50 Free Terraform Certification Exam Questions

DevOps / By Jeevitha TP / December 2, 2021

The Terraform Associate certification is for Cloud Engineers. You should have basic terminal skills and an understanding of on-premises and cloud architecture.

If you are new to the Terraform environment, we recommend you to practice in the demo environment and also check our Terraform Associate practice exam.

This set of 50 free Terraform certification exam questions should give you a solid understanding of how Terraform associate exam is structured, the question format, and the exam pattern.

Table of Contents

What will you learn in Hashicorp Terraform Associate Exam?

This certification will help you learn DevOps and cloud engineering and it focuses mainly on upgrading your skills in Infrastructure as as Code(IaC). Other than that you will learn the 9 following topics.

- Comprehending Infrastructure as Code (IaC)
- Assessing Terraform's purpose
- Fundamentals of Terraform
- Handling Terraform CLI (outside workflow)
- Managing Terraform Workflows

- Deploy and handle state
- Managing Terraform Modules
- Familiarity with Terraform cloud and Enterprise capabilities
- Read, produce, and alter configurations

Latest Updates 2023

Hashicorp Certified Terraform Associate 003 – coming soon

HashiCorp Certified Terraform Certification Exam Questions

Q 1. One of your colleagues is new to Terraform and wants to add a new workspace named new-hire. What command he should execute from the following?

A. terraform workspace –new –new-hire

B. terraform workspace new new-hire

C. terraform workspace init new-hire

D. terraform workspace new-hire

Option B:

terraform workspace new new-hire is the right syntax to be used whenever you want to create a new workspace.

Example :

terraform workspace new new-hire

Created and switched to workspace "new-hire "!

Q 2. John is a newbie to Terraform and wants to enable detailed logging to find all the details
Which environment variable does he need to set?

A. TF_help

B. TF_LOG

C. TF_Debug

D. TF_var_log

Option: B is correct

By default, Terraform does not provide detailed logging. To enable detailed logging, we have to set the environment variable TF_LOG.

By enabling TF_LOG, you can set to TRACE, INFO, WARN or ERROR, DEBUG Levels.

Q 3. Which option will you use to run provisioners that are not associated with any resources?

A. Local-exec

B. Null_resource

C. Salt-masterless

D. Remote-exec

Option B is correct

If you need to run provisioners that aren't directly associated with a specific resource, you can associate them with a null_resource.
Refer the link below for Explanation:

Q 4. Which language does terraform support from the below list?

A. XML

B. Javascript

C. Hashicorp Language & JSON

D. Plaintext

Option C is correct

Terraform supports Hashicorp Language & JSON, files ending in .tf and tf.json format.

Q 5. What is the provider version of Google Cloud being used in Terraform?
Google = "~> 1.9.0"

A. 1.9.1

B. 1.0.0

C. 1.8.0

D. 1.9.2

Options A and D are correct

According to the Terraform doc, the operator ~>(Pessimistic Constraint Operator) means only the minor (rightmost version increase) updates are accepted. Therefore, ~> 1.9.0 means the related module/provider requirement accepts 1.9.1 to 1.9.x, but not 1.10.0, and absolutely not 1.0.0 or 1.8.0.
Source: expression/version constraints in the Terraform docs.
In reference to this question, terraform is looking for any update above 1.9.0, which can be either 1.9.1 or 1.9.2. Hence, both answers are correct.

Q 6. On executing terraform plan, terraform scans the code and appends any missing argument before terraform apply.

A. True

B. False

Option: False

On executing terraform plan, terraform scans the code and checks for syntactical errors, missing arguments. Users need to fix these warnings before executing the code successfully.

Q 7. Do terraform workspaces help in adding/allowing multiple state files for a single configuration?

A. True

B. False

Option: True

Terraform workspaces allow configuring multiple state files and associating with a single configuration file

Q 8. Does terraform standard backend type support remote management system?

A. True

B. False

Option: False

The docs outline two **types of backends**: enhanced and standard. Enhanced **backends** are local, which is the default, and remote, which generally refers to **Terraform** Cloud. The one major feature of an enhanced **backend** is the **support** for remote operations

Q 9. Does terraform refresh command updates the state files?

