CompTIA Linux+ XK0-005

Full Learning Guide

*Unlike most study guides, this one will **teach you Linux by doing**, using your very own **simple cloud Linux server on AWS**. This approach prepares you not only to pass the exam — but to **prove your skills in a job interview or on the job**.*

Learning Objectives and Expectations

You'll master:

- Every official CompTIA Linux+ (XK0-005) exam objective
- All critical commands, services, filesystems, permissions, tools, and logs
- How to troubleshoot like a Linux admin
- How to automate tasks and manage containers

Each domain includes:

- Full concept breakdowns, memory aids, and exam tips
- Command-by-command practice
- Real-world examples

Linux+ XK0-005 Domains

Each domain is weighted differently on the exam, with Security Operations being the largest:

- **Domain 1:** System Management (32%)
- **Domain 2:** Security (21%)
- **Domain 3:** Scripting, Containers & Automation (19%)
- **Domain 4:** Troubleshooting (28%)

Quick Reminder: How the Exam Works

- Number of Questions: Up to 90
- Format: Multiple choice + Performance-Based Questions (PBQs)
- Time Limit: 90 minutes
- Passing Score: 720/900 (about 80%)
- Test Provider: Pearson VUE (onsite or online)

Remember — you don’t need to be perfect to pass!

The passing score is about **80%**, meaning you can **miss up to 18 questions** and still pass. Focus on **understanding and practice**, not memorizing every flag.

Missing a few tricky questions won't ruin your chances — **stay calm**, trust your preparation, and keep moving forward.

Why Linux + AWS = Career Power

- **Linux powers 90% of servers and cloud infrastructure**
- **AWS is the #1 cloud provider**, used by companies like Netflix, Meta, NASA, and Amazon itself
- Learning both gives you:
  - Cloud admin experience
  - Linux CLI confidence
  - Resume-ready practice
  - A competitive edge in IT, DevOps, cybersecurity, and support roles

Coming Up Next: Build Your Practice Lab

The very next page of this guide shows you:

- How to create an **AWS Free Tier account**
- How to launch a **Ubuntu EC2 server**
- How to connect via **SSH**
- And how to start practicing Linux — right away

Get ready to log in, type real commands, and build real skills from day one

Before You Begin: Get Ready for Linux+ Practice

The **Linux+ exam** assumes you can work from a command line, manage services, secure a system, and troubleshoot problems. The best way to learn is by doing — and the best practice environment is **in the cloud**.

Why Use AWS to Practice?

Amazon Web Services (AWS) offers a **Free Tier** that includes:

- **750 hours/month** of EC2 compute (t2.micro or t3.micro)
- **Free for 12 months**
- Easily spin up a **Linux server** (e.g., Ubuntu)
- Safe, disposable sandbox for practicing commands

How to Get Started with AWS and EC2?

Step 1: Create an AWS Free Tier Account

1. Visit: https://aws.amazon.com/free
2. Click **"Create a Free Account"**
3. Enter:
    - Your email and password
    - Choose an AWS root user name
4. Select **"Personal" account**
5. Add a **credit card** (required, but you won't be charged if you stay within the Free Tier)
6. Choose a **support plan** (select "Basic" – it's free)
7. Wait for email confirmation (~5 minutes)

Step 2: Launch a Free Ubuntu EC2 Instance

1. Log in to https://console.aws.amazon.com
2. Go to **EC2 Dashboard** → Click **"Launch instance"**
3. Configure:

- **Name**: LinuxPracticeBox
- **Amazon Machine Image (AMI)**: Choose **Ubuntu Server 22.04 LTS (Free Tier Eligible)**
- **Instance type**: Select **t2.micro** or **t3.micro** (both Free Tier)
- **Key pair (login)**:
  - Create a **new key pair** (RSA)
  - Download the .pem file (e.g., linux-key.pem)
- **Network settings**:
  - Allow **SSH (port 22)** from your IP

4. Click **“Launch Instance”**
5. Wait for instance to be in **"running"** state

Step 3: Connect to Your EC2 Instance

On macOS:

- chmod 400 linux-key.pem
- ssh -i linux-key.pem ubuntu@<your-ec2-public-ip>

On Windows:

- *Use **Windows Terminal** or **Git Bash***
- *Or install **PuTTY**, and convert the .pem to .ppk using PuTTYgen*

Step 4: Start Practicing

Once logged in, type:

1. ls
2. pwd
3. sudo apt update && sudo apt upgrade

Try:

- Creating files (touch, nano)
- Adding users (sudo adduser)
- Installing packages (sudo apt install)

Use ChatGPT if stuck at any point….

What is Linux and Why Is It So Popular?

Linux is a **free and open-source operating system** modeled after UNIX. It is the software that sits between hardware and applications, managing system resources and providing the interface users and programs interact with.

Unlike Windows or macOS, Linux isn't just a single product — it's a **kernel** (the core of the operating system) that forms the foundation for **many different Linux distributions** (also called “distros”) such as:

- Ubuntu
- CentOS / RHEL
- Debian
- Fedora
- Kali Linux
- Arch Linux
- openSUSE

Each distribution packages the Linux kernel with software tools, libraries, and user interfaces to serve different use cases (servers, desktops, security testing, etc.).

