01 July 2025 13:34

LINUX

��What is Linux?

• Linux is an Operating System (OS) — like Windows or macOS. • But unlike them, Linux is open-source, meaning it's free and its code is publicly available. •

Linux is used in servers, cloud computing, mobile (Android), embedded systems, and DevOps tools.

��Basic Components of Linux OS Component Explanation

Kernel The core part of Linux that talks to hardware (like CPU, RAM, disks). Shell A program that lets you interact with the OS via commands. File System Organizes and stores data. Everything in Linux is treated as a file (even devices).

GUI Optional — Linux can have graphical interfaces like GNOME or KDE, but often used via terminal.

��Linux Directory Structure

Linux starts with root (/) directory. Below it, there are folders with fixed purposes: Directory Purpose

/ Root of the file system /home User folders (like C:\Users)

/bin Essential command binaries (like ls, cp, mv)

/etc System config files

/var Variable files (like logs)

/tmp Temporary files

/usr User programs and libraries

/root Home directory of the root (admin) user

/dev Device files (like USB, hard disks)

/proc System and process info (virtual folder)

��Basic Linux Commands

Command Use

pwd Show current directory

ls List files and folders

cd Change directory

mkdir Make a directory

touch Create an empty file

cp Copy files/folders

Basic info Page 1

mv Move or rename files

rm Delete files

cat Show file contents

clear Clear the terminal

exit Close the terminal or shell

��User and Permissions

Linux is multi-user. Every file and folder has permissions and ownership. Users:

• root: Superuser (admin)

• Normal users: Limited access

Permissions:

•

Each file has 3 types of permissions for:

○ Owner

○ Group

○ Others

Permission types:

• r → read

• w → write

• x → execute

Example:

-rwxr-xr-- 1 user group 1234 Jul 1 file.sh

Breakdown:

•

-rwxr-xr--

○ rwx (owner): full permission

○ r-x (group): read & execute

○ r-- (others): read only

��Package Management (Installing Software) •

Ubuntu/Debian-based:

○ apt-get install <package-name>

•

Red Hat/CentOS:

○ yum install <package-name> or dnf install

• Others: May use pacman, zypper, snap, flatpak

��Processes and System Monitoring Command Description

ps Show running processes

top Live view of system usage (CPU, memory)

kill Stop a process

htop Better version of top (if installed)

��File Compression and Archiving Command Description

tar -cvf Create archive

Basic info Page 2

tar -cvf Create archive

tar -xvf Extract archive

gzip, gunzip Compress and decompress files

��Important Concepts

Concept Meaning

Shell Script A file with commands you can execute like a program (.sh) Cron Job Schedule a task to run automatically Log files Located in /var/log, useful for troubleshooting SSH Remote login to a Linux machine (ssh user@ip)

Sudo Temporarily run commands as root user (sudo <command>) ��C – CREATE Operations in Linux

⚙ How It Helps in DevOps

Linux is the foundation of:

• Cloud servers (AWS EC2, GCP, Azure)

• CI/CD pipelines

• Docker & Kubernetes

• Automation tools like Ansible, Terraform

��What Are CRUD Operations? CRUD stands for:

• C – Create

• R – Read

• U – Update

• D – Delete

In Linux, CRUD operations can be performed on:

• Files

• Directories

• File content

• System records (like user data or processes)

➤ Create a File

touch filename.txt

Example:

touch notes.txt

➤ Create a File with Content

echo "Hello, Linux!" > hello.txt

OR

cat > data.txt

Type your content

Press Ctrl+D to save

➤ Create a Directory (Folder)

mkdir foldername

Example:

Basic info Page 3

��R – READ Operations in Linux

Example:

mkdir projects

��U – UPDATE Operations in Linux

➤ Create Nested Directories

mkdir -p devops/scripts/yaml

➤ View File Content

cat filename.txt

less filename.txt # Scrollable

more filename.txt # Page-wise output

➤ View First or Last Lines

head filename.txt # First 10 lines

tail filename.txt # Last 10 lines

➤ Read a Directory

ls # List files

ls -l # Detailed view ls -a # Show hidden files

➤ Read a Specific Line (Using sed or awk) ��D – DELETE Operations in Linux

sed -n '3p' filename.txt # Show 3rd line

awk 'NR==3' filename.txt # Another method

➤ Append to File

echo "New line" >> file.txt

➤ Edit File (Using Editors)

• nano filename.txt – beginner-friendly terminal editor • vim filename.txt – powerful but advanced editor

➤ Replace Text in File

sed -i 's/oldtext/newtext/g' filename.txt

Example:

sed -i 's/Devops/Linux DevOps/g' notes.txt

➤ Rename File or Folder

mv oldname.txt newname.txt

➤ Move File to Different Location mv file.txt /home/user/docs/

➤ Delete a File

rm filename.txt

➤ Delete a Directory

rm -r foldername

➤ Force Delete

rm -rf foldername

⚠ Basic info Page 4

⚠ Be careful with rm -rf / — it can wipe out the enƟre OS.