A. True

B. False

Option: True

Yes, terraform refresh updates the state files to the latest unless there are any manual changes.

Q 10. Which command is used to launch terraform console?

A. terraform apply -config

B. terraform console

C. terrafrom plan

D. terrafrom consul

Answer : B

terraform console [options] [dir]

This command helps with an interactive command-line console for evaluating and experimenting with expressions

Q 11. Which of the following below helps users to deploy policy as a code?

A. Resources

B. Functions

C. Sentinel

D. Workspaces

Answer: C

Policy as code is the idea of writing code in a high-level language to manage and automate policies. By representing policies as code in text files, proven software development best practices can be adopted such as version control, automated testing, and automated deployment.
Sentinel is built around the idea and provides all the benefits of policy as code.

Q 12. You have been asked to stop using static values and make code more dynamic. How can you achieve it? Select the correct option from below.

A. Local values

B. Input variables

C. Depends_on

D. Functions

Answer: B

If we compare terraform with any conventional programming language, then,
Input Variables: Input variables are equivalent to function arguments.(Correct Answer)
Local Values: Local value are like function's temporary local variables.
Terraform Functions: The Terraform language includes a number of built-in functions that you can call from within expressions to transform and combine values. The Terraform language does not support user-defined functions, and so only the functions built in to the language are available for use.
Depends_On: It is a meta argument to signify explicit dependencies in terraform resource creation.

Q 13. Which of the following flags can be used with terraform apply command?

A. Auto-approve

B. Init

C. Get

D. Console

Answer – A

The behavior of terraform apply differs significantly depending on whether you

pass it the filename of a previously-saved plan file.
The terraform apply command executes the actions proposed in a Terraform plan.
-auto-approve: Skips interactive approval of the plan before applying. This option is ignored when you pass a previously-saved plan file because Terraform considers you passing the plan file as the approval and so will never prompt in that case.
-compact-warnings: This shows any warning messages in a compact form which includes only the summary messages unless the warnings are accompanied by at least one error and thus the warning text might be useful context for the errors.
-input=false: Disables all of Terraform's interactive prompts. Note that this also prevents Terraform from prompting for interactive approval of a plan, so Terraform will conservatively assume that you do not wish to apply the plan, causing the operation to fail. If you wish to run Terraform in a non-interactive context, see Running Terraform in Automation for some different approaches.
-lock=false: Disables Terraform's default behavior of attempting to take a read/write lock on the state for the duration of the operation.
-lock-timeout=DURATION: Unless locking is disabled with -lock=false, instructs Terraform to retry acquiring a lock for a period of time before returning an error. A duration syntax is a number followed by a time unit letter, such as "3s" for three seconds.
-no-color: Disables terminal formatting sequences in the output. Use this if you are running Terraform in a context where its output will be rendered by a system that cannot interpret terminal formatting.
-parallelism=n: Limit the number of concurrent operations as Terraform walks the graph. Defaults to 10.
For configurations using the local backend only, terraform apply also accepts the legacy options -state, -state-out, and -backup.

Q 14. What is the default number of concurrent operations supported by terraform apply command?

A. 100

B. 10

C. 5

D. 1

Answer – B

The default number of concurrent operations supported by Terraform apply command is 10.
-parallelism=n – Limit the number of concurrent operations as Terraform walks the graph and the default is 10

Q 15. You are trying to login into Terraform Enterprise. Which of the following command is used to save the API token?

A. terraform get
B. terraform API-get
C. terraform login
D. terraform cloud – get api

Answer – C
terraform login command can be used to automatically obtain and save an API token for Terraform Cloud, Terraform Enterprise, or any other host that offers Terraform services.
By default, Terraform will obtain an API token and save it in plain text in a local CLI configuration file called credentials.tfrc.json. When you run terraform login, it will explain specifically where it intends to save the API token and give you a chance to cancel if the current configuration is not as desired.
The syntax for Terraform login:
terraform login [hostname]
If you don't provide an explicit hostname, Terraform will assume you want to log in to Terraform Cloud at app.terraform.io.

Q 16. What are the two supported backend types in Terraform?

