

## Terraform-Associate-003 Dumps

### HashiCorp Certified: Terraform Associate (003)

<https://www.certleader.com/Terraform-Associate-003-dumps.html>



**NEW QUESTION 1**

When does Terraform create the .terraform.lock.hcl file?

- A. After your first terraform plan
- B. After your first terraform apply
- C. After your first terraform init
- D. When you enable state locking

**Answer:** C

**Explanation:**

Terraform creates the .terraform.lock.hcl file after the first terraform init command. This lock file ensures that the dependencies for your project are consistent across different runs by locking the versions of the providers and modules used.

**NEW QUESTION 2**

You've used Terraform to deploy a virtual machine and a database. You want to replace this virtual machine instance with an identical one without affecting the database. What is the best way to achieve this using Terraform?

- A. Use the terraform state rm command to remove the VM from state file
- B. Use the terraform taint command targeting the VMs then run terraform plan and terraform apply
- C. Use the terraform apply command targeting the VM resources only
- D. Delete the Terraform VM resources from your Terraform code then run terraform plan and terraform apply

**Answer:** B

**Explanation:**

The terraform taint command marks a resource as tainted, which means it will be destroyed and recreated on the next apply. This way, you can replace the VM instance without affecting the database or other resources. References = [Terraform Taint]

**NEW QUESTION 3**

HashiCorp Configuration Language (HCL) supports user-defined functions.

- A. True
- B. False

**Answer:** B

**Explanation:**

HashiCorp Configuration Language (HCL) does not support user-defined functions. You can only use the built-in functions that are provided by the language. The built-in functions allow you to perform various operations and transformations on values within expressions. The general syntax for function calls is a function name followed by comma-separated arguments in parentheses, such as max(5, 12, 9). You can find the documentation for all of the available built-in functions in the Terraform Registry or the Packer Documentation, depending on which tool you are using. References = : Functions - Configuration Language | Terraform : Functions - Configuration Language | Packer

**NEW QUESTION 4**

Why would you use the -replace flag for terraform apply?

- A. You want Terraform to ignore a resource on the next apply
- B. You want Terraform to destroy all the infrastructure in your workspace
- C. You want to force Terraform to destroy a resource on the next apply
- D. You want to force Terraform to destroy and recreate a resource on the next apply

**Answer:** D

**Explanation:**

The -replace flag is used with the terraform apply command when there is a need to explicitly force Terraform to destroy and then recreate a specific resource during the next apply. This can be necessary in situations where a simple update is insufficient or when a resource must be re-provisioned to pick up certain changes.

**NEW QUESTION 5**

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

- 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 version<sup>1</sup>. Terraform will use the newest version of the provider that meets the configuration's version constraints<sup>2</sup>. You can also use the dependency lock file to ensure Terraform is using the correct provider version<sup>3</sup>.

References =

- 1: Providers - Configuration Language | Terraform | HashiCorp Developer
- 2: Multiple provider versions with Terraform - Stack Overflow
- 3: Lock and upgrade provider versions | Terraform - HashiCorp Developer

**NEW QUESTION 6**

You add a new resource to an existing Terraform configuration, but do not update the version constraint in the configuration. The existing and new resources use the same provider. The working contains a .terraform.lock, hc1 file.

How will Terraform choose which version of the provider to use?

- A. Terraform will use the version recorded in your lock file
- B. Terraform will use the latest version of the provider for the new resource and the version recorded in the lock file to manage existing resources
- C. Terraform will check your state file to determine the provider version to use
- D. Terraform will use the latest version of the provider available at the time you provision your new resource

**Answer:** A

**Explanation:**

This is how Terraform chooses which version of the provider to use, when you add a new resource to an existing Terraform configuration, but do not update the version constraint in the configuration. The lock file records the exact version of each provider that was installed in your working directory, and ensures that Terraform will always use the same provider versions until you run terraform init -upgrade to update them.

**NEW QUESTION 7**

What feature stops multiple users from operating on the Terraform state at the same time?

- A. State locking
- B. Version control
- C. Provider constraints
- D. Remote backends

**Answer:** A

**Explanation:**

State locking prevents other users from modifying the state file while a Terraform operation is in progress. This prevents conflicts and data loss1.

**NEW QUESTION 8**

A module can always refer to all variables declared in its parent module.

- A. True
- B. False

**Answer:** B

**Explanation:**

A module cannot always refer to all variables declared in its parent module, as it needs to explicitly declare input variables and assign values to them from the parent module??s arguments. A module cannot access the parent module??s variables directly, unless they are passed as input arguments.

