HashiCorp

Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)

NEW QUESTION 1
When does Terraform create the .terraform.lock.hc1 file?

A. After your first terraform plan
B. After your first terraform apply
C. After your first terraform init
D. When you enable state locking

Answer: C

Explanation:
Terraform creates the .terraform.lock.hcl file after the first terraform init command. This lock file ensures that the dependencies for your project are consistent across different runs by locking the versions of the providers and modules used.

NEW QUESTION 2
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_lnstance.example.ebs_block_device.[*].volume_id
C. aws_lnstance.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 3
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 4
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 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
In Terraform HCL, an object type of object({name=string, age-number}) would match this value.
A)

```
{
  name = "John"
  age = fifty two
}
```

B)

```
{
  name = "John"
  age = 52
}
```

C)

```
{
  name = John
  age = "52"
}
```

D)

```
{
  name = John
  age = fifty two
}
```

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

Answer: B

NEW QUESTION 7
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 8
Which of the following statements about Terraform modules is not true?

A. Modules can call other modules
B. A module is a container for one or more resources
C. Modules must be publicly accessible
D. You can call the same module multiple times

Answer: C

Explanation:
This is not true, as modules can be either public or private, depending on your needs and preferences. You can use the Terraform Registry to publish and consume public modules, or use Terraform Cloud or Terraform Enterprise to host and manage private modules.

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

A. Tree
B. Map
C. List
D. set

Answer: A

Explanation:
This is not a valid Terraform collection type, as Terraform only supports three collection types: list, map, and set. A tree is a data structure that consists of nodes with parent-child relationships, which is not supported by Terraform.

NEW QUESTION 11
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 14
What Terraform command always causes a state file to be updated with changes that might have been made outside of Terraform?

A. Terraform plan –refresh-only
B. Terraform show –json
C. Terraform apply –lock-false
D. Terraform plan target-state

Answer: A

Explanation:
This is the command that always causes a state file to be updated with changes that might have been made outside of Terraform, as it will only refresh the state file with the current status of the real resources, without making any changes to them or creating a plan.

NEW QUESTION 18
All standard backend types support state locking, and remote operations like plan, apply, and destroy.

A. True
B. False

Answer: B

Explanation:
Not all standard backend types support state locking and remote operations like plan, apply, and destroy. For example, the local backend does not support remote operations and state locking. State locking is a feature that ensures that no two users can make changes to the state file at the same time, which is crucial for preventing race conditions. Remote operations allow running Terraform commands on a remote server, which is supported by some backends like remote or consul, but not all.
References:
? Terraform documentation on backends: Terraform Backends
? Detailed backend support: Terraform Backend Types

NEW QUESTION 20
Only the user that generated a plan may apply it.

A. True
B. False

Answer: B

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

NEW QUESTION 22
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 24
Which method for sharing Terraform configurations fulfills the following criteria:
* 1. Keeps the configurations confidential within your organization
* 2. Support Terraform??s semantic version constrains
* 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 26
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
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 30
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 31
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 36
What value does the Terraform Cloud private registry provide over the public Terraform Module Registry?

A. The ability to share modules publicly with any user of Terraform
B. The ability to restrict modules to members of Terraform Cloud or Enterprise organizations
C. The ability to tag modules by version or release
D. The ability to share modules with public Terraform users and members of Terraform Cloud Organizations

Answer: B

Explanation:
The Terraform Cloud private registry provides the ability to restrict modules to members of Terraform Cloud or Enterprise organizations. This allows you to share modules within your organization without exposing them to the public. The private registry also supports importing modules from your private VCS repositories. The public Terraform Module Registry, on the other hand, publishes modules from public Git repositories and makes them available to any user of Terraform. References = : Private Registry - Terraform Cloud : Terraform Registry - Provider Documentation

NEW QUESTION 38
You want to define a single input variable to capture configuration values for a server. The values must represent memory as a number, and the server name as a string.
Which variable type could you use for this input?

A. List
B. Object
C. Map
D. Terraform does not support complex input variables of different types

Answer: B

Explanation:
This is the variable type that you could use for this input, as it can store multiple attributes of different types within a single value. The other options are either invalid or incorrect for this use case.

NEW QUESTION 43
How is terraform import run?

A. As a part of terraform init
B. As a part of terraform plan
C. As a part of terraform refresh
D. By an explicit call
E. All of the above

Answer: D

Explanation:
The terraform import command is not part of any other Terraform workflow. It must be explicitly invoked by the user with the appropriate arguments, such as the resource address and the ID of the existing infrastructure to import. References = [Importing Infrastructure]

NEW QUESTION 46
In a Terraform Cloud workpace linked to a version control repository speculative plan rum 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 47
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 51
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 fire
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
Which of these ate secure options for storing secrets for connecting to a Terraform remote backend? Choose two correct answers.

