

HashiCorp

Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)



NEW QUESTION 1

When does Terraform create the .terraform.lock.hcl 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

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 3

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

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

Answer: D

Explanation:

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

NEW QUESTION 4

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 in 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 5

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

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

Answer: D

Explanation:

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

NEW QUESTION 6

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 7

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 8

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_PAIRH
- 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 9

terraform validate confirms that your infrastructure matches the Terraform state file.

- A. True

B. False

Answer: B

Explanation:

terraform validate does not confirm that your infrastructure matches the Terraform state file. It only checks whether the configuration files in a directory are syntactically valid and internally consistent³. To confirm that your infrastructure matches the Terraform state file, you need to use terraform plan or terraform apply with the -refresh- only option.

NEW QUESTION 10

Which of these are 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 10

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 11

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 16

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

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

Answer: D

Explanation:

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

NEW QUESTION 20

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 23

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 27

You should run terraform fmt to rewrite all Terraform configurations within the current working directory to conform to Terraform-style conventions.

- A. True
- B. False

Answer: A

Explanation:

You should run terraform fmt to rewrite all Terraform configurations within the current working directory to conform to Terraform-style conventions. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability. It is recommended to use this command to ensure consistency of style across different Terraform codebases. The command is optional, opinionated, and has no customization options, but it can help you and your team understand the code more quickly and easily. References = : Command: fmt : Using Terraform fmt Command to Format Your Terraform Code

NEW QUESTION 29

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

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

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

Answer: A

Explanation:

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

NEW QUESTION 34

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 loss.

NEW QUESTION 36

Before you can use a remote backend, you must first execute terraform 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 37

What are some benefits of using Sentinel with Terraform Cloud/Terraform 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 40

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 45

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 50

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

- A. True
- B. False

Answer: B

Explanation:

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

NEW QUESTION 53

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 57

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

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

Answer: A

Explanation:

This is not a valid string function in Terraform. The other options are valid string functions that can manipulate strings in various ways2.

NEW QUESTION 60

What type of block is used to construct a collection of nested configuration blocks?

- A. Dynamic
- B. For_each
- C. Nesting
- D. repeated.

Answer: A

Explanation:

This is the type of block that is used to construct a collection of nested configuration blocks, by using a for_each argument to iterate over a collection value and generate a nested block for each element. For example, you can use a dynamic block to create multiple ingress rules for a security group resource.

NEW QUESTION 62

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 67

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 71

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 anything. The other options are either incorrect or inefficient.

NEW QUESTION 72

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 73

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

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

Answer: AD

Explanation:

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

NEW QUESTION 78

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 80

You are using a networking module in your Terraform configuration with the name label my-network. In your main configuration you have the following code:

```
output "net_id" {
  value = module.my_network.vnet_id
}
```

When you run terraform validate, you get the following error:

```
Error: Reference to undeclared output value

on main.tf line 12, in output "net_id":
12:   value = module.my_network.vnet_id
```

What must you do to successfully retrieve this value from your networking module?

- A. Change the reference value to my-network,outputs,vmet_id
- B. Define the attribute vmet_id as a variable in the networking module
- C. Define the attribute vnet_id as an output in the networking module
- D. Change the reference value module.my,network,outputs,vnet_id

Answer: C

Explanation:

This is what you must do to successfully retrieve this value from your networking module, as it will expose the attribute as an output value that can be referenced by other modules or resources. The error message indicates that the networking module does not have an output value named vnet_id, which causes the reference to fail.

NEW QUESTION 82

What does state locking accomplish?

- A. Prevent accidental 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 84

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 88

You have a Terraform configuration that defines a single virtual machine with no references to it, You have run terraform apply to create the resource, and then removed the resource definition from your Terraform configuration file.

What will happen you run terraform apply in the working directory again?

- A. Terraform will remove the virtual machine from the state file, but the resource will still exist
- B. Nothing
- C. Terraform will error
- D. Terraform will destroy the virtual machine

Answer: D

Explanation:

This is what will happen if you run terraform apply in the working directory again, after removing the resource definition from your Terraform configuration file. Terraform will detect that there is a resource in the state file that is not present in the configuration file, and will assume that you want to delete it.

NEW QUESTION 90

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 95

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 100

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 header. References = [Provider Configuration]

NEW QUESTION 101

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. 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 105

You have used Terraform to create an ephemeral development environment in the cloud and are now ready to destroy all the Infrastructure described by your Terraform configuration. To be safe, you would like to first see all the infrastructure that Terraform will delete.

