HashiCorp

Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)

NEW QUESTION 1
When does Terraform create the .terraform.lock.hc1 file?

A. After your first terraform plan
B. After your first terraform apply
C. After your first terraform init
D. When you enable state locking

Answer: C

Explanation:
Terraform creates the .terraform.lock.hcl file after the first terraform init command. This lock file ensures that the dependencies for your project are consistent across different runs by locking the versions of the providers and modules used.

NEW QUESTION 2
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 3
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 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
How does Terraform determine dependencies between resources?

A. Terraform requires resource dependencies to be defined as modules and sourced in order
B. Terraform automatically builds a resource graph based on resources provisioners, special meta-parameters, and the stale file (if present}
C. Terraform requires resources in a configuration to be listed m the order they will be created to determine dependencies
D. Terraform requires all dependencies between resources to be specified using the depends_on parameter

Answer: B

Explanation:
This is how Terraform determines dependencies between resources, by using the references between them in the configuration files and other factors that affect the order of operations.

NEW QUESTION 6
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 7
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 8
What does this code do?

```
terraform {
  required_providers {
    aws = "~> 3.0"
  }
}
```

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 11
A developer on your team is going lo leaf down an existing deployment managed by Terraform and deploy a new one. However, there is a server resource named aws instant.ubuntu[l] they would like to keep. What command should they use to tell Terraform to stop managing that specific resource?

A. Terraform plan rm:aws_instance.ubuntu[1]
B. Terraform state rm:aws_instance.ubuntu[1]
C. Terraform apply rm:aws_instance.ubuntu[1]
D. Terraform destory rm:aws_instance.ubuntu[1]

Answer: B

Explanation:
To tell Terraform to stop managing a specific resource without destroying it, you can use the terraform state rm command. This command will remove the resource from the Terraform state, which means that Terraform will no longer track or update the corresponding remote object. However, the object will still exist in the remote system and you can later use terraform import to start managing it again in a different configuration or workspace. The syntax for this command is terraform state rm <address>,
where <address> is the resource address that identifies the resource instance to remove.
For example, terraform state rm aws_instance.ubuntu[1] will remove the second instance of the aws_instance resource named ubuntu from the state. References = : Command: state rm : Moving Resources

NEW QUESTION 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 17
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 files4. The other options are valid backends that can store state files in a central location.

NEW QUESTION 18
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 22
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 23
When you use a remote backend that needs authentication, HashiCorp recommends that you:

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

Answer: D

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

NEW QUESTION 27
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 31
The Terraform binary version and provider versions must match each other in a single configuration.

A. True
B. False

Answer: B

Explanation:
The Terraform binary version and provider versions do not have to match each other in a single configuration. Terraform allows you to specify provider version constraints in the configuration??s terraform block, which can be different from the Terraform binary version1. Terraform will use the newest version of the provider that meets the configuration??s version constraints2. You can also use the dependency lock file to ensure Terraform is using the correct provider version3.
References =
•1: Providers - Configuration Language | Terraform | HashiCorp Developer
•2: Multiple provider versions with Terraform - Stack Overflow
•3: Lock and upgrade provider versions | Terraform - HashiCorp Developer

NEW QUESTION 36
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 41
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 loss1.

NEW QUESTION 42
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 46
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 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 49
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 53
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 60
Which of the following is not a valid siring function in Terraform?

A. choaf
B. join
C. Split
D. slice

Answer: A

Explanation:
This is not a valid string function in Terraform. The other options are valid string functions that can manipulate strings in various ways2.

NEW QUESTION 62
Which configuration consistency errors does terraform validate report?

A. Terraform module isn't the latest version
B. Differences between local and remote state
C. Declaring a resource identifier more than once
D. A mix of spaces and tabs in configuration files

Answer: C

Explanation:
Terraform validate reports configuration consistency errors, such as declaring a resource identifier more than once. This means that the same resource type and name combination is used for multiple resource blocks, which is not allowed in Terraform. For example, resource "aws_instance" "example" {...} cannot be used more than once in the same configuration. Terraform validate does not report errors related to module versions, state differences, or formatting issues, as these are not relevant for checking the configuration syntax and structure. References = [Validate Configuration], [Resource Syntax]

NEW QUESTION 64
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 67

Module variable assignments are inherited from the parent module and you do not need to explicitly set them.

