

HashiCorp Terraform Quiz

Gabe Maentz

Question 1: When multiple arguments with single-line values appear on consecutive lines at the same nesting level, HashiCorp recommends that you:

A place all arguments using a variable at the top

```
1 | ami = var.aws_ami
2 | instance_type = var.instance_size
3 | subnet_id = "subnet-0bb1c79de3EXAMPLE"
4 | tags = {
5 |     Name = "HelloWorld"
6 | }
```

C put arguments in alphabetical order

```
1 | name = "www.example.com"
2 | records = [aws_eip.lb.public_ip]
3 | type = "A"
4 | ttl = "300"
5 | zone_id = aws_route53_zone.primary.zone_id
```

B place a space in between each line

```
1 | type = "A"
2 |
3 | ttl = "300"
4 |
5 | zone_id = aws_route53_zone.primary.zone_id
```

D align their equals signs

```
1 | ami           = "abc123"
2 | instance_type = "t2.micro"
```

Question 1: When multiple arguments with single-line values appear on consecutive lines at the same nesting level, HashiCorp recommends that you:

A place all arguments using a variable at the top

```
1 | ami = var.aws_ami
2 | instance_type = var.instance_size
3 | subnet_id = "subnet-0bb1c79de3EXAMPLE"
4 | tags = {
5 |     Name = "HelloWorld"
6 | }
```

C put arguments in alphabetical order

```
1 | name = "www.example.com"
2 | records = [aws_eip.lb.public_ip]
3 | type = "A"
4 | ttl = "300"
5 | zone_id = aws_route53_zone.primary.zone_id
```

B place a space in between each line

```
1 | type = "A"
2 |
3 | ttl = "300"
4 |
5 | zone_id = aws_route53_zone.primary.zone_id
```

D align their equals signs

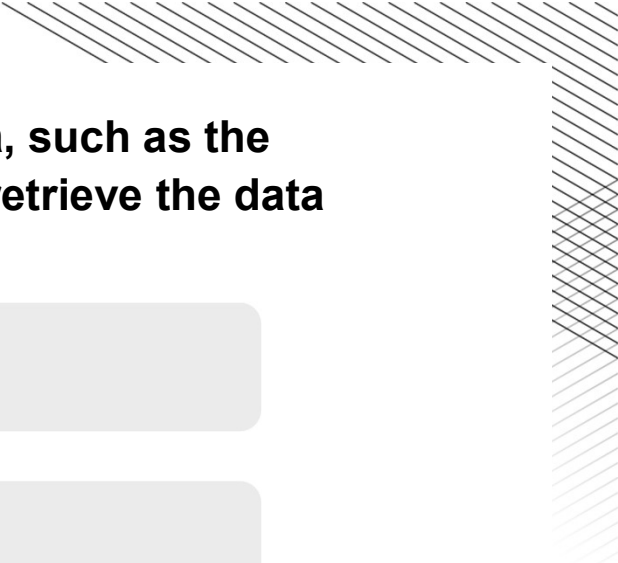
```
1 | ami           = "abc123"
2 | instance_type = "t2.micro"
```

Explanation

HashiCorp style conventions suggest you that align the equals sign for consecutive arguments for easing readability for configurations

```
ami           = "abc123"
```

```
instance_type = "t2.micro"
```



Question 2: When using providers that require the retrieval of data, such as the HashiCorp Vault provider, in what phase does Terraform actually retrieve the data required?:

A terraform delete

B terraform init

C terraform apply

D terraform plan

Question 2: When using providers that require the retrieval of data, such as the HashiCorp Vault provider, in what phase does Terraform actually retrieve the data required?:

A **terraform delete**

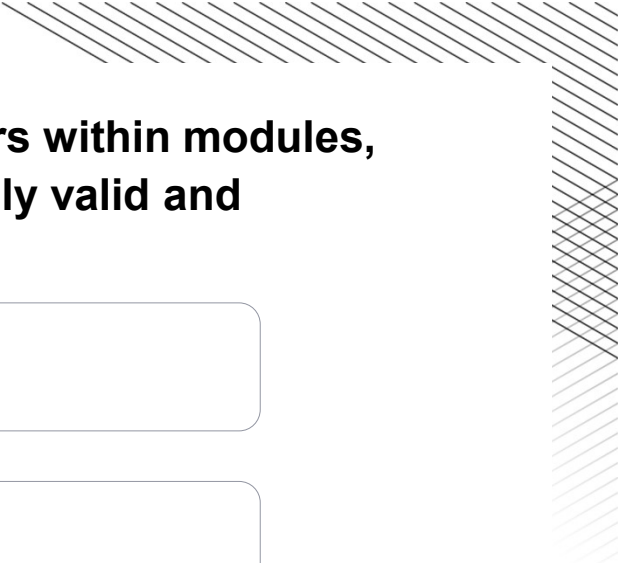
B **terraform init**

C **terraform apply**

D **terraform plan**

Explanation

It is important to consider that Terraform reads from data sources during the **plan** phase and writes the result into the plan. For something like a Vault token which has an explicit TTL, the **apply** must be run before the data, or token, in this case, expires, otherwise, Terraform will fail during the apply phase.



Question 3: Which Terraform command will check and report errors within modules, attribute names, and value types to make sure they are syntactically valid and internally consistent?

A

terraform validate

B

terraform format

C

terraform show

D

terraform fmt

Question 3: Which Terraform command will check and report errors within modules, attribute names, and value types to make sure they are syntactically valid and internally consistent?

A

terraform validate

B

terraform format

C

terraform show

D

terraform fmt

Explanation

The `terraform validate` command validates the configuration files in a directory, referring only to the configuration and not accessing any remote services such as remote state, provider APIs, etc.

Validate runs checks that verify whether a configuration is syntactically valid and internally consistent, regardless of any provided variables or existing state. It is thus primarily useful for general verification of reusable modules, including the correctness of attribute names and value types

Question 4: True or False: When using the Terraform provider for Vault, the tight integration between these HashiCorp tools provides the ability to mask secrets in the `terraform plan` and state files.

A

False

B

True

Question 4: True or False: When using the Terraform provider for Vault, the tight integration between these HashiCorp tools provides the ability to mask secrets in the `terraform plan` and state files.

A

False

B

True

Explanation

Currently, Terraform has no mechanism to redact or protect secrets that are returned via data sources, so secrets read via this provider will be persisted into the Terraform state, into any plan files, and in some cases in the console output produced while planning and applying. These artifacts must, therefore, all be protected accordingly.



Question 5: Terraform Enterprise (also referred to as pTFE) requires what type of backend database for a clustered deployment?

A PostgreSQL

B MySQL

C Cassandra

D MSSQL

Question 5: Terraform Enterprise (also referred to as pTFE) requires what type of backend database for a clustered deployment?

A PostgreSQL

B MySQL

C Cassandra

D MSSQL

Explanation

Production - External Services mode stores the majority of the stateful data used by the instance in an external PostgreSQL database and an external S3-compatible endpoint or Azure blob storage. There is still critical data stored on the instance that must be managed with snapshots. Be sure to check the PostgreSQL Requirements for information that needs to be present for Terraform Enterprise to work. This option is best for users with expertise managing PostgreSQL or users that have access to managed PostgreSQL offerings like AWS RDS.

Question 6: The following is a snippet from a terraform configuration file:-

```
1 provider "aws" {  
2     region = "us-east-1"  
3 }  
4  
5 provider "aws" {  
6     region = "us-west-1"  
7 }
```

which, when validated, results in the following error:-

```
1 Error: Duplicate provider configuration  
2  
3 on main.tf line 5:  
4 5: provider "aws" {  
5  
6 A default provider configuration for "aws" was already given at  
7 main.tf:1,1-15. If multiple configurations are required, set the "_____"  
8 argument for alternative configurations.
```

Fill in the blank in the error message with the correct string from the list below.

A

label

C

multi

B

version

D

alias

Question 6: The following is a snippet from a terraform configuration file:-

```
1 provider "aws" {  
2     region = "us-east-1"  
3 }  
4  
5 provider "aws" {  
6     region = "us-west-1"  
7 }
```

which, when validated, results in the following error:-

```
1 Error: Duplicate provider configuration  
2  
3 on main.tf line 5:  
4 5: provider "aws" {  
5  
6 A default provider configuration for "aws" was already given at  
7 main.tf:1,1-15. If multiple configurations are required, set the "  
8 argument for alternative configurations.
```

Explanation

An **alias** meta-argument is used when using the same provider with different configurations for different resources.