A. Remote-backend
B. Enhanced
C. Local- backend
D. Standard

Answer B, D
Terraform's backends are divided into two main types, according to how they handle state and operations:

- **Enhanced** backends can both store state and perform operations. There are only two enhanced backends: local and remote.
- **Standard** backends only store state and rely on the local backend for performing operations.

Q 17. Is terraform state-unlock command used to unlock the locked state file?

A. True

B. False

Answer: False

The correct command is given below-

terraform force-unlock [options] LOCK_ID [DIR]

The above command is used to unlock the state file.

Q 18. SMB(Server Message Block) and RDP(Remote Desktop) are supported connection types in the remote-exec provisioner. True or False?

A. True

B. False

Answer: False

Ssh and winrm are the supported connection types in remote-exec provisioner but not SMB and RDP.

Q 19. Community providers are downloaded automatically using terraform init command. True or False?

A. True

B. False

Answer: True

Any community provider can be automatically downloaded from a Terraform registry by running the terraform init command.

Community providers are installed in the same way as other providers

Q 20. By using the count meta-argument, you can scale the resources by incrementing the number.

A. True

B. False

Answer: True

The count is one of the reserved words. One can use count for scaling instead of repeating the resources again.

The count meta-argument accepts a whole number and creates that many instances of the resource or module.

Q 21. A user wants to list all resources which are deployed using Terraform. How can this be done?

A. terraform state show

B. terraform state list

C. terraform show

D. terraform show list

Answer: B

- **Option A is INCORRECT** because this command shows the attributes of a single resource in the Terraform state file
- **Option B is CORRECT** because the terraform state list command is used to list resources within a Terraform state.
- **Option C is INCORRECT** because this command will output all resources and attributes in a human-readable format.
- **Option D is INCORRECT** because this option will try to display all resources and attributes in the list file. This command takes the list as an input file for the show command.

  The command will list all resources in the state file matching the given addresses (if any). If no addresses are given, all resources are listed.

  The resources listed are sorted according to module depth order followed by alphabetical. This means that resources that are in your immediate configuration are listed first, and resources that are more deeply nested

within modules are listed last.

For complex infrastructures, the state can contain thousands of resources it can filter using the id option.

Q 22. Which among the following log command should be set to get Maximum verbosity of terraform logs?

A. set the TF_LOG=DEBUG in environment variable

B. set the TF_LOG=INFO in environment variable

C. set the TF_LOG=TRACE in environment variable

D. set the TF_LOG=WARN in environment variable

Answer: C

- **Option A is INCORRECT** because this command will not give detailed verbose information compared to trace
- **Option B is INCORRECT** because this command will just give info but will output detailed information.
- **Option C is CORRECT** because this command will output more verbose information compared all other options.
- **Option D is INCORRECT** because this option will only print warn messages.

  Terraform has detailed logs which can be enabled by setting the TF_LOG environment variable to any value. This will cause detailed logs to appear on stderr.

  To persist logged output you can set TF_LOG_PATH in order to force the log to always be appended to a specific file when logging is enabled. Note that even when TF_LOG_PATH is set, TF_LOG must be set in order for any logging to be enabled.

Q 23. Which among the following are not module source options?

A. Local Path

B. Terraform registry

C. Bit bucket

D. HTTP URLs

E. BLOB storage

Answer: E

Options A, B, C, and D are INCORRECT because these are valid source options for a module.

Option E is CORRECT because we cannot use BLOB storage as a module source option.

The source argument in a module block tells Terraform where to find the source code for the desired child module. Terraform uses this during the module installation step of terraform init to download the source code to a directory on a local disk so that it can be used by other Terraform commands.

Currently following are the valid source options for a module:

- Local Paths
- Terraform Registry
- Github
- Bitbucket
- Generic Git, Mercurial repositories
- HTTP URLs
- S3 buckets
- GCS buckets

Q 24. Which of the following command can be used to syntactically check to terraform configuration before using apply or plan command?

A. terraform fmt

B. terraform validate

C. terraform show

D. terraform check

Answer: B

Option A is INCORRECT because fmt is used to rewrite Terraform configuration files to a canonical format and style.

Option B is CORRECT because it is used to validate the terraform configuration.

Option C is INCORRECT because the show is used to provide human-readable output from a state or plan file.

