

# HashiCorp

## Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)



#### NEW QUESTION 1

Terraform configuration can only import modules from the public registry.

- A. True
- B. False

**Answer:** B

#### Explanation:

Terraform configuration can import modules from various sources, not only from the public registry. Modules can be sourced from local file paths, Git repositories, HTTP URLs, Mercurial repositories, S3 buckets, and GCS buckets. Terraform supports a number of common conventions and syntaxes for specifying module sources, as documented in the [Module Sources] page. References = [Module Sources]

#### NEW QUESTION 2

While attempting to deploy resources into your cloud provider using Terraform, you begin to see some odd behavior and experience slow responses. In order to troubleshoot you decide to turn on Terraform debugging. Which environment variables must be configured to make Terraform's logging more verbose?

- A. TF\_LOG\_PAIRH
- B. TF\_LOG
- C. TF\_VAR\_log\_path
- D. TF\_VAR\_log\_level

**Answer:** B

#### Explanation:

To make Terraform's logging more verbose for troubleshooting purposes, you must configure the TF\_LOG environment variable. This variable controls the level of logging and can be set to TRACE, DEBUG, INFO, WARN, or ERROR, with TRACE providing the most verbose output. References = Detailed debugging instructions and the use of environment variables like TF\_LOG for increasing verbosity are part of Terraform's standard debugging practices

#### NEW QUESTION 3

Your DevOps team is currently using the local backend for your Terraform configuration. You would like to move to a remote backend to store the state file in a central location. Which of the following backends would not work?

- A. Artifactory
- B. Amazon S3
- C. Terraform Cloud
- D. Git

**Answer:** D

#### Explanation:

This is not a valid backend for Terraform, as it does not support locking or versioning of state files<sup>4</sup>. The other options are valid backends that can store state files in a central location.

#### NEW QUESTION 4

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 5

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 6

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 7**

You should run terraform fmt 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 8**

What feature stops multiple users from operating on the Terraform state at the same time?

- A. State locking
- B. Version control
- C. Provider constraints
- D. Remote backends

**Answer:** A

**Explanation:**

State locking prevents other users from modifying the state file while a Terraform operation is in progress. This prevents conflicts and data loss<sup>1</sup>.

**NEW QUESTION 9**

If a DevOps team adopts AWS CloudFormation as their standardized method for provisioning public cloud resources, which of the following scenarios poses a challenge for this team?

- A. The team is asked to manage a new application stack built on AWS-native services
- B. The organization decides to expand into Azure wishes to deploy new infrastructure
- C. The team is asked to build a reusable code based that can deploy resources into any AWS region
- D. The DevOps team is tasked with automating a manual, web console-based provisioning.

**Answer:** B

**Explanation:**

This is the scenario that poses a challenge for this team, if they adopt AWS CloudFormation as their standardized method for provisioning public cloud resources, as CloudFormation only supports AWS services and resources, and cannot be used to provision infrastructure on other cloud platforms such as Azure.

**NEW QUESTION 10**

Running terraform fmt without any flags in a directory with Terraform configuration files check the formatting of those files without changing their contents.

- A. True
- B. False

**Answer:** B

**Explanation:**

Running terraform fmt without any flags in a directory with Terraform configuration files will not check the formatting of those files without changing their contents, but will actually rewrite them to a canonical format and style. If you want to check the formatting without making changes, you need to use the -check flag.

**NEW QUESTION 10**

What information does the public Terraform Module Registry automatically expose about published modules?

- A. Required input variables
- B. Optional inputs variables and default values
- C. Outputs
- D. All of the above
- E. None of the above

**Answer:** D

**Explanation:**

The public Terraform Module Registry automatically exposes all the information about published modules, including required input variables, optional input variables and default values, and outputs. This helps users to understand how to use and configure the modules.

**NEW QUESTION 12**

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 13**

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 17**

If you manually destroy infrastructure, what is the best practice reflecting this change in Terraform?

