2022 Easy Success HashiCorp TA-002-P Exam in First Try [Q78-Q92

2022 Easy Success HashiCorp TA-002-P Exam in First Try Best TA-002-P Exam Dumps for the Preparation of Latest Exam Questions

How to schedule HashiCorp Certified: Terraform Associate TA-002-P Professional Exam

To apply for the HashiCorp Certified: Terraform Associate TA-002-P Professional Exam, You have to follow these steps:
? Step 1: Go to the HashiCorp Certified: Terraform Associate TA-002-P Professional Official Site. You must first create an account, use your email address to register. You must purchase your training through your local distributor. If you are a partner, you must first create an account on the Partner Portal. You must use your company email address to register.
? Step 2: Read the instruction Carefully
? Step 3: Follow the given steps
? Step 4: Apply for the HashiCorp Certified: Terraform Associate TA-002-P-Professional Exam

How to Prepare for HashiCorp Certified: Terraform Associate TA-002-P Professional Exam **Preparation Guide for HashiCorp Certified: Terraform Associate TA-002-P Professional Exam Introduction for HashiCorp Certified: Terraform Associate TA-002-P Professional Exam**

This guide provides a step by step framework of the HashiCorp Certified: Terraform Associate TA-002-P Professional course exam including a broad array of essentials of the test, the exam design, themes, test complexities and readiness techniques, and the intended interest group profile. Thus, we prepare various HashiCorp Certified: Terraform Associate TA-002-P PROFESSIONAL exam dumps as we understand understudy determinations. Our content, helps candidates' total assessments.
Cloud engineers can utilize the Terraform Associate exam from HashiCorp to check their basic infrastructure automation abilities. Terraform associate is a foundational level of certification that assesses an individual's knowledge of fundamental concepts and skills on Terraform OSS and the characteristics that exist on Terraform Cloud & Terraform Enterprise packages.
The Terraform Associate certification is for Cloud Engineers having experience in operations, IT, who recognize the fundamental concepts and abilities linked with open-source HashiCorp Terraform. Applicants will be best equipped for this exam if they have professional expertise using Terraform in production, but performing the exam in a personal demo atmosphere may also be adequate. This individual knows which enterprise features exist and what can and cannot be made using the open-source edition.
After finishing this course, the candidate will be able to:

- Use Terraform Cloud interactively with Katacoda- Be familiar with Terraform with HashiCorp's official learning platform- Understand the Terraform Core workflow

The contents of **HASHICORP TA-002 practice exam** and **HASHICORP TA-002 practice exams** will help candidates to prepare for this exam.

QUESTION 78

Which of the following connection types are supported by the remote-exec provisioner? (select two)
* WinRM
* UDP
* SMB
* RDP
* ssh

Explanation

The remote-exec provisioner invokes a script on a remote resource after it is created. The remote-exec

provisioner supports both ssh and winrm type connections.

remote-exec connection types –

* ssh on Linux

* winrm on Windows

https://www.terraform.io/docs/provisioners/remote-exec.html

QUESTION 79

Provider dependencies are created in several different ways. Select the valid provider dependencies from the following list: (select three)
* Explicit use of a provider block in configuration, optionally including a version constraint.
* Use of any resource belonging to a particular provider in a resource or data block in configuration.
* Existence of any resource instance belonging to a particular provider in the current state.
* Existence of any provider plugins found locally in the working directory.

Explanation

The existence of a provider plugin found locally in the working directory does not itself create a provider dependency. The plugin can exist without any reference to it in the terraform configuration. https://www.terraform.io/docs/commands/providers.html

QUESTION 80

Refer below code where pessimistic constraint operator has been used to specify a version of a provider.

terraform { required_providers { aws = “~> 1.1.0” }}

Which of the following options are valid provider versions that satisfy the above constraint. (select two)
* 1.1.1
* 1.2.9
* 1.1.8
* 1.2.0

Pessimistic constraint operator, constraining both the oldest and newest version allowed. For example, ~> 0.9 is equivalent to >= 0.9, < 1.0, and ~> 0.8.4, is equivalent to >= 0.8.4, < 0.9

QUESTION 81

terraform refresh command will not modify infrastructure, but does modify the state file.
* True
* False

The terraform refresh command is used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure. This can be used to detect any drift from the last-known state, and to update the state file. This does not modify infrastructure, but does modify the state file.

https://www.terraform.io/docs/commands/refresh.html

QUESTION 82

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

QUESTION 83

Terraform Cloud always encrypts state at rest and protects it with TLS in transit. Terraform Cloud also knows

the identity of the user requesting state and maintains a history of state changes.
* False
* True

Explanation

Terraform Cloud always encrypts state at rest and protects it with TLS in transit. Terraform Cloud also knows

the identity of the user requesting state and maintains a history of state changes. This can be used to control

access and track activity. Terraform Enterprise also supports detailed audit logging.

https://www.terraform.io/docs/state/sensitive-data.html#recommendations

QUESTION 84

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.)
* Provision new VMs using Terraform with the same VM names
* Use the terraform import command for the existing VMs
* Write Terraform configuration for the existing VMs
* Run the terraform import-gcp command

The terraform import command is used to import existing infrastructure. Import existing Google Cloud resources into Terraform with Terraformer.

