Question 1Skipped

What happens if multiple users attempt to run a terraform apply simultaneously when using a remote backend? (select two)

both users will get an error

the Terraform apply will work for both users

Correct selection

if the backend does not support locking, the state file could become corrupted

Correct selection

if the backend supports locking, the first *terraform apply* will lock the file for changes, preventing the second user from running the *apply*

Overall explanation

If the state is configured for remote state, the backend selected will determine what happens. If the backend supports locking, the file will be locked for the first user, and that user's configuration will be applied. The second user's terraform apply will return an error that the state is locked.

If the remote backend does not support locking, the state file could become corrupted, since multiple users are trying to make changes at the same time.

https://developer.hashicorp.com/terraform/language/state/locking

Domain

Objective 7 - Implement and Maintain State

Question 2Skipped

Jeff is a DevOps Engineer for a large company and is currently managing the infrastructure for many different applications using Terraform. Recently, Jeff received a request to remove a specific VMware virtual machine from Terraform as the application team no longer needs it. Jeff opens his terminal and issues the command:

1. $ terraform state rm vsphere_virtual_machine.app1
2.
3. Removed vsphere_virtual_machine.app1
4. Successfully removed 1 resource instance(s).

The next time that Jeff runs a terraform apply, the resource is not marked to be deleted. In fact, Terraform is stating that it is creating another identical resource.

1. .....
2. An execution plan has been generated and is shown below.
3. Resource actions are indicated with the following symbols:
4. + create
5.
6. Terraform will perform the following actions:
7.
8. # vsphere_virtual_machine.app1 will be created

What would explain this behavior?

the state file was not saved before the terraform apply was executed, therefore Terraform sees that the resource is still in the state file

Correct answer

Jeff removed the resource from the *state file*, and not the *configuration file*. Therefore, Terraform is no longer aware of the virtual machine and assumes Jeff wants to create a new one since the virtual machine is still in the Terraform configuration file

after running the terraform rm command, Jeff needs to run a Terraform plan first to tell Terraform of the updated configuration. A plan will instruct Terraform that the resource should be deleted upon the next terraform apply

the resource was manually deleted within the VMware infrastructure and needs to be recreated

Overall explanation

Because Jeff manually deleted the resource from the state file, Terraform was no longer aware of the virtual machine. When Jeff ran a terraform apply, it refreshed the state file and discovered that the configuration file declared a virtual machine but it was not in state, therefore Terraform needed to create a virtual machine so the provisioned infrastructure matched the desired configuration, which is the Terraform configuration file.

Hopefully, this isn't a tricky one but I thought it was good to test on, especially since terraform state commands are listed in Objective 4 of the exam. In this case, Jeff should NOT have removed the resource from the state file, but rather remove it from the configuration file and run a terraform plan/apply. In this scenario, Terraform would recognize that the virtual machine was no longer needed and would have destroyed it.

https://developer.hashicorp.com/terraform/cli/commands/state/list

Domain

Objective 4 - Use Terraform Outside of Core Workflow

Question 3Skipped

Terraform has detailed logs that can be enabled using the TF_LOG environment variable. Which of the following log levels is the most verbose, meaning it will log the most specific logs?

INFO

ERROR

DEBUG

Correct answer

TRACE

Overall explanation

You can set TF_LOG to one of the log levels TRACE, DEBUG, INFO, WARN or ERROR to change the verbosity of the logs. TRACE is the most verbose and it is the default if TF_LOG is set to something other than a log level name.

https://developer.hashicorp.com/terraform/internals/debugging

Domain

Objective 4 - Use Terraform Outside of Core Workflow

Question 4Skipped

Given the following snippet of code, what does servers = 4 reference?

```
1. module "servers" {
2.   source = "./modules/aws-servers"
3.
4.   servers = 4
5. }
```

the number of times the module will be executed

Correct answer

the value of an input variable

servers is not a valid configuration for a module

the output variable of the module

Overall explanation

When calling a child module, values can be passed to the module to be used within the module itself.

https://developer.hashicorp.com/terraform/language/modules/develop/composition

Domain

Objective 5 - Interact with Terraform Modules

Question 5Skipped

Which block type is used to assign a name to an expression that can be used multiple times within a module without having to repeat it?

provider {}

resource {}

terraform {}

Correct answer

locals {}

Overall explanation

A local value assigns a name to an expression, so you can use it multiple times within a module without repeating it. These local values are declared within a locals block

https://developer.hashicorp.com/terraform/language/values/locals

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 6Skipped

When running the terraform validate command, which issue will be brought to your attention?

parameters inside of a resource block are not lined up with spaces

there is no existing state file for the configuration

there is configuration drift within the managed infrastructure

Correct answer

a variable is being used in a resource block but has not been declared

Overall explanation

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

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

https://developer.hashicorp.com/terraform/cli/commands/validate

Domain

Objective 6 - Use the Core Terraform Workflow

Question 7Skipped

You have a Terraform configuration file defining resources to deploy on VMware, yet there is no related state file. You have successfully run terraform init already. What happens when you run a *terraform apply*?

