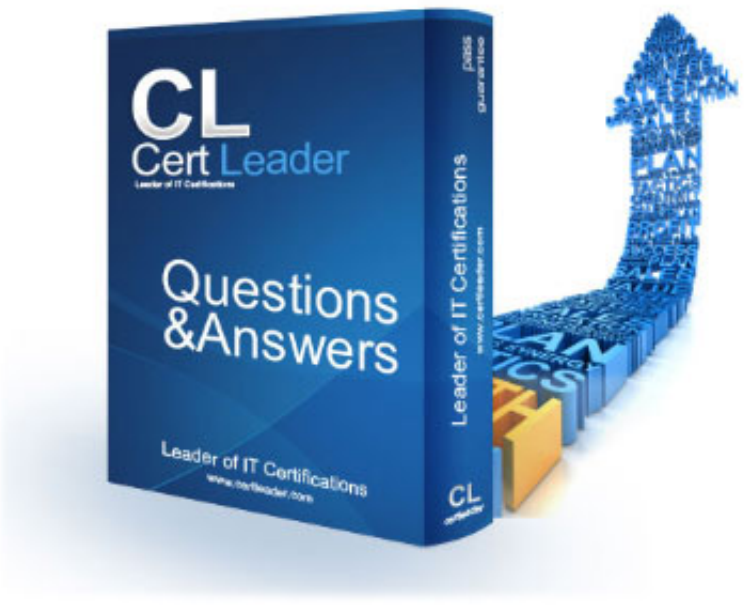
# Terraform-Associate-003 Dumps

# HashiCorp Certified: Terraform Associate (003)

## https://www.certleader.com/Terraform-Associate-003-dumps.html



**NEW QUESTION 1**
You've used Terraform to deploy a virtual machine and a database. You want to replace this virtual machine instance with an identical one without affecting the database. What is the best way to achieve this using Terraform?

A. Use the terraform state rm command to remove the VM from state file
B. Use the terraform taint command targeting the VMs then run terraform plan and terraform apply
C. Use the terraform apply command targeting the VM resources only
D. Delete the Terraform VM resources from your Terraform code then run terraform plan and terraform apply

**Answer:** B

**Explanation:**
The terraform taint command marks a resource as tainted, which means it will be destroyed and recreated on the next apply. This way, you can replace the VM instance without affecting the database or other resources. References = [Terraform Taint]

**NEW QUESTION 2**
A terraform apply can not infrastructure.

A. change
B. destroy
C. provision
D. import

**Answer:** D

**Explanation:**
The terraform import command is used to import existing infrastructure into Terraform??s state. This allows Terraform to manage and destroy the imported infrastructure as part of the configuration. The terraform import command does not modify the configuration, so the imported resources must be manually added to the configuration after the import. References = [Importing Infrastructure]

**NEW QUESTION 3**
Changing the Terraform backend from the default "local" backend to a different one after performing your first terrafom apply is:

A. Optional
B. Impossible
C. Mandatory
D. Discouraged

**Answer:** D

**Explanation:**
Changing the Terraform backend after performing the initial terraform apply is technically possible but strongly discouraged. This is because changing backends can lead to complexities in state management, requiring manual intervention such as state migration to ensure consistency. Terraform's documentation and best practices advise planning the backend configuration carefully before applying Terraform configurations to avoid such changes.References = This guidance is consistent with Terraform's official documentation, which recommends careful consideration and planning of backend configurations to avoid the need for changes.

**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**
You much initialize your working directory before running terraform validate.

A. True
B. False

**Answer:** A

**Explanation:**
You must initialize your working directory before running terraform validate, as it will ensure that all the required plugins and modules are installed and configured properly. If you skip this step, you may encounter errors or inconsistencies when validating
your configuration files.