Key Features of Linux

- **Free and Open Source**: Anyone can use, modify, and distribute Linux.
- **Multitasking and Multiuser**: Supports multiple users and tasks at the same time.
- **Highly Secure**: Designed with strong permission models, firewall tools, and security modules (like SELinux).
- **Customizable**: From the command-line shell to the kernel itself, Linux is flexible and modular.
- **Portable**: Runs on everything from smart TVs and routers to servers and supercomputers.
- **Stable and Reliable**: Linux systems can run for years without reboots or crashes.

Why Is Linux So Popular?

1. **Dominates the Server Market**

Linux powers:

- Over 90% of the world's **web servers**
- Most **cloud infrastructure** (AWS, Azure, GCP)
- The majority of **supercomputers**

2. **Preferred by Developers and IT Professionals**

- Full control via command-line
- Native support for **open-source tools**, containers, scripting, and automation
- Strong integration with DevOps, CI/CD, and cloud workflows

3. **Ideal for Learning and Experimentation**

- Free to install and run
- Safe sandbox for learning administration, security, and development
- Certifications like **CompTIA Linux+**, **RHCSA**, and **LPIC** are highly valued

4. **Powers Android and Embedded Systems**

- The **Android OS** is based on the Linux kernel
- Used in routers, smart devices, automotive systems, and IoT

5. **Security and Performance**

- Built-in firewalls, SELinux/AppArmor, minimal default services
- Efficient resource usage — ideal for cloud and edge computing

---

Conclusion

Linux is everywhere — from your phone to the cloud. It is the **foundation of modern IT infrastructure**, and learning Linux is a must for anyone in **cybersecurity, systems administration, DevOps, cloud engineering, or networking**.

Its openness, flexibility, and proven reliability make it the **operating system of choice** for professionals and enterprises around the globe.

Domain 1: System Management (32%)

This domain covers the following key areas:

- Linux Filesystem Structure
- Boot Process and Kernel
- Managing Files and Directories
- Partitioning and Filesystems
- Logical Volume Manager (LVM)
- RAID and Disk Management
- Mounting Local and Network Filesystems
- Managing Processes and Services
- Scheduling Jobs
- Software Installation and Management

Each section below provides a **detailed, beginner-friendly explanation** of key topics and commands you need to master.

1.1 Linux Filesystem Structure

Linux follows a standard directory structure defined by the FHS. It determines where files should be located.

Key Directories:

Directory	Purpose
/	Root directory. Everything starts here.
/bin	Essential user binaries (e.g., ls, cat, cp).
/sbin	System binaries (e.g., ifconfig, shutdown).
/etc	System configuration files.
/home	User home directories.
/root	Root user's home directory.
/var	Variable files (e.g., logs, spool files).
/tmp	Temporary files. Cleared on reboot.
/usr	User applications and utilities.
/lib, /lib64	Shared libraries for binaries in /bin and /sbin.
/dev	Device files (e.g., /dev/sda, /dev/null).
/mnt, /media	Mount points for removable or temporary filesystems.
/opt	Optional software.

Command to View Filesystem Tree:

```
tree /
```

1.2 Boot Process and Kernel

Concept: Linux Boot Process

1. **BIOS/UEFI**: Initializes hardware.
2. **Bootloader (GRUB)**: Loads the Linux kernel.
3. **Kernel**: Loads essential drivers and mounts the root filesystem.
4. **init/systemd**: Starts user-space services and targets.

GRUB (GRand Unified Bootloader):

- **/boot/grub/grub.cfg**: Main config file.
- **grub2-mkconfig -o /boot/grub2/grub.cfg**: Regenerate GRUB config.

Kernel Image:

- Stored in /boot as vmlinuz.

Initial RAM Disk:

- Temporary root filesystem to help kernel boot.
- Usually named initrd or initramfs.

Key Commands:

```
uname -r      # Show running kernel version
lsmod         # List loaded kernel modules
modprobe <mod>  # Load a kernel module
rmmod <mod>     # Remove a kernel module
```

1.3 Managing Files and Directories

Concept: File Operations

Command	Description
ls	List files
cd	Change directory
pwd	Show current directory
mkdir	Create directory
rmdir	Remove empty directory

rm	Delete files/directories
cp	Copy files/directories
mv	Move/rename files/directories
touch	Create empty file
find	Locate files
file	Determine file type
stat	View metadata about a file

Example:

```
find /etc -name "*.conf"
stat /var/log/syslog
file myscript.sh
```

Concept: File Links

- **Hard link**: Shares same inode. Cannot link to directories or across filesystems.
- **Symbolic link (symlink)**: Pointer to another file. Can link across filesystems.

Commands:

```
ln file1 file2     # Hard link
ln -s target link  # Symbolic link
```

1.4 Partitioning and Filesystems

Concept: Partition Tables

- **MBR** (Master Boot Record): Max 4 primary partitions.
- **GPT** (GUID Partition Table): Supports >4 partitions, required for UEFI.

