DUMPSHQ

Free Questions for VA-002-P by dumpshq

Shared by Wilder on 22-07-2024

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

True or False:

Multiple providers can be declared within a single Terraform configuration file.

Options:

A- False

B- True

Answer:

B

Explanation:

Multiple provider blocks can exist if a Terraform configuration is composed of multiple providers, which is a common situation. To add multiple providers in your configuration, declare the providers, and create resources associated with those providers.

Question 2

Question Type: **MultipleChoice**

Which of the following best describes the default local backend?

Options:

A- The local backend stores state on the local filesystem locks the state using system APIs and performs operations locally.

B- The local backend is the directory where resources deployed by Terraform have direct access to in order to update their current state

C- The local backend is how Terraform connects to public cloud services, such as AWS, Azure, or GCP.

D- The local backend is where Terraform Enterprise stores logs to be processed by a log collector

Answer:

A

Explanation:

Information on the default local backend can be found at this link.

Example:

```
terraform {

backend 'local' {

path = 'relative/path/to/terraform.tfstate'

}

}
```

Question 3

Question Type: MultipleChoice

Select two answers to complete the following sentence:

Before a new provider can be used, it must be ______ and _______.

Options:

A- approved by HashiCorp

B- declared in the configuration

C- initialized

D- uploaded to source control

Answer:

B, C

Explanation:

Each time a new provider is added to configuration -- either explicitly via a provider block or by adding a resource from that provider -- Terraform must initialize the provider before it can be used. Initialization downloads and installs the provider's plugin so that it can later be executed.

Question 4

Question Type: MultipleChoice

When configuring a remote backend in Terraform, it might be a good idea to purposely omit some of the required arguments to ensure secrets and other relevant data are not inadvertently shared with others. What are the ways the remaining configuration can be added to Terraform so it can initialize and communicate with the backend? (select three)

Options:

A- directly querying HashiCorp Vault for the secrets

B- command-line key/value pairs

C- use the -backend-config=PATH to specify a separate config file

D- interactively on the command line

Answer:

B, C, D

Explanation:

You do not need to specify every required argument in the backend configuration. Omitting certain arguments may be desirable to avoid storing secrets, such as access keys, within the main configuration. When some or all of the arguments are omitted, we call this a partial configuration.

With a partial configuration, the remaining configuration arguments must be provided as part of the initialization process. There are several ways to supply the remaining arguments:

Interactively: Terraform will interactively ask you for the required values unless interactive input is disabled. Terraform will not prompt for optional values.

File: A configuration file may be specified via the init command line. To specify a file, use the -backend-config=PATH option when running terraform init. If the file contains secrets it may be kept in a secure data store, such as Vault, in which case it must be downloaded to the local disk before running Terraform.

Command-line key/value pairs: Key/value pairs can be specified via the init command line. Note that many shells retain command-line flags in a history file, so this isn't recommended for secrets. To specify a single key/value pair, use the -backend-config='KEY=VALUE' option when running terraform init.

Question 5

Question Type: MultipleChoice

Which Terraform command will force a marked resource to be destroyed and recreated on the next apply?

Options:

A- terraform fmt

B- terraform destroy

C- terraform taint

D- terraform refresh

Answer:

C

Explanation:

The terraform taint command manually marks a Terraform-managed resource as tainted, forcing it to be destroyed and recreated on the next apply. This command will not modify infrastructure but does modify the state file in order to mark a resource as tainted. Once a resource is marked as tainted, the next plan will show that the resource will be destroyed and recreated. The next terraform apply will

implement this change.

Question 6

Question Type: MultipleChoice

In order to make a Terraform configuration file dynamic and/or reusable, static values should be converted to use what?

Options:

A- regular expressions

B- module

C- input parameters

D- output value

Answer:

C

Explanation:

Input variables serve as parameters for a Terraform module, allowing aspects of the module to be customized without altering the module's own source code, and allowing modules to be shared between different configurations.

Question 7

Question Type: **MultipleChoice**

What is a downside to using a Terraform provider, such as the Vault provider, to interact with sensitive data, such as reading secrets from Vault?

Options:

A- Terraform and Vault must be running on the same physical host

B- Terraform and Vault must be running on the same version

C- Terraform requires a unique auth method to work with Vault

D- Secrets are persisted to the state file and plans

Answer:

D

Explanation:

Interacting with Vault from Terraform causes any secrets that you read and write to be persisted in both Terraform's state file and in any generated plan files. For any Terraform module that reads or writes Vault secrets, these files should be treated as sensitive and protected accordingly.