

## Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)

<https://www.2passeasy.com/dumps/Terraform-Associate-003/>



**NEW QUESTION 1**

How would you reference the volume IDs associated with the ebs\_block\_device blocks in this configuration?

```
resource "aws_instance" "example" {  
  ami = "ami-abc123"  
  instance_type = "t2.micro"  
  
  ebs_block_device {  
    device_name = "sda2"  
    volume_size = 16  
  }  
  
  ebs_block_device {  
    device_name = "sda3"  
    volume_size = 20  
  }  
}
```

- A. aws\_instance.example.ebs\_block\_device[sda2,sda3].volume\_id
- B. aws\_Instance.example.ebs\_block\_device.[\*].volume\_id
- C. aws\_Instance.example.ebs\_block\_device.volume\_ids
- D. aws\_instance.example-ebs\_block\_device.\*.volume\_id

**Answer:** D

**Explanation:**

This is the correct way to reference the volume IDs associated with the ebs\_block\_device blocks in this configuration, using the splat expression syntax. The other options are either invalid or incomplete.

**NEW QUESTION 2**

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

- 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 is golden 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.

? Versioned infrastructure: This means that the infrastructure code is stored in a version control system, such as Git, that tracks the changes and history of the code. This makes the infrastructure code reusable, auditable, and collaborative, and enables practices such as branching, merging, and rollback. References = [Introduction to Infrastructure as Code with Terraform], [Infrastructure as Code in a Private or Public Cloud]

**NEW QUESTION 3**

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

- 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 as max(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. References = : Functions - Configuration Language | Terraform : Functions - Configuration Language | Packer

**NEW QUESTION 4**

Which option cannot be used to keep secrets out of Terraform configuration files?

- A. A Terraform provider
- B. Environment variables
- C. A -var flag
- D. secure string

**Answer:** D

**Explanation:**

A secure string is not a valid option to keep secrets out of Terraform configuration files. A secure string is a feature of AWS Systems Manager Parameter Store that allows you to store sensitive data encrypted with a KMS key. However, Terraform does not support secure strings natively and requires a custom data source to retrieve them. The other options are valid ways to keep secrets out of Terraform configuration files. A Terraform provider can expose secrets as data sources that can be referenced in the configuration. Environment variables can be used to set values for input variables that contain secrets. A -var flag can be used to pass values for input variables that contain secrets from the command line or a file. References = [AWS Systems Manager Parameter Store], [Terraform AWS Provider Issue #55], [Terraform Providers], [Terraform Input Variables]

**NEW QUESTION 5**

Terraform providers are always installed from the Internet.

- A. True
- B. False

**Answer:** B

**Explanation:**

Terraform providers are not always installed from the Internet. There are other ways to install provider plugins, such as from a local mirror or cache, from a local filesystem directory, or from a network filesystem. These methods can be useful for offline or air-gapped environments, or for customizing the installation process. You can configure the provider installation methods using the provider\_installation block in the CLI configuration file.

**NEW QUESTION 6**

A developer on your team is going to leave down an existing deployment managed by Terraform and deploy a new one. However, there is a server resource named aws\_instance.ubuntu[1] they would like to keep. What command should they use to tell Terraform to stop managing that specific resource?

- 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 destroy rm:aws\_instance.ubuntu[1]

**Answer:** B

**Explanation:**

To tell Terraform to stop managing a specific resource without destroying it, you can use the terraform state rm command. 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 use terraform import to start managing it again in a different configuration or workspace. The syntax for this command is terraform state rm <address>,

where <address> 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 the aws\_instance resource named ubuntu from the state. References = : Command: state rm : Moving Resources

**NEW QUESTION 7**

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

A)



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

B)



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

C)

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

D)

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

- A. Option A
- B. Option B
- C. Option C
- D. Option D

**Answer:** B

#### NEW QUESTION 8

Which is the best way to specify a tag of v1.0.0 when referencing a module stored in Git (for example. Git::https://example.com/vpc.git)?

- A. Append pref=v1.0.0 argument to the source path
- B. Add version = ??1.0.0?? parameter to module block
- C. Nothing modules stored on GitHub always default to version 1.0.0

**Answer:** A

#### Explanation:

The best way to specify a tag of v1.0.0 when referencing a module stored in Git is to append ?ref=v1.0.0 argument to the source path. This tells Terraform to use a specific Git reference, such as a branch, tag, or commit, when fetching the module source code. For example, source = "git::https://example.com/vpc.git?ref=v1.0.0". This ensures that the module version is consistent and reproducible across different environments. References = [Module Sources], [Module Versions]

#### NEW QUESTION 9

Which of these are features of Terraform Cloud? Choose two correct answers.

- A. Automated infrastructure deployment visualization
- B. Automatic backups
- C. A web-based user interface (UI)
- D. Remote state storage

**Answer:** CD

#### Explanation:

These are features of Terraform Cloud, which is a hosted service that provides a web-based UI, remote state storage, remote operations, collaboration features, and more for managing your Terraform infrastructure.

#### NEW QUESTION 10

Which of the following is not a valid Terraform collection type?

- A. Tree
- B. Map
- C. List
- D. set

**Answer:** A

#### Explanation:

This is not a valid Terraform collection type, as Terraform only supports three collection types: list, map, and set. A tree is a data structure that consists of nodes with parent-child relationships, which is not supported by Terraform.

#### NEW QUESTION 10

If a module declares a variable with a default, that variable must also be defined within the module.

- A. True
- B. False

**Answer:** B

#### Explanation:

A module can declare a variable with a default value without requiring the caller to define it. This allows the module to provide a sensible default behavior that can be customized by the caller if needed. References = [Module Variables]

#### NEW QUESTION 11

What Terraform command always causes a state file to be updated with changes that might have been made outside of Terraform?

