HashiCorp

Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)

NEW QUESTION 1
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 2
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 3
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 4
HashiCorp Configuration Language (HCL) supports user-denned functions.

A. True
B. False

Answer: B

Explanation:
 HashiCorp Configuration Language (HCL) does not support user-defined functions. You can only use the built-in functions that are provided by the language. The built-in functions allow you to perform various operations and transformations on values within expressions. The general syntax for function calls is a function name followed by comma-separated arguments in parentheses, such as max(5, 12, 9). You can find the documentation for all of the available built-in functions in the Terraform Registry or the
Packer Documentation, depending on which tool you are using. References = : Functions - Configuration Language | Terraform : Functions - Configuration Language | Packer

NEW QUESTION 5
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 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
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 8
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 9
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
In Terraform HCL, an object type of object({name=string, age-number}) would match this value.
A)

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

B)

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

C)

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

D)

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

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

Answer: B

NEW QUESTION 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 15
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 17
All standard backend types support state locking, and remote operations like plan, apply, and destroy.

A. True
B. False

Answer: B

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

NEW QUESTION 22
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 24
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 27
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 28
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 32
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 35
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 39
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 40
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 44
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 45
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 50
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 55
What type of block is used to construct a collection of nested configuration blocks?

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

Answer: A

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

NEW QUESTION 59
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 60
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 61
In a Terraform Cloud workpace linked to a version control repository speculative plan rum start automatically commit changes to version control.

A. True
B. False

Answer: A

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

NEW QUESTION 65
How could you reference an attribute from the vsphere_datacenter data source for use with the datacenter_id argument within the vsphere_folder resource in the following configuration?

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

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

A. Data.vsphere_datacenter.DC.id
B. Vsphere_datacenter.dc.id
C. Data,dc,id
D. Data.vsphere_datacenter,dc

Answer: A

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

NEW QUESTION 67
Which of the following are advantages of using infrastructure as code (IaC) instead of provisioning with a graphical user interface (GUI)? Choose two correct answers.

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

Answer: BE

Explanation:
Infrastructure as code (IaC) is a way of managing and provisioning cloud infrastructure using programming techniques instead of manual processes1. IaC has

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

NEW QUESTION 68
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 71
The public Terraform Module Registry is free to use.

A. True
B. False

Answer: A

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

NEW QUESTION 72
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 74
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 78
You have a Terraform configuration that defines a single virtual machine with no references to it, You have run terraform apply to create the resource, and then removed the resource definition from your Terraform configuration file.
What will happen you run terraform apply in the working directory again?

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

Answer: D

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

NEW QUESTION 81
Which of the following is not a valid source path for specifying a module?

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

Answer: B

Explanation:
Terraform modules are referenced by specifying a source location. This location can be a URL or a file path. However, specifying query parameters such as ?version=vl.6.0 directly within the source path is not a valid or supported method for specifying a module version in Terraform. Instead, version constraints are specified using the version argument within the module block, not as part of the source string.
References
= This clarification is based on Terraform's official documentation regarding module usage, which outlines the correct methods for specifying module sources and versions.

NEW QUESTION 84
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 87
The _____ determines how Terraform creates, updates, or delete resources.

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

Answer: C

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

NEW QUESTION 91
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 95
Once you configure a new Terraform backend with a terraform code block, which command(s) should you use to migrate the state file?

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

Answer: A

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

NEW QUESTION 96
Your security team scanned some Terraform workspaces and found secrets stored in plaintext in state files. How can you protect that data?

A. Edit your state file to scrub out the sensitive data
B. Always store your secrets in a secrets.tfvars file
C. Delete the state file every time you run Terraform
D. Store the state in an encrypted backend

Answer: D

Explanation:
 This is a secure way to protect sensitive data in the state file, as it will be encrypted at rest and in transit2. The other options are not recommended, as they could lead to data loss, errors, or security breaches.

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 100
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 101
Which Terraform collection type should you use to store key/value pairs?

