

Terraform & Infrastructure as Code

Course Duration ~60 Hours · 8 Weeks · Live Online · Lab-Heavy · Real Cloud Resources, Real State, Real Recovery

Curriculum reviewed: July 2026 — the engine landscape is shifting; this syllabus is checked against official documentation

[HCL](#)
[State & Recovery](#)
[Modules](#)
[Policy as Code](#)
[Terraform & OpenTofu](#)
[GitOps](#)

In short: Stop clicking in consoles. Define infrastructure once, review it like code, and rebuild it identically — in a language that outlives whichever engine you run it on. This is the course that makes every other DevOps skill repeatable: containers, clusters, pipelines and cloud accounts all have to be provisioned, versioned and reproduced, and this is how. The durable skill is HCL and infrastructure-as-code practice, not any one engine, and the scarce capability is not writing configuration but managing state, controlling blast radius, and bringing hand-built estates under management without downtime. It also refuses to duck the defining issue of 2026: HashiCorp relicensed Terraform from an open-source licence to the Business Source Licence; the community forked the last openly licensed version as OpenTofu, now under the Linux Foundation; and HashiCorp — and therefore Terraform — is now owned by IBM. We teach the fork as an **evidence-based decision you learn to make, scrupulously neutral**: what's shared (HCL, provider protocol, state format — migration close to drop-in), where OpenTofu leads (native state and plan encryption, extra iteration and targeting options, an openly licensed registry), where Terraform leads (higher-level multi-configuration orchestration, a policy-as-code engine, a mature managed platform), and the one-way doors (enabling OpenTofu state encryption is effectively irreversible). Labs run on either. Across 9 modules you recover a corrupted state file, import an undocumented brownfield estate, gate non-compliant infrastructure in a pull request, and run the same config on both engines with a written recommendation — closing with a complete production-environment capstone. The AI question: AI writes HCL fluently and produces plans that are clean and wrong in expensive ways — destroy-and-recreate instead of update-in-place, open security groups, careless state. The value is judgment about state, blast radius and change safety.

Duration	8 weeks · ~60 hours · live + hands-on	Format	Live online · lab-heavy · real cloud resources, real state, real recovery
Prerequisite	Basic cloud familiarity with at least one provider, and command line comfort. No prior IaC experience required.	Engines	Taught engine-neutral — labs run on Terraform or OpenTofu, and you graduate able to justify the choice
Certification	Aligned to the HashiCorp Terraform Associate path — the skills transfer across engines	Ideal for	Fresh IT graduates, sysadmins provisioning through consoles, and organisations evaluating the engine decision

Terraform or OpenTofu? A Decision Framework, Not an Opinion

We are deliberately neutral — your peers include enterprises with IBM relationships and startups with open-source mandates, and the right answer depends on context. The course gives you the framework to decide with evidence:

- **What's shared:** HCL, the provider protocol and the state format remain largely common, so migration is close to a drop-in change — though syntax is beginning to diverge in both directions.
- **Where OpenTofu leads:** capabilities the Terraform CLI lacks, most notably native state and plan encryption, additional iteration and targeting options, and an openly licensed provider registry under community governance.
- **Where Terraform leads:** higher-level multi-configuration orchestration, a policy-as-code engine, and a mature managed platform with commercial support behind it.
- **The one-way door:** enabling OpenTofu's state encryption is effectively irreversible, because encrypted state cannot be read back by Terraform — you learn which decisions can be reversed and which cannot.
- **The practitioner reality:** for an ordinary internal user the licence change has little day-to-day effect; it matters most to organisations with vendor policies, compliance requirements, or products built on top of the tool.

Why Learn This When AI Writes HCL?

AI writes HCL fluently — and produces configurations that plan cleanly and are wrong in expensive ways: resources destroyed and recreated rather than updated in place, security groups left open, state and secrets handled carelessly, and modules copied from a blog for a different provider version. Infrastructure mistakes cost money and downtime, not a failed build. The value is judgment about state, blast radius and change

safety — reading a plan and knowing what it will actually do. You'll use AI tooling as a reviewed accelerator and learn the failure patterns generated infrastructure code repeats.