A. True
B. False

Answer: B

Explanation:
Module variable assignments are not inherited from the parent module and you need to explicitly set them using the source argument. This allows you to customize the behavior of each module instance.

NEW QUESTION 71
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 75
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 78
You're building a CI/CD (continuous integration/continuous delivery) pipeline and need to inject sensitive variables into your Terraform run. How can you do this safely?

A. Copy the sensitive variables into your Terraform code
B. Store the sensitive variables in a secure_varS.tf file
C. Store the sensitive variables as plain text in a source code repository
D. Pass variables to Terraform with a -var flag

Answer: D

Explanation:
This is a secure way to inject sensitive variables into your Terraform run, as they will not be stored in any file or source code repository. You can also use environment variables or variable files with encryption to pass sensitive variables to Terraform.

NEW QUESTION 83
You are using a networking module in your Terraform configuration with the name label my-network. In your main configuration you have the following code:

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

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

```
Error: Reference to undeclared output value

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

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

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

Answer: C

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

NEW QUESTION 84
When do changes invoked by terraform apply take effect?

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

Answer: B

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

NEW QUESTION 87
What is the workflow for deploying new infrastructure with Terraform?

A. Write Terraform configuration, run terraform init to initialize the working directory or workspace, and run terraform apply
B. Write Terraform configuration, run terraform show to view proposed changes, and terraform apply to create new infrastructure
C. Write Terraform configuration, run terraform apply to create infrastructure, use terraform validate to confirm Terraform deployed resources correctly
D. Write Terraform configuration, run terraform plan to initialize the working directory or workspace, and terraform apply to create the infrastructure

Answer: A

Explanation:
This is the workflow for deploying new infrastructure with Terraform, as it will create a plan and apply it to the target environment. The other options are either incorrect or incomplete.

NEW QUESTION 89
A provider configuration block is required in every Terraform configuration.
Example:

```
provider "provider_name" {
   ...
}
```

A. True
B. False

Answer: B

Explanation:
A provider configuration block is not required in every Terraform configuration. A provider configuration block can be omitted if its contents would otherwise be empty. Terraform assumes an empty default configuration for any provider that is not explicitly configured. However, some providers may require some configuration arguments (such as endpoint URLs or cloud regions) before they can be used. A provider??s documentation should list which configuration arguments it expects. For providers distributed on the Terraform Registry, versioned documentation is available on each provider??s page, via the ??Documentation?? link in the provider??s header1. References = [Provider Configuration]1

NEW QUESTION 92
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 95
Terraform providers are part of the Terraform core binary.

A. True
B. False

Answer: B

Explanation:
Terraform providers are not part of the Terraform core binary. Providers are distributed separately from Terraform itself and have their own release cadence and version numbers. Providers are plugins that Terraform uses to interact with various APIs, such as cloud providers, SaaS providers, and other services. You can find and install providers from the Terraform Registry, which hosts providers for most major infrastructure platforms. You can also load providers from a local mirror or cache, or develop your own custom providers. To use a provider in your Terraform configuration, you need to declare it in the provider requirements block and optionally configure its settings in the provider
block. References = : Providers - Configuration Language | Terraform : Terraform Registry
- Providers Overview | Terraform

NEW QUESTION 98
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 103
You have used Terraform lo create an ephemeral development environment in the (loud 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 mil 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 106
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 111
You can configure Terraform to log to a file using the TF_LOG environment variable.

A. True
B. False

Answer: A

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

NEW QUESTION 114
When using Terraform to deploy resources into Azure, which scenarios are true regarding state files? (Choose two.)

A. When you change a Terraform-managed resource via the Azure Cloud Console, Terraform updates the state file to reflect the change during the next plan or apply
B. Changing resources via the Azure Cloud Console records the change in the current state file

C. When you change a resource via the Azure Cloud Console, Terraform records the changes in a new state file
D. Changing resources via the Azure Cloud Console does not update current state file

Answer: AD

Explanation:
Terraform state is a representation of the infrastructure that Terraform manages. Terraform uses state to track the current status of the resources it creates and to plan future changes. However, Terraform state is not aware of any changes made to the resources outside of Terraform, such as through the Azure Cloud Console, the Azure CLI, or the Azure API. Therefore, changing resources via the Azure Cloud Console does not update the current state file, and it may cause inconsistencies or conflicts with Terraform??s desired configuration. To avoid this, it is recommended to manage resources exclusively through Terraform or to use the terraform import command to bring existing resources under Terraform??s control.
When you change a Terraform-managed resource via the Azure Cloud Console, Terraform does not immediately update the state file to reflect the change. However, the next time you run terraform plan or terraform apply, Terraform will compare the state file with the actual state of the resources in Azure and detect any drifts or differences. Terraform will
then update the state file to match the current state of the resources and show you the proposed changes in the execution plan. Depending on the configuration and the change, Terraform may try to undo the change, modify the resource further, or recreate the resource entirely. To avoid unexpected or destructive changes, it is recommended to review the execution plan carefully before applying it or to use the terraform
refresh command to update the state file without applying any changes.
References = Purpose of Terraform State, Terraform State, Managing State, Importing Infrastructure, [Command: plan], [Command: apply], [Command: refresh]

NEW QUESTION 115
Which of these is true about Terraform's plugin-based architecture?

A. Terraform can only source providers from the internet
B. Every provider in a configuration has its own state file for its resources
C. You can create a provider for your API if none exists
D. All providers are part of the Terraform core binary

Answer: C

Explanation:
Terraform is built on a plugin-based architecture, enabling developers to extend Terraform by writing new plugins or compiling modified versions of existing plugins1. Terraform plugins are executable binaries written in Go that expose an implementation for a specific service, such as a cloud resource, SaaS platform, or API2. If there is no existing provider for your API, you can create one using the Terraform Plugin SDK3 or the Terraform Plugin Framework4. References =
•1: Plugin Development - How Terraform Works With Plugins | Terraform | HashiCorp Developer
•2: Lab: Terraform Plug-in Based Architecture - GitHub
•3: Terraform Plugin SDK - Terraform by HashiCorp
•4: HashiCorp Terraform Plugin Framework Now Generally Available

NEW QUESTION 116
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 120
You have provisioned some virtual machines (VMs) on Google Cloud Platform (GCP) using the gcloud command line tool. However, you are standardizing with Terraform and want to manage these VMs using Terraform instead. What are the two things you must do to achieve this? Choose two correct answers.

A. Run the terraform Import-gcp command
B. Write Terraform configuration for the existing VMs
C. Use the terraform import command for the existing VMs
