Free Questions for Terraform-Associate-003 by certsinside

Shared by Parrish on 15-04-2024

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

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.

Options:

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:

B, E

Explanation:

Infrastructure as code (IaC) is a way of managing and provisioning cloud infrastructure using programming techniques instead of manual processes1. IaC has many advantages over using a graphical user interface (GUI) for provisioning infrastructure, such as:

* Versioning: IaC allows you to store your infrastructure configuration in a version control system, such as Git, and track changes over time. This enables you to roll back to previous versions, compare differences, and collaborate with other developers2.

* Reusability: IaC allows you to create reusable modules and templates that can be applied to different environments, such as development, testing, and production. This reduces duplication, improves consistency, and speeds up deployment3.

* Sharing: IaC allows you to share your infrastructure configuration with other developers, teams, or organizations, and leverage existing code from open source repositories or registries. This fosters best practices, innovation, and standardization4.

* Risk reduction: IaC reduces the risk of human error, configuration drift, and security breaches that can occur when provisioning infrastructure manually or using a GUI. IaC also enables you to perform automated testing, validation, and compliance checks on your infrastructure before deploying it5.

Reference =

* 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

Question 2

Question Type: MultipleChoice

Which of the following is not true of Terraform providers?

Options:

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

Explanation:

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. Reference =

* 1: Providers - Configuration Language | Terraform | HashiCorp Developer

* 2: Plugin Development - How Terraform Works With Plugins | Terraform | HashiCorp Developer

* 3: Terraform Registry

* : Terraform Registry

Question 3

Question Type: **MultipleChoice**

The Terraform binary version and provider versions must match each other in a single configuration.

Options:

A- True

B- False

Answer:

B

Explanation:

The Terraform binary version and provider versions do not have to match each other in a single configuration. Terraform allows you to specify provider version constraints in the configuration's terraform block, which can be different from the Terraform binary version1. Terraform will use the newest version of the provider that meets the configuration's version constraints2. You can also use the dependency lock file to ensure Terraform is using the correct provider version3. Reference =

* 1: Providers - Configuration Language | Terraform | HashiCorp Developer

* 2: Multiple provider versions with Terraform - Stack Overflow

* 3: Lock and upgrade provider versions | Terraform - HashiCorp Developer

Question 4

Question Type: **MultipleChoice**

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

Options:

A- Terraform can only source providers from the internet

B- Every provider in a configuration has its own state file for its resources

C- You can create a provider for your API if none exists

D- All providers are part of the Terraform core binary

Answer:

C

Explanation:

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

* 1: Plugin Development - How Terraform Works With Plugins | Terraform | HashiCorp Developer

* 2: Lab: Terraform Plug-in Based Architecture - GitHub

* 3: Terraform Plugin SDK - Terraform by HashiCorp

* 4: HashiCorp Terraform Plugin Framework Now Generally Available

Question 5

Question Type: **MultipleChoice**

Terraform can only manage resource dependencies if you set them explicitly with the depends_on argument.

Options:

A- True

B- False

Answer:

B

Explanation:

Terraform can manage resource dependencies implicitly or explicitly. Implicit dependencies are created when a resource references another resource or data source in its arguments. Terraform can infer the dependency from the reference and create or destroy the resources in the correct order. Explicit dependencies are created when you use thedepends_onargument to specify that a resource depends on another resource or module. This is useful when Terraform cannot infer the dependency from the configuration or when you need to create a dependency for some reason outside of Terraform's scope.Reference= :Create resource dependencies:Terraform Resource Dependencies Explained

Question 6

Question Type: MultipleChoice

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

Options:

A- True

B- False

Answer:

A

Explanation:

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