# Linux for Web Developers — Cross?Distro Foundations (Course TOC) ?

> I'd like you to design a course for me that teaches the basics in a structured and progressive way. So, I'd like to start by laying the groundwork for using the most popular Linux distributions (without committing to any specific one), so that I'll find it easier to navigate later on when I take a closer look at Debian, for example. First, I'd like you to create the course table of contents, including chapters and subchapters. I'll then write to you, for example, "k3.2" (meaning "Execute: Chapter 3, Subchapter 2") to have you output the specific course content.

---

This course builds **portable Linux skills** first (works across Debian/Ubuntu/Rocky/Alma), then introduces the **"translation layer"** (package managers, security frameworks, distro conventions) so you can later specialize confidently.

> **How to use it:** Send `kX.Y` (e.g., `k3.2`) and I'll output that lesson's content.
> If you send only `kX`, I'll summarize the whole chapter + provide a mini roadmap.

---

# 1) Orientation: What "Linux" Means Across Distros

1.1 **Distributions & families**

- Debian/Ubuntu family vs RHEL family (Rocky/Alma)
- What is shared: kernel, GNU userland, filesystem standards

1.2 **What differs (and why it matters)**

- Package managers, release cadence, defaults, security frameworks

- “Server distro” vs “desktop distro” in practice

1.3 **Your working environment**

- Local machine vs VPS vs cloud instances
- Terminal options (native, SSH, VS Code remote)

1.4 **Safety & learning workflow**

- Using VMs/containers/snapshots
- “Break it, fix it” habits and how to avoid data loss

---

# 2) The Command Line Essentials (Portable Skills)

2.1 **Shell fundamentals**

- Prompts, commands, arguments, quoting, exit codes

2.2 **Navigation & working with files**

- `pwd`, `ls`, `cd`, `tree`, `touch`, `cp`, `mv`, `rm`, globbing

2.3 **Reading files and manuals**

- `cat`, `less`, `head`, `tail`, `man`, `--help`

2.4 **Pipes & redirection**

- `|`, `>`, `>>`, `2>`, `&>`, `tee`

2.5 **Text searching & filtering**

- `grep` basics, `cut`, `sort`, `uniq`, `wc`, `xargs`

2.6 **Editing in the terminal (minimal but sufficient)**

- Nano or Vim “survival kit”
- Editing config files safely

---

# 3) Filesystem, Permissions, and Ownership

3.1 **Linux filesystem layout (FHS)**

- `/etc`, `/var`, `/home`, `/usr`, `/tmp`, `/opt`
- What you're likely to touch as a web dev

3.2 **Permissions model**

- Users/groups/other, `rwx`, directories vs files

3.3 **Changing permissions and ownership**

- `chmod`, `chown`, `chgrp` (+ common patterns)

3.4 **Special bits & defaults**

- `umask`, setuid/setgid, sticky bit (when it matters)

3.5 **Links**

- Hard links vs symlinks, practical use cases

3.6 **Finding things**

- `find`, `locate`, basic file discovery patterns

---

# 4) Users, Groups, and Secure Access (SSH)

4.1 **Users & groups**

- Creating users, group membership, least privilege

4.2 **sudo and privilege elevation**

- Safe admin workflow, editing sudoers correctly

4.3 **SSH basics**

- `ssh`, `scp`, `sftp`, host keys, known_hosts

4.4 **SSH keys**

- Creating keys, `authorized_keys`, passphrases, agents

4.5 **Hardening the basics**

- Disabling password auth (when appropriate)
- Allowlist users, non-default port (pros/cons)

---

# 5) Processes, Services, and Systemd (Modern Linux Reality)

5.1 **Processes & jobs**

- `ps`, `top/htop`, foreground/background, `nohup`

5.2 **Signals and stopping things**

- `kill`, `pkill`, SIGTERM vs SIGKILL

5.3 **systemd essentials**

- Units, `systemctl status/start/stop/enable`

5.4 **Logs with journald**

- `journalctl` filters, service logs, boot logs

5.5 **Scheduling**

- `cron` and timers: when to use which

---

# 6) Networking Fundamentals for Developers

6.1 **IP, DNS, ports (practical mental model)**

- Local vs public IP, NAT, resolving names

6.2 **Inspecting network state**

- `ip a`, `ip r`, `ss -tulpn`, `ping`, `traceroute`

6.3 **Testing HTTP(S)**

- `curl` essentials, headers, redirects, TLS basics

6.4 **Firewalls (concepts first)**

- Inbound vs outbound rules, “default deny” thinking

6.5 **Distro tooling overview**

- UFW (common on Ubuntu) vs firewalld (common on RHEL-family)

