



HashiCorp

Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)

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

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 3

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 4

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 5

terraform validate confirms that your infrastructure matches the Terraform state file.

- A. True
- B. False

Answer: B

Explanation:

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

NEW QUESTION 6

A developer on your team is going to tear down an existing deployment managed by Terraform and deploy a new one. However, there is a server resource named aws_instance.ubuntu[1] they would like to keep. What command should they use to tell Terraform to stop managing that specific resource?

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

Answer: B

Explanation:

To tell Terraform to stop managing a specific resource without destroying it, you can use the terraform state rm command. This command will remove the resource from the Terraform state, which means that Terraform will no longer track or update the corresponding remote object. However, the object will still exist in the remote system and you can later use terraform import to start managing it again in a different configuration or workspace. The syntax for this command is terraform state rm <address>,

where <address> is the resource address that identifies the resource instance to remove.

For example, terraform state rm aws_instance.ubuntu[1] will remove the second instance of the aws_instance resource named ubuntu from the state. References = : Command: state rm : Moving Resources

NEW QUESTION 7

If a module declares a variable with a default, that variable must also be defined within the module.

- A. True
- B. False

Answer: B

Explanation:

A module can declare a variable with a default value without requiring the caller to define it. This allows the module to provide a sensible default behavior that can be customized by the caller if needed. References = [Module Variables]

NEW QUESTION 8

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 9

What are some benefits of using Sentinel with Terraform Cloud/Terraform Enterprise? Choose three correct answers.

- A. You can restrict specific resource configurations, such as disallowing the use of CIDR=0.0.0.0/0.
- B. You can check out and check in cloud access keys
- C. Sentinel Policies can be written in HashiCorp Configuration Language (HCL)
- D. Policy-as-code can enforce security best practices
- E. You can enforce a list of approved AWS AMIs

Answer: ADE

Explanation:

Sentinel is a policy-as-code framework that allows you to define and enforce rules on your Terraform configurations, states, and plans¹. Some of the benefits of using Sentinel with Terraform Cloud/Terraform Enterprise are:

- You can restrict specific resource configurations, such as disallowing the use of CIDR=0.0.0.0/0, which would open up your network to the entire internet. This can help you prevent misconfigurations or security vulnerabilities in your infrastructure².

- Policy-as-code can enforce security best practices, such as requiring encryption, authentication, or compliance standards. This can help you protect your data and meet regulatory requirements³.

- You can enforce a list of approved AWS AMIs, which are pre-configured images that contain the operating system and software you need to run your applications. This can help you ensure consistency, reliability, and performance across your infrastructure⁴. References =

- 1: Terraform and Sentinel | Sentinel | HashiCorp Developer

- 2: Terraform Learning Resources: Getting Started with Sentinel in Terraform Cloud

- 3: Exploring the Power of HashiCorp Terraform, Sentinel, Terraform Cloud ??

- 4: Using New Sentinel Features in Terraform Cloud – Medium

NEW QUESTION 10

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 10

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 15

If a DevOps team adopts AWS CloudFormation as their standardized method for provisioning public cloud resources, which of the following scenarios poses a challenge for this team?

- A. The team is asked to manage a new application stack built on AWS-native services
- B. The organization decides to expand into Azure wishes to deploy new infrastructure
- C. The team is asked to build a reusable code based that can deploy resources into any AWS region
- D. The DevOps team is tasked with automating a manual, web console-based provisioning.

Answer: B

Explanation:

This is the scenario that poses a challenge for this team, if they adopt AWS CloudFormation as their standardized method for provisioning public cloud resources, as CloudFormation only supports AWS services and resources, and cannot be used to provision infrastructure on other cloud platforms such as Azure.

NEW QUESTION 18

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 19

What are some benefits of using Sentinel with Terraform Cloud/Terraform Enterprise? 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 22

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 26

Terraform can only manage resource dependencies if you set them explicitly with the depends_on argument.

- A. True
- B. False

Answer: B

Explanation:

Terraform can manage resource dependencies implicitly or explicitly. Implicit dependencies are created when a resource references another resource or data source in its arguments. Terraform can infer the dependency from the reference and create or destroy the resources in the correct order. Explicit dependencies are created when you use the depends_on argument to specify that a resource depends on another resource or module. This is useful when Terraform cannot infer the dependency from the configuration or when you need to create a dependency for some reason outside of Terraform's scope. References = : Create resource dependencies : Terraform Resource Dependencies Explained

NEW QUESTION 31

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 33

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 35

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 39

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 43

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 44

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 45

What kind of configuration block will create an infrastructure object with settings specified within the block?

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

Answer: D

Explanation:

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

NEW QUESTION 50

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 location. The other commands are not relevant for this task.

NEW QUESTION 55

Define the purpose of state in Terraform.

- A. State maps real world resources to your configuration and keeps track of metadata
- B. State lets you enforce resource configurations that relate to compliance policies
- C. State stores variables and lets you quickly reuse existing code
- D. State codifies the dependencies of related resources

Answer: A

Explanation:

The purpose of state in Terraform is to keep track of the real-world resources managed by Terraform, mapping them to the configuration. The state file contains metadata about these resources, such as resource IDs and other important attributes, which Terraform uses to plan and manage infrastructure changes. The state enables Terraform to know what resources are managed by which configurations and helps in maintaining the desired state of the infrastructure. This role of state in Terraform is outlined in Terraform's official documentation, emphasizing its function in mapping configuration to real-world resources and storing vital metadata.