- A. Terraform plan --refresh-only
- B. Terraform show --json
- C. Terraform apply --lock=false
- D. Terraform plan target-state

**Answer:** A

#### Explanation:

This is the command that always causes a state file to be updated with changes that might have been made outside of Terraform, as it will only refresh the state file with the current status of the real resources, without making any changes to them or creating a plan.

#### NEW QUESTION 13

All standard backend types support state locking, and remote operations like plan, apply, and destroy.

- A. True
- B. False

**Answer:** B

#### Explanation:

Not all standard backend types support state locking and remote operations like plan, apply, and destroy. For example, the local backend does not support remote operations and state locking. State locking is a feature that ensures that no two users can make changes to the state file at the same time, which is crucial for preventing race conditions. Remote operations allow running Terraform commands on a remote server, which is supported by some backends like remote or consul, but not all.

References:

? Terraform documentation on backends: Terraform Backends

? Detailed backend support: Terraform Backend Types

#### NEW QUESTION 17

When you use a remote backend that needs authentication, HashiCorp recommends that you:

- A. Write the authentication credentials in the Terraform configuration files
- B. Keep the Terraform configuration files in a secret store
- C. Push your Terraform configuration to an encrypted git repository
- D. Use partial configuration to load the authentication credentials outside of the Terraform code

**Answer:** D

#### Explanation:

This is the recommended way to use a remote backend that needs authentication, as it allows you to provide the credentials via environment variables, command-line arguments, or interactive prompts, without storing them in the Terraform configuration files.

#### NEW QUESTION 21

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

- 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:** ADE

#### Explanation:

Sentinel is a policy-as-code framework that allows you to define and enforce rules on your Terraform configurations, states, and plans<sup>1</sup>. 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 infrastructure<sup>2</sup>.
- 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 requirements<sup>3</sup>.
- 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 infrastructure<sup>4</sup>. References =
- 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

#### NEW QUESTION 25

You should run terraform fnt to rewrite all Terraform configurations within the current working directory to conform to Terraform-style conventions.

- A. True
- B. False

**Answer:** A

#### Explanation:

You should run terraform fmt to rewrite all Terraform configurations within the current working directory to conform to Terraform-style conventions. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability. It is recommended to use this command to ensure consistency of style across different Terraform codebases. The command is optional, opinionated, and has no customization options, but it can help you and your team understand the code more quickly and easily. References = : Command: fmt : Using Terraform fmt Command to Format Your Terraform Code

#### NEW QUESTION 28

Before you can use a remote backend, you must first execute terra-form init.

- A. True
- B. False

**Answer:** A

#### Explanation:

Before using a remote backend in Terraform, it is mandatory to run terraform init. This command initializes a Terraform working directory, which includes configuring the backend. If a remote backend is specified, terraform init will set up the working directory to use it, including copying any existing state to the remote backend if necessary. References = This principle is a fundamental part of working with Terraform and its backends, as outlined in general Terraform documentation and best practices. The specific HashiCorp Terraform Associate (003) study materials in the provided files did not include direct references to this information.

#### NEW QUESTION 30

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

- A. You can enforce a list of approved AWS AMIs
- B. Policy-as-code can enforce security best practices
- C. You can check out and check in cloud access keys
- D. You can restrict specific resource configurations, such as disallowing the use of CIDR=0.0.0.0/0.
- E. Sentinel Policies can be written in HashiCorp Configuration Language (HCL)

**Answer:** ABD

#### Explanation:

These are some of the benefits of using Sentinel with Terraform Cloud/Terraform Enterprise, as they allow you to implement logic-based policies that can access and evaluate the Terraform plan, state, and configuration. The other options are not true, as Sentinel does not manage cloud access keys, and Sentinel policies are written in Sentinel language, not HCL.

#### NEW QUESTION 32

You modified your Terraform configuration and run Terraform plan to review the changes. Simultaneously, your teammate manually modified the infrastructure component you are working on. Since you already ran terraform plan locally, the execution plan for terraform apply will be the same.

- A. True
- B. False

**Answer:** B

#### Explanation:

The execution plan for terraform apply will not be the same as the one you ran locally with terraform plan, if your teammate manually modified the infrastructure component you are working on. This is because Terraform will refresh the state file before applying any changes, and will detect any differences between the state and the real resources.

#### NEW QUESTION 33

You want to know from which paths Terraform is loading providers referenced in your Terraform configuration (\* files). You need to enable additional logging messages to find this out. Which of the following would achieve this?

- A. Set verbose for each provider in your Terraform configuration
- B. Set the environment variable TF\_LOG\_TRACE
- C. Set the environment variable TF\_LOG\_PATH
- D. Set the environment variable TF\_log\_TRACE

**Answer:** B

#### Explanation:

This will enable additional logging messages to find out from which paths Terraform is loading providers referenced in your Terraform configuration files, as it will set the log level to TRACE, which is the most verbose and detailed level.

#### NEW QUESTION 37

You want to define multiple data disks as nested blocks inside the resource block for a virtual machine. What Terraform feature would help you define the blocks using the values in a variable?

- A. Local values
- B. Count arguments
- C. Collection functions
- D. Dynamic blocks

**Answer:** D

#### Explanation:

Dynamic blocks in Terraform allow you to define multiple nested blocks within a resource based on the values of a variable. This feature is particularly useful for scenarios where the number of nested blocks is not fixed and can change based on variable input.

#### NEW QUESTION 41

Terraform can only manage resource dependencies if you set them explicitly with the depends\_on argument.

- A. True
- B. False

**Answer:** B

#### Explanation:

Terraform can manage resource dependencies implicitly or explicitly. Implicit dependencies are created when a resource references another resource or data source in its arguments. Terraform can infer the dependency from the reference and create or destroy the resources in the correct order. Explicit dependencies are created when you use the depends\_on argument to specify that a resource depends on another resource or module. This is useful when Terraform cannot infer the dependency from the configuration or when you need to create a dependency for some reason outside of Terraform's scope. References = : Create resource dependencies : Terraform Resource Dependencies Explained

#### NEW QUESTION 45

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?