- A. Run terraform refresh
- B. It will happen automatically
- C. Manually update the state file
- D. Run terraform import

**Answer:** B

**Explanation:**

If you manually destroy infrastructure, Terraform will automatically detect the change and update the state file during the next plan or apply. Terraform compares the current state of the infrastructure with the desired state in the configuration and generates a plan to reconcile the differences. If a resource is missing from the infrastructure but still exists in the state file, Terraform will attempt to recreate it. If a resource is present in the infrastructure but not in the state file, Terraform will ignore it unless you use the terraform import command to bring it under Terraform's management. References = [Terraform State]

**NEW QUESTION 20**

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 25**

Module version is required to reference a module on the Terraform Module Registry.

- A. True
- B. False

**Answer:** B

**Explanation:**

Module version is optional to reference a module on the Terraform Module Registry. If you omit the version constraint, Terraform will automatically use the latest available version of the module

**NEW QUESTION 27**

Once you configure a new Terraform backend with a terraform code block, which command(s) should you use to migrate the state file?

- A. terraform destroy, then terraform apply
- B. terraform init
- C. terraform push
- D. terraform apply

**Answer:** A

**Explanation:**

This command will initialize the new backend and prompt you to migrate the existing state file to the new location4. The other commands are not relevant for this task.

**NEW QUESTION 31**

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 34**

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 37**

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 .destory

**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 42**

How can you trigger a run in a Terraform Cloud workspace that is connected to a Version Control System (VCS) repository?

- A. Only Terraform Cloud organization owners can set workspace variables on VCS connected workspaces
- B. Commit a change to the VCS working directory and branch that the Terraform Cloud workspace is connected to
- C. Only Terraform Cloud organization owners can approve plans in VCS connected workspaces
- D. Only members of a VCS organization can open a pull request against repositories that are connected to Terraform Cloud workspaces

**Answer:** B

**Explanation:**

This will trigger a run in the Terraform Cloud workspace, which will perform a plan and apply operation on the infrastructure defined by the Terraform configuration files in the VCS repository.

**NEW QUESTION 45**

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 48**

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 52**

Variables declared within a module are accessible outside of the module.

- A. True
- B. False

**Answer:** B

**Explanation:**

Variables declared within a module are only accessible within that module, unless they are explicitly exposed as output values<sup>1</sup>.

**NEW QUESTION 56**

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 58**

A developer accidentally launched a VM (virtual machine) outside of the Terraform workflow and ended up with two servers with the same name. They don't know which VM Terraform manages but do have a list of all active VM IDs.

Which of the following methods could you use to discover which instance Terraform manages?

- A. Run terraform state list to find the names of all VMs, then run terraform state show for each of them to find which VM ID Terraform manages
- B. Update the code to include outputs for the ID of all VMs, then run terraform plan to view the outputs
- C. Run terraform taint/code on all the VMs to recreate them
- D. Use terraform refresh/code to find out which IDs are already part of state

**Answer:** A

**Explanation:**

The terraform state list command lists all resources that are managed by Terraform in the current state file1. The terraform state show command shows the attributes of a single resource in the state file2. By using these two commands, you can compare the VM IDs in your list with the ones in the state file and identify which one is managed by Terraform.

**NEW QUESTION 61**

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 63**

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 65**

Where can Terraform not load a provider from?

- A. Plugins directory
- B. Provider plugin cache
- C. Official HashCorp Distribution on releases.hashicorp.com
- D. Source code

**Answer:** D

**Explanation:**

This is where Terraform cannot load a provider from, as it requires a compiled binary file that implements the provider protocol. You can load a provider from a plugins directory, a provider plugin cache, or the official HashiCorp distribution on releases.hashicorp.com.

**NEW QUESTION 69**

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 72**

Which task does terraform init not perform?

