

Exam Questions Terraform-Associate-003

HashiCorp Certified: Terraform Associate (003)

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



NEW QUESTION 1

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 2

You must initialize your working directory before running terraform validate.

- A. True
- B. False

Answer: A

Explanation:

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

NEW QUESTION 3

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 4

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 5

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 6

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 7

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 8

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 9

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 10

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 10

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 11

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 16

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

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

Answer: B

Explanation:

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

NEW QUESTION 17

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 19

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 23

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 27

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 30

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 33

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

- A. True
- B. False

Answer: B

Explanation:

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

NEW QUESTION 37

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 41

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 46

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 51

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]1

NEW QUESTION 54

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 .destory

Answer: CD

Explanation:

To see all the resources that Terraform will delete, you can use either of these two commands:
? terraform destroy will show the plan of destruction and ask for your confirmation before proceeding. You can cancel the command if you do not want to destroy the resources.
? terraform plan -destroy will show the plan of destruction without asking for confirmation. You can use this command to review the changes before running terraform destroy. References = : Destroy Infrastructure : Plan Command: Options

NEW QUESTION 59

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 60

How can you trigger a run in a Terraform Cloud workspace that is connected to a Version Control System (VCS) repository?

- A. Only Terraform Cloud organization owners can set workspace variables on VCS connected workspaces

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

Answer: B

Explanation:

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

NEW QUESTION 65

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 68

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 69

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 72

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

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

Answer: DE

Explanation:

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

NEW QUESTION 73

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 75

Where can Terraform not load a provider from?

- A. Plugins directory
- B. Provider plugin chance
- C. Official HashCrop Distribution on releases.hashcrop.com
- D. Source code

Answer: D

Explanation:

This is where Terraform cannot load a provider from, as it requires a compiled binary file that implements the provider protocol. You can load a provider from a plugins directory, a provider plugin cache, or the official HashiCorp distribution on releases.hashicorp.com.

NEW QUESTION 76

Which task does terraform init not perform?

- A. Validates all required variables are present
- B. Sources any modules and copies the configuration locally
- C. Connects to the backend
- D. Sources all providers used in the configuration and downloads them

Answer: A

Explanation:

The terraform init command is used to initialize a working directory containing Terraform configuration files. This command performs several different initialization steps to prepare the current working directory for use with Terraform, which includes initializing the backend, installing provider plugins, and copying any modules referenced in the configuration. However, it does not validate whether all required variables are present; that is a task performed by terraform plan or terraform apply1.

References = This information can be verified from the official Terraform documentation on the terraform init command provided by HashiCorp Developer1.

NEW QUESTION 81

Which backend does the Terraform CU use by default?

- A. Depends on the cloud provider configured
- B. HTTP
- C. Remote
- D. Terraform Cloud
- E. Local

Answer: E

Explanation:

This is the backend that the Terraform CLI uses by default, unless you specify a different backend in your configuration. The local backend stores the state file in a local file named terraform.tfstate, which can be used to track and manage the state of your infrastructure.

NEW QUESTION 86

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 91

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 95

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



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

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

- A. top(numcpus)
- B. max(numcpus)
- C. ceil (numcpus)
- D. high[numcpus]

Answer: B

Explanation:

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

References:

? Terraform documentation on max function: Terraform Functions - max

NEW QUESTION 96

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 101

Your risk management organization requires that new AWS S3 buckets must be private and encrypted at rest. How can Terraform Cloud automatically and proactively enforce this security control?

