Linux Administration: Day 1

> Friendly, compact, and action-oriented. Includes quick visuals, fixed typos from the source slides, and DevOps-grade scenarios for each concept.

---

0) What you'll learn (fast)

- What Linux is and why it wins in DevOps.
- How the filesystem is laid out (with a visual tree).
- Must-know commands for files, directories, and viewing text.
- Vim/vi insert & navigation essentials (with tiny mnemonics).
- Copy/move/delete safely.
- Using `man` like a pro.
- DevOps scenarios after each section so you know when it matters.

---

1) What is Linux?

Linux is an open-source operating system kernel. Most of the time you'll use it as a **Linux distribution** (kernel + userland + package manager). Open source, stable, secure, and scriptable—perfect for CI/CD, containers, and cloud.

DevOps When-It-Matters

- **Build runners & containers**: Most CI agents and containers run Linux.
- **IaC**: Terraform/Ansible pipelines commonly target Linux hosts.

---

2) Linux distributions (short & useful)

- **Debian family**: Debian (stable), *Ubuntu* (popular, beginner-friendly), *Kali* (security).
- **Red Hat family**: *RHEL*, *Rocky Linux* (RHEL-compatible), *Fedora* (fast-moving).
- **SUSE family**: *SUSE Linux Enterprise*, *openSUSE*.
- **Desktop-friendly**: Linux *Mint* (not "Minit").
- **Unix (not Linux) examples**: Oracle Solaris, IBM **AIX** (not "HIX"), HP-UX.

> Tip: Choose **Ubuntu LTS** or **Rocky Linux** for servers unless you have a reason not to.

DevOps When-It-Matters

- **Prod vs. Lab**: Use LTS/enterprise clones for prod; Fedora/openSUSE Tumbleweed for labs.

---

3) Components: kernel, shell, terminal (clear mental model)

- **Kernel** → talks to hardware, schedules processes, controls memory.
- **Shell** (bash/zsh/fish) → the command interpreter.
- **Terminal** → the app window that hosts your shell. GUI is **not** a "default shell"; it's a desktop environment.

DevOps When-It-Matters

- Kernel version affects container features (cgroups/ebpf).
- Shell choice impacts developer ergonomics and script portability.

---

4) Filesystem visual (FHS)

```
/
├── bin/        # user commands
├── sbin/       # admin commands
├── etc/        # system config (passwd, group, shadow)
├── var/        # logs, spools, caches (e.g., /var/log)
├── usr/        # userland apps: /usr/bin, /usr/sbin
├── dev/        # devices
├── tmp/        # temp files
├── home/       # user homes (/home/$USER)
└── root/       # root user's home
```

DevOps When-It-Matters

- **Logs** in `/var/log` are gold during incidents.
- **Binaries** in `/usr/bin` vs `/usr/local/bin` decide override precedence.

---

5) Paths & navigation

- **Absolute** path: starts with `/` → `/home/abdulrahman/projects`.
- **Relative** path: from where you stand → `../docs`.
- `.` = current dir, `..` = parent.

Quick nav

```
pwd          # where am I
cd ~         # to home
```

```
cd -          # back to previous dir
ls -lah       # human, all, long
```

DevOps When-It-Matters

- CI steps often rely on correct working dir; prefer **absolute paths** in scripts.

---

6) Command syntax (case & spacing matter)

```
command -options arguments
# Example
ls -l /etc
```

- Case-sensitive. `File` ≠ `file`.
- Spaces separate tokens; quote paths with spaces: `"My File"`.

DevOps When-It-Matters

- YAML + shell quoting issues can break pipelines; always test locally.

---

7) Basic system commands (one-liners you'll actually use)

```
uname -a          # kernel & host info
whoami            # current user
date "+%F %T"     # ISO date time
cal 2025          # calendar
```

DevOps When-It-Matters

- Quick host inventory in incident bridges; paste `uname -a` to share kernel.

---

8) Directories (create, list, inspect)

```
pwd               # print working dir
ls -l             # long list
ls -la            # include hidden
ls -ld dir/       # list directory itself (not contents)
mkdir newdir      # create dir
mkdir -p a/b/c    # create parents
```

Notes fixed

- `dir --color` exists but `ls --color=auto` is more common.

DevOps When-It-Matters

- Creating artifact dirs (`build/`, `dist/`) in CI and ensuring parents exist.

---

9) Files (create & view quickly)

```
touch file.txt           # create empty
cat file.txt             # print whole file
head -n 5 file.txt       # first 5 lines
tail -n 10 file.txt      # last 10 lines
tail -f /var/log/syslog  # follow logs
wc -l file.txt           # count lines
```

DevOps When-It-Matters

- `tail -f` on app logs during rollouts; `wc -l` to measure bulk operations.

