

Terraform-Associate-003 Dumps

HashiCorp Certified: Terraform Associate (003)

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



NEW QUESTION 1

When does Terraform create the .terraform.lock.hcl 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 does Terraform determine dependencies between resources?

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

Answer: B

Explanation:

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

NEW QUESTION 3

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 4

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 5

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 releases
- 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 6

Which is the best way to specify a tag of v1.0.0 when referencing a module stored in Git (for example. Git::https://example.com/vpc.git)?

- A. Append pref=v1.0.0 argument to the source path
- B. Add version = ??1.0.0?? parameter to module block
- C. Nothing modules stored on GitHub always default to version 1.0.0

Answer: A

Explanation:

The best way to specify a tag of v1.0.0 when referencing a module stored in Git is to append ?ref=v1.0.0 argument to the source path. This tells Terraform to use a specific Git reference, such as a branch, tag, or commit, when fetching the module source code. For example, source = "git::https://example.com/vpc.git?ref=v1.0.0". This ensures that the module version is consistent and reproducible across different environments. References = [Module Sources], [Module Versions]

NEW QUESTION 7

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

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

Answer: CD

Explanation:

These are features of Terraform Cloud, which is a hosted service that provides a web-based UI, remote state storage, remote operations, collaboration features, and more for managing your Terraform infrastructure.

NEW QUESTION 8

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 9

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

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

Answer: D

Explanation:

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

NEW QUESTION 10

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 10

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 12

You should run terraform fmt to rewrite all Terraform configurations within the current working directory to conform to Terraform-style conventions.

- A. True
- B. False

Answer: A

Explanation:

You should run terraform fmt to rewrite all Terraform configurations within the current working directory to conform to Terraform-style conventions. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability. It is recommended to use this command to ensure consistency of style across different Terraform codebases. The command is optional, opinionated, and has no customization options, but it can help you and your team understand the code more quickly and easily. References = : Command: fmt : Using Terraform fmt Command to Format Your Terraform Code

NEW QUESTION 15

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 20

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

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

Answer: A

Explanation:

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

NEW QUESTION 21

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 25

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 26

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 27

You modified your Terraform configuration and run Terraform plan to review the changes. Simultaneously, your teammate manually modified the infrastructure component you are working on. Since you already ran terraform plan locally, the execution plan for terraform apply will be the same.

- A. True
- B. False

Answer: B

Explanation:

The execution plan for terraform apply will not be the same as the one you ran locally with terraform plan, if your teammate manually modified the infrastructure component you are working on. This is because Terraform will refresh the state file before applying any changes, and will detect any differences between the state and the real resources.

NEW QUESTION 32

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 34

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 38

In a Terraform Cloud workspace linked to a version control repository speculative plan run 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 42

Which of these are 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 43

Which of the following does terraform apply change after you approve the execution plan? (Choose two.)

- A. Cloud infrastructure Most Voted
- B. The `.terraform` directory
- C. The execution plan
- D. State file
- E. Terraform code

Answer: AD

Explanation:

The terraform apply command changes both the cloud infrastructure and the state file after you approve the execution plan. The command creates, updates, or destroys the infrastructure resources to match the configuration. It also updates the state file to reflect the new state of the infrastructure. The `.terraform` directory, the execution plan, and the Terraform code are not changed by the terraform apply command. References = Command: apply and Purpose of Terraform State

NEW QUESTION 48

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 52

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 53

What does state locking 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 55

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=v1.6.0"
- 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=v1.6.0 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 58

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 63

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 66

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 header. References = [Provider Configuration]

NEW QUESTION 67

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. 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 68

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 plugins. Terraform plugins are executable binaries written in Go that expose an implementation for a specific service, such as a cloud resource, SaaS platform, or API. If there is no existing provider for your API, you can create one using the Terraform Plugin SDK or the Terraform Plugin Framework. 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 71

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

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

Answer: BC

Explanation:

To import existing resources into Terraform, you need to do two things:

• Write a resource configuration block for each resource, matching the type and name used in your state file.

• Run terraform import for each resource, specifying its address and ID. There is no such command as terraform Import-gcp, and provisioning new VMs with the same names will not import them into Terraform.

