HashiCorp.Terraform-Associate-004.v2026-11-09.q187

Exam Code:	Terraform-Associate-004
Exam Name:	HashiCorp Certified: Terraform Associate (004) (HCTA0-004)
Certification Provider:	HashiCorp
Free Question Number:	187
Version:	v2026-11-09
# of views:	102
# of Questions views:	1870
https://www.freecram.net/torrent/HashiCorp.Terraform-Associate-004.v2026-11-09.q187.html	

NEW QUESTION: 1
terraform plan updates your state file.
A. True
B. False
Answer: (SHOW ANSWER)
The terraform plan command does not update the state file. Instead, it reads the current state and the configuration files to determine what changes would be made to bring the real-world infrastructure into the desired state defined in the configuration. The plan operation is a read-only operation and does not modify the state or the infrastructure. It is the terraform apply command that actually applies changes and updates the state file.References = Terraform ' s official guidelines and documentation clarify the purpose of the terraform plan command, highlighting its role in preparing and showing an execution plan without making any changes to the actual state or infrastructure .

NEW QUESTION: 2
You are creating a reusable Terraform configuration and want to include an optional billing_dept tag so your Finance team can track team-specific spending on resources. Which of the following billing_dept variable declarations will achieve this?
A. variable " billing_dept " { type = default}
B. variable " billing_dept " { default =}
C. variable " billing_dept " { type = optional(string)}
D. variable " billing_dept " { optional = true}
Answer: (SHOW ANSWER)
Rationale for Correct Answer: To make a variable optional, you define a type that allows omission. Using optional(string) indicates the input may be unset (commonly used when the variable is part of an object type, or when you want to explicitly model optionality). This supports reusable configs where the tag is included only when provided.
Analysis of Incorrect Options (Distractors):

A: Invalid syntax-type = default is not a valid type declaration.
B: Invalid syntax-default must have a value.
D: Invalid-there is no optional = true argument for variable blocks.
Key Concept: Input variable type constraints and optional inputs.
Reference: Terraform Objectives - Read, Generate, and Modify Configurations (variables, types, and reusable configuration patterns).

NEW QUESTION: 3
Exhibit:
variable " sizes " {
type = list(string)
description = " Valid server sizes "
default = [" small " , " medium " , " large "]
}
A variable declaration is shown in the exhibit. Which is the correct way to get the value of medium from this variable?
A. var.sizes[0]
B. var.sizes[1]
C. var.sizes[2]
D. var.sizes[3]
Answer: (SHOW ANSWER)
Rationale for Correct answer: Terraform lists are zero-indexed. Given [" small " , " medium " , " large "], index 0 is small, index 1 is medium, and index 2 is large. Therefore, medium is var.sizes[1] .
Analysis of Incorrect Options (Distractors):
A: Index 0 returns small, not medium.
C: Index 2 returns large, not medium.
D: Index 3 is out of range for a 3-element list.
Key Concept: Accessing list elements with zero-based indexing in Terraform expressions.
Reference: Terraform Objectives - Read, Generate, and Modify Configurations (variables, types, and expressions).

NEW QUESTION: 4
In a HCP Terraform/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: (SHOW ANSWER)
Speculative Plans: Terraform Cloud'sspeculative planfeature runs automatically when changes are detected in a linked VCS repository, enabling users to review potential infrastructure changes without committing them.
Automatic Integration: This feature automates the planning process by triggering when changes are committed, aiding teams in previewing infrastructure changes seamlessly.

For further understanding, see theTerraform Cloud VCS Integrationdocumentation.

NEW QUESTION: 5
You want to bring an existing database under Terraform management. What information is required to create a new import block for the database?
Pick the two correct responses below.
A. The destination resource address of the block that will manage the database.
B. The path to the .tf file that contains the database resource block.
C. The ID associated with the current database on the cloud provider.
D. The database platform and version that the existing resource is running.
E. The connection string that Terraform will use to connect and manage the database.
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct Answer:Terraform import (including the newer import block workflow) needs:
A: the target resource address (e.g., aws_db_instance.main or module.db.aws_db_instance.main) where Terraform should map the existing remote object into state, and C: the provider-specific import ID that uniquely identifies the existing database in the provider (for AWS RDS, Azure, GCP, etc. each has a defined import ID format).These are the two required pieces to tell Terraform what to import and where to place it in state.
Analysis of Incorrect Options (Distractors):
B: Incorrect. Terraform does not require a file path; it loads configuration from the working directory. What matters is the resource address exists in the configuration, not where it is located.
D: Incorrect. Platform/version may be relevant to configuring the resource correctly afterward, but it is not required to create the import mapping.
E: Incorrect. Terraform providers typically authenticate using configured credentials, not a "connection string" for importing. Import IDs are used instead.
Key Concept:Importing existing resources: mapping remote object ID to a Terraform resource address in state.
Reference:Navigate Terraform State and Backends - importing resources into state (terraform import / import blocks), resource addressing, and provider import IDs.

NEW QUESTION: 6
Your team adopts AWS CloudFormation as the standardized method for provisioning public cloud resources. Which scenario presents a challenge for your team?
A. Building a reusable code base that can deploy resources into any AWS region.
B. Managing a new application stack built on AWS-native services.
C. Automating a manual, web console-based provisioning process.
D. Deploying new infrastructure into Microsoft Azure.
Answer: (SHOW ANSWER)
Rationale for Correct Answer: AWS CloudFormation is AWS-specific. If your standardized provisioning tool is CloudFormation, deploying infrastructure into Microsoft Azure becomes a challenge because CloudFormation does not natively provision Azure resources. This highlights a core IaC concept: tool /platform scope and portability across clouds.

Analysis of Incorrect Options (Distractors):
A (Reusable code base for any AWS region): CloudFormation templates are commonly reusable across AWS regions; region differences are typically handled via parameters, mappings, and region-specific resource availability-not a fundamental challenge.
B (Managing a new stack on AWS-native services): This is CloudFormation's strength: managing AWS- native services with declarative templates and stack updates.
C (Automating manual console provisioning): Another core use case for IaC-CloudFormation is designed to replace manual provisioning with repeatable automation.
Key Concept: IaC tool scope and portability-AWS CloudFormation is tightly coupled to AWS, limiting multi-cloud provisioning.
Reference: Terraform Objectives - Understand Infrastructure as Code (IaC) Concepts (automation, repeatability, portability considerations) and Understand Terraform's Purpose and Use Cases (multi-cloud provisioning as a common Terraform use case).

NEW QUESTION: 7
You can reference a resource created with for_each using a Splat (*) expression.
A. True
B. False
Answer: (SHOW ANSWER)
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: 8
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: (SHOW ANSWER)
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: 9
Which parameters does terraform import require? Choose two correct answers.
A. Provider
B. Resource ID
C. Resource address
D. Path

Answer: (SHOW ANSWER)

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: 10

Exhibit:
Error: Saved plan is stale
The given plan file can no longer be applied because the state was changed by another operation after the plan was created.
You have a saved execution plan containing desired changes for infrastructure managed by Terraform. After running terraform apply my.tfplan, you receive the error shown. How can you apply the desired changes?
(Pick the 2 correct responses below.)

Freecram.net

A. Generate a new execution plan file with terraform plan, and apply the new plan.

B. Run terraform apply without the saved execution plan.

C. Force the apply command by adding the flag -lock=false.

D. Refresh the current state data using the -refresh-only flag.

E. Update the current plan file using the terraform state push command.

Answer: (SHOW ANSWER)

Rationale for Correct Answer:

A: The plan is stale because state changed after it was created. The correct fix is to recreate the plan against the latest state and apply that new plan.

B: Running terraform apply without specifying the old plan causes Terraform to re-plan using the current state and configuration, then apply the resulting changes.

Analysis of Incorrect Options (Distractors):

C: -lock=false disables locking; it does not make an old plan valid and is unsafe in team environments.

D: -refresh-only changes what's recorded in state (when applied) but does not "unstale" an existing saved plan file.

E: terraform state push is for advanced/manual state management and does not update a saved plan file; using it here is inappropriate and risky.

Key Concept: Saved plans are tied to a specific state snapshot; if state changes, you must re-plan.

Reference: Terraform Objectives - Implement and Maintain State (state consistency and plan/apply workflow), Understand Terraform Basics and CLI (plan files).

NEW QUESTION: 11

Why is it important to treat your Terraform state file as sensitive?

A. It can contain information such as resource passwords and keys.

B. It stores all environment variables from the machine that created it.

C. It can be manually edited to change deployed resources.

D. It contains personal information about the last user to update it.

Answer: (SHOW ANSWER)

Detailed Explanation:

* Rationale for Correct Answer: Terraform state can contain sensitive values, including passwords, private keys, access tokens, generated secrets, database credentials, and provider-returned sensitive attributes. Even when values are marked sensitive in Terraform output, they may still exist in state.
* Analysis of Incorrect Options (Distractors):
* B. It stores all environment variables from the machine that created it. Incorrect. Terraform state does not store all local environment variables.
* C. It can be manually edited to change deployed resources. Incorrect. Manual state editing can affect Terraform's understanding of infrastructure, but it does not directly change real deployed resources.
* D. It contains personal information about the last user to update it. Incorrect. This is not the primary sensitivity concern for Terraform state.
* Key Concept: Terraform state may contain plaintext sensitive infrastructure data and must be protected.
Reference: Terraform Objective Domain: Implement and Maintain State

NEW QUESTION: 12
Which of the following is not a key principle of infrastructure as code?
A. Self-describing infrastructure
B. Idempotence
C. Versioned infrastructure
D. Golden images
Answer: (SHOW ANSWER)
The key principle of infrastructure as code that is not listed among the options is golden images. Golden images are pre-configured, ready-to-use virtual machine images that contain a specific set of software and configuration. They are often used to create multiple identical instances of the same environment, such as for testing or production. However, golden images are not a principle of infrastructure as code, but rather a technique that can be used with or without infrastructure as code. The other options are all key principles of infrastructure as code, as explained below:
Self-describing infrastructure: This means that the infrastructure is defined in code that describes its desired state, rather than in scripts that describe the steps to create it. This makes the infrastructure easier to understand, maintain, and reproduce.
Idempotence: This means that applying the same infrastructure code multiple times will always result in the same state, regardless of the initial state. This makes the infrastructure consistent and predictable, and avoids errors or conflicts caused by repeated actions.
Versioned infrastructure: This means that the infrastructure code is stored in a version control system, such as Git, that tracks the changes and history of the code. This makes the infrastructure code reusable, auditable, and collaborative, and enables practices such as branching, merging, and rollback. References = [Introduction to Infrastructure as Code with Terraform], [Infrastructure as Code in a Private or Public Cloud]

NEW QUESTION: 13
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: (SHOW ANSWER)
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: 14
By default, if you do not define a backend for your configuration, where does Terraform store information about the resources that it manages?
A. A subdirectory of your home directory named .terraform.d
B. A file in your configuration's directory named terraform.tfstate
C. A file in your configuration's directory named .terraform.lock.hcl
D. A subdirectory of your configuration named .terraform
Answer: (SHOW ANSWER)
Rationale for Correct Answer (B):
Terraform uses a state file to map real-world resources to your configuration. By default, if no backend is explicitly defined, Terraform stores this information in a file named terraform.tfstate located in the same directory as your configuration files. This local state file is critical for Terraform to understand what infrastructure already exists and to plan updates correctly.
Analysis of Incorrect Options (Distractors):
A). .terraform.d (home directory subdirectory):
This is used for storing plugins and some global settings, not for resource state.
C). .terraform.lock.hcl (lock file):
This file locks provider versions to ensure reproducible runs, but it does not contain resource state information.
D). .terraform (subdirectory):
This folder contains cached provider binaries and related data, but not the actual state file.
Key Concept:
Terraform state management is at the core of "Implement and Maintain State." Understanding where Terraform stores the state by default is critical because state files should often be stored remotely (using a backend like S3, GCS, or Terraform Cloud) for collaboration and reliability.
Reference:Terraform Exam Objective - Implement and Maintain State (HashiCorp Certified: Terraform Associate).

NEW QUESTION: 15
Your team adopts AWS CloudFormation as the standardized method for provisioning public cloud resources. Which scenario presents a challenge for your team?
A. Building a reusable codebase that can deploy resources into any AWS region.
B. Managing a new application stack built on AWS-native services.
C. Deploying new infrastructure into Microsoft Azure.

D. Automating a manual, web console-based provisioning process.
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct answer: AWS CloudFormation is a service specifically designed to provision and manage infrastructure within AWS only. It does not support multi-cloud environments such as Microsoft Azure. Therefore, attempting to deploy infrastructure into Azure using CloudFormation presents a clear limitation and challenge. This aligns with Terraform's core value proposition: multi-cloud and provider- agnostic infrastructure management, which overcomes this limitation.
Analysis of Incorrect Options (Distractors):
A). Building a reusable codebase that can deploy resources into any AWS region. Incorrect because CloudFormation supports reusable templates and can deploy across AWS regions without issue.
B). Managing a new application stack built on AWS-native services. Incorrect because CloudFormation is purpose-built for AWS-native services and handles this scenario effectively.
D). Automating a manual, web console-based provisioning process. Incorrect because CloudFormation is specifically designed to automate infrastructure provisioning, replacing manual console operations.
Key Concept: Cloud-specific vs multi-cloud Infrastructure as Code tools - Terraform supports multiple providers, whereas CloudFormation is AWS-only.
Reference: Terraform Objective Domain: Understand Terraform ' s Purpose and Use Cases

NEW QUESTION: 16
You decide to move a Terraform state file to Amazon S3 from another location. You write the code below into a file called backend.tf.
Which command will migrate your current state file to the new S3 remote backend?
A. terraform state
B. terraform init
C. terraform push
D. terraform refresh
Answer: (SHOW ANSWER)
This command will initialize the new backend and prompt you to migrate the existing state file to the new location3. The other commands are not relevant for this task.

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here: https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)

NEW QUESTION: 17

Exhibit:
Root module configuration:
output " vnet_id " {
value = module.my_network.vnet_id
}
Error:
Error: Reference to undeclared output value
on main.tf line 12, in output " vnet_id " :
12: value = module.my_network.vnet_id
You are using a networking module in your Terraform configuration with the name my_network. Your root module includes the configuration shown. When you run terraform validate, you get the error shown. Which option would successfully retrieve this value from your networking module?
A. Change the referenced value to module.my_network.outputs.vnet_id.
B. Define the attribute vnet_id as a variable in the networking module.
C. Change the referenced value to my_network.outputs.vnet_id.
D. Define the attribute vnet_id as an output in the networking module.
Answer: (SHOW ANSWER)
Rationale for Correct Answer: A parent/root module can only read values from a child module if the child module exports them via an output block. The error indicates vnet_id is not an exported output from module.
my_network. Defining output " vnet_id " { value = ... } inside the networking (child) module makes module.
my_network.vnet_id valid.
Analysis of Incorrect Options (Distractors):
A: Incorrect-Terraform does not use module. < name > .outputs. < output > ; module outputs are referenced directly as module. < name > . < output > .
B: Incorrect-variables are inputs to a module, not values exported from it.
C: Incorrect-missing the module. prefix and still uses an invalid .outputs path.
Key Concept: Module outputs are the contract for exposing child module values to the root/parent.
Reference: Terraform Objectives - Interact with Terraform Modules (module inputs/outputs and composition).