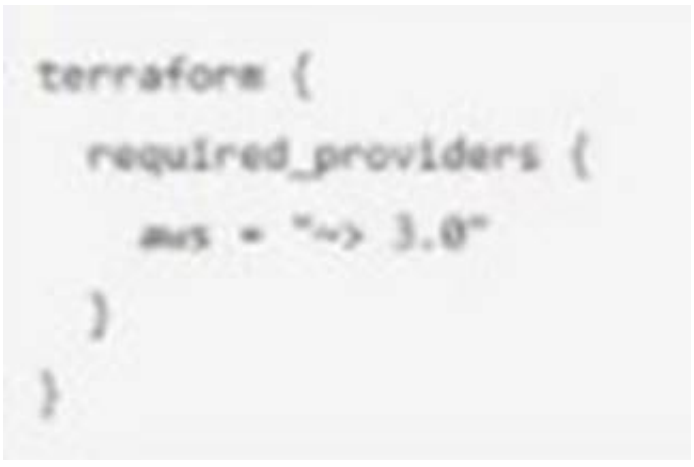
**NEW QUESTION 7**
Why would you use the -replace flag for terraform apply?

A. You want Terraform to ignore a resource on the next apply
B. You want Terraform to destroy all the infrastructure in your workspace
C. You want to force Terraform to destroy a resource on the next apply
D. You want to force Terraform to destroy and recreate a resource on the next apply

**Answer:** D

**Explanation:**
The -replace flag is used with the terraform apply command when there is a need to explicitly force Terraform to destroy and then recreate a specific resource during the next apply. This can be necessary in situations where a simple update is insufficient or when a resource must be re-provisioned to pick up certain changes.

**NEW QUESTION 8**
What does this code do?




A. Requires any version of the AWS provider > = 3.0 and <4.0
B. Requires any version of the AWS provider >= 3.0
C. Requires any version of the AWS provider > = 3.0 major releas
D. like 4.1
E. Requires any version of the AWS provider > 3.0

**Answer:** A

**Explanation:**
This is what this code does, by using the pessimistic constraint operator (~>), which specifies an acceptable range of versions for a provider or module.

**NEW QUESTION 9**
When should you use the force-unlock command?

A. You have a high priority change
B. Automatic unlocking failed
C. apply failed due to a state lock
D. You see a status message that you cannot acquire the lock

**Answer:** B

**Explanation:**
You should use the force-unlock command when automatic unlocking failed. Terraform will lock your state for all operations that could write state, such as plan, apply, or destroy. This prevents others from acquiring the lock and potentially corrupting your state. State locking happens automatically on all operations that could write state and you won??t see any message that it is happening. If state locking fails, Terraform will not continue. You can disable state locking for most commands with the -lock flag but it is not recommended. If acquiring the lock is taking longer than expected, Terraform will output a status message. If Terraform doesn??t output a message, state locking is still occurring if your backend supports it. Terraform has a force-unlock command to manually unlock the state if unlocking failed. Be very careful with this command. If you unlock the state when someone else is holding the lock it could cause multiple writers. Force unlock should only be used to unlock your own lock in the situation where automatic unlocking failed. To protect you, the force-unlock command requires a unique lock ID. Terraform will output this lock ID if unlocking fails. This lock ID acts as a nonce, ensuring that locks and unlocks target the correct lock. The other situations are not valid reasons to use the force-unlock command. You should not use the force-unlock command if you have a high priority change, if apply failed due to a state lock, or if you see a status message that you cannot acquire the lock. These situations indicate that someone else is holding the lock and you should wait for them to finish their operation or contact them to resolve the issue. Using the force-unlock command in these cases could result in data loss or inconsistency. References = [State Locking], [Command: force-unlock]

**NEW QUESTION 10**
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_PAIH
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 10**
terraform validate confirms that your infrastructure matches the Terraform state file.

A. True
B. False

**Answer:** B

**Explanation:**
terraform validate does not confirm that your infrastructure matches the Terraform state file. It only checks whether the configuration files in a directory are syntactically valid and internally consistent3. To confirm that your infrastructure matches the Terraform state file, you need to use terraform plan or terraform apply with the -refresh- only option.

**NEW QUESTION 12**
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)


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

**Answer:** B

**NEW QUESTION 13**
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 14**
Which of the following statements about Terraform modules is not true?

A. Modules can call other modules
B. A module is a container for one or more resources
C. Modules must be publicly accessible
D. You can call the same module multiple times

**Answer:** C

**Explanation:**
This is not true, as modules can be either public or private, depending on your needs and preferences. You can use the Terraform Registry to publish and consume public modules, or use Terraform Cloud or Terraform Enterprise to host and manage private modules.

**NEW QUESTION 16**
Which command should you run to check if all code in a Terraform configuration that references multiple modules is properly formatted without making changes?