�� Bonus: Real-Time DevOps Use Cases of CRUD in LinuxTask Linux Command

Create a config file touch nginx.conf

Read server logs less /var/log/syslog

Update YAML for deployment nano deployment.yaml

Delete temporary files rm -rf /tmp/\*

Move backup mv backup.tar.gz /mnt/backup/

��Sample Exercise to Practice

# Step 1: Create a directory and file

mkdir devops\_practice

cd devops\_practice

touch inventory.txt

# Step 2: Add data

echo "Server1: 192.168.1.1" >> inventory.txt

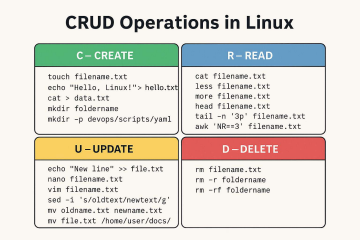
echo "Server2: 192.168.1.2" >> inventory.txt

# Step 3: Read and edit

cat inventory.txt

sed -i 's/192.168.1.2/10.0.0.2/' inventory.txt

# Step 4: Delete the file

��VIM Editor in Linux — Complete Notes with Modes

rm inventory.txt

& Editing Operations

�� Basic info Page 5

��1. NORMAL MODE – The Control Hub

��What is Vim?

• Vim (Vi IMproved) is a modal text editor in Linux.

•

Used for editing config files, shell scripts, YAML, Docker, Terraform, and more —especially in headless servers and DevOps workflows.

• Runs in the terminal and is lightweight, powerful, and scriptable.

��Vim Has 3 Primary Modes

Mode Name Purpose Enter Mode Exit Mode1. Normal Mode Default mode — for navigation,

Open Vim or press Esc Press i, v, or :

deleting, copying

2. Insert Mode Typing text (like in Notepad) Press i, I, a, A, o, OPress Esc3. Command-Line

Save, exit, search, run commands Press : from Normal

Press Enter or

Mode

Mode Esc

��2. INSERT MODE – For Writing or Editing Text

✅Actions in Normal Mode:

Action Key

Move cursor h (left), l (right), j (down), k (up)

Word jump w (next word), b (previous word)

Start of line 0

End of line $

Top of file gg

Bottom of file G

Delete line dd

Copy (yank) line yy

Paste p

Undo u

Redo Ctrl + r

✅Enter Insert Mode:

Key Function

i Insert before cursor

I Insert at start of line

a Insert after cursor

A Insert at end of line

o Open a new line below O Open a new line above

Once in insert mode, you type normally like any text editor.

✅ Basic info Page 6

��3. COMMAND-LINE MODE – For Save, Quit, Search

✅Exit Insert Mode:

• Press Esc to go back to Normal Mode.

✅Enter Command Mode:

• From Normal Mode, press : (colon)

✅Common Commands:

Command Meaning

:w Save file

:q Quit

:wq or ZZ Save and quit

:q! Force quit without saving

:x Save and exit (if changes made)

:set nu Show line numbers

:set nonu Hide line numbers

✏EdiƟng OperaƟons in Vim (DevOps Oriented) ��File & Config Editing:

vim /etc/nginx/nginx.conf

1. Navigate to the line using arrow keys or j/k.

2. Press i to enter insert mode.

3. Make changes.

4. Press Esc, then type :wq to save and exit.

��Searching and Replacing

•

Search for a word:

/nginx

Then press n (next match) or N (previous match)

•

Replace all occurrences in file:

:%s/oldtext/newtext/g

��Copy, Cut, Paste (Yank, Delete, Put) Action Key

Copy (line) yy

Copy (3 lines) 3yy

Delete (cut) line dd

Delete 5 lines 5dd

Paste below p

Paste above P

�� Basic info Page 7

��Line and Block Editing

• Visual Mode (v): Select characters/words

• Visual Line Mode (V): Select entire lines

• Visual Block Mode (Ctrl + v): Select columns

Then you can press y, d, or p to copy, cut, or paste.

��Practical DevOps Examples

Task Vim Command

Edit crontab crontab -e (uses Vim by default)

Change pod name in YAML vim pod.yaml, /name, i, edit

Format Terraform code vim main.tf, use =G to auto-indent

Edit Dockerfile vim Dockerfile

��Linux File & Directory Permissions – Full Guide

Restart NGINX config Edit nginx.conf then sudo systemctl restart nginx

��Sample Practice

vim test.sh

1.

Press i, type:

#!/bin/bash

echo "Hello, Vim World!"

2. Press Esc

3. Save: :wq

4. Make executable: chmod +x test.sh

5. Run it: ./test.sh

��Vim Cheat Sheet Summary

Mode Enter Purpose Exit

Normal Open Vim / Esc Navigation, editing i, :, v

Insert i, a, o, etc. Typing text Esc

Command : Save, quit, search Enter or Esc

✅Why Permissions Matter?

•

Linux is a multi-user OS, and permissions protect files, scripts, and system settings from unauthorized access or modification.

• Common in DevOps, cloud environments, and CI/CD pipelines where secure access is critical.