D. Provision new VMs using Terraform with the same VM names

Answer: BC

Explanation:
To import existing resources into Terraform, you need to do two things1:
? Write a resource configuration block for each resource, matching the type and name used in your state file.
? Run terraform import for each resource, specifying its address and ID. There is no such command as terraform Import-gcp, and provisioning new VMs with the same names will not import them into Terraform.

NEW QUESTION 121
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 123
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 124
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 128
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 132
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 136
When should you run terraform init?

A. Every time you run terraform apply
B. Before you start coding a new Terraform project

C. After you run terraform plan for the time in a new terraform project and before you run terraform apply
D. After you start coding a new terraform project and before you run terraform plan for thefirst time.

Answer: D

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

NEW QUESTION 140
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 143
Infrastructure as Code (laC) 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 146
You're writing a Terraform configuration that needs to read input from a local file called id_rsa.pub . Which built-in Terraform function can you use to import the file's contents as a string?

A. file("id_rsa.pub")
B. templaTefil("id_rsa.pub")
C. filebase64("id_rsa.pub")
D. fileset<"id_rsa.pub")

Answer: A

Explanation:
To import the contents of a local file as a string in Terraform, you can use the built-in file function. By specifying file("id_rsa.pub"), Terraform reads the contents of the id_rsa.pub file and uses it as a string within your Terraform configuration. This function is particularly useful for scenarios where you need to include file data directly into your configuration, such as including an SSH public key for provisioning cloud
instances.References = This information is a standard part of Terraform's functionality with built-in functions, as outlined in Terraform's official documentation and commonly used in various Terraform configurations.

NEW QUESTION 150
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 155
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 156
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
Which of these commands makes your code more human readable?

A. Terraform validate
B. Terraform output
C. Terraform show
D. Terraform fmt

Answer: D

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

NEW QUESTION 162
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 166
As a member of an operations team that uses infrastructure as code (lac) practices, you are tasked with making a change to an infrastructure stack running in a public cloud. Which pattern would follow laC best practices for making a change?

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 168
You have never used Terraform before and would like to test it out using a shared team account for a cloud provider. The shared team account already contains 15 virtual machines (VM). You develop a Terraform configuration containing one VM. perform terraform apply, and see that your VM was created successfully. What should you do to delete the newly-created VM with Terraform?

A. The Terraform state file contains all 16 VMs in the team accoun
B. Execute terraform destroy and select the newly-created VM.
C. Delete the Terraform state file and execute terraform apply.
D. The Terraform state file only contains the one new V
E. Execute terraform destroy.
F. Delete the VM using the cloud provider console and terraform apply to apply the changes to the Terraform state file.

Answer: C

Explanation:
This is the best way to delete the newly-created VM with Terraform, as it will only affect the resource that was created by your configuration and state file. The other options are either incorrect or inefficient.

NEW QUESTION 169
Which of these are features of Terraform Cloud? Choose two correct answers.

A. A web-based user interface (Ul)
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 170
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 172
Which of the following is not a benefit of adopting infrastructure as code?

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

Answer: B

Explanation:
Infrastructure as Code (IaC) provides several benefits, including the ability to version control infrastructure, reuse code, and automate infrastructure management. However, IaC is typically associated with declarative configuration files and does not inherently provide a graphical user interface (GUI). A GUI is a feature that may be provided by specific tools or platforms built on top of IaC principles but is not a direct benefit of IaC itself1.
References = The benefits of IaC can be verified from the official HashiCorp documentation on ??What is Infrastructure as Code with Terraform??? provided by HashiCorp Developer1.

NEW QUESTION 175
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 them1.

NEW QUESTION 176
When using a remote backend or terraform Cloud integration, where does Terraform save resource sate?

A. In an environment variable
B. On the disk
C. In the remote backend or Terraform Cloud
D. In memory

Answer: C

Explanation:
This is where Terraform saves resource state when using a remote backend or Terraform Cloud integration, as it allows you to store and manage your state file in a remote location, such as a cloud storage service or Terraform Cloud??s servers. This enables collaboration, security, and scalability for your Terraform infrastructure.

NEW QUESTION 180
Which of the following is not true of Terraform providers?

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

Answer: F

Explanation:
All of the statements are true of Terraform providers. Terraform providers are plugins that enable Terraform to interact with various APIs and services1. Anyone can write a Terraform provider, either as an individual or as part of a community2. HashiCorp maintains some providers, such as the AWS, Azure, and Google Cloud providers3. Cloud providers and infrastructure vendors can also write, maintain, or collaborate on Terraform providers, such as the VMware, Oracle, and Alibaba Cloud providers. References =