Tools:

```
fdisk /dev/sdX     # For MBR
gdisk /dev/sdX     # For GPT
parted /dev/sdX    # Advanced GPT/MBR tool
```

Concept: Filesystem Types

Filesystem	Use
ext4	Most common Linux FS
xfs	High performance, RHEL default
btrfs	Snapshotting, copy-on-write
vfat/ntfs	Windows compatibility

Creating and Checking Filesystems:

```
mkfs.ext4 /dev/sdX1
mkfs.xfs /dev/sdX1
fsck /dev/sdX1        # Check and repair
mount /dev/sdX1 /mnt   # Mount filesystem
umount /mnt           # Unmount
```

1.5 Logical Volume Manager (LVM)

Concept: Flexible disk management

Component	Description
PV (Physical Volume)	Disk/partition (e.g., /dev/sdb1)
VG (Volume Group)	Pool of PVs
LV (Logical Volume)	Usable partitions from VG

Workflow:

```
pvcreate /dev/sdb1
vgcreate my_vg /dev/sdb1
lvcreate -L 10G -n my_lv my_vg
mkfs.ext4 /dev/my_vg/my_lv
mount /dev/my_vg/my_lv /mnt
```

Resizing LVM:

```
lvextend -L +5G /dev/my_vg/my_lv
resize2fs /dev/my_vg/my_lv
```

1.6 RAID and Disk Management

Concept: RAID (Redundant Array of Independent Disks)

Level	Description
RAID 0	Striping (no redundancy, high speed)
RAID 1	Mirroring (redundancy)
RAID 5	Striping with parity
RAID 6	Double parity
RAID 10	Mirrored striping

Software RAID using mdadm:

```
mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
cat /proc/mdstat     # View RAID status
mdadm --detail /dev/md0
```

1.7 Mounting Local and Network Filesystems

Concept: Mounting Filesystems

Mount a partition to a directory:

```
mount /dev/sdb1 /mnt
umount /mnt
```

Persistent Mount (in /etc/fstab):

```
UUID=xxx-xxx /mnt ext4 defaults 0 2
```

View UUIDs:

```
blkid
```

Network Mounts:

- **NFS**: mount -t nfs server:/share /mnt
- **CIFS (Windows)**: mount -t cifs //server/share /mnt -o user=username

---

1.8 Managing Processes and Services

Concept: Process Management

Command	Description
ps aux	Show running processes
top / htop	Interactive monitoring
kill PID	Terminate process
nice / renice	Adjust priority
bg / fg	Background/foreground jobs
jobs	List current jobs

Concept: System Services (systemd)

Command	Description
systemctl start nginx	Start service
systemctl stop nginx	Stop service

systemctl restart nginx	Restart
systemctl status nginx	Status
systemctl enable nginx	Start at boot
systemctl disable nginx	Disable at boot

1.9 Scheduling Jobs

Concept: Cron Jobs (repeating)

- User jobs: crontab -e
- System jobs: /etc/crontab

Cron Syntax:

```
* * * * * command
| | | | |
| | | | +----- Day of week (0 - 7)
| | | +------- Month (1 - 12)
| | +--------- Day of month (1 - 31)
| +----------- Hour (0 - 23)
+------------- Minute (0 - 59)
```

Example:

0 2 * * * /usr/local/bin/backup.sh

Concept: at (one-time jobs)

echo "shutdown -h now" | at 23:00

1.10 Software Installation and Management

Debian-based systems (APT):

apt update
apt install package
apt remove package
dpkg -i package.deb

RHEL-based systems (YUM/DNF):

dnf install package
dnf remove package
dnf update

rpm -ivh package.rpm

Snap and Flatpak (Universal Package Formats):

snap install <pkg>
flatpak install <pkg>

Summary of Domain 1

- Understand Linux filesystem layout
- Learn boot sequence and systemd role
- Use file/directory manipulation commands
- Partition disks and manage filesystems (ext4, xfs, etc.)
- Set up and manage LVM volumes
- Configure and monitor software RAID
- Mount local and network filesystems
- Monitor and control processes
- Schedule recurring and one-time jobs
- Install and manage software packages (APT, DNF, RPM, Snap)

Domain 2: Security (21%)

2.1 Security Concepts and Best Practices

Learn This: Security best practices form the foundation for protecting Linux systems.

Key Concepts:

- **CIA Triad**:
  - **Confidentiality** – prevent unauthorized access (e.g., encryption, file permissions).
  - **Integrity** – prevent unauthorized changes (e.g., hashes, checksums).
  - **Availability** – ensure services remain accessible (e.g., uptime, backups, RAID).
- **PKI (Public Key Infrastructure):**
  - Uses **asymmetric encryption** (public/private key pairs).
  - Certificates (e.g., .crt, .pem) bind identities to public keys.
  - Stored in: /etc/pki/, /etc/ssl/
- **Digital Signatures & Hashing:**
  - Ensure file **integrity** and **non-repudiation**.
  - Common hash tools:

    ```
    sha256sum file.txt
    md5sum file.txt
    ```

- **Secure Boot:**
  - Ensures only **signed** OS bootloaders/kernels run.
  - Helps prevent rootkits and tampering at boot.
- **Linux Hardening:**
  - Disable unused services: systemctl disable telnet
  - Restrict root login: /etc/ssh/sshd_config → PermitRootLogin no
  - Enforce password policies with **PAM**, chage, and passwd
  - Use **firewalls** and **SELinux/AppArmor**
  - Configure **automatic security updates**:

    ```
    apt install unattended-upgrades
    dnf install dnf-automatic
    ```

2.2 User and Group Management

Learn This: Control who can access a system and what they can do.

