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
How would you reference the volume IDs associated with the ebs_block_device blocks in this configuration?

```
resource "aws_instance" "example" {
  ami = "ami-abc123"
  instance_type = "t2.micro"

  ebs_block_device {
    device_name = "sda2"
    volume_size = 16
  }

  ebs_block_device {
    device_name = "sda3"
    volume_size = 20
  }
}
```

A. aws_instance.example.ebs_block_device[sda2,sda3).volume_id
B. aws_lnstance.example.ebs_block_device.[*].volume_id
C. aws_lnstance.example.ebs_block_device.volume_ids
D. aws_instance.example-ebs_block_device.*.volume_id

Answer: D

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

NEW QUESTION 3
Which of the following is not a key principle of infrastructure as code?

A. Self-describing infrastructure
B. Idempotence
C. Versioned infrastructure
D. Golden images

Answer: D

Explanation:
The key principle of infrastructure as code that is not listed among the options is golden images. Golden images are pre-configured, ready-to-use virtual machine images that contain a specific set of software and configuration. They are often used to create multiple identical instances of the same environment, such as for testing or production. However, golden images are not a principle of infrastructure as code, but rather a technique that can be used with or without infrastructure as code. The other options are all key principles of infrastructure as code, as explained below:
? Self-describing infrastructure: This means that the infrastructure is defined in code that describes its desired state, rather than in scripts that describe the steps to create it. This makes the infrastructure easier to understand, maintain, and reproduce.
? Idempotence: This means that applying the same infrastructure code multiple times will always result in the same state, regardless of the initial state. This makes the infrastructure consistent and predictable, and avoids errors or conflicts caused by repeated actions.
? Versioned infrastructure: This means that the infrastructure code is stored in a version control system, such as Git, that tracks the changes and history of the code. This makes the infrastructure code reusable, auditable, and collaborative, and enables practices such as branching, merging, and rollback. References = [Introduction to Infrastructure as Code with Terraform], [Infrastructure as Code in a Private or Public Cloud]

NEW QUESTION 4
You have to initialize a Terraform backend before it can be configured.

A. True
B. False

Answer: B

Explanation:
You can configure a backend in your Terraform code before initializing it. Initializing a backend will store the state file remotely and enable features like locking and workspaces. References = [Terraform Backends]

NEW QUESTION 5
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 6
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 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
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 9
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 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
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 14
You cannot install third party plugins using terraform init.

A. True
B. False

Answer: B

Explanation:
You can install third party plugins using terraform init, as long as you specify the plugin directory in your configuration or as a command-line argument. You can also use the terraform providers mirror command to create a local mirror of providers from any source.

NEW QUESTION 19
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 24
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 28
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 32
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 35
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 38
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 40
As a developer, you want to ensure your plugins are up to date with the latest versions. Which Terraform command should you use?

A. terraform refresh -upgrade
B. terraform apply -upgrade
C. terraform init -upgrade
D. terraform providers -upgrade

Answer: C

Explanation:
This command will upgrade the plugins to the latest acceptable version within the version constraints specified in the configuration. The other commands do not have an - upgrade option.

NEW QUESTION 45
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 50
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 54
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 58
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 61
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 66
Before you can use a remote backend, you must first execute terra-form init.

A. True
B. False

Answer: A

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

NEW QUESTION 68
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 72
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 75
What information does the public Terraform Module Registry automatically expose about published modules?

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

Answer: D

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

NEW QUESTION 76
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 79
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 81
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 83
You add a new provider to your configuration and immediately run terraform apply in the CD using the local backend. Why does the apply fail?

A. The Terraform CD needs you to log into Terraform Cloud first
B. Terraform requires you to manually run terraform plan first
C. Terraform needs to install the necessary plugins first
D. Terraform needs you to format your code according to best practices first

Answer: C

Explanation:
The reason why the apply fails after adding a new provider to the configuration and immediately running terraform apply in the CD using the local backend is because Terraform needs to install the necessary plugins first. Terraform providers are plugins that Terraform uses to interact with various cloud services and other APIs. Each provider has a source address that determines where to download it from. When Terraform encounters a new provider in the configuration, it needs to run terraform init first to install the provider plugins in a local directory. Without the plugins, Terraform cannot
communicate with the provider and perform the desired actions. References = [Provider Requirements], [Provider Installation]

NEW QUESTION 84
How is terraform import run?

A. As a part of terraform init
B. As a part of terraform plan
C. As a part of terraform refresh
D. By an explicit call
E. All of the above

Answer: D

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

NEW QUESTION 87
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 88
You have deployed a new webapp with a public IP address on a cloud provider. However, you did not create any outputs for your code. What is the best method to quickly find the IP address of the resource you deployed?

