# 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**
You've used Terraform to deploy a virtual machine and a database. You want to replace this virtual machine instance with an identical one without affecting the database. What is the best way to achieve this using Terraform?

A. Use the terraform state rm command to remove the VM from state file
B. Use the terraform taint command targeting the VMs then run terraform plan and terraform apply
C. Use the terraform apply command targeting the VM resources only
D. Delete the Terraform VM resources from your Terraform code then run terraform plan and terraform apply

**Answer:** B

**Explanation:**
The terraform taint command marks a resource as tainted, which means it will be destroyed and recreated on the next apply. This way, you can replace the VM instance without affecting the database or other resources. References = [Terraform Taint]

**NEW QUESTION 4**
A terraform apply can not infrastructure.

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

**Answer:** D

**Explanation:**

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

**NEW QUESTION 5**
You much initialize your working directory before running terraform validate.

A. True
B. False

**Answer:** A

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

**NEW QUESTION 6**
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 7**
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)


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

**Answer:** B

**NEW QUESTION 8**
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 9**
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 10**
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 10**
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 14**
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 16**
The Terraform binary version and provider versions must match each other in a single configuration.

A. True
B. False

**Answer:** B

**Explanation:**
The Terraform binary version and provider versions do not have to match each other in a single configuration. Terraform allows you to specify provider version constraints in the configuration??s terraform block, which can be different from the Terraform binary version1. Terraform will use the newest version of the provider that meets the configuration??s version constraints2. You can also use the dependency lock file to ensure Terraform is using the correct provider version3.
References =
•1: Providers - Configuration Language | Terraform | HashiCorp Developer
•2: Multiple provider versions with Terraform - Stack Overflow
•3: Lock and upgrade provider versions | Terraform - HashiCorp Developer

**NEW QUESTION 19**
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 20**
A module can always refer to all variables declared in its parent module.

A. True
B. False

**Answer:** B

**Explanation:**
A module cannot always refer to all variables declared in its parent module, as it needs to explicitly declare input variables and assign values to them from the parent module??s arguments. A module cannot access the parent module??s variables directly, unless they are passed as input arguments.

**NEW QUESTION 24**
Running terraform fmt without any flags in a directory with Terraform configuration files check the formatting of those files without changing their contents.

A. True
B. False

**Answer:** B

**Explanation:**
Running terraform fmt without any flags in a directory with Terraform configuration files will not check the formatting of those files without changing their contents, but will actually rewrite them to a canonical format and style. If you want to check the formatting without making changes, you need to use the -check flag.

**NEW QUESTION 25**
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 27**
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 30**
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 31**
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 36**
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 37**
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 40**
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 45**
You are writing a child Terraform module that provisions an AWS instance. You want to reference the IP address returned by the child module in the root configuration. You name the instance resource "main'.
Which of these is the correct way to define the output value?
A)

```
output "instance_ip_addr" {
  return aws_instance.main.private_ip
}
```

B)

```
output "aws_instance.instance_ip_addr" {
  return aws_instance.main.private_ip
}
```

C)

```
output "aws_instance.instance_ip_addr" {
  value = ${main.private_ip}
}
```

D)

```
output "instance_ip_addr" {
  value = aws_instance.main.private_ip
}
```

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

**Answer:** D

**NEW QUESTION 46**
Which of the following are advantages of using infrastructure as code (IaC) instead of provisioning with a graphical user interface (GUI)? Choose two correct answers.

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

**Answer:** BE

**Explanation:**
Infrastructure as code (IaC) is a way of managing and provisioning cloud infrastructure using programming techniques instead of manual processes1. IaC has many advantages over using a graphical user interface (GUI) for provisioning infrastructure, such as:
•Versioning: IaC allows you to store your infrastructure configuration in a version control system, such as Git, and track changes over time. This enables you to roll back to previous versions, compare differences, and collaborate with other developers2.
•Reusability: IaC allows you to create reusable modules and templates that can be applied to different environments, such as development, testing, and production. This reduces duplication, improves consistency, and speeds up deployment3.
•Sharing: IaC allows you to share your infrastructure configuration with other developers, teams, or organizations, and leverage existing code from open source repositories or registries. This fosters best practices, innovation, and standardization4.
•Risk reduction: IaC reduces the risk of human error, configuration drift, and security breaches that can occur when provisioning infrastructure manually or using a GUI. IaC also enables you to perform automated testing, validation, and compliance checks on your infrastructure before deploying it5. References =
•1: What is Infrastructure as Code? Explained for Beginners - freeCodeCamp.org
•2: The benefits of Infrastructure as Code - Microsoft Community Hub
•3: Infrastructure as Code : Best Practices, Benefits & Examples - Spacelift
•4: 5 Benefits of Infrastructure as Code (IaC) for Modern Businesses in the Cloud
•5: The 7 Biggest Benefits of Infrastructure as Code - DuploCloud

**NEW QUESTION 51**
Outside of the required_providers block, Terraform configurations always refer to providers by their local names.

A. True
B. False

**Answer:** B

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

**NEW QUESTION 52**
What kind of configuration block will create an infrastructure object with settings specified within the block?

A. provider
B. state
C. data
D. resource

**Answer:** D

**Explanation:**
This is the kind of configuration block that will create an infrastructure object with settings specified within the block. The other options are not used for creating infrastructure objects, but for configuring providers, accessing state data, or querying data sources.

**NEW QUESTION 55**
When do changes invoked by terraform apply take effect?

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

**Answer:** B

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

**NEW QUESTION 58**
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 62**
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 65**
The _____ determines how Terraform creates, updates, or delete resources.

A. Terraform configuration
B. Terraform provisioner
C. Terraform provider
D. Terraform core

**Answer:** C

**Explanation:**
This is what determines how Terraform creates, updates, or deletes resources, as it is responsible for understanding API interactions with some service and exposing resources and data sources based on that API.

