Terraform Intro 1-day

© Elephant Scale

April 15, 2025

Overview

- IaC (Infrastructure as Code) is one of the most important developments in application deployment in many years. It means that instead of manually configuring your hardware and software infrastructure, you describe all this configuration as code.
- While every cloud gives you their implementation of it, and while Amazon heavily promotes their version called CloudFormation, Terraform is a maverick who appeared on the scene a few years ago and became more popular than all other IaC implementations. Recently, Google recognized the superiority of Terraform and started using that instead of their tool. This speaks to the importance of learning Terraform.
- Terraform is open source and can be used on any cloud provider, such as Amazon Web Services, IBM Cloud, Google Cloud Platform, DigitalOcean, Linode, Microsoft Azure, Oracle Cloud Infrastructure, VMware vSphere, OpenStack, and many others.
- Terraform is declarative and idempotent. These are good design principles, and many competitors have also implemented Terraform ideas, making Terraform a de facto standard.

What you will learn

- DevOps and IaC
- How to use Terraform
- HashiCorp ecosystem

Audience

- DevOps, Developers, Architects
- Anyone working toward their Terraform Certification

Duration

1 day

Format

Lectures and hands-on labs. (50%, 50%)

Prerequisites

- Some background with Unix or Linux including the command line
- Some knowledge of a programming language such as Java, C#, Python, Node.js, etc.

Lab environment

- A reasonably modern laptop
- Unrestricted connection to the Internet. Laptops with overly restrictive VPNs or firewalls may not work properly
- Chrome browser
  - SSH client for your platform

Lab flavor

- Terraform can be used for most clouds
- Currently available flavors of the labs include
  - Amazon AWS
  - **Azure**
  - Google cloud GCP
  - Oracle cloud OIC

Detailed Outline

- Why Terraform
  - The Rise of DevOps
  - What Is Infrastructure as Code?
  - Ad Hoc Scripts
  - Configuration Management Tools
  - Server Templating Tools
  - Orchestration Tools
  - Provisioning Tools
  - The Benefits of Infrastructure as Code
  - How Terraform Works
  - How Terraform Compares to Other IaC Tools
  - Configuration Management Versus Provisioning
  - Mutable Infrastructure Versus Immutable Infrastructure
  - Procedural Language Versus Declarative Language
  - Master Versus Masterless
  - Agent Versus Agentless
  - Large Community Versus Small Community
  - Mature Versus Cutting Edge
  - Using Multiple Tools Together
  - Conclusion
- Getting Started with Terraform

  - Setting Up Your AWS Account
  - Install Terraform
  - Deploy a Single Server
  - Deploy a Single Web Server
  - Deploy a Configurable Web Server
  - Deploying a Cluster of Web Servers
  - Deploying a Load Balancer
  - Cleanup
  - Conclusion
- How to Manage Terraform State
  - What Is Terraform State?
  - Shared Storage for State Files
  - Limitations with Terraform's Backends
  - Isolating State Files
  - Isolation via Workspaces
  - Isolation via File Layout
  - The terraform_remote_state Data Source
  - Conclusion
- How to Create Reusable Infrastructure with Terraform Modules
  - Module Basics
  - Module Inputs
  - Module Locals
  - Module Outputs
  - Module Gotchas
  - File Paths
  - Inline Blocks
  - Module Versioning
  - Conclusion
- How to Use Terraform as a Team
  - Adopting IaC in Your Team
  - Convince Your Boss
  - Work Incrementally
  - Give Your Team the Time to Learn
  - A Workflow for Deploying Application Code
  - Use Version Control
  - Run the Code Locally
  - Make Code Changes
  - Submit Changes for Review
  - Run Automated Tests

- Putting It All Together
- Conclusion