Option D is INCORRECT because there is no command in terraform called to check.

terraform validate command checks whether a configuration is syntactically valid and internally consistent, regardless of any provided variables or existing state. Validation requires an initialized working directory with any referenced plugins and modules installed.

Q 25. When multiple team members are working on the same state file, the state file gets locked. How to remove the lock?

A. terraform force-unlock LOCK_ID

B. terraform force-unlock STATE_FILE

C. terraform unlock LOCK_ID

D. terraform force-unlock=true

Answer: A

- Option A is CORRECT force-unlock with LOCK_ID is used to remove lock on state file.
- Option B is INCORRECT because we need to pass LOCK_ID as argument not STATE_FILE.
- Option C is INCORRECT because force-unlock is used not unlock is used to remove state lock.
- Option D is INCORRECT because force-unlock needs LOCK_ID as an argument. force-unlock manually unlock the state for the defined configuration. This will not modify your infrastructure. This command removes the lock on the state for the current configuration. The behavior of this lock is dependent on the backend being used.
  Local state files cannot be unlocked by another process. Usage: terraform force-unlock LOCK_ID [DIR]

Domain : Use the Terraform CLI (outside of core workflow)

Q26 : If you want to replace a particular object even though there are no configuration changes in the code, which command from below would be best suited.

A. terraform destroy

B. terraform taint

C. terraform replace

D. terraform state rm

Correct Answer: C

Explanation

If your intent is to force replacement of a particular object even though there are no configuration changes that would require it, we recommend instead to use the -replace option with terraform apply. For example:

terraform apply -replace="aws_instance.example[0]"

Creating a plan with the "replace" option is superior to using terraform taint because it will allow you to see the full effect of that change before you take any externally-visible action. When you use terraform taint to get a similar effect, you risk someone else on your team creating a new plan against your tainted object before you've had a chance to review the consequences of that change yourself.

Option A is incorrect as it would destroy the current resource and other resources that need to be created again.
Option B is incorrect because in previous versions it used to happen but as per current version of terraform, the best suited command is terraform replace.
Option D is incorrect because this command is used to manipulate the state file.

Reference Link: https://www.terraform.io/docs/cli/commands/taint.html

Domain : Use the Terraform CLI (outside of core workflow)

Q27 : The data directory in terraform is used to retain data that must persist from one command to the next, so it's important to have this variable set consistently throughout all the Terraform workflow commands (starting with terraform init) or else Terraform may be

unable to find providers, modules, and other artifacts. Which ENVIRONMENT VARIABLE is used from below to 'set' per-working-directory data?

A. TF_DATA_DIR

B. TF_WORKSPACE

C. TF_DATA

D. TF_DATA_WORKSPACE

Correct Answer: A

Explanation

By default per-working-directory data is written into a .terraform subdirectory of the current directory. The TF_DATA_DIR changes the path where Terraform keeps its per-working-directory data, such as the current remote backend configuration.

Option B: For multi-environment deployment, in order to select a workspace, instead of doing '*terraform workspace select your_workspace*', it is possible to use *TF_WORKSPACE* environment variable. It cannot be used to 'set' per-working-directory data.
Option C: No Such ENVIRONMENT VARIABLE exists
Option D: No Such ENVIRONMENT VARIABLE exists

Reference Link: https://www.terraform.io/docs/cli/config/environment-variables.html#tf_data_dir

Domain : Understand Terraform Cloud and Enterprise capabilities

Q28 : Which of the following commands can be used to logout from terraform cloud?

A. Terraform logout

B. Terraform –logout

C. Terraform log out

D. Terraform –log-out

Correct Answer: A

Explanation

The terraform logout command is used to remove credentials stored by terraform login. These credentials are API tokens for Terraform Cloud, Terraform Enterprise, or any other host that offers Terraform services.

Reference Link: https://www.terraform.io/docs/cli/commands/logout.html

Q29 : Debug is the most verbose log level in Terraform.

A. True
B. False

Correct Answer: B

Explanation

Trace is the most verbose log level in Terraform.

You can set TF_LOG to one of the log levels TRACE, DEBUG, INFO, WARN or ERROR to change the verbosity of the logs.

