



HashiCorp

Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)

About ExamBible

Your Partner of IT Exam

Found in 1998

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

Our Advances

* 99.9% Uptime

All examinations will be up to date.

* 24/7 Quality Support

We will provide service round the clock.

* 100% Pass Rate

Our guarantee that you will pass the exam.

* Unique Gurantee

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

NEW QUESTION 1

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

A terraform apply can not infrastructure.

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

Answer: D

Explanation:

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

NEW QUESTION 3

HashiCorp Configuration Language (HCL) supports user-defined 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 4

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 5

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 6

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 7

In Terraform HCL, an object type of object({name=string, age=number}) would match this value.

A)

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

B)

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

C)

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

D)

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

A. Option A

- B. Option B
- C. Option C
- D. Option D

Answer: B

NEW QUESTION 8

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 9

If a module declares a variable with a default, that variable must also be defined within the module.

- A. True
- B. False

Answer: B

Explanation:

A module can declare a variable with a default value without requiring the caller to define it. This allows the module to provide a sensible default behavior that can be customized by the caller if needed. References = [Module Variables]

NEW QUESTION 10

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

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

Answer: A

Explanation:

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

NEW QUESTION 10

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

- A. True
- B. False

Answer: B

Explanation:

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

References:

? Terraform documentation on backends: Terraform Backends

? Detailed backend support: Terraform Backend Types

NEW QUESTION 11

When you use a remote backend that needs authentication, HashiCorp recommends that you:

- A. Write the authentication credentials in the Terraform configuration files
- B. Keep the Terraform configuration files in a secret store
- C. Push your Terraform configuration to an encrypted git repository
- D. Use partial configuration to load the authentication credentials outside of the Terraform code

Answer: D

Explanation:

This is the recommended way to use a remote backend that needs authentication, as it allows you to provide the credentials via environment variables, command-

line arguments, or interactive prompts, without storing them in the Terraform configuration files.

NEW QUESTION 16

What are some benefits of using Sentinel with Terraform Cloud/Terraform Cloud? Choose three correct answers.

- A. You can restrict specific resource configurations, such as disallowing the use of CIDR=0.0.0.0/0.
- B. You can check out and check in cloud access keys
- C. Sentinel Policies can be written in HashiCorp Configuration Language (HCL)
- D. Policy-as-code can enforce security best practices
- E. You can enforce a list of approved AWS AMIs

Answer: ADE

Explanation:

Sentinel is a policy-as-code framework that allows you to define and enforce rules on your Terraform configurations, states, and plans¹. Some of the benefits of using Sentinel with Terraform Cloud/Terraform Enterprise are:

- You can restrict specific resource configurations, such as disallowing the use of CIDR=0.0.0.0/0, which would open up your network to the entire internet. This can help you prevent misconfigurations or security vulnerabilities in your infrastructure².
- Policy-as-code can enforce security best practices, such as requiring encryption, authentication, or compliance standards. This can help you protect your data and meet regulatory requirements³.
- You can enforce a list of approved AWS AMIs, which are pre-configured images that contain the operating system and software you need to run your applications. This can help you ensure consistency, reliability, and performance across your infrastructure⁴. References =
- 1: Terraform and Sentinel | Sentinel | HashiCorp Developer
- 2: Terraform Learning Resources: Getting Started with Sentinel in Terraform Cloud
- 3: Exploring the Power of HashiCorp Terraform, Sentinel, Terraform Cloud ??
- 4: Using New Sentinel Features in Terraform Cloud – Medium

NEW QUESTION 19

Which method for sharing Terraform configurations fulfills the following criteria:

- * 1. Keeps the configurations confidential within your organization
- * 2. Support Terraform??s semantic version constrains
- * 3. Provides a browsable directory

- A. Subfolder within a workspace
- B. Generic git repository
- C. Terraform Cloud private registry
- D. Public Terraform module registry

Answer: C

Explanation:

This is the method for sharing Terraform configurations that fulfills the following criteria:

- ? Keeps the configurations confidential within your organization
- ? Supports Terraform??s semantic version constraints
- ? Provides a browsable directory

The Terraform Cloud private registry is a feature of Terraform Cloud that allows you to host and manage your own modules within your organization, and use them in your Terraform configurations with versioning and access control.

NEW QUESTION 23

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 26

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 31

If a DevOps team adopts AWS CloudFormation as their standardized method for provisioning public cloud resources, 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 33

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

- A. True
- B. False

Answer: B

Explanation:

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

NEW QUESTION 35

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 38

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 42

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 46

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 50

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 ways.

