Learn Infrastructure as Code with Terraform

Robert Jordan

Pearson

About Me

- Over 25 years in the IT industry
- 10 years primary focus in cloud technologies
- Networking and HPC operations background

About this course

- Beginner level
  - No previous Terraform or IaC experience required.
  - Familiarity with command line is recommended.

About this course

- Beginner level
  - No previous Terraform or IaC experience required.
  - Familiarity with command line is recommended.
- Demo based
  - Online format can be hard to follow… Review the recordings.
  - Use devcontainers to set up your environment.

About this course

- Beginner level
  - No previous Terraform or IaC experience required.
  - Familiarity with command line is recommended.
- Demo based
  - Online format can be hard to follow… Review the recordings.
  - Use devcontainers to set up your environment.
- Ask questions!

About this course

- Focused on open source tools.
  - We may touch on some of the Terraform and IaC ecosystem but won’t cover commercial offerings.

About this course

- Focused on open source tools.
  - We may touch on some of the Terraform and IaC ecosystem but won’t cover commercial offerings.
- Certification prep
  - This is **not** a certification prep course however, I will try to point out concepts that are likely to be covered on the exam.

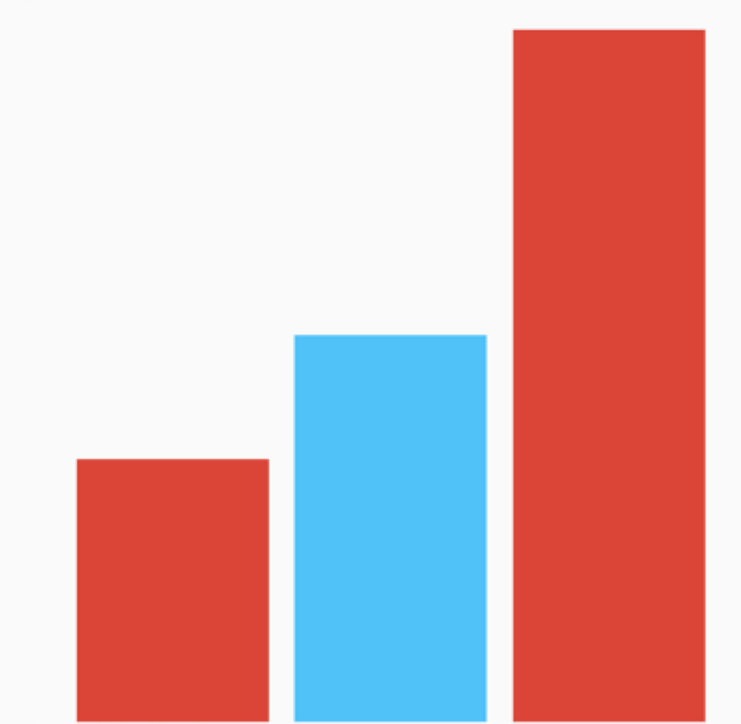
About this course

- Focused on open source tools.
  - We may touch on some of the Terraform and IaC ecosystem but won’t cover commercial offerings.
- Certification prep
  - This is **not** a certification prep course however, I will try to point out concepts that are likely to be covered on the exam.
- Not affiliated with or approved by Hashicorp.

