- Failure example:

```
# nc -z -v bootstrap-it.com 80
nc: connect to bootstrap-it.com port 80 (tcp) failed: Connection timed out
```

3️⃣ Alternative: nmap

```
# nmap -sT -p80 bootstrap-it.com

PORT   STATE SERVICE
80/tcp open  http
```

- Scan a range (1-1023) to discover unexpected open ports.

⚙️ Managing Peripherals with Kernel Modules

> **Kernel modules** (.ko files) extend the kernel's functionality without rebooting.

- Modules live under /lib/modules/$(uname -r)/.
- List loaded modules: lsmod.
- Find a specific module (e.g., ath9k for an Atheros Wi-Fi card):

```
$ find /lib/modules/$(uname -r) -type f -name 'ath9k*'
/lib/modules/4.4.0-104-generic/kernel/drivers/net/wireless/ath/ath9k/ath9k.ko
```

- Load it manually:

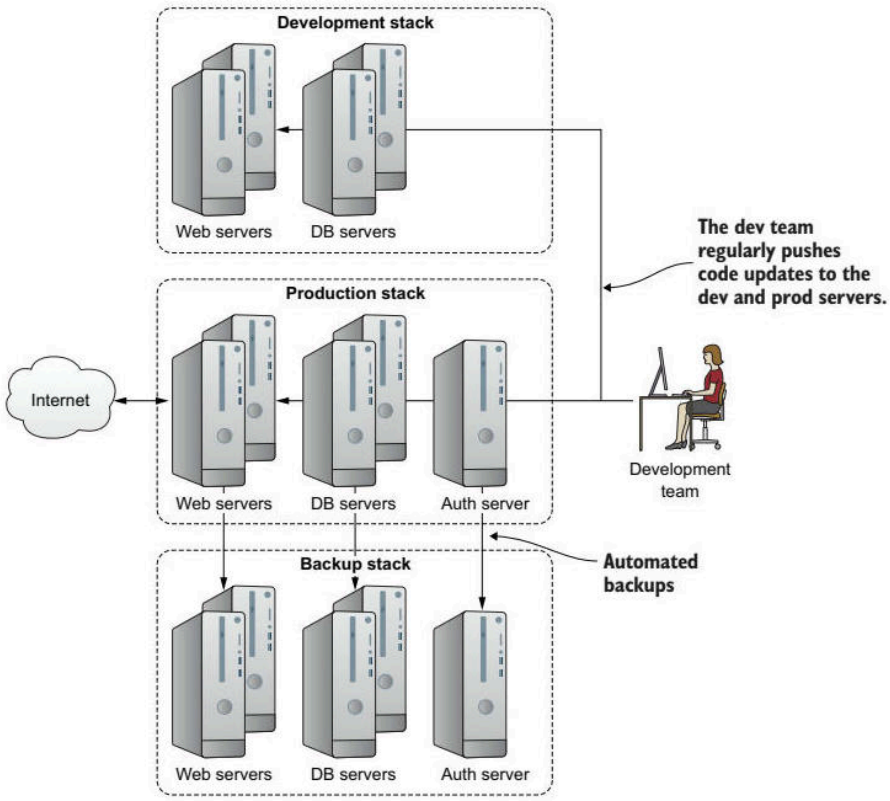
```
# modprobe ath9k
```

- Unload and reload a misbehaving module (e.g., webcam driver uvcvideo):

```
# rmmod uvcvideo
# modprobe uvcvideo
```

📦 DevOps Tools – Ansible Overview

Ansible automates multi-tier server provisioning using **SSH** (no agents required).



The diagram shows Development, Production, and Backup stacks. Ansible can orchestrate code pushes, configuration changes, and automated backups across all three environments.

- **Key benefits**: idempotent playbooks, simple YAML syntax, built-in inventory, and seamless integration with version control.

📦 Ansible Installation & Setup

1. **Install Python (2.7 or 3.x)** on the control node and all target hosts.
2. **Add the Ansible repository** (Ubuntu example) and install:

```
# apt install software-properties-common
# add-apt-repository ppa:ansible/ansible
# apt update
# apt install ansible
```

3. **Create password-less SSH keys** and copy them to each host:

```
$ ssh-keygen                # accept defaults
$ ssh-copy-id -i ~/.ssh/id_rsa.pub ubuntu@10.0.3.142
```

4. **Define hosts** in /etc/ansible/hosts:

```
[webservers]
10.0.3.45
192.168.2.78

[databases]
database1.mydomain.com
```

5. **Test connectivity**:

```
# ansible all -m ping
10.0.3.45 | SUCCESS => {"changed":false,"ping":"pong"}
```