Account Management Commands:

Task	Command
Add user	useradd bob
Set password	passwd bob
Add group	groupadd devs
Add user to group	usermod -aG devs bob
Delete user	userdel -r bob
View user info	id bob, getent passwd bob

Key Files to Know:

- /etc/passwd – user account info.
- /etc/shadow – hashed passwords and expiration.
- /etc/group – group info.
- /etc/gshadow – group passwords (rarely used).
- /etc/skel/ – files copied into new users' home dirs.

Password Aging and Expiry:

```
chage -l bob          # View password aging info
chage -M 90 -m 7 bob     # Set max and min age
passwd -l bob          # Lock user account
```

Login Access Control:

- /etc/security/access.conf – limit access by user/tty/host.
- /etc/ssh/sshd_config – SSH access control: AllowUsers, DenyUsers

---

2.3 Network Security: Firewalls

Learn This: Firewalls protect your system from unwanted traffic.

Firewall Tools:

- **iptables**: Classic CLI firewall tool (netfilter backend).
- **nftables**: Modern replacement for iptables.

- **firewalld**: RHEL-based systems. Zone-based frontend to iptables/nftables.
- **ufw**: Ubuntu's Uncomplicated Firewall.

iptables Basics:

```
iptables -L             # List rules
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -P INPUT DROP     # Default deny
```

firewalld Commands:

```
firewall-cmd --list-all
firewall-cmd --add-service=http --permanent
firewall-cmd --reload
```

Concepts to Learn:

- **Stateful firewall**: Tracks ongoing connections (most common).
- **Stateless firewall**: Each packet is evaluated on its own.
- **Zones**: firewalld grouping by trust level (e.g., public, internal).
- **Services vs Ports**:
  - --add-service=http is easier than --add-port=80/tcp

2.4 Remote Access and Privilege Management

Learn This: SSH allows secure remote login; sudo/su allow elevated privileges.

SSH Security Best Practices:

- **Disable root login**:
  Edit /etc/ssh/sshd_config → PermitRootLogin no
- **Use key-based authentication:**

  ```
  ssh-keygen            # On client
  ssh-copy-id user@host    # Upload public key
  ```

SSH Port Forwarding (Tunneling):

```
ssh -L 8080:localhost:80 user@remote    # Local port forward
ssh -R 9090:localhost:22 user@remote    # Remote port forward
ssh -D 1080 user@remote             # Dynamic SOCKS proxy
```

Privilege Elevation Tools:

Tool	Use
sudo	Run single command as root
su -	Become another user (usually root)
pkexec	Graphical equivalent of sudo
visudo	Safe edit of sudoers file

Sudoers Example:

bob ALL=(ALL) NOPASSWD: /bin/systemctl restart apache2

2.5 File Permissions and Access Control

Learn This: Permissions and controls regulate who can access files.

Standard Unix Permissions:

chmod 755 file # rwxr-xr-x
chown user:group file

Symbol	Meaning
r	Read
w	Write
x	Execute

Special Permission Bits:

Bit	Use
Setuid (chmod 4755)	Run as file owner (e.g., /usr/bin/passwd)
Setgid (chmod 2755)	Run with file's group or inherit directory group
Sticky (chmod 1777)	Prevent others from deleting files (e.g., /tmp)

View with:

ls -l /tmp # drwxrwxrwt

ACLs (Access Control Lists):

Used to give more than one user/group specific permissions.

setfacl -m u:bob:rwx file
getfacl file

Immutable Files (chattr/lsattr):

chattr +i file # Make file immutable

lsattr file

2.6 SELinux and AppArmor

Learn This: These are Mandatory Access Control (MAC) systems.

SELinux (Security-Enhanced Linux)

- **Modes:**
  - **Enforcing**: Policy is active.
  - **Permissive**: Logs but does not enforce.
  - **Disabled**: Turned off (not recommended).

```
getenforce          # Show current mode
setenforce 1        # Enforce mode temporarily
```

- **Contexts:**

  ```
  ls -Z             # Show SELinux context
  chcon -t httpd_sys_content_t file
  restorecon -Rv /var/www/
  ```

- **Booleans:**

  ```
  getsebool -a
  setsebool -P httpd_can_network_connect on
  ```

- **Policy Types:**
  - **Targeted**: Default, applies to selected daemons (e.g., Apache).
  - **Minimum**: More minimal policy coverage.