- 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 running terraform apply in 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 run terraform init first to install the provider plugins in a local directory. Without the plugins, Terraform cannot communicate with the provider and perform the desired actions. References = [Provider Requirements], [Provider Installation]

#### NEW QUESTION 48

How is terraform import run?

- 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. All of the above

**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. References = [Importing Infrastructure]

#### NEW QUESTION 51

In a Terraform Cloud workspace linked to a version control repository speculative plan run start automatically commit changes to version control.

- A. True
- B. False

**Answer:** A

#### Explanation:

When you use a remote backend that needs authentication, HashiCorp recommends that you:

#### NEW QUESTION 53

How could you reference an attribute from the vsphere\_datacenter data source for use with the datacenter\_id argument within the vsphere\_folder resource in the following configuration?

```
data "vsphere_datacenter" "dc" {}

resource "vsphere_folder" "parent" {
  path = "Production"
  type = "vm"
  datacenter_id = _____
}
```

- A. Data.vsphere\_datacenter.DC.id
- B. Vsphere\_datacenter.dc.id
- C. Data,dc,id
- D. Data.vsphere\_datacenter,dc

**Answer:** A

**Explanation:**

The correct way to reference an attribute from the vsphere\_datacenter data source for use with the datacenter\_id argument within the vsphere\_folder resource in the following configuration is data.vsphere\_datacenter.dc.id. This follows the syntax for accessing data source attributes, which is data.TYPE.NAME.ATTRIBUTE. In this case, the data source type is vsphere\_datacenter, the data source name is dc, and the attribute we want to access is id. The other options are incorrect because they either use the wrong syntax, the wrong punctuation, or the wrong case. References = [Data Source: vsphere\_datacenter], [Data Source: vsphere\_folder], [Expressions: Data Source References]

**NEW QUESTION 56**

Which of the following methods, used to provision resources into a public cloud, demonstrates the concept of infrastructure as code?

- A. curl commands manually run from a terminal
- B. A sequence of REST requests you pass to a public cloud API endpoint Most Voted
- C. A script that contains a series of public cloud CLI commands
- D. A series of commands you enter into a public cloud console

**Answer:** C

**Explanation:**

The concept of infrastructure as code (IaC) is to define and manage infrastructure using code, rather than manual processes or GUI tools. A script that contains a series of public cloud CLI commands is an example of IaC, because it uses code to provision resources into a public cloud. The other options are not examples of IaC, because they involve manual or interactive actions, such as running curl commands, sending REST requests, or entering commands into a console. References = [Introduction to Infrastructure as Code with Terraform] and [Infrastructure as Code]

**NEW QUESTION 61**

Which of these are secure options for storing secrets for connecting to a Terraform remote backend? Choose two correct answers.

- A. A variable file
- B. Defined in Environment variables
- C. Inside the backend block within the Terraform configuration
- D. Defined in a connection configuration outside of Terraform

**Answer:** BD

**Explanation:**

Environment variables and connection configurations outside of Terraform are secure options for storing secrets for connecting to a Terraform remote backend. Environment variables can be used to set values for input variables that contain secrets, such as backend access keys or tokens. Terraform will read environment variables that start with TF\_VAR\_ and match the name of an input variable. For example, if you have an input variable called backend\_token, you can set its value with the environment variable TF\_VAR\_backend\_token1. Connection configurations outside of Terraform are files or scripts that provide credentials or other information for Terraform to connect to a remote backend. For example, you can use a credentials file for the S3 backend2, or a shell script for the HTTP backend3. These files or scripts are not part of the Terraform configuration and can be stored securely in a separate location. The other options are not secure for storing secrets. A variable file is a file that contains values for input variables. Variable files are usually stored in the same directory as the Terraform configuration or in a version control system. This exposes the secrets to anyone who can access the files or the repository. You should not store secrets in variable files1. Inside the backend block within the Terraform configuration is where you specify the type and settings of the remote backend. The backend block is part of the Terraform configuration and is usually stored in a version control system. This exposes the secrets to anyone who can access the configuration or the repository. You should not store secrets in the backend block4. References = [Terraform Input Variables]1, [Backend Type: s3]2, [Backend Type: http]3, [Backend Configuration]4

**NEW QUESTION 63**

When should you write Terraform configuration files for existing infrastructure that you want to start managing with Terraform?

- A. You can import infrastructure without corresponding Terraform code
- B. Terraform will generate the corresponding configuration files for you
- C. Before you run terraform Import



D. After you run terraform import

**Answer:** C

**Explanation:**

You need to write Terraform configuration files for the existing infrastructure that you want to import into Terraform, otherwise Terraform will not know how to manage it. The configuration files should match the type and name of the resources that you want to import.

**NEW QUESTION 68**

How does the Terraform cloud integration differ from other state backends such as S3, Consul,etc?

- A. It can execute Terraform runs on dedicated infrastructure in Terraform Cloud
- B. It doesn't show the output of a terraform apply locally
- C. It is only arable lo paying customers
- D. All of the above

**Answer:** A

**Explanation:**

This is how the Terraform Cloud integration differs from other state backends such as S3, Consul, etc., as it allows you to perform remote operations on Terraform Cloud??s servers instead of your local machine. The other options are either incorrect or irrelevant.

**NEW QUESTION 71**

Which of the following does terraform apply change after you approve the execution plan? (Choose two.)

- A. Cloud infrastructure Most Voted
- B. The .terraform directory
- C. The execution plan
- D. State file
- E. Terraform code

**Answer:** AD

**Explanation:**

The terraform apply command changes both the cloud infrastructure and the state file after you approve the execution plan. The command creates, updates, or destroys the infrastructure resources to match the configuration. It also updates the state file to reflect the new state of the infrastructure. The .terraform directory, the execution plan, and the Terraform code are not changed by the terraform apply command. References = Command: apply and Purpose of Terraform State

**NEW QUESTION 74**

The public Terraform Module Registry is free to use.

