CLOUD, DEVOPS, AND CONTAINERS

Infrastructure as Code with Terraform

Level: Practitioner • 2 days (expandable to 3) • Virtual, In-person

Overview

Infrastructure built by hand is infrastructure nobody can reproduce: environments drift, knowledge lives in one person's head, and every change is a small act of courage. Infrastructure as code fixes this by making environments declarative, versioned, and reviewable, and Terraform is the tool the industry has largely standardized on for doing it across providers.

This is a hands-on, practitioner course. Rather than tour every Terraform feature and provider, it builds depth in the workflow that matters: writing clear configuration, understanding state (where most Terraform pain actually lives), structuring code with modules, and collaborating safely as a team. The gradient runs from why infrastructure as code exists, through the core write-plan-apply loop, to state, modules, and production-grade team workflows, and it lays a strong base for the HashiCorp Terraform Associate certification. Every module includes a lab against a real cloud provider, and each module builds on the one before.

Who Should Attend

- DevOps and platform engineers automating infrastructure provisioning
- Developers who need to define the infrastructure their applications run on
- Cloud engineers preparing for the HashiCorp Terraform Associate certification

Prerequisites

- Working familiarity with at least one cloud provider (Azure or AWS); true beginners should take *Cloud Computing Essentials* first
- Comfort at a command line and with git
- No prior Terraform experience required

What You Will Learn

- Explain what infrastructure as code changes and why declarative beats imperative for infrastructure
- Write clear Terraform configuration with providers, resources, and data sources
- Explain what state is, why it exists, and how to manage it safely with remote backends
- Structure configurations with variables, outputs, and reusable modules
- Collaborate on infrastructure through code review, remote state, and pipeline-driven runs
- Apply production practices: secrets handling, drift, and importing existing infrastructure

Course Outline

Day one: the core workflow

- Why Infrastructure as Code
  - Snowflake servers, drift, and the cost of hand-built environments
  - Declarative versus imperative, and where Terraform sits among the tools

  - Lab: provision a first resource with Terraform and destroy it cleanly
- The Write, Plan, Apply Loop
  - HCL: providers, resources, data sources, and dependencies
  - Reading a plan properly before you apply it
  - Lab: build a small multi-resource environment and evolve it through several plan-apply cycles
- State
  - What state is, what is in it, and why Terraform needs it
  - Remote backends, locking, and keeping state safe
  - What goes wrong: drift, state surgery, and honest recovery options
  - Lab: move local state to a remote backend and recover from a simulated state problem

Day two: structure, teamwork, and production

- Variables, Outputs, and Modules
  - Parameterizing configuration without making it unreadable
  - Writing and consuming modules; using the public registry with judgment
  - Lab: refactor the environment into a reusable module consumed by two configurations
- Terraform as a Team
  - Code review for infrastructure: what a good Terraform pull request looks like
  - Environments and workspaces, and their tradeoffs
  - Running Terraform from a pipeline instead of a laptop
  - Lab: run plan and apply through a CI pipeline with review before apply
- Production Practices
  - Secrets and sensitive values done properly
  - Importing existing infrastructure and taming brownfield estates
  - Structuring a growing codebase, and where the Terraform Associate exam fits
  - Lab: import a hand-built resource under Terraform management without breaking it

Extended Version

The three-day version keeps the same gradient and adds:

- Testing Terraform: validation, policy as code, and automated checks
- Multi-environment and multi-account patterns at organization scale
- HCP Terraform and alternatives for remote execution and governance
- A capstone: design, build, and review a complete environment as code, from empty account to running application infrastructure