Course Highlights

- Every module leads with what you build, operate or recover from — not what you "learn about"
- State treated as the module that actually matters — inspection, surgery, import, recovery
- A scrupulously neutral Terraform-versus-OpenTofu decision framework
- Brownfield import — bring an undocumented hand-built estate under code without downtime
- Policy-as-code gates that block non-compliant infrastructure before it exists
- Secrets that never enter state or version control; dynamic credentials for pipelines
- CI/CD and GitOps for infrastructure — plan on PR, gated apply, drift remediation
- AI tooling as an accelerator you review, never a substitute for reading the plan

Hands-on Projects & Portfolio

- **Full Cloud Environment** — network, compute, database, cluster — built from scratch, torn down and rebuilt identically
- **Reusable Module Library** — versioning, documented interfaces and contract tests
- **State Disaster Lab** — recover from a corrupted state file, a partially applied change and an accidental resource deletion
- **Brownfield Import Exercise** — bring a hand-built, undocumented environment under code management without downtime
- **Refactoring Workshop** — restructure modules and move resources between states with zero destruction
- **Policy-as-Code Gate** — block non-compliant infrastructure in a pull request, with cost estimation attached
- **Infrastructure CI/CD Pipeline** — automated plan on PR, gated apply, and scheduled drift detection
- **Engine Evaluation Exercise** — run the same configuration on both engines and produce a written recommendation for a given context
- **Capstone** — complete production environment: tested, governed, cost-aware and defended

Tools & Stack

Terraform · OpenTofu · HCL · Terragrunt · AWS · Azure · Google Cloud · Kubernetes provider · Helm provider · Checkov · Trivy · tfint · Terratest · Open Policy Agent · Sentinel · Infracost · Vault · Atlantis · Spacelift or env0 · GitHub Actions · Git · Docker

Certification Alignment

- Syllabus mapped to the **HashiCorp Terraform Associate** certification path where relevant
- The credential covers Terraform specifically — but the skills taught here transfer across engines and across cloud providers
- Presented as preparation and alignment — no official partnership or accreditation implied, and no exam outcome guaranteed

Career Outcomes

Infrastructure as code is now the assumed baseline for cloud work rather than a specialism — provisioning by hand is treated as a defect — and HCL skills transfer across cloud providers and across both engines, which makes them unusually portable. The scarce capability is not writing configuration but managing state, controlling blast radius, and bringing existing hand-built estates under management safely. This course targets:

- DevOps Engineer · Platform Engineer · Cloud Engineer · Site Reliability Engineer
- Infrastructure Engineer · Cloud Architect (with experience) · Security engineer working on infrastructure compliance · Backend engineer owning their own environments

Who This Course Is For

- Final-year students and fresh IT graduates targeting DevOps, cloud or platform roles
- **System administrators and infrastructure engineers still provisioning through consoles** — a large group with immediate need
- Developers who want to own their own environments end to end

- Cloud engineers who inherited someone else's configurations and are afraid to run apply
- SREs and platform teams standardising infrastructure across many teams
- **Organisations evaluating the engine decision** who need their team to understand both

Module 1: Infrastructure as Code Foundations

6 Hours

YOU BUILD

- Your first real resource provisioned, changed and destroyed through the plan/apply loop — on both engines — with a written read of what each plan intends to do

TOPICS

- Declarative versus imperative provisioning; the plan/apply model and the reconciliation loop; providers and the registry ecosystem
- The engine landscape and the licensing and governance context — Terraform, OpenTofu, and what the fork means in practice
- Where configuration management and container orchestration fit alongside rather than compete

Assessment: Quiz + Lab

Module 2: HCL in Depth

7 Hours

YOU SHIP

- A configuration that uses for_each, dynamic blocks, validated typed variables and pinned provider versions — readable, DRY, and safe from a surprise version upgrade