- A. True
- B. False

**Answer:** A

**Explanation:**

The public Terraform Module Registry is free to use, as it is a public service that hosts thousands of self-contained packages called modules that are used to provision infrastructure. You can browse, use, and publish modules to the registry without any cost.

**NEW QUESTION 75**

You are using a networking module in your Terraform configuration with the name label my-network. In your main configuration you have the following code:

```
output "net_id" {  
  value = module.my_network.vnet_id  
}
```

When you run terraform validate, you get the following error:

```
Error: Reference to undeclared output value  
  
on main.tf line 12, in output "net_id":  
12:   value = module.my_network.vnet_id
```

What must you do to successfully retrieve this value from your networking module?

- A. Change the reference value to my-network,outputs,vmet\_id
- B. Define the attribute vmet\_id as a variable in the networking modeule
- C. Define the attribute vnet\_id as an output in the networking module
- D. Change the reference value module.my,network,outputs,vnet\_id

**Answer:** C

**Explanation:**

This is what you must do to successfully retrieve this value from your networking module, as it will expose the attribute as an output value that can be referenced by other modules or resources. The error message indicates that the networking module does not have an output value named vnet\_id, which causes the reference to fail.

#### NEW QUESTION 80

Which are forbidden actions when the terraform state file is locked? Choose three correct answers.

- A. Terraform state list
- B. Terraform destroy
- C. Terraform validate
- D. Terraform validate
- E. Terraform for
- F. Terraform apply

**Answer:** BCF

#### Explanation:

The terraform state file is locked when a Terraform operation that could write state is in progress. This prevents concurrent state operations that could corrupt the state.

The forbidden actions when the state file is locked are those that could write state, such as terraform apply, terraform destroy, terraform refresh, terraform taint, terraform

untaint, terraform import, and terraform state \*. The terraform validate command is also forbidden, because it requires an initialized working directory with the state file. The allowed actions when the state file is locked are those that only read state, such as terraform plan, terraform show, terraform output, and terraform console. References = [State Locking] and [Command: validate]

#### NEW QUESTION 85

What does state locking accomplish?

- A. Prevent accidental Prevent accident deletion of the state file
- B. Blocks Terraform commands from modifying, the state file
- C. Copies the state file from memory to disk
- D. Encrypts any credentials stored within the state file

**Answer:** B

#### Explanation:

This is what state locking accomplishes, by preventing other users from modifying the state file while a Terraform operation is in progress. This prevents conflicts and data loss.

#### NEW QUESTION 88

When do changes invoked by terraform apply take effect?

- A. After Terraform has updated the state file
- B. Once the resource provider has fulfilled the request
- C. Immediately
- D. None of the above are correct

**Answer:** B

#### Explanation:

Changes invoked by terraform apply take effect once the resource provider has fulfilled the request, not after Terraform has updated the state file or immediately. The state file is only a reflection of the real resources, not a source of truth.

#### NEW QUESTION 93

You have a Terraform configuration that defines a single virtual machine with no references to it, You have run terraform apply to create the resource, and then removed the resource definition from your Terraform configuration file.

What will happen you run terraform apply in the working directory again?

- A. Terraform will remove the virtual machine from the state file, but the resource will still exist
- B. Nothing
- C. Terraform will error
- D. Terraform will destroy the virtual machine

**Answer:** D

#### Explanation:

This is what will happen if you run terraform apply in the working directory again, after removing the resource definition from your Terraform configuration file. Terraform will detect that there is a resource in the state file that is not present in the configuration file, and will assume that you want to delete it.

#### NEW QUESTION 98

Which of the following is not a valid source path for specifying a module?

- A. source - "github.com/hashicorp/examplePref-ul.0.8M
- B. source = "./module?version=v1.6.0"
- C. source - "hashicorp/consul/aws"
- D. source - "./module"

**Answer:** B

#### Explanation:

Terraform modules are referenced by specifying a source location. This location can be a URL or a file path. However, specifying query parameters such as ?version=v1.6.0 directly within the source path is not a valid or supported method for specifying a module version in Terraform. Instead, version constraints are

specified using the version argument within the module block, not as part of the source string.

References

= This clarification is based on Terraform's official documentation regarding module usage, which outlines the correct methods for specifying module sources and versions.

#### NEW QUESTION 99

The \_\_\_\_\_ determines how Terraform creates, updates, or delete resources.

- A. Terraform configuration
- B. Terraform provisioner
- C. Terraform provider
- D. Terraform core

**Answer:** C

#### Explanation:

This is what determines how Terraform creates, updates, or deletes resources, as it is responsible for understanding API interactions with some service and exposing resources and data sources based on that API.

#### NEW QUESTION 104

Define the purpose of state in Terraform.

- A. State maps real world resources to your configuration and keeps track of metadata
- B. State lets you enforce resource configurations that relate to compliance policies
- C. State stores variables and lets you quickly reuse existing code
- D. State codifies the dependencies of related resources

**Answer:** A

#### Explanation:

The purpose of state in Terraform is to keep track of the real-world resources managed by Terraform, mapping them to the configuration. The state file contains metadata about these resources, such as resource IDs and other important attributes, which Terraform uses to plan and manage infrastructure changes. The state enables Terraform to know what resources are managed by which configurations and helps in maintaining the desired state of the infrastructure. References = This role of state in Terraform is outlined in Terraform's official documentation, emphasizing its function in mapping configuration to real-world resources and storing vital metadata .

#### NEW QUESTION 107

What is the provider for this resource?

```
resource "aws_vpc" "main" {  
    name = "test"  
}
```

- A. Vpc
- B. Test
- C. Main
- D. aws

**Answer:** D

#### Explanation:

In the given Terraform configuration snippet: resource "aws\_vpc" "main" {  
name = "test"  
}

The provider for the resource aws\_vpc is aws. The provider is specified by the prefix of the resource type. In this case, aws\_vpc indicates that the resource type vpc is provided by the aws provider.