A. terraform fmt -write-false
B. terraform fmt -list -recursive
C. terraform fmt -check -recursive
D. terraform fmt -check

**Answer:** C

**Explanation:**
This command will check if all code in a Terraform configuration that references multiple modules is properly formatted without making changes, and will return a non-zero exit code if any files need formatting. The other commands will either make changes, list the files that need formatting, or not check the modules.

**NEW QUESTION 18**
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 21**
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 23**
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 26**
Only the user that generated a plan may apply it.

A. True
B. False

**Answer:** B

**Explanation:**
Any user with permission to apply a plan can apply it, not only the user that generated it. This allows for collaboration and delegation of tasks among team members.

**NEW QUESTION 28**
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 plans1. Some of the benefits of using Sentinel with Terraform Cloud/Terraform Enterprise are:
•You can restrict specific resource configurations, such as disallowing the use of CIDR=0.0.0.0/0, which would open up your network to the entire internet. This can help you prevent misconfigurations or security vulnerabilities in your infrastructure2.
•Policy-as-code can enforce security best practices, such as requiring encryption, authentication, or compliance standards. This can help you protect your data and meet regulatory requirements3.
•You can enforce a list of approved AWS AMIs, which are pre-configured images that contain the operating system and software you need to run your applications. This can help you ensure consistency, reliability, and performance across your infrastructure4. 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 32**
You add a new resource to an existing Terraform configuration, but do not update the version constraint in the configuration. The existing and new resources use the same provider. The working contains a .terraform.lock, hc1 file.
How will Terraform choose which version of the provider to use?

A. Terraform will use the version recorded in your lock file
B. Terraform will use the latest version of the provider for the new resource and the version recorded in the lock file to manage existing resources
C. Terraform will check your state file to determine the provider version to use
D. Terraform will use the latest version of the provider available at the time you provision your new resource

**Answer:** A

**Explanation:**
This is how Terraform chooses which version of the provider to use, when you add a new resource to an existing Terraform configuration, but do not update the version constraint in the configuration. The lock file records the exact version of each provider that was installed in your working directory, and ensures that

Terraform will always use the same provider versions until you run terraform init -upgrade to update them.

**NEW QUESTION 33**
A module can always refer to all variables declared in its parent module.

A. True
B. False

**Answer:** B

**Explanation:**
A module cannot always refer to all variables declared in its parent module, as it needs to explicitly declare input variables and assign values to them from the parent module??s arguments. A module cannot access the parent module??s variables directly, unless they are passed as input arguments.

**NEW QUESTION 38**
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 40**
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 43**
Setting the TF_LOG environment variable to DEBUG causes debug messages to be logged into stdout.

A. True
B. False

**Answer:** A

**Explanation:**
Setting the TF_LOG environment variable to DEBUG causes debug messages to be logged into stdout, along with other log levels such as TRACE, INFO, WARN, and ERROR. This can be useful for troubleshooting or debugging purposes.

**NEW QUESTION 47**
You can develop a custom provider to manage its resources using Terraform.

A. True
B. False

**Answer:** A

**Explanation:**
You can develop a custom provider to manage its resources using Terraform, as Terraform is an extensible tool that allows you to write your own plugins in Go language. You can also publish your custom provider to the Terraform Registry or use it privately.

**NEW QUESTION 50**
You want to define a single input variable to capture configuration values for a server. The values must represent memory as a number, and the server name as a string.
Which variable type could you use for this input?

A. List
B. Object
C. Map
D. Terraform does not support complex input variables of different types

**Answer:** B

**Explanation:**
This is the variable type that you could use for this input, as it can store multiple attributes of different types within a single value. The other options are either invalid or incorrect for this use case.

**NEW QUESTION 55**
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 56**
What type of block is used to construct a collection of nested configuration blocks?

A. Dynamic
B. For_each
C. Nesting
D. repeated.

**Answer:** A

**Explanation:**
This is the type of block that is used to construct a collection of nested configuration blocks, by using a for_each argument to iterate over a collection value and generate a nested block for each element. For example, you can use a dynamic block to create multiple ingress rules for a security group resource.

