Vendor: HashiCorp

Exam Code: Terraform-Associate-003

Exam Name: HashiCorp Certified: Terraform Associate (003)

Version: DEMO

QUESTION 1
What does Terraform not reference when running a terraform apply -refresh-only ?

A. State file
B. Credentials
C. Cloud provider
D. Terraform resource definitions in configuration files

Answer: D
Explanation:
When running a terraform apply -refresh-only, Terraform does not reference the configuration files, but only the state file, credentials, and cloud provider. The purpose of this command is to update the state file with the current status of the real resources, without making any changes to them.

QUESTION 2
Which of the following is not a valid Terraform variable type?

A. list
B. array
C. nap
D. string

Answer: B
Explanation:
This is not a valid Terraform variable type. The other options are valid variable types that can store different kinds of values.

QUESTION 3
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.

```
terraform {
  backend "s3" {
    bucket - "my-tf-bucket"
    region = "us-east-1"
  }
}
```

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: B
Explanation:
This command will initialize the new backend and prompt you to migrate the existing state file to

the new location. The other commands are not relevant for this task.

QUESTION 4
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: D
Explanation:
This is not a valid backend for Terraform, as it does not support locking or versioning of state files.The other options are valid backends that can store state files in a central location.

QUESTION 5
You have deployed a new webapp with a public IP address on a cloud provider. However, you did not create any outputs for your code. What is the best method to quickly find the IP address of the resource you deployed?

A. In a new folder, use the terraform_remote_state data source to load in the state file, then write an output for each resource that you find the state file
B. Run terraform state list to find the name of the resource, then terraform state show to find the attributes including public IP address
C. Run terraform output ip_address to view the result
D. Run terraform destroy then terraform apply and look for the IP address in stdout

Answer: B
Explanation:
This is a quick way to inspect the state file and find the information you need without modifying anything. The other options are either incorrect or inefficient.

QUESTION 6
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: C
Explanation:
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.

QUESTION 7

What information does the public Terraform Module Registry automatically expose about published modules?

A. Required input variables
B. Optional inputs variables and default values
C. Outputs
D. All of the above
E. None of the above

Answer: D
Explanation:
The public Terraform Module Registry automatically exposes all the information about published modules, including required input variables, optional input variables and default values, and outputs. This helps users to understand how to use and configure the modules.

QUESTION 8
You must use different Terraform commands depending on the cloud provider you use.

A. True
B. False

Answer: B
Explanation:
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.

QUESTION 9
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: E
Explanation:
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.

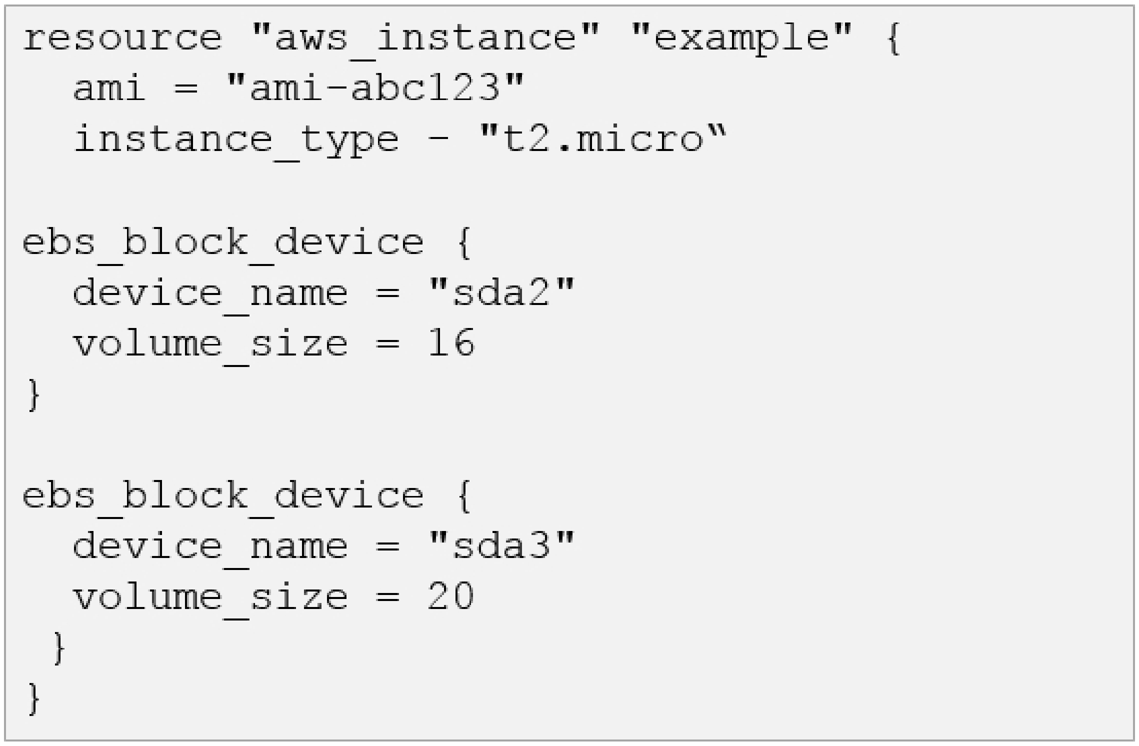
QUESTION 10
You ate making changes to existing Terraform code to add some new infrastructure. When is the best time to run terraform validate?