<https://www.terraform.io/docs/configuration/providers.html#alias-multiple-provider-instances>

Fill in the blank in the error message with the correct string from the list below.

A

label

C

multi

B

version

D

alias

Question 7: A user has created three workspaces from the command line - prod, dev and test. The user wants to create a fourth, stage. Which command will the user execute to accomplish this?

A terraform workspace – create stage

B terraform workspace – new stage

C terraform workspace create stage

D terraform workspace new stage

Question 7: A user has created three workspaces from the command line - prod, dev and test. The user wants to create a fourth, stage. Which command will the user execute to accomplish this?

A **terraform workspace – create stage**

B **terraform workspace – new stage**

C **terraform workspace create stage**

D **terraform workspace new stage**

Explanation

The **terraform workspace new** command is used to create a new workspace.

<https://www.terraform.io/docs/commands/workspace/new.html>

Question 8: In terraform, most resource dependencies are handled automatically. Which of the following statements describes best how terraform resource dependencies are handled?

- A Resource dependencies are handled automatically by the `depends_on` meta_argument which is set to true by default
- B The terraform binary contain a built-in reference map of all the defined terraform resource dependencies. Updates to this dependency map are reflected in terraform versions. To ensure you are work with the latest resource dependency map you must be running the latest version of terraform.
- C Resource dependencies are identified and maintained in a file called `resource.dependencies` .Each terraform provider is required to maintain a list of all resource dependencies for the provider and its included with the plugin during initialization when terraform init is executed. The file is located in the `terraform.d` folder
- D Terraform analyses any expression within a resource block to find references to other objects and treats those references as implicit ordering requirement when creating, uploading or destroying resources.

Question 8: In terraform, most resource dependencies are handled automatically. Which of the following statements describes best how terraform resource dependencies are handled?

- A Resource dependencies are handled automatically by the `depends_on` meta_argument which is set to true by default
- B The terraform binary contain a built-in reference map of all the defined terraform resource dependencies. Updates to this dependency map are reflected in terraform versions. To ensure you are work with the latest resource dependency map you must be running the latest version of terraform.
- C Resource dependencies are identified and maintained in a file called `resource.dependencies` .Each terraform provider is required to maintain a list of all resource dependencies for the provider and its included with the plugin during initialization when terraform init is executed. The file is located in the `terraform.d` folder
- D** Terraform analyses any expression within a resource block to find references to other objects and treats those references as implicit ordering requirement when creating, uploading or destroying resources.

Explanation: <https://www.terraform.io/docs/configuration/resources.html>

Question 9: Why might a user opt to include the following snippet in their configuration file?

```
1 terraform {  
2   required_version = ">= 0.12"  
3 }
```

A

Terraform 0.12 introduced substantial changes to the syntax used to write Terraform configuration

B

Versions before Terraform 0.12 were not approved by HashiCorp to be used in production

C

The user wants to ensure that the application being deployed is a minimum version of 0.12

D

This ensures that all Terraform providers are above a certain version to match the application being deployed

Explanation:

You can use the `required_version` to ensure that the user deploying infrastructure is using Terraform 0.12 or great, due to the vast number of changes that were introduced. As a result, many previously written configurations had to be converted or rewritten.



Question 10: In order to reduce the time it takes to provision resources, Terraform uses parallelism. By default, how many resources will Terraform provision concurrently?

A

5

B

10

C

20

D

50

Question 10: In order to reduce the time it takes to provision resources, Terraform uses parallelism. By default, how many resources will Terraform provision concurrently?

A 5

B 10

C 20

D 50

Explanation

Terraform can limit the number of concurrent operations as Terraform walks the graph using the `-parallelism=n` argument. The default value for this setting is 10. This setting might be helpful if you're running into API rate limits.



Question 11: Which of the following commands will launch the Interactive console for Terraform interpolations?

A

terraform

B

terraform console

C

terraform cli

D

terraform cmdline

Question 11: Which of the following commands will launch the Interactive console for Terraform interpolations?

A

terraform

B

terraform console

C

terraform cli

D

terraform cmdline

Explanation

The `terraform console` command provides an interactive console for evaluating expressions.

<https://www.terraform.io/docs/commands/console.html>



Question 12: Which of the following is not a valid Terraform string function?