**NEW QUESTION 59**
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 64**
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 65**
You are writing a child Terraform module that provisions an AWS instance. You want to reference the IP address returned by the child module in the root configuration. You name the instance resource "main'.
Which of these is the correct way to define the output value?
A)

```
output "instance_ip_addr" {
  return aws_instance.main.private_ip
}
```

B)

```
output "aws_instance.instance_ip_addr" {
  return aws_instance.main.private_ip
}
```

C)

```
output "aws_instance.instance_ip_addr" {
  value = ${main.private_ip}
}
```

D)

```
output "instance_ip_addr" {
  value = aws_instance.main.private_ip
}
```

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

**Answer:** D

**NEW QUESTION 69**
Which of these ate 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 73**
Which of the following are advantages of using infrastructure as code (IaC) instead of provisioning with a graphical user interface (GUI)? Choose two correct answers.

A. Prevents manual modifications to your resources
B. Lets you version, reuse, and share infrastructure configuration
C. Secures your credentials
D. Provisions the same resources at a lower cost
E. Reduces risk of operator error

**Answer:** BE

**Explanation:**
Infrastructure as code (IaC) is a way of managing and provisioning cloud infrastructure using programming techniques instead of manual processes1. IaC has many advantages over using a graphical user interface (GUI) for provisioning infrastructure, such as:
•Versioning: IaC allows you to store your infrastructure configuration in a version control system, such as Git, and track changes over time. This enables you to roll back to previous versions, compare differences, and collaborate with other developers2.
•Reusability: IaC allows you to create reusable modules and templates that can be applied to different environments, such as development, testing, and production. This reduces duplication, improves consistency, and speeds up deployment3.
•Sharing: IaC allows you to share your infrastructure configuration with other developers, teams, or organizations, and leverage existing code from open source repositories or registries. This fosters best practices, innovation, and standardization4.
•Risk reduction: IaC reduces the risk of human error, configuration drift, and security breaches that can occur when provisioning infrastructure manually or using a GUI. IaC also enables you to perform automated testing, validation, and compliance checks on your
infrastructure before deploying it5. References =
•1: What is Infrastructure as Code? Explained for Beginners - freeCodeCamp.org
•2: The benefits of Infrastructure as Code - Microsoft Community Hub
•3: Infrastructure as Code : Best Practices, Benefits & Examples - Spacelift
•4: 5 Benefits of Infrastructure as Code (IaC) for Modern Businesses in the Cloud
•5: The 7 Biggest Benefits of Infrastructure as Code - DuploCloud

**NEW QUESTION 75**
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 78**
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

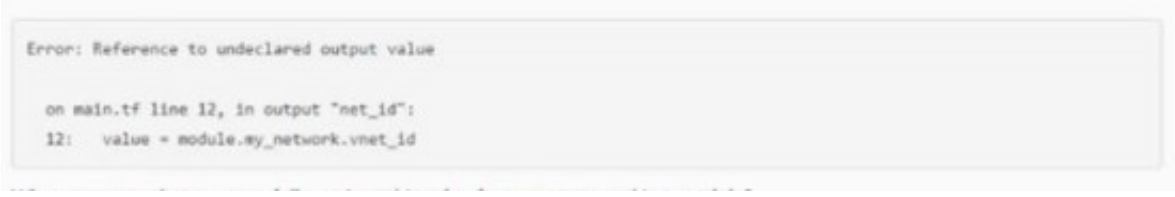
**Answer:** A

**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 82**
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:




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 86**
What does state looking 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 91**
What kind of configuration block will create an infrastructure object with settings specified
within the block?

A. provider
B. state
C. data
D. resource

**Answer:** D

**Explanation:**
This is the kind of configuration block that will create an infrastructure object with settings specified within the block. The other options are not used for creating infrastructure objects, but for configuring providers, accessing state data, or querying data sources.

**NEW QUESTION 95**
Which Terraform command checks that your configuration syntax is correct?

A. terraform validate
B. terraform init
C. terraform show
D. terraform fmt

**Answer:** A

**Explanation:**
The terraform validate command is used to check that your Terraform configuration files are syntactically valid and internally consistent. It is a useful command for ensuring your Terraform code is error-free before applying any changes to your infrastructure.

**NEW QUESTION 96**
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 101**
Which of these actions will prevent two Terraform runs from changing the same state file at the same time?