**NEW QUESTION 9**

Running terraform fmt without any flags in a directory with Terraform configuration files check the formatting of those files without changing their contents.

- A. True
- B. False

**Answer:** B

**Explanation:**

Running terraform fmt without any flags in a directory with Terraform configuration files will not check the formatting of those files without changing their contents, but will actually rewrite them to a canonical format and style. If you want to check the formatting without making changes, you need to use the -check flag.

**NEW QUESTION 10**

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.

**NEW QUESTION 10**

You want to know from which paths Terraform is loading providers referenced in your Terraform configuration (\* files). You need to enable additional logging messages to find this out. Which of the following would achieve this?

- A. Set verbose for each provider in your Terraform configuration
- B. Set the environment variable TF\_LOG\_TRACE
- C. Set the environment variable TF\_LOG\_PATH
- D. Set the environment variable TF\_log\_TRACE

**Answer:** B

**Explanation:**

This will enable additional logging messages to find out from which paths Terraform is loading providers referenced in your Terraform configuration files, as it will set the log level to TRACE, which is the most verbose and detailed level.

**NEW QUESTION 12**

You add a new provider to your configuration and immediately run terraform apply in the CD using the local backend. Why does the apply fail?

- A. The Terraform CD needs you to log into Terraform Cloud first
- B. Terraform requires you to manually run terraform plan first
- C. Terraform needs to install the necessary plugins first
- D. Terraform needs you to format your code according to best practices first

**Answer:** C

**Explanation:**

The reason why the apply fails after adding a new provider to the configuration and immediately running terraform apply in the CD using the local backend is because Terraform needs to install the necessary plugins first. Terraform providers are plugins that Terraform uses to interact with various cloud services and other APIs. Each provider has a source address that determines where to download it from. When Terraform encounters a new provider in the configuration, it needs to run terraform init first to install the provider plugins in a local directory. Without the plugins, Terraform cannot communicate with the provider and perform the desired actions. References = [Provider Requirements], [Provider Installation]

**NEW QUESTION 15**

In a Terraform Cloud workspace linked to a version control repository speculative plan run start automatically commit changes to version control.

- A. True
- B. False

**Answer:** A

**Explanation:**

When you use a remote backend that needs authentication, HashiCorp recommends that you:

**NEW QUESTION 20**

How could you reference an attribute from the vsphere\_datacenter data source for use with the datacenter\_id argument within the vsphere\_folder resource in the following configuration?

```
data "vsphere_datacenter" "dc" {}

resource "vsphere_folder" "parent" {
  path = "Production"
  type = "vm"
  datacenter_id = _____
}
```

- A. Data.vsphere\_datacenter.DC.id
- B. Vsphere\_datacenter.dc.id
- C. Data,dc,id
- D. Data.vsphere\_datacenter,dc

**Answer:** A

**Explanation:**

The correct way to reference an attribute from the vsphere\_datacenter data source for use with the datacenter\_id argument within the vsphere\_folder resource in the following configuration is data.vsphere\_datacenter.dc.id. This follows the syntax for accessing data source attributes, which is data.TYPE.NAME.ATTRIBUTE. In this case, the data source type is vsphere\_datacenter, the data source name is dc, and the attribute we want to access is id. The other options are incorrect because they either use the wrong syntax, the wrong punctuation, or the wrong case. References = [Data Source: vsphere\_datacenter], [Data Source: vsphere\_folder], [Expressions: Data Source References]

**NEW QUESTION 22**

You are writing a child Terraform module that provisions an AWS instance. You want to reference the IP address returned by the child module in the root configuration. You name the instance resource "main".

Which of these is the correct way to define the output value?

- A)

```
output "instance_ip_addr" {  
    return aws_instance.main.private_ip  
}
```

B)

```
output "aws_instance.instance_ip_addr" {  
    return aws_instance.main.private_ip  
}
```

C)

```
output "aws_instance.instance_ip_addr" {  
    value = ${main.private_ip}  
}
```

D)

```
output "instance_ip_addr" {  
    value = aws_instance.main.private_ip  
}
```

- A. Option A
- B. Option B
- C. Option C
- D. Option D

**Answer:** D**NEW QUESTION 24**

Module variable assignments are inherited from the parent module and you do not need to explicitly set them.

