

HashiCorp

Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)



NEW QUESTION 1

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 2

You must initialize your working directory before running terraform validate.

- A. True
- B. False

Answer: A

Explanation:

You must initialize your working directory before running terraform validate, as it will ensure that all the required plugins and modules are installed and configured properly. If you skip this step, you may encounter errors or inconsistencies when validating your configuration files.

NEW QUESTION 3

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 4

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_PAID
- 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 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

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 7

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 8

Only the user that generated a plan may apply it.

- A. True
- B. False

Answer: B

Explanation:

Any user with permission to apply a plan can apply it, not only the user that generated it. This allows for collaboration and delegation of tasks among team members.

NEW QUESTION 9

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

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 13

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 14

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 15

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 18

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 19

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 20

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 23

Which of the following methods, used to provision resources into a public cloud, demonstrates the concept of infrastructure as code?

- A. curl commands manually run from a terminal
- B. A sequence of REST requests you pass to a public cloud API endpoint Most Voted
- C. A script that contains a series of public cloud CLI commands
- D. A series of commands you enter into a public cloud console

Answer: C

Explanation:

The concept of infrastructure as code (IaC) is to define and manage infrastructure using code, rather than manual processes or GUI tools. A script that contains a series of public cloud CLI commands is an example of IaC, because it uses code to provision resources into a public cloud. The other options are not examples of IaC, because they involve manual or interactive actions, such as running curl commands, sending REST requests, or entering commands into a console. References = [Introduction to Infrastructure as Code with Terraform] and [Infrastructure as Code]

NEW QUESTION 28

Module version is required to reference a module on the Terraform Module Registry.

- A. True
- B. False

Answer: B

Explanation:

Module version is optional to reference a module on the Terraform Module Registry. If you omit the version constraint, Terraform will automatically use the latest available version of the module

NEW QUESTION 30

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 module
- 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 35

Which are forbidden actions when the terraform state file is locked? Choose three correct answers.

- A. Terraform state list
- B. Terraform destroy
- C. Terraform validate
- D. Terraform validate
- E. Terraform for
- F. Terraform apply

Answer: BCF

Explanation:

The terraform state file is locked when a Terraform operation that could write state is in progress. This prevents concurrent state operations that could corrupt the state.

The forbidden actions when the state file is locked are those that could write state, such as terraform apply, terraform destroy, terraform refresh, terraform taint, terraform

untaint, terraform import, and terraform state *. The terraform validate command is also forbidden, because it requires an initialized working directory with the state file. The allowed actions when the state file is locked are those that only read state, such as terraform plan, terraform show, terraform output, and terraform console. References = [State Locking] and [Command: validate]

NEW QUESTION 37

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 41

When do changes invoked by terraform apply take effect?

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

Answer: B

Explanation:

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

NEW QUESTION 45

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 47

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 48

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]1

NEW QUESTION 49

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

- A. When you change a Terraform-managed resource via the Azure Cloud Console, Terraform updates the state file to reflect the change during the next plan or apply
- B. Changing resources via the Azure Cloud Console records the change in the current state file
- C. When you change a resource via the Azure Cloud Console, Terraform records the changes in a new state file
- D. Changing resources via the Azure Cloud Console does not update current state file

Answer: AD

Explanation:

Terraform state is a representation of the infrastructure that Terraform manages. Terraform uses state to track the current status of the resources it creates and to plan future changes. However, Terraform state is not aware of any changes made to the resources outside of Terraform, such as through the Azure Cloud Console, the Azure CLI, or the Azure API. Therefore, changing resources via the Azure Cloud Console does not update the current state file, and it may cause inconsistencies or conflicts with Terraform's desired configuration. To avoid this, it is recommended to manage resources exclusively through Terraform or to use the terraform import command to bring existing resources under Terraform's control.

When you change a Terraform-managed resource via the Azure Cloud Console, Terraform does not immediately update the state file to reflect the change. However, the next time you run terraform plan or terraform apply, Terraform will compare the state file with the actual state of the resources in Azure and detect any drifts or differences. Terraform will then update the state file to match the current state of the resources and show you the proposed changes in the execution plan. Depending on the configuration and the change, Terraform may try to undo the change, modify the resource further, or recreate the resource entirely. To avoid unexpected or destructive changes, it is recommended to review the execution plan carefully before applying it or to use the terraform refresh command to update the state file without applying any changes.

References = Purpose of Terraform State, Terraform State, Managing State, Importing Infrastructure, [Command: plan], [Command: apply], [Command: refresh]

NEW QUESTION 54

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 58

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 60

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 63

Where can Terraform not load a provider from?

- A. Plugins directory
- B. Provider plugin chance
- C. Official HashCrop Distribution on releases.hashcrop.com
- D. Source code

Answer: D

Explanation:

This is where Terraform cannot load a provider from, as it requires a compiled binary file that implements the provider protocol. You can load a provider from a plugins directory, a provider plugin cache, or the official HashiCorp distribution on releases.hashicorp.com.

NEW QUESTION 66

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

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

Answer: D

Explanation:

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

NEW QUESTION 71

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

- A. Terraform automatically formats configuration on terraform apply

- B. Run terraform validate prior to executing terraform plan or terraform apply
- C. Use terraform fmt
- D. Replace all tabs with spaces

Answer: C

Explanation:

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

References:

? Terraform documentation on terraform fmt: Terraform Fmt

NEW QUESTION 73

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 76

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 81

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 85

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 89

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

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

Answer: BC

Explanation:

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

NEW QUESTION 92

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 93

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 96

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 itself.

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

NEW QUESTION 98

When using a remote backend or terraform Cloud integration, where does Terraform save resource state?

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

Answer: C

Explanation:

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

NEW QUESTION 103

Which of the following is not true of Terraform providers?

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

Answer: F

Explanation:

All of the statements are true of Terraform providers. Terraform providers are plugins that enable Terraform to interact with various APIs and services¹. Anyone can write a Terraform provider, either as an individual or as part of a community². HashiCorp maintains some providers, such as the AWS, Azure, and Google Cloud providers³. Cloud providers and infrastructure vendors can also write, maintain, or collaborate on Terraform providers, such as the VMware, Oracle, and Alibaba Cloud providers. References =

- ¹: Providers - Configuration Language | Terraform | HashiCorp Developer
- ²: Plugin Development - How Terraform Works With Plugins | Terraform | HashiCorp Developer
- ³: Terraform Registry
- : Terraform Registry

NEW QUESTION 106

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 107

backends support state locking.

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

Answer: C

Explanation:

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

NEW QUESTION 110

What is terraform refresh-only intended to detect?

- A. Terraform configuration code changes
- B. Corrupt state files
- C. State file drift
- D. Empty state files

Answer: C

Explanation:

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

NEW QUESTION 111

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 115

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 119

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 121

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 123

Which of these statements about Terraform Cloud workspaces is false?

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

Answer: B

Explanation:

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

NEW QUESTION 124

Which command must you first run before performing further Terraform operations in a working directory?

- A. terraform import
- B. terraform workspace
- C. terraform plan
- D. terraform init

Answer: D

Explanation:

terraform init is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control. It initializes a working directory containing Terraform configuration files and downloads any required providers and modules. The other commands are used for different purposes, such as importing existing resources, switching between workspaces, generating execution plans, etc.

NEW QUESTION 128

.....

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](#)