About this course

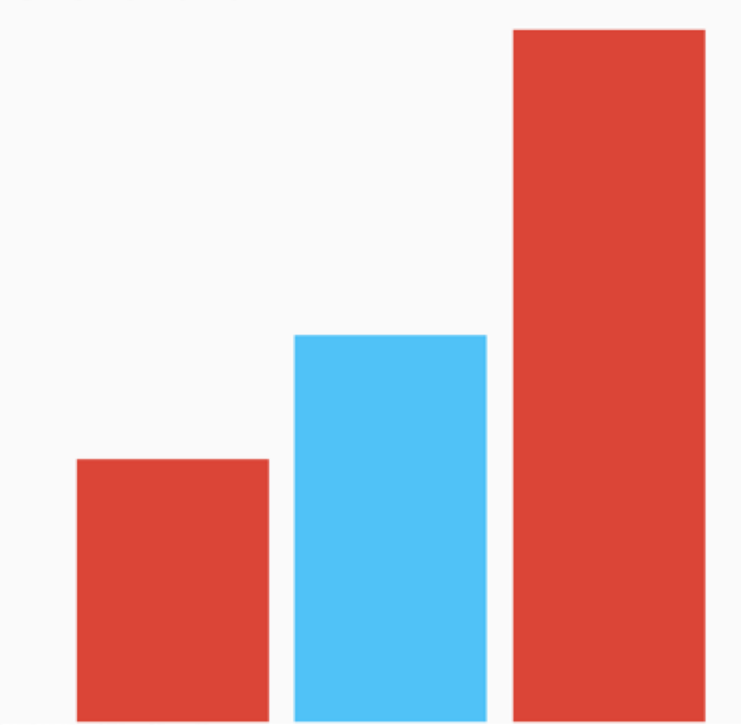
- Materials
  - The example code is in GitHub.
  - There is a link in the resources panel.
  - https://github.com/bananalab/Learn-Infrastructure-as-Code-with-Terraform

Course overview



1. Introduction to IaC
2. Terraform Basics
3. Terraform in Practice

Course overview



1. Introduction to IaC
2. Terraform Basics
3. Terraform in Practice

Introduction to IaC

- What is Infrastructure as Code?

Introduction to IaC

- What is Infrastructure as Code?
  - A software engineering practice that involves managing and provisioning infrastructure resources using machine-readable definition files rather than manually configuring them.

Introduction to IaC

- What is Infrastructure as Code?
  - A software engineering practice that involves managing and provisioning infrastructure resources using machine-readable definition files rather than manually configuring them.
  - Treats infrastructure resources, such as virtual machines, networks, storage, and other components, as code.

Introduction to IaC

- Benefits of IaC
  - Consistency

Introduction to IaC

- Benefits of IaC
  - Consistency
  - Change Management

Introduction to IaC

- Benefits of IaC
  - Consistency
  - Change Management
  - Reproducibility

Introduction to IaC

- Benefits of IaC
  - Consistency
  - Change Management
  - Reproducibility
  - Scalability

Introduction to IaC

- Benefits of IaC
  - Consistency
  - Change Management
  - Reproducibility
  - Scalability
  - Collaborative

Introduction to IaC

Approaches to IaC

Imperative	Declarative

Introduction to IaC

Approaches to IaC

Imperative Instruction	Declarative Instruction
Create a VM	There should be a VM

Approaches to IaC

Imperative Tools	Declarative Tools
Cloud SDKs Cloud CLIs	Terraform CloudFormation Pulumi (Hybrid) CDK (Hybrid)

Introduction to IaC

- Benefits of Declarative IaC
  - Simplicity
    - Focus on desired state. No need to understand **how** desired state is achieved.

Introduction to IaC

- Benefits of Declarative IaC
  - Simplicity
    - Focus on desired state. No need to understand **how** desired state is achieved.
  - Idempotence
    - Code can be applied any number of times without changing the infrastructure.

Introduction to IaC

- Benefits of Declarative IaC
  - Simplicity
    - Focus on desired state. No need to understand **how** desired state is achieved.
  - Idempotence
    - Code can be applied any number of times without changing the infrastructure.
  - Change management
    - Easy to understand changes to desired state.

Introduction to IaC

- Benefits of Declarative IaC
  - Auditability and compliance.
    - Static analysis of desired state can flag compliance issues before application.
  - Single Source of Truth (SSoT).
    - IaC should be authoritative of the desired state of the infrastructure.
    - If infrastructure doesn’t match then apply the code.

