Faysal Aziz

Hands-On Infrastructure Automation with Terraform

From Local Setup to Azure & AWS - Real-World Infrastructure-as-Code in Two Days

FA-0238 · Two Days

DevOps, Cloud & Infrastructure · Automation, RPA & Power Platform

Introduction

This focused two-day course is designed for professionals who want to move beyond the theoretical and get straight into using Terraform for real infrastructure automation. Your instructor brings over 30 years of industry experience, so the emphasis is on practical, demand-driven workflows used in production-not academic exercises. You'll begin with installing and configuring Terraform on a Linux workstation (via remote SSH and VS Code), then run Terraform locally to get comfortable. After that, you'll deploy infrastructure on both Microsoft Azure and Amazon Web Services, giving you a full-lifecycle view of infrastructure-as-code across major cloud platforms. Recent Terraform developments such as improved provider support and new CLI capabilities will be woven into the content.

Learning outcomes

By the end of this course you will be able to:

- Install and configure Terraform on a Linux machine, connect via remote SSH and use VS Code effectively.
- Write, validate, and apply simple Terraform configurations locally (on your workstation) to gain confidence in the workflow.
- Initialize, plan and apply Terraform-managed infrastructure on Azure: create a resource group, virtual network, subnet, and a VM.
- Do the same on AWS: set up a VPC, subnets, security groups, and an EC2 instance via Terraform.

- Understand provider versioning, state management basics, and best practices for modularizing Terraform code.
- Recognize contemporary concerns such as infrastructure drift, tool licensing and governance (for example, new features and provisioning model changes in Terraform 1.x).
- Develop confidence to take these patterns back into your organisation and scale up with cloud infrastructure automation.

Prerequisites

Before attending this course you should have:

- Capability to connect via SSH to remote servers.
- Basic familiarity with the Linux command line is recommended but not mandatory.
- A VS Code installation (or ability to install) and knowledge of basic editing.
- Access credentials (or ability to create credentials) for Azure and AWS (you'll need permissions to create resource groups / VPCs / subnets / VMs).
- An understanding of what cloud resources are (VMs, subnets, security groups) and basic networking concepts (VPC, subnet, firewall).
- Basic understanding of version control (Git) is helpful but not strictly required.

Training outline

1. Introduction to Terraform and the Infrastructure-as-Code mindset

1. What is Terraform: purpose, architecture, HashiCorp Configuration Language (HCL) basics.
2. Overview of recent changes (Terraform 1.x series, provider maturity, governance concerns).
3. Why automate infrastructure: consistency, repeatability, scalability, safety.

2. Installation and Setup on Linux with Remote SSH and VS Code

1. Installing Terraform binary on Linux (from official release).
2. Configuring your Linux environment: path, version check, validation.
3. Using SSH to connect to a remote Linux server (setup, keys, security).
4. Installing and configuring VS Code with Terraform extensions (language support, linting, formatting).
5. Creating a workspace folder, initial Git repository, basic VS Code project layout.

6. Terraform CLI commands: terraform init, terraform fmt, terraform validate, terraform plan, terraform apply, terraform destroy.

3. Hands-On: Terraform Locally (Warm-Up)

1. Create your first Terraform configuration: a local “hello” infrastructure (e.g., create a local file, or if available provision a simple VM in a local sandbox).
2. Define variables, outputs, resources in Terraform.
3. Use terraform plan and apply to validate and execute changes.
4. Update configuration and apply incremental change.
5. Use terraform destroy to clean up.
6. Introduce state file concept: what it is, what to avoid, basic best-practices.
7. Introduce modules at a basic level: writing simple module, calling module from root.
8. Discuss provider blocks, required_providers version pinning, and compatibility.

4. Deploying Infrastructure on Azure with Terraform

1. Overview of Azure provider: azurerm provider, versioning, authentication methods (service principal, environment variables).
2. Setting up Azure credentials: create service principal, assign role, export environment variables.
3. Define Terraform configuration targeting Azure: resource group, virtual network, subnet, network security group, and a VM (Linux).
4. Variables and outputs in the Azure configuration: region, resource group name, VM size, admin SSH key.
5. Using terraform init (with Azure provider), terraform plan and review plan, terraform apply.
6. Validate the deployment via Azure portal/CLI: verify resources exist.
7. Modify the configuration (for example, adding a second subnet or changing VM size) and re-apply.
8. Clean up with terraform destroy.
9. Discussion: state file backends (local vs remote), remote state in Azure (Azure Storage). Introduce concept of state locking and collaboration.

5. Deploying Infrastructure on AWS with Terraform

1. Overview of AWS provider: aws provider, versioning best practices.
2. Setting up AWS credentials: IAM user/role, access key, export environment variables or use shared credentials file.
3. Define Terraform configuration targeting AWS: VPC, one or two subnets, internet gateway, route table, security group, and an EC2 instance (Linux).
4. Variables and outputs: region, instance type, key pair, tags.

5. Running terraform init, terraform plan, review and terraform apply.
6. Validate the AWS deployment: via AWS console or CLI, check VPC, subnets, EC2.
7. Modify configuration: e.g., add an S3 bucket, or autoscaling group basics (if time permits).
8. Introduce Terraform modules for AWS: separate networking module, compute module. Use module calls.
9. Clean up with terraform destroy.
10. Discussion: splitting state, performance considerations at scale (state file size, resource count)

This outline will give participants a fast-paced, hands-on experience using Terraform across local, Azure and AWS environments-guided by a seasoned instructor and anchored in real-world production practices.

© 2026 Faysal Aziz · Kuala Lumpur