

Terraform for Azure and AWS: Multi-Cloud Workshop

A two-day practical introduction with selected advanced deployment patterns

FA-0554 · 2 days

DevOps, Cloud & Infrastructure

Practise Terraform configuration, remote state and selected Azure/AWS deployments, then review event-driven and multi-region patterns.

Introduction

This two-day workshop develops a disciplined Terraform workflow across Azure and AWS. Participants work through HCL, providers, remote state, modules and selected sandbox resource deployments, using a local or approved remote Ubuntu development environment.

The core labs cover manageable infrastructure examples and a small AWS messaging/function path. ECS, data services, cross-region replication and production-operating patterns are demonstrations or design clinics. The workshop does not promise a complete production-grade multi-cloud platform from a clean slate in two days.

Use approved sandbox accounts with appropriate scoped permissions, resource budgets and cleanup procedures. State and credentials require controlled access. Terraform/provider versions and the chosen cloud runtimes are checked and pinned for the training environment.

Learning outcomes

The workshop teaches participants to:

- Write and review HCL with variables, expressions, outputs and reusable modules.
- Use the init-plan-apply workflow and explain controlled resource cleanup.
- Configure remote state with suitable locking, encryption, versioning and access controls.
- Configure Azure/AWS providers and aliases for selected account/region scenarios.

- Provision and inspect a guided Azure VM/network example and a small AWS event-driven stack.
- Evaluate .NET/container, data-service and multi-region patterns through design clinics.
- Review plans, drift, safe replacement and operational handoff requirements before real deployment.

Prerequisites

- Comfortable with basic Linux shell and Git.
- Approved Azure and AWS sandbox accounts with permissions needed for the selected networking/compute/state labs; confirm quotas, budget and cleanup responsibility.
- VS Code installed locally; ability to connect to the provided Ubuntu server via Remote-SSH (we'll validate extension setup together).

Cloud permissions should be scoped to approved training resources rather than unrestricted production administrator access. Participants need compatible Terraform/provider tools and a sandbox where resources can be cleaned up safely.

Training outline

1. Day 1 — Terraform Workflow, Remote State and Azure

Foundations

- What Terraform is and isn't; workflow (init, plan, apply, destroy)
- HCL syntax essentials: variables, locals, outputs, expressions, for-each/count, dynamic blocks

Providers and authentication

- Provider blocks, versions, and constraints
- Multiple provider configurations and alias for multi-region/multi-account patterns
- Credentials patterns: env vars, profiles, service principals, OIDC workload identities

State management

- Local vs remote state; why remote matters
- S3 backend with encryption, versioning and native lockfile-based state locking; discuss legacy DynamoDB locking only for migration/compatibility.
- Azure Storage backend: access controls, locking/recovery behaviour and appropriate data-protection settings.
- State drift, import, and moved blocks; safe refactors

Composition and reuse

- Modules: inputs/outputs, version pinning, registries
- Terragrunt vs pure Terraform modules (pros/cons)

Quality and safety

- fmt, validate, tflint, checkov basics
- Plans in CI; policy as code intro (Sentinel/OPA)

Remote Development Environment

- Remote-SSH setup and troubleshooting
- Folder layout for multi-cloud repos; .terraform.lock.hcl hygiene
- Using the Terraform CLI on the remote box from VS Code terminals
- Secrets handling and environment parity on remote hosts

Azure Compute and Networking

- AzureRM provider configuration and approved authentication, including workload identity or service-principal patterns where appropriate.
- Resource hierarchy: resource groups, tags, naming conventions
- Networking primitives: VNets, subnets, public IPs, NICs, NSGs, route tables

VM resource choices and requirements for the pinned AzureRM provider version.