Introduction to IaC

- Drawbacks of Declarative IaC
  - Limited flexibility.
    - Some things are tedious or impossible to express in a declarative approach.
      - Loops
      - Conditionals

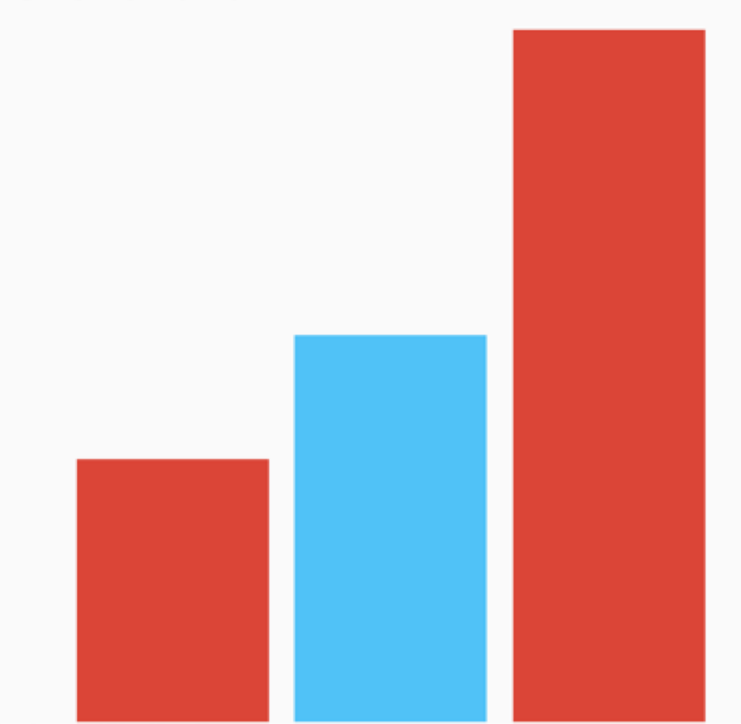
Introduction to IaC

- Drawbacks of Declarative IaC
  - Limited flexibility.
    - Some things are tedious or impossible to express in a declarative approach.
      - Loops
      - Conditionals
  - Opaque operations.
    - Difficult to debug or reason about.

Introduction to IaC

- Drawbacks of Declarative IaC
  - Learning curve.
    - Custom DSLs and runtimes.

Course overview



1. Introduction to IaC
2. Terraform Basics
3. Terraform in Practice

What is Terraform?

Terraform is an infrastructure as code tool that lets you build, change, and version cloud and on-prem resources safely and efficiently.

HashiCorp Terraform is an infrastructure as code tool that lets you define both cloud and on-prem resources in human-readable configuration files that you can version, reuse, and share. You can then use a consistent workflow to provision and manage all of your infrastructure throughout its lifecycle. Terraform can manage low-level components like compute, storage, and networking resources, as well as high-level components like DNS entries and SaaS features.

Terraform Basics

- What is Terraform?
  - Open source tool.

Terraform Basics

- What is Terraform?
  - Open source tool.
  - Command line interface (cli).

Terraform Basics

- What is Terraform?
  - Open source tool.
  - Command line interface (cli).
  - Multi Cloud Declarative IaC.

Terraform Basics

- What Terraform is **not**.

Terraform Basics

- What Terraform is **not**.
  - A multi cloud abstraction layer.

Terraform Basics

- What Terraform is **not**.
  - A multi cloud abstraction layer.
  - A shortcut to cloud provider management.

