HashiCorp

Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)

About Exambible

Your Partner of IT Exam

Found in 1998

Exambible is a company specialized on providing high quality IT exam practice study materials, especially Cisco CCNA, CCDA, CCNP, CCIE, Checkpoint CCSE, CompTIA A+, Network+ certification practice exams and so on. We guarantee that the candidates will not only pass any IT exam at the first attempt but also get profound understanding about the certificates they have got. There are so many alike companies in this industry, however, Exambible has its unique advantages that other companies could not achieve.

Our Advances

* 99.9% Uptime

All examinations will be up to date.

* 24/7 Quality Support

We will provide service round the clock.

* 100% Pass Rate

Our guarantee that you will pass the exam.

* Unique Gurantee

If you do not pass the exam at the first time, we will not only arrange FULL REFUND for you, but also provide you another exam of your claim, ABSOLUTELY FREE!

NEW QUESTION 1
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 2
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 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 releas
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_PAIH
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
A developer on your team is going lo leaf down an existing deployment managed by Terraform and deploy a new one. However, there is a server resource named aws instant.ubuntu[l] 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 destory 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 6
Which is the best way to specify a tag of v1.0.0 when referencing a module stored in Git (for example.
Git::https://example.com/vpc.git)?

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

Answer: A

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

NEW QUESTION 7
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 8
Your DevOps team is currently using the local backend for your Terraform configuration. You would like to move to a remote backend to store the state file in a central location. Which of the following backends would not work?

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

Answer: D

Explanation:
This is not a valid backend for Terraform, as it does not support locking or versioning of state files4. The other options are valid backends that can store state files in a central location.

NEW QUESTION 9
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 10
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 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 14
You modified your Terraform configuration and run Terraform plan to review the changes. Simultaneously, your teammate manually modified the infrastructure component you are working on. Since you already ran terraform plan locally, the execution plan for terraform apply will be the same.

A. True
B. False

Answer: B

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

NEW QUESTION 15
You want to know from which paths Terraform is loading providers referenced in your Terraform configuration (* files). You need to enable additional logging messages to find this out. Which of the following would achieve this?

A. Set verbose for each provider in your Terraform configuration
B. Set the environment variable TF_LOG_TRACE
C. Set the environment variable TF_LOG_PATH
D. Set the environment variable TF_log_TRACE

Answer: B

Explanation:
This will enable additional logging messages to find out from which paths Terraform is loading providers referenced in your Terraform configuration files, as it will set the log level to TRACE, which is the most verbose and detailed level.

NEW QUESTION 19
Which of the following is not a valid siring 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 ways2.

NEW QUESTION 20
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 22
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 anything5. The other options are either incorrect or inefficient.

NEW QUESTION 25
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 27
Which of the following does terraform apply change after you approve the execution plan? (Choose two.)

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

Answer: AD

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

NEW QUESTION 32
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
Once you configure a new Terraform backend with a terraform code block, which command(s) should you use to migrate the state file?

A. terraform destroy, then terraform apply
B. terraform init
C. terraform push
D. terraform apply

Answer: A

Explanation:
This command will initialize the new backend and prompt you to migrate the existing state file to the new location4. The other commands are not relevant for this task.

NEW QUESTION 41
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 43
Which of these actions will prevent two Terraform runs from changing the same state file at the same time?

A. Refresh the state after running Terraform
B. Delete the state before running Terraform
C. Configure state locking for your state backend
D. Run Terraform with parallelism set to 1

Answer: B

Explanation:
To prevent two Terraform runs from changing the same state file simultaneously, state locking is used. State locking ensures that when one Terraform operation is running, others will be blocked from making changes to the same state, thus preventing conflicts and data corruption. This is achieved by configuring the state backend to support locking, which will lock the state for all operations that could write to the state.References = This information is supported by Terraform's official documentation,
which explains the importance of state locking and how it can be configured for different backends to prevent concurrent state modifications .

NEW QUESTION 46
You have used Terraform lo create an ephemeral development environment in the (loud 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 mil 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 48
It is best practice to store secret data in the same version control repository as your Terraform configuration.

A. True
B. False

Answer: B

Explanation:
It is not a best practice to store secret data in the same version control repository as your Terraform configuration, as it could expose your sensitive information to unauthorized parties or compromise your security. You should use environment variables, vaults, or other mechanisms to store and provide secret data to Terraform.

NEW QUESTION 52
Which Terraform collection type should you use to store key/value pairs?

A. Set
B. Map
C. Tuple
D. list

Answer: B

Explanation:
The Terraform collection type that should be used to store key/value pairs is map. A map is a collection of values that are accessed by arbitrary labels, called keys.
The keys and values can be of any type, but the keys must be unique within a map. For example, var = { key1 = "value1", key2 = "value2" } is a map with two key/value pairs. Maps are useful for grouping related values together, such as configuration options or
metadata. References = [Collection Types], [Map Type Constraints]

NEW QUESTION 55
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 57
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 61
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 63
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 68
Which two steps are required to provision new infrastructure in the Terraform workflow? Choose two correct answers.

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

Answer: DE

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

NEW QUESTION 71
How does Terraform manage most dependencies between resources?

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

Answer: A

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

NEW QUESTION 76
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 79
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 82
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 83
Which backend does the Terraform CU use by default?

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

Answer: E

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

NEW QUESTION 85
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
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 92
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 97
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 99
As a member of an operations team that uses infrastructure as code (lac) practices, you are tasked with making a change to an infrastructure stack running in a public cloud. Which pattern would follow laC 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 103
What does Terraform use the .terraform.lock.hc1 file for?