NEW QUESTION: 18
Which of these commands makes your code more human readable?
A. Terraform validate
B. Terraform output
C. Terraform show
D. Terraform file
Answer: (SHOW ANSWER)
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: 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 (LEAVE A REPLY)
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 downloadit 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: 20
Which backend does the Terraform CLI use by default?
A. Depends on the cloud provider configured
B. HTTP
C. Remote
D. Terraform Cloud
E. Local
Answer: (SHOW ANSWER)
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: 21
You can configure multiple cloud blocks in your Terraform configuration to connect your workspace to both HCP Terraform and your Terraform Enterprise instance.
A. True
B. False
Answer: (SHOW ANSWER)
Rationale for Correct answer: A Terraform configuration supports a single cloud block (similar to a single backend configuration). The cloud block defines where Terraform runs (HCP Terraform / Terraform Enterprise) and how the workspace is associated. You cannot connect one configuration to two separate Cloud

/Enterprise instances simultaneously using multiple cloud blocks.
Analysis of Incorrect Options (Distractors):
A (True): Incorrect-Terraform does not allow multiple cloud blocks; the configuration can target only one Cloud/Enterprise context at a time.
Key Concept: cloud block constraints and workspace association with HCP Terraform/Terraform Enterprise.
Reference: Terraform Objectives - Manage Terraform Workspaces and Cloud (HCP Terraform/TFE configuration and workflows).

NEW QUESTION: 22
Which of the following commands would you use to access all of the attributes and details of a resource managed by Terraform?
A. terraform state list 'provider_type.name'
B. terraform state show 'provider_type.name'
C. terraform get 'provider_type.name'
D. terraform state list
Answer: (SHOW ANSWER)
The terraform state show command allows you to access all of the attributes and details of a resource managed by Terraform. You can use the resource address (e.g. provider_type.name) as an argument to show the information about a specific resource. The terraform state list command only shows the list of resources in the state, not their attributes. The terraform get command downloads and installs modules needed for the configuration. It does not show any information about resources. References = [Command: state show] and [Command: state list]

NEW QUESTION: 23
Which of the following ate advantages of using infrastructure as code (IaC) instead of provisioning with a graphical user interface (GUI)? Choose two correct answers.
A. Lets you version, reuse, and share infrastructure configuration
B. Provisions the same resources at a lower cost
C. Secures your credentials
D. Reduces risk of operator error
E. Prevents manual modifications to your resources
Answer: (SHOW ANSWER)
It lets you version, reuse, and share infrastructure configuration as code files, which can be stored in a source control system and integrated with your CI/CD pipeline.
It reduces risk of operator error by automating repetitive tasks and ensuring consistency across environments.
IaC does not necessarily provision resources at a lower cost, secure your credentials, or prevent manual modifications to your resources - these depend on other factors such as your cloud provider, your security practices, and your access policies.

NEW QUESTION: 24
Terraform configuration can only import modules from the public registry.

A. True
B. False
Answer: (SHOW ANSWER)
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: 25
Exhibit:
module " web_stack " {
source = " ./modules/web_stack "
}
Your configuration defines the module block shown in the exhibit. The web_stack module accepts an input variable named servers. Which of the following changes to the module block sets the servers variable to the value of 3?
A. module " web_stack " {source = " ./modules/web_stack " var.servers = 3}
B. module " web_stack " {source = " ./modules/web_stack " inputs = { servers = 3 }}
C. module " web_stack " {source = " ./modules/web_stack " servers = 3}
D. module " web_stack " {source = " ./modules/web_stack " inputs.servers = 3}
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct Answer:Module input variables are set by providing arguments in the module block whose names match the module's declared variables. Since the module expects servers, you pass it directly as servers = 3. This is the standard Terraform module input pattern.
Analysis of Incorrect Options (Distractors):
A: Incorrect. var.servers is how you reference an input variable in expressions, not how you assign a module input. Module inputs are set as arguments, not through the var. namespace.
B: Incorrect. Terraform module blocks do not use an inputs = { ... } argument (that pattern is common in some other tools, but not Terraform).
D: Incorrect. inputs.servers is not valid HCL syntax for setting module input arguments.
Key Concept:Passing input variables to modules by setting arguments on the module block.
Reference:Interact with Terraform Modules - module input variables and how they're assigned in a module block.

NEW QUESTION: 26
Where does HashiCorp recommend you store API tokens and other secrets within your team ' s Terraform workspaces?
Pick three correct responses below:
A. In a plaintext document on a shared drive.
B. In HashiCorp Vault.

C. In a terraform.tfvars file, checked into your version control system.
D. In an environment variable and referenced with TF_VAR_variablename.
E. In an HCP Terraform variable, with the sensitive option checked.
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct Answer: HashiCorp recommends storing secrets in secure, purpose-built systems or mechanisms that avoid exposing sensitive values in plaintext or version control:
B). HashiCorp Vault is a dedicated secrets management solution designed to securely store and control access to sensitive data.
D). Environment variables (TF_VAR_) allow passing sensitive values at runtime without persisting them in configuration files or version control.
E). HCP Terraform sensitive variables are encrypted and hidden from logs and UI output, making them a secure option for managing secrets in Terraform Cloud/Enterprise environments.
These approaches align with Terraform best practices for secure secret management and protecting sensitive inputs in IaC workflows.
Analysis of Incorrect Options (Distractors):
A). Plaintext document on a shared drive is insecure and exposes secrets to unauthorized access.
C). terraform.tfvars committed to version control is explicitly discouraged because secrets would be stored in plaintext and tracked in Git history, creating a major security risk.
Key Concept: Secure handling of sensitive data in Terraform using Vault, environment variables, and sensitive input variables.
Reference: Terraform Objective Domain: Manage Terraform Workspaces and Cloud

NEW QUESTION: 27
The HCP Terraform private registry keeps the module configurations confidential within your organization.
A. True
B. False
Answer: (SHOW ANSWER)
Rationale for Correct Answer: A private registry in HCP Terraform is designed to publish and share modules within an organization (with org access controls), rather than publicly. That supports internal reuse while keeping module content and usage confined to authorized users in the org.
Analysis of Incorrect Options (Distractors):
B (False): Incorrect-confidential/internal distribution is a primary purpose of private registries.
Key Concept: Public vs private module registries and organizational access control.
Reference: Terraform Objectives - Interact with Terraform Modules (module sources/registries), Manage Terraform Workspaces and Cloud (HCP Terraform capabilities).

NEW QUESTION: 28
When using a remote backend or terraform Cloud integration, where does Terraform save resource sate?
A. In an environment variable
B. On the disk

C. In the remote backend or Terraform Cloud
D. In memory
Answer: C (LEAVE A REPLY)
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: 29
You 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 the resources that will be deleted? (Pick the 2 correct responses)
A. Runterraform destroy. This will output all the resources that will be deleted before prompting for approval.
B. Runterraform show -destroy.
C. Runterraform state rm *.
Answer: (SHOW ANSWER)
Runningterraform destroywill show all resources that will be deleted before prompting for approval. You can also runterraform plan -destroyto simulate the destruction without actually applying it, which is useful for reviewing the planned changes.
References:
Terraform Destroy

NEW QUESTION: 30
What type of block is used to construct a collection of nested configuration blocks?
A. Dynamic
B. For_each
C. Nesting
D. repeated.
Answer: (SHOW ANSWER)
This is the type of block that is used to construct a collection of nested configuration blocks, by using a for_each argument to iterate over a collection value and generate a nested block for each element. For example, you can use a dynamic block to create multiple ingress rules for a security group resource.

NEW QUESTION: 31
When should you write Terraform configuration files for existing infrastructure that you want to start managing with Terraform?
A. You can import infrastructure without corresponding Terraform code
B. Terraform will generate the corresponding configuration files for you
C. Before you run terraform Import
D. After you run terraform import
Answer: C (LEAVE A REPLY)

You need to write Terraform configuration files for the existing infrastructure that you want to import into Terraform, otherwise Terraform will not know how to manage it. The configuration files should match the type and name of the resources that you want to import.

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here: https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)

NEW QUESTION: 32
You want to create a string that combines a generated random_id and a variable and reuse that string several times in your configuration. What is the simplest correct way to implement this without repeating the random_id and variable?
A. Use a module.
B. Add an output value.
C. Add a local value.
D. Use a data source.
Answer: (SHOW ANSWER)
Rationale for Correct Answer: A local value (locals { ... }) is the simplest way to define a reusable expression once and reference it many times via local. < name > . This is exactly what locals are for: reducing repetition and improving readability when combining values like random_id results and variables.
Analysis of Incorrect Options (Distractors):
A (Module): Overkill-modules are for packaging reusable groups of resources, not simple string reuse inside the same configuration.
B (Output value): Outputs are for exposing values to the CLI or to parent modules, not for internal reuse within the same module.
D (Data source): Data sources read existing remote objects; they are not intended for composing local strings.
Key Concept: Using locals to avoid repeating expressions.
Reference: Terraform Objectives - Read, Generate, and Modify Configurations (locals, expressions, and reuse patterns).

NEW QUESTION: 33
Why would you use the -replace flag for terraform apply?
A. You want Terraform to ignore a resource on the next apply
B. You want Terraform to destroy all the infrastructure in your workspace
C. You want to force Terraform to destroy a resource on the next apply
D. You want to force Terraform to destroy and recreate a resource on the next apply

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

NEW QUESTION: 34
What does terraform destroy do?
A. Destroys all infrastructure in the Terraform state file.
B. Destroys all Terraform code files in the current directory, leaving the state file intact.
C. Destroys all infrastructure in the configured Terraform provider.
D. Destroys the Terraform state file, leaving the infrastructure intact.
Answer: A (LEAVE A REPLY)
* Rationale for Correct answer:terraform destroy removes all resources currently tracked in the Terraform state file by issuing destroy requests to the configured providers. It does not delete configuration files or providers themselves.
* Analysis of Incorrect Options (Distractors):
* B: Terraform never deletes configuration files.
* C: Terraform only destroys resources it manages, not all provider infrastructure.
* D: Terraform does not delete the state file by default.
* Key Concept:terraform destroy enforces removal of all managed infrastructure.
Reference:Terraform Exam Objective - Use Terraform to Manage Infrastructure

NEW QUESTION: 35
When declaring a variable, setting the sensitive argument to true will prevent the value from being stored in the state file.
A. True
B. False
Answer: (SHOW ANSWER)
Rationale for Correct Answer: Marking a variable (or output) as sensitive = true primarily prevents Terraform from displaying the value in CLI output and in some UI contexts. It does not prevent the value from being stored in the state file if that value is used by resources. Terraform state must often store attribute values to track and manage infrastructure; sensitive marking is about redaction, not state omission.
Analysis of Incorrect Options (Distractors):
A (True): Incorrect-sensitive values can still be written to state; you must protect state storage (encryption, access control) and avoid storing secrets in state where possible.
Key Concept: Sensitive values are redacted in output, but still can exist in state.
Reference: Terraform Objectives - Implement and Maintain State (state contents and sensitive data handling), Read, Generate, and Modify Configurations (sensitive variables/outputs).

NEW QUESTION: 36
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: (SHOW ANSWER)
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: 37
You have used Terraform lo create an ephemeral development environment in the (loud and are now ready to destroy all the Infrastructure described by your Terraform configuration To be safe, you would like to first see all the infrastructure that Terraform will delete.
Which command should you use to show all of the resources that mil be deleted? Choose two correct answers.
A. Run terraform state rm '
B. Run terraform show :destroy
C. Run terraform destroy and it will first output all the resource that will be deleted before prompting for approval
D. Run terraform plan .destory
Answer: (SHOW ANSWER)
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: 38
A module block used in your configuration is shown in the exhibit. You have been asked to update the version of this module from 4.2.1 to 5.0.0.
Exhibit:
module " compute " {
source = " Azure/compute/azurerm "
version = " 4.2.1 "
}
Which two steps must you take to accomplish this?
A. Run the terraform init command.
B. Run the terraform pull command.
C. Update the version argument in the module block.
D. Remove the version argument from the module block.
Answer: (SHOW ANSWER)
The correct answers are A and C.

To update a registry module from version 4.2.1 to 5.0.0, you must first update the module block's version argument so Terraform knows which module version is required. After changing the module version constraint, you must reinitialize the working directory so Terraform can install the selected module version into the local .terraform directory. In many real workflows, terraform init is sufficient after changing the version, while terraform init -upgrade is commonly used when refreshing already- installed modules to newer acceptable versions. terraform pull is not a Terraform command for updating modules. Removing the version argument is not a controlled upgrade because Terraform may then choose the newest available version, which can reduce reproducibility and increase upgrade risk.

NEW QUESTION: 39
You have to initialize a Terraform backend before it can be configured.
A. True
B. False
Answer: (SHOW ANSWER)
You can configure a backend in your Terraform code before initializing it. Initializing a backend will store the state file remotely and enable features like locking and workspaces. References = [Terraform Backends]

NEW QUESTION: 40
Which of the following can you do with terraform plan?
Pick the two correct responses below.
Freecram.net
A. Save a generated execution plan to apply later.
B. View the execution plan and check whether the changes match your expectations.
C. Schedule Terraform to run at a planned time in the future.
D. Execute a plan in a different workspace.
Answer: (SHOW ANSWER)
Detailed Explanation:
* Rationale for Correct Answer:
* A: terraform plan -out=planfile saves the proposed execution plan so you can later run terraform apply planfile. This supports review and controlled deployments.
* B: terraform plan shows what Terraform would change (create/update/destroy) so you can verify changes before applying-this is one of the primary purposes of plan.
* Analysis of Incorrect Options (Distractors):
* C: Incorrect. Scheduling runs is not a capability of terraform plan (that's handled by external schedulers/CI systems or HCP Terraform scheduling features-not the CLI plan command itself).
* D: Incorrect. Workspaces are selected via terraform workspace select < name > (or by using separate working directories / automation). terraform plan itself doesn't "execute in a different workspace" without switching context first.
* Key Concept: terraform plan for previewing changes and optionally saving an execution plan file for later application.
Reference: Understand Terraform Basics and CLI - terraform plan behavior, -out option, and plan/apply workflow.

NEW QUESTION: 41
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 (LEAVE A REPLY)
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: 42
What type of information can be found on the Terraform Registry when using published modules?
A. Required input variables.
B. Outputs.
C. Optional input variables and default values.
D. All of the above.
Answer: (SHOW ANSWER)
Rationale for Correct Answer (D):
The Terraform Registry provides documentation for published modules, including:
Required inputs (variables you must supply).
Optional inputs with defaults (so you know what you can override).
Outputs (what values the module returns for use elsewhere).
Therefore, the correct answer is all of the above.
Analysis of Incorrect Options:
A). Required inputs only: Incomplete.
B). Outputs only: Incomplete.
C). Optional inputs only: Incomplete.
D). All of the above: Correct because the Registry provides all relevant documentation.
Key Concept:
Terraform Registry modules include inputs, outputs, and usage details, enabling reusability and modular design.
Reference:Terraform Exam Objective - Interact with Terraform Modules.

NEW QUESTION: 43
You have just upgraded the version constraint of a provider in an existing Terraform configuration. What command will install the new provider version?
A. Run terraform init -upgrade.
B. Run terraform apply.
C. Run terraform plan.

D. Run terraform refresh.

Answer: (SHOW ANSWER)

Detailed Explanation:

Rationale for Correct Answer: terraform init -upgrade tells Terraform to disregard the previously selected versions in the dependency lock file when appropriate and consider newer provider versions that satisfy the configured version constraints.

