**Prerequisites**

Basic terminal skills

Basic understanding of on premises and cloud architecture

**Assessment Type** Multiple choice

**Format** Online proctored

**Duration** 1 hour

**Price** $70.50 USD

**Language** English

**Expiration** 2 years

Exam Objectives

1 Understand infrastructure as code (IaC) concepts

1a Explain what IaC is

Infrastructure as Code is the management and configuration of infrastructure using source code. Any change to the infrastructure is done by editing the source code first and applying it. This source code can be stored in an enterprise’s Git or other source repository which allows for versioning, accountability and compliance.

1b Describe advantages of IaC patterns

Versioning – because the infrastructure is written code, it can be stored in a versioning system such as Git. Any changes to the code can be tracked and rolled back if necessary.

Reusability – once a particular infrastructure has been created in code it can be used elsewhere to create the same environment. Proponents of IaC refer to the concept if Idempotence, which is the idea that a given piece of code will yield the same environment every time.

Automation – using a template makes creating and tearing down environments much quicker and easier and allows for greater automation.

2 Understand Terraform's purpose (vs other IaC)

2a Explain multi-cloud and provider-agnostic benefits

A multi-cloud environment can increase fault tolerance if a cloud provider loses connectivity. However, multi-cloud environments are more complex to maintain than a single cloud environment. All major cloud providers have an IaC offering. AWS has CloudFormation, Azure has Azure Resource Manager and GCP has Cloud Build. However, these tools are not compatible with other cloud environments. Terraform is provider agnostic and can be used in every major cloud prover’s environment.

2b Explain the benefits of state

3 Understand Terraform basics

3a Handle Terraform and provider installation and versioning

3b Describe plugin based architecture

3c Demonstrate using multiple providers

3d Describe how Terraform finds and fetches providers

3e Explain when to use and not use provisioners and when to use

local-exec or remote-exec

4 Use the Terraform CLI (outside of core workflow)

4a Given a scenario: choose when to use terraform fmt to format code

4b Given a scenario: choose when to use terraform taint to taint Terraform resources

4c Given a scenario: choose when to use terraform import to import existing infrastructure into your Terraform state

4d Given a scenario: choose when to use terraform workspace to create workspaces

4e Given a scenario: choose when to use terraform state to view Terraform state

4f Given a scenario: choose when to enable verbose logging and what the outcome/value is

5 Interact with Terraform modules

5a Contrast module source options

5b Interact with module inputs and outputs

5c Describe variable scope within modules/child modules

5d Discover modules from the public Terraform Module Registry

5e Defining module version

6 Navigate Terraform workflow

6a Describe Terraform workflow ( Write -> Plan -> Create )

6b Initialize a Terraform working directory ( terraform init )

6c Validate a Terraform configuration ( terraform validate )

6d

6 Generate and review an execution plan for Terraform ( terraform

Navigate Terraform workflow

plan )

6e Execute changes to infrastructure with Terraform ( terraform apply )

6f Destroy Terraform managed infrastructure ( terraform destroy )

7 Implement and maintain state

7a Describe default local backend

7b Outline state locking

7c Handle backend authentication methods

7d Describe remote state storage mechanisms and supported standard backends

7e Describe effect of Terraform refresh on state

7f Describe backend block in configuration and best practices for partial configurations

7g Understand secret management in state files

8 Read, generate, and modify configuration

8a Demonstrate use of variables and outputs

8b Describe secure secret injection best practice

8c Understand the use of collection and structural types

8d Create and differentiate resource and data configuration

8e Use resource addressing and resource parameters to connect resources together

8f Use Terraform built-in functions to write configuration

8g Configure resource using a dynamic block

8h Describe built-in dependency management (order of execution based)

9 Understand Terraform Cloud and Enterprise capabilities

9a Describe the benefits of Sentinel, registry, and workspaces

9b Differentiate OSS and TFE workspaces

9c Summarize features of Terraform Cloud