- A. True
- B. False

**Answer:** B**Explanation:**

Module variable assignments are not inherited from the parent module and you need to explicitly set them using the source argument. This allows you to customize the behavior of each module instance.

**NEW QUESTION 25**

You're building a CI/CD (continuous integration/continuous delivery) pipeline and need to inject sensitive variables into your Terraform run. How can you do this safely?

- A. Copy the sensitive variables into your Terraform code
- B. Store the sensitive variables in a secure\_varS.tf file
- C. Store the sensitive variables as plain text in a source code repository
- D. Pass variables to Terraform with a -var flag

**Answer:** D**Explanation:**



This is a secure way to inject sensitive variables into your Terraform run, as they will not be stored in any file or source code repository. You can also use environment variables or variable files with encryption to pass sensitive variables to Terraform.

**NEW QUESTION 30**

Which are forbidden actions when the terraform state file is locked? Choose three correct answers.

- A. Terraform state list
- B. Terraform destroy
- C. Terraform validate
- D. Terraform validate
- E. Terraform for
- F. Terraform apply

**Answer:** BCF

**Explanation:**

The terraform state file is locked when a Terraform operation that could write state is in progress. This prevents concurrent state operations that could corrupt the state.

The forbidden actions when the state file is locked are those that could write state, such as terraform apply, terraform destroy, terraform refresh, terraform taint, terraform

untaint, terraform import, and terraform state \*. The terraform validate command is also forbidden, because it requires an initialized working directory with the state file. The allowed actions when the state file is locked are those that only read state, such as terraform plan, terraform show, terraform output, and terraform console. References = [State Locking] and [Command: validate]

**NEW QUESTION 34**

Once you configure a new Terraform backend with a terraform code block, which command(s) should you use to migrate the state file?

- A. terraform destroy, then terraform apply
- B. terraform init
- C. terraform push
- D. terraform apply

**Answer:** A

**Explanation:**

This command will initialize the new backend and prompt you to migrate the existing state file to the new location<sup>4</sup>. The other commands are not relevant for this task.

**NEW QUESTION 37**

Define the purpose of state in Terraform.

- A. State maps real world resources to your configuration and keeps track of metadata
- B. State lets you enforce resource configurations that relate to compliance policies
- C. State stores variables and lets you quickly reuse existing code
- D. State codifies the dependencies of related resources

**Answer:** A

**Explanation:**

The purpose of state in Terraform is to keep track of the real-world resources managed by Terraform, mapping them to the configuration. The state file contains metadata about these resources, such as resource IDs and other important attributes, which Terraform uses to plan and manage infrastructure changes. The state enables Terraform to know what resources are managed by which configurations and helps in maintaining the desired state of the infrastructure. References = This role of state in Terraform is outlined in Terraform's official documentation, emphasizing its function in mapping configuration to real-world resources and storing vital metadata .

**NEW QUESTION 39**

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 Terraform will delete.

Which command should you use to show all of the resources that will be deleted? Choose two correct answers.

- A. Run terraform state rm ??
- B. Run terraform show :destroy
- C. Run terraform destroy and it will first output all the resource that will be deleted before prompting for approval
- D. Run terraform plan -destroy

**Answer:** CD

**Explanation:**

To see all the resources that Terraform will delete, you can use either of these two commands:

? terraform destroy will show the plan of destruction and ask for your confirmation

before proceeding. You can cancel the command if you do not want to destroy the resources.

? terraform plan -destroy will show the plan of destruction without asking for

confirmation. You can use this command to review the changes before

running terraform destroy. References = : Destroy Infrastructure : Plan Command: Options

**NEW QUESTION 42**

You can configure Terraform to log to a file using the TF\_LOG environment variable.

- A. True
- B. False

**Answer:** A

**Explanation:**

You can configure Terraform to log to a file using the TF\_LOG environment variable. This variable can be set to one of the log levels: TRACE, DEBUG, INFO, WARN or ERROR. You can also use the TF\_LOG\_PATH environment variable to specify a custom log file location. References = : Debugging Terraform

**NEW QUESTION 47**

You are working on some new application features and you want to spin up a copy of your production deployment to perform some quick tests. In order to avoid having to configure a new state backend, what open source Terraform feature would allow you create multiple states but still be associated with your current code?

