

Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)

<https://www.2passeasy.com/dumps/Terraform-Associate-003/>



NEW QUESTION 1

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

```
resource "aws_instance" "example" {  
  ami = "ami-abc123"  
  instance_type = "t2.micro"  
  
  ebs_block_device {  
    device_name = "sda2"  
    volume_size = 16  
  }  
  
  ebs_block_device {  
    device_name = "sda3"  
    volume_size = 20  
  }  
}
```

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

Answer: D

Explanation:

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

NEW QUESTION 2

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 3

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 4

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 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

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 method for sharing Terraform configurations fulfills the following criteria:

- * 1. Keeps the configurations confidential within your organization
- * 2. Support Terraform's semantic version constraints
- * 3. Provides a browsable directory

- A. Subfolder within a workspace
- B. Generic git repository
- C. Terraform Cloud private registry
- D. Public Terraform module registry

Answer: C

Explanation:

This is the method for sharing Terraform configurations that fulfills the following criteria:

- ? Keeps the configurations confidential within your organization
- ? Supports Terraform's semantic version constraints
- ? Provides a browsable directory

The Terraform Cloud private registry is a feature of Terraform Cloud that allows you to host and manage your own modules within your organization, and use them in your Terraform configurations with versioning and access control.

NEW QUESTION 9

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 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 loss¹.

NEW QUESTION 10

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 13

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 18

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 21

Setting the TF_LOG environment variable to DEBUG causes debug messages to be logged into stdout.

- A. True
- B. False

Answer: A

Explanation:

Setting the TF_LOG environment variable to DEBUG causes debug messages to be logged into stdout, along with other log levels such as TRACE, INFO, WARN, and ERROR. This can be useful for troubleshooting or debugging purposes.

NEW QUESTION 24

You can develop a custom provider to manage its resources using Terraform.

- A. True
- B. False

Answer: A

Explanation:

You can develop a custom provider to manage its resources using Terraform, as Terraform is an extensible tool that allows you to write your own plugins in Go language. You can also publish your custom provider to the Terraform Registry or use it privately.

NEW QUESTION 26

You want to define multiple data disks as nested blocks inside the resource block for a virtual machine. What Terraform feature would help you define the blocks using the values in a variable?

- A. Local values
- B. Count arguments
- C. Collection functions
- D. Dynamic blocks

Answer: D

Explanation:

Dynamic blocks in Terraform allow you to define multiple nested blocks within a resource based on the values of a variable. This feature is particularly useful for scenarios where the number of nested blocks is not fixed and can change based on variable input.

NEW QUESTION 30

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 33

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 38

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

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

Answer: B

Explanation:

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

NEW QUESTION 40

When should you write Terraform configuration files for existing infrastructure that you want to start managing with Terraform?

- A. You can import infrastructure without corresponding Terraform code

- B. Terraform will generate the corresponding configuration files for you
- C. Before you run terraform Import
- D. After you run terraform import

Answer: C

Explanation:

You need to write Terraform configuration files for the existing infrastructure that you want to import into Terraform, otherwise Terraform will not know how to manage it. The configuration files should match the type and name of the resources that you want to import.

NEW QUESTION 44

How does the Terraform cloud integration differ from other state backends such as S3, Consul,etc?

- A. It can execute Terraform runs on dedicated infrastructure in Terraform Cloud
- B. It doesn't show the output of a terraform apply locally
- C. It is only arable lo paying customers
- D. All of the above

Answer: A

Explanation:

This is how the Terraform Cloud integration differs from other state backends such as S3, Consul, etc., as it allows you to perform remote operations on Terraform Cloud's servers instead of your local machine. The other options are either incorrect or irrelevant.

NEW QUESTION 48

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 51

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 53

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 54

A provider configuration block is required in every Terraform configuration.

