HashiCorp Terraform-Associate-004 HCTA0-004 Practice Questions

Shared by Michael on 18-08-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

Which of these commands makes your code more human readable?

Options:

A- Terraform validate
B- Terraform output
C- Terraform show
D- Terraform file

Answer:

D

Explanation:

The command that makes your code more human readable isterraform fmt. This command is used to rewrite Terraform configuration files to a canonical format and style, following the Terraform language style conventions and other minor adjustments for readability. The command is optional, opinionated, and has no customization options, but it is recommended to ensure consistency of style across different Terraform codebases. Consistency can help your team understand the code more quickly and easily, making the use ofterraform fmtvery important. You can run this command on your configuration files before committing them to source control or as part of your CI/CD pipeline.Reference= :Command: fmt:Using Terraform fmt Command to Format Your Terraform Code

Question 2

Question Type: MultipleChoice

What are some benefits of using Sentinel with Terraform Cloud/Terraform Cloud? Choose three correct answers.

Options:

A- You can restrict specific resource configurations, such as disallowing the use of CIDR=0.0.0.0/0.

B- You can check out and check in cloud access keys
C- Sentinel Policies can be written in HashiCorp Configuration Language (HCL)
D- Policy-as-code can enforce security best practices
E- You can enforce a list of approved AWS AMIs

Answer:

A, D, E

Explanation:

Sentinel is a policy-as-code framework that allows you to define and enforce rules on your Terraform configurations, states, and plans1. Some of the benefits of using Sentinel with Terraform Cloud/Terraform Enterprise are:

*You can restrict specific resource configurations, such as disallowing the use of CIDR=0.0.0.0/0, which would open up your network to the entire internet. This can help you prevent misconfigurations or security vulnerabilities in your infrastructure2.

*Policy-as-code can enforce security best practices, such as requiring encryption, authentication, or compliance standards. This can help you protect your data and meet regulatory requirements3.

*You can enforce a list of approved AWS AMIs, which are pre-configured images that contain the operating system and software you need to run your applications. This can help you ensure consistency, reliability, and performance across your infrastructure4.

Reference =

*1: Terraform and Sentinel | Sentinel | HashiCorp Developer

*2: Terraform Learning Resources: Getting Started with Sentinel in Terraform Cloud

*3: Exploring the Power of HashiCorp Terraform, Sentinel, Terraform Cloud ...

*4: Using New Sentinel Features in Terraform Cloud -- Medium

Question 3

Question Type: MultipleChoice

How is terraform import run?

Options:

A- As a part of terraform init
B- As a part of terraform plan
C- As a part of terraform refresh
D- By an explicit call
E- terraform import

Answer:

D

Explanation:

The terraform import command is not part of any other Terraform workflow. It must be explicitly invoked by the user with the appropriate arguments, such as the resource address and the ID of the existing infrastructure to import.Reference= [Importing Infrastructure]

Question 4

Question Type: MultipleChoice

When using Terraform to deploy resources into Azure, which scenarios are true regarding state files? (Select two.)

Options:

A- When you change a Terraform-managed resource via the Azure Cloud Console, Terraform updates the state file to reflect the change during the next plan or apply
B- Changing resources via the Azure Cloud Console records the change in the current state file
C- When you change a resource via the Azure Cloud Console, Terraform records the changes in a new state file
D- Changing resources via the Azure Cloud Console does not update current state file

Answer:

A, D

Explanation:

Terraform state is a representation of the infrastructure that Terraform manages. Terraform

uses state to track the current status of the resources it creates and to plan future changes. However, Terraform state is not aware of any changes made to the resources outside of Terraform, such as through the Azure Cloud Console, the Azure CLI, or the Azure API. Therefore, changing resources via the Azure Cloud Console does not update the current state file, and it may cause inconsistencies or conflicts with Terraform's desired configuration. To avoid this, it is recommended to manage resources exclusively through Terraform or to use theterraform importcommand to bring existing resources under Terraform's control.