NEW QUESTION 55

How could you reference an attribute from the vsphere_datacenter data source for use with the datacenter_id argument within the vsphere_folder resource in the following configuration?

```
data "vsphere_datacenter" "dc" {}

resource "vsphere_folder" "parent" {
  path = "Production"
  type = "vm"
  datacenter_id = _____
}
```

- A. Data.vsphere_datacenter.DC.id
- B. Vsphere_datacenter.dc.id
- C. Data,dc,id
- D. Data.vsphere_datacenter,dc

Answer: A

Explanation:

The correct way to reference an attribute from the vsphere_datacenter data source for use with the datacenter_id argument within the vsphere_folder resource in the following configuration is data.vsphere_datacenter.dc.id. This follows the syntax for accessing data source attributes, which is data.TYPE.NAME.ATTRIBUTE. In this case, the data source type is vsphere_datacenter, the data source name is dc, and the attribute we want to access is id. The other options are incorrect because they either use the wrong syntax, the wrong punctuation, or the wrong case. References = [Data Source: vsphere_datacenter], [Data Source: vsphere_folder], [Expressions: Data Source References]

NEW QUESTION 59

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 63

If you manually destroy infrastructure, what is the best practice reflecting this change in Terraform?

- A. Run terraform refresh
- B. It will happen automatically
- C. Manually update the state file

D. Run terraform import

Answer: B

Explanation:

If you manually destroy infrastructure, Terraform will automatically detect the change and update the state file during the next plan or apply. Terraform compares the current state of the infrastructure with the desired state in the configuration and generates a plan to reconcile the differences. If a resource is missing from the infrastructure but still exists in the state file, Terraform will attempt to recreate it. If a resource is present in the infrastructure but not in the state file, Terraform will ignore it unless you use the terraform import command to bring it under Terraform's management. References = [Terraform State]

NEW QUESTION 64

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 67

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 68

How does the Terraform cloud integration differ from other state backends such as S3, Consul,etc?

- A. It can execute Terraform runs on dedicated infrastructure in Terraform Cloud
- B. It doesn't show the output of a terraform apply locally
- C. It is only arable lo paying customers
- D. All of the above

Answer: A

Explanation:

This is how the Terraform Cloud integration differs from other state backends such as S3, Consul, etc., as it allows you to perform remote operations on Terraform Cloud's servers instead of your local machine. The other options are either incorrect or irrelevant.

NEW QUESTION 69

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 71

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 76

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 79

What does state locking 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 81

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 82

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 85

What is the workflow for deploying new infrastructure with Terraform?

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

Answer: A

Explanation:

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

NEW QUESTION 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 91

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 93

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

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

Answer: A

Explanation:

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

NEW QUESTION 98

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 99

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 102

You can configure Terraform to log to a file using the TF_LOG environment variable.

- A. True
- B. False

Answer: A

Explanation:

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

NEW QUESTION 104

Which of these is true about Terraform's plugin-based architecture?

- A. Terraform can only source providers from the internet
- B. Every provider in a configuration has its own state file for its resources
- C. You can create a provider for your API if none exists
- D. All providers are part of the Terraform core binary

Answer: C

Explanation:

Terraform is built on a plugin-based architecture, enabling developers to extend Terraform by writing new plugins or compiling modified versions of existing plugins¹. Terraform plugins are executable binaries written in Go that expose an implementation for a specific service, such as a cloud resource, SaaS platform, or API². If there is no existing provider for your API, you can create one using the Terraform Plugin SDK³ or the Terraform Plugin Framework⁴. References =

- 1: Plugin Development - How Terraform Works With Plugins | Terraform | HashiCorp Developer
- 2: Lab: Terraform Plug-in Based Architecture - GitHub
- 3: Terraform Plugin SDK - Terraform by HashiCorp
- 4: HashiCorp Terraform Plugin Framework Now Generally Available

NEW QUESTION 105

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

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

Answer: D

Explanation:

Terraform workspaces allow you to create multiple states but still be associated with your current code. Workspaces are like ??environments?? (e.g. staging, production) for the same configuration. You can use workspaces to spin up a copy of your production deployment for testing purposes without having to configure

a new state backend. Terraform data sources, local values, and modules are not features that allow you to create multiple states. References = Workspaces and How to Use Terraform Workspaces

NEW QUESTION 106

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 110

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 120

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 121

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 values¹.

NEW QUESTION 125

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 126

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 127

Where can Terraform not load a provider from?

- A. Plugins directory
- B. Provider plugin cache
- C. Official HashCrop Distribution on releases.hashcorp.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 131

Where does the Terraform local backend store its state?

- A. In the terraform file
- B. In the /tmp directory
- C. In the terraform,tfstate file
- D. In the user's terraform,state file

Answer: C

Explanation:

This is where the Terraform local backend stores its state, by default, unless you specify a different file name or location in your configuration. The local backend is the simplest backend type that stores the state file on your local disk.

NEW QUESTION 135

When does Sentinel enforce policy logic during a Terraform Cloud run?

