Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)

https://www.2passeasy.com/dumps/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_Instance.example.ebs_block_device.[*].volume_id
C. aws_Instance.example.ebs_block_device.volume_ids
D. aws_instance.example-ebs_block_device.*.volume_id

Answer: D

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

NEW QUESTION 3
Which of the following is not a key principle of infrastructure as code?

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

Answer: D

Explanation:
 The key principle of infrastructure as code that is not listed among the options is golden images. Golden images are pre-configured, ready-to-use virtual machine images that contain a specific set of software and configuration. They are often used to create multiple identical instances of the same environment, such as for testing or production. However, golden images are not a principle of infrastructure as code, but rather a technique that can be used with or without infrastructure as code. The other options are all key principles of infrastructure as code, as explained below:
? Self-describing infrastructure: This means that the infrastructure is defined in code that describes its desired state, rather than in scripts that describe the steps to create it. This makes the infrastructure easier to understand, maintain, and reproduce.
? Idempotence: This means that applying the same infrastructure code multiple times will always result in the same state, regardless of the initial state. This makes the infrastructure consistent and predictable, and avoids errors or conflicts caused by repeated actions.
? Versioned infrastructure: This means that the infrastructure code is stored in a version control system, such as Git, that tracks the changes and history of the code. This makes the infrastructure code reusable, auditable, and collaborative, and enables practices such as branching, merging, and rollback. References = [Introduction to Infrastructure as Code with Terraform], [Infrastructure as Code in a Private or Public Cloud]

NEW QUESTION 4
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 5
HashiCorp Configuration Language (HCL) supports user-denned functions.

A. True
B. False

Answer: B

Explanation:
HashiCorp Configuration Language (HCL) does not support user-defined functions. You can only use the built-in functions that are provided by the language. The built-in functions allow you to perform various operations and transformations on values within expressions. The general syntax for function calls is a function name followed by comma-separated arguments in parentheses, such as max(5, 12, 9). You can find the documentation for all of the available built-in functions in the Terraform Registry or the
Packer Documentation, depending on which tool you are using. References = : Functions - Configuration Language | Terraform : Functions - Configuration Language | Packer

NEW QUESTION 6
Changing the Terraform backend from the default "local" backend to a different one after performing your first terrafom apply is:

A. Optional
B. Impossible
C. Mandatory
D. Discouraged

Answer: D

Explanation:
Changing the Terraform backend after performing the initial terraform apply is technically possible but strongly discouraged. This is because changing backends can lead to complexities in state management, requiring manual intervention such as state migration to ensure consistency. Terraform's documentation and best practices advise planning the backend configuration carefully before applying Terraform configurations to avoid such changes.References = This guidance is consistent with Terraform's official documentation, which recommends careful consideration and planning of backend configurations to avoid the need for changes.

NEW QUESTION 7
Which option cannot be used to keep secrets out of Terraform configuration files?

A. A Terraform provider
B. Environment variables
C. A -var flag
D. secure string

Answer: D

Explanation:
A secure string is not a valid option to keep secrets out of Terraform configuration files. A secure string is a feature of AWS Systems Manager Parameter Store that allows you to store sensitive data encrypted with a KMS key. However, Terraform does not support secure strings natively and requires a custom data source to retrieve them. The other options are valid ways to keep secrets out of Terraform configuration files. A Terraform provider can expose secrets as data sources that can be referenced in the configuration. Environment variables can be used to set values for input variables that contain secrets. A -var flag can be used to pass values for input variables that contain secrets from the command line or a file. References = [AWS Systems Manager Parameter Store], [Terraform AWS Provider Issue #55], [Terraform Providers], [Terraform Input Variables]

NEW QUESTION 8
How does Terraform determine dependencies between resources?

A. Terraform requires resource dependencies to be defined as modules and sourced in order
B. Terraform automatically builds a resource graph based on resources provisioners, special meta-parameters, and the stale file (if present}
C. Terraform requires resources in a configuration to be listed m the order they will be created to determine dependencies
D. Terraform requires all dependencies between resources to be specified using the depends_on parameter

Answer: B

Explanation:
This is how Terraform determines dependencies between resources, by using the references between them in the configuration files and other factors that affect the order of operations.

NEW QUESTION 9
Terraform configuration can only import modules from the public registry.

A. True
B. False

Answer: B

Explanation:
 Terraform configuration can import modules from various sources, not only from the public registry. Modules can be sourced from local file paths, Git repositories, HTTP URLs, Mercurial repositories, S3 buckets, and GCS buckets. Terraform supports a number of common conventions and syntaxes for specifying module sources, as documented in the [Module Sources] page. References = [Module Sources]

NEW QUESTION 10
Terraform providers are always installed from the Internet.