For more information, refer to the link below: https://www.terraform.io/docs/internals/debugging.html

Domain : Understand Terraform's purpose (vs other IaC)

Q30 : What is a multicloud deployment?

A. The possibility to run a simple .tf into multiple cloud using a single provider to deploy into multiple cloud providers
B. The possibility to run your Terraform code using multiple cloud providers to

deploy your infrastructure into multiple cloud providers

C. The possibility to run your Terraform code using a single-global provider to deploy your infrastructure into multiple cloud providers

D. The possibility to run your Terraform code by other tools such as Amazon Cloudformation

Correct Answer: B

Explanation

The idea of multi cloud deployment is to run our Terraform code, using multiple providers like AWS, GCP or Azure and deployed the infrastructure into multiple clouds in a single terraform deployment

Option A is incorrect as you have to specify different providers to deploy into multiple cloud-providers

Option C is incorrect as there isn't a single-global provider to deploy the infrastructure in multiple cloud-providers

Option D is incorrect as Terraform is not designed to be used in third party applications.

Reference: https://www.terraform.io/intro/use-cases#multi-cloud-deployment

Domain : Understand Terraform basics

Q31 : You have the following provider configuration for AWS:

```
provider "aws" {
  region = "eu-west-1"
}
provider "aws" {
  alias  = "frankfurt"
  region = "eu-central-1"
}
```

How do you specify an instance creation on eu-central-1 ?

A. resource "aws_instance" "whizlabs" { provider = aws.central ... }

B. resource “aws_instance” “whizlabs” { provider = aws.frankfurt … }
C. resource “aws_instance” “whizlabs” { … }
D. resource “aws_instance” “whizlabs” { provider = aws.west … }

Correct Answer: B

Explanation

The correct way to reference a provider is specifying the provider and the alias that have been configured on the provider configuration.

Option A is incorrect as the alias assigned into the resource doesn’t exist on the provider configuration.
Option C is incorrect as if you specify a provider, this will be using the default one.
Option D is incorrect as the alias assigned into the resource doesn’t exist on the provider.

Reference: https://www.terraform.io/language/providers/configuration#selecting-alternate-provider-configurations

Domain : Interact with Terraform modules

Q32 : What of the following is not a source type for a module?

A. SSH
B. Github
C. Bitbucket
D. S3

Correct Answer: A

Explanation

You can’t use a module using a SSH source from another computer or station.

Option A is correct as SSH is not the source type for a module.
Options B, C and D are true but not the only ones. All of this sources are part of the

sources allowed:

- Local paths
- Terraform Registry
- GitHub
- Bitbucket
- Generic Git, Mercurial repositories
- HTTP URLs
- S3 buckets
- GCS buckets
- Modules in Package Subdirectories

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

Domain : Interact with Terraform modules

Q33 : How do you download a module configured in your Terraform code?

```
module “ecs_cluster” {
source  = “terraform-aws-modules/ecs/aws”
version = “2.8.0”
// inputs
}
```

A. terraform get module ecs_cluster

B. terraform install modules ecs_cluster

C. terraform init

D. terraform module init

Correct Answer: C

Explanation

During terraform init, terraform searches for module blocks and is retrieved.

Option A, B and D are incorrect as those are not valid options when initializing

plugins or modules in Terraform.

Reference: https://www.terraform.io/cli/commands/init

Domain : Navigate Terraform workflow

Q34 : You are a DevOps Engineer working in a CI/CD Pipeline using Jenkins. You have three stages identified: Init, Plan and Apply. After your terraform plan, you need to apply your infrastructure. In your pipeline script basically you wrote: "terraform apply".
After triggering the pipeline you see that there was no progress and the Apply stage is waiting to confirm the changes. How can you automatically apply the changes when you type "terraform apply" ?

A. terraform apply -auto-approve

B. terraform apply -yes

C. terraform apply | echo "yes"

D. terraform apply | yes

Correct Answer: A

Option A is correct. Using this flag to skip the interactive auto-approve.

Option B is incorrect as this flag doesn't exists.

Option C is incorrect as this is not part of the best practices using Terraform and also this is not interacting with terraform output.

Option D is incorrect as this will prompt a syntax error in your command line.