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

Answer: B

Explanation:
 The Terraform collection type that should be used to store key/value pairs is map. A map is a collection of values that are accessed by arbitrary labels, called keys.
The keys and values can be of any type, but the keys must be unique within a map. For example, var = { key1 = "value1", key2 = "value2" } is a map with two key/value pairs. Maps are useful for grouping related values together, such as configuration options or
metadata. References = [Collection Types], [Map Type Constraints]

NEW QUESTION 102
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 105
Which command lets you experiment with terraform expressions?

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

Answer: A

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

NEW QUESTION 106
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 111
You are working on some new application features and you want to spin up a copy of your production deployment to perform some quick tests. In order to avoid having to configure a new state backend, what open source Terraform feature would allow you create multiple states but still be associated with your current code?

A. Terraform data sources
B. Terraform local values
C. Terraform modules
D. Terraform workspaces
E. None of the above

Answer: D

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

NEW QUESTION 112
Which of the following command would be use to access all of the attributes and details of a resource managed by Terraform?

A. Terraform state show ?? provider_type_name
B. Terraform state list

C. Terraform get provider_type_name
D. Terraform state list provider_type_name

Answer: A

Explanation:
This is the command that you would use to access all of the attributes and details of a resource managed by Terraform, by providing the resource address as an argument. For example, terraform state show 'aws_instance.example' will show you all the information about the AWS instance named example.

NEW QUESTION 113
Variables declared within a module are accessible outside of the module.

A. True
B. False

Answer: B

Explanation:
Variables declared within a module are only accessible within that module, unless they are explicitly exposed as output values1.

NEW QUESTION 117
A developer accidentally launched a VM (virtual machine) outside of the Terraform workflow and ended up with two servers with the same name. They don't know which VM Terraform manages but do have a list of all active VM IDs.
Which of the following methods could you use to discover which instance Terraform manages?

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

Answer: A

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

NEW QUESTION 118
Where does the Terraform local backend store its state?

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

Answer: C

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

NEW QUESTION 123
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 126
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 131
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 136
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 139
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 142
Which of the following should you put into the required_providers block?

A. version >= 3.1
B. version = ??>= 3.1??
C. version ~> 3.1

Answer: B

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

NEW QUESTION 145
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 148
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 150
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 154
Which of these are features of Terraform Cloud? Choose two correct answers.

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

Answer: AD

Explanation:
Terraform Cloud includes several features designed to enhance collaboration and infrastructure management. Two of these features are:
? A web-based user interface (UI): This allows users to interact with Terraform Cloud
through a browser, providing a centralized interface for managing Terraform configurations, state files, and workspaces.
? Remote state storage: This feature enables users to store their Terraform state
files remotely in Terraform Cloud, ensuring that state is safely backed up and can be accessed by team members as needed.

NEW QUESTION 159
Which parameters does terraform import require? Choose two correct answers.

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

Answer: BC

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

NEW QUESTION 162
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 164
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 165
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 170
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 173
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 177
You can access state stored with the local backend by using terraform_remote_state data source.

A. True
B. False

Answer: B

Explanation:
You cannot access state stored with the local backend by using the terraform_remote_state data source. The terraform_remote_state data source is used to retrieve the root module output values from some other Terraform configuration using the latest state snapshot from the remote backend. It requires a backend that supports remote state storage, such as S3, Consul, AzureRM, or GCS. The local backend stores the state file locally on the filesystem, which terraform_remote_state cannot access. References:
? Terraform documentation on terraform_remote_state data source: Terraform
Remote State Data Source
? Example usage of remote state: Example Usage (remote Backend)

NEW QUESTION 178
What is one disadvantage of using dynamic blocks in Terraform?

A. Dynamic blocks can construct repeatable nested blocks
B. Terraform will run more slowly
C. They cannot be used to loop through a list of values
D. They make configuration harder to read and understand

Answer: D

Explanation:
This is one disadvantage of using dynamic blocks in Terraform, as they can introduce complexity and reduce readability of the configuration. The other options are either advantages or incorrect statements.

NEW QUESTION 180
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 182
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 183
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 184
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 185
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 190
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 193
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 195
......

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

100% Actual & Verified — Instant Download, Please Click
Order The Terraform-Associate-003 Practice Test Here