- Linux VM via azurerm_linux_virtual_machine (managed disks, images, SSH)
- Windows VM via azurerm_windows_virtual_machine (WinRM, passwords, images)
- VM extensions (Custom Script), boot diagnostics, availability sets vs zones
- Managed identities and Key Vault access; attaching data disks
- Reusable module for Azure VM deployments (inputs for size, image, networking)
- Packaging: variables/outputs, depends_on, lifecycle, data sources

2. Day 2 — AWS State, Eventing and Deployment Review

AWS Provider and State Setup

AWS provider configuration, version constraints and supported region patterns.

- Provider configuration, profiles, STS; tagging strategies
- Compare provider aliases and supported resource-level region configuration in the chosen provider version.

S3 remote state with native lockfile locking, encryption and restricted access.

- Bootstrap the state bucket, lockfile permissions and IAM policies; a DynamoDB lock table is a legacy migration option, not the default recommendation.
- State file protections, versioning, and recovery playbook

Selected Messaging and Function Lab

Amazon SNS

- Topics, subscriptions, delivery policies, encryption, access control

Amazon SQS

- Standard vs FIFO, DLQs and redrive policies, visibility timeouts, SSE/KMS

Amazon EventBridge

- Event buses (default/custom), rules, targets, input transformers, archives/replays
- Pipes and Scheduler: routing and time-based invocations

AWS Lambda

- Function packaging, including a .NET example using a currently supported Lambda runtime; execution roles and VPC networking.
- Event source mappings from SQS, DynamoDB Streams, Kinesis

Cross-service wiring patterns

- Fan-out: EventBridge rule → SNS topic → SQS subscriptions
- Queue-to-Lambda with DLQ and retries
- Discuss cross-region event-mirroring designs, service constraints and failure scenarios; replication alone does not guarantee high availability.
- Reusable Terraform module structure for eventing stacks

Configuration and Secrets

AWS Systems Manager Parameter Store

- Hierarchies, types (String, SecureString), KMS, policies and tiers
- Parameter references from Lambda/ECS and Terraform: inspect state exposure and avoid assuming a sensitive flag prevents secret storage.
- Azure Key Vault vs Parameter Store: when to use which in multi-cloud setups
- Terraform resources and data sources: `aws_ssm_parameter`, `data.aws_ssm_parameter`

Repository and Operational Review

- Repo layout: `environments/`, `modules/`, `stacks/`
- Makefile/task runner for common Terraform tasks
- CI plans and gated applies; tagging/chargeback; documentation generation
- Incident and recovery review: planned resource replacement with `-replace`, cautious import/refactoring and retained state recovery evidence; avoid deprecated taint as the normal workflow.

Advanced AWS and Cross-Region Design Clinics

Containers and .NET Services

- ECS fundamentals: clusters, capacity (Fargate vs EC2), task definitions, services
- Networking and discovery: ALB/NLB, target groups, Service Connect/service discovery
- CI/CD handoff: image tags, immutable deployments, blue/green considerations
- Terraform resources & modules: `aws_ecs_cluster`, `aws_ecs_task_definition`, `aws_ecs_service`
- Observability hooks: CloudWatch logs, metrics/alarms, autoscaling policies

Data Services

Amazon RDS

- Subnet groups, parameter/option groups, storage, engine versions, backups
- Blue/green updates and maintenance windows

Amazon DynamoDB

- Table design inputs (PK/SK), GSIs/LSIs, TTL, streams, on-demand vs provisioned
- Global Tables v2 and multi-region replication considerations
- Terraform resources & modules for RDS and DynamoDB; import strategies and drift handling

EC2 and Foundational Compute

- `aws_instance` for quickstarts; AMI lookups; key pairs; user data
- Security groups vs NACLs; IMDSv2; instance profiles
- Patterns for immutable vs in-place changes; lifecycle caveats

Cross-Region Patterns

- Provider aliasing vs region arguments; scoping state/workspaces
- Replication primitives: S3 CRR for artifacts, DynamoDB Global Tables for stateful apps
- EventBridge buses/rules across regions and accounts
- Testing and promotion strategies (per-region plans)