---

10) Copy / Move / Delete (safe patterns)

```
# Copy
cp source.txt dest/
cp -r dir/ backup/        # recursive directory copy
cp -i file.txt dest/      # prompt before overwrite (safe)

# Move / rename
mv oldname newname
mv file.txt dir/

# Remove
rm file.txt               # delete file
rmdir emptydir            # remove empty dir only
rm -r dir/                # recursive delete
rm -rf dir/               # force recursive delete (DANGER)

# Remote copy
scp file user@host:/path/
```

Corrections

- Typo fixed: `destinamtion_dir` → `destination_dir`.

DevOps When-It-Matters

- Use `cp -a` to preserve perms/owners when baking AMIs or layering Docker contexts.

---

11) Viewing text (choose the right tool)

```
cat file        # fastest single screen
more file       # page forward only
less file       # page up/down, search with /pattern
head -n 20 f    # top N lines
tail -n 50 f    # last N lines
```

Rule of thumb: use `less` for anything longer than one screen.

DevOps When-It-Matters

- Pipeline logs are huge → `less +F logfile` to follow then break back to browsing with `Ctrl-C`.

---

12) vi/vim essentials (with mnemonic)

Insert & append

- `i` = **insert** before cursor
- `a` = **append** after cursor
- `o` = **open** new line below
- `I` = insert at **line start**
- `A` = append at **line end**
- `O` = open new line **above**

Move

- `h j k l` = left, down, up, right
- `0` = start of line, `$` = end of line
- `G` = last line, `1G` = first line, `nG` = line n
- `e` = end of word

Search

- `/word` forward, `?word` backward, `n` / `N` next/prev

Delete/Copy/Paste

- `dd` delete line, `yy` yank line, `p` paste

Save/Quit

- `:w` save, `:q` quit, `:wq` save & quit, `:q!` discard

  One-step edit at end of line: press `A` (jump to EOL and enter insert).

DevOps When-It-Matters

- SSH into pods/nodes and quickly patch configs under `/etc` or manifests.

---

13) Manual pages (do real discovery)

```
man cat                 # open manual for cat
man -f passwd           # where does “passwd” exist (sections)?
man -k calendar         # keyword search (fixed spelling)
# Open all matching man pages in order:
man -a passwd           # e.g., section 1 (command), 5 (file format)
# Specific section:
man 5 passwd            # (equivalently: man -s 5 passwd)
```

Corrections

- “calender” → **calendar**.
- Shown both `man 5 passwd` and `man -s 5 passwd` forms.

DevOps When-It-Matters

- Reading **file format** man pages (section 5) saves hours when templating config files.

---

14) Terminal shortcuts (speed matters)

- `TAB` completion
- `\` for line-continuation in long commands
- `Ctrl-C` cancel, `Ctrl-Z` suspend, `fg` resume
- `!!` re-run last command; `!ssh` = last command starting with `ssh`

DevOps When-It-Matters

- During outages, speed + precision is everything—lean on history and completion.

---

15) DevOps mini-playbooks (practice bites)

A) Inspect a host in 30 seconds

```
uname -a
lsb_release -a || cat /etc/os-release
uptime; free -h; df -h
ip -br a; ss -tulnp | head
sudo journalctl -p 3 -xn --no-pager
```

B) Triage disk pressure

```
df -h | sort -k5 -h
sudo du -xh /var | sort -h | tail -n 20
sudo journalctl --vacuum-time=7d
```

C) Ship a quick artifact

```
tar -czf app.tar.gz dist/
scp app.tar.gz user@server:/srv/apps/
```

D) Safe delete pattern (audit first)

```
find /var/log -type f -name "*.gz" -mtime +30 -print   # see what will be
deleted
# then
find /var/log -type f -name "*.gz" -mtime +30 -delete
```

---

16) Common pitfalls fixed from slides

- IBM **AIX** (not HIX).
- **Linux Mint** (not Minit).
- "Bourne" shell spelling (not Bourn).
- GUI is **not** a shell; the shell runs inside a terminal.
- `man -k calendar` spelling.
- `man 5 passwd` (a correct way to target section 5).
- SCP placeholder typos fixed.

---

17) 60-second quiz (teach it back)

1. What's the difference between **absolute** and **relative** paths? Give an example of each.
2. Which directory holds system logs, and how do you follow them live?
3. In vim, what's the difference between `a` and `A`?
4. When would you use `rmdir` vs `rm -r`?
5. What does `man -k` do in your own words?

---

18) One-page command wall (tear-off)

```
# Navigation
pwd; cd ~; cd -; ls -lah

# Files/Dirs
mkdir -p a/b; touch f; cp -r dir/ backup/; mv old new; rm -r dir/

# View
cat f; less f; head -n 5 f; tail -f /var/log/syslog

# Vim
i a o | I A O | dd yy p | :w :q :q!

# Docs
man ls; man -k calendar; man 5 passwd
```

Final note

If you practice the **mini-playbooks** daily and answer the quiz out loud, these commands will stick—so when a deploy breaks at 2AM, your fingers already know what to do.