��Permission Types

Each file or directory in Linux has 3 types of permissions for 3 types of users: ��User Categories:

Symbol User Type

u User (Owner)

Basic info Page 8

u User (Owner)

g Group

o Others (Everyone else)

��Permission Types:

Symbol Permission Meaning

r Read View content

w Write Modify content

x Execute Run file (scripts, binaries) or enter directory

��Example Output from ls -l

-rwxr-xr-- 1 devops team 2048 Jul 1 deploy.sh Breakdown:

Part Meaning

- File type (- = file, d = directory)

rwx Owner: read, write, execute

r-x Group: read, execute

r-- Others: read only

devops Owner username

team Group name

��chmod – Change Permissions

��Numeric (Octal) Permissions Each permission has a numeric value:

Permission Binary Octal

r 100 4

w 010 2

x 001 1

Add values to get final permission:

rwx combo Value

rwx 7

rw- 6

r-- 4

r-x 5

So:

chmod 755 filename

# = rwxr-xr-x

��Symbolic Method:

chmod u+x script.sh # Add execute to user

chmod g-w file.txt # Remove write from group chmod o=r file.txt # Others: read-only

Basic info Page 9

��chown – Change Ownership

chmod o=r file.txt # Others: read-only

��Numeric (Octal) Method:

chmod 644 file.txt # rw-r--r--

chmod 755 script.sh # rwxr-xr-x

chmod 700 secrets.txt # rwx------ (only owner can access)

chown user:group file

Example:

chown devops:team deploy.sh

��Permissions for Directories

Permission Effect

r Can list files (ls)

w Can add/remove files

x Can enter directory (cd)

Example:

drwxr-x--- 2 riyas team 4096 Jul 1 logs/

• Owner: full access

• Group: can read and enter

• Others: no access

��Special Permissions

Symbol Name Use

s Setuid / Setgid Run program with owner/group privileges t Sticky Bit Used in /tmp so only file owner can delete their files Example:

chmod +t /shared/folder

ls -ld /shared/folder

# drwxrwxrwt → sƟcky bit set

֥DevOps Use Cases

Task Related Permission Make shell script executable chmod +x script.sh Restrict .env to user only chmod 600 .env

Secure private keys chmod 400 key.pem Allow group to deploy chown :deployers deploy.sh Lock config file chmod 444 config.yml (read-only)

��Summary Table

Octal Symbolic Meaning

777 rwxrwxrwx Everyone full access

Basic info Page 10

��Linux User Management – Full Guide

777 rwxrwxrwx Everyone full access

755 rwxr-xr-x Owner full, others can read/execute 700 rwx------ Only owner access

644 rw-r--r-- Read/write owner, read others

600 rw------- Only owner can read/write

��Practice Exercise

# Create a file

touch test.sh

# Give only the owner full access

chmod 700 test.sh

# Make it executable

chmod +x test.sh

# Change ownership to "devops" user and "ci" group chown devops:ci test.sh

# Set read-only for everyone

chmod 444 test.sh

��Why Is User Management Important? In a multi-user system (like Linux servers or cloud environments), managing users, groups, and permissions ensures:

• Security & isolation

• Controlled access to files/scripts/services

• Auditing and accountability

��1. Understanding Users & Groups ��Types of Users

User Type Description

Root Superuser with all permissions

System Users Created by services (e.g., nginx, mysql)

Regular Users Created by admins for login and tasks

��Groups

• Groups are collections of users.

• Users can belong to multiple groups.

• Groups simplify permission management (e.g., give access to a folder for all devops users).

��2. Create, Delete, and Manage Users ➕Create User

sudo adduser riyas

• Adds user with home directory: /home/riyas

• Prompts to set password

OR basic (less interactive):

Basic info Page 11

OR basic (less interactive):

sudo useradd -m riyas

sudo passwd riyas

➖Delete User

sudo deluser riyas # Keeps home dir

sudo deluser --remove-home riyas

Or:

sudo userdel -r riyas

�� Modify User

sudo usermod -aG devops riyas # Add to 'devops' group sudo usermod -l newname oldname # Rename user

sudo usermod -d /new/home/path riyas # Change home dir

��3. User Info and Files

��Key Files

File Purpose

/etc/passwd User account info (username, UID, GID, shell) /etc/shadow Encrypted passwords

/etc/group Group definitions

/etc/sudoers Who can use sudo

��View User Info

cat /etc/passwd | grep riyas

id riyas # Show UID, GID, groups

groups riyas # Show group memberships

��4. Group Management

➕Create Group

sudo groupadd devops

➕Add User to Group

sudo usermod -aG devops riyas

➖Remove User from Group (manually): sudo gpasswd -d riyas devops

➖Delete Group

sudo groupdel devops

��5. Sudo (Superuser Do) Access Grant Sudo Access to a User

sudo usermod -aG sudo riyas

OR edit sudoers file safely:

sudo visudo

Then add:

riyas ALL=(ALL:ALL) ALL

Basic info Page 12

riyas ALL=(ALL:ALL) ALL

��Limited Sudo (for security):