AppArmor (Ubuntu):

- Uses path-based profiles stored in /etc/apparmor.d/
- Commands:

  ```
  aa-status
  aa-enforce /etc/apparmor.d/usr.sbin.apache2
  ```

Summary of Domain 2

- Understand security principles (CIA, PKI, Secure Boot).
- Harden Linux systems (disable services, enforce password policy).
- Use PAM and login controls to lock users, expire passwords.
- Set and manage permissions, ACLs, and special bits.
- Configure firewalls with firewalld, iptables, ufw.
- Secure remote access with SSH and port forwarding.
- Use SELinux/AppArmor to enforce access controls.

Domain 3: Scripting, Containers, and Automation (19%)

3.1 Shell Scripting and Shell Tools

Learn This: Shell scripting automates routine Linux tasks.

Key Concepts:

- **Shebang (#!/bin/bash)**
  First line in scripts that tells the shell which interpreter to use.

Script Elements to Know:

Element	Example	Purpose
Variable	name="John"	Stores data
If-Else	if [$x -gt 10]; then	Conditional logic
For Loop	for i in 1 2 3; do echo $i; done	Repetition over list
While Loop	while true; do ... done	Repeat until condition is false
Case	case $var in pattern)	Switch-case logic
Functions	myfunc() { echo "Hi"; }	Reusable code blocks
Exit Code	echo $?	Shows last command status (0 = success)

Common Bash Operators:

Comparison	Usage
-eq	Equals (integers)
-lt, -gt	Less/greater than
-f	File exists
-d	Directory exists
-z, -n	Empty or non-empty string

Example: A Simple Script

```
#!/bin/bash
echo "Enter a number:"
read num
if [ "$num" -gt 100 ]; then
  echo "Big number"
else
  echo "Small number"
```

fi

3.2 Command-Line Utilities for Scripting

Learn This: Use standard tools to manipulate data.

Tool	Purpose
grep	Search in text
sed	Stream editor (modify text)
awk	Field-based data processing
cut	Cut specific columns
sort	Sort lines
uniq	Remove duplicates
tr	Translate/delete characters
wc	Word/line/char count
head / tail	First/last lines
find	Locate files
xargs	Build command lines from input
tee	Output to file and stdout

Examples:

```
grep "error" /var/log/syslog
cut -d: -f1 /etc/passwd
awk -F: '{print $1, $3}' /etc/passwd
find /etc -name "*.conf"
```

3.3 Containers: Basics and Usage

Learn This: Containers are lightweight, isolated environments to run applications.

Concepts:

- **Image**: Read-only template.
- **Container**: Running instance of an image.
- **Registry**: Where images are stored (e.g., Docker Hub).

Common Docker/Podman Commands:

Task	Command
Pull Image	docker pull ubuntu
Run Container	docker run -it ubuntu /bin/bash
List Containers	docker ps -a

Start/Stop	docker start / docker stop
Remove	docker rm <id>
Logs	docker logs <name>
Exec	docker exec -it <name> /bin/bash
Build Image	docker build -t myimage .

Port Mapping:

docker run -d -p 8080:80 nginx

Volumes (Persistent Storage):

docker run -v /host/data:/container/data ...

3.4 Git Version Control

Learn This: Git tracks changes in files and code repositories.

Basic Git Workflow:

git clone https://github.com/example/repo.git
cd repo
git status
git add file.txt
git commit -m "Add new feature"
git push origin main

Task	Command
View Log	git log
New Branch	git branch new-feature
Switch Branch	git checkout new-feature
Merge	git merge new-feature
Ignore Files	.gitignore file
View Differences	git diff

Tagging and Releases:

git tag v1.0
git push origin v1.0

3.5 Infrastructure as Code and Automation Tools

Learn This: Automate infrastructure and configuration using tools.

Configuration Management Tools:

Tool	Description
Ansible	Agentless, uses SSH. YAML-based playbooks.
Puppet	Agent-based, uses its own DSL. Declarative.
Chef	Ruby-based configuration automation.
SaltStack	Python-based, agent or agentless.
Terraform	Manages cloud infrastructure (VMs, networks).

YAML and JSON Formats:

- **YAML** is used in Ansible and Kubernetes.
- **JSON** is more strict; used in APIs and Terraform (optionally).

YAML Example:

```
- name: Install Nginx
  apt:
    name: nginx
    state: present
```

Terraform Example:

```
resource "aws_instance" "web" {
  ami = "ami-123456"
  instance_type = "t2.micro"
}
```

3.6 CI/CD and DevOps Concepts

Learn This: CI/CD automates testing and deployment of code.

Term	Description
CI (Continuous Integration)	Automatically test and build code after every change.
CD (Continuous Deployment)	Automatically deploy code to production or staging.
GitOps	Manage infrastructure via Git version control.
Pipeline	Sequence of steps (build, test, deploy).

Tools: Jenkins, GitHub Actions, GitLab CI/CD, CircleCI, TravisCI.

Example Workflow:

- Developer pushes code.

- CI pipeline runs tests.
- CD deploys it if tests pass.

3.7 Orchestration and Advanced Container Topics

Learn This: Kubernetes manages containers across multiple nodes.

