🚀 Linux for DevOps Engineers — Premium Master Documentation

> Advanced Linux Notes for DevOps, Cloud, SRE, Platform Engineering, Containers, Kubernetes, and Production Infrastructure.

🔷 Linux Introduction

📌 What is Linux?

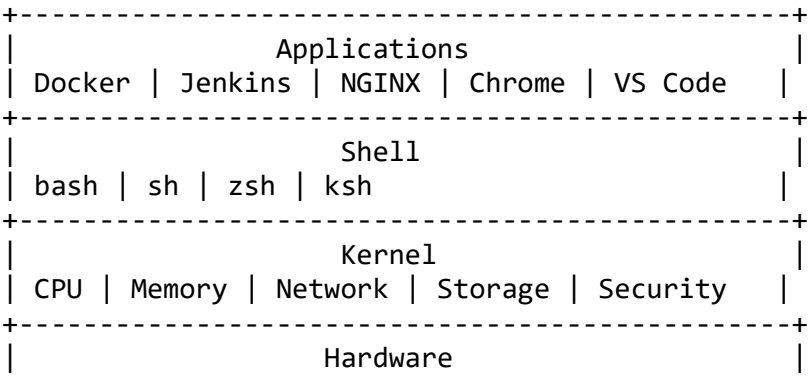
Linux is an open-source operating system kernel created by Linus Torvalds in 1991.

Linux powers:

- Cloud Infrastructure
- AWS, Azure, GCP
- Kubernetes clusters
- Docker containers
- DevOps platforms
- CI/CD servers
- Networking devices
- Supercomputers
- Android devices

Linux is the backbone of modern infrastructure engineering.

🔷 Linux Architecture



```
| RAM | CPU | SSD | NIC | GPU                    |
+-----------------------------------------------+
```

Applications Layer

Applications are software programs users interact with.

Examples:

- Docker
- Jenkins
- Kubernetes CLI
- Terraform
- NGINX
- Apache
- Git

Applications cannot directly access hardware. They communicate with the Linux kernel through system calls.

Shell Layer

Shell acts as a communication bridge between:

- User
- Kernel

It interprets commands and sends requests to kernel.

Types of Shells

Shell	Description
sh	Bourne Shell
bash	Bourne Again Shell
zsh	Z Shell
ksh	Korn Shell

📌 Linux Prompt Symbols

Symbol	Meaning
$	Normal User
#	Root User

Example:

```
$
```

Root User:

```
#
```

◇ Kernel Layer

Kernel is the heart of Linux.

Responsibilities:

- Process management
- Memory management
- Device management
- File system management
- Security management
- Network management

🧠 Internal Working of Linux Kernel

When user runs:

```
cat file.txt
```

Flow:

```
User
 ↓
Shell
 ↓
System Call
 ↓
Kernel
 ↓
```

```
Disk Access
 ↓
Kernel
 ↓
Shell
 ↓
Output
```

Kernel communicates directly with hardware drivers.

◇ Hardware Layer

Hardware components include:

- CPU
- RAM
- SSD/HDD
- NIC
- GPU
- Motherboard

Kernel controls hardware resources.

🚀 Linux Boot Process

📌 Boot Sequence

```
BIOS/UEFI
   ↓
GRUB Bootloader
   ↓
Linux Kernel
   ↓
systemd/init
   ↓
Services Start
   ↓
User Login
```

Bootloader

Bootloader loads operating system into memory.

Most common bootloader:

- GRUB

Full Form:

- Grand Unified Bootloader

Remote Access Protocols

RDP

RDP = Remote Desktop Protocol

Used for:

- GUI remote access
- Windows servers

Default Port:

3389

SSH

SSH = Secure Shell

Used for:

- Remote Linux access
- Server management
- Automation
- Secure communication

Default Port:

22

SSH Internal Working

SSH uses:

- Encryption
- Authentication
- Key exchange

Protocols:

- RSA
- ED25519
- AES Encryption

SSH Commands

SSH Login

```
ssh user@server-ip
```

Generate SSH Key

```
ssh-keygen
```

Creates:

- Public key
- Private key

Connect Using Key

```
ssh -i key.pem ubuntu@10.0.0.10
```

SSH Security Best Practices

❌ Never:

- Share private key
- Use root login directly

- Use weak passwords

☑ Always:

- Use SSH keys
- Disable password authentication
- Rotate keys
- Use MFA where possible
- Restrict IP access

🚀 Linux File System

Everything starts from:

/

Called:

- Root filesystem

◇ Important Linux Directories

Directory	Purpose
/	Root directory
/home	User home directories
/etc	Configuration files
/var	Logs and variable data
/bin	Essential binaries
/usr	Applications
/tmp	Temporary files
/dev	Device files

🧠 Real Production Examples

NGINX logs:

`/var/log/nginx/`

SSH configuration:

`/etc/ssh/sshd_config`

🚀 Linux Commands

◇ ls Command

List files and directories.

```
ls
```

Detailed Listing

```
ls -l
```

Example Output:

```
drwxr-xr-x 2 root root 4096 Jan 1 test
```

Breakdown:

Part	Meaning
d	Directory
rwxr-xr-x	Permissions
root	Owner
root	Group

Hidden Files

```
ls -a
```

Hidden files begin with:

`.`

Example:

`.bashrc`

Detailed + Hidden

```
ls -al
```

◇ pwd Command

Present Working Directory.

```
pwd
```

◇ mkdir Command

Create directory.

```
mkdir project
```

◇ cd Command

Change directory.

```
cd /etc
```

Parent directory:

```
cd ..
```

◇ touch Command

Create empty file.

```
touch app.log
```

◇ rm Command

Remove files.

```
rm file.txt
```

Recursive delete:

```
rm -r folder
```

Dangerous Command Warning

```
rm -rf /
```

⚠ Deletes entire filesystem.

Never run this command.

---

rmdir Command

Remove empty directory.

```
rmdir folder
```

Echo Command

Print text:

```
echo "hello"
```

Save output:

```
echo "hello" > file.txt
```

Append output:

```
echo "hello" >> file.txt
```

🚀 File Reading Commands

◇ cat Command

Display complete file.

```
cat file.txt
```

◇ head Command

Display first 10 lines.

```
head file.txt
```

First 5 lines:

```
head -n 5 file.txt
```

◇ tail Command

Display last 10 lines.

```
tail file.txt
```

Live monitoring:

```
tail -f app.log
```

🧠 Real DevOps Example

Monitor NGINX logs:

```
tail -f /var/log/nginx/access.log
```

◇ less Command

Read large files efficiently.

```
less app.log
```

Advantages:

- Search support
- Scrolling
- Efficient memory usage

🚀 File Operations

◇ cp Command

Copy files.

```
cp source.txt destination.txt
```

Recursive copy:

```
cp -r folder1 folder2
```

◇ mv Command

Move or rename files.

```
mv old.txt new.txt
```

◇ wc Command

Word count.

```
wc file.txt
```

Shows:

- Lines

- Words
- Bytes

🚀 Links in Linux

🔷 Soft Link

Shortcut/reference file.

```
ln -s original.txt shortcut.txt
```

If original deleted: ❌ Link breaks

🔷 Hard Link

Shares same inode.

```
ln file1 file2
```

If original deleted: ☑ Still works

🧠 Inode Concept

Inode stores:

- Metadata
- Permissions
- Ownership
- Disk block locations

🚀 Process Management

◇ ps Command

Show processes.

```
ps aux
```

◇ top Command

Real-time process monitoring.

```
top
```

Quit:

```
q
```

◇ kill Command

Kill process.

```
kill PID
```

Force kill:

```
kill -9 PID
```

◇ Linux Process States

State	Meaning
Running	Currently executing
Sleeping	Waiting for resource
Zombie	Dead but entry exists
Stopped	Paused

🧠 Zombie Process

Zombie process:

- Finished execution
- Parent process did not collect status

Find zombie process:

```
ps aux | grep Z
```

🚀 Memory Commands

◇ free Command

Show RAM information.

```
free -h
```

◇ vmstat Command

Virtual memory statistics.

```
vmstat
```

Shows:

- CPU
- Memory
- IO
- Swap

🚀 User Management

◇ Create User

```
useradd -m devops
```

◇ Set Password

```
passwd devops
```

◇ Switch User

```
su - devops
```

◇ Delete User

```
userdel -r devops
```

🚀 Group Management

Create group:

```
groupadd developers
```

Add user:

```
gpasswd -a devops developers
```

🚀 Linux Permissions

```
-rwxr-xr--
```

Breakdown:

```
Type User Group Others
```

◇ Permission Values

Number	Permission

Number	Permission
0	—
1	–x
2	-w-
3	-wx
4	r–
5	r-x
6	rw-
7	rwx

◇ chmod Command

```
chmod 755 script.sh
```

Meaning:

- Owner → rwx
- Group → r-x
- Others → r-x

🛑 Security Warning

Avoid:

```
chmod 777
```

Reason:

- Huge security risk
- Anyone can modify files

◇ chown Command

Change ownership.

```
chown user file.txt
```

◇ chgrp Command

Change group.

```
chgrp devops file.txt
```

🚀 Compression Commands

◇ zip Command

```
zip -r backup.zip folder/
```

◇ unzip Command

```
unzip backup.zip
```

◇ tar Command

Create archive:

```
tar -cvzf backup.tar.gz folder/
```

Extract archive:

```
tar -xvzf backup.tar.gz
```

🚀 Networking Commands

◇ ping Command

Check connectivity.

```
ping google.com
```

5 packets:

```
ping -c 5 google.com
```

◇ ifconfig Command

Show network interface information.

Modern replacement:

```
ip addr
```

◇ netstat Command

Show network connections.

```
netstat -tulpn
```

◇ ss Command

Modern replacement for netstat.

```
ss -tulpn
```

◇ traceroute Command

Trace network path.

```
traceroute google.com
```

◇ nslookup Command

DNS lookup.

```
nslookup google.com
```

◇ dig Command

Advanced DNS lookup.

```
dig google.com
```

◇ curl Command

Call APIs.

```
curl https://api.github.com
```

Formatted output:

```
curl api | jq
```

◇ wget Command

Download files.

```
wget https://example.com/file.zip
```

🚀 Storage Management

◇ lsblk Command

List block devices.

```
lsblk
```

◇ df Command

Show disk usage.

```
df -h
```

◇ mount Command

Attach filesystem.

```
mount /dev/xvdf /mnt/data
```

◇ umount Command

Detach filesystem.

```
umount /mnt/data
```

🚀 LVM (Logical Volume Manager)

🧠 Why LVM?

Traditional partitions: ❌ Hard to resize

LVM: ☑ Dynamic storage management

◇ LVM Architecture

```
Physical Volume → Volume Group → Logical Volume
```

◇ Create Physical Volume

```
pvcreate /dev/xvdf
```

◇ Create Volume Group

```
vgcreate data_vg /dev/xvdf
```

◇ Create Logical Volume

```
lvcreate -L 10G -n data_lv data_vg
```

◇ Format Filesystem

```
mkfs.ext4 /dev/data_vg/data_lv
```

◇ Mount Logical Volume

```
mount /dev/data_vg/data_lv /mnt/data
```

◇ Extend Logical Volume

```
lvextend -L +5G /dev/data_vg/data_lv
```

Resize filesystem:

```
resize2fs /dev/data_vg/data_lv
```

🚀 awk Command

Pattern scanning and processing.

Example:

```
awk '{print $1}' file.txt
```

🚀 sed Command

Stream editor.

Replace text:

```
sed 's/info/log/g' app.log
```

🚀 grep Command

Search text.

```
grep "ERROR" app.log
```

Case insensitive:

```
grep -i error app.log
```

🚀 Production DevOps Scenarios

◇ Scenario 1 — High CPU Usage

Check:

```
top
```

Find high CPU process:

```
ps aux --sort=-%cpu
```

Kill process:

```
kill -9 PID
```

◇ Scenario 2 — Disk Full

Check:

```
df -h
```

Find large files:

```
du -sh /*
```

◇ Scenario 3 — Service Not Running

Check service:

```
systemctl status nginx
```

View logs:

```
journalctl -u nginx
```

🚀 Linux Security Best Practices

☑ Use SSH keys ☑ Disable root login ☑ Use least privilege access ☑ Enable firewalls ☑ Keep systems updated ☑ Rotate credentials ☑ Enable logging and monitoring ☑ Use MFA

🚀 Senior DevOps Engineer Tips

Use aliases:

```
alias ll='ls -al'
```

Reverse search:

```
Ctrl + r
```

Live monitoring:

```
watch -n 2 df -h
```

Use tmux/screen for long-running sessions.

🚀 Common Interview Questions

❓ What is Linux Kernel?

Kernel is the core component of Linux responsible for communication between hardware and software.

❓ Difference Between Hard Link and Soft Link?

Hard Link	Soft Link
Shares inode	Separate inode
Works after source deletion	Breaks after source deletion

❓ Difference Between grep, sed, awk?

Command	Purpose
grep	Search
sed	Stream editing
awk	Structured data processing

🚀 Linux Firewall Management

◇ What is Firewall?