Only allow specific commands:

riyas ALL=(ALL) NOPASSWD: /bin/systemctl restart nginx

��6. Home Directory & Shell

Task Command

View home echo $HOME

Change default shell chsh -s /bin/bash riyas

Set home manually usermod -d /custom/home riyas

��7. Lock and Unlock Users

��Lock Account

sudo usermod -L riyas

��Unlock Account

sudo usermod -U riyas

��Disable Login Shell (block login)

sudo usermod -s /usr/sbin/nologin riyas

��8. Real DevOps Use Cases

Task Command Create a user for Jenkins agent adduser jenkins\_agent

Restrict a user to deploy only usermod -aG docker deployer Setup SSH-only user Create user, disable password, set up SSH key Group-level permission for /var/www Create webdev group, set directory group ownershipGive NGINX team restart permission only sudo visudo + command restriction SFTP-only access Create chrooted SFTP users with nologin shell

��Cheat Sheet Summary

Task Command

Add user adduser <name>

Delete user deluser <name> or userdel -r

Add to group usermod -aG <group> <user>

List groups groups <user>

Create group groupadd <group>

Lock account usermod -L <user>

Give sudo access usermod -aG sudo <user>

⚙ Basic info Page 13

⚙ Linux Process Management – Detailed Guide

��What Is a Process?

A process is any program or command that is running on your Linux system. It has a unique PID (Process ID) and can be running, sleeping, stopped, or zombie.

��Process States

State Meaning

R Running

S Sleeping (idle but waiting)

T Stopped

Z Zombie (terminated but not cleaned up)

D Uninterruptible sleep (I/O wait)

��1. View Running Processes

��List All Processes

ps -ef

• Shows user, PID, parent PID (PPID), time, and command ��Real-Time Monitoring

top

• Interactive, live view of CPU, memory, and processes htop

• Better version of top (requires installation):

sudo apt install htop # Debian/Ubuntu

sudo yum install htop # RHEL/CentOS

��2. Search for a Specific Process ps aux | grep nginx

pgrep nginx # Show PID(s) of nginx

��3. Understand Process Hierarchy pstree

• Visualizes parent-child process structure

��4. Start a Process

./script.sh

��Run in Background

./script.sh &

⏱Keep Running AŌer Logout (Use nohup) nohup ./script.sh &

��5. Kill/Stop/Manage a Process Basic info Page 14

Action Command

Kill by PID kill <PID>

Force kill kill -9 <PID>

Kill by name pkill nginx

Kill all matching killall nginx

Example:

kill -9 1243

pkill -f python

��6. Foreground vs Background Jobs ��Run in Foreground

./longtask.sh

➕Move to Background

• Press Ctrl + Z → pauses task

• Run bg → resumes in background

��View Jobs

jobs

↩ Bring Back to Foreground fg %1

��7. Set Priority (nice, renice)

��Launch with Low Priority

nice -n 10 ./task.sh

��Change Priority of Running Process renice -n 5 -p 1234

• Lower value = higher priority

• -20 (highest), 19 (lowest)

��8. Process Details & Memory/CPU Stats ��Check Memory Usage

ps aux --sort=-%mem | head

��Check CPU Usage

ps aux --sort=-%cpu | head

േ9. DevOps Use Cases

Task Command Restart Jenkins if not running `pgrep jenkins

Monitor Docker CPU usage top, htop, or docker stats Check stuck process during deploy `ps aux

Kill zombie Python script kill -9 $(pgrep -f script.py) Auto-restart failed process Use supervisord or systemd

Basic info Page 15

��10. Advanced Tools

Tool Use

strace Trace system calls of a process

lsof List files used by processes

watch Re-run a command repeatedly

systemctl Manage background services (systemd)

cron Schedule recurring tasks (not real-time process management but related)

��Summary Table

Command Purpose

ps -ef List all processes

top / htop Live process monitor

��Linux Package Management – Complete Guide

kill, pkill, killall Stop processes

jobs, fg, bg Manage background/foreground jobs

nice, renice Adjust process priority

pgrep, pstree Search or view hierarchy

lsof, strace Debug processes/files

��Practice Scenario

# 1. Create a test script

echo -e "#!/bin/bash\nsleep 300" > sleeper.sh

chmod +x sleeper.sh

# 2. Run it in background

./sleeper.sh &

# 3. View process

ps -ef | grep sleeper

# 4. Kill it

kill -9 <PID>

��What Is Package Management? • Packages are compressed files that contain programs, libraries, or tools. • Package managers are tools to install, update, remove, or search for software on Linux. •

DevOps often relies on package management to automate infrastructure setup (e.g., installing Docker, Ansible, or monitoring tools).

��Types of Linux Package Managers Linux distros use different package managers:

Distro Family Tool File Extension Basic info Page 16

��APT (Advanced Packaging Tool) – Ubuntu/Debian

Distro Family Tool File Extension Debian/Ubuntu apt, dpkg .deb RHEL/CentOS/Fedora yum, dnf, rpm .rpm Arch Linux pacman .pkg.tar.zst

��YUM/DNF – RHEL, CentOS, Fedora

Universal snap, flatpak, AppImage —

��Common Commands

Task Command

Update list of packages sudo apt update

Upgrade all packages sudo apt upgrade

Install package sudo apt install nginx

Remove package sudo apt remove nginx

Search package apt search <name>

Get info apt show <package>

��pacman – Arch Linux & Derivatives

Clean old packages sudo apt autoremove

��Install from .deb File

sudo dpkg -i file.deb

sudo apt -f install # Fix missing dependencies

��snap – Universal Package Manager

DNF is the newer replacement for YUM.