Since there is no state file associated with this configuration file, the defined resources will be not created on the VMware infrastructure.

All existing infrastructure on VMware will be deleted, and the resources defined in the configuration file will be created.

Terraform will produce an error since there is no state file

Correct answer

Terraform will scan the VMware infrastructure, create a new state file, and deploy the new resources as defined in the configuration file.

Overall explanation

If there is no state file associated with a Terraform configuration file, a terraform apply will create the resources defined in the configuration file. This is a normal workflow during the first terraform apply that is executed against a configuration file. This, of course, assumes that the directory has been initialized using a terraform init

https://developer.hashicorp.com/terraform/language/state/purpose

Domain

Objective 7 - Implement and Maintain State

Question 8Skipped

What feature of Terraform provides an abstraction above the upstream API and is responsible for understanding API interactions and exposing resources?

Terraform configuration file

Terraform provisioner

Terraform backend

Correct answer

Terraform provider

Overall explanation

Terraform relies on plugins called "providers" to interact with remote systems.

Terraform configurations must declare which providers they require so that Terraform can install and use them. Additionally, some providers require configuration (like endpoint URLs or cloud regions) before they can be used.

https://developer.hashicorp.com/terraform/language/providers

Domain

Objective 3 - Understand Terraform Basics

Question 9Skipped

Based on the code provided, how many subnets will be created in the AWS account?

variables.tf

```
1. variable "private_subnet_names" {
2.   type    = list(string)
3.   default = ["private_subnet_a", "private_subnet_b", "private_subnet_c"]
4. }
5. variable "vpc_cidr" {
6.   type    = string
7.   default = "10.0.0.0/16"
8. }
9. variable "public_subnet_names" {
10.  type    = list(string)
11.  default = ["public_subnet_1", "public_subnet_2"]
12. }
```

main.tf

```
1. resource "aws_subnet" "private_subnet" {
2.   count           = length(var.private_subnet_names)
3.   vpc_id          = aws_vpc.vpc.id
4.   cidr_block      = cidrsubnet(var.vpc_cidr, 8, count.index)
5.   availability_zone = data.aws_availability_zones.available.names[count.index]
6.
7.   tags = {
8.     Name     = var.private_subnet_names[count.index]
9.     Terraform = "true"
10.  }
11. }
```

2

Correct answer

3

0

1

Overall explanation

The code above will create **three** subnets. The value of count is determined by the number of strings included in the private_subnet_names variable.

https://developer.hashicorp.com/terraform/language/functions/length

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 10Skipped

Which of the following commands can be used to detect configuration drift?

terraform get

terraform init

terraform fmt

Correct answer

terraform apply -refresh-only

Overall explanation

If the state has drifted from the last time Terraform ran,terraform plan -refresh-only or terraform apply -refresh-only allows drift to be detected.

https://www.hashicorp.com/blog/detecting-and-managing-drift-with-terraform

Domain

Objective 7 - Implement and Maintain State

Question 11Skipped

Which of the following best describes a "data source"?

a file that contains the current working version of Terraform

Correct answer

enables Terraform to fetch data for use elsewhere in the Terraform configuration

provides required data for declared variables used within the Terraform configuration

maintains a list of strings to store the values of declared outputs in Terraform

Overall explanation

Data sources allow data to be fetched or computed for use elsewhere in Terraform configuration. Use of data sources allows a Terraform configuration to make use of information defined outside of Terraform, or defined by another separate Terraform configuration.

https://developer.hashicorp.com/terraform/language/data-sources

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 12Skipped

A provider alias is used for what purpose in a Terraform configuration file?

alias isn't used with providers, they are used with provisioners

to signify what resources should be deployed to a certain cloud provider

to use as shorthand for resources to be deployed with the referenced provider

Correct answer

using the same provider with different configurations for different resources

Overall explanation

To create multiple configurations for a given provider, include multiple provider blocks with the same provider name. For each additional non-default configuration, use the alias meta-argument to provide an extra name segment.

https://developer.hashicorp.com/terraform/language/providers/configuration

Domain

Objective 3 - Understand Terraform Basics

Question 13Skipped

You are an Infrastructure Engineer at Strategies, Inc, which is a new organization that provides marketing services to startups. All of your infrastructure is provisioned and managed by Terraform. Despite your pleas to not make changes outside of Terraform, sometimes the other engineers log into the cloud platform and make minor changes to resolve problems.

