Faysal Aziz

Terraform Fundamentals

From zero to confident Terraform user—on real Linux, not sandboxes. - 2 days

FA-0441 · 2 days

DevOps, Cloud & Infrastructure

Introduction

Terraform has become the default language operations teams use to describe and automate infrastructure. The tool has matured quickly—recent releases stabilized testing, refactoring, and import workflows—and the ecosystem now expects engineers who can treat infrastructure like high-quality software. In this two-day, hands-on course you'll work on **real Linux servers** (your own VMs/bare-metal, not a browser sandbox) to master the core Terraform workflow, state management, modules, and the practices that keep production safe.

The instructor brings **30+ years of industry experience** and teaches **what teams actually do at work**, not academic exercises. We'll align with the latest stable Terraform (v1.12 at the time of writing) and highlight important changes such as native S3 state locking, the built-in test framework, and config-driven import and refactoring.

Learning outcomes

By the end, you will be able to:

- Install and run Terraform on Linux, manage provider versions, and use the CLI safely.
- Write composable HCL for variables, locals, outputs, and data sources.
- Execute the core workflow (init → plan → apply → destroy) with confidence.
- Manage state locally and remotely; enable **native S3 state locking** and versioning.
- Refactor code safely with **moved blocks**; import unmanaged resources with **import blocks**.
- Create and consume modules (local, VCS, and Registry) with version pinning.
- Validate and test modules using **terraform test** and input validations.
- Apply Linux-first workflows: editors, shell tools, Git hygiene, makefiles, and CI hooks.

Prerequisites

- Comfortable with Linux shell (ssh, systemd, file permissions).
- Basic networking/IAM concepts.
- Git basics.
- A Linux server/VM you can SSH into (e.g., Ubuntu/RHEL) with outbound internet.

Training outline

1. Environment & workflow foundations

- Install Terraform on Linux via HashiCorp repos; verify PATH and signatures.
- Project scaffolding: standard repo layout, .tool-versions/tfenv (optional), terraform version pinning in required_version.
- Editor & linting: VS Code + Terraform extension, terraform fmt and validate.
- Makefile & pre-commit hooks for repeatable local automation.

2. Providers & versions

- Provider discovery & trust model; required_providers, terraform.lock.hcl.
- Authentication patterns (env vars, profiles, CLI helpers); secure secrets handling.
- Hands-on: pin and upgrade a provider; inspect the lock file changes.

3. Authoring HCL like software

- Inputs/outputs, locals, expressions, functions, for-each vs count, dynamic blocks (when and when not).
- Data sources and resource graph; implicit vs explicit dependencies.

4. Core CLI workflow, start to finish

- init, plan, apply, refresh, taint deprecation context, destroy.
- Planning discipline: targeted plans/apply and why to avoid them in teams.
- Drift awareness and safe rollbacks.

5. State deep dive

- What's in state; sensitive data risks; CLI inspection.

- Remote state: **S3 backend with native lockfile** (use_lockfile = true) and bucket versioning; compare to older DynamoDB locking (now deprecated).
- Referencing remote state via terraform_remote_state for cross-stack outputs.
- Backup/restore drills; state encryption & access controls.

6. Modules for reuse

- Local modules vs VCS vs Public Registry; semantic versioning; inputs/outputs.
- Hands-on: extract resources into a module, publish to a Git repo, consume with a version constraint.

7. Refactoring without outages

- Safe renames and moves using **moved { from … to … }**; release notes for module consumers.
- Live exercise: refactor a resource into a child module with a moved block and verify no destroy/create.

8. Import & brownfield adoption

- Config-driven **import blocks** (since v1.5): author import { … } + companion resource and run plan/apply.
- Multi-import with for_each; documenting imports for future maintainers.

9. Validations & testing

- Variable validations; cross-object checks introduced in v1.9 (why this reduces bad inputs).
- **terraform test**: test file structure, assertions, ephemeral resources; using it locally on Linux.

10. Linux-first practices

- SSH keys, file permissions for state workdirs, sudo pitfalls.
- Using system packages, curl + gpg, and network egress controls in regulated environments.

© 2026 Faysal Aziz · Kuala Lumpur