A. Refresh the state after running Terraform
B. Delete the state before running Terraform
C. Configure state locking for your state backend
D. Run Terraform with parallelism set to 1

**Answer:** B

**Explanation:**
To prevent two Terraform runs from changing the same state file simultaneously, state locking is used. State locking ensures that when one Terraform operation is running, others will be blocked from making changes to the same state, thus preventing conflicts and data corruption. This is achieved by configuring the state backend to support locking, which will lock the state for all operations that could write to the state.References = This information is supported by Terraform's official documentation,
which explains the importance of state locking and how it can be configured for different backends to prevent concurrent state modifications .

**NEW QUESTION 105**
It is best practice to store secret data in the same version control repository as your Terraform configuration.

A. True
B. False

**Answer:** B

**Explanation:**
It is not a best practice to store secret data in the same version control repository as your Terraform configuration, as it could expose your sensitive information to unauthorized parties or compromise your security. You should use environment variables, vaults, or other mechanisms to store and provide secret data to Terraform.

**NEW QUESTION 108**
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 109**
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 111**
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 115**
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 values1.

**NEW QUESTION 117**
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 118**
Where can Terraform not load a provider from?

A. Plugins directory
B. Provider plugin chance
C. Official HashCrop Distribution on releases.hashcrop.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 119**
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 123**
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 executed3. This allows you to enforce rules on your infrastructure before it is created or modified.

**NEW QUESTION 127**
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 apply1.
References = This information can be verified from the official Terraform documentation on the terraform init command provided by HashiCorp Developer1.

**NEW QUESTION 132**
terraform validate reports syntax check errors for which of the following?

A. Code contains tabs for indentation instead of spaces
B. There is a missing value for a variable
C. The state file does not match the current infrastructure
D. None of the above

**Answer:** D

**Explanation:**
The terraform validate command is used to check for syntax errors and internal consistency within Terraform configurations, such as whether all required arguments are specified. It does not check for indentation styles, missing variable values (as variables might not be defined at validation time), or state file consistency with the current infrastructure. Therefore, none of the provided options are correct in the context of what terraform validate reports.References = Terraform's official documentation details the purpose and function of the terraform validate command, specifying that it focuses on syntax and consistency checks within Terraform configurations themselves, not on external factors like the state file or infrastructure state. Direct references from the HashiCorp Terraform Associate (003) study materials to this specific detail were not found in the provided files.

**NEW QUESTION 133**
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 134**
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 139**
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 144**
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 149**
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 154**
How would you reference the "name???? value of the second instance of this resource?

```
resource "aws_instance" "web" {
  count = 2
  name = "terraform-${count.index}"
}
```

A. aws_instance.web(2),name
B. element(aws_instance.web, 2)
C. aws_instance-web(1)

D. aws_instance_web(1),name
E. Aws_instance,web,* , name

**Answer:** D

**Explanation:**
In Terraform, when you use the count meta-argument, you can reference individual instances using an index. The indexing starts at 0, so to reference the "name" value of the second instance, you would use aws_instance.web[1].name. This syntax allows you to access the properties of specific instances in a list generated by the count argument.
References:
? Terraform documentation on count and accessing resource instances: Terraform Count

**NEW QUESTION 157**
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 161**
Which of the following ate advantages of using infrastructure as code (laC) instead of provisioning with a graphical user interface (GUI)? Choose two correct answers.

A. Lets you version, reuse, and share infrastructure configuration
B. Provisions the same resources at a lower cost
C. Secures your credentials
D. Reduces risk of operator error
E. Prevents manual modifications to your resources

**Answer:** AD

**Explanation:**
? It lets you version, reuse, and share infrastructure configuration as code files, which can be stored in a source control system and integrated with your CI/CD pipeline.
? It reduces risk of operator error by automating repetitive tasks and ensuring consistency across environments. IaC does not necessarily provision resources at a lower cost, secure your credentials, or prevent manual modifications to your resources - these depend on other factors such as your cloud provider, your security practices, and your access policies.

**NEW QUESTION 164**
What does Terraform use the .terraform.lock.hc1 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 167**
What does the default "local" Terraform backend store?

A. tfplan files
B. State file
C. Provider plugins
D. Terraform binary

**Answer:** B

**Explanation:**
The default ??local?? Terraform backend stores the state file in a local file named terraform.tfstate, which can be used to track and manage the state of your infrastructure3.