What Terraform command can you use to reconcile the state with the real-world infrastructure in order to detect any drift from the last-known state?

terraform graph

terraform state show

terraform validate

Correct answer

terraform apply -refresh-only

Overall explanation

The terraform apply -refresh-only command is used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure. This can be used to detect any drift from the last-known state, and to update the state file.

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

Domain

Objective 7 - Implement and Maintain State

Question 14Skipped

When using Terraform Cloud, what is the easiest way to ensure the security and integrity of modules when used by multiple teams across different projects?

apply TFC organization permissions to all workspaces that allow them to only use certain modules

Create a list of approved modules and send them to your team to ensure they don't use modules that aren't approved by the team

use only modules that are published to the Terraform public registry

Correct answer

Use the TFC Private Registry to ensure only approved modules are consumed by your organization

Overall explanation

To simplify the management of approved modules, you can host all the approved Terraform modules in your organization's Private Registry on Terraform Cloud. The private registry allows you to control access to the modules and ensures they are not publicly available. By implementing a private registry, your organization can effectively control and restrict module consumption to only approved modules hosted in the Terraform Private Registry. This enhances security, maintains consistency in infrastructure deployments, and reduces the risk of using unverified or potentially harmful modules in your Terraform configurations.

Wrong Answers:

- Creating a list is probably a bad idea as it doesn't simplify the management of modules that can be used

- modules published to the public registry aren't "approved" modules, and these modules may not contain or implement security measures required by your organization

- TFC permissions wouldn't work here since they wouldn't be used to control access to certain modules

https://developer.hashicorp.com/terraform/cloud-docs/registry

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 15Skipped

When running a terraform plan, how can you save the plan so it can be applied at a later time?

use the -file flag

Correct answer

use the -out flag

you cannot save a plan

use the -save flag

Overall explanation

The optional -out flag can be used to save the generated plan to a file for later execution with terraform apply, which can be useful when running Terraform in automation.

https://developer.hashicorp.com/terraform/cli/commands/plan

Domain

Objective 6 - Use the Core Terraform Workflow

Question 16Skipped

Larissa is an experienced IT professional and is working to learn Terraform to manage the F5 load balancers that front-end customer-facing applications. Larissa writes great code, but her formatting seldom meets the Terraform canonical formatting and style recommended by HashiCorp. What built-in tool or command can Larissa use to easily format her code to meet the recommendations for formatting Terraform code?

terraform validate

terraform env

Correct answer

terraform fmt

terraform plan

Overall explanation

The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability.

https://developer.hashicorp.com/terraform/cli/commands/fmt

Domain

Objective 4 - Use Terraform Outside of Core Workflow

Question 17Skipped

Rigby is implementing Terraform and was given a configuration that includes the snippet below. Where is this particular module stored?

```
1. module "consul" {
2.   source = "hashicorp/consul/aws"
3.   version = "0.1.0"
4. }
```

locally but a directory back from the current directory

locally in the hashicorp/consul/aws directory

a private registry supported by your organization

Correct answer

public Terraform registry

Overall explanation

Modules on the public Terraform Registry can be referenced using a registry source address of the form <NAMESPACE>/<NAME>/<PROVIDER>, with each module's information page on the registry site including the exact address to use.

https://developer.hashicorp.com/terraform/language/modules/sources#terraform-registry

Domain

Objective 5 - Interact with Terraform Modules

Question 18Skipped

True or False? A remote backend configuration is required for using Terraform.

True

Correct answer

False

Overall explanation

This is false. If you don't provide a backend configuration, Terraform will use the local default backend. **Remote Backends are completely optional**. You can successfully use Terraform without ever having to learn or use a remote backend. However, they do solve pain points that afflict teams at a certain scale. If you're an individual, you can likely get away with never using backends.

https://developer.hashicorp.com/terraform/language/settings/backends/configuration

Domain

Objective 7 - Implement and Maintain State

Question 19Skipped

When using a Terraform provider, it's common that Terraform needs credentials to access the API for the underlying platform, such as VMware, AWS, or Google Cloud. While there are many ways to accomplish this, what are three options that you can provide these credentials? (select three)

Correct selection

use environment variables

Explanation

Using environment variables is another common method to provide credentials to Terraform. By setting environment variables, Terraform can access the necessary credentials without exposing them directly in the configuration files.