References:

? Terraform documentation on providers: Terraform Providers

#### NEW QUESTION 108

You have multiple team members collaborating on infrastructure as code (IaC) using Terraform, and want to apply formatting standards for readability. How can you format Terraform HCL (HashiCorp Configuration Language) code according to standard Terraform style convention?

- A. Run the terraform fmt command during the code linting phase of your CI/CD process Most Voted
- B. Designate one person in each team to review and format everyone's code
- C. Manually apply two spaces indentation and align equal sign "=" characters in every Terraform file (\*.tf)
- D. Write a shell script to transform Terraform files using tools such as AWK, Python, and sed

**Answer:** A

#### Explanation:

The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability. Running this command on your configuration files before committing them to source control can help ensure consistency of style between different Terraform codebases, and can also make diffs easier to read. You can also use the -check and -diff options to check if the files are formatted and display the formatting changes respectively<sup>2</sup>. Running the terraform fmt command during the code linting phase of your CI/CD process can help automate this process and enforce the formatting standards for your team. References = [Command: fmt]<sup>2</sup>

#### NEW QUESTION 111

You have used Terraform to create an ephemeral development environment in the cloud and are now ready to destroy all the Infrastructure described by your Terraform configuration. To be safe, you would like to first see all the infrastructure that Terraform will delete. Which command should you use to show all of the resources that will be deleted? Choose two correct answers.

- A. Run terraform state rm ??
- B. Run terraform show :destroy
- C. Run terraform destroy and it will first output all the resource that will be deleted before prompting for approval
- D. Run terraform plan -destroy

**Answer:** CD

#### Explanation:

To see all the resources that Terraform will delete, you can use either of these two commands:  
? terraform destroy will show the plan of destruction and ask for your confirmation before proceeding. You can cancel the command if you do not want to destroy the resources.  
? terraform plan -destroy will show the plan of destruction without asking for confirmation. You can use this command to review the changes before running terraform destroy. References = : Destroy Infrastructure : Plan Command: Options

#### NEW QUESTION 114

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

- A. Set
- B. Map
- C. Tuple
- D. list

**Answer:** B

#### Explanation:

The Terraform collection type that should be used to store key/value pairs is map. 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. References = [Collection Types], [Map Type Constraints]

#### NEW QUESTION 119

You can configure Terraform to log to a file using the TF\_LOG environment variable.

- A. True
- B. False

**Answer:** A

#### Explanation:

You can configure Terraform to log to a file using the TF\_LOG environment variable. This variable can be set to one of the log levels: TRACE, DEBUG, INFO, WARN or ERROR. You can also use the TF\_LOG\_PATH environment variable to specify a custom log file location. References = : Debugging Terraform

#### NEW QUESTION 120

Which command lets you experiment with terraform expressions?

- A. Terraform console
- B. Terraform validate
- C. Terraform env
- D. Terraform test

**Answer:** A

#### Explanation:

This is the command that lets you experiment with Terraform expressions, by providing an interactive console that allows you to evaluate expressions and see their results. You can use this command to test your expressions before using them in your configuration files.

#### NEW QUESTION 124

You are working on some new application features and you want to spin up a copy of your production deployment to perform some quick tests. In order to avoid having to configure a new state backend, what open source Terraform feature would allow you create multiple states but still be associated with your current code?

- A. Terraform data sources
- B. Terraform local values
- C. Terraform modules
- D. Terraform workspaces

E. None of the above

**Answer:** D

**Explanation:**

Terraform workspaces allow you to create multiple states but still be associated with your current code. Workspaces are like ??environments?? (e.g. staging, production) for the same configuration. You can use workspaces to spin up a copy of your production deployment for testing purposes without having to configure a new state backend. Terraform data sources, local values, and modules are not features that allow you to create multiple states. References = Workspaces and How to Use Terraform Workspaces

**NEW QUESTION 125**

What is a key benefit of the Terraform state file?

- A. A state file can schedule recurring infrastructure tasks
- B. A state file is a source of truth for resources provisioned with Terraform
- C. A state file is a source of truth for resources provisioned with a public cloud console
- D. A state file is the desired state expressed by the Terraform code files

**Answer:** B

**Explanation:**

This is a key benefit of the Terraform state file, as it stores and tracks the metadata and attributes of the resources that are managed by Terraform, and allows Terraform to compare the current state with the desired state expressed by your configuration files.

**NEW QUESTION 127**

In a Terraform Cloud workspace linked to a version control repository, speculative plan runs start automatically when you merge or commit changes to version control.

- A. True
- B. False

**Answer:** B

**Explanation:**

In Terraform Cloud, speculative plan runs are not automatically started when changes are merged or committed to the version control repository linked to a workspace. Instead, speculative plans are typically triggered as part of proposed changes in merge requests or pull requests to give an indication of what would happen if the changes were applied, without making any real changes to the infrastructure. Actual plan and apply operations in Terraform Cloud workspaces are usually triggered by specific events or configurations defined within the Terraform Cloud workspace settings. References = This behavior is part of how Terraform Cloud integrates with version control systems and is documented in Terraform Cloud's usage guidelines and best practices, especially in the context of VCS-driven workflows.

**NEW QUESTION 129**

You are creating a Terraform configuration which needs to make use of multiple providers, one for AWS and one for Datadog. Which of the following provider blocks would allow you to do this?