NEW QUESTION 74

You are making changes to existing Terraform code to add some new infrastructure. When is the best time to run terraform validate?

- A. After you run terraform apply so you can validate your infrastructure
- B. Before you run terraform apply so you can validate your provider credentials
- C. Before you run terraform plan so you can validate your code syntax
- D. After you run terraform plan so you can validate that your state file is consistent with your infrastructure

Answer: C

Explanation:

This is the best time to run terraform validate, as it will check your code for syntax errors, typos, and missing arguments before you attempt to create a plan. The other options are either incorrect or unnecessary.

NEW QUESTION 75

How can a ticket-based system slow down infrastructure provisioning and limit the ability to scale? Choose two correct answers.

- A. End-users have to request infrastructure changes
- B. Ticket based systems generate a full audit trail of the request and fulfillment process
- C. Users can access catalog of approved resources from drop down list in a request form

D. The more resources your organization needs, the more tickets your infrastructure team has to process

Answer: A

Explanation:

These are some of the ways that a ticket-based system can slow down infrastructure provisioning and limit the ability to scale, as they introduce delays, bottlenecks, and manual interventions in the process of creating and modifying infrastructure.

NEW QUESTION 80

Which backend does the Terraform CU use by default?

- A. Depends on the cloud provider configured
- B. HTTP
- C. Remote
- D. Terraform Cloud
- E. Local

Answer: E

Explanation:

This is the backend that the Terraform CLI uses by default, unless you specify a different backend in your configuration. The local backend stores the state file in a local file named terraform.tfstate, which can be used to track and manage the state of your infrastructure.

NEW QUESTION 85

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

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

Answer: C

Explanation:

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

NEW QUESTION 88

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 89

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 91

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

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

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

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

Answer: B

Explanation:

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

References:

? Terraform documentation on max function: Terraform Functions - max

NEW QUESTION 95

Infrastructure as Code (IaC) can be stored in a version control system along with application code.

- A. True
- B. False

Answer: A

Explanation:

Infrastructure as Code (IaC) can indeed be stored in a version control system along with application code. This practice is a fundamental principle of modern infrastructure management, allowing teams to apply software development practices like versioning, peer review, and CI/CD to infrastructure management. Storing IaC configurations in version control facilitates collaboration, history tracking, and change management. References = While this concept is a foundational aspect of IaC and is widely accepted in the industry, direct references from the HashiCorp Terraform Associate (003) study materials were not found in the provided files. However, this practice is encouraged in Terraform's best practices and various HashiCorp learning resources.

NEW QUESTION 100

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. templafil("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 104

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 109

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 113

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 114

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

- A. True
- B. False

Answer: B

Explanation:

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

References:

? Terraform Module Registry documentation: Terraform Registry

NEW QUESTION 117

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 120

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 124

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 128

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 131

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 132

What does terraform import do?

- A. Imports existing resources into the state file
- B. Imports all infrastructure from a given cloud provider
- C. Imports a new Terraform module
- D. Imports clean copies of tainted resources
- E. None of the above

Answer: A

Explanation:

The terraform import command is used to import existing infrastructure into your Terraform state. This command takes the existing resource and associates it with a resource defined in your Terraform configuration, updating the state file accordingly. It does not generate configuration for the resource, only the state.

NEW QUESTION 137

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 142

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 143

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 148

Which of the following module source paths does not specify a remote module?

- A. Source = ??module/consul????
- B. Source = ???github.com/crop/example????
- C. Source = ???git@github.com:hasicrop/example.git????
- D. Source = ???hasicrop/consul/aws????

Answer: A

Explanation:

The module source path that does not specify a remote module is source = "module/consul". This specifies a local module, which is a module that is stored in a subdirectory of the current working directory. The other options are all examples of remote modules, which are modules that are stored outside of the current working directory and can be accessed by various protocols, such as Git, HTTP, or the Terraform Registry. Remote modules are useful for sharing and reusing code across different configurations and environments. References = [Module Sources], [Local Paths], [Terraform Registry], [Generic Git Repository], [GitHub]

NEW QUESTION 153

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 158

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 161

.....

Thank You for Trying Our Product

* 100% Pass or Money Back

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

* One year free update

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

* Trusted by Millions

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

* Shop Securely

All transactions are protected by VeriSign!

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

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