Example:

```
provider "provider_name" {
  ...
}
```

- A. True
- B. False

Answer: B

Explanation:

A provider configuration block is not required in every Terraform configuration. A provider configuration block can be omitted if its contents would otherwise be empty. Terraform assumes an empty default configuration for any provider that is not explicitly configured. However, some providers may require some configuration arguments (such as endpoint URLs or cloud regions) before they can be used. A provider's documentation should list which configuration arguments it expects. For providers distributed on the Terraform Registry, versioned documentation is available on each provider's page, via the Documentation link in the provider's header¹. References = [Provider Configuration]¹

NEW QUESTION 56

What is the provider for this resource?

```
resource "aws_vpc" "main" {
  name = "test"
}
```

- A. Vpc
- B. Test
- C. Main
- D. aws

Answer: D

Explanation:

In the given Terraform configuration snippet: resource "aws_vpc" "main" { name = "test" }

The provider for the resource aws_vpc is aws. The provider is specified by the prefix of the resource type. In this case, aws_vpc indicates that the resource type vpc is provided by the aws provider.

References:

? Terraform documentation on providers: Terraform Providers

NEW QUESTION 57

You have multiple team members collaborating on infrastructure as code (IaC) using Terraform, and want to apply formatting standards for readability. How can you format Terraform HCL (HashiCorp Configuration Language) code according to standard Terraform style convention?

- A. Run the terraform fmt command during the code linting phase of your CI/CD process Most Voted
- B. Designate one person in each team to review and format everyone's code
- C. Manually apply two spaces indentation and align equal sign "=" characters in every Terraform file (*.tf)
- D. Write a shell script to transform Terraform files using tools such as AWK, Python, and sed

Answer: A

Explanation:

The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability. Running this command on your configuration files before committing them to source control can help ensure consistency of style between different Terraform codebases, and can also make diffs easier to read. You can also use the -check and -diff options to check if the files are formatted and display the formatting changes respectively². Running the terraform fmt command during the code linting phase of your CI/CD process can help automate this process and enforce the formatting standards for your team. References = [Command: fmt]²

NEW QUESTION 59

Terraform providers are part of the Terraform core binary.

- A. True
- B. False

Answer: B

Explanation:

Terraform providers are not part of the Terraform core binary. Providers are distributed separately from Terraform itself and have their own release cadence and version numbers. Providers are plugins that Terraform uses to interact with various APIs, such as cloud providers, SaaS providers, and other services. You can find and install providers from the Terraform Registry, which hosts providers for most major infrastructure platforms. You can also load providers from a local mirror or cache, or develop your own custom providers. To use a provider in your Terraform configuration, you need to declare it in the provider requirements block and optionally configure its settings in the provider block. References = : Providers - Configuration Language | Terraform : Terraform Registry
- Providers Overview | Terraform

NEW QUESTION 64

You have used Terraform to create an ephemeral development environment in the cloud and are now ready to destroy all the Infrastructure described by your Terraform configuration. To be safe, you would like to first see all the infrastructure that Terraform will delete. Which command should you use to show all of the resources that will be deleted? Choose two correct answers.

- A. Run `terraform state rm` ??
- B. Run `terraform show :destroy`
- C. Run `terraform destroy` and it will first output all the resource that will be deleted before prompting for approval
- D. Run `terraform plan .destroy`

Answer: CD

Explanation:

To see all the resources that Terraform will delete, you can use either of these two commands:
? `terraform destroy` will show the plan of destruction and ask for your confirmation before proceeding. You can cancel the command if you do not want to destroy the resources.
? `terraform plan -destroy` will show the plan of destruction without asking for confirmation. You can use this command to review the changes before running `terraform destroy`. References = : Destroy Infrastructure : Plan Command: Options

NEW QUESTION 66

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 70

Which of these is true about Terraform's plugin-based architecture?

- A. Terraform can only source providers from the internet
- B. Every provider in a configuration has its own state file for its resources
- C. You can create a provider for your API if none exists
- D. All providers are part of the Terraform core binary

Answer: C

Explanation:

Terraform is built on a plugin-based architecture, enabling developers to extend Terraform by writing new plugins or compiling modified versions of existing plugins¹. Terraform plugins are executable binaries written in Go that expose an implementation for a specific service, such as a cloud resource, SaaS platform, or API². If there is no existing provider for your API, you can create one using the Terraform Plugin SDK³ or the Terraform Plugin Framework⁴. References =
•1: Plugin Development - How Terraform Works With Plugins | Terraform | HashiCorp Developer
•2: Lab: Terraform Plug-in Based Architecture - GitHub
•3: Terraform Plugin SDK - Terraform by HashiCorp
•4: HashiCorp Terraform Plugin Framework Now Generally Available

NEW QUESTION 75

What is a key benefit of the Terraform state file?