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

Answer: C

Explanation:

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

NEW QUESTION 136

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 140

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 143

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 148

How would you reference the "name" value of the second instance of this resource?

```
resource "aws_instance" "web" {
  count = 2
  name = "terraform-${count.index}"
}
```

- A. aws_instance.web(2),name
- B. element(aws_instance.web, 2)
- C. aws_instance-web(1)
- D. aws_instance_web(1),name
- E. Aws_instance,web,* , name

Answer: D

Explanation:

In Terraform, when you use the count meta-argument, you can reference individual instances using an index. The indexing starts at 0, so to reference the "name" value of the second instance, you would use aws_instance.web[1].name. This syntax allows you to access the properties of specific instances in a list generated by the count argument.

References:

? Terraform documentation on count and accessing resource instances: Terraform Count

NEW QUESTION 150

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 152

What does Terraform use the .terraform.lock.hcl file for?

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

Answer: B

Explanation:

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

NEW QUESTION 157

Which parameters does terraform import require? Choose two correct answers.

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

Answer: BC

Explanation:

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

NEW QUESTION 162

You have created a main.tf Terraform configuration consisting of an application server, a database and a load balancer. You ran terraform apply and Terraform created all of the resources successfully.

Now you realize that you do not actually need the load balancer, so you run terraform destroy without any flags. What will happen?

- A. Terraform will prompt you to pick which resource you want to destroy
- B. Terraform will destroy the application server because it is listed first in the code
- C. Terraform will prompt you to confirm that you want to destroy all the infrastructure
- D. Terraform will destroy the main, tf file
- E. Terraform will immediately destroy all the infrastructure

Answer: C

Explanation:

This is what will happen if you run terraform destroy without any flags, as it will attempt to delete all the resources that are associated with your current working directory or workspace. You can use the -target flag to specify a particular resource that you want to destroy.

NEW QUESTION 166

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 itself.

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 168

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 170

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 175

Which command add existing resources into Terraform state?

- A. Terraform init
- B. Terraform plan
- C. Terraform refresh
- D. Terraform import
- E. All of these

Answer: D

Explanation:

This is the command that can add existing resources into Terraform state, by matching them with the corresponding configuration blocks in your files.

NEW QUESTION 176

A Terraform output that sets the "sensitive" argument to true will not store that value in the state file.

- A. True
- B. False

Answer: A

Explanation:

A Terraform output that sets the "sensitive" argument to true will store that value in the state file. The purpose of setting sensitive = true is to prevent the value from being displayed in the CLI output during terraform plan and terraform apply, and to mask it in the Terraform UI. However, it does not affect the storage of the value in the state file. Sensitive outputs are still written to the state file to ensure that Terraform can manage resources correctly during subsequent operations.

References:

? Terraform documentation on sensitive outputs: Terraform Output Values

NEW QUESTION 177

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 182

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 186

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 188

Select the command that doesn't cause Terraform to refresh its state.

- A. Terraform destroy
- B. Terraform apply
- C. Terraform plan
- D. Terraform state list

Answer: D

Explanation:

This is the command that does not cause Terraform to refresh its state, as it only lists the resources that are currently managed by Terraform in the state file. The other commands will refresh the state file before performing their operations, unless you use the `-refresh=false` flag.

NEW QUESTION 190

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

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

Answer: B

Explanation:

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

NEW QUESTION 191

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

- A. `Source = "module/consul"`
- B. `Source = "github.com:hasicrop/example.git"`
- C. `Source = "https://github.com/hasicrop/example.git"`
- D. `Source = "github.com/hasicrop/consul/aws"`

Answer: A

Explanation:

The module source path that does not specify a remote module is `source = "module/consul"`. This specifies a local module, which is a module that is stored in a subdirectory of the current working directory. The other options are all examples of remote modules, which are modules that are stored outside of the current working directory and can be accessed by various protocols, such as Git, HTTP, or the Terraform Registry. Remote modules are useful for sharing and reusing code across different configurations and environments. References = [Module Sources], [Local Paths], [Terraform Registry], [Generic Git Repository], [GitHub]

NEW QUESTION 193

FILL IN THE BLANK

What is the name of the default file where Terraform stores the state?

Type your answer in the field provided. The text field is not case-sensitive and all variations of the correct answer are accepted.

- A. Mastered
- B. Not Mastered

Answer: A

Explanation:

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

NEW QUESTION 198

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 200

A Terraform provider is NOT responsible for:

- A. Exposing resources and data sources based on an API
- 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 204

How would you output returned values from a child module in the Terraform CLI output?

- A. Declare the output in the root configuration
- B. Declare the output in the child module
- C. Declare the output in both the root and child module
- D. None of the above

Answer: C

Explanation:

To output returned values from a child module in the Terraform CLI output, you need to declare the output in both the child module and the root module. The child module output will return the value to the root module, and the root module output will display the value in the CLI. References = [Terraform Outputs]

NEW QUESTION 205

.....

Relate Links

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

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

Contact us

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

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