Correct selection

directly in the provider block by hardcoding or using a variable

Explanation

Storing credentials directly in the provider block is a common way to provide access to the underlying platform's API. This can be done by hardcoding the credentials or using Terraform variables to keep them secure and manageable.

using the resources block in your configuration

Explanation

The resources block in Terraform configuration files is used to define the infrastructure components that Terraform manages, not for providing credentials. Storing credentials in the resources block is not a recommended practice for security reasons.

Correct selection

integrated services, such as AWS IAM or Azure Managed Service Identity

Explanation

Integrated services like AWS IAM or Azure Managed Service Identity can be utilized to provide credentials to Terraform. These services offer secure and managed ways to access the API of the underlying platform without exposing sensitive information in the Terraform configuration.

Overall explanation

You can use methods such as static credentials, environment variables, share credentials/configuration file, or other methods. For example, the AWS provider can use many different options as seen here:

https://registry.terraform.io/providers/hashicorp/aws/latest/docs#authentication

Each provider is different, and you should check the documentation to see what is supported for each one you want to use.

Domain

Objective 3 - Understand Terraform Basics

Question 20Skipped

Aaron is new to Terraform and has a single configuration file ready for deployment. What can be true about this configuration file? (select three)

Correct selection

the configuration file can deploy both QA and Staging infrastructure for applications

Correct selection

the state file can be stored in Azure but provision applications in AWS

Correct selection

Aaron's configuration file can deploy applications in both AWS and GCP

the state can be disabled when deploying to multiple clouds to prevent sensitive data from being shared across cloud platforms

Overall explanation

There are a ton of benefits of deploying with Terraform and the solution is very capable of managing deployments across multiple clouds. However, state is still required and cannot be disabled.

https://developer.hashicorp.com/terraform/intro/use-cases#multi-cloud-deployment

Domain

Objective 2 - Understand Terraform's Purpose (vs other IAC)

Question 21Skipped

Given the following snippet of code, what will the value of the "Name" tag equal after a terraform apply?

```
variable "name" {
  description = "The username assigned to the infrastructure"
  default = "data_processing"
}

variable "team" {
  description = "The team responsible for the infrastructure"
  default = "IS Team"
}

locals {
  name  = (var.name != "" ? var.name : random_id.id.hex)
  owner = var.team
  common_tags = {
    Owner = local.owner
    Name  = local.name
```

17. }
18. }

an empty string

Correct answer

data_processing

IS Team

a random hex value

Overall explanation

The syntax of a conditional expression first names the condition. In this example, if var.name is not (!=) empty, assign the var.name value; else, assign the new random_id resource as the name value. Since var.name equals **data_processing**, then the value of Name will equal data_processing.

https://developer.hashicorp.com/terraform/language/expressions/conditionals

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 22Skipped

Which feature of Terraform Cloud can be used to enforce fine-grained policies to enforce standardization and cost controls before resources are provisioned with Terraform?

workspaces

private registry

Correct answer

sentinel

remote runs

Overall explanation

Sentinel is an embedded policy-as-code framework integrated with the HashiCorp Enterprise products. It enables fine-grained, logic-based policy decisions and can be extended to use information from external sources.

Please Note: HashiCorp announced at HashiConf Global '22 that Open Policy Agent (OPA) is now supported in Terraform Cloud. However, this new feature will likely take a while to appear in the actual Terraform exam. I'm already working with HashiCorp to address any conflicts that might appear on the real exam.

https://developer.hashicorp.com/terraform/cloud-docs/policy-enforcement

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 23Skipped

There are multiple ways to provide sensitive values when using Terraform. However, sensitive information provided in your configuration can be written to the state file, which is not desirable. Which method below will not result in sensitive information being written to the state file?

using a declared variable

Correct answer

none of the above

using a tfvars file

retrieving the credentials from a data source, such as HashiCorp Vault

Overall explanation

When using sensitive values in your Terraform configuration, all of the configurations mentioned above will result in the sensitive value being written to the state file. Terraform stores the state as plain text, including variable values, even if you have flagged them as sensitive. Terraform needs to store these values in your state so that it can tell if you have changed them since the last time you applied your configuration.