TOPICS

- Resources, data sources, variables and outputs; locals; expressions and functions
- Meta-arguments including count, for_each and lifecycle; dynamic blocks; type constraints and validation
- Provider and version constraints, and why unpinned versions are a future outage

Assessment: Quiz + Lab

Module 3: State — the Module That Actually Matters

8 Hours

YOU RECOVER

- The state disaster lab — a corrupted state file, a partially applied change and an accidental deletion recovered; plus a remote backend with locking and existing infrastructure imported under management

TOPICS

- What state is and why everything hinges on it; remote backends and locking; workspaces; sensitive data in state and encryption options
- State inspection and surgery; importing existing infrastructure; refactoring with move operations; splitting and merging state
- Drift detection and reconciliation; recovering from a corrupted or lost state file

Assessment: Quiz + Lab

Module 4: Modules & Composition

7 Hours

YOU SHIP

- A reusable module library — clean interfaces, sensible defaults, versioned and published to a private registry — structured to serve many environments without copy-paste

TOPICS

- Writing reusable modules; interface design and sensible defaults; versioning and publishing; private registries
- Composition patterns; avoiding over-abstraction; dependency management between layers
- Structuring a repository across many environments and teams

Assessment: Quiz + Lab

Module 5: Multi-Environment & Team Workflows

6 Hours

YOU SHIP

- A repository serving dev, staging and production from shared modules with per-environment configuration — blast radius contained, and applies that don't collide with a teammate's

TOPICS

- Environment promotion strategies compared honestly; separating configuration from environment; remote execution
- Orchestrating many configurations; blast radius control and targeted operations
- Code review for infrastructure changes; collaborating without stepping on each other's applies

Assessment: Quiz + Lab**Module 6: Testing, Validation & Policy as Code**

7 Hours

YOU SHIP

- A policy-as-code gate that blocks non-compliant infrastructure before it exists, with security scanning and cost estimation attached to the pull request

TOPICS

- Static analysis and linting; security scanning of configurations; the native test framework and integration testing against real resources
- Contract testing for modules; policy enforcement to block non-compliant infrastructure before it exists
- Compliance-as-code; cost estimation in the pull request

Assessment: Quiz + Lab**Module 7: Secrets, Security & Compliance**

6 Hours

YOU SHIP

- A pipeline that provisions with dynamic, short-lived credentials and least-privilege roles — no secret ever in state or version control, and state encrypted at rest

TOPICS

- Secrets that never enter state or version control; dynamic credentials and workload identity for pipelines
- Least-privilege provisioning roles; provider authentication patterns; encryption at rest for state
- Audit trails; least-privilege infrastructure by default

Assessment: Quiz + Lab**Module 8: CI/CD & GitOps for Infrastructure**

6 Hours

YOU SHIP

- A pull-request-driven pipeline: automated plan as a review artefact, gated apply, scheduled drift detection, and a rehearsed rollback strategy for changes that can't simply be reverted

TOPICS

- Pull-request driven plan and apply; automation platforms and self-hosted runners; approval gates and drift remediation pipelines
- Handling plan output as a review artefact; rollback strategy for infrastructure that cannot simply be reverted
- Integrating with application delivery pipelines

Assessment: Quiz + Lab**Module 9: Real-World Practice & Capstone**

7 Hours

YOU SHIP

- Your capstone: one complete production environment — networking, compute, managed data and Kubernetes — built, tested, governed and cost-aware, with a legacy estate imported and an engine migration rehearsed, and every decision defended

TOPICS

- Provisioning a full cloud environment across networking, compute, managed data services and Kubernetes; provider-specific gotchas
- Multi-cloud and multi-region patterns; importing a legacy hand-built estate under management; migrating between engines; upgrade planning
- Cost optimisation; capstone design, build, test and defense

Assessment: Capstone Project + Review

www.progmatrix.com