Reference: https://www.terraform.io/cli/commands/apply#auto-approve

Domain : Implement and maintain state

Q35 : What is the name of the workspace when you execute "terraform init"?

A. New

B. No workspace is created

C. Workspace

D. Default

Correct Answer: D

Explanation

When you initialize a working directory, a default workspace is created with the name “default”.

Options A, B and C are incorrect.

Reference: https://www.terraform.io/cli/workspaces#managing-workspaces

Domain : Implement and maintain state

Q36 : How can you delete the default workspace?

A. terraform workspace delete default
B. terraform delete workspace default
C. terraform workspace -rm default
D. None of the options are correct

Correct Answer: D

Explanation

You can’t delete the default workspace.

Option A is incorrect because you can’t delete the default workspace
Options B and C are incorrect because those commands have syntax failures and also you can’t delete the default workspace.

Reference: https://www.terraform.io/language/state/workspaces#using-workspaces

Domain : Read, generate, and modify configuration

Q37 : You are a Senior DevOps Engineer and you want to provision your infrastructure Terraform code in different environments having your Terraform configuration DRY. What is the best way to do it?

You also want to minimize the number of changes in your code (Choose the best answer regarding best practices in Terraform and DevOps)

A. Have a different var files per environment and apply those files to your Terraform Configuration

B. Have different branches in your Git repository with different var files

C. Move out from Terraform and use Terragrunt

D. Both A and B are correct

Correct Answer: A

Explanation

To manage and have the Terraform configuration DRY, the best way is to execute terraform apply and have your terraform variables in a separate file with the variables for each environment.

For example:

// For Development environment

terraform apply -var-file="development.tfvars"

// For Production

terraform apply -var-file="production.tfvars"

Option B is incorrect because having multiple branches you will need to modify the Terraform configuration in different branches. Also, this is not a good DevOps practice

Option C is incorrect as you will need to modify the templates to make your code working with Terragrunt and this will not minimize the number of changes.

Option D is incorrect as Option A is correct.

Reference: https://www.terraform.io/language/values/variables#variable-definitions-tfvars-files

Domain : Read, generate, and modify configuration

Q38 : How can you view the value of a particular output using the CLI? The output you want to query was declared like

```
output "ips" {
  value = aws_instance.frontend.*.public_ip
}
```

A. terraform output show

B. terraform output show *ips*

C. terraform output

D. terraform output ips

Correct Answer: D

Explanation

To show the value of a particular output you have to specify the command terraform output followed by the name of the output which you want to query:

$ terraform output ips

ips = [

"54.43.114.15",

"52.12.13.4",

"52.4.161.69"

]

Options A and B are incorrect as to query an output you don't have to specify the sub-command "show". Also this will prompt an issue.
Option C is incorrect as this will be showing all the output values and not a particular one.

Reference: https://www.terraform.io/cli/commands/output#examples

Domain : Understand Terraform Cloud and Enterprise capabilities

Q39 : You are working with different Terraform States and you need access to the Terraform state for the organization "whizlabs" and the workspace "prod". How will you configure the Terraform Datasource?

A. data "terraform_remote_state" "remote_state" { backend = "remote" config = { organization = "whizlabs" workspaces = { name = "prod" } } }

B. data "terraform_remote_state" "remote_state" { backend = "remote" organization = "whizlabs" workspaces = "prod" } }

C. data "terraform_remote_state" "remote_state" { backend = "remote" organization = "whizlabs.prod" } }

D. None of the above

Correct Answer: A

Explanation

A datasource "terraform_remote_state" must be configured. In this configuration, the organization and the workspaces must be set under the config block to access the remote state as a datasource.

Options B and C are incorrect as the there is not "config" block for the terraform_remote_state datasource configuration
Option D is incorrect as Option A is correct.

References: https://www.terraform.io/language/state/remote-state-data, https://www.terraform.io/cloud-docs/workspaces/state#data-source-configuration

Domain : Understand infrastructure as code (IaC) concepts

Q40 : What benefits can provide the Infrastructure as Code for organizations?