**NEW QUESTION 69**
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 70**
Your security team scanned some Terraform workspaces and found secrets stored in plaintext in state files. How can you protect that data?

A. Edit your state file to scrub out the sensitive data
B. Always store your secrets in a secrets.tfvars file
C. Delete the state file every time you run Terraform
D. Store the state in an encrypted backend

**Answer:** D

**Explanation:**
This is a secure way to protect sensitive data in the state file, as it will be encrypted at rest and in transit2. The other options are not recommended, as they could lead to data loss, errors, or security breaches.

**NEW QUESTION 73**
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 76**
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 80**
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 82**
Which command lets you experiment with terraform expressions?

A. Terraform console
B. Terraform validate
C. Terraform env
D. Terraform test

**Answer:** A

**Explanation:**
This is the command that lets you experiment with Terraform expressions, by providing an interactive console that allows you to evaluate expressions and see their results. You can use this command to test your expressions before using them in your configuration files.

**NEW QUESTION 86**
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 88**
You are working on some new application features and you want to spin up a copy of your production deployment to perform some quick tests. In order to avoid having to configure a new state backend, what open source Terraform feature would allow you create multiple states but still be associated with your current code?

A. Terraform data sources
B. Terraform local values
C. Terraform modules
D. Terraform workspaces
E. None of the above

**Answer:** D

**Explanation:**
Terraform workspaces allow you to create multiple states but still be associated with your current code. Workspaces are like ??environments?? (e.g. staging, production) for the same configuration. You can use workspaces to spin up a copy of your production deployment for testing purposes without having to configure a new state backend. Terraform data sources, local values, and modules are not features that allow you to create multiple states. References = Workspaces and How to Use Terraform Workspaces

**NEW QUESTION 93**
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 96**
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 97**
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 98**
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**
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 107**
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 109**
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 111**
Terraform variable names are saved in the state file.

A. True
B. False

**Answer:** B

**Explanation:**
Terraform variable names are not saved in the state file, only their values are. The state file only stores the attributes of the resources and data sources that are managed by Terraform, not the variables that are used to configure them.

**NEW QUESTION 112**
When does Sentinel enforce policy logic during a Terraform Cloud run?

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

**Answer:** C

**Explanation:**
Sentinel policies are checked after the plan stage of a Terraform run, but before it can be confirmed or the terraform apply is executed3. This allows you to enforce rules on your infrastructure before it is created or modified.

**NEW QUESTION 115**
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 120**
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 124**
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 129**
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 134**
Infrastructure as Code (IaC) can be stored in a version control system along with application code.

A. True
B. False

**Answer:** A

**Explanation:**
Infrastructure as Code (IaC) can indeed be stored in a version control system along with application code. This practice is a fundamental principle of modern infrastructure management, allowing teams to apply software development practices like versioning, peer review, and CI/CD to infrastructure management. Storing IaC configurations in version control facilitates collaboration, history tracking, and change management.References = While this concept is a foundational aspect of IaC and is widely accepted in the industry, direct references from the HashiCorp Terraform Associate (003) study materials were not found in the provided files. However, this practice is encouraged in Terraform's best practices and various HashiCorp learning resources.

**NEW QUESTION 135**
Why does this backend configuration not follow best practices?

```
terraform {
  backend "s3" {
    bucket     = "terraform-state-prod"
    key        = "network/terraform.tfstate"
    region     = "us-east-1"
    access_key = "AKIAIOSFODNN7EXAMPLE"
    secret_key = "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
  }

  required_providers {
       aws = {
         source  = "hashicorp/aws"
         version = "~> 3.38"
       }
     }

     required_version = ">= 0.15"
}
```

A. An alias meta-argument should be included in backend blocks whenever possible
B. You should use the local enhanced storage backend whenever possible
C. You should not store credentials in Terraform configuration
D. The backend configuration should contain multiple credentials so that more than one user can execute terraform plan and terraform apply

**Answer:** C

**Explanation:**
This is a bad practice, as it exposes your credentials to anyone who can access your configuration files or state files. You should use environment variables, credential files, or other mechanisms to provide credentials to Terraform.

**NEW QUESTION 140**
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 145**
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 146**
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 151**
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 153**
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 157**
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 161**
Which of these are features of Terraform Cloud? Choose two correct answers.

A. A web-based user interface (UI)
B. Automated infrastructure deployment visualization
C. Automatic backups
D. Remote state storage

**Answer:** AD

**Explanation:**
Terraform Cloud includes several features designed to enhance collaboration and infrastructure management. Two of these features are:
? A web-based user interface (UI): This allows users to interact with Terraform Cloud
through a browser, providing a centralized interface for managing Terraform configurations, state files, and workspaces.
? Remote state storage: This feature enables users to store their Terraform state
files remotely in Terraform Cloud, ensuring that state is safely backed up and can be accessed by team members as needed.

**NEW QUESTION 164**
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 167**
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 171**
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 176**
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 177**
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 181**
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 185**
terraform plan updates your state file.

A. True
B. False

**Answer:** B

**Explanation:**
The terraform plan command does not update the state file. Instead, it reads the current state and the configuration files to determine what changes would be made to bring the real-world infrastructure into the desired state defined in the configuration. The plan operation is a read-only operation and does not modify the state or the infrastructure. It is the terraform apply command that actually applies changes and updates the state file.References = Terraform's official guidelines and documentation clarify the purpose of the terraform plan command, highlighting its role in preparing and showing an execution plan without making any changes to the actual state or infrastructure .

**NEW QUESTION 186**
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 191**
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 192**
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 193**
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 195**
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 196**
.....

# 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