- A. Auditing cloud storage buckets with a vulnerability scanning tool
- B. By adding variables to each Terraform Cloud workspace to ensure these settings are always enabled
- C. With an S3 module with proper settings for buckets
- D. With a Sentinel policy, which runs before every apply

Answer: D

Explanation:

The best way to automatically and proactively enforce the security control that new AWS S3 buckets must be private and encrypted at rest is with a Sentinel policy, which runs before every apply. Sentinel is a policy as code framework that allows you to define and enforce logic-based policies for your infrastructure. Terraform Cloud supports Sentinel policies for all paid tiers, and can run them before any terraform plan or terraform apply operation. You can write a Sentinel policy that checks the configuration of the S3 buckets and ensures that they have the proper settings for privacy and encryption, and then assign the policy to your Terraform Cloud organization or workspace. This way, Terraform Cloud will prevent any changes that violate the policy from being applied. References = [Sentinel Policy Framework], [Manage Policies in Terraform Cloud], [Write and Test Sentinel Policies for Terraform]

NEW QUESTION 105

If you update the version constraint in your Terraform configuration, Terraform will update your lock file the next time you run terraform Init.

- A. True
- B. False

Answer: A

Explanation:

If you update the version constraint in your Terraform configuration, Terraform will update your lock file the next time you run terraform init. This will ensure that you use the same provider versions across different machines and runs.

NEW QUESTION 109

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 111

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 113

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 116

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 118

All modules published on the official Terraform Module Registry have been verified by HashiCorp.

- A. True
- B. False

Answer: B

Explanation:

Not all modules published on the official Terraform Module Registry have been verified by HashiCorp. While HashiCorp verifies some modules, there are many community-contributed modules that are not verified. Verified modules have a "Verified" badge indicating that HashiCorp has reviewed them for security and best practices, but the registry also includes unverified modules.

References:

? Terraform Module Registry documentation: Terraform Registry

NEW QUESTION 120

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 124

One remote backend configuration always maps to a single remote workspace.

- A. True
- B. False

Answer: A

Explanation:

The remote backend can work with either a single remote Terraform Cloud workspace, or with multiple similarly-named remote workspaces (like networking-dev and networking-prod). The workspaces block of the backend configuration determines which mode it uses. To use a single remote Terraform Cloud workspace, set workspaces.name to the remote workspace's full name (like networking-prod). To use multiple remote workspaces, set workspaces.prefix to a prefix used in all of the desired remote workspace names. For example, set prefix = networking- to use Terraform cloud workspaces with names like networking-dev and networking-prod. This is helpful when mapping multiple Terraform CLI workspaces used in a single Terraform configuration to multiple Terraform Cloud workspaces. However, one remote backend configuration always maps to a single remote workspace, either by name or by prefix. You cannot use both name and prefix in the same backend configuration, or omit both. Doing so will result in a configuration error. References = [Backend Type: remote]

NEW QUESTION 126

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 129

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 134

What does terraform import do?

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

Answer: A

Explanation:

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

NEW QUESTION 137

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 140

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 143

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

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

Answer: D

Explanation:

None of the above methods prevent credentials from being stored in the state file. Terraform stores the provider configuration in the state file, which may include sensitive information such as credentials. This is a potential security risk and should be avoided if possible. To prevent credentials from being stored in the state file, you can use one of the following methods:

? Use environment variables to pass credentials to the provider. This way, the credentials are not part of the provider configuration and are not stored in the state file. However, this method may not work for some providers that require credentials to be set in the provider block.

? Use dynamic credentials to authenticate with your cloud provider. This way,

Terraform Cloud or Enterprise will request temporary credentials from your cloud provider for each run and use them to provision your resources. The credentials are not stored in the state file and are revoked after the run is completed. This method is supported for AWS, Google Cloud Platform, Azure, and

Vault. References = : [Sensitive Values in State] : Authenticate providers with dynamic credentials

NEW QUESTION 146

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

- A. Source = ??module/consul????
- B. Source = ???github.com:hasicrop/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 148

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 152

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 153

.....

THANKS FOR TRYING THE DEMO OF OUR PRODUCT

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

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

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

Money Back Guarantee

Terraform-Associate-003 Practice Exam Features:

- * Terraform-Associate-003 Questions and Answers Updated Frequently
- * Terraform-Associate-003 Practice Questions Verified by Expert Senior Certified Staff
- * Terraform-Associate-003 Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- * Terraform-Associate-003 Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year