- A. Validates all required variables are present
- B. Sources any modules and copies the configuration locally
- C. Connects to the backend
- D. Sources all providers used in the configuration and downloads them

**Answer:** A

**Explanation:**

The terraform init command is used to initialize a working directory containing Terraform configuration files. This command performs several different initialization steps to prepare the current working directory for use with Terraform, which includes initializing the backend, installing provider plugins, and copying any modules referenced in the configuration. However, it does not validate whether all required variables are present; that is a task performed by terraform plan or terraform apply<sup>1</sup>.

References = This information can be verified from the official Terraform documentation on the terraform init command provided by HashiCorp Developer<sup>1</sup>.

**NEW QUESTION 75**

Where in your Terraform configuration do you specify a state backend?

- A. The resource block
- B. The data source block
- C. The terraform block
- D. The provider block

**Answer:** C

**Explanation:**

In Terraform, the backend configuration, which includes details about where and how state is stored, is specified within the terraform block of your configuration. This block is the correct place to define the backend type and its configuration parameters, such as the location of the state file for a local backend or the bucket details for a remote backend like S3. References = This practice is outlined in Terraform's core documentation, which provides examples and guidelines on how to configure various aspects of Terraform's behavior, including state backends .

**NEW QUESTION 77**

You have a list of numbers that represents the number of free CPU cores on each virtual cluster:



```
numcpus = [ 18, 3, 7, 11, 2 ]
```

What Terraform function could you use to select the largest number from the list?

- A. top(numcpus)
- B. max(numcpus)
- C. ceil (numcpus)
- D. hight[numcpus]

**Answer:** B

**Explanation:**

In Terraform, the max function can be used to select the largest number from a list of numbers. The max function takes multiple arguments and returns the highest one. For the list numcpus = [18, 3, 7, 11, 2], using max(numcpus...) will return 18, which is the largest number in the list.

References:

? Terraform documentation on max function: Terraform Functions - max

**NEW QUESTION 80**

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 81**

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/bPxrFiCYEXAMPLEKEY"  
  }  
  
  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 83**

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 86**

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?

- 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 any terraform plan or terraform apply operation. 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. References = [Sentinel Policy Framework], [Manage Policies in Terraform Cloud], [Write and Test Sentinel Policies for Terraform]

#### NEW QUESTION 90

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 93

When using multiple configuration of the same Terraform provider, what meta-argument must you include in any non-default provider configurations?

- A. Alias
- B. Id
- C. Depends\_on
- D. name

**Answer:** A

#### Explanation:

This is the meta-argument that you must include in any non-default provider configurations, as it allows you to give a friendly name to the configuration and reference it in other parts of your code. The other options are either invalid or irrelevant for this purpose.

#### NEW QUESTION 97

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 102

What does Terraform use the .terraform.lock.hcl file for?

- A. There is no such file
- B. Tracking specific provider dependencies
- C. Preventing Terraform runs from occurring
- D. Storing references to workspaces which are locked

**Answer:** B

#### Explanation:

The .terraform.lock.hcl file is a new feature in Terraform 0.14 that records the exact versions of each provider used in your configuration. This helps ensure consistent and reproducible behavior across different machines and runs.

#### NEW QUESTION 103

All modules published on the official Terraform Module Registry have been verified by HashiCorp.

- A. True
- B. False

**Answer:** B

#### Explanation:

Not all modules published on the official Terraform Module Registry have been verified by HashiCorp. While HashiCorp verifies some modules, there are many community-contributed modules that are not verified. Verified modules have a "Verified" badge indicating that HashiCorp has reviewed them for security and best practices, but the registry also includes unverified modules.

References:

? Terraform Module Registry documentation: Terraform Registry

#### NEW QUESTION 105

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

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

**Answer:** AD

**Explanation:**

Terraform Cloud includes several features designed to enhance collaboration and infrastructure management. Two of these features are:

? A web-based user interface (UI): This allows users to interact with Terraform Cloud

through a browser, providing a centralized interface for managing Terraform configurations, state files, and workspaces.

? Remote state storage: This feature enables users to store their Terraform state

files remotely in Terraform Cloud, ensuring that state is safely backed up and can be accessed by team members as needed.