Which command should you use to show all of the resources that will be deleted? Choose two correct answers.

- A. Run terraform state rm
- B. Run terraform show :destroy
- C. Run terraform destroy and it will first output all the resource that will be deleted before prompting for approval
- D. Run terraform plan .destroy

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 109

It is best practice to store secret data in the same version control repository as your Terraform configuration.

- A. True
- B. False

Answer: B

Explanation:

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

NEW QUESTION 113

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 115

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 119

You are 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 122

You are 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 127

Variables declared within a module are accessible outside of the module.

- A. True
- B. False

Answer: B

Explanation:

Variables declared within a module are only accessible within that module, unless they are explicitly exposed as output values.

NEW QUESTION 128

Which two steps are required to provision new infrastructure in the Terraform workflow? Choose two correct answers.

- A. Plan
- B. Import
- C. Validate
- 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 132

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 file. The terraform state show command shows the attributes of a single resource in the state file. 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 136

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

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

Answer: A

Explanation:

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

NEW QUESTION 140

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 143

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 145

Which backend does the Terraform CLI 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 150

Which type of block fetches or computes information for use elsewhere in a Terraform configuration?

- A. data
- B. local
- C. resource
- D. provider

Answer: A

Explanation:

In Terraform, a data block is used to fetch or compute information from external sources for use elsewhere in the Terraform configuration. Unlike resource blocks that manage infrastructure, data blocks gather information without directly managing any resources. This can include querying for data from cloud providers, external APIs, or other Terraform states. References = This definition and usage of data blocks are covered in Terraform's official documentation, highlighting their role in fetching external information to inform Terraform configurations.

NEW QUESTION 155

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 160

When should you run terraform init?

- A. Every time you run terraform apply
- B. Before you start coding a new Terraform project
- C. After you run terraform plan for the time in a new terraform project and before you run terraform apply
- D. After you start coding a new terraform project and before you run terraform plan for the first time.

Answer: D

Explanation:

You should run terraform init after you start coding a new Terraform project and before you run terraform plan for the first time. This command will initialize the working directory by downloading the required providers and modules, creating the initial state file, and performing other necessary tasks. References = : Initialize a Terraform Project

NEW QUESTION 163

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 167

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 170

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. templatefile("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 173

Which of the following arguments are required when declaring a Terraform output?

- A. value
- B. description
- C. default
- D. sensitive

Answer: A

Explanation:

When declaring a Terraform output, the value argument is required. Outputs are a way to extract information from Terraform-managed infrastructure, and the value argument specifies what data will be outputted. While other arguments like description and sensitive can provide additional context or security around the output, value is the only mandatory argument needed to define an output. References = The requirement of the value argument for outputs is specified in

Terraform's official documentation, which provides guidelines on defining and using outputs in Terraform configurations.

NEW QUESTION 175

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 180

Which of these commands makes your code more human readable?

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

Answer: D

Explanation:

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

NEW QUESTION 184

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

- A. `list`
- B. `array`
- C. `map`
- 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 values.

NEW QUESTION 186

When using multiple configuration of the same Terraform provider, what meta-argument must you include in any non-default provider configurations?

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

Answer: A

Explanation:

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

NEW QUESTION 189

As a member of an operations team that uses infrastructure as code (IaC) practices, you are tasked with making a change to an infrastructure stack running in a public cloud. Which pattern would follow IaC best practices for making a change?

- A. Make the change via the public cloud API endpoint
- B. Clone the repository containing your infrastructure code and then run the code
- C. Use the public cloud console to make the change after a database record has been approved
- D. Make the change programmatically via the public cloud CLI
- E. Submit a pull request and wait for an approved merge of the proposed changes

Answer: E

Explanation:

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

NEW QUESTION 191

What does the default "local" Terraform backend store?

- A. tfplan files
- B. State file
- C. Provider plugins
- D. Terraform binary

Answer: B

Explanation:

The default "local" Terraform backend stores the state file in a local file named terraform.tfstate, which can be used to track and manage the state of your infrastructure.

NEW QUESTION 196

You have never used Terraform before and would like to test it out using a shared team account for a cloud provider. The shared team account already contains 15 virtual machines (VM). You develop a Terraform configuration containing one VM. perform terraform apply, and see that your VM was created successfully. What should you do to delete the newly-created VM with Terraform?