A)



```
terraform {  
  provider "aws" {  
    profile = var.aws_profile  
    region  = var.aws_region  
  }  
  
  provider "datadog" {  
    api_key = var.datadog_api_key  
    app_key = var.datadog_app_key  
  }  
}
```

B)

```
provider "aws" {  
  profile = var.aws_profile  
  region  = var.aws_region  
}  
  
provider "datadog" {  
  api_key = var.datadog_api_key  
  app_key = var.datadog_app_key  
}
```

C)

```
provider "aws" {  
  profile = var.aws_profile  
  region  = var.aws_region  
}  
  
provider "datadog" {  
  api_key = var.datadog_api_key  
  app_key = var.datadog_app_key  
}
```

D)

```
provider {  
  "aws" {  
    profile = var.aws_profile  
    region  = var.aws_region  
  }  
  
  "datadog" {  
    api_key = var.datadog_api_key  
    app_key = var.datadog_app_key  
  }  
}
```

- A. Option A
- B. Option B
- C. Option C
- D. Option D

**Answer:** C

**Explanation:**

Option C is the correct way to configure multiple providers in a Terraform configuration. Each provider block must have a name attribute that specifies which provider it configures<sup>2</sup>. The other options are either missing the name attribute or using an invalid syntax.

**NEW QUESTION 133**

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

- 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.

**NEW QUESTION 137**

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

- A. Plan
- B. Import
- C. Alidate
- D. Init
- E. apply

**Answer:** DE

**Explanation:**

The two steps that are required to provision new infrastructure in the Terraform workflow are init and apply. The terraform init command 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. The terraform apply command 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. References = [The Core Terraform Workflow], [Initialize a Terraform working directory with init], [Apply Terraform Configuration with apply]

**NEW QUESTION 141**

How does Terraform manage most dependencies between resources?

- A. Terraform will automatically manage most resource dependencies
- B. Using the depends\_on parameter
- C. By defining dependencies as modules and including them in a particular order
- D. The order that resources appear in Terraform configuration indicates dependencies

**Answer:** A

**Explanation:**

This is how Terraform manages most dependencies between resources, by using the references between them in the configuration files. For example, if resource A depends on resource B, Terraform will create resource B first and then pass its attributes to resource A.

**NEW QUESTION 144**

How can a ticket-based system slow down infrastructure provisioning and limit the ability to scale? Choose two correct answers.

- A. End-users have to request infrastructure changes
- B. Ticket based systems generate a full audit trail of the request and fulfillment process
- C. Users can access catalog of approved resources from drop down list in a request form
- D. The more resources your organization needs, the more tickets your infrastructure team has to process

**Answer:** A

**Explanation:**

These are some of the ways that a ticket-based system can slow down infrastructure provisioning and limit the ability to scale, as they introduce delays, bottlenecks, and manual interventions in the process of creating and modifying infrastructure.

**NEW QUESTION 145**

Where does the Terraform local backend store its state?

- A. In the terraform file
- B. In the /tmp directory
- C. In the terraform.tfstate file
- D. In the user's terraform.state file

**Answer:** C

**Explanation:**

This is where the Terraform local backend stores its state, by default, unless you specify a different file name or location in your configuration. The local backend is the simplest backend type that stores the state file on your local disk.

**NEW QUESTION 146**

When does Sentinel enforce policy logic during a Terraform Cloud run?

- A. Before the plan phase
- B. During the plan phase
- C. Before the apply phase
- D. After the apply phase

**Answer:** C

**Explanation:**

Sentinel policies are checked after the plan stage of a Terraform run, but before it can be confirmed or the terraform apply is executed<sup>3</sup>. This allows you to enforce rules on your infrastructure before it is created or modified.

**NEW QUESTION 149**

Which backend does the Terraform CU use by default?

- A. Depends on the cloud provider configured
- B. HTTP
- C. Remote
- D. Terraform Cloud
- E. Local

**Answer:** E

**Explanation:**

This is the backend that the Terraform CLI uses by default, unless you specify a different backend in your configuration. The local backend stores the state file in a local file named terraform.tfstate, which can be used to track and manage the state of your infrastructure.

**NEW QUESTION 151**

When should you run terraform init?

- A. Every time you run terraform apply
- B. Before you start coding a new Terraform project
- C. After you run terraform plan for the time in a new terraform project and before you run terraform apply
- D. After you start coding a new terraform project and before you run terraform plan for the first time.

**Answer:** D

**Explanation:**

You should run terraform init after you start coding a new Terraform project and before you run terraform plan for the first time. This command will initialize the working directory by downloading the required providers and modules, creating the initial state file, and performing other necessary tasks. References = : Initialize a Terraform Project

**NEW QUESTION 154**

Multiple team members are collaborating on infrastructure using Terraform and want to format the\* Terraform code following standard Terraform-style convention. How should they ensure the code satisfies conventions?

- A. Terraform automatically formats configuration on terraform apply
- B. Run terraform validate prior to executing terraform plan or terraform apply
- C. Use terraform fmt
- D. Replace all tabs with spaces

**Answer:** C

**Explanation:**

The terraform fmt command is used to format Terraform configuration files to a canonical format and style. This ensures that all team members are using a consistent style, making the code easier to read and maintain. It automatically applies Terraform's standard formatting conventions to your configuration files, helping maintain consistency across the team's codebase.

References:

? Terraform documentation on terraform fmt: Terraform Fmt

**NEW QUESTION 156**

How can terraform plan aid in the development process?

- A. Initializes your working directory containing your Terraform configuration files
- B. Validates your expectations against the execution plan without permanently modifying state
- C. Formats your Terraform configuration files
- D. Reconciles Terraform's state against deployed resources and permanently modifies state using the current status of deployed resources

**Answer:** B

**Explanation:**

The terraform plan command is used to create an execution plan. It allows you to see what actions Terraform will take to reach the desired state defined in your configuration files. It evaluates the current state and configuration, showing a detailed outline of the resources that will be created, updated, or destroyed. This is a critical step in the development process as it helps you verify that the changes you are about to apply will perform as expected, without actually modifying any state or infrastructure.

References:

? Terraform documentation on terraform plan: Terraform Plan

**NEW QUESTION 159**

Infrastructure as Code (IaC) can be stored in a version control system along with application code.