Terraform runs will receive the full text of sensitive variables and might print the value in logs and state files if the configuration pipes the value through to an output or a resource parameter. Additionally, Sentinel mocks downloaded from runs will contain the sensitive values of Terraform (but not environment) variables. Take care when writing your configurations to avoid unnecessary credential disclosure. (Environment variables can end up in log files if TF_LOG is set to TRACE.)

https://developer.hashicorp.com/terraform/cloud-docs/workspaces/variables#sensitive-values

https://developer.hashicorp.com/terraform/tutorials/configuration-language/sensitive-variables

Domain

Objective 7 - Implement and Maintain State

Question 24Skipped

Teddy is using Terraform to deploy infrastructure using modules. Where is the module below stored?

```
module "monitoring_tools" {
  source = "./modules/monitoring_tools"

  cluster_hostname = module.k8s_cluster.hostname
}
```

in a public GitLab repository

Correct answer

locally on the instance running Terraform

on the Terraform public registry

a private registry in Terraform Cloud (free)

Overall explanation

A local path must begin with either ./ or ../ to indicate that a local path is intended, to distinguish from a registry address.

https://developer.hashicorp.com/terraform/language/modules/sources#terraform-registry

Domain

Objective 5 - Interact with Terraform Modules

Question 25Skipped

True or False? Terraform is designed to work only with public cloud platforms, and organizations that wish to use it for on-premises infrastructure (private cloud) should look for an alternative solution.

True

Correct answer

False

Overall explanation

Terraform is designed to work with almost any infrastructure that provides an API. Terraform is very frequently used to provision infrastructure atop VMware infrastructure, along with traditional, physical security or infrastructure service solutions.

Additional information can be found in this article
- https://www.hashicorp.com/blog/infrastructure-as-code-in-a-private-or-public-cloud

Domain

Objective 1 - Understand Infrastructure as Code concepts

Question 26Skipped

True or False? A terraform plan is a required step before running a terraform apply?

Correct answer

False

True

Overall explanation

If no explicit plan file is given on the command line, terraform apply will create a new plan automatically and prompt for approval to apply it

https://developer.hashicorp.com/terraform/intro/core-workflow

Domain

Objective 6 - Use the Core Terraform Workflow

Question 27Skipped

When a terraform apply is executed, where is the AWS provider retrieving credentials to create cloud resources in the code snippet below?

```
provider "aws" {
  region     = us-east-1
  access_key = data.vault_aws_access_credentials.creds.access_key
  secret_key = data.vault_aws_access_credentials.creds.secret_key
}
```

from the .tfvars file called vault

Correct answer

from a data source that is retrieving credentials from HashiCorp Vault. Vault is dynamically generating the credentials on Terraform's behalf.

from a script that is executing commands against Vault

from a variable called vault_aws_access_credentials

Overall explanation

In this case, Terraform is using a data source to gather credentials from Vault. The data block would look something like this:

```
data "vault_aws_access_credentials" "creds" {
  backend = vault_aws_secret_backend.aws.path
  role    = vault_aws_secret_backend_role.role.name
}
```

https://developer.hashicorp.com/terraform/language/data-sources

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 28Skipped

There are an endless number of benefits to using Terraform within your organization. Which of the following are true statements regarding Terraform? (select three)

Correct selection

Terraform can simplify both management and orchestration of deploying large-scale, multi-cloud infrastructure

Correct selection

A single Terraform configuration file can be used to manage multiple providers

Terraform can manage dependencies within a single cloud, but not cross-cloud

Correct selection

Terraform is cloud-agnostic but requires a specific provider for the cloud platform

Overall explanation

All of the answers are benefits to using Terraform, except Terraform can manage dependencies within a single cloud, but not cross-cloud. Terraform isn't limited to ***only*** managing dependencies for a single cloud, it can manage dependencies across multiple cloud providers as well.

https://developer.hashicorp.com/terraform/intro/use-cases#multi-cloud-deployment

Domain

Objective 2 - Understand Terraform's Purpose (vs other IAC)

Question 29Skipped

Michael has deployed many resources in AWS using Terraform and can easily update or destroy resources when required by the application team. A new employee, Dwight, is working with the application team and deployed a new EC2 instance through the AWS console. When Michael finds out, he decided he wants to manage the new EC2 instance using Terraform moving forward. He opens his terminal and types:

$ terraform import aws_instance.web_app_42 i-b54a26b28b8acv7233

However, Terraform returns the following error: Error: resource address "aws_instance.web_app_42" does not exist in the configuration.