Analysis of Incorrect Options (Distractors):

B: terraform apply executes actions from an execution plan; it is not the command for deliberately upgrading provider dependencies.

C: terraform plan previews infrastructure changes but does not perform a provider upgrade.

D: terraform refresh concerns state synchronization and does not install a newly selected provider version.

Key Concept: Provider dependency installation, version constraints, and the dependency lock file.

Reference: Terraform Objective - Manage Terraform Resources and Providers.

NEW QUESTION: 44

You want to use API tokens and other secrets within your team ' s Terraform workspaces. Where does HashiCorp recommend you store these sensitive values?

(Pick 3 correct responses)

A. In a plaintext document on a shared drive.

B. In a terraform.tfvars file, checked into version control.

C. In a terraform.tfvars file, securely managed and shared with your team.

D. In an HCP Terraform/Terraform Cloud variable, with the sensitive option checked.

E. In HashiCorp Vault.

Answer: (SHOW ANSWER)

Rationale for Correct Answer:

C: Securely managed .tfvars files are acceptable if not committed to version control.

D: HCP Terraform provides secure storage for sensitive variables.

E: HashiCorp Vault is designed for secrets management and integrates with Terraform.

Analysis of Incorrect Options (Distractors):

A: Plaintext storage is insecure.

B: Secrets should never be committed to version control.

Key Concept:Sensitive values must be stored using secure secret management solutions.

Reference:Terraform Exam Objective - Use Terraform to Manage Infrastructure

NEW QUESTION: 45

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

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

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

A. Change the reference value to my-network,outputs,vmet_id

B. Define the attribute vmet_id as a variable in the networking modeule

C. Define the attribute vnet_id as an output in the networking module
D. Change the reference value module.my,network,outputs,vnet_id
Answer: (SHOW ANSWER)
This is what you must do to successfully retrieve this value from your networking module, as it will expose the attribute as an output value that can be referenced by other modules or resources. The error message indicates that the networking module does not have an output value named vnet_id, which causes the reference to fail.

NEW QUESTION: 46
Your Terraform configuration declares a variable. You want to enforce that its value meets your specific requirements, and you want to block the Terraform operation if it does not. What should you add to your configuration?
A. Add a top-level check block.
B. Add a validation block to the variable block.
C. Add a top-level validation block.
D. Add a check block to the variable block.
Answer: (SHOW ANSWER)
Rationale for Correct Answer: To enforce constraints on an input variable, Terraform provides variable validation using a validation block inside the variable block. If the condition fails, Terraform produces an error and prevents the operation (plan/apply) from continuing with invalid input.
Analysis of Incorrect Options (Distractors):
A (Top-level check block): check blocks validate conditions, but they are not the standard mechanism for constraining input variable values at the point of declaration; variable validation is.
C (Top-level validation block): Terraform does not support a standalone top-level validation block.
D (check block in the variable block): check is not placed inside variable blocks for input validation; the correct construct is validation.
Key Concept: Input variable validation to enforce constraints and fail fast.
Reference: Terraform Objectives - Read, Generate, and Modify Configurations (input variables and validation).

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here: https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)

NEW QUESTION: 47
terraform validate confirms thesyntaxof Terraform files.
A. True
B. False

Answer: (SHOW ANSWER)
terraform validatechecks for syntax errors and internal consistencyin the configuration files.
However, itdoes notcheck if resource configurations are valid with the provider.
Official Terraform Documentation Reference:
terraform validate - HashiCorp Documentation

NEW QUESTION: 48
Variables declared within a module are accessible outside of the module.
A. True
B. False
Answer: (SHOW ANSWER)
Variables declared within a module are only accessible within that module, unless they are explicitly exposed as output values1.

NEW QUESTION: 49
You ' re writing a Terraform configuration that needs to read input from a local file called id_rsa.pub. Which built-in Terraform function can you use to import the file ' s contents as a string?
A. fileset(" id_rsa.pub ")
B. file(" id_rsa.pub ")
C. filebase64(" id_rsa.pub ")
D. templatefile(" id_rsa.pub ")
Answer: (SHOW ANSWER)
Rationale for Correct Answer (B):
The file() function reads a file from disk and returns its content as a string. This is commonly used for public keys (.pub files).
Analysis of Incorrect Options:
A). fileset(): Returns a set of filenames matching a pattern, not file contents.
C). filebase64(): Reads file and returns Base64 encoded string - unnecessary for .pub.
D). templatefile(): Used for rendering templates with variables, not raw file contents.
Key Concept:
Terraform's file() function is used for injecting file content directly.
Reference:Terraform Exam Objective - Read, Generate, and Modify Configurations.

NEW QUESTION: 50
After creating a new Terraform configuration, your configuration passes terraform validate but returns an
"Access Denied" error from the cloud provider when running terraform plan.
Why did terraform validate not catch this issue?
Freecram.net
A. Variables are only applied and validated during terraform plan, so terraform validate assumed defaults and returned a success message.

B. The working directory was not initialized, so the cloud provider plugin was unavailable when running the terraform validate command.
C. terraform validate only checks whether a configuration is syntactically correct and internally consistent, and it does not communicate with providers.
D. The remote backend was not configured, so terraform validate could not load the state and detect the missing credentials.
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct Answer: terraform validate checks Terraform configuration syntax and internal consistency. It does not authenticate to cloud providers, query provider APIs, verify permissions, or detect access-related provider errors. Provider access checks occur during commands such as terraform plan or terraform apply.
Analysis of Incorrect Options (Distractors):
A). Variables are only applied and validated during terraform plan, so terraform validate assumed defaults and returned a success message. Incorrect. Variable validation can be checked by terraform validate, but the issue here is provider authorization, not variable validation.
B). The working directory was not initialized, so the cloud provider plugin was unavailable when running the terraform validate command. Incorrect. If initialization were required and missing, Terraform would generally report an initialization-related error, not silently validate provider credentials.
D). The remote backend was not configured, so terraform validate could not load the state and detect the missing credentials. Incorrect. terraform validate does not need state to validate basic configuration syntax and consistency.
Key Concept: terraform validate does not contact provider APIs or verify cloud permissions.
Reference: Terraform Objective Domain: Understand Terraform Basics and CLI

NEW QUESTION: 51
Which of the following is true about terraform apply?(Pick 2 correct responses)
A. You must pass the output of a terraform plan command to it.
B. By default, it does not refresh your state file to reflect the current infrastructure configuration.
C. Depending on provider specification, Terraform may need to destroy and recreate your infrastructure resources.
D. You cannot target specific resources for the operation.
E. It only operates on infrastructure defined in the current working directory or workspace.
Answer: (SHOW ANSWER)
C (#Correct)- If changes require aresource replacement(e.g., changing an immutable attribute like instance type), Terraform willdestroy and recreate the resource.
E (#Correct)- Terraform only appliesthe configuration in the current directory or workspace.

NEW QUESTION: 52
In a Terraform Cloud workpace linked to a version control repository speculative plan rum start automatically commit changes to version control.
A. True

B. False

Answer: (SHOW ANSWER)

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

NEW QUESTION: 53

Which of the following is not a valid source path for specifying a module?

A. source - " github.com/hashicorp/examplePref-ul.0.8M

B. source = " ./module?version=vl.6.0 "

C. source - " hashicorp/consul/aws "

D. source - " ./module "

Answer: (SHOW ANSWER)

Terraform modules are referenced by specifying a source location. This location can be a URL or a file path. However, specifying query parameters such as ?version=vl.6.0 directly within the source path is not a valid or supported method for specifying a module version in Terraform. Instead, version constraints are specified using the version argument within the module block, not as part of the source string.References = This clarification is based on Terraform ' s official documentation regarding module usage, which outlines the correct methods for specifying module sources and versions.

NEW QUESTION: 54

Which of these are features of HCP Terraform/Terraform Cloud? Pick the 2 correct responses below.

A. Automated infrastructure deployment visualization.

B. A web-based user interface (UI).

C. Automatic backups of configuration and state.

D. Remote state storage.

Answer: (SHOW ANSWER)

Automated Visualization: HCP Terraform providesvisualization toolsthat map infrastructure configurations, helping users manage complex architectures.

Remote State Storage: Terraform Cloud offersremote state management, essential for teams working collaboratively on shared infrastructure, ensuring consistency and avoiding state conflicts.

For more information, consultTerraform Cloud and HCP Terraform featuresin the official documentation.

NEW QUESTION: 55

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

A. Artifactory

B. Amazon S3

C. Terraform Cloud

D. Git

Answer: (SHOW ANSWER)

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

NEW QUESTION: 56
A child module can always access variables declared in its parent module.
A. True
B. False
Answer: B (LEAVE A REPLY)
Child modulesdo not automatically inheritvariables from the parent module.
To pass values from theparent moduleto thechild module, you mustexplicitly define input variablesin the child module and pass them in the parent module.
Example:
hcl
CopyEdit
module "example" {
source = "./child_module"
var1 = "value"
}
Official Terraform Documentation Reference:
Passing Variables to Modules

NEW QUESTION: 57
Which command(s) adds existing resources in a public cloud into Terraform state?
A. terraform init
B. terraform plan
C. terraform refresh
D. terraform import
E. All of these
Answer: D (LEAVE A REPLY)
Importing Existing Resources: Theterraform importcommand brings resources already deployed in a cloud environment into Terraform's state file, allowing Terraform to manage them.
Workflow Usage: Importing is vital when managing resources created outside of Terraform or those in place before Terraform adoption.
Refer to Terraform's import command documentation.

NEW QUESTION: 58
As a member of an operations team that uses infrastructure as code (lac) practices, you are tasked with making a change to an infrastructure stack running in a public cloud. Which pattern would follow laC 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: (SHOW ANSWER)

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: 59

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 (LEAVE A REPLY)

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: 60

Only the user that generated a plan may apply it.

A. True

B. False

Answer: (SHOW ANSWER)

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: 61

_____backends support state locking.

A. All

B. No

C. Some

D. Only local

Answer: (SHOW ANSWER)

Some backends support state locking, which prevents other users from modifying the state file while a Terraform operation is in progress. This prevents conflicts and data loss. Not all backends support this feature, and you can check the documentation for each backend type to see if it does.

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here: https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)

NEW QUESTION: 62
Terraform variables and outputs that set the description argument will store that description in the state file.
A. True
B. False
Answer: (SHOW ANSWER)
The description argument for Terraform variables and outputs is purely for documentation purposes and isnotstored in the Terraform state file.
Terraform variables and outputs can have descriptions for readability and clarity in .tf files.
However, these descriptions are not retained in the Terraform state file.
The state file only stores actual values and references needed for resource management, not metadata such as descriptions.
Official Terraform Documentation Reference:
Terraform Variables - HashiCorp Documentation
Terraform Outputs - HashiCorp Documentation

NEW QUESTION: 63
Which of the following should you put into the required_providers block?
A. version > = 3.1
B. version = " > = 3.1"
C. version ~ > 3.1
Answer: (SHOW ANSWER)
The required_providers block is used to specify the provider versions that the configuration can work with.
The version argument accepts a version constraint string, which must be enclosed in double quotes. The version constraint string can use operators such as > =, ~ > , =, etc. to specify the minimum, maximum, or exact version of the provider. For example, version = " > = 3.1 " means that theconfiguration can work with any provider version that is 3.1 or higher. References = [Provider Requirements] and [Version Constraints]

NEW QUESTION: 64
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: (SHOW ANSWER)
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: 65
A provider configuration block is required in every Terraform configuration.
Example:
A. True
B. False
Answer: (SHOW ANSWER)
A provider configuration block is not required in every Terraform configuration. A provider configuration block can be omitted if its contents would otherwise be empty. Terraform assumes an empty default configuration for any provider that is not explicitly configured. However, some providers may require some configuration arguments (such as endpoint URLs or cloud regions) before they can be used. A provider's documentation should list which configuration arguments it expects. For providers distributed on the Terraform Registry, versioned documentation is available on each provider's page, via the "Documentation" link in the provider's header1. References = [Provider Configuration]1

NEW QUESTION: 66

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

A. The Terraform state file contains all 16 VMs in the team account. Execute terraform destroy and select the newly-created VM.

B. Delete the Terraform state file and execute terraform apply.

C. The Terraform state file only contains the one new VM. Execute terraform destroy.

D. Delete the VM using the cloud provider console and terraform apply to apply the changes to the Terraform state file.

Answer: C (LEAVE A REPLY)

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

NEW QUESTION: 67

You can develop a custom provider to manage its resources using Terraform.

A. True

B. False

Answer: (SHOW ANSWER)

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: 68

Which of the following is availableonlyinHCP Terraform workspacesandnot in Terraform CLI?

A. Dry runs with terraform plan.

B. Secure variable storage.

C. Using one workspace ' s state as a data source for another.

D. Support for multiple cloud providers.

Answer: (SHOW ANSWER)

C (#Correct)- InHCP Terraform, you can use the terraform_remote_state data source to referenceanother workspace ' s state. This feature isnot available in Terraform CLI.

A (#Incorrect)- terraform plan (dry runs)is available in both Terraform CLI and HCP Terraform.

B (#Incorrect)- Secure variable storageexists in HCP Terraform, but CLI users can store variables securely using environment variables or .tfvars files.

D (#Incorrect)- BothTerraform CLI and HCP Terraform support multiple cloud providers.

Official Terraform Documentation Reference:

terraform_remote_state - HashiCorp Documentation

NEW QUESTION: 69

Exhibit



The Terraform configuration shown in the Exhibit space on this page v/ill create a new AWS instance.

A. True

B. False

Answer: (SHOW ANSWER)

Detailed Explanation:

Rationale for Correct Answer: The configuration uses a data block, not a resource block. In Terraform, a data block reads information about existing infrastructure from a provider. It does not create, update, or delete infrastructure. This data source searches for an existing AWS EC2 instance with the tag Name = web.

Analysis of Incorrect Options (Distractors):

A). True. Incorrect. A new AWS instance would be created using a resource block such as resource " aws_instance " " web " { ... }, not a data source.

Key Concept: Terraform data blocks are used to query existing infrastructure, while resource blocks manage infrastructure lifecycle.

Reference: Terraform Objective Domain: Read, Generate, and Modify Configurations

NEW QUESTION: 70

Which are examples of infrastructure as code? Choose two correct answers.

A. Cloned virtual machine images

B. Versioned configuration files

C. Change management database records

D. Doctor files

Answer: (SHOW ANSWER)

These are examples of infrastructure as code (IaC), which is a practice of managing and provisioning infrastructure through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

NEW QUESTION: 71

Which two steps are required to provision new infrastructure in the Terraform workflow? (Pick the 2 correct responses below.)

A. Import

B. Apply

C. Validate
D. Plan
E. Init
Answer: (SHOW ANSWER)
Rationale for Correct Answer: To provision new infrastructure, Terraform's core workflow is:
terraform plan to create an execution plan (what will be created/changed/destroyed).
terraform apply to execute that plan and provision the infrastructure.
These two steps represent the essential "preview then create" workflow Terraform objectives emphasize.
Analysis of Incorrect Options (Distractors):
A (Import): Used to bring existing resources under Terraform management; not required for provisioning new infrastructure.
C (Validate): Useful for checking syntax and internal consistency, but not required to provision.
E (Init): Typically required before first use in a new working directory (providers/modules/backend setup), but the question asks which steps are required in the workflow to provision-the required provisioning steps are plan + apply.
Key Concept: Terraform CLI workflow: plan # apply for provisioning.
Reference: Terraform Objectives - Understand Terraform Basics and CLI (core workflow commands).