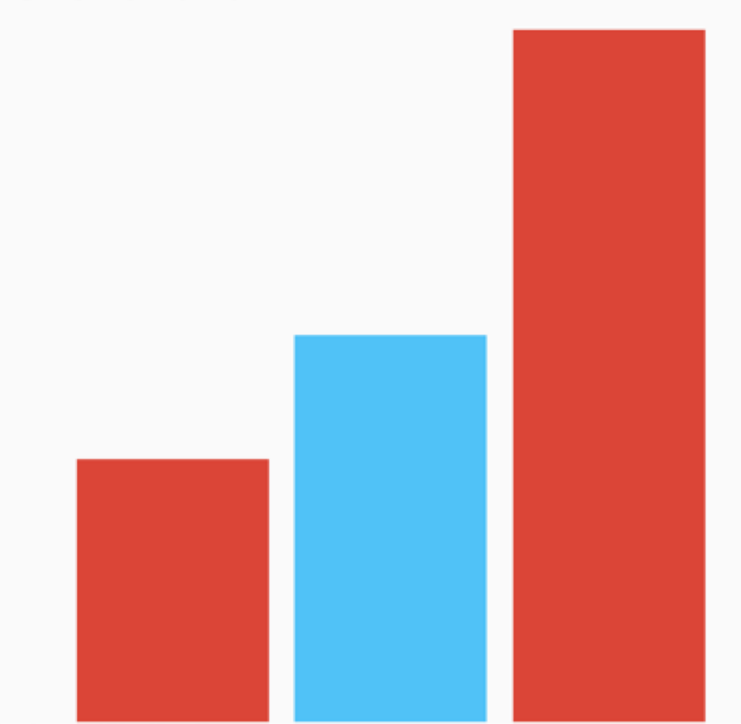
Terraform Basics

Demo: Hello Terraform

Terraform Basics

Demo: Terraform Syntax

Course overview



1. Introduction to IaC
2. Terraform Basics
3. Terraform in Practice

Terraform in Practice

- State
  - Terraform’s view of the world.

Terraform in Practice

- State
  - Terraform’s view of the world.
  - Can be local or remote.

Terraform in Practice

- State
  - Terraform’s view of the world.
  - Can be local or remote.
  - Many providers.

Terraform in Practice

- State
  - Terraform’s view of the world.
  - Can be local or remote.
  - Many providers.
  - Locks.

Terraform in Practice

- State
  - Do **not** edit state.

Terraform in Practice

- State
  - Do **not** edit state.
  - Use `state` command:

```
Usage: terraform [global options] state <subcommand> [options] [args]

  This command has subcommands for advanced state management.

  These subcommands can be used to slice and dice the Terraform state.
  This is sometimes necessary in advanced cases. For your safety, all
  state management commands that modify the state create a timestamped
  backup of the state prior to making modifications.

  The structure and output of the commands is specifically tailored to work
  well with the common Unix utilities such as grep, awk, etc. We recommend
  using those tools to perform more advanced state tasks.

Subcommands:
    list                List resources in the state
    mv                  Move an item in the state
    pull                Pull current state and output to stdout
    push                Update remote state from a local state file
    replace-provider    Replace provider in the state
    rm                  Remove instances from the state
    show                Show a resource in the state
```

Terraform in Practice

- Modules

Terraform in Practice

- Modules
  - Reusable libraries of Terraform code.

Terraform in Practice

- Modules
  - Reusable libraries of Terraform code.
  - Can be local or remote.

Terraform in Practice

- Modules
  - Reusable libraries of Terraform code.
  - Can be local or remote.
  - Versioned.

Terraform in Practice

- Registry
  - Hashicorp maintains a public repository of reusable modules. https://registry.terraform.io/

Terraform in Practice

- Registry
  - Hashicorp maintains a public repository of reusable modules. https://registry.terraform.io/
    - Quality is not guaranteed.

Terraform in Practice

Demo: Modules

LEARN-INFRASTRUCTUR...

- .devcontainer
- slides
- terraform
  - 00_hello_terraform
    - 00_setup
    - 01_workflow
  - 01_syntax
  - 02_modules
  - 03_example
  - 04_extras
- .gitignore

terraform > 01_syntax > 04_Data_Sources_and_Resources > resources.tf > ...

```
#Resources
# Objects managed by Terraform such as VMs or S3 Buckets.
# Declaring a Resource tells Terraform that it should CREATE
# and manage the resource described.  If the Resource already exists
# it must be imported into Terraform's state.
resource "aws_s3_bucket" "simple_bucket" {
}

# Resources accept configuration in the form of attributes.
# Each resource supports a unique set of attributes.  To understand a resource
# use the reference in the Terraform Registry:
# https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket

# Terraform will return an object representing the resource including all the
# calculated values.

output "simple_bucket" {
  value = aws_s3_bucket.simple_bucket
```

Terraform in Practice

Demo: Putting it together