- A. True
- B. False

**Answer:** A

**Explanation:**

Infrastructure as Code (IaC) can indeed be stored in a version control system along with application code. This practice is a fundamental principle of modern infrastructure management, allowing teams to apply software development practices like versioning, peer review, and CI/CD to infrastructure management. Storing IaC configurations in version control facilitates collaboration, history tracking, and change management. References = While this concept is a foundational aspect of IaC and is widely accepted in the industry, direct references from the HashiCorp Terraform Associate (003) study materials were not found in the provided files. However, this practice is encouraged in Terraform's best practices and various HashiCorp learning resources.

**NEW QUESTION 161**

Why does this backend configuration not follow best practices?

```
terraform {  
  backend "s3" {  
    bucket      = "terraform-state-prod"  
    key         = "network/terraform.tfstate"  
    region      = "us-east-1"  
    access_key  = "AKIAIOSFODNN7EXAMPLE"  
    secret_key  = "wJalrXUtnFEMI/K7MDENG/bPxrF1CYEXAMPLEKEY"  
  }  
  
  required_providers {  
    aws = {  
      source  = "hashicorp/aws"  
      version = "~> 3.38"  
    }  
  }  
  
  required_version = ">= 0.15"  
}
```

- A. An alias meta-argument should be included in backend blocks whenever possible
- B. You should use the local enhanced storage backend whenever possible
- C. You should not store credentials in Terraform configuration
- D. The backend configuration should contain multiple credentials so that more than one user can execute terraform plan and terraform apply

**Answer:** C

**Explanation:**

This is a bad practice, as it exposes your credentials to anyone who can access your configuration files or state files. You should use environment variables, credential files, or other mechanisms to provide credentials to Terraform.

**NEW QUESTION 162**

Which of the following arguments are required when declaring a Terraform output?

- A. value
- B. description
- C. default
- D. sensitive

**Answer:** A

**Explanation:**

When declaring a Terraform output, the value argument is required. Outputs are a way to extract information from Terraform-managed infrastructure, and the value argument specifies what data will be outputted. While other arguments like description and sensitive can provide additional context or security around the output, value is the only mandatory argument needed to define an output. References = The requirement of the value argument for outputs is specified in Terraform's official documentation, which provides guidelines on defining and using outputs in Terraform configurations.

**NEW QUESTION 165**

Which of the following should you put into the required\_providers block?

- A. version >= 3.1
- B. version = ??>= 3.1??



C. version ~> 3.1

**Answer:** B

**Explanation:**

The required\_providers block is used to specify the provider versions that the configuration can work with. The version argument accepts a version constraint string, which must be enclosed in double quotes. The version constraint string can use operators such as >=, ~>, =, etc. to specify the minimum, maximum, or exact version of the provider. For example, version = ">= 3.1" means that the configuration can work with any provider version that is 3.1 or higher. References = [Provider Requirements] and [Version Constraints]

**NEW QUESTION 168**

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

- 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.

**NEW QUESTION 173**

You have declared a variable called var.list which is a list of objects that all have an attribute id . Which options will produce a list of the IDs? Choose two correct answers.

- A. [ var.list [ \* ] , id ]
- B. [ for o in var.list : o.id ]
- C. var.list[\*].id
- D. { for o in var.list : o => o.id }

**Answer:** BC

**Explanation:**

These are two ways to produce a list of the IDs from a list of objects that have an attribute id, using either a for expression or a splat expression syntax.

**NEW QUESTION 174**

Which parameters does terraform import require? Choose two correct answers.

- A. Provider
- B. Resource ID
- C. Resource address
- D. Path

**Answer:** BC

**Explanation:**

These are the parameters that terraform import requires, as they allow Terraform to identify the existing resource that you want to import into your state file, and match it with the corresponding configuration block in your files.

**NEW QUESTION 175**

You must use different Terraform commands depending on the cloud provider you use.

- A. True
- B. False

**Answer:** B

**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.

**NEW QUESTION 180**

One remote backend configuration always maps to a single remote workspace.

- A. True
- B. False

**Answer:** A

**Explanation:**

The remote backend can work with either a single remote Terraform Cloud workspace, or with multiple similarly-named remote workspaces (like networking-dev and networking-prod). The workspaces block of the backend configuration determines which mode it uses. To use a single remote Terraform Cloud workspace, set

workspaces.name to the remote workspace's full name (like networking-prod). To use multiple remote workspaces, set workspaces.prefix to a prefix used in all of the desired remote workspace names. For example, set prefix = networking- to use Terraform cloud workspaces with names like networking-dev and networking-prod. This is helpful when mapping multiple Terraform CLI workspaces used in a single Terraform configuration to multiple Terraform Cloud workspaces<sup>3</sup>. However, one remote backend configuration always maps to a single remote workspace, either by name or by prefix. You cannot use both name and prefix in the same backend configuration, or omit both. Doing so will result in a configuration error<sup>3</sup>. References = [Backend Type: remote]<sup>3</sup>

#### NEW QUESTION 185

Which of the following is not a benefit of adopting infrastructure as code?

- A. Versioning
- B. A Graphical User Interface
- C. Reusability of code
- D. Automation

**Answer:** B

#### Explanation:

Infrastructure as Code (IaC) provides several benefits, including the ability to version control infrastructure, reuse code, and automate infrastructure management. However, IaC is typically associated with declarative configuration files and does not inherently provide a graphical user interface (GUI). A GUI is a feature that may be provided by specific tools or platforms built on top of IaC principles but is not a direct benefit of IaC itself<sup>1</sup>.

References = The benefits of IaC can be verified from the official HashiCorp documentation on "What is Infrastructure as Code with Terraform" provided by HashiCorp Developer<sup>1</sup>.

#### NEW QUESTION 190

Which of the following is not true of Terraform providers?