Key Concepts:

Term	Description
Pod	A group of one or more containers with shared storage/network.
Service	Stable access point to pods (with load balancing).
Sidecar	Secondary container in a pod (e.g., for logging).
Ambassador	Container that proxies connections to outside resources.
Persistent Volume	Storage that survives pod restarts.
Overlay Network	Allows cross-node container communication.

Compose vs Kubernetes:

- **Compose**: Local, multi-container Docker apps (YAML).
- **Kubernetes**: Scalable orchestration (pods, services, controllers).

Service Mesh:

- A system like **Istio** or **Linkerd** manages service-to-service communication.
- Features: encryption, retries, observability.
- Uses **sidecars** (e.g., Envoy) to intercept traffic.

3.8 Cloud-Init and Bootstrapping

Learn This: Cloud-init automates first-boot setup for cloud servers.

Common Cloud-init Tasks:

- Set hostname
- Add users and SSH keys
- Install packages
- Run scripts

Example:

```
#cloud-config
packages:
 - nginx
runcmd:
 - systemctl enable --now nginx
```

Summary of Domain 3

- Write basic bash scripts with conditionals, loops, functions.
- Use grep, sed, awk, cut, xargs for text/data processing.
- Manage containers using Docker or Podman (start, stop, logs, ports).
- Track code/configuration changes with Git.
- Understand Ansible, Terraform, Puppet, and YAML syntax.
- Grasp CI/CD principles and pipeline stages.
- Recognize Kubernetes concepts: pods, sidecars, overlay networking.
- Use cloud-init for automating VM boot setup in cloud environments.

Domain 4: Troubleshooting (28%)

4.1 Troubleshooting Storage Issues

Learn This: Understand how to detect, diagnose, and fix disk, file system, and I/O problems.

Key Symptoms:

- Low disk space
- Filesystem errors
- Mount issues
- RAID/LVM failures
- High latency or low IOPS

Diagnostic Tools:

Tool	Purpose
df -h	Shows used/free disk space
du -sh /dir	Show space used by a directory
find / -type f -size +1G	Locate large files
iostat	Monitor I/O performance (from sysstat)
lsblk	Display block devices and mount points
blkid	Show block device UUIDs and labels
mount, umount	Mount/unmount filesystems
cat /proc/mdstat	Check RAID status
lvs, vgs, pvs	Check LVM state

Common Storage Issues:

- **Disk full**
  Use df to check space and du/find to locate large files.
- **Inode exhaustion**
  Too many small files → df -i shows inode usage.
- **Unmounted or failed mount**
  Check /etc/fstab for syntax errors or missing devices.
- **RAID degraded or failed**
  Use mdadm --detail and /proc/mdstat to inspect array.
- **LVM volume not mounted or full**
  Extend logical volume:

  lvextend -L +5G /dev/vgname/lvname
  resize2fs /dev/vgname/lvname

4.2 Troubleshooting Network Issues

Learn This: Network issues may result from bad configs, cable failure, DNS problems, or firewalls.

Key Tools:

Tool	Purpose
ip addr, ip link	Show interface state
ip route	View routing table
ping, traceroute	Test connectivity
mtr	Real-time trace + ping
ss, netstat	Check open ports and connections
dig, host, nslookup	DNS queries
ethtool	NIC settings, speed, link
tcpdump	Capture packets
firewall-cmd, ufw, iptables	Check firewall rules

Common Issues:

- **No IP address**
  Use ip addr. If using DHCP, check dhclient, NetworkManager, or systemd-networkd.
- **Cannot ping gateway or internet**
  Use ping, ip route, and ip link to confirm routing and link state.
- **DNS not working**
  Check /etc/resolv.conf or resolvectl. Use dig or host to query DNS directly.
- **Firewall blocking ports**
  Check firewall-cmd --list-all or ufw status.
- **Dropped packets or errors**
  Use ip -s link or ethtool eth0 to look for errors, CRC faults, or collisions.

4.3 Troubleshooting CPU and Memory Issues

Learn This: Detect system overloads and manage performance bottlenecks.

Key Tools:

Tool	Purpose
top, htop	Live system monitoring
ps aux --sort=-%cpu	CPU usage
free -h	Memory usage
vmstat	Memory, CPU, I/O metrics

uptime	Load averages
dstat, iostat	Detailed performance stats

Common Issues:

- **High CPU usage**
  Identify with top or ps, kill or renice the offending process:

  kill -9 PID
  renice +10 PID

- **Out-of-Memory (OOM) killer triggered**
  Check dmesg | grep -i kill to see which process was killed. Consider adding swap:

  swapon --show
  dd if=/dev/zero of=/swapfile bs=1G count=4
  chmod 600 /swapfile
  mkswap /swapfile
  swapon /swapfile

- **Zombie processes**
  Show as <defunct> in ps. Cause: parent didn't wait() on child. Solution: restart parent or orphan them to be reaped by init.

4.4 Troubleshooting User Access and Authentication

Learn This: Users may face login failures due to account, password, PAM, or SSH issues.