What does Michael need to do first in order to manage the new Amazon EC2 instance with Terraform?

import the configuration of the EC2 instance called web_app_42 from AWS first

configure the appropriate tags on the Amazon EC2 resource so Terraform knows that it should manage the resource moving forward

Terraform cannot manage resources that were provisioned manually

Correct answer

create a configuration for the new resource in the Terraform configuration file, such as:

1. resource "aws_instance" "web_app_42" {
2. # (resource arguments)
3. }

Overall explanation

The terraform import command is used to import existing resources into Terraform. However, Terraform will not create a configuration for the imported resource. The Terraform operator must create/add a configuration for the resource that will be imported first. Once the configuration is added to the configuration file, the terraform import command can be executed to manage the resource using Terraform.

https://developer.hashicorp.com/terraform/cli/commands/import

Domain

Objective 4 - Use Terraform Outside of Core Workflow

Question 30Skipped

Ralphie has executed a terraform apply using a complex Terraform configuration file. However, a few resources failed to deploy due to incorrect variables. After the error is discovered, what happens to the resources that were successfully provisioned?

Correct answer

the resources that were successfully provisioned will remain as deployed

Terraform deletes the resources on the next run

Terraform rolls back the configuration due to the error, therefore the resources are automatically destroyed

resources successfully deployed are marked for replacement

Overall explanation

During a terraform apply, any resources that are successfully provisioned are maintained as deployed.

On the other hand, resources that failed during the provisioning process, such as a provisioned, will be tainted to be recreated during the next run. https://developer.hashicorp.com/terraform/language/resources/provisioners/syntax#creation-time-provisioners

Domain

Objective 7 - Implement and Maintain State

Question 31Skipped

Variables and their default values are typically declared in a main.tf or variables.tf file. What type of file can be used to set explicit values for the current working directory that will override the default variable values?

.sh file

.tfstate file

.txt file

Correct answer

.tfvars file

Overall explanation

To set lots of variables, it is more convenient to specify their values in a *variable definitions file* (with a filename ending in either .tfvars or .tfvars.json)

https://developer.hashicorp.com/terraform/language/values/variables

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 32Skipped

Both Terraform CLI and Terraform Cloud offer a feature called "workspaces". Which of the following statements are true regarding workspaces? (select three)

Correct selection

Terraform Cloud manages infrastructure collections with a workspace whereas CLI manages collections of infrastructure resources with a persistent working directory

Run history is logged in a file underneath the working directory of a CLI workspace

Correct selection

Terraform Cloud maintains the state version and run history for each workspace

Each CLI workspace coincides with a different VCS repo

Correct selection

CLI workspaces are alternative state files in the same working directory

Overall explanation

Workspaces are similar concepts in all versions of Terraform, although they behave differently depending on the platform they are being used on.

https://developer.hashicorp.com/terraform/cloud-docs/workspaces

https://developer.hashicorp.com/terraform/language/state/workspaces

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 33Skipped

What function does the terraform init -upgrade command perform?

upgrades all of the referenced modules and providers to the latest version of Terraform

Correct answer

update all previously installed plugins and modules to the newest version that complies with the configuration's version constraints

upgrades the Terraform configuration file(s) to use the referenced Terraform version

upgrades the backend to the latest supported version

Overall explanation

The -upgrade will upgrade all previously-selected plugins an modules to the newest version that complies with the configuration's version constraints. This will cause Terraform to ignore any selections recorded in the dependency lock file, and to take the newest available version matching the configured version constraints.

https://developer.hashicorp.com/terraform/cli/commands/init#upgrade-1

Domain

Objective 6 - Use the Core Terraform Workflow

Question 34Skipped

True or False? A main.tf file is always required when using Terraform?

True

Correct answer

False

Overall explanation

Although main.tf is the standard name, it's not necessarily required. Terraform will look for any file with a .tf or .tf.json extension when running terraform commands.

https://developer.hashicorp.com/terraform/language#code-organization

Domain

Objective 3 - Understand Terraform Basics

Question 35Skipped

Based on the following code, which code block will create a resource first?

```
resource "aws_instance" "data_processing" {
  ami           = data.aws_ami.amazon_linux.id
  instance_type = "t2.micro"

  depends_on = [aws_s3_bucket.customer_data]
}

module "example_sqs_queue" {
  source  = "terraform-aws-modules/sqs/aws"
```

```
10.  version = "2.1.0"
11.
12.  depends_on = [aws_s3_bucket.customer_data, aws_instance.data_processing]
13. }
14.
15. resource "aws_s3_bucket" "customer_data" {
16.  acl = "private"
17. }
18.
19. resource "aws_eip" "ip" {
20.  vpc      = true
21.  instance = aws_instance.data_processing.id
22. }
```

example_sqs_queue

aws_instance.data_processing

Correct answer

aws_s3_bucket.customer_data

aws_eip.ip

Overall explanation

In this example, the only resource that does not have an implicit or an explicit dependency is the aws_s3_bucket.customer_data. Every other resource defined in this configuration has a dependency on another resource.