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: C
Explanation:
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.

QUESTION 11
How would you reference the volume IDs associated with the ebs_block_device blocks in this configuration?



```
resource "aws_instance" "example" {
  ami = "ami-abc123"
  instance_type - "t2.micro"

ebs_block_device {
  device_name = "sda2"
  volume_size = 16
}

ebs_block_device {
  device_name = "sda3"
  volume_size = 20
 }
}
```

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
Explanation:
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.

QUESTION 12
Which command should you run to check if all code in a Terraform configuration that references

multiple modules is properly formatted without making changes?

A. terraform fmt -write-false
B. terraform fmt -list -recursive
C. terraform fmt -check -recursive
D. terraform fmt -check

Answer: C
Explanation:
This command will check if all code in a Terraform configuration that references multiple modules is properly formatted without making changes, and will return a non-zero exit code if any files need formatting. The other commands will either make changes, list the files that need formatting, or not check the modules.

QUESTION 13
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: D
Explanation:
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.

QUESTION 14
What is the Terraform style convention for indenting a nesting level compared to the one above it?

A. With a tab
B. With two spaces
C. With four spaces
D. With three spaces

Answer: B
Explanation:
This is the Terraform style convention for indenting a nesting level compared to the one above it. The other options are not consistent with the Terraform style guide.

QUESTION 15
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
Explanation:
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.

QUESTION 16
When do changes invoked by terraform apply take effect?

A. After Terraform has updated the state file
B. Once the resource provider has fulfilled the request
C. Immediately
D. None of the above are correct

Answer: B
Explanation:
Changes invoked by terraform apply take effect once the resource provider has fulfilled the request, not after Terraform has updated the state file or immediately. The state file is only a reflection of the real resources, not a source of truth.

QUESTION 17
What is the workflow for deploying new infrastructure with Terraform?

A. Write Terraform configuration, run terraform init to initialize the working directory or workspace, and run terraform apply
B. Write Terraform configuration, run terraform show to view proposed changes, and terraform apply to create new infrastructure
C. Write Terraform configuration, run terraform apply to create infrastructure, use terraform validate to confirm Terraform deployed resources correctly
D. Write Terraform configuration, run terraform plan to initialize the working directory or workspace, and terraform apply to create the infrastructure

Answer: A
Explanation:
This is the workflow for deploying new infrastructure with Terraform, as it will create a plan and apply it to the target environment. The other options are either incorrect or incomplete.

QUESTION 18
You can develop a custom provider to manage its resources using Terraform.

A. True
B. False

Answer: A
Explanation:
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.

QUESTION 19
What is one disadvantage of using dynamic blocks in Terraform?

A. Dynamic blocks can construct repeatable nested blocks
B. Terraform will run more slowly
C. They cannot be used to loop through a list of values
D. They make configuration harder to read and understand

Answer: D
Explanation:
This is one disadvantage of using dynamic blocks in Terraform, as they can introduce complexity and reduce readability of the configuration. The other options are either advantages or incorrect statements.

QUESTION 20
Which backend does the Terraform CU use by default?

A. Depends on the cloud provider configured
B. HTTP
C. Remote
D. Terraform Cloud
E. Local

Answer: E
Explanation:
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.

QUESTION 21
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: A
Explanation:
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.