Key Checks:

Area	Command/File
User account exists	id user, /etc/passwd
Password aging	chage -l user, /etc/shadow
Locked account	passwd -S user, faillock --user user
Shell available	/etc/passwd field 7
SSH access	/etc/ssh/sshd_config, journalctl -u sshd
PAM logs	/var/log/auth.log or /var/log/secure
Sudo permissions	/etc/sudoers, groups user

Common Issues:

- **Account expired or locked**
  Unlock:

  ```
  passwd -u user
  chage -E -1 user   # Remove expiry
  ```

- **Password expired**
  Change password: passwd user
- **Too many failed logins (pam_faillock)**
  Reset:

  ```
  faillock --user user --reset
  ```

- **Wrong shell or missing home dir**
  Check /etc/passwd. Shell should be valid (e.g., /bin/bash), and home should exist with correct ownership.
- **SSH public key not accepted**
  Check permissions on ~/.ssh, authorized_keys (must be 700 and 600 respectively).
  Confirm key is correctly installed and matches client private key.

4.5 Troubleshooting Services

Learn This: Systemd manages services and their startup behavior.

Key Commands:

Task	Command
List services	systemctl list-units --type=service
Start/stop	systemctl start sshd, systemctl stop nginx
Enable/disable	systemctl enable apache2
Status and logs	systemctl status nginx, journalctl -u nginx

Common Issues:

- **Service fails to start**
  Use systemctl status and journalctl to view logs. Often it's a config error or missing dependency.
- **Port conflict**
  Use ss -tulnp to check if another process is using the port.
- **Service not enabled at boot**
  Enable it:

  ```
  systemctl enable service
  ```

- **Configuration syntax error**
  Test config (if supported):

  apachectl configtest
  nginx -t
  sshd -t

- **Log file issues**
  Check if logs are filling disk. Rotate logs with logrotate.

4.6 Troubleshooting Boot and Kernel Issues

Learn This: Boot issues often stem from kernel errors, bad fstab, or misconfigured GRUB.

Boot Process Overview:

1. BIOS/UEFI
2. GRUB
3. Kernel
4. initramfs
5. systemd

Tools and Checks:

Task	Tool/File
View boot logs	journalctl -b
Check fstab	/etc/fstab
GRUB config	/boot/grub2/grub.cfg or /etc/default/grub
Boot parameters	Press e at GRUB screen
Kernel messages	dmesg
Kernel version	uname -r

Common Issues and Fixes:

- **/etc/fstab errors prevent booting**
  Solution: Boot to recovery mode or Live CD, comment out bad entries in /etc/fstab.
- **Kernel panic or missing initramfs**
  Rebuild initramfs:

  dracut -f /boot/initramfs-$(uname -r).img $(uname -r)

- **Boot into wrong kernel**
  Use GRUB menu to select older kernel, or modify GRUB_DEFAULT in /etc/default/grub and regenerate:

  grub2-mkconfig -o /boot/grub2/grub.cfg

- **GRUB issues (e.g., black screen, grub rescue>)**
  Reinstall GRUB from Live CD:

  mount /dev/sdaX /mnt
  grub2-install --boot-directory=/mnt/boot /dev/sda

Summary of Domain 4

- Interpret and resolve disk space, filesystem, RAID, and LVM issues.
- Use network tools to diagnose DNS, routing, and interface problems.
- Find and manage CPU and memory bottlenecks using top, ps, free, etc.
- Diagnose login and authentication problems (locked accounts, PAM).
- Restart, enable, and inspect services with systemctl and journalctl.
- Troubleshoot boot issues via GRUB, initramfs, kernel logs, and rescue mode.

Full Linux+ XK0-005 Commands List

1. Filesystem and File Management

Command	Description	Common Options	Example
ls	List directory contents	-l (long), -a (all), -h (human-readable)	ls -lah
cd	Change directory	.. (up one level)	cd /etc
pwd	Show current directory	—	pwd
mkdir	Create a new directory	-p (make parent dirs)	mkdir -p /data/logs
rmdir	Remove an empty directory	—	rmdir emptydir
rm	Delete files/directories	-r (recursive), -f (force)	rm -rf /tmp/files
cp	Copy files/directories	-r, -p (preserve), -u (update)	cp -ruv src/ dest/
mv	Move or rename files	—	mv oldname.txt newname.txt
touch	Create empty files	—	touch file.txt
find	Search for files/directories	-name, -type, -size, -mtime, -exec	find /var -name "*.log"
file	Show file type	—	file script.sh
stat	Show detailed file info	—	stat /etc/passwd
du	Disk usage per file/dir	-sh, -a, --max-depth=1	du -sh *
df	Show disk space	-h (human), -i (inodes)	df -hi

mount	Mount a filesystem	-t (type), -o (options)	mount -t ext4 /dev/sdb1 /mnt
umount	Unmount filesystem	—	umount /mnt
lsblk	Show block devices	-f, -o NAME,SIZE,MOUNTPOINT	lsblk -f
blkid	Show device UUIDs	—	blkid /dev/sda1
mkfs	Format a partition	-t ext4/xfs	mkfs -t ext4 /dev/sdb1
fsck	Check filesystem	-y (auto fix)	fsck -y /dev/sdb1