- A. Terraform data sources
- B. Terraform local values
- C. Terraform modules
- D. Terraform workspaces
- E. None of the above

**Answer:** D

**Explanation:**

Terraform workspaces allow you to create multiple states but still be associated with your current code. Workspaces are like ??environments?? (e.g. staging, production) for the same configuration. You can use workspaces to spin up a copy of your production deployment for testing purposes without having to configure a new state backend. Terraform data sources, local values, and modules are not features that allow you to create multiple states. References = Workspaces and How to Use Terraform Workspaces

**NEW QUESTION 49**

How can you trigger a run in a Terraform Cloud workspace that is connected to a Version Control System (VCS) repository?

- A. Only Terraform Cloud organization owners can set workspace variables on VCS connected workspaces
- B. Commit a change to the VCS working directory and branch that the Terraform Cloud workspace is connected to
- C. Only Terraform Cloud organization owners can approve plans in VCS connected workspaces
- D. Only members of a VCS organization can open a pull request against repositories that are connected to Terraform Cloud workspaces

**Answer:** B

**Explanation:**

This will trigger a run in the Terraform Cloud workspace, which will perform a plan and apply operation on the infrastructure defined by the Terraform configuration files in the VCS repository.

**NEW QUESTION 52**

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 correct answers.

- A. Run the terraform Import-gcp command
- B. Write Terraform configuration for the existing VMs
- C. Use the terraform import command for the existing VMs
- D. Provision new VMs using Terraform with the same VM names

**Answer:** BC

**Explanation:**

To import existing resources into Terraform, you need to do two things1:

? Write a resource configuration block for each resource, matching the type and name used in your state file.

? Run terraform import for each resource, specifying its address and ID. There is no such command as terraform Import-gcp, and provisioning new VMs with the same names will not import them into Terraform.

**NEW QUESTION 57**

You ate creating a Terraform configuration which needs to make use of multiple providers, one for AWS and one for Datadog. Which of the following provider blocks would allow you to do this?

- A)

```
terraform {  
  provider "aws" {  
    profile = var.aws_profile  
    region  = var.aws_region  
  }  
  
  provider "datadog" {  
    api_key = var.datadog_api_key  
    app_key = var.datadog_app_key  
  }  
}
```

B)

```
provider "aws" {  
  profile = var.aws_profile  
  region  = var.aws_region  
}  
  
provider "datadog" {  
  api_key = var.datadog_api_key  
  app_key = var.datadog_app_key  
}
```

C)



```
provider "aws" {  
    profile = var.aws_profile  
    region  = var.aws_region  
}  
  
provider "datadog" {  
    api_key = var.datadog_api_key  
    app_key = var.datadog_app_key  
}
```

D)

```
provider {  
    "aws" {  
        profile = var.aws_profile  
        region  = var.aws_region  
    }  
  
    "datadog" {  
        api_key = var.datadog_api_key  
        app_key = var.datadog_app_key  
    }  
}
```

- A. Option A
- B. Option B
- C. Option C
- D. Option D

**Answer:** C**Explanation:**

Option C is the correct way to configure multiple providers in a Terraform configuration. Each provider block must have a name attribute that specifies which provider it configures<sup>2</sup>. The other options are either missing the name attribute or using an invalid syntax.

**NEW QUESTION 58**

You are 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.

**NEW QUESTION 59**

Variables declared within a module are accessible outside of the module.

- A. True
- B. False

**Answer:** B

**Explanation:**

Variables declared within a module are only accessible within that module, unless they are explicitly exposed as output values<sup>1</sup>.

**NEW QUESTION 64**

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.

**NEW QUESTION 68**

Terraform variable names are saved in the state file.

- A. True
- B. False

**Answer:** B

**Explanation:**

Terraform variable names are not saved in the state file, only their values are. The state file only stores the attributes of the resources and data sources that are managed by Terraform, not the variables that are used to configure them.

**NEW QUESTION 71**

When does Sentinel enforce policy logic during a Terraform Cloud run?

- A. Before the plan phase
- B. During the plan phase
- C. Before the apply phase
- D. After the apply phase

**Answer:** C

**Explanation:**

Sentinel policies are checked after the plan stage of a Terraform run, but before it can be confirmed or the terraform apply is executed<sup>3</sup>. This allows you to enforce rules on your infrastructure before it is created or modified.

**NEW QUESTION 76**

You have a list of numbers that represents the number of free CPU cores on each virtual cluster:

```
numcpus = [ 18, 3, 7, 11, 2 ]
```

What Terraform function could you use to select the largest number from the list?