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

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 36Skipped

Which of the following Terraform CLI commands are valid? (select five)

Correct selection

$ terraform show

Correct selection

$ terraform fmt

Correct selection

terraform apply -refresh-only

Correct selection

$ terraform workspace select

Correct selection

$ terraform login

$ terraform delete

$ terraform initialize

Overall explanation

terraform delete and terraform initialize are not valid Terraform CLI commands.

Correct Answers:

The terraform apply -refresh-only command creates a plan whose goal is only to update the Terraform state and any root module output values to match changes made to remote objects outside of Terraform.

he terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style.

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

The terraform show command is used to provide human-readable output from a state or plan file. This can be used to inspect a plan to ensure that the planned operations are expected, or to inspect the current state as Terraform sees it.

The terraform login command can be used to automatically obtain and save an API token for Terraform Cloud, Terraform Enterprise, or any other host that offers Terraform services.

https://developer.hashicorp.com/terraform/cli/commands/fmt

Domain

Objective 4 - Use Terraform Outside of Core Workflow

Question 37Skipped

Which of the following are the benefits of using *modules* in Terraform? (select three)

allows modules to be stored anywhere accessible by Terraform

Correct selection

enables code reuse

Correct selection

supports versioning to maintain compatibility

Correct selection

supports modules stored locally or remotely

Overall explanation

All of these are examples of the benefits of using Terraform modules except where they can be stored. Modules can only be supported in certain sources found at the following link:

https://developer.hashicorp.com/terraform/language/modules/sources

Domain

Objective 5 - Interact with Terraform Modules

Question 38Skipped

You have a Terraform configuration file with no defined resources. However, there is a related state file for resources that were created on AWS. What happens when you run a *terraform apply*?

Correct answer

Terraform will destroy all of the resources

Terraform will not perform any operations.

Terraform will scan the AWS infrastructure and create a new configuration file based on the state file.

Terraform will produce an error since there are no resources defined

Overall explanation

In this case, since there is a state file with resources, Terraform will match the desired state of no resources since the configuration file doesn't include any resources. Therefore, all resources defined in the state file will be destroyed.

https://developer.hashicorp.com/terraform/language/state/purpose

Domain

Objective 7 - Implement and Maintain State

Question 39Skipped

You have deployed your production environment with Terraform, and a developer has requested that you update a load balancer configuration to improve its compatibility with their application. Once you have modified your Terraform configuration, how can you conduct a dry run to verify that no unexpected changes will be made?

Correct answer

run terraform plan and inspect the proposed changes

run terraform console and validate the result of any expressions

run terraform plan -auto-approve to push the changes

run terraform state list to view the existing resources and ensure they won't conflict with the new changes

Overall explanation

The ultimate goal of a terraform plan is to compare the configuration file against the current state and propose any changes needed to apply the desired configuration.

https://developer.hashicorp.com/terraform/cli/commands/plan

Domain

Objective 6 - Use the Core Terraform Workflow

Question 40Skipped

What happens when you apply a Terraform configuration using terraform apply? (select two)

Correct selection

Terraform updates the state file with configuration changes made during the execution

Terraform downloads any required plugins

Terraform recreates all the infrastructure defined in the configuration file

Correct selection

Terraform makes infrastructure changes defined in your configuration.

Terraform formats your configuration to the standard canonical format and style

Overall explanation

The terraform apply command is used to apply the changes required to reach the desired state of the configuration, or the pre-determined set of actions generated by a terraform plan execution plan.

https://developer.hashicorp.com/terraform/cli/commands/apply

Domain

Objective 6 - Use the Core Terraform Workflow

Question 41Skipped

You have created a new workspace for a project and added all of your Terraform configuration files in the new directory. Before you execute a terraform plan, you want to validate the configuration using the terraform validate command. However, Terraform returns the error:

1. $ terraform validate
2. Error: Could not load plugin

What would cause this error when trying to validate the configuration?

