TA-002-P Q&As

HashiCorp Certified: Terraform Associate

Pass HashiCorp TA-002-P Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.passapply.com/ta-002-p.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by HashiCorp Official Exam Center

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

By default, provisioners that fail will also cause the Terraform apply itself to error. How can you change this default behavior within a provisioner?

A. provisioner "local-exec" { on_failure = "next" }

B. provisioner "local-exec" { when = "failure" terraform apply }

C. provisioner "local-exec" { on_failure = "continue" }

D. provisioner "local-exec" { on_failure = continue }

Correct Answer: C

https://www.terraform.io/docs/provisioners/index.html

QUESTION 2

Your company has a lot of workloads in AWS , and Azure that were respectively created using CloudFormation , and AzureRM Templates. However , now your CIO has decided to use Terraform for all new projects , and has asked you to

check how to integrate the existing environment with terraform code. What should be your next plan of action?

A. Tell the CIO that this is not possible . Resources created in CloudFormation , and AzureRM templates cannot be tracked using terraform.

B. Use terraform import command to import each resource one by one .

C. This is only possible in Terraform Enterprise , which has the TerraformConverter exe that can take any other template language like AzureRM and convert to Terraform code.

D. Just write the terraform config file for the new resources , and run terraform apply , the state file will automatically be updated with the details of the new resources to be imported.

Correct Answer: B

QUESTION 3

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

A. list

B. map

C. tree

D. set

Correct Answer: C

https://www.terraform.io/language/expressions/type-constraints#collection-types

QUESTION 4

When using multiple configurations of the same Terraform provider, what meta-argument must be included in any non-default provider configurations?

A. name

B. alias

C. depends_on

D. id

Correct Answer: B

QUESTION 5

Terraform import command can import resources into modules as well directly into the root of your state.

A. True

B. False

Correct Answer: A

Import will find the existing resource from ID and import it into your Terraform state at the given ADDRESS. ADDRESS must be a valid resource address. Because any resource address is valid, the import command can import resources into

modules as well directly into the root of your state.

Terraform is able to import existing infrastructure. This allows us take resources we\\'ve created by some other means (i.e. via console) and bring it under Terraform management. This is a great way to slowly transition infrastructure to

Terraform. The terraform import command is used to import existing infrastructure. To import a resource, first write a resource block for it in our configuration, establishing the name by which it will be known to Terraform. For example:

resource "aws_instance" "import_example" {

...instance configuration...

}

Now terraform import can be run to attach an existing instance to this resource configuration:

$ terraform import aws_instance.import_example i-03efafa258104165f aws_instance.import_example: Importing from ID "i-03efafa258104165f"...

aws_instance.import_example: Import complete!

Imported aws_instance (ID: i-03efafa258104165f)

aws_instance.import_example: Refreshing state... (ID: i-03efafa258104165f) Import successful!

The resources that were imported are shown above. These resources are now in your Terraform state and will henceforth be managed by Terraform. This command locates the AWS instance with ID i-03efafa258104165f (which has been

created outside Terraform) and attaches its existing settings, as described by the EC2 API, to the name aws_instance.import_example in the Terraform state. As a result of the above command, the resource is recorded in the state file. We

can now run terraform plan to see how the configuration compares to the imported resource, and make any adjustments to the configuration to align with the current (or desired) state of the imported object.

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