A. A variable file
B. Defined in Environment variables
C. Inside the backend block within the Terraform configuration
D. Defined in a connection configuration outside of Terraform

Answer: BD

Explanation:
Environment variables and connection configurations outside of Terraform are secure options for storing secrets for connecting to a Terraform remote backend. Environment variables can be used to set values for input variables that contain secrets, such as backend access keys or tokens. Terraform will read environment variables that start with TF_VAR_ and match the name of an input variable. For example, if you have an input variable called backend_token, you can set its value with the environment variable TF_VAR_backend_token1. Connection configurations outside of Terraform are files or scripts that provide credentials or other information for Terraform to connect to a remote backend. For example, you can use a credentials file for the S3 backend2, or a shell script for the HTTP backend3. These files or scripts are not part of the Terraform configuration and can be stored securely in a separate location. The other options are not secure for storing secrets. A variable file is a file that contains values for input variables. Variable files are usually stored in the same directory as the Terraform configuration or in a version control system. This exposes the secrets to anyone who can access the files or the repository. You should not store secrets in variable files1. Inside the backend block within the Terraform configuration is where you specify the type and settings of the remote backend. The backend block is part of the Terraform configuration and is usually stored in a version control system. This exposes the secrets to anyone who can access the configuration or the repository. You should not store secrets in the backend block4. References = [Terraform Input Variables]1, [Backend Type: s3]2, [Backend Type: http]3, [Backend Configuration]4

NEW QUESTION 58
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 60
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 61
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 62
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 66
What is the workflow for deploying new infrastructure with Terraform?

A. Write Terraform configuration, run terraform init to initialize the working directory or workspace, and run terraform apply
B. Write Terraform configuration, run terraform show to view proposed changes, and terraform apply to create new infrastructure
C. Write Terraform configuration, run terraform apply to create infrastructure, use terraform validate to confirm Terraform deployed resources correctly
D. Write Terraform configuration, run terraform plan to initialize the working directory or workspace, and terraform apply to create the infrastructure

Answer: A

Explanation:
This is the workflow for deploying new infrastructure with Terraform, as it will create a plan and apply it to the target environment. The other options are either incorrect or incomplete.

NEW QUESTION 69
Which of the following is not a valid source path for specifying a module?

A. source - "github.com/hashicorp/examplePref-ul.0.8M
B. source = "./module?version=vl.6.0"
C. source - "hashicorp/consul/aws"
D. source - "./module"

Answer: B

Explanation:
Terraform modules are referenced by specifying a source location. This location can be a URL or a file path. However, specifying query parameters such as ?version=vl.6.0 directly within the source path is not a valid or supported method for specifying a module version in Terraform. Instead, version constraints are specified using the version argument within the module block, not as part of the source string.
References
= This clarification is based on Terraform's official documentation regarding module usage, which outlines the correct methods for specifying module sources and versions.

NEW QUESTION 72
Which Terraform command checks that your configuration syntax is correct?

A. terraform validate
B. terraform init
C. terraform show
D. terraform fmt

Answer: A

Explanation:
The terraform validate command is used to check that your Terraform configuration files are syntactically valid and internally consistent. It is a useful command for ensuring your Terraform code is error-free before applying any changes to your infrastructure.

NEW QUESTION 73
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 78
Define the purpose of state in Terraform.

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

Answer: A

Explanation:
The purpose of state in Terraform is to keep track of the real-world resources managed by Terraform, mapping them to the configuration. The state file contains metadata about these resources, such as resource IDs and other important attributes, which Terraform uses to plan and manage infrastructure changes. The state enables Terraform to know what resources are managed by which configurations and helps in maintaining the desired state of the infrastructure.References = This

role of state in Terraform is outlined in Terraform's official documentation, emphasizing its function in mapping configuration to real-world resources and storing vital metadata .

NEW QUESTION 79
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 83
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 86
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 88
You can configure Terraform to log to a file using the TF_LOG environment variable.

A. True
B. False

Answer: A

Explanation:
You can configure Terraform to log to a file using the TF_LOG environment variable. This variable can be set to one of the log levels: TRACE, DEBUG, INFO, WARN or ERROR. You can also use the TF_LOG_PATH environment variable to specify a custom log file location. References = : Debugging Terraform

NEW QUESTION 91
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 94
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 97
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 98
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 102
You ate 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 103
You ate 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 108
A developer accidentally launched a VM (virtual machine) outside of the Terraform workflow and ended up with two servers with the same name. They don't know which VM Terraform manages but do have a list of all active VM IDs.
Which of the following methods could you use to discover which instance Terraform manages?

A. Run terraform state list to find the names of all VMs, then run terraform state show for each of them to find which VM ID Terraform manages
B. Update the code to include outputs for the ID of all VMs, then run terraform plan to view the outputs
C. Run terraform taint/code on all the VMs to recreate them
D. Use terraform refresh/code to find out which IDs are already part of state

Answer: A

Explanation:
The terraform state list command lists all resources that are managed by Terraform in the current state file1. The terraform state show command shows the attributes of a single resource in the state file2. By using these two commands, you can compare the VM IDs in your list with the ones in the state file and identify which one is managed by Terraform.

