Question 1

Question Type: MultipleChoice

HashiCorp Configuration Language (HCL) supports user-denned functions.

Options:

A- True

B- False

Answer:

B

Explanation:

HashiCorp Configuration Language (HCL) does not support user-defined functions. You can only use the built-in functions that are provided by the language. The built-in functions allow you to perform various operations and transformations on values within expressions. The general syntax for function calls is a function name followed by comma-separated arguments in parentheses, such asmax(5, 12, 9). You can find the documentation for all of the available built-in functions in the Terraform Registry or the Packer Documentation, depending on which tool you are using.Reference= :Functions - Configuration Language | Terraform:Functions - Configuration Language | Packer

Question 2

Question Type: MultipleChoice

A developer on your team is going lo leaf down an existing deployment managed by Terraform and deploy a new one. However, there is a server resource named aws instant.ubuntu[l] they would like to keep. What command should they use to tell Terraform to stop managing that specific resource?

Options:

A- Terraform plan rm:aws_instance.ubuntu[1]

B- Terraform state rm:aws_instance.ubuntu[1]

C- Terraform apply rm:aws_instance.ubuntu[1]

D- Terraform destory rm:aws_instance.ubuntu[1]

Answer:

B

Explanation:

To tell Terraform to stop managing a specific resource without destroying it, you can use theterraform state rmcommand. This command will remove the resource from the Terraform state, which means that Terraform will no longer track or update the corresponding remote object. However, the object will still exist in the remote system and you can later useterraform importto start managing it again in a different configuration or workspace. The syntax for this command isterraform state rm

, where

is the resource address that identifies the resource instance to remove. For example,terraform state rm aws_instance.ubuntu[1]will remove the second instance of theaws_instanceresource namedubuntufrom the state.Reference= :Command: state rm:Moving Resources

Question 3

Question Type: **MultipleChoice**

Which of the following is not a key principle of infrastructure as code?

Options:

A- Self-describing infrastructure

B- Idempotence

C- Versioned infrastructure

D- Golden images

Answer:

D

Explanation:

The key principle of infrastructure as code that is not listed among the options isgolden images. Golden images are pre-configured, ready-to-use virtual machine images that contain a specific set of software and configuration. They are often used to create multiple identical instances of the same environment, such as for testing or production. However, golden images are not a principle of infrastructure as code, but rather a technique that can be used with or without infrastructure as code. The other options are all key principles of infrastructure as code, as explained below:

Self-describing infrastructure: This means that the infrastructure is defined in code that describes its desired state, rather than in scripts that describe the steps to create it. This makes the infrastructure easier to understand, maintain, and reproduce.

Idempotence: This means that applying the same infrastructure code multiple times will always result in the same state, regardless of the initial state. This makes the infrastructure consistent and predictable, and avoids errors or conflicts caused by repeated actions.

Question 4

Question Type: **MultipleChoice**

Your risk management organization requires that new AWS S3 buckets must be private and encrypted at rest. How can Terraform Cloud automatically and proactively enforce this security control?

Options:

A- Auditing cloud storage buckets with a vulnerability scanning tool

B- By adding variables to each Terraform Cloud workspace to ensure these settings are always enabled

C- With an S3 module with proper settings for buckets

D- With a Sentinel policy, which runs before every apply

Answer:

D

Explanation:

The best way to automatically and proactively enforce the security control that new AWS S3 buckets must be private and encrypted at rest is with a Sentinel policy, which runs before every apply. Sentinel is a policy as code framework that allows you to define and enforce logic-based policies for your infrastructure. Terraform Cloud supports Sentinel policies for all paid tiers, and can run them before anyterraform planorterraform applyoperation. You can write a Sentinel policy that checks the configuration of the S3 buckets and ensures that they have the proper settings for privacy and encryption, and then assign the policy to your Terraform Cloud organization or workspace. This way, Terraform Cloud will prevent any changes that violate the policy from being applied.Reference= [Sentinel Policy Framework], [Manage Policies in Terraform Cloud], [Write and Test Sentinel Policies for Terraform]

Question 5

Question Type: MultipleChoice

What is the name of the default file where Terraform stores the state?

Type your answer in the field provided. The text field is not case-sensitive and all variations of the correct answer are accepted.

Options:

A- Terraformtfstate

Answer:

A

Explanation:

The name of the default file where Terraform stores the state isterraform.tfstate. This file contains a JSON representation of the current state of the infrastructure managed by Terraform. Terraform uses this file to track the metadata and attributes of the resources, and to plan and apply changes. By default, Terraform stores the state file locally in the same directory as the configuration files, but it can also be configured to store the state remotely in a backend.Reference= [Terraform State], [State File Format]

Question 6

Question Type: MultipleChoice

Which two steps are required to provision new infrastructure in the Terraform workflow? Choose two correct answers.