A. There is no such file
B. Tracking specific provider dependencies
C. Preventing Terraform runs from occurring
D. Storing references to workspaces which are locked

Answer: B

Explanation:
The .terraform.lock.hcl file is a new feature in Terraform 0.14 that records the exact versions of each provider used in your configuration. This helps ensure consistent
and reproducible behavior across different machines and runs.

NEW QUESTION 107
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 accoun
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 108
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 113
Which parameters does terraform import require? Choose two correct answers.

A. Provider
B. Resource ID
C. Resource address
D. Path

Answer: BC

Explanation:
These are the parameters that terraform import requires, as they allow
Terraform to identify the existing resource that you want to import into your state file, and match it with the corresponding configuration block in your files.

NEW QUESTION 117
You must use different Terraform commands depending on the cloud provider you use.

A. True
B. False

Answer: B

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

NEW QUESTION 118
What does Terraform not reference when running a terraform apply -refresh-only ?

A. State file
B. Credentials
C. Cloud provider
D. Terraform resource definitions in configuration files

Answer: D

Explanation:
When running a terraform apply -refresh-only, Terraform does not reference the configuration files, but only the state file, credentials, and cloud provider. The purpose of this command is to update the state file with the current status of the real resources, without making any changes to them1.

NEW QUESTION 123
When using a remote backend or terraform Cloud integration, where does Terraform save resource sate?

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 124
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 125
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 127
Any user can publish modules to the public Terraform Module Registry.

A. True
B. False

Answer: A

Explanation:
The Terraform Registry allows any user to publish and share modules. Published modules support versioning, automatically generate documentation, allow browsing version histories, show examples and READMEs, and more. Public modules are managed via Git and GitHub, and publishing a module takes only a few minutes. Once a module is published, releasing a new version of a module is as simple as pushing a properly formed Git tag1.
References = The information can be verified from the Terraform Registry documentation on Publishing Modules provided by HashiCorp Developer1.

NEW QUESTION 130
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 131
You can access state stored with the local backend by using terraform_remote_state data source.

A. True
B. False

Answer: B

Explanation:
You cannot access state stored with the local backend by using the terraform_remote_state data source. The terraform_remote_state data source is used to retrieve the root module output values from some other Terraform configuration using the latest state snapshot from the remote backend. It requires a backend that

supports remote state storage, such as S3, Consul, AzureRM, or GCS. The local backend stores the state file locally on the filesystem, which terraform_remote_state cannot access. References:
? Terraform documentation on terraform_remote_state data source: Terraform
Remote State Data Source
? Example usage of remote state: Example Usage (remote Backend)

NEW QUESTION 134
What is one disadvantage of using dynamic blocks in Terraform?

A. Dynamic blocks can construct repeatable nested blocks
B. Terraform will run more slowly
C. They cannot be used to loop through a list of values
D. They make configuration harder to read and understand

Answer: D

Explanation:
This is one disadvantage of using dynamic blocks in Terraform, as they can introduce complexity and reduce readability of the configuration. The other options are either advantages or incorrect statements.

NEW QUESTION 138
What is the Terraform style convention for indenting a nesting level compared to the one above it?

A. With a tab
B. With two spaces
C. With four spaces
D. With three spaces

Answer: B

Explanation:
This is the Terraform style convention for indenting a nesting level compared to the one above it. The other options are not consistent with the Terraform style guide.

NEW QUESTION 143
Which provider authentication method prevents credentials from being stored in the state file?

A. Using environment variables
B. Specifying the login credentials in the provider block
C. Setting credentials as Terraform variables
D. None of the above

Answer: D

Explanation:
None of the above methods prevent credentials from being stored in the state file. Terraform stores the provider configuration in the state file, which may include sensitive information such as credentials. This is a potential security risk and should be avoided if possible. To prevent credentials from being stored in the state file, you can use one of the following methods:
? Use environment variables to pass credentials to the provider. This way, the credentials are not part of the provider configuration and are not stored in the state file. However, this method may not work for some providers that require credentials to be set in the provider block.
? Use dynamic credentials to authenticate with your cloud provider. This way,
Terraform Cloud or Enterprise will request temporary credentials from your cloud provider for each run and use them to provision your resources. The credentials are not stored in the state file and are revoked after the run is completed. This method is supported for AWS, Google Cloud Platform, Azure, and Vault. References = : [Sensitive Values in State] : Authenticate providers with dynamic credentials

NEW QUESTION 148
Which of the following module source paths does not specify a remote module?

A. Source = ??module/consul????
B. Source = ????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 149
FILL IN THE BLANK
What is the name of the default file where Terraform stores the state?
Type your answer in the field provided. The text field is not case-sensitive and all variations of the correct answer are accepted.

A. Mastered
B. Not Mastered

Answer: A

Explanation:
The name of the default file where Terraform stores the state is terraform.tfstate. This file contains a JSON representation of the current state of the infrastructure managed by Terraform. Terraform uses this file to track the metadata and attributes of the resources, and to plan and apply changes. By default, Terraform stores the state file locally in the same directory as the configuration files, but it can also be configured to store the state remotely in a backend. References = [Terraform State], [State File Format]

NEW QUESTION 154
.....

Relate Links

100% Pass Your Terraform-Associate-003 Exam with Exambible Prep Materials

https://www.exambible.com/Terraform-Associate-003-exam/

Contact us

We are proud of our high-quality customer service, which serves you around the clock 24/7.

Viste - https://www.exambible.com/