- A. The Terraform state file contains all 16 VMs in the team account
- B. Execute terraform destroy and select the newly-created VM.
- C. Delete the Terraform state file and execute terraform apply.
- D. The Terraform state file only contains the one new VM
- E. Execute terraform destroy.
- F. Delete the VM using the cloud provider console and terraform apply to apply the changes to the Terraform state file.

Answer: C

Explanation:

This is the best way to delete the newly-created VM with Terraform, as it will only affect the resource that was created by your configuration and state file. The other options are either incorrect or inefficient.

NEW QUESTION 200

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.list : 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 204

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 205

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 208

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 209

When using a remote backend or terraform Cloud integration, where does Terraform save resource state?

- A. In an environment variable
- B. On the disk
- C. In the remote backend or Terraform Cloud
- D. In memory

Answer: C

Explanation:

This is where Terraform saves resource state when using a remote backend or Terraform Cloud integration, as it allows you to store and manage your state file in a remote location, such as a cloud storage service or Terraform Cloud's servers. This enables collaboration, security, and scalability for your Terraform infrastructure.

NEW QUESTION 213

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 214

You can access state stored with the local backend by using terraform_remote_state data source.

- A. True
- B. False

Answer: B

Explanation:

You cannot access state stored with the local backend by using the terraform_remote_state data source. The terraform_remote_state data source is used to retrieve the root module output values from some other Terraform configuration using the latest state snapshot from the remote backend. It requires a backend that supports remote state storage, such as S3, Consul, AzureRM, or GCS. The local backend stores the state file locally on the filesystem, which terraform_remote_state cannot access. References:

- ? Terraform documentation on terraform_remote_state data source: Terraform Remote State Data Source
- ? Example usage of remote state: Example Usage (remote Backend)

NEW QUESTION 216

What is one disadvantage of using dynamic blocks in Terraform?

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

Answer: D

Explanation:

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

NEW QUESTION 219

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

- A. Source = ??module/consul????
- B. Source = ?????github.com: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 220

How do you specify a module??s version when publishing it to the public terraform Module Registry?

- A. Configuration it in the module's Terraform code
- B. Mention it on the module's configuration page on the Terraform Module Registry
- C. The Terraform Module Registry does not support versioning modules
- D. Tag a release in the associated repo

Answer: D

Explanation:

This is how you specify a module??s version when publishing it to the public Terraform Module Registry, as it uses the tags from your version control system (such as GitHub or GitLab) to identify module versions. You need to use semantic versioning for your tags, such as v1.0.0.

NEW QUESTION 222

Which of these statements about Terraform Cloud workspaces is false?

- A. They have role-based access controls
- B. You must use the CLI to switch between workspaces
- C. Plans and applies can be triggered via version control system integrations
- D. They can securely store cloud credentials

Answer: B

Explanation:

The statement that you must use the CLI to switch between workspaces is false. Terraform Cloud workspaces are different from Terraform CLI workspaces. Terraform Cloud workspaces are required and represent all of the collections of infrastructure in an organization. They are also a major component of role-based access in Terraform Cloud. You can grant individual users and user groups permissions for one or more workspaces that dictate whether they can manage variables, perform runs, etc. You can create, view, and switch between Terraform Cloud workspaces using the Terraform Cloud UI, the Workspaces API, or the Terraform Enterprise Provider⁵. Terraform CLI workspaces are optional and allow you to create multiple distinct instances of a single configuration within one working directory. They are useful for creating disposable environments for testing or experimenting without affecting your main or production environment. You can create, view, and switch between Terraform CLI workspaces using the terraform workspace command⁶. The other statements about Terraform Cloud workspaces are true. They have role-based access controls that allow you to assign permissions to users and teams based on their roles and responsibilities. You can create and manage roles using the Teams API or the Terraform Enterprise Provider⁷. Plans and applies can be triggered via version control system integrations that allow you to link your Terraform Cloud workspaces to your VCS repositories. You can configure VCS settings, webhooks, and branch tracking to automate your Terraform Cloud workflow⁸. They can securely store cloud credentials as sensitive variables that are encrypted at rest and only decrypted when needed. You can manage variables using the Terraform Cloud UI, the Variables API, or the Terraform Enterprise Provider⁹. References = [Workspaces]⁵, [Terraform CLI Workspaces]⁶, [Teams and Organizations]⁷, [VCS Integration]⁸, [Variables]⁹

NEW QUESTION 223

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 227

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 230

.....

Thank You for Trying Our Product

We offer two products:

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

2nd - Questions 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

100% Actual & Verified — Instant Download, Please Click
[Order The Terraform-Associate-003 Practice Test Here](#)