QUESTION 85

What information does the public Terraform Module Registry automatically expose about published modules?
* Required input variables
* Optional inputs variables and default values
* Outputs
* All of the above
* None of the above

Reference: https://www.terraform.io/docs/registry/modules/publish.html

QUESTION 86

In Terraform 0.13 and above, outside of the required_providers block, Terraform configurations always refer to providers by their local names.
* True
* False

Explanation

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

Reference: https://www.terraform.io/docs/language/providers/requirements.html

QUESTION 87

terraform destroy is the only way to remove infrastructure.
* True
* False

QUESTION 88

Every region in AWS has a different AMI ID for Linux and these are keep on changing. What is the best approach to create the EC2 instances that can deal with different AMI IDs based on regions?
* Use data source aws_ami.
* Create a map of region to ami id.
* Create different configuration file for different region.
* None of the above

Explanation

https://www.terraform.io/docs/configuration/data-sources.html

QUESTION 89

Which of the following is not a valid string function in Terraform?
* split
* join
* slice
* chomp
Reference: https://www.terraform.io/docs/language/functions/chomp.html

QUESTION 90

When using a module block to reference a module stored on the public Terraform Module Registry such as:

```
module "consul" {
  source = "hashicorp/consul/aws"
}
```

How do you specify version 1.0.0?
* Modules stored on the public Terraform Module Registry do not support versioning
* Append ?ref=v1.0.0 argument to the source path
* Add version = “1.0.0” attribute to module block
* Nothing – modules stored on the public Terraform Module Registry always default to version 1.0.0
Explanation

Version

When using modules installed from a module registry, we recommend explicitly constraining the acceptable

version numbers to avoid unexpected or unwanted changes.

Use the version argument in the module block to specify versions:

module “consul” {

source = “hashicorp/consul/aws”

version = “0.0.5”

servers = 3

}

Reference: https://www.terraform.io/docs/language/modules/sources.html

QUESTION 91

You have used Terraform to create an ephemeral development environment in the cloud and are now ready to destroy all the infrastructure described by your Terraform configuration. To be safe, you would like to first see all the infrastructure that will be deleted by Terraform.

Which command should you use to show all of the resources that will be deleted? (Choose two.)
* Run terraform plan -destroy.
* This is not possible. You can only show resources that will be created.
* Run terraform state rm *.
* Run terraform destroy and it will first output all the resources that will be deleted before prompting for approval.
Explanation/Reference: https://www.terraform.io/docs/cli/commands/state/rm.html

QUESTION 92

Talal is a DevOps engineer and he has deployed the production infrastructure using Terraform. He is using a very large configuration file to maintain and update the actual infrastructure. As the infrastructure have grown to a very complex and large, he has started experiencing slowness when he run runs terraform plan. What are the options for him to resolve this slowness?
* Use -refresh=true flag as well as the -target flag with terraform plan in order to work around this.
* Run terraform refresh every time before running terraform plan.
* Break large configurations into several smaller configurations that can each be independently applied.
* Use -refresh=false flag as well as the -target flag with terraform plan in order to work around this.
Explanation

For larger infrastructures, querying every resource is too slow. Many cloud providers do not provide APIs to query multiple resources at once, and the round trip time for each resource is hundreds of milliseconds. On top of this, cloud providers almost always have API rate limiting so Terraform can only request a certain number of resources in a period of time. Larger users of Terraform make heavy use of the -refresh=false flag as well as the -target flag in order to work around this. In these scenarios, the cached state is treated as the record of truth.

Although ‘Use -refresh=false flag as well as the -target flag with terraform plan in order to work around this.’ is a solution, but its not always recommended. Instead of using -target as a means to operate on isolated portions of very large configurations, prefer instead to break large configurations into several smaller configurations that can each be independently applied. Data sources can be used to access information about resources created in other configurations, allowing a complex system architecture to be broken down into more manageable parts that can be updated independently.

Option ‘Run terraform refresh every time before running terraform plan.’ and ‘Use -refresh=true flag as well as the -target flag with terraform plan in order to work around this.’ is not correct because in both the cases terraform will query every resources of the infrastructure.

TA-002-P Study Material, Preparation Guide and PDF Download:

https://www.examslabs.com/HashiCorp/HashiCorp-Infrastructure-Automation/best-TA-002-P-exam-dumps.html]