A. True
B. False

Answer: B

Explanation:
 Terraform providers are not always installed from the Internet. There are other ways to install provider plugins, such as from a local mirror or cache, from a local filesystem directory, or from a network filesystem. These methods can be useful for offline or air-gapped environments, or for customizing the installation process. You can configure the provider installation methods using the provider_installation block in the CLI configuration file.

NEW QUESTION 10
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 11
What does this code do?

```
terraform {
  required_providers {
    aws = "~> 3.0"
  }
}
```

A. Requires any version of the AWS provider > = 3.0 and <4.0
B. Requires any version of the AWS provider >= 3.0
C. Requires any version of the AWS provider > = 3.0 major releas
D. like 4.1
E. Requires any version of the AWS provider > 3.0

Answer: A

Explanation:
 This is what this code does, by using the pessimistic constraint operator (~>), which specifies an acceptable range of versions for a provider or module.

NEW QUESTION 16
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 19

While attempting to deploy resources into your cloud provider using Terraform, you begin to see some odd behavior and experience slow responses. In order to troubleshoot you decide to turn on Terraform debugging. Which environment variables must be configured to make Terraform's logging more verbose?

A. TF_LOG_PAIH
B. TF_LOG
C. TF_VAR_log_path
D. TF_VAR_log_level

Answer: B

Explanation:
To make Terraform's logging more verbose for troubleshooting purposes, you must configure the TF_LOG environment variable. This variable controls the level of logging and can be set to TRACE, DEBUG, INFO, WARN, or ERROR, with TRACE providing the most verbose output.References = Detailed debugging instructions and the use of environment variables like TF_LOG for increasing verbosity are part of Terraform's standard debugging practices

NEW QUESTION 24

A developer on your team is going lo leaf down an existing deployment managed by Terraform and deploy a new one. However, there is a server resource named aws instant.ubuntu[l] they would like to keep. What command should they use to tell Terraform to stop managing that specific resource?

A. Terraform plan rm:aws_instance.ubuntu[1]
B. Terraform state rm:aws_instance.ubuntu[1]
C. Terraform apply rm:aws_instance.ubuntu[1]
D. Terraform destory rm:aws_instance.ubuntu[1]

Answer: B

Explanation:
To tell Terraform to stop managing a specific resource without destroying it, you can use the terraform state rm command. This command will remove the resource from the Terraform state, which means that Terraform will no longer track or update the corresponding remote object. However, the object will still exist in the remote system and you can later use terraform import to start managing it again in a different configuration or workspace. The syntax for this command is terraform state rm <address>,
where <address> is the resource address that identifies the resource instance to remove.
For example, terraform state rm aws_instance.ubuntu[1] will remove the second instance of the aws_instance resource named ubuntu from the state. References = : Command: state rm : Moving Resources

NEW QUESTION 27

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 32

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 35

Which of these ate features of Terraform Cloud? Choose two correct answers.

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

Answer: CD

Explanation:
These are features of Terraform Cloud, which is a hosted service that provides a web-based UI, remote state storage, remote operations, collaboration features, and more for managing your Terraform infrastructure.

NEW QUESTION 37
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 40
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 45
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 49
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 50
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 54
What feature stops multiple users from operating on the Terraform state at the same time?

A. State locking
B. Version control
C. Provider constraints
D. Remote backends

Answer: A

Explanation:
 State locking prevents other users from modifying the state file while a Terraform operation is in progress. This prevents conflicts and data loss1.

NEW QUESTION 56
If a DevOps team adopts AWS CloudFormation as their standardized method for provisioning public cloud resoruces, which of the following scenarios poses a challenge for this team?

A. The team is asked to manage a new application stack built on AWS-native services
B. The organization decides to expand into Azure wishes to deploy new infrastructure
C. The team is asked to build a reusable code based that can deploy resources into any AWS region
D. The DevOps team is tasked with automating a manual, web console-based provisioning.

Answer: B

Explanation:
 This is the scenario that poses a challenge for this team, if they adopt AWS CloudFormation as their standardized method for provisioning public cloud resources, as CloudFormation only supports AWS services and resources, and cannot be used to provision infrastructure on other cloud platforms such as Azure.

NEW QUESTION 57
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 59
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 62
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 66
You can develop a custom provider to manage its resources using Terraform.

A. True
B. False

Answer: A

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

NEW QUESTION 69
You want to know from which paths Terraform is loading providers referenced in your Terraform configuration (* files). You need to enable additional logging messages to find this out. Which of the following would achieve this?

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

Answer: B

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

NEW QUESTION 73
You want to define multiple data disks as nested blocks inside the resource block for a virtual machine. What Terraform feature would help you define the blocks using the values in a variable?

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

Answer: D

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

NEW QUESTION 78
You add a new provider to your configuration and immediately run terraform apply in the CD using the local backend. Why does the apply fail?