---

# 7) Package Management (Cross?Distro Translation Layer)

7.1 **How packages work**

- Repos, signing, updates, dependencies

7.2 **Debian/Ubuntu: APT**

- install/remove/search, update/upgrade, holding packages

7.3 **Rocky/Alma: DNF**

- install/remove/search, update, modules/streams (overview)

7.4 **Building a “package vocabulary”**

- Mapping common packages: Nginx, PHP, MySQL/MariaDB, Node.js, build tools

7.5 **Alternative installers (use carefully)**

- Snap/Flatpak (conceptual), vendor scripts, language package managers

---

# 8) Storage & Disk Basics (Enough to Not Get Surprised)

8.1 **Disk usage and planning**

- `df -h`, `du -sh`, what fills servers

8.2 **Mounts and filesystems**

- `mount`, `/etc/fstab` (conceptual), removable volumes

8.3 **Archives and compression**

- `tar`, `gzip`, `xz`, zip/unzip for real workflows

8.4 **Backups (foundational habits)**

- What to back up for websites/apps
- Simple rotation thinking

---

# 9) Web Stack Foundations (WordPress ?Relevant, Distro?Neutral)

9.1 **How the request flows**

- DNS → server → reverse proxy → app/PHP → DB

9.2 **Web servers**

- Nginx vs Apache: roles and typical setups

9.3 **PHP runtime**

- PHP-FPM concept, pools, common config locations (pattern-based)

9.4 **Database basics**

- MariaDB/MySQL basics, users, permissions, backups (high level)

9.5 **TLS with Let’s Encrypt**

- Certificates, renewal, the moving parts

9.6 **The “where are configs/logs?” mental model**

- Finding service configs and logs regardless of distro

---

# 10) Security Basics You’ll Actually Use

10.1 **Updates and security posture**

- Patch habits, unattended upgrades concept

10.2 **File permissions for web roots**

- Common WordPress pitfalls and safe defaults (principles first)

10.3 **Firewall baseline**

- Expose only what you need, SSH + HTTP/HTTPS

10.4 **SELinux vs AppArmor (intro)**

- What they are, how they block you, how to diagnose safely

10.5 **Secrets**

- `.env`, config files, permissions, avoiding leaks

---

# 11) Troubleshooting & Observability (How to Fix Things Calmly)

11.1 **A universal debugging checklist**

- “Is it running? is it listening? is it reachable? are logs clean?”

11.2 **Reading logs effectively**

- Nginx/Apache, PHP-FPM, database, system logs

11.3 **Common web server errors**

- 403/404/500, bad gateway, redirect loops

11.4 **Resource issues**

- CPU/RAM/disk exhaustion, swap overview

11.5 **Rollback & recovery**

- Backups, config diffing, safe change management

---

# 12) Developer Workflow on Linux (Node/React?Ready)

12.1 **Toolchain basics**

- `git`, SSH keys for Git hosts, build essentials

12.2 **Node.js installation approaches**

- Distro packages vs NodeSource vs `nvm` (tradeoffs)

12.3 **Process management overview**

- Running apps, env vars, ports, reverse proxy patterns

12.4 **Containers as a portability tool**

- Docker mental model, why it reduces distro lock-in

12.5 **Project deployment patterns**

- Static builds, SSR apps, API services (high-level map)

---

# Optional “Track” Appendices (We can add later)

- A1) Debian deep dive (paths, defaults, common admin tasks)
- A2) Rocky/Alma deep dive (SELinux basics, firewalld, dnf modules)
- A3) Ubuntu LTS deep dive (UFW/AppArmor, common cloud defaults)

---

If you want, tell me your preferred learning mode—**hands-on labs**, **short theory + commands**, or **project-based**—and I’ll shape each `kX.Y` lesson accordingly.

---

Revision #2
Created 2026-04-04 22:15:10 UTC by art10m
Updated 2026-04-04 23:05:07 UTC by art10m