A **format**

B **tostring**

C **join**

D **replace**

Question 12: Which of the following is not a valid Terraform string function?

A **format**

B **tostring**

C **join**

D **replace**

Explanation

tostring is not a string function, it is a type conversion function. **tostring** converts its argument to a string value.

<https://www.terraform.io/docs/configuration/functions/tostring.html>

Question 13: Anyone can publish and share modules on the **Terraform Public Module Registry** and meeting the requirements for publishing a module is extremely easy. Select from the following list all valid requirements.

A

The registry uses tags to identify module versions. Release tag names must be for the format x.y.z, and can optionally be prefixed with a **v**

B

Module repositories must use this three-part name format, terraform-**<PROVIDER>-<Name>**.

C

The module must be PCI/HIPPA compliant

D

The module must be on Github and must be a public repo.

Question 13: Anyone can publish and share modules on the **Terraform Public Module Registry** and meeting the requirements for publishing a module is extremely easy. Select from the following list all valid requirements.

A

The registry uses tags to identify module versions. Release tag names must be for the format x.y.z, and can optionally be prefixed with a v

B

Module repositories must use this three-part name format, terraform- <PROVIDER>-<Name>.

C

The module must be PCI/HIPPA compliant

D

The module must be on Github and must be a public repo.

Explanation

The list below contains all the requirements for publishing a module. Meeting the requirements for publishing a module is extremely easy. The list may appear long only to ensure we're detailed but adhering to the requirements should happen naturally.

GitHub. The module must be on GitHub and must be a public repo. This is only a requirement for the public registry. If you're using a private registry, you may ignore this requirement.

Named terraform-**<PROVIDER>-<NAME>**. Module repositories must use this three-part name format, where **<NAME>** reflects the type of infrastructure the module manages and **<PROVIDER>** is the main provider where it creates that infrastructure. The **<NAME>** segment can contain additional hyphens. Examples: **terraform-google-vault** or **terraform-aws-ec2-instance**.

Repository description. The GitHub repository description is used to populate the short description of the module. This should be a simple one sentence description of the module.

Standard module structure. The module must adhere to the standard module structure. This allows the registry to inspect your module and generate documentation, track resource usage, parse submodules and examples, and more.

x.y.z tags for releases. The registry uses tags to identify module versions. Release tag names must be a semantic version, which can optionally be prefixed with a v. For example, v1.0.4 and 0.9.2. To publish a module initially, at least one release tag must be present. Tags that don't look like version numbers are ignored.

<https://www.terraform.io/docs/registry/modules/publish.html#requirements>

Question 14: A user has created a module called "my_test_module" and committed it to GitHub. Over time, several commits have been made with updates to the module, each tagged in GitHub with an incremental version number. Which of the following lines would be required in a module configuration block in terraform to select tagged version v1.0.4?

A `source = "git::https://example.com/my_test_module.git#tag=v1.0.4"`

B `source = "git::https://example.com/my_test_module.git@tag=v1.0.4"`

C `source = "git::https://example.com/my_test_module.git&ref=v1.0.4"`

D `source = "git::https://example.com/my_test_module.git?ref=v1.0.4"`

Question 14: A user has created a module called "my_test_module" and committed it to GitHub. Over time, several commits have been made with updates to the module, each tagged in GitHub with an incremental version number. Which of the following lines would be required in a module configuration block in terraform to select tagged version v1.0.4?

A `source = "git::https://example.com/my_test_module.git#tag=v1.0.4"`

B `source = "git::https://example.com/my_test_module.git@tag=v1.0.4"`

C `source = "git::https://example.com/my_test_module.git&ref=v1.0.4"`

D `source = "git::https://example.com/my_test_module.git?ref=v1.0.4"`

Explanation

By default, Terraform will clone and use the default branch (referenced by HEAD) in the selected repository. You can override this using the ref argument:

```
module "vpc" {  
  source =  
    "git::https://example.com/vpc.git?ref=v1.2.0"  
}
```

The value of the ref argument can be any reference that would be accepted by the git checkout command, including branch and tag names.

<https://www.terraform.io/docs/modules/sources.html#selecting-a-revision>



Question 15: Select all Operating Systems that Terraform is available for.

A **macOS**

B **Solaris**

C **FreeBSD**

D **Windows**

E **Linux**

Question 15: Select all Operating Systems that Terraform is available for.