A. The Terraform CD needs you to log into Terraform Cloud first
B. Terraform requires you to manually run terraform plan first
C. Terraform needs to install the necessary plugins first
D. Terraform needs you to format your code according to best practices first

Answer: C

Explanation:
 The reason why the apply fails after adding a new provider to the configuration and immediately running terraform apply in the CD using the local backend is because Terraform needs to install the necessary plugins first. Terraform providers are plugins that Terraform uses to interact with various cloud services and other APIs. Each provider has a source address that determines where to download it from. When Terraform encounters a new provider in the configuration, it needs to run terraform init first to install the provider plugins in a local directory. Without the plugins, Terraform cannot communicate with the provider and perform the desired actions. References = [Provider Requirements], [Provider Installation]

NEW QUESTION 81
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 86
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 88
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 92
You have deployed a new webapp with a public IP address on a cloud provider. However, you did not create any outputs for your code. What is the best method to quickly find the IP address of the resource you deployed?

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

Answer: B

Explanation:
This is a quick way to inspect the state file and find the information you need without modifying anything5. The other options are either incorrect or inefficient.

NEW QUESTION 93

Module variable assignments are inherited from the parent module and you do not need to explicitly set them.

A. True
B. False

Answer: B

Explanation:
Module variable assignments are not inherited from the parent module and you need to explicitly set them using the source argument. This allows you to customize the behavior of each module instance.

NEW QUESTION 98
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 101
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 103
Module version is required to reference a module on the Terraform Module Registry.

A. True
B. False

Answer: B

Explanation:
Module version is optional to reference a module on the Terraform Module Registry. If you omit the version constraint, Terraform will automatically use the latest available version of the module

NEW QUESTION 108
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 113
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 115
What does state looking accomplish?

A. Prevent accidental Prevent accident deletion of the state file
B. Blocks Terraform commands from modifying, the state file
C. Copies the state file from memory to disk
D. Encrypts any credentials stored within the state file

Answer: B

Explanation:
This is what state locking accomplishes, by preventing other users from modifying the state file while a Terraform operation is in progress. This prevents conflicts and data loss.

NEW QUESTION 119
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 120
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 124
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 127
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 129
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 131
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 135
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 140
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 144
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 149
How can you trigger a run in a Terraform Cloud workspace that is connected to a Version Control System (VCS) repository?

A. Only Terraform Cloud organization owners can set workspace variables on VCS connected workspaces
B. Commit a change to the VCS working directory and branch that the Terraform Cloud workspace is connected to
C. Only Terraform Cloud organization owners can approve plans in VCS connected workspaces
D. Only members of a VCS organization can open a pull request against repositories that are connected to Terraform Cloud workspaces

Answer: B

Explanation:
This will trigger a run in the Terraform Cloud workspace, which will perform a plan and apply operation on the infrastructure defined by the Terraform configuration files in the VCS repository.

NEW QUESTION 153
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 155
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 156
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 158
Which two steps are required to provision new infrastructure in the Terraform workflow? Choose two correct answers.

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

Answer: DE

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

NEW QUESTION 162
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 163
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 168
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 172
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 173
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 175
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 179
Terraform configuration (including any module references) can contain only one Terraform provider type.

A. True
B. False

Answer: B

Explanation:
Terraform configuration (including any module references) can contain more than one Terraform provider type. Terraform providers are plugins that Terraform uses to interact with various cloud services and other APIs. A Terraform configuration can use multiple providers to manage resources across different platforms and services. For example, a configuration can use the AWS provider to create a virtual machine, the Cloudflare provider to manage DNS records, and the GitHub provider to create a
repository. Terraform supports hundreds of providers for different use cases and scenarios. References = [Providers], [Provider Requirements], [Provider Configuration]

NEW QUESTION 180
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 185
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 187
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 188
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 192
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 196
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 200
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 204
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 205
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 210
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 215
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 216
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 219
You can reference a resource created with for_each using a Splat (*) expression.

A. True
B. False

Answer: B

Explanation:
You cannot reference a resource created with for_each using a splat (*) expression, as it will not work with resources that have non-numeric keys. You need to use a for expression instead to iterate over the resource instances.

NEW QUESTION 224
Which provider authentication method prevents credentials from being stored in the state file?

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

Answer: D

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

NEW QUESTION 227
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 229
A Terraform provider is NOT responsible for:

A. Exposing resources and data sources based on an APUI
B. Managing actions to take based on resources differences
C. Understanding API interactions with some service
D. Provisioning infrastructure in multiple

Answer: D

Explanation:
 This is not a responsibility of a Terraform provider, as it does not make sense grammatically or logically. A Terraform provider is responsible for exposing resources and data sources based on an API, managing actions to take based on resource differences, and understanding API interactions with some service.

NEW QUESTION 231
......

THANKS FOR TRYING THE DEMO OF OUR PRODUCT

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

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

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

Money Back Guarantee

Terraform-Associate-003 Practice Exam Features:

* Terraform-Associate-003 Questions and Answers Updated Frequently

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

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

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