**NEW QUESTION 108**

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 111**

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 113**

You have created a main.tf Terraform configuration consisting of an application server, a database and a load balanced. You ran terraform apply and Terraform created all of the resources successfully.

Now you realize that you do not actually need the load balancer, so you run terraform destroy without any flags. What will happen?

- A. Terraform will prompt you to pick which resource you want to destroy
- B. Terraform will destroy the application server because it is listed first in the code
- C. Terraform will prompt you to confirm that you want to destroy all the infrastructure
- D. Terraform will destroy the main, tf file
- E. Terraform will immediately destroy all the infrastructure

**Answer:** C

**Explanation:**

This is what will happen if you run terraform destroy without any flags, as it will attempt to delete all the resources that are associated with your current working directory or workspace. You can use the -target flag to specify a particular resource that you want to destroy.

**NEW QUESTION 116**

What does Terraform not reference when running a terraform apply -refresh-only ?

- A. State file
- B. Credentials
- C. Cloud provider
- D. Terraform resource definitions in configuration files

**Answer:** D

**Explanation:**

When running a terraform apply -refresh-only, Terraform does not reference the configuration files, but only the state file, credentials, and cloud provider. The purpose of this command is to update the state file with the current status of the real resources, without making any changes to them<sup>1</sup>.

**NEW QUESTION 118**

Which command add existing resources into Terraform state?

- A. Terraform init
- B. Terraform plan
- C. Terraform refresh

- D. Terraform import
- E. All of these

**Answer:** D

**Explanation:**

This is the command that can add existing resources into Terraform state, by matching them with the corresponding configuration blocks in your files.

**NEW QUESTION 121**

terraform plan updates your state file.

- A. True
- B. False

**Answer:** B

**Explanation:**

The terraform plan command does not update the state file. Instead, it reads the current state and the configuration files to determine what changes would be made to bring the real-world infrastructure into the desired state defined in the configuration. The plan operation is a read-only operation and does not modify the state or the infrastructure. It is the terraform apply command that actually applies changes and updates the state file. References = Terraform's official guidelines and documentation clarify the purpose of the terraform plan command, highlighting its role in preparing and showing an execution plan without making any changes to the actual state or infrastructure .

**NEW QUESTION 123**

You can access state stored with the local backend by using terraform\_remote\_state data source.

- A. True
- B. False

**Answer:** B

**Explanation:**

You cannot access state stored with the local backend by using the terraform\_remote\_state data source. The terraform\_remote\_state data source is used to retrieve the root module output values from some other Terraform configuration using the latest state snapshot from the remote backend. It requires a backend that supports remote state storage, such as S3, Consul, AzureRM, or GCS. The local backend stores the state file locally on the filesystem, which terraform\_remote\_state cannot access. References:

? Terraform documentation on terraform\_remote\_state data source: Terraform

Remote State Data Source

? Example usage of remote state: Example Usage (remote Backend)

**NEW QUESTION 124**

Which are examples of infrastructure as code? Choose two correct answers.

- A. Cloned virtual machine images
- B. Versioned configuration files
- C. Change management database records
- D. Doctor files

**Answer:** B

**Explanation:**

These are examples of infrastructure as code (IaC), which is a practice of managing and provisioning infrastructure through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

**NEW QUESTION 129**

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 134**

How would you output returned values from a child module in the Terraform CLI output?

- A. Declare the output in the root configuration
- B. Declare the output in the child module
- C. Declare the output in both the root and child module
- D. None of the above

**Answer:** C

**Explanation:**

To output returned values from a child module in the Terraform CLI output, you need to declare the output in both the child module and the root module. The child module output will return the value to the root module, and the root module output will display the value in the CLI. References = [Terraform Outputs]

**NEW QUESTION 136**

.....



## Thank You for Trying Our Product

### We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

### 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

**100% Actual & Verified — Instant Download, Please Click**  
**[Order The Terraform-Associate-003 Practice Test Here](#)**