**NEW QUESTION 168**
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 171**
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 177**
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 workspaces3. 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 error3. References = [Backend Type: remote]3

**NEW QUESTION 182**
You decide to move a Terraform state file to Amazon S3 from another location. You write the code below into a file called backend.tf.

```
terraform {
  backend "s3" {
    bucket = "my-tf-bucket"
    region = "us-east-1"
  }
}
```

Which command will migrate your current state file to the new S3 remote backend?

A. terraform state
B. terraform init
C. terraform push
D. terraform refresh

**Answer:** B

**Explanation:**
This command will initialize the new backend and prompt you to migrate the existing state file to the new location3. The other commands are not relevant for this task.

**NEW QUESTION 187**
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 192**
Which of the following commands would you use to access all of the attributes and details of a resource managed by Terraform?

A. terraform state list ??provider_type.name??
B. terraform state show ??provider_type.name??
C. terraform get ??provider_type.name??
D. terraform state list

**Answer:** B

**Explanation:**
The terraform state show command allows you to access all of the attributes and details of a resource managed by Terraform. You can use the resource address (e.g. provider_type.name) as an argument to show the information about a specific
resource. The terraform state list command only shows the list of resources in the state, not their attributes. The terraform get command downloads and installs modules needed for the configuration. It does not show any information about resources. References = [Command: state show] and [Command: state list]

**NEW QUESTION 193**
Select the command that doesn??t cause Terraform to refresh its state.

A. Terraform destroy
B. Terraform apply
C. Terraform plan
D. Terraform state list

**Answer:** D

**Explanation:**
This is the command that does not cause Terraform to refresh its state, as it only lists the resources that are currently managed by Terraform in the state file. The other commands will refresh the state file before performing their operations, unless you use the -refresh=false flag.

**NEW QUESTION 196**
What is the Terraform style convention for indenting a nesting level compared to the one above it?

A. With a tab
B. With two spaces
C. With four spaces
D. With three spaces

**Answer:** B

**Explanation:**
This is the Terraform style convention for indenting a nesting level compared to the one above it. The other options are not consistent with the Terraform style guide.

**NEW QUESTION 197**
Which provider authentication method prevents credentials from being stored in the state file?

A. Using environment variables
B. Specifying the login credentials in the provider block
C. Setting credentials as Terraform variables
D. None of the above

**Answer:** D

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

file, you can use one of the following methods:
? Use environment variables to pass credentials to the provider. This way, the credentials are not part of the provider configuration and are not stored in the state file. However, this method may not work for some providers that require credentials to be set in the provider block.
? Use dynamic credentials to authenticate with your cloud provider. This way,
Terraform Cloud or Enterprise will request temporary credentials from your cloud provider for each run and use them to provision your resources. The credentials are not stored in the state file and are revoked after the run is completed. This method is supported for AWS, Google Cloud Platform, Azure, and
Vault. References = : [Sensitive Values in State] : Authenticate providers with dynamic credentials

**NEW QUESTION 201**
Which command must you first run before performing further Terraform operations in a working directory?

A. terraform import
B. terraform workspace
C. terraform plan
D. terraform init

**Answer:** D

**Explanation:**
terraform init is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control. It initializes a working directory containing Terraform configuration files and downloads any required providers and modules. The other commands are used for different purposes, such as importing existing resources, switching between workspaces, generating execution plans, etc.

**NEW QUESTION 206**
A Terraform provider is NOT responsible for:

A. Exposing resources and data sources based on an APUI
B. Managing actions to take based on resources differences
C. Understanding API interactions with some service
D. Provisioning infrastructure in multiple

**Answer:** D

**Explanation:**
This is not a responsibility of a Terraform provider, as it does not make sense grammatically or logically. A Terraform provider is responsible for exposing resources and data sources based on an API, managing actions to take based on resource differences, and understanding API interactions with some service.

**NEW QUESTION 210**
.....

# Thank You for Trying Our Product

* 100% Pass or Money Back

All our products come with a 90-day Money Back Guarantee.

* One year free update

You can enjoy free update one year. 24x7 online support.

* Trusted by Millions

We currently serve more than 30,000,000 customers.

* Shop Securely

All transactions are protected by VeriSign!

## 100% Pass Your Terraform-Associate-003 Exam with Our Prep Materials Via below:

https://www.certleader.com/Terraform-Associate-003-dumps.html