•1: Providers - Configuration Language | Terraform | HashiCorp Developer
•2: Plugin Development - How Terraform Works With Plugins | Terraform | HashiCorp Developer
•3: Terraform Registry
•: Terraform Registry

NEW QUESTION 185
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 188
What is terraform refresh-only intended to detect?

A. Terraform configuration code changes
B. Corrupt state files

C. State file drift
D. Empty state files

Answer: C

Explanation:
The terraform refresh-only command is intended to detect state file drift. This command synchronizes the state file with the actual infrastructure, updating the state to reflect any changes that have occurred outside of Terraform.

NEW QUESTION 193
Any user can publish modules to the public Terraform Module Registry.

A. True
B. False

Answer: A

Explanation:
The Terraform Registry allows any user to publish and share modules. Published modules support versioning, automatically generate documentation, allow browsing version histories, show examples and READMEs, and more. Public modules are managed via Git and GitHub, and publishing a module takes only a few minutes. Once a module is published, releasing a new version of a module is as simple as pushing a properly formed Git tag1.
References = The information can be verified from the Terraform Registry documentation on Publishing Modules provided by HashiCorp Developer1.

NEW QUESTION 197
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 201
A Terraform output that sets the "sensitive" argument to true will not store that value in the state file.

A. True
B. False

Answer: A

Explanation:
A Terraform output that sets the "sensitive" argument to true will store that value in the state file. The purpose of setting sensitive = true is to prevent the value from being displayed in the CLI output during terraform plan and terraform apply, and to mask it in the Terraform UI. However, it does not affect the storage of the value in the state file. Sensitive outputs are still written to the state file to ensure that Terraform can manage resources correctly during subsequent operations.
References:
? Terraform documentation on sensitive outputs: Terraform Output Values

NEW QUESTION 203
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 205
You can reference a resource created with for_each using a Splat (*) expression.

A. True
B. False

Answer: B

Explanation:
You cannot reference a resource created with for_each using a splat (*) expression, as it will not work with resources that have non-numeric keys. You need to

use a for expression instead to iterate over the resource instances.

NEW QUESTION 206
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 211
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 212
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 214
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 218
How do you specify a module??s version when publishing it to the public terraform Module Registry?

A. Configuration it in the module's Terraform code
B. Mention it on the module's configuration page on the Terraform Module Registry
C. The Terraform Module Registry does not support versioning modules
D. Tag a release in the associated repo

Answer:

D

Explanation:
This is how you specify a module??s version when publishing it to the public Terraform Module Registry, as it uses the tags from your version control system (such as GitHub or GitLab) to identify module versions. You need to use semantic versioning for your tags, such as v1.0.0.

NEW QUESTION 221
Which of these statements about Terraform Cloud workspaces is false?

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

Answer: B

Explanation:
The statement that you must use the CLI to switch between workspaces is false. Terraform Cloud workspaces are different from Terraform CLI workspaces. Terraform Cloud workspaces are required and represent all of the collections of infrastructure in an organization. They are also a major component of role-based access in Terraform Cloud. You can grant individual users and user groups permissions for one or more workspaces that dictate whether they can manage variables, perform runs, etc. You can create, view, and switch between Terraform Cloud workspaces using the Terraform Cloud UI, the Workspaces API, or the Terraform Enterprise Provider5. Terraform CLI workspaces are optional and allow you to create multiple distinct instances of a single configuration within one working directory. They are useful for creating disposable environments for testing or experimenting without affecting your main or production environment. You can create, view, and switch between Terraform CLI workspaces using the terraform workspace command6. The other statements about Terraform Cloud workspaces are true. They have role-based access controls that allow you to assign permissions to users and teams based on their roles and responsibilities. You can create and manage roles using the Teams API or the Terraform Enterprise Provider7. Plans and applies can be triggered via version control system integrations that allow you to link your Terraform Cloud workspaces to your VCS repositories. You can configure VCS settings, webhooks, and branch tracking to automate your Terraform Cloud workflow8. They can securely store cloud credentials as sensitive variables that are encrypted at rest and only decrypted when needed. You can manage variables using the Terraform Cloud UI, the Variables API, or the Terraform Enterprise Provider9. References = [Workspaces]5, [Terraform CLI Workspaces]6, [Teams and Organizations]7, [VCS Integration]8, [Variables]9

NEW QUESTION 222
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 226
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 231
.....

Thank You for Trying Our Product

We offer two products:

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

2nd - Questons 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