- A. top(numcpus)
- B. max(numcpus)
- C. ceil (numcpus)

D. hight[numcpus]

**Answer:** B

**Explanation:**

In Terraform, the max function can be used to select the largest number from a list of numbers. The max function takes multiple arguments and returns the highest one. For the list numcpus = [18, 3, 7, 11, 2], using max(numcpus...) will return 18, which is the largest number in the list.

References:

? Terraform documentation on max function: Terraform Functions - max

**NEW QUESTION 78**

Why does this backend configuration not follow best practices?

```
terraform {  
  backend "s3" {  
    bucket      = "terraform-state-prod"  
    key         = "network/terraform.tfstate"  
    region      = "us-east-1"  
    access_key  = "AKIAIOSFODNN7EXAMPLE"  
    secret_key  = "wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY"  
  }  
  
  required_providers {  
    aws = {  
      source = "hashicorp/aws"  
      version = "~> 3.38"  
    }  
  }  
  
  required_version = ">= 0.15"  
}
```

- A. An alias meta-argument should be included in backend blocks whenever possible
- B. You should use the local enhanced storage backend whenever possible
- C. You should not store credentials in Terraform configuration
- D. The backend configuration should contain multiple credentials so that more than one user can execute terraform plan and terraform apply

**Answer:** C

**Explanation:**

This is a bad practice, as it exposes your credentials to anyone who can access your configuration files or state files. You should use environment variables, credential files, or other mechanisms to provide credentials to Terraform.

**NEW QUESTION 82**

Which parameters does terraform import require? Choose two correct answers.

- A. Provider
- B. Resource ID
- C. Resource address
- D. Path

**Answer:** BC

**Explanation:**

These are the parameters that terraform import requires, as they allow

Terraform to identify the existing resource that you want to import into your state file, and match it with the corresponding configuration block in your files.

**NEW QUESTION 85**

You have created a main.tf Terraform configuration consisting of an application server, a database and a load balanced. You ran terraform apply and Terraform created all of the resources successfully.

Now you realize that you do not actually need the load balancer, so you run terraform destroy without any flags. What will happen?

- A. Terraform will prompt you to pick which resource you want to destroy
- B. Terraform will destroy the application server because it is listed first in the code
- C. Terraform will prompt you to confirm that you want to destroy all the infrastructure
- D. Terraform will destroy the main, tf file
- E. Terraform will immediately destroy all the infrastructure

**Answer:** C

**Explanation:**

This is what will happen if you run terraform destroy without any flags, as it will attempt to delete all the resources that are associated with your current working directory or workspace. You can use the -target flag to specify a particular resource that you want to destroy.

**NEW QUESTION 90**

Which of the following is not a benefit of adopting infrastructure as code?

- A. Versioning
- B. A Graphical User Interface
- C. Reusability of code
- D. Automation

**Answer: B**

**Explanation:**

Infrastructure as Code (IaC) provides several benefits, including the ability to version control infrastructure, reuse code, and automate infrastructure management. However, IaC is typically associated with declarative configuration files and does not inherently provide a graphical user interface (GUI). A GUI is a feature that may be provided by specific tools or platforms built on top of IaC principles but is not a direct benefit of IaC itself<sup>1</sup>.

References = The benefits of IaC can be verified from the official HashiCorp documentation on ??What is Infrastructure as Code with Terraform??? provided by HashiCorp Developer<sup>1</sup>.

**NEW QUESTION 95**

When using a remote backend or terraform Cloud integration, where does Terraform save resource state?

- A. In an environment variable
- B. On the disk
- C. In the remote backend or Terraform Cloud
- D. In memory

**Answer: C**

**Explanation:**

This is where Terraform saves resource state when using a remote backend or Terraform Cloud integration, as it allows you to store and manage your state file in a remote location, such as a cloud storage service or Terraform Cloud's servers. This enables collaboration, security, and scalability for your Terraform infrastructure.

**NEW QUESTION 96**

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<sup>3</sup>. The other commands are not relevant for this task.

**NEW QUESTION 100**

Which of the following is not true of Terraform providers?

- 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 services<sup>1</sup>. Anyone can write a Terraform provider, either as an individual or as part of a community<sup>2</sup>. 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. References =

- 1: Providers - Configuration Language | Terraform | HashiCorp Developer
- 2: Plugin Development - How Terraform Works With Plugins | Terraform | HashiCorp Developer
- 3: Terraform Registry
- : Terraform Registry

**NEW QUESTION 103**

backends support state locking.