When you change a Terraform-managed resource via the Azure Cloud Console, Terraform does not immediately update the state file to reflect the change. However, the next time you runterraform planorterraform apply, Terraform will compare the state file with the actual state of the resources in Azure and detect any drifts or differences. Terraform will then update the state file to match the current state of the resources and show you the proposed changes in the execution plan. Depending on the configuration and the change, Terraform may try to undo the change, modify the resource further, or recreate the resource entirely. To avoid unexpected or destructive changes, it is recommended to review the execution plan carefully before applying it or to use theterraform refreshcommand to update the state file without applying any changes.

Reference=Purpose of Terraform State,Terraform State,Managing State,Importing Infrastructure, [Command: plan], [Command: apply], [Command: refresh]

Question 5

Question Type: MultipleChoice

Which option best command would be use to access all of the attributes and details of a resource managed by Terraform?

Options:

A- Terraform state show ' provider_type_name
B- Terraform state list
C- Terraform get provider_type_name
D- Terraform state list provider_type_name

Answer:

A

Explanation:

This is the command that you would use to access all of the attributes and details of a resource managed by Terraform, by providing the resource address as an argument. For

example,terraform state show 'aws_instance.example'will show you all the information about the AWS instance named example.

Question 6

Question Type: MultipleChoice

As a member of an operations team that uses infrastructure as code (lac) practices, you are tasked with making a change to an infrastructure stack running in a public cloud. Which pattern would follow laC best practices for making a change?

Options:

A- Make the change via the public cloud API endpoint
B- Clone the repository containing your infrastructure code and then run the code
C- Use the public cloud console to make the change after a database record has been approved
D- Make the change programmatically via the public cloud CLI
E- Submit a pull request and wait for an approved merge of the proposed changes

Answer:

E

Explanation:

You do not need to use different Terraform commands depending on the cloud provider you use. Terraform commands are consistent across different providers, as they operate on the Terraform configuration files and state files, not on the provider APIs directly.

Question 7

Question Type: MultipleChoice

A resource block is shown in the Exhibit section of this page. How would you reference the attribute name of this resource in HCL?

Options:

A- resource.kubernetes_namespace.example.name
B- kubernetes_namespace.example.name
C- data.kubernetes.namespace.name
D- kubernetes_namespace.test.name

Answer:

B

Explanation:

In Terraform, the correct way to reference a resource attribute is:

pgsql

CopyEdit

resource_type.resource_name.attribute

For example, if the resource block is:

hcl

CopyEdit

resource 'kubernetes_namespace' 'example' {

metadata {

name = 'my-namespace'

}

}

To reference the name attribute, use:

pgsql

CopyEdit

kubernetes_namespace.example.name

Explanation of incorrect answers:

A (resource.kubernetes_namespace.example.name)-- Incorrect. Terraform does not use the resource. prefix when referencing resources.

C (data.kubernetes.namespace.name)-- Incorrect. This syntax is used fordata sources, not resources.

D (kubernetes_namespace.test.name)-- Incorrect. The resource name is 'example', not 'test'.

Official Terraform Documentation Reference:

Terraform Resource Reference

Question 8

Question Type: MultipleChoice

When you run terraform apply, the Terraform CLI will print output values from both the root module and any child modules.

Options:

A- True
B- False

Answer:

A

Explanation:

Rationale for Correct Answer (True):

When terraform apply completes successfully, Terraform prints output values. Outputs from both root and child modules are displayed, but child module outputs must be explicitly exposed through the root module outputs to be visible at the CLI.

Analysis of Incorrect Option:

False: Incorrect, because Terraform does display output values, but only if they are exposed from child modules to the root module.

Key Concept:

Outputs help you extract important information (e.g., IP addresses, resource IDs) from your configuration.

Terraform Exam Objective -- Read, Generate, and Modify Configurations.