NEW QUESTION 60

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 63

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 66

How can you trigger a run in a Terraform Cloud workspace that is connected to a Version Control System (VCS) repository?

- A. Only Terraform Cloud organization owners can set workspace variables on VCS connected workspaces
- B. Commit a change to the VCS working directory and branch that the Terraform Cloud workspace is connected to
- C. Only Terraform Cloud organization owners can approve plans in VCS connected workspaces
- D. Only members of a VCS organization can open a pull request against repositories that are connected to Terraform Cloud workspaces

Answer: B

Explanation:

This will trigger a run in the Terraform Cloud workspace, which will perform a plan and apply operation on the infrastructure defined by the Terraform configuration files in the VCS repository.

NEW QUESTION 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 75

In a Terraform Cloud workspace linked to a version control repository, speculative plan runs start automatically when you merge or commit changes to version control.

- A. True
- B. False

Answer: B

Explanation:

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

NEW QUESTION 77

Which two steps are required to provision new infrastructure in the Terraform workflow? Choose two correct answers.

- A. Plan
- B. Import
- C. Alidate
- D. Init
- E. apply

Answer: DE

Explanation:

The two steps that are required to provision new infrastructure in the Terraform workflow are init and apply. The terraform init command initializes a working directory containing Terraform configuration files. It downloads and installs the provider plugins that are needed for the configuration, and prepares the backend for storing the state. The terraform apply command applies the changes required to reach the desired state of the configuration, as described by the resource definitions in the configuration files. It shows a plan of the proposed changes and asks for confirmation before making any changes to the infrastructure. References = [The Core Terraform Workflow], [Initialize a Terraform working directory with init], [Apply Terraform Configuration with apply]

NEW QUESTION 80

How does Terraform manage most dependencies between resources?

- A. Terraform will automatically manage most resource dependencies
- B. Using the depends_on parameter
- C. By defining dependencies as modules and including them in a particular order
- D. The order that resources appear in Terraform configuration indicates dependencies

Answer: A

Explanation:

This is how Terraform manages most dependencies between resources, by using the references between them in the configuration files. For example, if resource A depends on resource B, Terraform will create resource B first and then pass its attributes to resource A.

NEW QUESTION 81

Which task does terraform init not perform?

- A. Validates all required variables are present
- B. Sources any modules and copies the configuration locally
- C. Connects to the backend
- D. Sources all providers used in the configuration and downloads them

Answer: A

Explanation:

The terraform init command is used to initialize a working directory containing Terraform configuration files. This command performs several different initialization steps to prepare the current working directory for use with Terraform, which includes initializing the backend, installing provider plugins, and copying any modules referenced in the configuration. However, it does not validate whether all required variables are present; that is a task performed by terraform plan or terraform apply.

References = This information can be verified from the official Terraform documentation on the terraform init command provided by HashiCorp Developer.

NEW QUESTION 86

terraform validate reports syntax check errors for which of the following?

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

Answer: D

Explanation:

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

NEW QUESTION 91

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. 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 94

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 98

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. templatefile("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 102

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 107

If you update the version constraint in your Terraform configuration, Terraform will update your lock file the next time you run terraform Init.

- A. True
- B. False

Answer: A

Explanation:

If you update the version constraint in your Terraform configuration, Terraform will update your lock file the next time you run terraform init. This will ensure that you use the same provider versions across different machines and runs.

NEW QUESTION 110

Which of the following is not a valid Terraform variable type?

- A. list
- B. array
- C. map
- D. string

Answer: B

Explanation:

This is not a valid Terraform variable type. The other options are valid variable types that can store different kinds of values.

NEW QUESTION 114

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

- A. Make the change via the public cloud API endpoint
- B. Clone the repository containing your infrastructure code and then run the code
- C. Use the public cloud console to make the change after a database record has been approved
- D. Make the change programmatically via the public cloud CLI
- E. Submit a pull request and wait for an approved merge of the proposed changes

Answer: E

Explanation:

You do not need to use different Terraform commands depending on the cloud provider you use. Terraform commands are consistent across different providers, as they operate on the Terraform configuration files and state files, not on the provider APIs directly.

NEW QUESTION 117

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 account
- 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 122

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 126

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 129

You have created a main.tf Terraform configuration consisting of an application server, a database and a load balanced. You ran terraform apply and Terraform created all of the resources successfully.

Now you realize that you do not actually need the load balancer, so you run terraform destroy without any flags. What will happen?

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

Answer: C

Explanation:

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

NEW QUESTION 130

Which of the following is not a benefit of adopting infrastructure as code?

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

Answer: B

Explanation:

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

References = The benefits of IaC can be verified from the official HashiCorp documentation on [What is Infrastructure as Code with Terraform](#) provided by HashiCorp Developer1.

NEW QUESTION 133

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

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

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

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

Answer: B

Explanation:

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

NEW QUESTION 138

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 140

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 141

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 142

You can reference a resource created with `for_each` using a Splat (`*`) expression.

- A. True
- B. False

Answer: B

Explanation:

You cannot reference a resource created with `for_each` using a splat (`*`) expression, as it will not work with resources that have non-numeric keys. You need to use a `for` expression instead to iterate over the resource instances.

NEW QUESTION 144

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 145

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 147

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 152

A Terraform provider is NOT responsible for:

- A. Exposing resources and data sources based on an API
- B. Managing actions to take based on resource 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 153

.....

Thank You for Trying Our Product

We offer two products:

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

2nd - Questions 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](#)