A. In a new folder, use the terraform_remote_state data source to load in the state file, then write an output for each resource that you find the state file
B. Run terraform state list to find the name of the resource, then terraform state show to find the attributes including public IP address
C. Run terraform output ip_address to view the result
D. Run terraform destroy then terraform apply and look for the IP address in stdout

Answer: B

Explanation:
This is a quick way to inspect the state file and find the information you need without modifying anything5. The other options are either incorrect or inefficient.

NEW QUESTION 89
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 94
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 95
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 99
Outside of the required_providers block, Terraform configurations always refer to providers by their local names.

A. True
B. False

Answer: B

Explanation:
Outside of the required_providers block, Terraform configurations can refer to providers by either their local names or their source addresses. The local name is a short name that can be used throughout the configuration, while the source address is a global identifier for the provider in the format registry.terraform.io/namespace/type. For example, you can use either aws or registry.terraform.io/hashicorp/aws to refer to the AWS provider.

NEW QUESTION 102
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 103
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 108
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 112
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 113
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 118
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 121
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 125
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 130
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 131
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 133
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 134
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 136
In a Terraform Cloud workspace linked to a version control repository, speculative plan runs start automatically when you merge or commit changes to version control.

A. True
B. False

Answer: B

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

NEW QUESTION 141
Terraform variable names are saved in the state file.

A. True
B. False

Answer: B

Explanation:
Terraform variable names are not saved in the state file, only their values are. The state file only stores the attributes of the resources and data sources that are managed by Terraform, not the variables that are used to configure them.

NEW QUESTION 144
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 148
Which type of block fetches or computes information for use elsewhere in a Terraform configuration?

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

Answer: A

Explanation:
In Terraform, a data block is used to fetch or compute information from external sources for use elsewhere in the Terraform configuration. Unlike resource blocks that manage infrastructure, data blocks gather information without directly managing any resources. This can include querying for data from cloud providers, external APIs, or other Terraform states.References = This definition and usage of data blocks are covered in Terraform's official documentation, highlighting their role in fetching external information to inform Terraform configurations.

NEW QUESTION 150
Terraform configuration (including any module references) can contain only one Terraform provider type.

A. True
B. False

Answer: B

Explanation:
Terraform configuration (including any module references) can contain more than one Terraform provider type. Terraform providers are plugins that Terraform uses to interact with various cloud services and other APIs. A Terraform configuration can use multiple providers to manage resources across different platforms and services. For example, a configuration can use the AWS provider to create a virtual machine, the Cloudflare provider to manage DNS records, and the GitHub provider to create a
repository. Terraform supports hundreds of providers for different use cases and scenarios. References = [Providers], [Provider Requirements], [Provider Configuration]

NEW QUESTION 153
Multiple team members are collaborating on infrastructure using Terraform and want to format the* Terraform code following standard Terraform-style convention. How should they ensure the code satisfies conventions?

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

Answer: C

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

NEW QUESTION 158
How can terraform plan aid in the development process?

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

Answer: B

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

NEW QUESTION 161
Why does this backend configuration not follow best practices?

```
terraform {
  backend "s3" {
    bucket     = "terraform-state-prod"
    key        = "network/terraform.tfstate"
    region     = "us-east-1"
    access_key = "AKIAIOSFODNN7EXAMPLE"
    secret_key = "wJalrXUtnFEMI/K7MDENG/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 162
Which of the following arguments are required when declaring a Terraform output?

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

Answer: A

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

NEW QUESTION 163
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 167
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 169
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 172
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 174
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 178
All modules published on the official Terraform Module Registry have been verified by HasihCorp.

A. True
B. False

Answer: B

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

NEW QUESTION 179
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 184
You have declared a variable called var.list which is a list of objects that all have an attribute id . Which options will produce a list of the IDs? Choose two correct answers.

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

Answer: BC

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

NEW QUESTION 187
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 192
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 193

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 195
You have created a main.tf Terraform configuration consisting of an application server, a database and a load balanced. You ran terraform apply and Terraform created all of the resources successfully.
Now you realize that you do not actually need the load balancer, so you run terraform destroy without any flags. What will happen?

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

Answer: C

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

NEW QUESTION 200
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 205
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 206
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 210
backends support state locking.

A. All
B. No
C. Some
D. Only local

Answer: C

Explanation:
Some backends support state locking, which prevents other users from modifying the state file while a Terraform operation is in progress. This prevents conflicts and data loss. Not all backends support this feature, and you can check the documentation for each backend type to see if it does.

NEW QUESTION 211
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 213
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 218
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 221
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 225
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 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 230
How would you output returned values from a child module in the Terraform CLI output?

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

Answer: C

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

NEW QUESTION 235
.....

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