- A. An individual person can write a Terraform Provider
- B. A community of users can maintain a provider
- C. HashiCorp maintains some providers
- D. Cloud providers and infrastructure vendors can write, maintain, or collaborate on Terraform
- E. providers
- F. None of the above

**Answer:** F

#### Explanation:

All of the statements are true of Terraform providers. Terraform providers are plugins that enable Terraform to interact with various APIs and services<sup>1</sup>. Anyone can write a Terraform provider, either as an individual or as part of a community<sup>2</sup>. HashiCorp maintains some providers, such as the AWS, Azure, and Google Cloud providers<sup>3</sup>. Cloud providers and infrastructure vendors can also write, maintain, or collaborate on Terraform providers, such as the VMware, Oracle, and Alibaba Cloud providers. References =

- <sup>1</sup>: Providers - Configuration Language | Terraform | HashiCorp Developer
- <sup>2</sup>: Plugin Development - How Terraform Works With Plugins | Terraform | HashiCorp Developer
- <sup>3</sup>: Terraform Registry
- : Terraform Registry

#### NEW QUESTION 195

Any user can publish modules to the public Terraform Module Registry.

- A. True
- B. False

**Answer:** A

#### Explanation:

The Terraform Registry allows any user to publish and share modules. Published modules support versioning, automatically generate documentation, allow browsing version histories, show examples and READMEs, and more. Public modules are managed via Git and GitHub, and publishing a module takes only a few minutes. Once a module is published, releasing a new version of a module is as simple as pushing a properly formed Git tag<sup>1</sup>.

References = The information can be verified from the Terraform Registry documentation on Publishing Modules provided by HashiCorp Developer<sup>1</sup>.

#### NEW QUESTION 198

What does terraform import do?

- A. Imports existing resources into the state file
- B. Imports all infrastructure from a given cloud provider
- C. Imports a new Terraform module
- D. Imports clean copies of tainted resources
- E. None of the above

**Answer:** A

#### Explanation:

The terraform import command is used to import existing infrastructure into your Terraform state. This command takes the existing resource and associates it with a resource defined in your Terraform configuration, updating the state file accordingly. It does not generate configuration for the resource, only the state.

#### NEW QUESTION 199

You can reference a resource created with for\_each using a Splat ( \*) expression.

- A. True

B. False

**Answer:** B

**Explanation:**

You cannot reference a resource created with `for_each` using a splat (\*) expression, as it will not work with resources that have non-numeric keys. You need to use a `for` expression instead to iterate over the resource instances.

**NEW QUESTION 204**

Which of the following module source paths does not specify a remote module?

- A. Source = `??module/consul????`
- B. Source = `????github.comicrop/example????`
- C. Source = `????git@github.com:hasicrop/example.git????`
- D. Source = `????hasicrop/consul/aws????`

**Answer:** A

**Explanation:**

The module source path that does not specify a remote module is `source = "module/consul"`. This specifies a local module, which is a module that is stored in a subdirectory of the current working directory. The other options are all examples of remote modules, which are modules that are stored outside of the current working directory and can be accessed by various protocols, such as Git, HTTP, or the Terraform Registry. Remote modules are useful for sharing and reusing code across different configurations and environments. References = [Module Sources], [Local Paths], [Terraform Registry], [Generic Git Repository], [GitHub]

**NEW QUESTION 208**

FILL IN THE BLANK

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.

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

The name of the default file where Terraform stores the state is `terraform.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. References = [Terraform State], [State File Format]

**NEW QUESTION 213**

Which of these statements about Terraform Cloud workspaces is false?

- A. They have role-based access controls
- B. You must use the CLI to switch between workspaces
- C. Plans and applies can be triggered via version control system integrations
- D. They can securely store cloud credentials

**Answer:** B

**Explanation:**

The statement that you must use the CLI to switch between workspaces is false. Terraform Cloud workspaces are different from Terraform CLI workspaces. Terraform Cloud workspaces are required and represent all of the collections of infrastructure in an organization. They are also a major component of role-based access in Terraform Cloud. You can grant individual users and user groups permissions for one or more workspaces that dictate whether they can manage variables, perform runs, etc. You can create, view, and switch between Terraform Cloud workspaces using the Terraform Cloud UI, the Workspaces API, or the Terraform Enterprise Provider<sup>5</sup>. Terraform CLI workspaces are optional and allow you to create multiple distinct instances of a single configuration within one working directory. They are useful for creating disposable environments for testing or experimenting without affecting your main or production environment. You can create, view, and switch between Terraform CLI workspaces using the `terraform workspace` command<sup>6</sup>. The other statements about Terraform Cloud workspaces are true. They have role-based access controls that allow you to assign permissions to users and teams based on their roles and responsibilities. You can create and manage roles using the Teams API or the Terraform Enterprise Provider<sup>7</sup>. Plans and applies can be triggered via version control system integrations that allow you to link your Terraform Cloud workspaces to your VCS repositories. You can configure VCS settings, webhooks, and branch tracking to automate your Terraform Cloud workflow<sup>8</sup>. They can securely store cloud credentials as sensitive variables that are encrypted at rest and only decrypted when needed. You can manage variables using the Terraform Cloud UI, the Variables API, or the Terraform Enterprise Provider<sup>9</sup>. References = [Workspaces]<sup>5</sup>, [Terraform CLI Workspaces]<sup>6</sup>, [Teams and Organizations]<sup>7</sup>, [VCS Integration]<sup>8</sup>, [Variables]<sup>9</sup>

**NEW QUESTION 216**

.....

## THANKS FOR TRYING THE DEMO OF OUR PRODUCT

Visit Our Site to Purchase the Full Set of Actual Terraform-Associate-003 Exam Questions With Answers.

We Also Provide Practice Exam Software That Simulates Real Exam Environment And Has Many Self-Assessment Features. Order the Terraform-Associate-003 Product From:

<https://www.2passeasy.com/dumps/Terraform-Associate-003/>

## Money Back Guarantee

### **Terraform-Associate-003 Practice Exam Features:**

- \* Terraform-Associate-003 Questions and Answers Updated Frequently
- \* Terraform-Associate-003 Practice Questions Verified by Expert Senior Certified Staff
- \* Terraform-Associate-003 Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- \* Terraform-Associate-003 Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year