Firewall is a security layer that controls:

- Incoming traffic
- Outgoing traffic
- Network access rules

Purpose:

- Protect server from unauthorized access
- Allow only required ports/services
- Prevent attacks

◇ Types of Linux Firewalls

Firewall	Linux Distribution
iptables	Traditional Linux firewall
firewalld	RHEL/CentOS/Fedora
UFW	Ubuntu
nftables	Modern Linux firewall

🚀 UFW (Uncomplicated Firewall)

Mostly used in Ubuntu.

◇ Install UFW

```
sudo apt install ufw -y
```

◇ Enable Firewall

```
sudo ufw enable
```

◇ Check Firewall Status

```
sudo ufw status
```

◇ Allow SSH

```
sudo ufw allow 22
```

◇ Allow HTTP and HTTPS

```
sudo ufw allow 80
sudo ufw allow 443
```

◇ Deny Specific Port

```
sudo ufw deny 3306
```

◇ Delete Firewall Rule

```
sudo ufw delete allow 80
```

Important Security Warning

Before enabling firewall:

☑ Always allow SSH first.

Otherwise: ❌ You may lock yourself out of server.

Production Firewall Architecture

```
Internet
   ↓
Cloud Security Group
   ↓
Linux Firewall (UFW/iptables)
   ↓
NGINX / Application
```

---

iptables Firewall

iptables is low-level Linux firewall.

◇ Allow SSH Using iptables

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

◇ Allow HTTP

```
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

◇ Block IP Address

```
iptables -A INPUT -s 192.168.1.10 -j DROP
```

🧠 Why Firewalls Are Important?

Without firewall:

- Open ports exposed
- Server vulnerable to attacks
- Brute force possible
- Malware access possible

Firewall acts as first security checkpoint.

🚀 Multiple Websites on One Linux Instance

◇ Why Run Multiple Websites?

Companies run multiple applications on one server to:

- Reduce infrastructure cost
- Optimize resources
- Simplify management
- Share load balancer and reverse proxy

◇ Methods to Run Multiple Websites

Method	Description
Port-Based Hosting	Different ports
Name-Based Hosting	Different domain names
IP-Based Hosting	Multiple IP addresses
Reverse Proxy	NGINX/Apache routing
Containers	Docker-based isolation

🚀 Port-Based Hosting

Example:

Website	Port
app1	80
app2	8080
app3	3000

◇ Example Using Python Server

Website 1:

```
python3 -m http.server 80
```

Website 2:

```
python3 -m http.server 8080
```

Access:

```
http://IP
http://IP:8080
```

🚀 Name-Based Virtual Hosting (NGINX)

Most common production approach.

🧠 Architecture

```
Internet
   ↓
NGINX Reverse Proxy
   ↓
---------------------------------
|            |                  |
app1.com   app2.com       app3.com
```

◇ Install NGINX

```
sudo apt install nginx -y
```

◇ Create Website Directories

```
sudo mkdir -p /var/www/app1
sudo mkdir -p /var/www/app2
```

◇ Create Sample Pages

```
echo "App1 Website" | sudo tee /var/www/app1/index.html
```

```
echo "App2 Website" | sudo tee /var/www/app2/index.html
```

◇ Create Virtual Host Config

```
sudo vim /etc/nginx/sites-available/app1
```

Configuration:

```
server {
    listen 80;
    server_name app1.com;

    root /var/www/app1;
    index index.html;
}
```

◇ Enable Site

```
sudo ln -s /etc/nginx/sites-available/app1 /etc/nginx/sites-enabled/
```

◇ Test NGINX Config

```
sudo nginx -t
```

◇ Restart NGINX

```
sudo systemctl restart nginx
```

🚀 Alias in Linux

◇ What is Alias?

Alias creates shortcut for long commands.

Purpose:

- Faster command execution
- Reduce typing
- Improve productivity

◇ Temporary Alias

```
alias ll='ls -al'
```

Works only for current terminal session.

◇ Permanent Alias

Add inside:

`~/.bashrc`

Example:

```
alias k='kubectl'
alias d='docker'
alias ll='ls -al'
```

Apply changes:

```
source ~/.bashrc
```

🧠 Real DevOps Aliases

```
alias k='kubectl'
alias kgp='kubectl get pods'
alias dps='docker ps'
alias tf='terraform'
```

🚀 How to Secure Linux Instance

◇ What is Secure Server Hardening?

Hardening means:

- Reducing attack surface
- Removing vulnerabilities
- Restricting unauthorized access

Goal:

- Protect infrastructure
- Protect data
- Protect applications

🚀 Linux Server Hardening Checklist

Security Practice	Purpose
Disable root login	Prevent direct attacks
Use SSH keys	Strong authentication
Enable firewall	Restrict traffic
Update packages	Patch vulnerabilities
Disable unused services	Reduce attack surface
Use least privilege	Prevent misuse
Monitor logs	Detect attacks
Install Fail2Ban	Prevent brute force
Enable MFA	Extra security

◇ Disable Root SSH Login

Edit SSH config:

```
sudo vim /etc/ssh/sshd_config
```

Change:

```
PermitRootLogin no
```

Restart SSH:

```
sudo systemctl restart ssh
```

◇ Disable Password Authentication

Inside:

```
PasswordAuthentication no
```

Use only SSH keys.

◇ Keep Server Updated

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

◇ Install Fail2Ban

Protect against brute-force attacks.

```
sudo apt install fail2ban -y
```

◇ Check Open Ports

```
ss -tulpn
```

Close unnecessary ports.

Monitor Logs

Authentication logs:

`/var/log/auth.log`

NGINX logs:

`/var/log/nginx/access.log`

Secure File Permissions

Correct SSH key permission:

```
chmod 400 key.pem
```

Use Non-Root User

Create secure admin user:

```
useradd -m devops
passwd devops
usermod -aG sudo devops
```

Production Security Architecture

```
Internet
   ↓
Cloud Security Group
   ↓
Linux Firewall
   ↓
NGINX Reverse Proxy
   ↓
Application Layer
   ↓
Database Layer
```

Common Security Mistakes

❌ Using root login ❌ Opening all ports ❌ Using weak passwords ❌ chmod 777 ❌ Not updating packages ❌ Running everything as root

---

Senior DevOps Engineer Tips

Use SSH keys instead of passwords Restrict access using firewall Monitor logs continuously Use reverse proxy for applications Automate backups Enable monitoring tools Use Infrastructure as Code Rotate credentials regularly

---

Cron Jobs in Linux

What is Cron?

Cron is a Linux job scheduler used to:

- Run tasks automatically
- Execute scripts periodically
- Automate backups
- Run monitoring jobs
- Schedule maintenance tasks

Cron service runs in background continuously.

Daemon name:

`crond`

Why Cron Jobs Are Important?

Used heavily in production for:

- Backup automation

- Log cleanup
- Monitoring scripts
- Database dumps
- SSL renewal
- Security scanning
- Scheduled deployments

◇ Cron Job Architecture

```
User Crontab
     ↓
Cron Daemon
     ↓
Scheduled Time Match
     ↓
Execute Command/Script
```

◇ Open Crontab

```
crontab -e
```

◇ View Cron Jobs

```
crontab -l
```

◇ Remove Cron Jobs

```
crontab -r
```

◇ Cron Job Format

```
* * * * * command
- - - - -
| | | | |
| | | | └── Day of Week
```

```
| | | └──── Month
| | └────── Day
| └──────── Hour
└────────── Minute
```

◇ Common Cron Examples

Run every minute:

```
* * * * * /home/ubuntu/script.sh
```

Run every day at 11 AM:

```
0 11 * * * /home/ubuntu/backup.sh
```

Run every Sunday:

```
0 0 * * 0 script.sh
```

🚀 Shell Scripting

◇ What is Shell Scripting?

Shell scripting means writing multiple Linux commands inside one file for automation.

Used for:

- Server automation
- Monitoring
- Deployments
- Backups
- Health checks
- CI/CD automation

◇ Why Shell Scripting is Important in DevOps?

DevOps engineers automate repetitive tasks.

Without scripting: ❌ Manual operations ❌ Human errors ❌ Slow deployments

With scripting: ☑ Automation ☑ Faster operations ☑ Standardization ☑ Reduced human error

---

◇ Create Shell Script

Create file:

```
touch script.sh
```

Edit:

```
vim script.sh
```

◇ Basic Shell Script Example

```
#!/bin/bash

echo "Hello DevOps"
```

◇ Script Breakdown

Part	Meaning
#!/bin/bash	Shebang
echo	Print output

◇ Give Execute Permission