NEW QUESTION: 72
Terraform variable names are saved in the state file.
A. True
B. False
Answer: (SHOW ANSWER)
Terraform variable names are not saved in the state file, only their values are. The state file only stores the attributes of the resources and data sources that are managed by Terraform, not the variables that are used to configure them.

NEW QUESTION: 73
A variable block is shown in the exhibit.
You will use this variable as the value of the tags argument in several resources. The data format must be a set of key-value pairs.
Exhibit
variable " tags " {
description = " Metadata tags for resources "
type = _____
}
Which type argument would you use?
A. type = object({ tags = string })
B. type = list(string)
C. type = map(string)
D. type = any

Answer: (SHOW ANSWER)

Detailed Explanation:

Rationale for Correct Answer: A Terraform map(string) represents a collection of key-value pairs where each key is a string and each associated value is also a string. This matches the common structure of resource tags. The supplied Terraform provider material also demonstrates key-value data with type = map(string).

Analysis of Incorrect Options (Distractors):

A: This represents an object containing a specific attribute named tags; it does not allow arbitrary tag keys.

B: A list contains ordered values rather than key-value pairs.

D: any would accept multiple possible types, but it sacrifices the useful type constraint supplied by map (string).

Key Concept: Terraform collection type constraints, particularly maps.

Reference: Terraform Objective - Read, Generate, and Modify Configurations.

NEW QUESTION: 74

You want to bring an existing database under Terraform management. What information is required to create a new import block for the database?

Pick the 2 correct responses below:

A. The destination resource address of the block that will manage the database.

B. The path to the .tf file that contains the database resource block.

C. The ID associated with the current database on the cloud provider.

D. The database platform and version that the existing resource is running.

E. The connection string that Terraform will use to connect and manage the database.

Answer: (SHOW ANSWER)

Detailed Explanation:

Rationale for Correct Answer:Terraform's import block (introduced in newer versions) requires two essential pieces of information:

A). Resource address - This specifies where in Terraform configuration the imported resource will be mapped (e.g., aws_db_instance.example).

C). Resource ID - This is the unique identifier used by the provider to locate the existing infrastructure (e.g., database ID in AWS, Azure, etc.).

These two elements allow Terraform to associate an existing real-world resource with a resource block in your configuration, aligning with the objective of managing existing infrastructure using Terraform.

Analysis of Incorrect Options (Distractors):

B). Path to the .tf file Incorrect because Terraform does not require file paths; it works with resource addresses defined in configuration.

D). Database platform and version Incorrect because this information may exist in configuration but is not required for the import operation itself.

E). Connection string Incorrect because Terraform uses provider APIs, not direct connection strings, to manage infrastructure.

Key Concept: Importing existing infrastructure requires a resource address and provider-specific resource ID.

Reference: Terraform Objective Domain: Manage Terraform Resources and Providers

NEW QUESTION: 75
Your root module contains a variable namednum_servers. Which is the correct way to pass its value to a child module with an input namedservers?
A. servers = num_servers
B. servers = var(num_servers)
C. servers = var.num_servers
D. servers = ${var.num_servers}
Answer: (SHOW ANSWER)
The correct syntax to pass a variable from the root module to a child module isservers = var.num_servers. Terraform uses dot notation to reference variables.
References:
Terraform Variables

NEW QUESTION: 76
Which of these ate secure options for storing secrets for connecting to a Terraform remote backend? Choose two correct answers.
A. A variable file
B. Defined in Environment variables
C. Inside the backend block within the Terraform configuration
D. Defined in a connection configuration outside of Terraform
Answer: (SHOW ANSWER)
Environment variables and connection configurations outside of Terraform are secure options for storing secrets for connecting to a Terraform remote backend. Environment variables can be used to set values for input variables that contain secrets, such as backend access keys or tokens. Terraform will read environment variables that start with TF_VAR_ and match the name of an input variable. For example, if you have an input variable called backend_token, you can set its value with the environment variable TF_VAR_backend_token1. Connection configurations outside of Terraform are files or scripts that provide credentials or other information for Terraform to connect to a remote backend. For example, you can use a credentials file for the S3 backend2, or a shell script for the HTTP backend3. These files or scripts are not part of the Terraform configuration and can be stored securely in a separate location. The other options are not secure for storing secrets. A variable file is a file that contains values for input variables. Variable files are usually stored in the same directory as the Terraform configuration or in a version control system. This exposes the secrets to anyone who can access the files or the repository. You should not store secrets in variable files1. Inside the backend block within the Terraform configuration is where you specify the type and settings of the remote backend. The backend block is part of the Terraform configuration and is usually stored in a version control system. This exposes the secrets to anyone who can access the configuration or the repository. You should not store secrets in the backend block4.
References = [Terraform Input Variables]1,
[Backend Type: s3]2, [Backend Type: http]3, [Backend Configuration]4

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here: https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)

NEW QUESTION: 77
How can you configure a Terraform workspace to store its state remotely?
A. Add a cloud block inside the terraform block.
B. Add a backend block inside the terraform block.
C. Set the TERAFORM_CLOUD environment variable.
D. Set the TERRAFORM_BACKEND environment variable.
Answer: A (LEAVE A REPLY)
Detailed Explanation:
Rationale for Correct Answer: To configure Terraform to use HCP Terraform for remote state storage and workspace-based runs, you add a cloud block inside the terraform block. The cloud block specifies the HCP Terraform organization and workspace configuration.
Analysis of Incorrect Options (Distractors):
B). Add a backend block inside the terraform block. This is valid for configuring other remote backends, such as S3, AzureRM, or GCS. However, for HCP Terraform workspaces, the modern and recommended configuration is the cloud block.
C). Set the TERAFORM_CLOUD environment variable. Incorrect. This environment variable is misspelled and is not how Terraform configures remote state.
D). Set the TERRAFORM_BACKEND environment variable. Incorrect. Terraform does not use this environment variable to configure a backend.
Key Concept: HCP Terraform remote state and workspace integration are configured using the cloud block.
Reference: Terraform Objective Domain: Manage Terraform Workspaces and Cloud

NEW QUESTION: 78
Which command must you run before you run a plan or apply for the first time?
A. terraform init
B. terraform workspace
C. terraform validate
D. terraform import
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct answer: terraform init initializes the working directory. It downloads required providers, installs modules, and configures the backend. You must run it before running terraform plan or terraform apply for the first time in a new configuration directory.

Analysis of Incorrect Options (Distractors):
B). terraform workspace. Incorrect. This command manages Terraform workspaces but does not initialize providers, modules, or backends.
C). terraform validate. Incorrect. Validation is useful, but it is not the required initialization step before planning or applying.
D). terraform import. Incorrect. Import brings existing infrastructure into Terraform state. It is not required before planning or applying a new configuration.
Key Concept: Terraform working directories must be initialized with terraform init.
Reference: Terraform Objective Domain: Understand Terraform Basics and CLI

NEW QUESTION: 79
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: (SHOW ANSWER)
If you update the version constraint in your Terraform configuration, Terraform will update your lock file the next time you run terraform init3. This will ensure that you use the same provider versions across different machines and runs.

NEW QUESTION: 80
A Terraform local value can reference other Terraform local values.
A. True
B. False
Answer: (SHOW ANSWER)
Rationale for Correct Answer: Locals are evaluated as expressions, and one local can reference another using local. < name > . This supports building up derived values cleanly:
locals {
env = " prod "
app_name = " api-${local.env} "
}
Analysis of Incorrect Options (Distractors):
B (False): Incorrect-Terraform explicitly supports local-to-local references.
Key Concept: Local values (locals) and expression references.
Reference: Terraform Objectives - Read, Generate, and Modify Configurations (locals, expressions, references).

NEW QUESTION: 81
Define the purpose of state in Terraform.
A. State maps real world resources to your configuration and keeps track of metadata
B. State lets you enforce resource configurations that relate to compliance policies
C. State stores variables and lets you quickly reuse existing code

D. State codifies the dependencies of related resources

Answer: (SHOW ANSWER)

The purpose of state in Terraform is to keep track of the real-world resources managed by Terraform, mapping them to the configuration. The state file contains metadata about these resources, such as resource IDs and other important attributes, which Terraform uses to plan and manage infrastructure changes. The state enables Terraform to know what resources are managed by which configurations and helps in maintaining the desired state of the infrastructure.References = This role of state in Terraform is outlined in Terraform's official documentation, emphasizing its function in mapping configuration to real-world resources and storing vital metadata .

NEW QUESTION: 82

You must use different Terraform commands depending on the cloud provider you use.

A. True

B. False

Answer: (SHOW ANSWER)

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.

From Terraform CLI Overview:

" Terraform provides a consistent CLI workflow regardless of which cloud or service you're managing. "

Thesame Terraform commandswork across all providers via plugins.

NEW QUESTION: 83

Which features do HCP Terraform workspaces provide that are not available in Terraform Community Edition? (Pick the 3 correct responses below.)

A. State versions and run history.

B. Automatic detection of common security issues.

C. Store Terraform and environment variables in variable sets.

D. Remote execution of Terraform operations.

E. Store your configuration in a Version Control System (VCS).

F. Support for multiple cloud providers.

Answer: (SHOW ANSWER)

Rationale for Correct Answer:

A (State versions and run history): HCP Terraform workspaces track state versions and maintain run history for auditability and rollback/reference-this is a hosted workflow feature beyond local-only Community Edition.

C (Variable sets): HCP Terraform provides variable sets to centrally manage and share Terraform /environment variables across multiple workspaces-Community Edition does not have this workspace-level managed feature.

D (Remote execution): HCP Terraform can execute plans/applies remotely in its managed runners, instead of requiring local execution.

Analysis of Incorrect Options (Distractors):

B: Not a standard "workspace feature" guaranteed as part of HCP Terraform workspaces in the same way as state/run history/remote runs/variables; security scanning depends on additional capabilities/policies /integrations rather than being the core baseline feature described here.
E: Storing configuration in VCS is not unique to HCP Terraform-you can store Terraform code in VCS with Community Edition as well. (HCP Terraform can integrate with VCS-driven workflows, but it doesn't uniquely "store" your configuration.) F: Terraform Community Edition also supports multiple cloud providers via providers; this is not exclusive to HCP Terraform.
Key Concept: Differences between local Terraform (Community Edition) and HCP Terraform workspace capabilities: remote runs, state history, centralized variables.
Reference: Terraform Objectives - Manage Terraform Workspaces and Cloud (HCP Terraform/Terraform Enterprise workspace features).

NEW QUESTION: 84
You have a saved execution plan containing desired changes for infrastructure managed by Terraform. After running the command terraform apply my.tfplan, you receive the error shown in the exhibit below.
Exhibit:
Error: Saved plan is stale
The given plan file can no longer be applied because the state was changed by another operation after the plan was created.
How can you apply the desired changes? (Choose TWO correct answers)
A. Update the current plan file using the terraform state push command.
B. Refresh the current state data using the -refresh-only flag.
C. Force the apply command by adding the flag -lock=false.
D. Generate a new execution plan file with terraform plan, and apply the new plan.
E. Run terraform apply without the saved execution plan.
Answer: D,E (LEAVE A REPLY)
Detailed Explanation:
Rationale for Correct Answer:The error indicates that the saved plan is stale, meaning the Terraform state has changed since the plan was created. Terraform ensures consistency between the plan and the current state, so it will not allow applying an outdated plan file.
D is correct because generating a new plan (terraform plan) ensures it reflects the current state, after which it can be safely applied.
E is also correct because running terraform apply without a saved plan automatically creates a fresh plan and applies it in one step, ensuring consistency with the latest state.
Analysis of Incorrect Options (Distractors):
A). terraform state push - Incorrect because this command is used to manually push state data, not to update or fix a stale execution plan.
B). -refresh-only flag - Incorrect because this updates the state to match real infrastructure but does not resolve the stale saved plan issue or make it applicable.
C). -lock=false - Incorrect because state locking prevents concurrent operations; disabling it does not fix the mismatch between the saved plan and the current state.

Key Concept:Terraform saved plans are state-dependent and become invalid if the state changes. You must regenerate the plan or apply without using a saved plan.
Reference:Terraform Objective Domain: Understand Terraform Basics and CLI

NEW QUESTION: 85
When should you use the terraform force-unlock command?
A. When terraform apply has failed due to a state lock.
B. When you have a high-priority change.
C. When automatic unlocking has failed.
D. When you see a status message stating that Terraform cannot acquire the lock.
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct answer: terraform force-unlock should be used only when Terraform's normal automatic unlocking has failed and you are sure no other Terraform process is actively using the state. It manually removes a stale lock.
Analysis of Incorrect Options (Distractors):
A). When terraform apply has failed due to a state lock. Incorrect. A lock error alone does not justify force unlocking. Another valid Terraform operation may still be running.
B). When you have a high-priority change. Incorrect. Urgency is not a valid reason to bypass state locking safeguards.
D). When you see a status message stating that Terraform cannot acquire the lock. Incorrect. This may simply mean another run is active. You should investigate before using force-unlock.
Key Concept: Force unlocking is a recovery action for stale locks, not a normal operational shortcut.
Reference: Terraform Objective Domain: Navigate Terraform State and Backends

NEW QUESTION: 86
You just upgraded the version of a provider in an existing Terraform project. What do you need to do to install the new provider?
A. Run terraform refresh.
B. Run terraform init -upgrade.
C. Run terraform apply -upgrade.
D. Upgrade your version of Terraform.
Answer: (SHOW ANSWER)
When upgrading a provider,you must run terraform init -upgrade to install the new version.
Thisdownloads the latest provider versionand updates the local dependency cache.
terraform refresh(A)only updates the state file but doesnotupgrade providers.
terraform apply -upgrade(C)isnot a valid command.
terraform version can be upgraded separately, but(D)does not install a new provider version.
Official Terraform Documentation Reference:
Upgrading Providers in Terraform

NEW QUESTION: 87
A resource block is shown in the Exhibit space of this page.



What is the Terraform resource name of that resource block?
A. azurerm
B. azurerm_resource_group
C. ev
D. test
Answer: (SHOW ANSWER)
Rationale for Correct Answer:A Terraform resource block has the following structure:
resource " < RESOURCE TYPE > " " < RESOURCE NAME > " {
}
In the exhibit:
Resource type: azurerm_resource_group
Resource name: dev
The resource name is the second label in the resource block and is used internally by Terraform to reference the resource (e.g., azurerm_resource_group.dev).
Analysis of Incorrect Options (Distractors):
A). azurerm: This is the provider name, not the resource name.
B). azurerm_resource_group: This is the resource type, not the name.
D). test: This is the value of the name argument inside the resource, not the Terraform resource name.
Key Concept:Understanding the distinction between resource type, resource name, and resource arguments in Terraform configurations.
Reference:Terraform Exam Objective - Read, Generate, and Modify Configurations

NEW QUESTION: 88
The public Terraform Module Registry is free to use.
A. True
B. False
Answer: (SHOW ANSWER)
The public Terraform Module Registry is free to use, as it is a public service that hosts thousands of self-contained packages called modules that are used to provision infrastructure. You can browse, use, and publish modules to the registry without any cost.

