2-Day Ubuntu Developer Training Program

Goal:
By the end of this training, trainees will:

- Understand how Ubuntu works and how it differs from Windows.
- Comfortably use the terminal for real-world development.
- Install and manage developer tools (Node.js, Git, Docker, VS Code).
- Automate common tasks and follow best practices used in professional teams.

Day 1 — Ubuntu Basics & Command-Line Fundamentals

What to Learn (Why it Matters)

Ubuntu is the foundation of most production servers and developer systems.
Mastering the terminal builds speed, confidence, and troubleshooting skills.

Core Concepts

- **What is Linux/Ubuntu** — Open-source, secure, used in 90% of backend systems.
- **File System Structure** — Understand `/home`, `/etc`, `/usr`, `/var`.
- **Navigation Commands** — `cd`, `pwd`, `ls` to move and explore.
- **File Operations** — Create, delete, move, and rename files with `mkdir`, `rm`, `mv`.
- **View and Edit Files** — Use `cat`, `less`, `nano`, or `vim` for quick editing.
- **Permissions & Ownership** — Learn `chmod`, `chown`, and the meaning of `rwx`.
- **Users & Groups** — Add new users, give `sudo` access, manage roles.
- **Package Management** — Install or update software using `apt`.
- **System Monitoring** — Use `top`, `htop`, `df -h`, `free -m` to check CPU, memory, disk.

Task — Command-Line Confidence

1. Create a folder named `my_first_project`.
2. Inside it, create a file `info.txt` using `touch` and write your system info using `echo` + `uname -a`.

3. Change file permission to executable (`chmod +x`).
4. Move it to `/tmp`, view it using `cat`, and bring it back.
5. Take a screenshot of your terminal history.

Deliverable: Screenshot/log showing commands used and their output.

Day 2 — Developer Environment Setup & Real-World Tools

What to Learn (Why it Matters)

A modern developer needs Node.js, Git, Docker, databases, and editors ready on Ubuntu.
You'll learn to set up everything exactly like real projects.

Developer Setup Checklist

- **Update System:** `sudo apt update && sudo apt upgrade -y`
- **Install Git:** Version control with `sudo apt install git`.
- **Install Node.js + npm:** For backend dev; verify with `node -v`.
- **Install Database:** `sudo apt install mysql-server` or `mongodb`.
- **Install Docker:** `sudo apt install docker.io`; enable via `systemctl`.
- **Install VS Code:** Download `.deb` and install via `dpkg -i`.
- **Install Curl + Wget:** For fetching APIs and scripts.
- **Generate SSH Keys:** `ssh-keygen -t rsa -b 4096 -C "email@example.com"`.
- **Clean System:** `sudo apt autoremove`, `sudo apt clean`.
- **Manage Processes:** `ps aux`, `kill -9 <pid>`, `systemctl status`.

Task — Developer Workstation Setup

1. Install Node.js, npm, Git, and VS Code.

2. Clone a public GitHub repo of your choice.
3. Run a small `index.js` script that prints OS and user info.
4. Commit and push it back to GitHub.

Deliverable: Working local environment with one committed script.

Capstone Project — System Readiness Report

- Create a document or terminal output showing:
  - System info (`uname -a`, `lscpu`, `free -m`)
  - Installed tool versions (Node, Git, Docker)
  - Successful SSH test with GitHub
  - Screenshot: VS Code + Terminal + Git log

Bonus Topics for Advanced Developers

Shell & Productivity

- Create **aliases** for frequent commands (`alias gs='git status'`).
- Use **pipes & redirection** (`grep`, `>`, `>>`, `|`).
- Set **environment variables** (`export NODE_ENV=dev`).
- Automate with **bash scripts** (`#!/bin/bash`, `chmod +x deploy.sh`).

Networking & Connectivity

- Test with `ping google.com`, fetch APIs with `curl`.
- Check open ports with `ss -lntp`.
- Secure ports using `ufw allow 3000`.
- Edit `/etc/hosts` for local domains.

Compression & Archiving

- Zip and unzip files (`zip -r`, `unzip`).
- Archive projects using `tar -cvf` and `tar -xvf`.

Security Essentials

- Manage users (`adduser`, `usermod -aG sudo`).
- Connect via SSH (`ssh user@server`)
- Understand permissions (`chmod 644`, `chown root:root`).

System Monitoring & Logs

- Monitor with `htop`, `ps aux`, `df -h`.
- Follow logs live: `sudo tail -f /var/log/syslog`.
- Identify disk usage: `du -sh *`, `ncdu`.

Dev Environment Extras

- Install **Oh My Zsh** for a better terminal experience.
- Use **NVM** to manage multiple Node versions.
- Install **Docker Compose** for multi-service apps.
- Use **Snap** to install GUI tools easily (`snap install postman`).

Optional Final Challenge — Setup Automation Script

Create a `setup.sh` that automatically:

- Updates the system
- Installs Git, Node.js, Docker, VS Code
- Creates a `projects` folder
- Generates an SSH key
- Prints system summary

Goal: One-click system setup script for any new machine.