```
chmod +x script.sh
```

◇ Run Script

```
./script.sh
```

OR

```
bash script.sh
```

🚀 Variables in Shell Script

```bash
#!/bin/bash

name="Saumil"

echo "Hello $name"
```

🚀 User Input in Shell Script

```bash
#!/bin/bash

read -p "Enter your name: " name

echo "Welcome $name"
```

🚀 Conditions in Shell Script

```bash
#!/bin/bash

if [ $USER == "ubuntu" ]
then
    echo "Correct User"
else
    echo "Wrong User"
fi
```

🚀 Loops in Shell Script

◇ For Loop

```bash
#!/bin/bash

for i in 1 2 3 4 5

do
```

```
    echo $i

done
```

◇ While Loop

```
#!/bin/bash

count=1

while [ $count -le 5 ]
do
    echo $count
    count=$((count+1))
done
```

🚀 File Finding Commands

◇ find Command

Used to search files and directories.

◇ Find File by Name

```
find / -name file.txt
```

◇ Find All .log Files

```
find /var/log -name "*.log"
```

◇ Find Files Larger Than 1GB

```
find / -size +1G
```

◇ Find Files Modified in Last 7 Days

```
find /var/log -mtime -7
```

◇ Find and Delete Files

```
find /tmp -name "*.tmp" -delete
```

⚠ Use carefully.

🚀 Real Production Shell Script — Website Backup Automation

◇ Requirement

Create tar backup of:

`/var/www/html`

Destination:

`/home/ubuntu`

Run automatically every day at:

`11:00 AM`

◇ Create Backup Script

File:

`vim /home/ubuntu/backup.sh`

Script:

```
#!/bin/bash
```

```
DATE=$(date +%F-%H-%M-%S)

SOURCE="/var/www/html"
DESTINATION="/home/ubuntu"

TAR_FILE="website-backup-$DATE.tar.gz"

mkdir -p $DESTINATION

tar -cvzf $DESTINATION/$TAR_FILE $SOURCE

echo "Backup completed: $TAR_FILE"
```

◇ Script Breakdown

Line	Purpose
date command	Dynamic timestamp
SOURCE	Website path
DESTINATION	Backup location
tar -cvzf	Compress backup
mkdir -p	Create directory safely

◇ Give Execute Permission

```
chmod +x /home/ubuntu/backup.sh
```

◇ Test Script Manually

```
/home/ubuntu/backup.sh
```

Check backup:

```
ls -lh /home/ubuntu
```

🚀 Schedule Backup Using Cron Job

Open crontab:

```
crontab -e
```

Add:

```
0 11 * * * /home/ubuntu/backup.sh >> /home/ubuntu/backup.log 2>&1
```

🧠 Cron Job Breakdown

Part	Meaning
0	Minute
11	Hour
*	Every day
*	Every month
*	Every weekday

🔷 Logging in Cron Job

```
>> /home/ubuntu/backup.log 2>&1
```

Meaning:

- Save normal logs
- Save error logs
- Useful for troubleshooting

🪨 Common Cron Problems

Problem	Solution
Script not running	Give execute permission
Command not found	Use full path
Cron service stopped	Restart cron
Permission denied	Check ownership

◇ Restart Cron Service

Ubuntu:

```
sudo systemctl restart cron
```

Check status:

```
sudo systemctl status cron
```

🚀 Production Backup Best Practices

☑ Store backups on separate disk ☑ Use compressed backups ☑ Rotate old backups ☑ Test restore process ☑ Use monitoring for backup success ☑ Encrypt sensitive backups ☑ Upload backups to cloud storage

---

🧠 Real Enterprise Backup Architecture

```
Production Server
       ↓
Cron Job Scheduler
       ↓
Backup Script
       ↓
Tar Compression
       ↓
Local Backup
       ↓
Cloud Storage / S3
```

🚀 Advanced DevOps Tips

Use rsync for incremental backups Use AWS S3 lifecycle policies Monitor backup failures with Prometheus Send backup alerts using email or Slack Use logrotate for backup logs Store backups on remote servers

---

🚀 Quick Revision Notes

☑ Linux architecture ☑ Boot process ☑ Shell and kernel ☑ SSH ☑ Linux filesystem ☑ Linux commands ☑ Process management ☑ Networking ☑ Permissions ☑ User management ☑ LVM ☑ Firewall management ☑ Multiple website hosting ☑ Name-based virtual hosting ☑ Port-based hosting ☑ Linux hardening ☑ Alias commands ☑ Shell scripting ☑ File finding commands ☑ Cron jobs ☑ Backup automation ☑ Production backup strategy ☑ Security best practices ☑ Troubleshooting ☑ Production scenarios