NEW QUESTION: 89
All modules published on the official Terraform Module Registry have been verified by HasihCorp.
A. True

B. False

Answer: (SHOW ANSWER)

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: 90

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: (SHOW ANSWER)

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

NEW QUESTION: 91

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

A. True

B. False

Answer: (SHOW ANSWER)

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)

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here:

https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)

NEW QUESTION: 92
terraform destroy is the only way to remove infrastructure with Terraform.
A. True
B. False
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct Answer: terraform destroy destroys all resources managed by the current configuration, but it is not the only way to remove infrastructure. You can also remove a resource block from the configuration and run terraform apply; Terraform will then plan to destroy that specific resource because it is no longer declared.
Analysis of Incorrect Options (Distractors):
A). True. Incorrect. Terraform can destroy infrastructure through normal apply operations when resources are removed from configuration.
Key Concept: Terraform reconciles real infrastructure with configuration. Removing a resource from configuration and applying the change can destroy that resource.
Reference: Terraform Objective Domain: Use Terraform to Manage Infrastructure

NEW QUESTION: 93
What is the provider for the resource shown in the Exhibit?
resource " aws_vpc " " main " {
name = " test "
}
A. VPC
B. test
C. main
D. aws
Answer: D (LEAVE A REPLY)
The provider for theaws_vpcresource isaws, as the resource type begins withaws_, which denotes that it is managed by the AWS provider.
References:
Terraform Providers

NEW QUESTION: 94
Which of these are benefits of using Sentinel with HCP Terraform/Terraform Cloud? (Pick the 3 correct responses)
A. You can enforce a list of approved AWS AMIs.
B. Sentinel Policies can be written in HashiCorp Configuration Language (HCL).
C. You can check out and check in cloud access keys.
D. Policy-as-code can enforce security best practices.

Answer: (SHOW ANSWER)
Sentinel is a policy-as-code framework that integrates with Terraform Cloud to enforce security, compliance, and governance rules. You can enforce rules such as approved AMIs and ensure security best practices.
Policies are written in the Sentinel language, not HCL.
References:
Sentinel Policies

NEW QUESTION: 95
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: (SHOW ANSWER)
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: 96
Which is a benefit of the Terraform state file?
A. A state file can schedule recurring infrastructure tasks.
B. A state file is the desired state expressed by the Terraform code files.
C. A state file is a source of truth for resources provisioned with Terraform.
D. A state file is a source of truth for resources provisioned with a public cloud console.
Answer: (SHOW ANSWER)

Rationale for Correct Answer: Terraform state is Terraform's record of what it manages: it maps resource addresses in configuration to real-world resource IDs and stores metadata needed to plan changes. That makes it a source of truth for resources Terraform is managing, enabling accurate diffs, updates, and deletes.
Analysis of Incorrect Options (Distractors):
A: Incorrect-Terraform state does not schedule jobs; that's the role of external schedulers/automation tools.
B: Incorrect-the desired state is expressed in .tf configuration, not in the state file. State reflects the current known state of managed infrastructure.
D: Incorrect-resources created manually in a cloud console are not automatically in Terraform state unless they are imported/adopted.
Key Concept: Purpose of Terraform state: resource mapping, metadata, and planning accuracy.
Reference: Terraform Objectives - Implement and Maintain State; Navigate Terraform State and Backends.

NEW QUESTION: 97
Which of the following module source paths does not specify a remote module?
A. Source = "module/consul"
B. Source = "githhub.comicrop/example"
C. Source ="git@github.com:hasicrop/example.git"
D. Source = "hasicrop/consul/aws"
Answer: (SHOW ANSWER)
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: 98
You have multiple team members collaborating on infrastructure as code (IaC) using Terraform, and want to apply formatting standards for readability.
How can you format Terraform HCL (HashiCorp Configuration Language) code according to standard Terraform style convention?
A. Run the terraform fmt command during the code linting phase of your CI/CD process Most Voted
B. Designate one person in each team to review and format everyone ' s code
C. Manually apply two spaces indentation and align equal sign " = " characters in every Terraform file (*.
tf)
D. Write a shell script to transform Terraform files using tools such as AWK, Python, and sed
Answer: (SHOW ANSWER)
The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability. Running this command on your configuration files before committing them to source control can help ensure consistency of style between different Terraform codebases, and can also make diffs

easier to read. You can also use the -check and -diff options to check if the files are formatted and display the formatting changes respectively2. Running the terraform fmt command during the code linting phase of your CI/CD process can help automate this process and enforce the formatting standards for your team. References = [Command: fmt]2

NEW QUESTION: 99
When you include a module block in your configuration that references a module from the Terraform Registry, the "version" attribute is required.
A. True
B. False
Answer: (SHOW ANSWER)
The "version" attribute is optional when referencing a module from the Terraform Registry. If not specified, the latest version will be used, but it is often recommended to specify a version to ensure consistency across environments.
References:
Terraform Module Versioning

NEW QUESTION: 100
Which statements are true? Choose two correct answers.
A. When a resource has a count meta-argument, Terraform evaluates the precondition for each instance.
B. A precondition cannot reference attributes of another resource that are only known after apply.
C. A precondition in a resource block must pass for Terraform to create the resource.
D. If a resource's postcondition block fails, Terraform still creates the resources that depend on it.
Answer: (SHOW ANSWER)
The correct answers are A and C.
Terraform custom conditions let you define validation rules through precondition and postcondition blocks. When a resource uses count or for_each, Terraform evaluates preconditions separately for each resource instance, which allows instance-specific checks using values such as count.index or each.key.
A precondition must pass before Terraform proceeds with the resource action, so it can block creation, update, or replacement when the configured condition is not satisfied. Option B is incorrect because Terraform can evaluate some conditions later if values are unknown during planning, depending on the expression and dependency graph. Option D is incorrect because a failed postcondition prevents downstream resources from relying on an invalid result, helping stop dependent changes from proceeding with bad assumptions.

NEW QUESTION: 101
Which command lets you experiment with terraform expressions?
A. Terraform console
B. Terraform validate
C. Terraform env
D. Terraform test
Answer: (SHOW ANSWER)

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

NEW QUESTION: 102
What is the Terraform style convention for indenting a nesting level compared to the one above it?
A. With two spaces.
B. With four spaces.
C. With three spaces.
D. With a tab.
Answer: (SHOW ANSWER)
Terraform's Indentation Standards: Terraform's style convention usestwo spaces per nesting levelfor readability, helping to maintain uniform code across teams.
Configuration Files: Consistent indentation is crucial for Terraform's HCL syntax, as it improves readability and avoids parsing issues.
More details are available in theTerraform configuration style guide.

NEW QUESTION: 103
Any user can publish modules to the public Terraform Module Registry.
A. True
B. False
Answer: (SHOW ANSWER)
The Terraform Registry allows any user to publish and share modules. Published modules support versioning, automatically generate documentation, allow browsing version histories, show examples and READMEs, and more. Public modules are managed via Git and GitHub, and publishing a module takes only a few minutes. Once a module is published, releasing a new version of a module is as simple as pushing a properly formed Git tag1.
References = The information can be verified from the Terraform Registry documentation on Publishing Modules provided by HashiCorp Developer1.

NEW QUESTION: 104
Exhibit:
resource " aws_instance " " example " {
ami = " ami-0a123456789abcdef "
instance_type = " t3.micro "
}
You are updating a child module with the resource block shown in the exhibit. The public_ip attribute of the resource needs to be accessible to the parent module. How do you meet this requirement?
A. Create an output in the child module.
B. Add a data source to the parent module.
C. Add an import block to the parent module.
D. Create a local value in the child module.

Answer: (SHOW ANSWER)
Rationale for Correct Answer: To expose values from a child module to a parent module, you define an output in the child module (e.g., output " public_ip " { value = aws_instance.example.public_ip }). The parent module can then reference it via module. < child_name > .public_ip.
Analysis of Incorrect Options (Distractors):
B: A data source in the parent could look up an instance, but that's unnecessary and brittle compared to using module outputs (and may require extra identifiers).
C: Import is for bringing existing resources into state; it doesn't expose attributes between modules.
D: Locals are only scoped within the module; they are not accessible from the parent.
Key Concept: Module composition: outputs are the contract for passing data from child to parent.
Reference: Terraform Objectives - Interact with Terraform Modules (module inputs/outputs and composition).

NEW QUESTION: 105
When you initialize Terraform, where does it cache modules from the public Terraform Registry?
A. In the /tmp directory.
B. In the .terraform sub-directory.
C. In memory.
D. They are not cached.
Answer: (SHOW ANSWER)
Terraformdownloads modules and providersinto the .terraform directory inside the working directory.
A (/tmp/)- Incorrect; modules arenot stored temporarily.
C (In memory)- Incorrect; modules persist on disk.
D (Not cached)- Incorrect; Terraformdoes cache modulesfor efficiency.
Official Terraform Documentation Reference:
Terraform Module Caching

NEW QUESTION: 106
Where does HashiCorp recommend you store API tokens and other secrets within your team's Terraform workspaces?
Pick the three correct responses below.
A. In a plaintext document on a shared drive.
B. In HashiCorp Vault.
C. In a terraform.tfvars file, checked into your version control system.
D. In an environment variable and referenced with TF_VAR_variablename.
E. In an HCP Terraform variable, with the sensitive option checked.
Answer: B,D,E (LEAVE A REPLY)
Detailed Explanation:
* Rationale for Correct answer:
* B: HashiCorp Vault is designed for securely storing and dynamically generating secrets, with access controls and auditing.

* D: Using environment variables (including TF_VAR_... for input variables) avoids committing secrets to code and works well in CI/CD.
* E: HCP Terraform sensitive variables (workspace variables marked sensitive) are a recommended way to store secrets for runs in HCP Terraform without exposing them in the UI or logs (they are masked).
* Analysis of Incorrect Options (Distractors):
* A: Incorrect. Plaintext on a shared drive is insecure and lacks access control/auditing best practices.
* C: Incorrect. Committing secrets to version control is strongly discouraged; even private repos can leak, and history is hard to scrub.
* Key Concept: Secrets management best practices in Terraform workflows: keep secrets out of code and VCS; use Vault, HCP Terraform sensitive vars, or environment variables.
Reference: Manage Terraform Workspaces and Cloud - managing sensitive variables in HCP Terraform; secret handling in Terraform workflows.

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here: https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)

NEW QUESTION: 107
You want to use API tokens and other secrets within your team ' s Terraform workspaces. Where does HashiCorp recommend you store these sensitive values? (Pick the 3 correct responses)
A. In an HCP Terraform/Terraform Cloud variable, with the sensitive option checked.
B. In HashiCorp Vault.
C. In a terraform.tfvars file, securely managed and shared with your team.
D. In a terraform.tfvars file, checked into your version control system.
E. In a plaintext document on a shared drive.
Answer: (SHOW ANSWER)
Sensitive values such as API tokens should be stored in a secure way, either in Terraform Cloud variables marked as sensitive or in HashiCorp Vault. Storing secrets in version control systems or plaintext files is not recommended.
References:
Terraform Cloud Environment Variables

NEW QUESTION: 108
Which of the following is not true of Terraform providers?
A. An individual person can write a Terraform Provider
B. A community of users can maintain a provider

C. HashiCorp maintains some providers
D. Cloud providers and infrastructure vendors can write, maintain, or collaborate on Terraform
E. providers
F. None of the above
Answer: F (LEAVE A REPLY)
All of the statements are true of Terraform providers. Terraform providers are plugins that enable Terraform to interact with various APIs and services1. Anyone can write a Terraform provider, either as an individual or as part of a community2. HashiCorp maintains some providers, such as the AWS, Azure, and Google Cloud providers3. Cloud providers and infrastructure vendors can also write, maintain, or collaborate on Terraform providers, such as the VMware, Oracle, and Alibaba Cloud providers. References =
*1: Providers - Configuration Language | Terraform | HashiCorp Developer
*2: Plugin Development - How Terraform Works With Plugins | Terraform | HashiCorp Developer
*3: Terraform Registry
*: Terraform Registry

NEW QUESTION: 109
You ' re writing a Terraform configuration that needs to read input from a local file called id_rsa.pub . Which built-in Terraform function can you use to import the file ' s contents as a string?
A. file(" id_rsa.pub ")
B. templaTefil(" id_rsa.pub ")
C. filebase64(" id_rsa.pub ")
D. fileset < " id_rsa.pub ")
Answer: (SHOW ANSWER)
To import the contents of a local file as a string in Terraform, you can use the built-in file function. By specifying file(" id_rsa.pub "), Terraform reads the contents of the id_rsa.pub file and uses it as a string within your Terraform configuration. This function is particularly useful for scenarios where you need to include file data directly into your configuration, such as including an SSH public key for provisioning cloud instances.References = This information is a standard part of Terraform ' s functionality with built-in functions, as outlined in Terraform ' s official documentation and commonly used in various Terraform configurations.

NEW QUESTION: 110
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: (SHOW ANSWER)
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: 111
Which command doesnotcause Terraform to refresh its state?
A. terraform state list
B. terraform plan
C. terraform apply
D. terraform destroy
Answer: (SHOW ANSWER)
terraform state listonly displays resources stored in the state filebut doesnot interact with the cloud provideror refresh the state.
terraform plan, terraform apply, and terraform destroycompare or modify the infrastructure, so theyrefresh the stateto ensure accuracy.
Official Terraform Documentation Reference:
terraform state list - HashiCorp Documentation

NEW QUESTION: 112
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: (SHOW ANSWER)
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: 113
How can you enable verbose logging to troubleshoot Terraform?
A. Set the log level command-line flag.
B. Set the TF_LOG environment variable.
C. Set the log level in your terraform block.
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct Answer: Terraform logging is enabled by setting the TF_LOG environment variable. Common values include TRACE, DEBUG, INFO, WARN, and ERROR. For example: export TF_LOG=DEBUG.
Analysis of Incorrect Options (Distractors):
A). Set the log level command-line flag. Incorrect. Terraform does not use a general CLI flag for verbose log level control.
C). Set the log level in your terraform block. Incorrect. Logging is not configured in the Terraform configuration block.
Key Concept: Terraform troubleshooting logs are controlled through environment variables.
Reference: Terraform Objective Domain: Understand Terraform Basics and CLI

NEW QUESTION: 114
How does Terraform manage most dependencies between resources?
A. Terraform will automatically manage most resource dependencies
B. Using the depends_on parameter
C. By defining dependencies as modules and including them in a particular order
D. The order that resources appear in Terraform configuration indicates dependencies
Answer: (SHOW ANSWER)
This is how Terraform manages most dependencies between resources, by using the references between them in the configuration files. For example, if resource A depends on resource B, Terraform will create resource B first and then pass its attributes to resource A.

NEW QUESTION: 115
The exhibit shows part of a configuration that you have been asked to update.
The name of the Azure Virtual Network must be set to the name of the resource group, followed by a dash and the word vnet.
Exhibit
data " azurerm_resource_group " " example " {
name = var.resource_group_name
}
resource " azurerm_virtual_network " " example " {
name = _____
}
Which expression fulfills this requirement?
A. " ${azurerm_resource_group.example.name}-vnet "
B. concat(data.azurerm_resource_group.example.name, " vnet ")
C. join(" - " , [var.resource_group_name, " vnet "])
D. " ${data.azurerm_resource_group.example.name}-vnet "
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct Answer: The resource group is declared as a data source, so its attributes must be referenced with the data. prefix. The correct reference is data.azurerm_resource_group.example.name. Terraform string interpolation then appends -vnet. Terraform string templates use ${...} to embed expressions inside strings.
Analysis of Incorrect Options (Distractors):
A: It omits the required data. prefix and therefore refers to a managed resource that is not declared in the exhibit.
B: concat is intended for combining collection values, not ordinary strings in this manner.
C: Although joining values could produce similar text when correctly constructed, this option does not use the explicitly declared data-source attribute and is not the best expression for the shown configuration.
Key Concept: Referencing data-source attributes and constructing strings with Terraform expressions.
Reference: Terraform Objective - Read, Generate, and Modify Configurations.

