

Terraform-Associate-003 Dumps

HashiCorp Certified: Terraform Associate (003)

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



NEW QUESTION 1

Which of the following is not a key principle of infrastructure as code?

- A. Self-describing infrastructure
- B. Idempotence
- C. Versioned infrastructure
- D. Golden images

Answer: D

Explanation:

The key principle of infrastructure as code that is not listed among the options is golden images. Golden images are pre-configured, ready-to-use virtual machine images that contain a specific set of software and configuration. They are often used to create multiple identical instances of the same environment, such as for testing or production. However, golden images are not a principle of infrastructure as code, but rather a technique that can be used with or without infrastructure as code. The other options are all key principles of infrastructure as code, as explained below:

? Self-describing infrastructure: This means that the infrastructure is defined in code that describes its desired state, rather than in scripts that describe the steps to create it. This makes the infrastructure easier to understand, maintain, and reproduce.

? Idempotence: This means that applying the same infrastructure code multiple times will always result in the same state, regardless of the initial state. This makes the infrastructure consistent and predictable, and avoids errors or conflicts caused by repeated actions.

? Versioned infrastructure: This means that the infrastructure code is stored in a version control system, such as Git, that tracks the changes and history of the code. This makes the infrastructure code reusable, auditable, and collaborative, and enables practices such as branching, merging, and rollback. References = [Introduction to Infrastructure as Code with Terraform], [Infrastructure as Code in a Private or Public Cloud]

NEW QUESTION 2

A terraform apply can not infrastructure.

- A. change
- B. destroy
- C. provision
- D. import

Answer: D

Explanation:

The terraform import command is used to import existing infrastructure into Terraform's state. This allows Terraform to manage and destroy the imported infrastructure as part of the configuration. The terraform import command does not modify the configuration, so the imported resources must be manually added to the configuration after the import. References = [Importing Infrastructure]

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

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 5

Why would you use the -replace flag for terraform apply?

- A. You want Terraform to ignore a resource on the next apply
- B. You want Terraform to destroy all the infrastructure in your workspace
- C. You want to force Terraform to destroy a resource on the next apply
- D. You want to force Terraform to destroy and recreate a resource on the next apply

Answer: D

Explanation:

The `-replace` flag is used with the `terraform apply` command when there is a need to explicitly force Terraform to destroy and then recreate a specific resource during the next apply. This can be necessary in situations where a simple update is insufficient or when a resource must be re-provisioned to pick up certain changes.

NEW QUESTION 6

When should you use the `force-unlock` command?

- A. You have a high priority change
- B. Automatic unlocking failed
- C. apply failed due to a state lock
- D. You see a status message that you cannot acquire the lock

Answer: B

Explanation:

You should use the `force-unlock` command when automatic unlocking failed. Terraform will lock your state for all operations that could write state, such as `plan`, `apply`, or `destroy`. This prevents others from acquiring the lock and potentially corrupting your state. State locking happens automatically on all operations that could write state and you won't see any message that it is happening. If state locking fails, Terraform will not continue. You can disable state locking for most commands with the `-lock` flag but it is not recommended. If acquiring the lock is taking longer than expected, Terraform will output a status message. If Terraform doesn't output a message, state locking is still occurring if your backend supports it. Terraform has a `force-unlock` command to manually unlock the state if unlocking failed. Be very careful with this command. If you unlock the state when someone else is holding the lock it could cause multiple writers. Force unlock should only be used to unlock your own lock in the situation where automatic unlocking failed. To protect you, the `force-unlock` command requires a unique lock ID. Terraform will output this lock ID if unlocking fails. This lock ID acts as a nonce, ensuring that locks and unlocks target the correct lock. The other situations are not valid reasons to use the `force-unlock` command. You should not use the `force-unlock` command if you have a high priority change, if apply failed due to a state lock, or if you see a status message that you cannot acquire the lock. These situations indicate that someone else is holding the lock and you should wait for them to finish their operation or contact them to resolve the issue. Using the `force-unlock` command in these cases could result in data loss or inconsistency. References = [State Locking], [Command: force-unlock]

NEW QUESTION 7

`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 consistent. 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 8

A developer on your team is going to leave 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 9

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 10

You cannot install third party plugins using terraform init.

- A. True
- B. False

Answer: B

Explanation:

You can install third party plugins using terraform init, as long as you specify the plugin directory in your configuration or as a command-line argument. You can also use the terraform providers mirror command to create a local mirror of providers from any source.

NEW QUESTION 10

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 12

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 16

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 18

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 21

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 24

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 27

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 29

You add a new resource to an existing Terraform configuration, but do not update the version constraint in the configuration. The existing and new resources use the same provider. The working contains a .terraform.lock, hc1 file.

How will Terraform choose which version of the provider to use?

- A. Terraform will use the version recorded in your lock file
- B. Terraform will use the latest version of the provider for the new resource and the version recorded in the lock file to manage existing resources
- C. Terraform will check your state file to determine the provider version to use
- D. Terraform will use the latest version of the provider available at the time you provision your new resource

Answer: A

Explanation:

This is how Terraform chooses which version of the provider to use, when you add a new resource to an existing Terraform configuration, but do not update the version constraint in the configuration. The lock file records the exact version of each provider that was installed in your working directory, and ensures that Terraform will always use the same provider versions until you run terraform init -upgrade to update them.