NEW QUESTION 113
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 116
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 120
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 apply1.
References = This information can be verified from the official Terraform documentation on the terraform init command provided by HashiCorp Developer1.

NEW QUESTION 123

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 124
Where in your Terraform configuration do you specify a state backend?

A. The resource block
B. The data source block
C. The terraform block
D. The provider block

Answer: C

Explanation:
In Terraform, the backend configuration, which includes details about where and how state is stored, is specified within the terraform block of your configuration. This block is the correct place to define the backend type and its configuration parameters, such as the location of the state file for a local backend or the bucket details for a remote backend like S3.References = This practice is outlined in Terraform's core documentation, which provides examples and guidelines on how to configure various aspects of Terraform's behavior, including state backends .

NEW QUESTION 125
Multiple team members are collaborating on infrastructure using Terraform and want to format the* Terraform code following standard Terraform-style convention. How should they ensure the code satisfies conventions?

A. Terraform automatically formats configuration on terraform apply
B. Run terraform validate prior to executing terraform plan or terraform apply
C. Use terraform fmt
D. Replace all tabs with spaces

Answer: C

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

NEW QUESTION 128
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 129
You have a list of numbers that represents the number of free CPU cores on each virtual cluster:

```
numcpus = [ 18, 3, 7, 11, 2 ]
```

What Terraform function could you use to select the largest number from the list?

A. top(numcpus)
B. max(numcpus)

C. ceil (numcpus)
D. hight[numcpus]

Answer: B

Explanation:
In Terraform, the max function can be used to select the largest number from a list of numbers. The max function takes multiple arguments and returns the highest one. For the list numcpus = [18, 3, 7, 11, 2], using max(numcpus...) will return 18, which is the largest number in the list.
References:
? Terraform documentation on max function: Terraform Functions - max

NEW QUESTION 132
You're writing a Terraform configuration that needs to read input from a local file called id_rsa.pub . Which built-in Terraform function can you use to import the file's contents as a string?

A. file("id_rsa.pub")
B. templaTefil("id_rsa.pub")
C. filebase64("id_rsa.pub")
D. fileset<"id_rsa.pub")

Answer: A

Explanation:
To import the contents of a local file as a string in Terraform, you can use the built-in file function. By specifying file("id_rsa.pub"), Terraform reads the contents of the id_rsa.pub file and uses it as a string within your Terraform configuration. This function is particularly useful for scenarios where you need to include file data directly into your configuration, such as including an SSH public key for provisioning cloud instances.References = This information is a standard part of Terraform's functionality with built-in functions, as outlined in Terraform's official documentation and commonly used in various Terraform configurations.

NEW QUESTION 133
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 137
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 141
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 143
When using multiple configuration of the same Terraform provider, what meta-argument must you include in any non-default provider configurations?

A. Alias
B. Id
C. Depends_on
D. name

Answer: A

Explanation:
This is the meta-argument that you must include in any non-default provider configurations, as it allows you to give a friendly name to the configuration and reference it in other parts of your code. The other options are either invalid or irrelevant for this purpose.

NEW QUESTION 148
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 153
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 155
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 158
Which of the following is not a benefit of adopting infrastructure as code?

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

Answer: B

Explanation:
Infrastructure as Code (IaC) provides several benefits, including the ability to version control infrastructure, reuse code, and automate infrastructure management.

However, IaC is typically associated with declarative configuration files and does not inherently provide a graphical user interface (GUI). A GUI is a feature that may be provided by specific tools or platforms built on top of IaC principles but is not a direct benefit of IaC itself1.
References = The benefits of IaC can be verified from the official HashiCorp documentation on ??What is Infrastructure as Code with Terraform??? provided by HashiCorp Developer1.

NEW QUESTION 161
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 166
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 169
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 173
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 175
What does terraform import do?

A. Imports existing resources into the state file
B. Imports all infrastructure from a given cloud provider
C. Imports a new Terraform module
D. Imports clean copies of tainted resources
E. None of the above

Answer: A

Explanation:
The terraform import command is used to import existing infrastructure into your Terraform state. This command takes the existing resource and associates it with a resource defined in your Terraform configuration, updating the state file accordingly. It does not generate configuration for the resource, only the state.

NEW QUESTION 180
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 185
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 186
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 187
Which command must you first run before performing further Terraform operations in a working directory?

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

Answer: D

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

NEW QUESTION 191
.....

Thank You for Trying Our Product

We offer two products:

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

2nd - Questons and Answers in PDF Format

Terraform-Associate-003 Practice Exam Features:

* Terraform-Associate-003 Questions and Answers Updated Frequently

* Terraform-Associate-003 Practice Questions Verified by Expert Senior Certified Staff

* Terraform-Associate-003 Most Realistic Questions that Guarantee you a Pass on Your FirstTry

* Terraform-Associate-003 Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year