A

macOS

B

Solaris

C

FreeBSD

D

Windows

E

Linux

Explanation

Terraform is available for macOS, FreeBSD, OpenBSD, Linux, Solaris, Windows

<https://www.terraform.io/downloads.html>

Question 16: Select the most accurate statement to describe the Terraform language from the following list.

A

Terraform is an immutable, procedural, Infrastructure as Code configuration management language based on HashiCorp Configuration Language or optionally JSON.

B

Terraform is an immutable, declarative, Infrastructure as Code provisioning language based on HashiCorp Configuration Language or optionally JSON.

C

Terraform is an mutable, declarative, Infrastructure as Code configuration management language based on HashiCorp Configuration Language or optionally JSON.

D

Terraform is an mutable, procedural, Infrastructure as Code provisioning language based on HashiCorp Configuration Language or optionally YAML.

Question 16: Select the most accurate statement to describe the Terraform language from the following list.

A Terraform is an immutable, procedural, Infrastructure as Code configuration management language based on HashiCorp Configuration Language or optionally JSON.

B Terraform is an immutable, declarative, Infrastructure as Code provisioning language based on HashiCorp Configuration Language or optionally JSON.

C Terraform is an mutable, declarative, Infrastructure as Code configuration management language based on HashiCorp Configuration Language or optionally JSON.

D Terraform is an mutable, procedural, Infrastructure as Code provisioning language based on HashiCorp Configuration Language or optionally YAML.

Explanation

Terraform is not a configuration management tool - <https://www.terraform.io/intro/vs/chef-puppet.html>

Terraform is a declarative language - <https://www.terraform.io/docs/configuration/index.html>

Terraform supports a syntax that is JSON compatible - <https://www.terraform.io/docs/configuration/syntax-json.html>

Terraform is primarily designed on immutable infrastructure principles - <https://www.hashicorp.com/resources/what-is-mutable-vs-immutable-infrastructure>



Question 17: True or False: Each Terraform workspace uses its own state file to manage the infrastructure associated with that particular workspace.

A **False**

B **True**

Question 17: True or False: Each Terraform workspace uses its own state file to manage the infrastructure associated with that particular workspace.

A

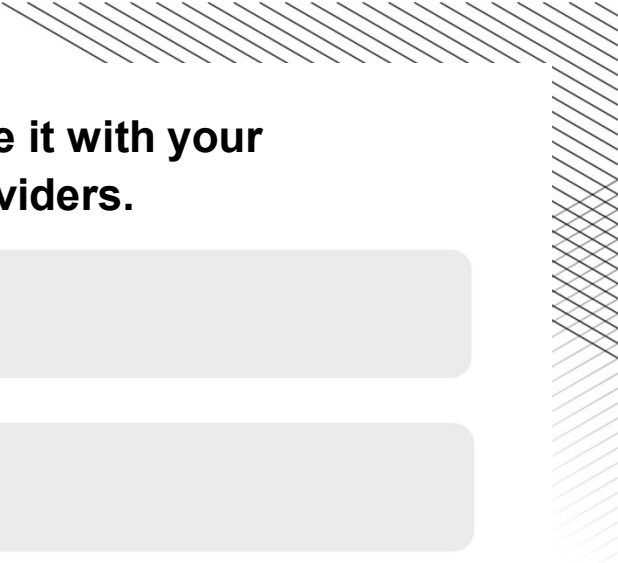
False

B

True

Explanation

The persistent data stored in the backend belongs to a workspace. Initially, the backend has only one workspace, called "default", and thus there is only one Terraform state associated with that configuration.



Question 18: Terraform Cloud is more powerful when you integrate it with your version control system (VCS) provider. List all supported VCS providers.

A Bitbucket Cloud

B Azure DevOps Server

C CVS Version Control

D Github

E GitHub Enterprise

Question 18: Terraform Cloud is more powerful when you integrate it with your version control system (VCS) provider. List all supported VCS providers.