📦 Writing a Simple Playbook

```
---                                  # playbook header
- hosts: webservers
  become: true                       # enable sudo
  tasks:
    - name: install Apache
      apt:
        name: apache2
        state: latest
        update_cache: yes

    - name: copy index.html
      copy:
        src: /home/ubuntu/project/index.html
        dest: /var/www/html/index2.html
      notify: Restart Apache

    - name: ensure Apache is running
      service:
        name: apache2
        state: started

  handlers:
    - name: Restart Apache
      service:
        name: apache2
        state: restarted
```

- **Execution**: ansible-playbook site.yml

```
$ ansible-playbook site.yml
SUDO password:
PLAY ***************************************************
TASK [setup] *******************************************
ok: [10.0.3.96]
TASK [ensure apache is at the latest version] ***
changed: [10.0.3.96]
TASK [copy an index.html file to the root directory] ****
changed: [10.0.3.96]
TASK [ensure apache is running] ************************
ok: [10.0.3.96]
RUNNING HANDLER [restart apache] ***********************
changed: [10.0.3.96]
PLAY RECAP *********************************************
10.0.3.96 : ok=5 changed=3 unreachable=0 failed=0
```

A brief summary of the task's purpose

The ok message tells you that a task has successfully completed.

A summary of the results of running the playbook

The output shows each task's status; three changes were made and all hosts reported success.

📦 Managing Passwords with Ansible Vault

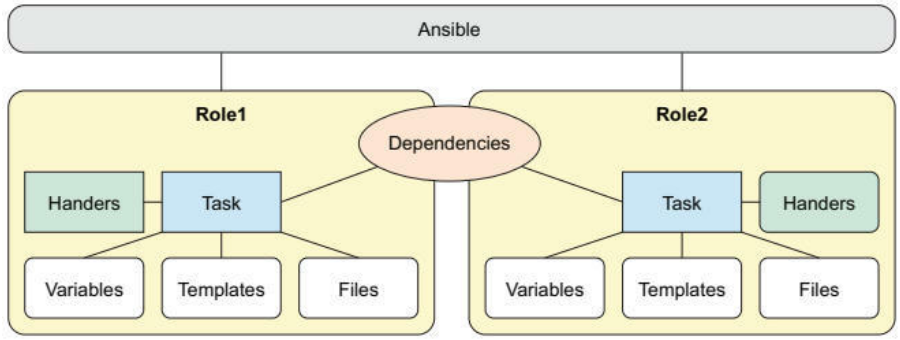
```
$ export EDITOR=nano
$ ansible-vault create mypasswordfile
```

```
New Vault password:
Confirm New Vault password:
```

- Use --ask-vault-pass (or --vault-password-file) when running a playbook that references encrypted variables.

📦 Role-Based Playbooks

Roles encapsulate **tasks**, **handlers**, **variables**, **templates**, and **files** for a single component (e.g., a web app).



Each role is a self-contained directory tree; dependencies can be shared between roles.

- Create a role skeleton:

```
$ mkdir -p ~/ansible/roles
$ cd ~/ansible/roles
$ ansible-galaxy init web-app
```

- The generated tree includes tasks/main.yml, handlers/main.yml, defaults/main.yml, etc., which can be referenced from a top-level playbook.

🔐 Security Best Practices

Area	Recommended Action
Port Scanning	Periodically run nmap or netcat to audit open ports.
Kernel Modules	Load only needed modules; unload unused ones (rmmod).
DNS	Use trusted public resolvers (e.g., Google 8.8.8.8).
Ansible	Store secrets with **Ansible Vault**; prefer SSH key authentication.
CUPS	Restrict /etc/cups/cupsd.conf to required networks; disable remote GUI if not needed.

📋 Command-Line Review

Command	Purpose
ip a	List active network interfaces and IP addresses.
dhclient <iface>	Request a DHCP lease for <iface>.
ip addr add <IP>/<mask> dev <iface>	Assign a temporary static address.
systemctl restart networking	Apply changes made in /etc/network/interfaces (Ubuntu).

` netstat -l	grep http `	
nc -z -v <host> <port>	Test if a remote port is open.
nmap -sT -p1-1023 <host>	Scan well-known ports on a host.
lsmod	List currently loaded kernel modules.
modprobe <module>	Load a kernel module.
ansible <group> -m ping	Verify SSH connectivity to a host group.
ansible-playbook site.yml	Execute a playbook.
ansible-vault create <file>	Create an encrypted vault file.

❓ Test Yourself

1. **Which command requests a dynamic IP address?**

   dhclient enp0s3

2. **How do you make a static address survive a reboot on Ubuntu?**

   Edit /etc/network/interfaces and change dhcp to static with address, netmask, and gateway lines.

3. **What file holds DNS server definitions on a CentOS host?**

   DNS1 and DNS2 entries in /etc/sysconfig/network-scripts/ifcfg-<iface>.

4. **Which tool can you use to verify that a remote web service is listening on port 80?**

   nc -z -v <host> 80 or nmap -sT -p80 <host>.

5. **How do you reload networking after editing /etc/network/interfaces on Ubuntu?**

   systemctl restart networking.