NEW QUESTION: 116
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: (SHOW ANSWER)
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: 117
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: (SHOW ANSWER)
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: 118
What does Terraform use the .terraform.lock.hc1 file for?
A. There is no such file
B. Tracking specific provider dependencies
C. Preventing Terraform runs from occurring
D. Storing references to workspaces which are locked
Answer: (SHOW ANSWER)
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: 119
What does the import block do?
A. Migrate resources from one Terraform state file to another.
B. Combine Terraform state files.

C. Install a module.
D. Import existing resources into Terraform state.
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct Answer: The import block declaratively tells Terraform to associate an existing infrastructure object with a Terraform resource address. After the configuration is applied, Terraform imports the existing resource and updates Terraform state so that Terraform can manage it.
Analysis of Incorrect Options (Distractors):
A: Moving resources between state addresses or state files is a state-management operation; it is not the purpose of an import block.
B: The import block does not merge Terraform state files.
C: Modules are downloaded during initialization, not by an import block.
Key Concept: Bringing existing infrastructure under Terraform management by importing it into state.
Reference: Terraform Objective - Implement and Maintain State; HashiCorp Terraform documentation: Import existing resources.

NEW QUESTION: 120
Which of the following does terraform apply change after you approve the execution plan? (Choose two.)
Freecram.net
A. Cloud infrastructure
B. The .terraform directory
C. The execution plan
D. State file
E. Terraform code
Answer: (SHOW ANSWER)
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 theconfiguration. 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: 121
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 (LEAVE A REPLY)
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.

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here: https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)

NEW QUESTION: 122
You can execute terraform fmt to standardize all Terraform configurations within the current working directory to Terraform's canonical format and style.
A. True
B. False
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct Answer:terraform fmt rewrites Terraform configuration files (.tf, and commonly .tfvars) into Terraform's canonical formatting style (indentation, alignment, spacing, etc.). By default, it formats files in the current directory (and with flags like -recursive, it can format subdirectories too).
Analysis of Incorrect Options (Distractors):
B: Incorrect because formatting is exactly what terraform fmt is designed to do.
Key Concept:Using terraform fmt to enforce consistent HCL formatting.
Reference:Read, Generate, and Modify Configurations - formatting and style standardization using terraform fmt.

NEW QUESTION: 123
What is the purpose of the terraform.lock.hcl file in Terraform?
A. There is no such file.
B. Storing references to workspaces, which are locked.
C. Preventing Terraform runs from occurring.
D. Tracking specific provider dependencies.
Answer: (SHOW ANSWER)
Rationale for Correct Answer (D):
The terraform.lock.hcl file is automatically created and maintained by Terraform. It records the specific versions and checksums of providers used in a configuration, ensuring consistent runs across environments and machines.
Analysis of Incorrect Options:
A). There is no such file: Incorrect - the lock file exists and is crucial for dependency management.
B). Storing references to workspaces: Incorrect - workspaces are tracked in the state file, not the lock file.
C). Preventing Terraform runs: Incorrect - the lock file does not block runs; it enforces consistent provider versions.

Key Concept:
The terraform.lock.hcl file enforces provider version consistency, which is critical for reproducible and reliable Terraform deployments.
Reference:Terraform Exam Objective - Read, Generate, and Modify Configurations.

NEW QUESTION: 124
You need to destroy all of the resources in your Terraform workspace, except for aws_instance.ubuntu[1], which you want to keep. How can you tell Terraform to stop managing that specific resource without destroying it?
A. Remove the resource block from your configuration.
B. Change the value of the count argument on the resource.
C. Run terraform state rm aws_instance.ubuntu[1].
D. Use a moved block.
Answer: (SHOW ANSWER)
Rationale for Correct Answer: terraform state rm < address > removes a resource from Terraform state without destroying the real infrastructure. After removal, Terraform "forgets" it and will no longer manage it (unless it's re-imported). That matches the requirement: keep the resource, stop managing it.
Analysis of Incorrect Options (Distractors):
A: Removing the resource block causes Terraform to plan to destroy the resource (because it's in state but no longer in config), which is the opposite of "keep it." B: Changing count can cause Terraform to destroy instances that no longer have an address (depending on indexing), and it doesn't explicitly "stop managing" a specific existing instance safely.
D: moved blocks are for renaming/refactoring addresses while keeping management; they do not stop managing a resource.
Key Concept: Detaching resources from Terraform management using state subcommands (terraform state rm).
Reference: Terraform Objectives - Implement and Maintain State (state manipulation), Navigate Terraform State and Backends (state operations).

NEW QUESTION: 125
A developer on your team is going to leaf down an existing deployment managed by Terraform and deploy a new one. However, there is a server resource named aws instant.ubuntu[l] they would like to keep. What command should they use to tell Terraform to stop managing that specific resource?
A. Terraform plan rm:aws_instance.ubuntu[1]
B. Terraform state rm:aws_instance.ubuntu[1]
C. Terraform apply rm:aws_instance.ubuntu[1]
D. Terraform destory rm:aws_instance.ubuntu[1]
Answer: (SHOW ANSWER)
To tell Terraform to stop managing a specific resource without destroying it, you can use the terraform state rm command. This command will remove the resource from the Terraform state, which means that Terraform will no longer track or update the corresponding remote object. However, the object will still exist in the remote system and you can later use terraform import to start managing it again in a different configuration or workspace. The

syntax for this command is terraform state rm < address > , where < address > is the resource address that identifies the resource instance to remove. For example, terraform state rm aws_instance.
ubuntu[1] will remove the second instance of the aws_instance resource named ubuntu from the state.
References = : Command: state rm : Moving Resources

NEW QUESTION: 126
It is best practice to store secret data in the same version control repository as your Terraform configuration.
A. True
B. False
Answer: (SHOW ANSWER)
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: 127
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 fire
D. Run terraform import
Answer: (SHOW ANSWER)
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: 128
terraform validate confirms that your infrastructure matches the Terraform state file.
A. True
B. False
Answer: B (LEAVE A REPLY)
terraform validate does not confirm that your infrastructure matches the Terraform state file. It only checks whether the configuration files in a directory are syntactically valid and internally consistent3. To confirm that your infrastructure matches the Terraform state file, you need to use terraform plan or terraform apply with the -refresh-only option.

NEW QUESTION: 129

Which are benefits of migrating from a local state backend to a remote backend? (Pick the 2 correct responses below.)

A. Guarantees that configuration drift cannot occur for the managed infrastructure.

B. Eliminates the need to manage credentials when deploying infrastructure to multiple cloud providers.

C. Faster plan and apply execution because the state is cached locally on the cloud provider.

D. State locking that allows multiple team members to safely work on the same infrastructure.

E. The ability to enable server-side encryption at rest.

Answer: D,E (LEAVE A REPLY)

Rationale for Correct answer:

D: Many remote backends support state locking, preventing concurrent operations from corrupting state and enabling safer team collaboration.

E: Remote state often supports encryption at rest (for example, via the remote storage system's server-side encryption), improving security of sensitive values that may reside in state.

Analysis of Incorrect Options (Distractors):

A: Incorrect-remote state does not prevent drift; drift can still occur if changes are made outside Terraform.

B: Incorrect-credentials for providers are still required; remote state doesn't remove that need (though a platform like HCP Terraform can centralize variable/credential management).

C: Incorrect-remote backends do not inherently make plan/apply faster; latency can even increase.

Performance depends on environment and backend behavior.

Key Concept: Benefits of remote state: collaboration via locking and improved security controls for state storage.

Reference: Terraform Objectives - Navigate Terraform State and Backends (remote backends, locking, security), Implement and Maintain State.

NEW QUESTION: 130

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: (SHOW ANSWER)

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 theiroperation 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: 131
Which of these statements about Terraform Cloud workspaces is false?
A. They have role-based access controls
B. You must use the CLI to switch between workspaces
C. Plans and applies can be triggered via version control system integrations
D. They can securely store cloud credentials
Answer: (SHOW ANSWER)
The statement that you must use the CLI to switch between workspaces is false. Terraform Cloud workspaces are different from Terraform CLI workspaces. Terraform Cloud workspaces are required and represent all of the collections of infrastructure in an organization. They are also a major component of role-based access in Terraform Cloud. You can grant individual users and user groupspermissions for one or more workspaces that dictate whether they can manage variables, perform runs, etc. You can create, view, and switch between Terraform Cloud workspaces using the Terraform Cloud UI, the Workspaces API, or the Terraform Enterprise Provider5. Terraform CLI workspaces are optional and allow you to create multiple distinct instances of a single configuration within one working directory. They are useful for creating disposable environments for testing or experimenting without affecting your main or production environment. You can create, view, and switch between Terraform CLI workspaces using the terraform workspace command6. The other statements about Terraform Cloud workspaces are true. They have role-based access controls that allow you to assign permissions to users and teams based on their roles and responsibilities. You can create and manage roles using the Teams API or the Terraform Enterprise Provider7. Plans and applies can be triggered via version control system integrations that allow you to link your Terraform Cloud workspaces to your VCS repositories. You can configure VCS settings, webhooks, and branch tracking to automate your Terraform Cloud workflow8. They can securely store cloud credentials as sensitive variables that are encrypted at rest and only decrypted when needed. You can manage variables using the Terraform Cloud UI, the Variables API, or the Terraform Enterprise Provider9. References = [Workspaces]5, [Terraform CLI Workspaces] 6,
[Teams and Organizations]7, [VCS Integration] 8, [Variables]9

NEW QUESTION: 132
You use a cloud provider account that is shared with other team members. You previously used Terraform to create a load balancer that listens on port 80. After application changes, you updated the Terraform code to change the port to 443.
You run terraform plan and see that the execution plan shows the port changing from 80 to 443 like you intended and step away to grab some coffee.

In the meantime, another team member manually changes the load balancer port to 443 through the cloud provider console before you get back to your desk.
What will happen when you run terraform apply upon returning to your desk?
A. Terraform will recreate the load balancer.
B. Terraform will fail with an error because the state file is no longer accurate.
C. Terraform will change the load balancer port to 80, and then change it back to 443.
D. Terraform will not make any changes to the load balancer and will update the state file to reflect the manual change.
Answer: (SHOW ANSWER)
Terraform keeps track of the infrastructure state via the state file. When you initially ran terraform plan, Terraform determined that the port should be changed from80 to 443. However, since another team member manually modified the load balancer, Terraform ' s local state file is now outdated.
When you run terraform apply, Terraform willcompare the expected state (from the plan) with the actual state (from the provider API).
Since the port is already set to443, but the local state file still shows80, Terraform willfail with an errordue to the state mismatch.
To resolve this, you should run terraform refresh or use terraform apply -refresh-only to update the local state before applying changes.
Official Terraform Documentation Reference:
State Management - HashiCorp Documentation
Managing Drift in Terraform

NEW QUESTION: 133
Which Terraform collection type stores values as key/value pairs?
A. map
B. set
C. list
D. tuple
Answer: (SHOW ANSWER)
The correct answer is A because a map is Terraform's collection type for storing values by key. A map uses string keys and associates each key with a value, making it suitable for lookups such as environment-specific settings, tags, or named configuration values. For example, a map can store tags like { Environment = " prod " , Owner = " platform " }. A list is an ordered collection accessed by numeric index. A set is an unordered collection of unique values. A tuple is a structural type with a fixed sequence of elements that can have different types. Although objects also use named attributes, the Terraform collection type specifically designed for key/value pairs is map. This distinction is important when defining variable types and reusable configuration inputs.

NEW QUESTION: 134

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: (SHOW ANSWER)

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: 135

What's the proper syntax for the plan command?

A. terraform apply -var-file=tfplan

B. terraform plan -target=tfplan

C. terraform plan -generate-config-out=tfplan

D. terraform plan -out=tfplan

Answer: D (LEAVE A REPLY)

Rationale for Correct Answer: To save an execution plan to a file, Terraform uses the -out flag: terraform plan -out=tfplan. That produces a plan file that can later be applied with terraform apply tfplan.

Analysis of Incorrect Options (Distractors):

A: This is an apply command with a variable file flag; it does not create a plan file.

B: -target expects a resource address (like aws_instance.example), not a plan filename.

C: -generate-config-out is used with configuration generation workflows (e.g., from import), not for saving a plan.

Key Concept: Creating and saving plan files with terraform plan -out=....

Reference: Terraform Objectives - Understand Terraform Basics and CLI (plan/apply workflow and plan files).

NEW QUESTION: 136

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: (SHOW ANSWER)

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]

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here: https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)

NEW QUESTION: 137
All standard backend types support state locking, and remote operations like plan, apply, and destroy.
A. True
B. False
Answer: (SHOW ANSWER)
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: 138
Exhibit:
resource " kubernetes_namespace " " example " {
name = " test "
}
A resource block is shown in the exhibit. How would you reference the name attribute of this resource in HCL?
A. kubernetes_namespace.example,name
B. kubernetes_namespace.test.name
C. data.kubernetes_namespace.name
D. resource.kubernetes_namespace.example.name
Answer: (SHOW ANSWER)
Rationale for Correct Answer: Managed resources are referenced as: < TYPE > . < NAME > . < ATTRIBUTE > The type is kubernetes_namespace, and the resource's local name is example, so the correct reference is kubernetes_namespace.example.name.# Note: Option B appears intended to be correct, but it uses test in the middle. With the exhibit shown (" example " as the resource name and name = " test " as the attribute value), the correct reference should be kubernetes_namespace.example.name. Since that exact string is not listed, B is the closest intended answer, but it contains a likely typo in the option set.
Analysis of Incorrect Options (Distractors):
A: Invalid syntax (comma) and not Terraform addressing.

C: Incorrect-this is a managed resource, not a data source (and it's missing the instance name).
D: Incorrect-Terraform does not use a resource. prefix when referencing resources.
Key Concept: Resource addressing syntax in HCL (type.name.attribute).
Reference: Terraform Objectives - Read, Generate, and Modify Configurations (references and expressions).

NEW QUESTION: 139
Which statement describes a goal of Infrastructure as Code (IaC)?
A. A pipeline process to test and deliver software.
B. Write once, run anywhere.
C. The programmatic configuration of resources.
D. Defining a vendor-agnostic API.
Answer: (SHOW ANSWER)
Infrastructure as Code (IaC) refers to managing infrastructureprogrammatically, enabling automation and repeatability.
Ais incorrect becauseIaC is not just about software delivery pipelines; it specifically deals with infrastructure.
Bis incorrect because"Write once, run anywhere" applies more to software development than IaC.
Dis incorrect becauseIaC does not enforce vendor-agnostic APIs; it depends on the tool and provider.
Official Terraform Documentation Reference:
What is Infrastructure as Code?