Options:

A- Plan

B- Import

C- Alidate

D- Init

E- apply

Answer:

D, E

Explanation:

The two steps that are required to provision new infrastructure in the Terraform workflow areinitandapply. Theterraform initcommand initializes a working directory containing Terraform configuration files. It downloads and installs the provider plugins that are needed for the configuration, and prepares the backend for storing the state. Theterraform applycommand applies the changes required to reach the desired state of the configuration, as described by the resource definitions in the configuration files. It shows a plan of the proposed changes and asks for confirmation before making any changes to the infrastructure.Reference= [The Core Terraform Workflow], [Initialize a Terraform working directory with init], [Apply Terraform Configuration with apply]

Question 7

Question Type: MultipleChoice

You add a new provider to your configuration and immediately run terraform apply in the CD using the local backend. Why does the apply fail?

Options:

A- The Terraform CD needs you to log into Terraform Cloud first

B- Terraform requires you to manually run terraform plan first

C- Terraform needs to install the necessary plugins first

D- Terraform needs you to format your code according to best practices first

Answer:

C

Explanation:

The reason why the apply fails after adding a new provider to the configuration and immediately runningterraform applyin the CD using the local backend is because Terraform needs to install the necessary plugins first. Terraform providers are plugins that Terraform uses to interact with various cloud services and other APIs. Each provider has a source address that determines where to download it from. When Terraform encounters a new provider in the configuration, it needs to runterraform initfirst to install the provider plugins in a local directory. Without the plugins, Terraform cannot communicate with the provider and perform the desired actions.Reference= [Provider Requirements], [Provider Installation]

Question 8

Question Type: **MultipleChoice**

Which Terraform collection type should you use to store key/value pairs?

Options:

A- Set

B- Map

C- Tuple

D- list

Answer:

B

Explanation:

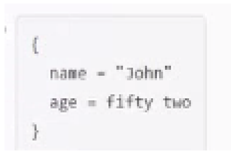
The Terraform collection type that should be used to store key/value pairs ismap. A map is a collection of values that are accessed by arbitrary labels, called keys. The keys and values can be of any type, but the keys must be unique within a map. For example,var = { key1 = 'value1', key2 = 'value2' }is a map with two key/value pairs. Maps are useful for grouping related values together, such as configuration options or metadata.Reference= [Collection Types], [Map Type Constraints]

Question 9

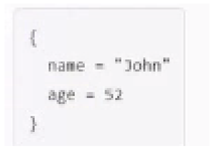
Question Type: **MultipleChoice**

In Terraform HCL, an object type of object({name=string, age-number}) would match this value.

A)

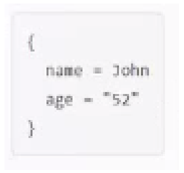


```
{
  name = "John"
  age = fifty two
}
```

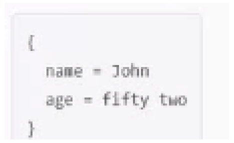


```
{
  name = "John"
  age = 52
}
```

C)


```
{
  name = John
  age = "52"
}
```

D)


```
{
  name = John
  age = fifty two
}
```

Options:

A- Option A

B- Option B

C- Option C

D- Option D

Answer:

B

Question 10

Question Type: MultipleChoice

Which provider authentication method prevents credentials from being stored in the state file?

Options:

A- Using environment variables

B- Specifying the login credentials in the provider block

C- Setting credentials as Terraform variables

D- None of the above

Answer:

D

Explanation:

None of the above methods prevent credentials from being stored in the state file. Terraform stores the provider configuration in the state file, which may include sensitive information such as credentials. This is a potential security risk and should be avoided if possible. To prevent credentials from being stored in the state file, you can use one of the following methods:

Use environment variables to pass credentials to the provider. This way, the credentials are not part of the provider configuration and are not stored in the state file. However, this method may not work for some providers that require credentials to be set in the provider block.

Use dynamic credentials to authenticate with your cloud provider. This way, Terraform Cloud or Enterprise will request temporary credentials from your cloud provider for each run and use them to provision your resources. The credentials are not stored in the state file and are revoked after the run is completed. This method is supported for AWS, Google Cloud Platform, Azure, and Vault.Reference= : [Sensitive Values in State] :Authenticate providers with dynamic credentials