A. IaC can be used to deploy the latest features of Cloud Services

B. Share and reusability of the code

C. Blueprint of the DataCenter

D. Versioning

Correct Answers: B, C and D

Explanation

IaC is the way to treat Infrastructure the same way Developers treat code. Versioning and reusability are the most important topics on IaC. Also there is a possibility to have the datacenter snapshot via IaC.

Option A is incorrect because the latest features for the Cloud Services depend on the existing of API calls exposed and used by the Terraform providers

Reference: https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code

Domain : Understand Terraform's purpose (vs other IaC)

Q41 : State is a necessary requirement for Terraform to function. What of the following is not a purpose of Terraform State?

A. Mapping to the real world

B. Performance

C. Syncing

D. Security

Correct Answer: D

Explanation

State is necessary to have Terraform running. Mapping to the real world, Metadata

Performance and Syncing are part of the purpose to have a Terraform State, However, Security is not part of this purpose as not always you need to manage sensitive data and there are some options to manage those values

Options A, B and C are incorrect as those options are part of the purpose to have Terraform State

Reference: https://www.terraform.io/language/state/purpose#purpose-of-terraform-state

Domain : Use the Terraform CLI (outside of core workflow)

Q42 : You want to test the "split" function of Terraform locally to verify the output that the split function will be returned. What is the best approach to test this function locally?

A. echo 'split(",", "foo,bar,baz")' | terraform console

B. echo 'split(",", "foo,bar,baz")' | terraform plan

C. echo 'split(",", "foo,bar,baz")' | terraform apply

D. None of above

Correct Answer: A

Explanation

Terraform console can be used to run non-interactive scripts. For example, we can make use of the split function and pipe this function to terraform console commands.

Options B and C are incorrect as terraform plan and terraform apply are used to execute configuration written in*.tf files or json files.

Resource: https://www.terraform.io/cli/commands/console#scripting

Domain : Interact with Terraform modules

Q43 : In Terraform, What are modules used for?

A. Re-use configuration

B. Ensure best practices

C. Encapsulate configuration

D. All of the above

Correct Answer: D

Modules are very useful to re-use configuration, promotion best practices, encapsule configuration and also to have a configuration organize

Options A, B and C are correct but are not matching with the best answer provided.

Reference: https://learn.hashicorp.com/tutorials/terraform/module#what-are-modules-for

Domain : Interact with Terraform modules

Q44 : Which of the following extensions is recognized by Terraform to fetch a module using an URL endpoint.

A. Zip

B. Tar.gz

C. Tar.xz

D. All of the above

Correct Answer: D

Explanation

All the extensions are recognized by Terraform:

zip

tar.bz2 and tbz2

tar.gz and tgz

tar.xz and txz

Reference: https://www.terraform.io/language/modules/sources#fetching-archives-over-http

Domain : Implement and maintain state

Q45 : What happens if the locking state fails when executing an operation in Terraform?

A. Terraform will continuously apply its configuration without modifying the state, then you can execute a Terraform refresh to update the state
B. Terraform will continuously apply its configuration and apply changes into the state
C. Terraform will not continue to plan/apply any changes
D. Terraform will continuous and will force lock the state and will refresh the state

Correct Answer: C

Explanation

If terraform fails to acquire the locking state, Terraform will not continue to apply any changes unless you specify the flag "-lock" to disable the locking state.

Options A, B and D are incorrect as terraform will not continue executing the plan or apply operations if terraform fails to acquire the state.

Reference: https://www.terraform.io/language/state/locking#state-locking

Domain : Read, generate, and modify configuration

Q46 : You want to assign the default value "No description set up" to a variable in your Terraform code just if a value has not been assigned on the variables.tf. If this value has content, you can assign the value to the variable. How can you perform this in your

Terraform code?

A. description = var.description == “null” ? “No description set up” : var.description
B. description = if var.description == “null” then “No description set up” else var.description
C. description = if (var.description == “null”) then { “No description set up” } else { var.description }
D. description = var.description == “null” : “No description set up” ? var.description

Correct Answer: A

Conditions in terraform have the following syntax:

condition ? true_val : false_val

In our example:

if var.description is null, then we assign description value “No description set up”. if not, it would be var.description.