NEW QUESTION 34

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 38

Which of the following is not a valid string function in Terraform?

- A. choaf
- B. join
- C. Split
- D. slice

Answer: A

Explanation:

This is not a valid string function in Terraform. The other options are valid string functions that can manipulate strings in various ways.

NEW QUESTION 40

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 43

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 46

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 47

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 48

If you manually destroy infrastructure, what is the best practice reflecting this change in Terraform?

- A. Run terraform refresh
- B. It will happen automatically
- C. Manually update the state file
- D. Run terraform import

Answer: B

Explanation:

If you manually destroy infrastructure, Terraform will automatically detect the change and update the state file during the next plan or apply. Terraform compares the current state of the infrastructure with the desired state in the configuration and generates a plan to reconcile the differences. If a resource is missing from the infrastructure but still exists in the state file, Terraform will attempt to recreate it. If a resource is present in the infrastructure but not in the state file, Terraform will ignore it unless you use the terraform import command to bring it under Terraform's management. References = [Terraform State]

NEW QUESTION 53

Outside of the required_providers block, Terraform configurations always refer to providers by their local names.

- A. True
- B. False

Answer: B

Explanation:

Outside of the required_providers block, Terraform configurations can refer to providers by either their local names or their source addresses. The local name is a short name that can be used throughout the configuration, while the source address is a global identifier for the provider in the format registry.terraform.io/namespace/type. For example, you can use either aws or registry.terraform.io/hashicorp/aws to refer to the AWS provider.

NEW QUESTION 54

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 59

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 60

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 63

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 64

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 65

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 .destory

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 70

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 71

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 75

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

- A. End-users have to request infrastructure changes
- B. Ticket based systems generate a full audit trail of the request and fulfillment process
- C. Users can access catalog of approved resources from drop down list in a request form
- D. The more resources your organization needs, the more tickets your infrastructure team has to process

Answer: A

Explanation:

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

NEW QUESTION 80

When does Sentinel enforce policy logic during a Terraform Cloud run?

- A. Before the plan phase
- B. During the plan phase
- C. Before the apply phase
- D. After the apply phase

Answer: C

Explanation:

Sentinel policies are checked after the plan stage of a Terraform run, but before it can be confirmed or the terraform apply is executed. This allows you to enforce rules on your infrastructure before it is created or modified.

NEW QUESTION 82

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 87

Which backend does the Terraform CU use by default?

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

Answer: E

Explanation:

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

NEW QUESTION 88

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 93

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 96

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 init3. This will ensure that you use the same provider versions across different machines and runs.

NEW QUESTION 98

How would you reference the "name???? value of the second instance of this resource?

```
resource "aws_instance" "web" {
  count = 2
  name = "terraform-${count.index}"
}
```

- A. aws_instance.web(2),name
- B. element(aws_instance.web, 2)
- C. aws_instance-web(1)
- D. aws_instance_web(1),name
- E. Aws_instance,web,* , name

Answer: D

Explanation:

In Terraform, when you use the count meta-argument, you can reference individual instances using an index. The indexing starts at 0, so to reference the "name" value of the second instance, you would use aws_instance.web[1].name. This syntax allows you to access the properties of specific instances in a list generated by the count argument.

References:

? Terraform documentation on count and accessing resource instances: Terraform Count

NEW QUESTION 99

Which of these commands makes your code more human readable?

- A. Terraform validate
- B. Terraform output
- C. Terraform show
- D. Terraform fmt

Answer: D

Explanation:

The command that makes your code more human readable is terraform fmt. This command is used to rewrite Terraform configuration files to a canonical format and style, following the Terraform language style conventions and other minor adjustments for readability. The command is optional, opinionated, and has no customization options, but it is recommended to ensure consistency of style across different Terraform codebases. Consistency can help your team understand the code more quickly and easily, making the use of terraform fmt very important. You can run this command on your configuration files before committing them to source control or as part of your CI/CD pipeline. References =

: Command: fmt : Using Terraform fmt Command to Format Your Terraform Code

NEW QUESTION 103

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

Answer: AD

Explanation:

? It lets you version, reuse, and share infrastructure configuration as code files, which can be stored in a source control system and integrated with your CI/CD pipeline.

? It reduces risk of operator error by automating repetitive tasks and ensuring consistency across environments. IaC does not necessarily provision resources at a lower cost, secure your credentials, or prevent manual modifications to your resources - these depend on other factors such as your cloud provider, your security practices, and your access policies.

NEW QUESTION 108

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 110

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 infrastructure.

NEW QUESTION 112

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

- A. True
- B. False

Answer: B

Explanation:

Not all modules published on the official Terraform Module Registry have been verified by HashiCorp. While HashiCorp verifies some modules, there are many

community-contributed modules that are not verified. Verified modules have a "Verified" badge indicating that HashiCorp has reviewed them for security and best practices, but the registry also includes unverified modules.

References:

? Terraform Module Registry documentation: Terraform Registry

NEW QUESTION 117

You 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 127

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 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 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 Developer1.

NEW QUESTION 131

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 135

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 137

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 141

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 143

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

- A. Source = ??module/consul????
- B. Source = ???githhub.comicrop/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 147

.....

Thank You for Trying Our Product

* 100% Pass or Money Back

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

* One year free update

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

* Trusted by Millions

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

* Shop Securely

All transactions are protected by VeriSign!

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

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