��Common Commands

Task YUM DNF

Install sudo yum install nginx sudo dnf install nginx Remove sudo yum remove nginx sudo dnf remove nginx Update system sudo yum update sudo dnf upgrade Search yum search nginx dnf search nginx

��Install from .rpm sudo rpm -ivh package.rpm

Use yum localinstall package.rpm to auto-resolve dependencies.

sudo pacman -Syu # Full system update

sudo pacman -S package-name # Install

sudo pacman -R package-name # Remove

sudo pacman -Ss search-term # Search

• Works across distros.

• Packages include all dependencies (like containers).

Basic info Page 17

��flatpak – Alternative Universal Manager

• Packages include all dependencies (like containers).

• Slower but very easy to use.

sudo snap install code --classic

sudo snap remove code

snap list

flatpak install flathub com.spotify.Client

flatpak run com.spotify.Client

��DevOps Use Cases

Task Command

Install Docker on Ubuntu sudo apt install docker.io Install Git on CentOS sudo yum install git

Setup Prometheus via RPM sudo rpm -ivh prometheus.rpmInstall Terraform via .deb sudo dpkg -i terraform\_1.7.deb Automate installs in shell scripts Include apt, yum, or snap commands Use Ansible to manage packages apt: name=nginx state=present

��Bonus Tools

Tool Use

dpkg Low-level Debian tool (for .deb)

rpm Low-level Red Hat tool (for .rpm)

aptitude TUI for apt package management

brew Popular on macOS and Linux (Homebrew)

conda/pip Python packages (for ML/dev tasks)

��Sample Practice Task (Ubuntu)

# 1. Update and install

sudo apt update

sudo apt install htop

# 2. Check installed version

htop --version

# 3. Remove it

sudo apt remove htop

��Summary Table

Action APT YUM/DNF RPM Snap Install apt install yum install rpm -ivh snap install Remove apt remove yum remove rpm -e snap remove Update system apt upgrade yum update — —List installed dpkg -l yum list installed rpm -qa snap list

Basic info Page 18

��1. systemctl – Modern Service Manager (systemd-based)

�� Linux Service Management – Complete Guide (Init + Systemd)

��What is a Service?

•

A service (also called a daemon) is a background process, like:

○ nginx, docker, sshd, mysql, etc.

•

Services start:

○ Manually (on demand)

○ Automatically (at boot)

•

In Linux, services are managed using init systems:

○ Older: SysVinit (commands: service, chkconfig)

○ Modern: systemd (commands: systemctl, journalctl)

Most modern Linux distributions (Ubuntu ≥15, CentOS 7+, Debian ≥8, RHEL 7+) use systemd. ✅Common systemctl Commands

Task Command

Start a service sudo systemctl start nginx

Stop a service sudo systemctl stop nginx

Restart a service sudo systemctl restart nginx

Reload without restart sudo systemctl reload nginx

Check status systemctl status nginx

��2. Enable/Disable at Boot

Action Command

Enable auto-start sudo systemctl enable nginx

Disable auto-start sudo systemctl disable nginx

Check if enabled systemctl is-enabled nginx

��3. View Service Logs

journalctl -u nginx

Option Description

-u Show logs for specific service

-f Follow logs (like tail -f)

--since "10 min ago" Filter by time

Example:

journalctl -u docker -f

��4. Service Files and Paths (systemd)

Basic info Page 19

Location Purpose

/etc/systemd/system/ Custom service unit files

/lib/systemd/system/ Default service unit files

\*.service files Define how to start/stop a service

��5. Anatomy of a .service File Example: /etc/systemd/system/myapp.service [Unit]

Description=My Flask App

After=network.target

[Service]

ExecStart=/usr/bin/python3 /opt/myapp/app.py WorkingDirectory=/opt/myapp

Restart=always

User=ubuntu

[Install]

WantedBy=multi-user.target

Enable and Start It:

sudo systemctl daemon-reload

sudo systemctl enable myapp

sudo systemctl start myapp

��6. Legacy Commands (SysVinit-based distros) If your system uses SysVinit (e.g., CentOS 6):

Task Command

Start sudo service nginx start

Stop sudo service nginx stop

Restart sudo service nginx restart

Enable at boot chkconfig nginx on

Disable chkconfig nginx off

��7. DevOps Real-World Use Cases Scenario Command Restart Jenkins after update systemctl restart jenkins Enable Docker at boot systemctl enable docker Check failed services systemctl --failed View service logs after crash journalctl -xe Set up custom monitoring service Write .service file + enable it Auto-restart failed app Add Restart=always in .service file

��Summary: systemctl Cheat Sheet Command Use

systemctl start <service> Start service now Basic info Page 20

systemctl start <service> Start service now ��Linux Network Management – Complete Guide

systemctl stop <service> Stop it

systemctl restart <service> Restart it

systemctl status <service> Show status

systemctl enable <service> Start on boot

systemctl disable <service> Don’t start on boot

journalctl -u <service> View logs

systemctl daemon-reexec Reload systemd itself

��Best Practices (DevOps)

• Use systemctl over service on modern distros.