2. User and Group Management

Command	Description	Common Options	Example
useradd	Add user	-m (home), -s (shell), -G (groups)	useradd -m -s /bin/bash -G wheel alice
passwd	Set/modify password	-l (lock), -u (unlock), -e (expire)	passwd alice
userdel	Delete user	-r (remove home)	userdel -r alice
usermod	Modify user	-aG (append to group), -s (shell)	usermod -aG dev alice
groupadd	Add group	—	groupadd devs
groupdel	Delete group	—	groupdel devs
groups	Show user's groups	—	groups alice
id	Show UID, GID	—	id alice
chage	Set password aging	-l, -M, -E	chage -M 90 -E 2025-12-31 bob

3. Permissions and Ownership

Command	Description	Common Options	Example
chmod	Change permissions	+x, 755, u+rwx	chmod 755 script.sh
chown	Change ownership	user:group	chown bob:devs file.txt
chgrp	Change group	—	chgrp devs file.txt
umask	Set default permissions	—	umask 022
getfacl	View ACL	—	getfacl file.txt
setfacl	Set ACL	-m, -x, -b	setfacl -m u:bob:rw file.txt
lsattr	List file attributes	—	lsattr file.txt
chattr	Change file attributes	+i, -i (immutable)	chattr +i config.cfg

4. Process and System Management

Command	Description	Common Options	Example
ps	List processes	aux, -ef	`ps aux
top	Real-time process monitor	-u (user)	top -u bob
htop	Interactive top	(Install separately)	htop
kill	Send signal to PID	-9 (SIGKILL)	kill -9 1234

killall	Kill by name	—	killall firefox
renice	Change priority	-n, -p	renice -n 10 -p 2345
nice	Start process with priority	-n	nice -n 19 backup.sh
uptime	Show load average	—	uptime
free	Show RAM/swap usage	-h	free -h
vmstat	Show memory, CPU stats	—	vmstat 2 5
dmesg	Show kernel logs	`	tail`

5. Scheduling Jobs

Command	Description	Common Options	Example
crontab	Manage user cron jobs	-e, -l, -r	crontab -e
at	One-time job scheduling	—	`echo "reboot"
systemctl	Manage system services	start, stop, enable, status	systemctl restart sshd
journalctl	View logs	-u, -b, -xe	journalctl -u nginx

6. Package Management

Debian-based:

Command	Description	Common Options	Example
apt update	Update repo index	—	apt update
apt upgrade	Upgrade packages	—	apt upgrade
apt install	Install package	—	apt install nginx
dpkg	Install/remove DEB files	-i, -r, -l	dpkg -i package.deb

RHEL-based:

Command	Description	Common Options	Example
dnf / yum	Package manager	install, remove, update	dnf install httpd
rpm	Manage RPMs directly	-qa, -ivh, -e	` rpm -qa

7. Network Troubleshooting

Command	Description	Common Options	Example
ip addr	Show IPs/interfaces	—	ip addr show eth0
ip link	Show NICs	—	ip link show
ip route	Show routing table	—	ip route
ping	Test host reachability	-c (count)	ping -c 4 google.com
traceroute	Show route path	—	traceroute google.com
ss	Show sockets	-tuln	ss -tulnp
netstat	Legacy socket tool	-an, -tulpn	netstat -tulnp

ethtool	NIC diagnostics	—	ethtool eth0
dig	DNS query	+short	dig +short google.com
host	DNS resolution	—	host linux.org
nmcli	NetworkManager CLI	con, device	nmcli device show

8. Firewall Management

Tool	Command	Description
ufw	enable, allow, status	ufw allow 22/tcp
firewall-cmd	--list-all, --add-service	firewall-cmd --add-service=ssh --permanent
iptables	-L, -A, -P, -D	iptables -A INPUT -p tcp --dport 22 -j ACCEPT
nft	list ruleset	nft list ruleset

9. SELinux and AppArmor

Command	Description	Example
getenforce	Show SELinux mode	getenforce
setenforce	Set mode (0=permissive, 1=enforcing)	setenforce 1
sestatus	View SELinux status	sestatus
ls -Z	Show SELinux contexts	ls -Z /var/www/html
chcon	Change context	chcon -t httpd_sys_content_t index.html

restorecon	Reset context	restorecon -Rv /var/www
setsebool	Set boolean	setsebool -P httpd_can_network_connect on
aa-status	AppArmor status	aa-status
aa-enforce	Enforce AppArmor profile	aa-enforce /etc/apparmor.d/usr.sbin.nginx

10. Containers and Automation

Command	Description	Example
docker run	Run container	docker run -it ubuntu bash
docker ps	List containers	docker ps -a
docker logs	View logs	docker logs webapp
docker exec	Run cmd inside container	docker exec -it nginx bash
docker pull	Download image	docker pull alpine
docker build	Build image	docker build -t myapp .
docker rm	Remove container	docker rm web1
docker rmi	Remove image	docker rmi alpine
git	Version control	git add ., git commit -m "msg", git push
terraform	IaC provisioning	terraform plan, terraform apply
ansible	Configuration mgmt	ansible-playbook site.yml
cloud-init	Cloud VM setup	in user-data config