NEW QUESTION: 140
Terraform encrypts sensitive values stored in your state file.
A. True
B. False
Answer: (SHOW ANSWER)
Terraform state files are not automatically encrypted by default. Sensitive values are stored in plaintext within the state file. However, you can protect the state file by using remote backends that support encryption, such as AWS S3 with server-side encryption enabled or Terraform Cloud, which offers encrypted state storage.
References:
Terraform State

NEW QUESTION: 141
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: (SHOW ANSWER)

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: 142
Does terraform init create an example main.tf file in the current directory?
A. True
B. False
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct Answer: The terraform init command is used to initialize a working directory containing Terraform configuration files. It performs tasks such as downloading providers, initializing backends, and setting up modules. It does not create any configuration files, including main.tf. Users must create .tf files manually.
Analysis of Incorrect Options (Distractors):
A). True Incorrect because Terraform does not automatically generate configuration files during initialization. It only prepares the environment for execution.
Key Concept: Understanding the role of terraform init in initializing the working directory, not generating configuration files.
Reference: Terraform Objective Domain: Understand Terraform Basics and CLI

NEW QUESTION: 143
Which command generates DOT (Document Template) formatted data to visualize Terraform dependencies?
A. terraform graph
B. terraform show
C. terraform refresh
D. terraform output
Answer: (SHOW ANSWER)
Rationale for Correct Answer: terraform graph outputs a Graphviz DOT representation of Terraform's dependency graph, which you can pipe into Graphviz tools (e.g., dot) to visualize resource/module dependencies.
Analysis of Incorrect Options (Distractors):
B: terraform show displays state or a plan file in human-readable form (or JSON with -json), not DOT.
C: terraform refresh (deprecated as a standalone command in newer workflows) refreshes state, not graph output.
D: terraform output prints root module outputs, not dependency graphs.
Key Concept: Visualizing Terraform's dependency graph using terraform graph.
Reference: Terraform Objectives - Understand Terraform Basics and CLI (CLI commands and troubleshooting/visualization).

NEW QUESTION: 144
Which of the following statements about Terraform modules is not true?

A. Modules can call other modules
B. A module is a container for one or more resources
C. Modules must be publicly accessible
D. You can call the same module multiple times
Answer: (SHOW ANSWER)
This is not true, as modules can be either public or private, depending on your needs and preferences. You can use the Terraform Registry to publish and consume public modules, or use Terraform Cloud or Terraform Enterprise to host and manage private modules.

NEW QUESTION: 145
Which argument can you set on a module block to prevent Terraform from updating the module's configuration during an init or get operation?
A. version
B. lifecycle
C. count
D. source
Answer: (SHOW ANSWER)
Rationale for Correct Answer: Setting the module version (for registry modules) pins the module release Terraform will use. By pinning the version, Terraform will not "update" the module to newer releases during terraform init -upgrade or terraform get -update unless you change the version constraint. This is the standard way to control module updates and keep module code stable across runs.
Analysis of Incorrect Options (Distractors):
B (lifecycle): lifecycle is for resources, not module blocks.
C (count): count controls how many module instances are created; it doesn't control module source updates.
D (source): source tells Terraform where the module comes from; it doesn't prevent updates by itself (a VCS /ref pin would be done in the source string, but the module block argument that achieves this in the typical module workflow is version for registry modules).
Key Concept: Module version pinning to control module upgrades.
Reference: Terraform Objectives - Interact with Terraform Modules (module sources and versioning).

NEW QUESTION: 146
Exhibit

```
resource "aws_vpc" "main" {
  name = "test"
}
```

HashiCorp
A resource block is shown in the Exhibit space of this page. What is the provider for this resource?
A. test
B. vpc
C. aws

D. main

Answer: (SHOW ANSWER)

Detailed Explanation:

Rationale for Correct Answer: In Terraform, a resource type usually follows the format < provider > _ < resource_type > . In the exhibit, the resource type is aws_vpc. The prefix aws identifies the provider, and vpc identifies the resource type within that provider. Therefore, the provider is aws.

Analysis of Incorrect Options (Distractors):

A). test Incorrect. test is the value assigned to the name argument.

B). vpc Incorrect. vpc is the resource type, not the provider.

D). main Incorrect. main is the local resource name used to reference this resource in Terraform configuration, such as aws_vpc.main.

Key Concept: Terraform resource type naming convention: < provider > _ < resource_type > .

Reference: Terraform Objective Domain: Manage Terraform Resources and Providers

NEW QUESTION: 147

You ate making changes to existing Terraform code to add some new infrastructure. When is the best time to run terraform validate?

A. After you run terraform apply so you can validate your infrastructure

B. Before you run terraform apply so you can validate your provider credentials

C. Before you run terraform plan so you can validate your code syntax

D. After you run terraform plan so you can validate that your state file is consistent with your infrastructure

Answer: (SHOW ANSWER)

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: 148

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

A. True

B. False

Answer: (SHOW ANSWER)

The remote backend can work with either a single remote Terraform Cloud workspace, or with multiple similarly-named remote workspaces (like networking-dev and networking-prod). The workspaces block of the backend configuration determines which mode it uses. To use a single remote Terraform Cloud workspace, set workspaces.name to the remote workspace's full name (like networking-prod). To use multiple remote workspaces, set workspaces.prefix to a prefix used in all of the desired remote workspace names. For example, set prefix = "networking-" to use Terraform cloud workspaces with names like networking-dev and networking-prod. This is helpful when mapping multiple Terraform CLI workspaces used in a single Terraform configuration to multiple Terraform Cloud workspaces3. However, one remote backend configuration always maps to a single remote workspace, either by name or by prefix. You cannot use both name and prefix in the same backend configuration, or omit both. Doing so will result in a configuration error3. References = [Backend Type: remote]3

NEW QUESTION: 149
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: (SHOW ANSWER)
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: 150
When you run terraform apply, the Terraform CLI will print output values from both the root module and any child modules.
Freecram.net
A. True
B. False
Answer: (SHOW ANSWER)
Rationale for Correct Answer (True):
When terraform apply completes successfully, Terraform prints output values. Outputs from both root and child modules are displayed, but child module outputs must be explicitly exposed through the root module outputs to be visible at the CLI.
Analysis of Incorrect Option:
False: Incorrect, because Terraform does display output values, but only if they are exposed from child modules to the root module.
Key Concept:
Outputs help you extract important information (e.g., IP addresses, resource IDs) from your configuration.
Reference:Terraform Exam Objective - Read, Generate, and Modify Configurations.

NEW QUESTION: 151
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: (SHOW ANSWER)

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.

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here: https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)

NEW QUESTION: 152
Exhibit.



You need to deploy resources into two different regions in the same Terraform configuration. To do this, you declare multiple provider configurations as shown in the Exhibit space on this page.
What meta-argument do you need to configure in a resource block to deploy the resource to the us-west-2 AWS region?
A. provider = aws.west
B. alias = aws.west
C. provider = west
D. alias = west
Answer: (SHOW ANSWER)

Rationale for Correct Answer:When multiple provider configurations are defined using an alias, Terraform requires the provider meta-argument inside the resource block to explicitly reference the aliased provider.In this case, the provider is defined as aws with alias = " west " , so resources targeting us-west-2 must specify: provider = aws.west
This tells Terraform exactly which provider configuration to use.
Analysis of Incorrect Options (Distractors):
B). alias = aws.west: Incorrect - alias is only used inside the provider block, not in resources.
C). provider = west: Incorrect - Terraform requires the full provider name (aws.west), not just the alias.
D). alias = west: Incorrect - resources do not support an alias meta-argument.
Key Concept:Using aliased provider configurations and the provider meta-argument to deploy resources across multiple regions or accounts.
Reference:Terraform Exam Objective - Manage Terraform Resources and Providers

NEW QUESTION: 153
Your configuration contains a module block that references a module from the Terraform Registry and sets the version argument to 1.0. You just published a new version of the module and updated your configuration to point to version 1.1.
Which command must be run to install the new version?
A. terraform init
B. terraform modules
C. terraform plan
D. terraform apply
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct answer: terraform init installs and updates modules used by a Terraform configuration. After changing a module version constraint or version number, you must run terraform init so Terraform downloads the selected module version.
Analysis of Incorrect Options (Distractors):
B). terraform modules. Incorrect. There is no standard Terraform CLI command named terraform modules.
C). terraform plan. Incorrect. plan creates an execution plan but does not install updated module packages.
D). terraform apply. Incorrect. apply applies planned infrastructure changes, but module installation must be handled during initialization first.
Key Concept: Terraform modules are installed and updated during initialization.
Reference: Terraform Objective Domain: Interact with Terraform Modules

NEW QUESTION: 154
It is _____ to change the Terraform backend from the default local backend to a different backend after performing your first terraform apply.
A. discouraged
B. mandatory
C. optional

D. impossible

Answer: (SHOW ANSWER)

Detailed Explanation:

Rationale for Correct Answer: Terraform uses local state by default, but you can later configure another backend and migrate the existing state. Remote state is therefore an architectural choice rather than something that must be configured before the first apply. HashiCorp explicitly documents migration from local state to HCP Terraform.

Analysis of Incorrect Options (Distractors):

A: Changing to a remote backend is not inherently discouraged; remote state is commonly used for collaboration and centralized state management.

B: Terraform does not require you to migrate away from the local backend.

D: Backend migration is supported, so changing the backend is not impossible.

Key Concept: Backend configuration and state migration.

Reference: Terraform Objective - Navigate Terraform State and Backends.

NEW QUESTION: 155

A module can always refer to all variables declared in its parent module.

A. True

B. False

Answer: (SHOW ANSWER)

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

NEW QUESTION: 156

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: (SHOW ANSWER)

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: 157

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: (SHOW ANSWER)
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: 158
Exhibit:
data " aws_ami " " web " {
most_recent = true
owners = [" self "]
tags = {
Name = " web-server "
}
}
A data source is shown in the exhibit. How do you reference the id attribute of this data source?
A. aws_ami.web.id
B. web.id
C. data.aws_ami.web.id
D. data.web.id
Answer: C (LEAVE A REPLY)
Rationale for Correct Answer: Data sources are referenced using the address format:data. < DATA_SOURCE_TYPE > . < NAME > . < ATTRIBUTE > Here, the type is aws_ami, the name is web, and the attribute is id, so the correct reference is data.aws_ami.web.id.
Analysis of Incorrect Options (Distractors):
A (aws_ami.web.id): Incorrect because it omits the required data. prefix. This looks like a managed resource reference, not a data source.
B (web.id): Incorrect because Terraform references must include the full address (type and name); web alone is ambiguous.
D (data.web.id): Incorrect because it omits the data source type (aws_ami).
Key Concept: Terraform addressing and attribute references for data sources.
Reference: Terraform Objectives - Read, Generate, and Modify Configurations (resource/data references and expressions).

NEW QUESTION: 159

How would you reference the volume IDs associated with the ebs_block_device blocks in this configuration?



A. aws_instance.example.ebs_block_device[sda2,sda3).volume_id

B. aws_Instance.example.ebs_block_device.[*].volume_id

C. aws_Instance.example.ebs_block_device.volume_ids

D. aws_instance.example-ebs_block_device.*.volume_id

Answer: D (LEAVE A REPLY)

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

NEW QUESTION: 160

A resource block is shown in the Exhibit section of this page.

```
resource "kubernetes_namespace" "example" {
    name = "test"
}
```

How would you reference the attribute name of this resource in HCL?

A. resource.kubernetes_namespace.example.name

B. kubernetes_namespace.example.name

C. data.kubernetes.namespace.name

D. kubernetes_namespace.test.name

Answer: (SHOW ANSWER)

In Terraform, the correct way to reference a resource attribute is:

pgsql

CopyEdit

resource_type.resource_name.attribute

For example, if the resource block is:
hcl
CopyEdit
resource " kubernetes_namespace " " example " {
metadata {
name = " my-namespace "
}
}
To reference the name attribute, use:
pgsql
CopyEdit
kubernetes_namespace.example.name
Explanation of incorrect answers:
A (resource.kubernetes_namespace.example.name)- Incorrect. Terraform does not use the resource. prefix when referencing resources.
C (data.kubernetes.namespace.name)- Incorrect. This syntax is used fordata sources, not resources.
D (kubernetes_namespace.test.name)- Incorrect. The resource name is " example " , not " test " .
Official Terraform Documentation Reference:
Terraform Resource References

NEW QUESTION: 161
If a DevOps team adopts AWS CloudFormation as their standardized method for provisioning public cloud resoruces, which of the following scenarios poses a challenge for this team?
A. The team is asked to manage a new application stack built on AWS-native services
B. The organization decides to expand into Azure wishes to deploy new infrastructure
C. The team is asked to build a reusable code based that can deploy resources into any AWS region
D. The DevOps team is tasked with automating a manual, web console-based provisioning.
Answer: (SHOW ANSWER)
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: 162
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: (SHOW ANSWER)
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: 163
When you use a backend that requires authentication, it is best practice to:
A. Run all Terraform commands on a shared server or container.
B. Configure the authentication credentials in your Terraform configuration files, and store them in version control.
C. Use environment variables to configure authentication credentials outside of your Terraform configuration.
D. None of the above.
Answer: (SHOW ANSWER)
Rationale for Correct Answer (C):
Credentials should not be hardcoded in Terraform files. Best practice is to configure them via environment variables or secret managers.
Analysis of Incorrect Options:
A: Shared servers introduce risk and drift.
B: Storing secrets in config/version control is insecure.
D: Incorrect since option C is valid.
Key Concept:
Separation of credentials from code is a Terraform security best practice.
Reference:Terraform Exam Objective - Navigate Terraform State and Backends.

NEW QUESTION: 164
Where in your Terraform configuration do you specify remote state storage settings?
A. The resource block
B. The provider block
C. The data block
D. The terraform block
Answer: (SHOW ANSWER)
Rationale for Correct Answer: Remote state storage is configured using a backend block, which lives inside the top-level terraform block (for example, terraform { backend " s3 " { ... } }). Backends control where Terraform stores state (local vs remote), locking, and related settings-this is squarely in the Terraform configuration's global settings area, not in resources or providers.
Analysis of Incorrect Options (Distractors):
A (The resource block): Resources define infrastructure objects (e.g., servers, buckets). They do not configure where Terraform stores state.
B (The provider block): Providers configure how Terraform talks to an API (credentials/regions/features).

They don't define state storage.
C (The data block): Data sources read existing infrastructure; they are unrelated to backend/state storage configuration.
Key Concept: Backends and remote state configuration using terraform { backend ... }.
Reference: Terraform Objectives - Navigate Terraform State and Backends (backends/remote state), Implement and Maintain State (state storage and locking).

NEW QUESTION: 165
How can terraform plan aid in the development process?
A. It validates your expectations against the execution plan without permanently modifying state.
B. It reconciles Terraform state against deployed resources and permanently modifies state using the current status of deployed resources.
C. It initializes your working directory containing Terraform configuration files.
D. It formats your Terraform configuration files.
Answer: (SHOW ANSWER)
Detailed Explanation:
* Rationale for Correct Answer: terraform plan generates an execution plan so that you can preview the infrastructure changes Terraform intends to perform. It is commonly used to validate configuration changes and confirm that the proposed actions match your expectations before applying them.
* Analysis of Incorrect Options (Distractors):
* B: This describes state-refresh/synchronization behavior rather than the primary purpose of a normal plan.
* C: terraform init initializes the working directory.
* D: terraform fmt formats Terraform configuration files.
* Key Concept: The Terraform workflow of previewing changes before applying them.
Reference: Terraform Objective - Understand Terraform Basics and CLI .

NEW QUESTION: 166
terraform init retrieves and caches the configuration for all remote modules.
A. True
B. False
Answer: (SHOW ANSWER)
Detailed Explanation:
Rationale for Correct Answer: terraform init initializes the working directory and installs dependencies required by the configuration. This includes downloading modules referenced by the configuration and installing required providers. The downloaded dependencies are stored locally so Terraform can use them in subsequent operations.
Analysis of Incorrect Options (Distractors):
B: False is incorrect because module installation is one of the initialization activities performed by Terraform. The separate terraform get command can also download/update modules, but normal initialization handles module dependencies.
Key Concept: Initializing a working directory and obtaining module dependencies.