A Bitbucket Cloud

B Azure DevOps Server

C CVS Version Control

D Github

E GitHub Enterprise

Explanation

Terraform Cloud supports the following VCS providers:

[GitHub](#)

[GitHub Enterprise](#)

[GitLab.com](#)

[GitLab EE and CE](#)

[Bitbucket Cloud](#)

[Bitbucket Server](#)

[Azure DevOps Server](#)

[Azure DevOps Services](#)

<https://www.terraform.io/docs/cloud/vcs/index.html#supported-vcs-providers>

Question 19: True or False? By default, Terraform `destroy` will prompt for confirmation before proceeding.

A False

B True

Question 19: True or False? By default, Terraform `destroy` will prompt for confirmation before proceeding.

A **False**

B **True**

Explanation

Terraform `destroy` will always prompt for confirmation before executing unless passed the `-auto-approve` flag.

```
$ terraform destroy
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

Enter a value:
```

Question 20: When using constraint expressions to signify a version of a provider, which of the following are valid provider versions that satisfy the expression found in the following code snippet:

```
1 terraform {  
2   required_providers {  
3     aws = "~> 1.2.0"  
4   }  
5 }
```

A **1.2.9**

B **1.3.1**

C **1.3.0**

D **1.2.3**

Question 20: When using constraint expressions to signify a version of a provider, which of the following are valid provider versions that satisfy the expression found in the following code snippet:

```
1 terraform {  
2   required_providers {  
3     aws = "~> 1.2.0"  
4   }  
5 }
```

Explanation

~> 1.2.0 will match any non-beta version of the provider between $\geq 1.2.0$ and $< 1.3.0$. For example, 1.2.X

A 1.2.9

B 1.3.1

C 1.3.0

D 1.2.3

Question 21: A user creates three workspaces from the command line - prod, dev and test. Which of the following commands will the user run to switch to the dev workspace?

A terraform workspace select dev

B terraform workspace switch dev

C terraform workspace dev

D terraform workspace -switch dev

Question 21: A user creates three workspaces from the command line - prod, dev and test. Which of the following commands will the user run to switch to the dev workspace?

A

terraform workspace select dev

B

terraform workspace switch dev

C

terraform workspace dev

D

terraform workspace -switch dev

Explanation

The **terraform workspace select** command is used to choose a different workspace to use for further operations.

<https://www.terraform.io/docs/commands/workspace/select.html>

Question 22: The terraform language supports a number of different syntaxes for comments. Select all that are supported.

A `/* and */`

B `<* and *>`

C `#`

D `//`

Question 22: The terraform language supports a number of different syntaxes for comments. Select all that are supported.

A

`/* and */`

B

`<* and *>`

C

`#`

D

`//`

Explanation

<https://www.terraform.io/docs/configuration/syntax.html#comments>

Question 23: Which of the following terraform subcommands could be used to remove the lock on the state for the current configuration?

A state-unlock

B unlock

C Removing the lock on a state file is not possible

D force unlock

Question 23: Which of the following terraform subcommands could be used to remove the lock on the state for the current configuration?

A **state-unlock**

B **unlock**


C **Removing the lock on a state file is not possible**

D **force unlock**

Explanation

terraform force-unlock removes the lock on the state for the current configuration.

<https://www.terraform.io/docs/commands/force-unlock.html>



Question 24: When Terraform needs to be installed in a location where it does not have internet access to download the installer and upgrades, the installation is generally known as to be _____.

A a private install

B disconnected

C air-gapped

D non-traditional

Question 24: When Terraform needs to be installed in a location where it does not have internet access to download the installer and upgrades, the installation is generally known as to be _____.

A a private install

B disconnected

C air-gapped

D non-traditional

Explanation

A Terraform Enterprise install that is provisioned on a network that does not have Internet access is generally known as an air-gapped install. These types of installs require you to pull updates, providers, etc. from external sources vs. being able to download them directly.

Question 25: The `terraform refresh` command is used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure. If drift is detected between the real-world infrastructure and the last know-state, it will modify the infrastructure to correct the drift. True or False.

A **False**

B **True**

Question 25: The `terraform refresh` command is used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure. If drift is detected between the real-world infrastructure and the last known-state, it will modify the infrastructure to correct the drift. True or False.

A

False

B

True

Explanation

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. If the state is changed, this may cause changes to occur during the next plan or apply.

<https://www.terraform.io/docs/commands/refresh.htm>

|



Question 26: While Terraform is generally written using the Terraform language, what other syntax can Terraform be expressed in?

A XML

B JSON

C YAML

D TypeScript



Thank You