•

For scripts, check service status before acting:

if systemctl is-active --quiet nginx; then

echo "Nginx is running"

fi

• Set Restart policiesin .service files for resilience.

• Always daemon-reload after editing service files.

��Why Network Management Matters In Linux-based systems (servers, containers, VMs), network management helps to: • Configure IP addresses, DNS, routing

• Check connectivity and troubleshoot issues

• Manage firewalls, ports, and network services

• Set up virtual networks (important in Docker/Kubernetes)

��1. Essential Networking Commands ��Check IP Address & Interfaces

ip addr show # Modern

ifconfig # Legacy (requires `net-tools`)

hostname -I # Show IP only

��View Routing Table

ip route show

route -n

��Check DNS Info

cat /etc/resolv.conf

��View/Change Hostname

hostname # Show hostname

hostnamectl set-hostname newname

��2. Connectivity & Troubleshooting Tools Basic info Page 21

Tool Use

ping <host> Check if host is reachable

��Using nmcli (NetworkManager CLI – GUI-Based Systems)

traceroute <host> Show path to host

nslookup <host> DNS resolution

dig <host> Advanced DNS info

telnet <host> <port> Test port connectivity

nmap <host> Scan open ports

netstat -tulnp Show listening ports (TCP/UDP)

ss -tuln Modern version of netstat

curl, wget Test HTTP/HTTPS connections

tcpdump Packet capture tool

ip a, ip r, ip link Interface and routing config

��3. Configure Network Interfaces ��Using ip (Modern Tool)

Assign IP manually:

sudo ip addr add 192.168.1.10/24 dev eth0

sudo ip link set eth0 up

Remove IP:

sudo ip addr del 192.168.1.10/24 dev eth0

nmcli device status # Show devices

nmcli connection show # List connections

nmcli connection up "Wired connection 1" # Enable connection

��4. Network Configuration Files File Purpose /etc/hosts Map hostnames to IPs manually /etc/hostname System's hostname /etc/resolv.conf DNS nameservers /etc/network/interfaces (Debian-based static IP config) /etc/sysconfig/network-scripts/ifcfg-\* (RHEL-based network config)

��5. Firewall Management

��UFW (Ubuntu/Debian)

sudo ufw status

sudo ufw allow 22 # Allow SSH

sudo ufw deny 80

sudo ufw enable

��Firewalld (RHEL/CentOS/Fedora)

sudo firewall-cmd --list-all

sudo firewall-cmd --add-port=8080/tcp --permanent

Basic info Page 22

sudo firewall-cmd --add-port=8080/tcp --permanent

sudo firewall-cmd --reload

��iptables (Legacy/Manual)

sudo iptables -L # List rules

sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT Many DevOps tools still use iptables rules underneath.

��6. Host to Host Communication Task Command

Test SSH ssh user@host

Share file scp file.txt user@host:/path Sync folders rsync -av /source/ user@host:/target/ Setup port forwarding ssh -L 8080:localhost:80 user@remote

��7. DevOps Real-World Use Cases Scenario Tool/Command Check server port before deploy `ss -tuln

Validate DNS for API call dig api.myapp.com Configure static IP on cloud VM /etc/netplan/\*.yaml or nmcli Expose container port to host docker run -p 8080:80 nginx Detect broken microservice curl http://service:port/health Monitor traffic iftop, nethogs, tcpdump Test firewall blocking telnet host port or nmap host

��Summary Table

Tool Purpose

ip Modern IP and route management ss, netstat View listening services

ping, traceroute Network diagnostics

curl, wget Test HTTP/HTTPS

dig, nslookup DNS tools

nmap Port scanner

iptables, ufw, firewalld Firewalls

nmcli, nmtui GUI/CLI network config

��Practice Task

# Check IP and routes

ip a

ip route

# Ping gateway

ping 192.168.1.1

Basic info Page 23

ping 192.168.1.1

# Test DNS resolution

dig google.com

# Scan open ports

nmap localhost

# Check service listening

ss -tuln | grep 22

# Allow HTTP in firewall

sudo ufw allow 80

��Linux TroubleshooƟng – Full Guide

��What Is Troubleshooting? Troubleshooting in Linux means identifying and fixing:

• System errors

• Service failures

• Performance bottlenecks

• Network issues

• Disk problems

• Permissions and access issues

��1. Basic Diagnostic Questions Before diving into tools, ask:

• What exactly isn’t working?

• When did it start?

• Has anything changed recently (updates, config, deployments)? • Can it be replicated?

• Logs available?