Options B and C are incorrect as this will be a syntax error as Terraform don’t understand about conditional blocks of if / else / then.
Option D is incorrect as the conditional syntax is not following the pattern:

condition ? true_val : false_val

Reference: https://www.terraform.io/cli/commands/console#scripting

Domain : Understand Terraform basics

Q47 : How do you force users to use a particular version of required providers in your terraform code?

A. terraform { required_providers { aws = { source = “hashicorp/aws” version =”3.74.1″ } } }
B. terraform { aws = { source = “hashicorp/aws” version ~> “3.74.1” } }
C. aws = { source = “hashicorp/aws” version = “3.74.1” } }

D. provider “aws” = { source = “hashicorp/aws” version = “3.74.1” }

Correct Answer: A

Explanation

To configure a specific version of a provider that is required, a version must be under the block required_providers.

Option B is incorrect as it is missing the block ‘required_providers’.
Option C is incorrect as there is no block called “aws”.
Option D is incorrect as you have to specify the block ‘required_providers’ inside your terraform configuration.

Reference: https://www.terraform.io/language/providers/configuration

Domain : Understand Terraform basics

Q48 : What of the following next arguments are not part of the generic meta-arguments for a provider?

A. Alias
B. Version
C. Profile
D. Region

Correct Answers: C and D

Explanation

The main generic meta-arguments for a provider are alias and version. Other parameters are related to the provider configuration itself.

Options A and B are incorrect as both are part of the generic meta-arguments for a provider configuration.

References: AWS provider configuration: https://registry.terraform.io/providers

/hashicorp/aws/latest/docs#argument-reference, https://www.terraform.io/language/providers/configuration?_ga=2.5999995.934997445.1642622103-536275114.1619454689#provider-configuration-1

Domain : Understand Terraform basics

Q49 : local-exec invokes a process on the resource that is being created by Terraform.

A. True

B. False

Correct Answer: B

False. The process is always invoked on the machine running Terraform.

Reference: https://www.terraform.io/language/resources/provisioners/local-exec#local-exec-provisioner

Domain : Understand infrastructure as code (IaC) concepts

Q50 : What are the main advantages to use Terraform as the IaC tool?

A. Manage infrastructure on multiple cloud providers

B. Versioning

C. Status of your infrastructure based on a State to track all the resources and components

D. All of above

Correct Answer: D

With Terraform as a tool for the IaC, you have multiple advantages, like use Terraform with multiple cloud providers, versioning your modules or even track all

the resources creation from the Terraform State

Options A, B and C are incorrect as you have to select the best possible option.

Reference: https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code

Conclusion:

This set of 50 free questions help you build an understanding of the exam pattern but not enough. To take the Terraform associate exam you may have to attempt some more practice questions. You can also sign up for a Terraform Demo environment in case your company is not using Terraform right now.

Reference Links:

- https://docs.hashicorp.com/sentinel/concepts/policy-as-code/
- https://www.hashicorp.com/certification/terraform-associate
- https://www.terraform.io/docs/cli/commands/plan.html

About the Author | More from Author

About Jeevitha TP

Jeevitha has a proven experience with a solid understanding of SEO activities such as content strategy, link building, and keyword strategy to increase rankings on all major search networks. Further, she works closely with the editorial and marketing teams to drive SEO in content creation and programming.

← Previous Post Next Post →

Related Posts

The Best Way To Prepare For Puppet Certification

Leave a Comment / DevOps / By Dharmalingam N

Docker Certified Associate Practice Tests Launched

Leave a Comment / DevOps, News & Updates / By Pavan Gumaste

Leave a Comment

Your email address will not be published. Required fields are marked *

Type here..

Name*

Email*

Website

Post Comment »

What you're looking for?

Search …

Popular Posts

15 Best Free Cloud Storage in 2023 – Up to 200…

New Microsoft Azure Certifications Path in 2023 [Updated]

Top 50 Business Analyst Interview Questions

Top 40+ Agile Scrum Interview Questions (Updated)

Top 25+ Fresher Java Interview Questions

Free AWS Solutions Architect Certification Exam…

Top 5 Agile Certifications in 2022 (Updated)

Top 50+ Azure Interview Questions and Answers [2023]

Top 50 Big Data Interview Questions And Answers…

10 Most Popular Business Analysis Techniques