- A. A state file can schedule recurring infrastructure tasks
- B. A state file is a source of truth for resources provisioned with Terraform
- C. A state file is a source of truth for resources provisioned with a public cloud console
- D. A state file is the desired state expressed by the Terraform code files

Answer:

B

Explanation:

This is a key benefit of the Terraform state file, as it stores and tracks the metadata and attributes of the resources that are managed by Terraform, and allows Terraform to compare the current state with the desired state expressed by your configuration files.

NEW QUESTION 76

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 things1:

? 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 78

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 79

You are creating a Terraform configuration which needs to make use of multiple providers, one for AWS and one for Datadog. Which of the following provider blocks would allow you to do this?

A)

```
terraform {  
  provider "aws" {  
    profile = var.aws_profile  
    region  = var.aws_region  
  }  
  
  provider "datadog" {  
    api_key = var.datadog_api_key  
    app_key = var.datadog_app_key  
  }  
}
```

B)

```
provider "aws" {  
  profile = var.aws_profile  
  region  = var.aws_region  
}  
  
provider "datadog" {  
  api_key = var.datadog_api_key  
  app_key = var.datadog_app_key  
}
```

C)

```
provider "aws" {  
  profile = var.aws_profile  
  region  = var.aws_region  
}  
  
provider "datadog" {  
  api_key = var.datadog_api_key  
  app_key = var.datadog_app_key  
}
```

D)

```
provider {  
  "aws" {  
    profile = var.aws_profile  
    region  = var.aws_region  
  }  
  
  "datadog" {  
    api_key = var.datadog_api_key  
    app_key = var.datadog_app_key  
  }  
}
```

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

Answer: C

Explanation:

Option C is the correct way to configure multiple providers in a Terraform configuration. Each provider block must have a name attribute that specifies which provider it configures². The other options are either missing the name attribute or using an invalid syntax.

NEW QUESTION 82

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 84

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 86

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

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

Answer: A

Explanation:

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

NEW QUESTION 89

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 91

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 93

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

- A. list
- B. array
- C. nap
- 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 values².

NEW QUESTION 95

What does the default "local" Terraform backend store?

- A. tfplan files
- B. State file
- C. Provider plugins
- D. Terraform binary

Answer: B

Explanation:

The default ??local?? Terraform backend stores the state file in a local file named terraform.tfstate, which can be used to track and manage the state of your infrastructure³.

NEW QUESTION 100

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 102

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 107

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 110

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 115

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 119

A Terraform output that sets the "sensitive" argument to true will not store that value in the state file.

- A. True
- B. False

Answer: A

Explanation:

A Terraform output that sets the "sensitive" argument to true will store that value in the state file. The purpose of setting sensitive = true is to prevent the value from being displayed in the CLI output during terraform plan and terraform apply, and to mask it in the Terraform UI. However, it does not affect the storage of the value in the state file. Sensitive outputs are still written to the state file to ensure that Terraform can manage resources correctly during subsequent operations.

References:

? Terraform documentation on sensitive outputs: Terraform Output Values

NEW QUESTION 120

Which of the following commands would you use to access all of the attributes and details of a resource managed by Terraform?

- A. terraform state list ??provider_type.name??
- B. terraform state show ??provider_type.name??
- C. terraform get ??provider_type.name??
- D. terraform state list

Answer: B

Explanation:

The terraform state show command allows you to access all of the attributes and details of a resource managed by Terraform. You can use the resource address (e.g. provider_type.name) as an argument to show the information about a specific resource. The terraform state list command only shows the list of resources in the state, not their attributes. The terraform get command downloads and installs modules needed for the configuration. It does not show any information about resources. References = [Command: state show] and [Command: state list]

NEW QUESTION 124

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 128

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 130

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 135

.....

THANKS FOR TRYING THE DEMO OF OUR PRODUCT

Visit Our Site to Purchase the Full Set of Actual Terraform-Associate-003 Exam Questions With Answers.

We Also Provide Practice Exam Software That Simulates Real Exam Environment And Has Many Self-Assessment Features. Order the Terraform-Associate-003 Product From:

<https://www.2passeasy.com/dumps/Terraform-Associate-003/>

Money Back Guarantee

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