��2. System Health Check

➤ CPU, Memory, Load

top

htop

uptime # Check load average

free -h # Check memory usage

vmstat 1 # System performance in real-time ➤ Disk Usage

df -h # Show available disk space

du -sh \* # Check size of subdirectories

lsblk # List block devices

➤ Check Disk Errors

dmesg | grep -i error

sudo smartctl -a /dev/sda

��3. Network Troubleshooting Task Command

Basic info Page 24

Task Command

Check IP address ip a or ifconfig

Ping a host ping 8.8.8.8 or ping google.com Trace route traceroute google.com DNS check dig, nslookup

Port test telnet host port, nmap, or nc -zv host port View open/listening ports ss -tuln, netstat -tulnp

Restart network systemctl restart NetworkManager or sudo service networking restart

��4. Log Files (Goldmine of Info)

Location Purpose

/var/log/syslog or /var/log/messages General system logs

/var/log/auth.log Login attempts, sudo access /var/log/dmesg Kernel and hardware logs /var/log/nginx/ or /var/log/httpd/ Web server logs

/var/log/mysql/ DB logs

/var/log/journal/ systemd logs (if journald enabled) View logs:

tail -n 50 /var/log/syslog

journalctl -xe

��5. Service Failures (Systemd-based systems) Check status and logs:

systemctl status nginx

journalctl -u nginx -b

Restart or reload:

sudo systemctl restart nginx

sudo systemctl daemon-reload

��6. Permission Issues

Common symptoms:

• Permission denied errors

• Files not executable

• Services unable to read config files

Commands:

ls -l filename

chmod +x script.sh

chown user:group file

Check if service is running as correct user (e.g., nginx, www-data).

⚡7. Application Debugging (e.g., Python, Bash, Node.js) Tool Use

Basic info Page 25

bash -x script.sh Debug bash scripts line by line python3 -m pdb script.py Python debugger

npm run dev or node app.js Watch console logs for Node.js curl -v Verbose API call debugging

��8. Database Troubleshooting Action Command

Check MySQL status systemctl status mysql

Access MySQL mysql -u root -p

View DB logs /var/log/mysql/error.log

PostgreSQL logs /var/log/postgresql/postgresql.log Test DB connection mysqladmin ping or psql -h host -U user

��9. Package/Dependency Problems •

Debian/Ubuntu:

sudo apt update

sudo apt install -f

sudo dpkg --configure -a

•

RHEL/CentOS:

sudo yum clean all

sudo yum check

��10. Process & Resource Issues

Task Command

List processes ps -ef, top, htop

Kill process kill <PID> or pkill <name>

Zombie process check `ps aux

Memory hogs `ps aux --sort=-%mem CPU hogs `ps aux --sort=-%cpu

��11. Reboot, Shutdown, Recovery Task Command

Reboot sudo reboot

Shutdown sudo shutdown now Rescue Mode (grub) Hold Shift at boot > Select Advanced > Recovery mode Remount root fs mount -o remount,rw /

Filesystem check fsck /dev/sda1 (carefully, needs unmounted disk)

��12. Troubleshooting Checklist

✅Is the service running?

✅ Basic info Page 26

✅Is the port open and listening?

✅Can you ping / curl the target?

✅Do you see anything in the logs?

✅Is it a permission or ownership problem?

✅Is the disk or memory full?

✅Has anything changed (deployment, config, updates)?

��Tools Worth Installing

Tool Description

htop Advanced resource monitor

iftop, nethogs Live network usage

ncdu Disk usage analyzer

strace Trace system calls

lsof List open files

tcpdump Network packet capture

��Sample Scenario to Practice

# Your service stopped working

systemctl status myapp.service # Step 1: Check status

journalctl -u myapp -n 50 # Step 2: Check logs

ss -tuln | grep 8080 # Step 3: Port listening?

df -h # Step 4: Disk space?

ps -ef | grep myapp # Step 5: Process alive?

• Full admin (sudo) access

• Partial (limited command) access

• With clear examples, commands, and explanations for each step

േScenario Overview

Assume you’re a DevOps/Linux admin and need to:

1. Add a new user

2.

Give them:

○ Either full sudo (admin) access

○ Or limited command access (e.g., restart Apache only)

✅Step-by-Step: Create and Manage Linux User Access ��Step 1: Create a New User

sudo adduser john

��This creates a new user john with a home directory.

��Step 2: Set a Password for the User

sudo passwd john

�� Basic info Page 27

��Prompt will ask for and confirm a password.

��Step 3: Give Full Admin (sudo) Access ➤ Option 1: Add to the sudo group (Debian/Ubuntu)

sudo usermod -aG sudo john

➤ Option 2: For CentOS/RHEL

sudo usermod -aG wheel john

��This gives full root-equivalent access via sudo.

✅Test:

Login as john and run:

sudo whoami

Expected output: root

��PARTIAL / LIMITED ACCESS (For Security or Compliance)��Step 4: Allow Specific Command via sudoers Use the visudo command to safely edit the sudoers file:

sudo visudo

➡ Inside the file, add the below rule at the end:

john ALL=(ALL) NOPASSWD: /bin/systemctl restart apache2 ��Explanation:

• john: The username

• ALL=(ALL): Can run as any user

• NOPASSWD: Won’t be prompted for password • /bin/systemctl restart apache2: Only this command allowed with sudo ✅Test:

Login as john, then run:

sudo /bin/systemctl restart apache2

Should work ✅ But:

sudo apt update

Will result in permission denied ❌

✳BONUS: Allow MulƟple Commands john ALL=(ALL) NOPASSWD: /sbin/ifconfig, /usr/bin/apt-get update

��How to Find Command Full Paths Use:

which <command>

Example:

which systemctl

��Restrict Shell Access (Optional)

To block shell login but allow cron/FTP:

sudo usermod -s /usr/sbin/nologin john

��Check Group of a User

groups john

Basic info Page 28

groups john

��Remove a User’s Sudo Access

Remove user from sudo group:

sudo deluser john sudo # Ubuntu/Debian

sudo gpasswd -d john wheel # RHEL/CentOS

Or remove entry from visudo.