- A. All
- B. No
- C. Some
- D. Only local

**Answer:** C

**Explanation:**

Some backends support state locking, which prevents other users from modifying the state file while a Terraform operation is in progress. This prevents conflicts and data loss. Not all backends support this feature, and you can check the documentation for each backend type to see if it does.

**NEW QUESTION 105**

terraform plan updates your state file.

- A. True
- B. False

**Answer:** B

**Explanation:**

The terraform plan command does not update the state file. Instead, it reads the current state and the configuration files to determine what changes would be made to bring the real-world infrastructure into the desired state defined in the configuration. The plan operation is a read-only operation and does not modify the state or the infrastructure. It is the terraform apply command that actually applies changes and updates the state file. References = Terraform's official guidelines and documentation clarify the purpose of the terraform plan command, highlighting its role in preparing and showing an execution plan without making any changes to the actual state or infrastructure .

**NEW QUESTION 109**

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.

**NEW QUESTION 110**

Which are examples of infrastructure as code? Choose two correct answers.

- A. Cloned virtual machine images
- B. Versioned configuration files
- C. Change management database records
- D. Doctor files

**Answer:** B

**Explanation:**

These are examples of infrastructure as code (IaC), which is a practice of managing and provisioning infrastructure through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

**NEW QUESTION 113**

Which of the following commands would you use to access all of the attributes and details of a resource managed by Terraform?

- A. terraform state list ??provider\_type.name??
- B. terraform state show ??provider\_type.name??
- C. terraform get ??provider\_type.name??
- D. terraform state list

**Answer:** B

**Explanation:**

The terraform state show command allows you to access all of the attributes and details of a resource managed by Terraform. You can use the resource address (e.g. provider\_type.name) as an argument to show the information about a specific



resource. The terraform state list command only shows the list of resources in the state, not their attributes. The terraform get command downloads and installs modules needed for the configuration. It does not show any information about resources. References = [Command: state show] and [Command: state list]

**NEW QUESTION 116**

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.

**NEW QUESTION 120**

FILL IN THE BLANK

What is the name of the default file where Terraform stores the state?

Type your answer in the field provided. The text field is not case-sensitive and all variations of the correct answer are accepted.

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

The name of the default file where Terraform stores the state is terraform.tfstate. This file contains a JSON representation of the current state of the infrastructure managed by Terraform. Terraform uses this file to track the metadata and attributes of the resources, and to plan and apply changes. By default, Terraform stores the state file locally in the same directory as the configuration files, but it can also be configured to store the state remotely in a backend. References = [Terraform State], [State File Format]

**NEW QUESTION 123**

Which command must you first run before performing further Terraform operations in a working directory?

- A. terraform import
- B. terraform workspace
- C. terraform plan
- D. terraform init

**Answer:** D

**Explanation:**

terraform init is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control. It initializes a working directory containing Terraform configuration files and downloads any required providers and modules. The other commands are used for different purposes, such as importing existing resources, switching between workspaces, generating execution plans, etc.

**NEW QUESTION 126**

A Terraform provider is NOT responsible for:

- A. Exposing resources and data sources based on an API
- B. Managing actions to take based on resources differences
- C. Understanding API interactions with some service
- D. Provisioning infrastructure in multiple

**Answer:** D

**Explanation:**

This is not a responsibility of a Terraform provider, as it does not make sense grammatically or logically. A Terraform provider is responsible for exposing resources and data sources based on an API, managing actions to take based on resource differences, and understanding API interactions with some service.

**NEW QUESTION 127**

How would you output returned values from a child module in the Terraform CLI output?

- A. Declare the output in the root configuration
- B. Declare the output in the child module
- C. Declare the output in both the root and child module
- D. None of the above

**Answer:** C

**Explanation:**

To output returned values from a child module in the Terraform CLI output, you need to declare the output in both the child module and the root module. The child module output will return the value to the root module, and the root module output will display the value in the CLI. References = [Terraform Outputs]

**NEW QUESTION 128**

.....

## Thank You for Trying Our Product

\* 100% Pass or Money Back

All our products come with a 90-day Money Back Guarantee.

\* One year free update

You can enjoy free update one year. 24x7 online support.

\* Trusted by Millions

We currently serve more than 30,000,000 customers.

\* Shop Securely

All transactions are protected by VeriSign!

**100% Pass Your Terraform-Associate-003 Exam with Our Prep Materials Via below:**

<https://www.certleader.com/Terraform-Associate-003-dumps.html>