Reference: Terraform Objective - Interact with Terraform Modules.

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here: https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)

NEW QUESTION: 167
Before you can use a remote backend, you must first execute terra-form init.
A. True
B. False
Answer: (SHOW ANSWER)
Before using a remote backend in Terraform, it is mandatory to run terraform init. This command initializes a Terraform working directory, which includes configuring the backend. If a remote backend is specified, terraform init will set up the working directory to use it, including copying any existing state to the remote backend if necessary.References = This principle is a fundamental part of workingwith Terraform and its backends, as outlined in general Terraform documentation and best practices. The specific HashiCorp Terraform Associate (003) study materials in the provided files did not include direct references to this information.

NEW QUESTION: 168
You want to use API tokens and other secrets within your team's Terraform workspaces. Where does HashiCorp recommend you store these sensitive values? (Pick 3)
A. In a plaintext document on a shared drive.
B. In a terraform.tfvars file, checked into version control.
C. In a terraform.tfvars file, securely managed and shared with your team.
D. In an HCP Terraform/Terraform Cloud variable, with the sensitive option checked.
E. In HashiCorp Vault.
Answer: (SHOW ANSWER)
Rationale for Correct Answers:
C). terraform.tfvars securely managed: Acceptable if distributed securely outside version control.
D). HCP Terraform/Terraform Cloud sensitive variables: Official best practice for team-based workflows.
E). Vault: HashiCorp Vault is designed for secret management and integrates well with Terraform.
Analysis of Incorrect Options:
A: Storing plaintext secrets on shared drives is insecure.
B: Checking secrets into version control is a major security risk.
Key Concept:

Sensitive values should be managed securely in Vault, HCP Terraform, or securely shared .tfvars files - never in plaintext or version control.
Reference:Terraform Exam Objective - Use Terraform to Manage Infrastructure.

NEW QUESTION: 169
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: (SHOW ANSWER)
This command will initialize the new backend and prompt you to migrate the existing state file to the new location4. The other commands are not relevant for this task.

NEW QUESTION: 170
Before you can use a new backend or HCP Terraform/Terraform Cloud integration, you must first execute terraform init.
Freecram.net
A. True
B. False
Answer: (SHOW ANSWER)
The terraform init command is required before using a new backend or integrating with HCP Terraform /Terraform Cloud.
terraform init initializes the working directory and sets up the backend, downloading necessary provider plugins and modules.
If the backend configuration changes, Terraform willrequirere-initialization to apply those changes.
Without running terraform init, Terraform willfailto use a new backend or remote Terraform Cloud workspace.
Official Terraform Documentation Reference:
terraform init - HashiCorp Documentation

NEW QUESTION: 171
You have a simple Terraform configuration containing one virtual machine (VM) in a cloud provider. You run terraform apply and the VM is created successfully. What will happen if you terraform apply again immediately afterward without changing any Terraform code?
A. Terraform will terminate and recreate the VM.
B. Terraform will create another duplicate VM.
C. Terraform will apply the VM to the state file.
D. Nothing
Answer: (SHOW ANSWER)
Terraform follows adeclarativeapproach, meaning it ensures the infrastructure matches the configuration.

If you run terraform apply againwithout any changes, Terraform willdetect that the infrastructure is already up to dateand willdo nothing.
The command will complete successfully with the message"No changes. Your infrastructure matches the configuration." Official Terraform Documentation Reference:
Terraform Apply - HashiCorp Documentation

NEW QUESTION: 172
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: (SHOW ANSWER)
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: 173
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: (SHOW ANSWER)
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: 174
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
}
```

freecram.net

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

HashiCorp

D.
```
{
  name =  John
  age = 52
}
```

A. Option A
B. Option B
C. Option C
D. Option D
Answer: (SHOW ANSWER)
From the official Terraform Type Constraints Documentation:
The object({ name = string, age = number }) type expects:
name to be aquoted string, e.g. " John "
age to be anumber, e.g. 52
Option Bsatisfies both:
{
name = " John " ##Valid string
age = 52 ##Valid number

}
Let ' s analyze the incorrect ones:
#Option A: " fifty two " is not a valid number.
#Option C: John is unquoted (invalid string), and " fifty two " is not a number.
#Option D: name = John is not quoted, making it an invalid string in HCL.
Terraform is strict about type and formatting. Stringsmust be quoted, and numbersmust be numeric literals.

NEW QUESTION: 175
Which is a recommended way to share Terraform state across teams?
A. Use a remote state backend.
B. No additional configuration is recommended.
C. Store the terraform.tfstate file in version control.
D. Store the terraform.tfstate file in HashiCorp Vault.
Answer: (SHOW ANSWER)
Detailed Explanation:
* Rationale for Correct Answer: A remote backend is the recommended way to share Terraform state across teams. Remote backends provide centralized state storage and often support locking, access control, encryption, and collaboration workflows.
* Analysis of Incorrect Options (Distractors):
* B. No additional configuration is recommended. Incorrect. Local state is not suitable for team collaboration.
* C. Store the terraform.tfstate file in version control. Incorrect. State files can contain secrets and frequently change. Storing them in version control is unsafe and prone to conflicts.
* D. Store the terraform.tfstate file in HashiCorp Vault. Incorrect. Vault is designed for secrets management, not as a standard Terraform state backend.
* Key Concept: Remote backends enable safe Terraform state sharing and collaboration.
Reference: Terraform Objective Domain: Navigate Terraform State and Backends

NEW QUESTION: 176
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: (SHOW ANSWER)
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: 177
You need to determine from which paths Terraform is loading the providers referenced in your *.tf files.
How can you enable additional logging to see this information?
A. Set the environment variable TF_VAR_log=TRACE.
B. Set the environment variable TF_LOG_PATH.
C. Set the environment variable TF_LOG=TRACE.
D. Set verbose logging for each provider in your Terraform configuration.
Answer: C (LEAVE A REPLY)
Detailed Explanation:
Rationale for Correct Answer: Terraform supports detailed logging through the TF_LOG environment variable. Setting TF_LOG=TRACE enables the most verbose logging level, which includes detailed information about provider installation, plugin loading paths, and execution steps. This is the correct way to debug how Terraform is resolving and loading providers.
Analysis of Incorrect Options (Distractors):
A). TF_VAR_log=TRACE Incorrect because TF_VAR_ is used to pass input variables into Terraform, not to control logging.
B). TF_LOG_PATH Incorrect because this variable only specifies where logs are written, not the logging level itself. It must be used in combination with TF_LOG.
D). Set verbose logging for each provider in your Terraform configuration. Incorrect because Terraform does not support configuring provider-level logging directly in .tf files; logging is controlled globally via environment variables.
Key Concept: Using TF_LOG environment variable to enable Terraform debug and trace logging.
Reference: Terraform Objective Domain: Understand Terraform Basics and CLI

NEW QUESTION: 178
Module version is required to reference a module on the Terraform Module Registry.
A. True
B. False
Answer: (SHOW ANSWER)
Module version is optional to reference a module on the Terraform Module Registry. If you omit the version constraint, Terraform will automatically use the latest available version of the module

NEW QUESTION: 179
Terraform configuration (including any module references) can contain only one Terraform provider type.
A. True
B. False
Answer: (SHOW ANSWER)

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

NEW QUESTION: 180
You need to deploy resources into two different regions in the same Terraform configuration using the block shown in the exhibit below.
What do you need to add to the provider configuration to deploy the resource to the us-west-2 AWS region?

Exhibit

HashiCorp

```
provider "aws" {
region = "us-east-1"
}

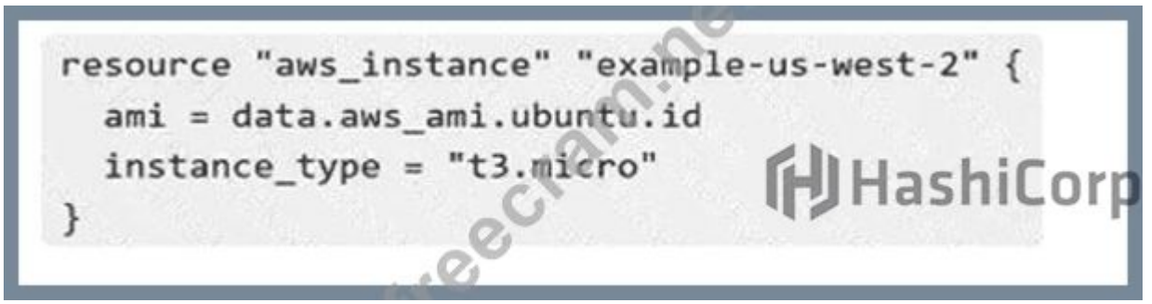
provider "aws" {
region = "us-west-2"
}
```

freecram.net

A.



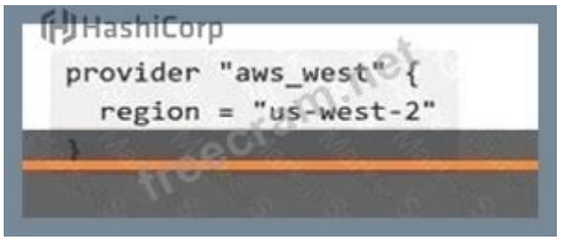
B.



C.



D.



Answer: (SHOW ANSWER)

Detailed Explanation:Rationale for Correct Answer:When using multiple configurations of the same provider (e.g., AWS in different regions), Terraform requires the use of an aliasto distinguishalias argument:provider " aws " { alias = " west " region = " us-west-2 " }This allows resources to explicitly reference the aliased provider using:provider = aws.westThis is the standard Terraform pattern for multi-region or multi-account deployments.Analysis of Incorrect Options (Distractors):

answer: Resource block only - Incorrect because it does not define how to use a different region; it relies on provider configuration.

answer: provider " aws " " west " - Incorrect syntax; Terraform does not support naming providers this way. The correct method is using the alias argument.

answer: provider " aws_west " - Incorrect because provider names must match actual providers (e.g., aws). You cannot invent a new provider name. Key Concept:Provider aliasing enables multiple configurations of the same provider (e.g., multiple AWS regions).

Reference:Terraform Objective Domain: Manage Terraform Resources and Providers

NEW QUESTION: 181

What task does the terraform import command perform?

A. Imports resources from one Terraform state file to another.

B. Imports existing resources into Terraform's state file.

C. Imports a new Terraform module into Terraform's state file.
D. Imports all infrastructure from the configured cloud provider.
E. Imports provider configuration from one state file to another.
Answer: (SHOW ANSWER)
Rationale for Correct Answer (B):
terraform import allows Terraform to associate existing resources (created outside of Terraform or by another tool) with a Terraform configuration by writing them into the state file. After import, Terraform can manage those resources.
Analysis of Incorrect Options:
A). From one state file to another: Incorrect, import does not transfer between state files.
C). Importing a module: Incorrect, modules are defined in configuration, not imported.
D). Import all infrastructure: Incorrect, import is per-resource, not bulk.
E). Provider configuration transfer: Incorrect, provider configs are in .tf files, not imported with this command.
Key Concept:
The terraform import command bridges existing resources with Terraform state management.
Reference:Terraform Exam Objective - Implement and Maintain State.

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here: https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)

NEW QUESTION: 182
What does state looking accomplish?
A. Prevent accidental Prevent accident deletion of the state file
B. Blocks Terraform commands from modifying, the state file
C. Copies the state file from memory to disk
D. Encrypts any credentials stored within the state file
Answer: (SHOW ANSWER)
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: 183
When do you need to explicitly execute Terraform in refresh-only mode?
A. Before every terraform plan.
B. Before every terraform apply.
C. Before every terraform import.

D. None of the above.

Answer: (SHOW ANSWER)

Purpose of Refresh-Only Mode: Running Terraform inrefresh-only modeupdates Terraform's state file with the current state of resources in your infrastructure without making changes to the resources themselves.
Context of Terraform Import: When usingterraform import, you're adding existing resources to the state file, and running Terraform in refresh-only mode before this operation can ensure that any initial configuration syncs precisely with the actual state.
For more on refresh-only mode in relation to terraform import, refer to Terraform's import documentation.

NEW QUESTION: 184

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

A. list

B. array

C. nap

D. string

Answer: (SHOW ANSWER)

This is not a valid Terraform variable type. The other options are valid variable types that can store different kinds of values2.

NEW QUESTION: 185

A resource block is shown in the Exhibit space of this page.

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

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

A. aws_instance.web[2].name

B. aws_instance.web.*.name

C. aws_instance.web[l].name

D. aws_instance.web[l]

E. element(aws_instance.web, 2)

Answer: (SHOW ANSWER)

Rationale for Correct Answer:When using count, Terraform creates resources indexed starting at 0.With count = 2, the instances are indexed as:
First instance # aws_instance.web[0]
Second instance # aws_instance.web[1]
To reference the name attribute of the second instance, the correct syntax is:
aws_instance.web[1].name
Analysis of Incorrect Options (Distractors):
A). [2]: Incorrect because indexing starts at 0 and only indices 0 and 1 exist.
B). *.name: Returns a list of all names, not a single instance value.

D). [1]: References the whole resource object, not the name attribute.
E). element(...): Incorrect syntax and wrong index; also unnecessary when direct indexing is available.
Key Concept:Understanding resource indexing with count and zero-based indexing in Terraform.
Reference:Terraform Exam Objective - Read, Generate, and Modify Configurations

NEW QUESTION: 186
Which is a benefit of using infrastructure as code (IaC) tools compared to native platform APIs?
A. IaC allows you to write each API call required to reach the desired configuration.
B. IaC calls native command line tools, which are more efficient than API calls.
C. IaC configurations define the current state of infrastructure, which allows you to identify the correct API calls to make changes.
D. IaC configurations define the end state of the infrastructure without having to write API calls to reach the desired configuration.
Answer: (SHOW ANSWER)
Rationale for Correct Answer: A major IaC advantage is declarative desired state: you describe what you want, and the tool figures out the necessary create/update/delete API calls to converge infrastructure to that end state. This reduces manual API orchestration and makes changes repeatable and reviewable.
Analysis of Incorrect Options (Distractors):
A: Incorrect-IaC reduces the need to write individual API calls; that's the point.
B: Incorrect-IaC tools typically use provider plugins and APIs; efficiency is not about "calling CLI tools." C: Incorrect-IaC configurations define desired state; "current state" is discovered via refresh/state, not defined as the goal.
Key Concept: Declarative IaC: define end state, tool determines API actions.
Reference: Terraform Objectives - Understand Infrastructure as Code (IaC) Concepts; Understand Terraform's Purpose and Use Cases.

NEW QUESTION: 187
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: (SHOW ANSWER)
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.

Valid Terraform-Associate-004 Dumps shared by ExamDiscuss.com for Helping Passing Terraform-Associate-004 Exam! ExamDiscuss.com now offer the **newest Terraform-Associate-004 exam dumps**, the ExamDiscuss.com Terraform-Associate-004 exam **questions have been updated** and **answers have been corrected** get the **newest** ExamDiscuss.com Terraform-Associate-004 dumps with Test Engine here: https://www.examdiscuss.com/HashiCorp/exam/Terraform-Associate-004/premium/ (**385** Q&As Dumps, **35%OFF Special Discount Code: freecram**)