��Summary Table

Task Command

Add user sudo adduser john

Set password sudo passwd john

Full admin sudo usermod -aG sudo john

Partial access sudo visudo ➝ add custom rule Check access sudo -l -U john

Remove sudo sudo deluser john sudo

��Real-World Example Use Case (DevOps): Use Case: Give DevOps intern only permission to restart Nginx and view logs 1. Create user:

sudo adduser devopsintern

1. Edit sudoers via visudo:

devopsintern ALL=(ALL) NOPASSWD: /bin/systemctl restart nginx, /usr/bin/tail -n 100 /var/log/nginx/error.log

✅Now they can:

sudo /bin/systemctl restart nginx

sudo /usr/bin/tail -n 100 /var/log/nginx/error.log

But cannot do anything else with sudo.

�� Linux File System Structure – Detailed Notes The Linux file system is a hierarchical directory structure that starts from the root (/) directory and branches out to other subdirectories.

��1. Root Directory /

• The top-level directory of the Linux file system.

• All files and directories start from here.

• It is the parent of all other directories.

��2. Standard Directories Under / Directory Description

/bin Essential user binaries (commands): ls, cp, mv, rm, etc.

/sbin System binaries: Admin-level tools like iptables, reboot, ifconfig. /boot Contains bootloader files, Linux kernel (vmlinuz), initrd, GRUB config.

Basic info Page 29

/boot Contains bootloader files, Linux kernel (vmlinuz), initrd, GRUB config. /dev Device files: e.g., /dev/sda, /dev/tty0. Represents hardware devices as files. /etc System-wide configuration files: e.g., /etc/passwd, /etc/hosts, /etc/fstab. /home User home directories: /home/riyas, /home/mohan. /lib Essential shared libraries for binaries in /bin and /sbin. /media Mount point for removable media: USB, CD/DVD auto-mounted here. /mnt Used for temporarily mounting file systems manually by the user. /opt Optional application packages (e.g., third-party apps like Chrome, VMware). /proc Virtual filesystem providing process and kernel info (e.g., /proc/cpuinfo). /root Home directory of root user. Different from /home/root. /run Runtime data since the last boot, like PID files, socket files. /srv Service-related data for servers like FTP, WWW.

/sys Virtual filesystem showing kernel info about devices. /tmp Temporary files. Cleared on reboot.

/usr Secondary hierarchy for read-only user data; has subdirs like /usr/bin, /usr/lib. /var Variable data like logs, spool files, cache, and emails.

േ3. Important /etc Files and Their Use File Purpose

/etc/passwd Stores user account information /etc/shadow Contains user password hashes

/etc/group Stores group info

/etc/hostname Sets system hostname

/etc/fstab Contains auto-mount information /etc/hosts Maps IP addresses to hostnames

/etc/resolv.conf DNS configuration

/etc/systemd/ Contains service files and system boot configuration

�� 4. Important /var Subdirectories Directory Purpose

/var/log System log files (important for troubleshooting) /var/spool Print and mail spools

/var/tmp Temp files preserved between reboots

/var/cache Cached data from applications

/var/www Web files (for Apache/Nginx servers)

��5. Virtual Filesystems: /proc, /sys, /dev •

/proc:

○ Dynamic, virtual filesystem.

Used to access process info, system uptime, memory stats.

Basic info Page 30

○ Used to access process info, system uptime, memory stats.

○ Example: cat /proc/cpuinfo, cat /proc/meminfo.

•

/sys:

○ Interfaces with kernel devices.

○ Helps in examining and interacting with hardware.

•

/dev:

○ Represents devices as files.

○ Example: /dev/sda = hard disk, /dev/ttyUSB0 = USB device.

��6. Package & Binary Locations

Path Use

/bin Essential system binaries

/usr/bin Non-essential user commands

/sbin, /usr/sbin Admin commands

/lib, /usr/lib Shared libraries

/opt Third-party packages

��7. User & Permissions

• Each user has a home directory inside /home.

• The root user’s home is /root.

• Use ls -l / to view permissions and ownership of top-level directories.

�� 8. Mount Points

• Devices or partitions are attached to the Linux file system via mount points. •

Example:

○ mount /dev/sdb1 /mnt/usb

○ /mnt, /media, /run/media are common mount paths.

��9. Key Concepts to Remember

• Everything in Linux is a file (including hardware and processes). • The file system follows FHS (Filesystem Hierarchy Standard).

• Root (/) is the origin — all other directories are children of it. • Separation of concerns: logs, binaries, configs, libraries —all have dedicated locations.

��Diagram of Linux File System Hierarchy /

├── bin

├── boot

├── dev

├── etc

├── home

│ ├── user1

│ └── user2

├── lib

├── media

├──

Basic info Page 31

├── mnt

├── opt

├── proc

├── root

├── run

├── sbin

├── srv

├── sys

├── tmp

├── usr

│ ├── bin

│ ├── lib

│ └── share

└── var

├── log

├── cache

└── tmp

Basic info Page 32