Correct answer

the directory was not initialized

the credentials for the provider are invalid

the configuration is invalid

the directory does not contain valid Terraform configuration files

Overall explanation

terraform validate requires an initialized working directory with any referenced plugins and modules installed. If you don't initiate the directory, you will get an error stating you need to run a terraform init

https://developer.hashicorp.com/terraform/cli/commands/validate

Domain

Objective 6 - Use the Core Terraform Workflow

Question 42Skipped

Infrastructure as Code (IaC) provides many benefits to help organizations deploy application infrastructure much faster than clicking around in the console. What are the additional benefits of IaC? (select three)

eliminates parallelism

Correct selection

allows infrastructure to be versioned

can always be used to deploy the latest features and services

Correct selection

code can easily be shared and reused

Correct selection

creates a blueprint of your data center

Overall explanation

Infrastructure is described using a high-level configuration syntax. This allows a blueprint of your datacenter to be versioned and treated as you would any other code. Additionally, infrastructure can be shared and re-used.

Infrastructure as Code almost always uses parallelism to deploy resources faster. And depending on the solution being used, it doesn't always have access to the latest features and services available on cloud platforms or other solutions.

https://developer.hashicorp.com/terraform/intro#infrastructure-as-code

Domain

Objective 1 - Understand Infrastructure as Code concepts

Question 43Skipped

AutoPlants, Inc is a new startup that uses AI and robotics to grow sustainable and organic vegetables for California farmer's markets. The organization can quickly burst into the public cloud during the busy season using Terraform to provision additional resources to process AI computations and images. Since its compute stack is proprietary and critical to the organization, it needs a solution to create and publish Terraform modules that only its engineers and architects can use.

Which feature can provide this functionality?

Terraform Cloud Workspaces

Correct answer

Private Registry

Terraform Registry

HashiCorp Sentinel

Overall explanation

One feature that can provide this functionality is Terraform's **Private Registry**. This feature allows organizations to create and manage private modules that authorized users within the organization can only access. With Private Registry, AutoPlants, Inc can create and publish Terraform modules that are only available to its engineers and architects. This ensures that their proprietary compute stack remains secure while also streamlining the process of provisioning additional resources during the busy season.

https://developer.hashicorp.com/terraform/cloud-docs/registry

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 44Skipped

Fill in the correct answers below:

Infrastructure as Code (IaC) makes infrastructure changes _____, _____, _____, and _____. (select four)

Correct selection

repeatable

Correct selection

idempotent

Correct selection

predictable

Correct selection

consistent

highly-available

Overall explanation

IaC makes changes idempotent, consistent, repeatable, and predictable. Without IaC, scaling up infrastructure to meet increased demand may require an operator to remotely connect to each machine and then manually provision and configure many servers by executing a series of commands/scripts. They might open multiple sessions and move between screens, which often results in skipped steps or slight variations between how work is completed, necessitating rollbacks. Perhaps a command was run incorrectly on one instance and reverted before being re-run correctly.

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

Domain

Objective 1 - Understand Infrastructure as Code concepts

Question 45Skipped

Philip works at a payment processing company and manages the organization's VMware environment. He recently provisioned a new cluster for a production environment. To ensure everything is working as expected, Philip has been using Terraform and the VMware vSphere client to create and destroy new virtual machines. Currently, there are three virtual machines running on the new cluster, so Philip runs terraform destroy to remove the remaining virtual machines from the cluster. However, Terraform only removes two of the virtual machines, leaving one virtual machine still running.

Why would Terraform only remove two of the three virtual machines?

Correct answer

the remaining virtual machine was not created by Terraform, therefore Terraform is not aware of the virtual machine and cannot destroy it

the vSphere provider credentials are invalid, and therefore Terraform cannot reach the third virtual machine

Terraform can only destroy a maximum of 2 resources per terraform destroy execution

the virtual machine was marked with vSphere tags to prevent it from being destroyed

Overall explanation

The terraform destroy command terminates resources defined in your Terraform configuration. This command is the reverse of terraform apply in that it terminates all the resources specified by the configuration. It does *not* destroy resources running elsewhere that are not described in the current configuration.

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

Domain

Objective 6 - Use the Core Terraform Workflow

Question 46Skipped

Given a Terraform config that includes the following code, how would you reference the last instance that will be created?

```
resource "aws_instance" "database" {
  # ...
  for_each = {
    "vault": "secrets",
    "terraform": "infrastructure",
    "consul": "networking",
    "nomad": "scheduler"
  }
}
```

aws_instance.database[4]

aws_instance.database[2]

Correct answer

aws_instance.database["nomad"]

aws_instance.nomad

Overall explanation

The following specifications apply to index values on modules and resources with multiple instances:

[N] where N is a 0-based numerical index into a resource with multiple instances specified by the count meta-argument. Omitting an index when addressing a resource where count > 1 means that the address references all instances.

["INDEX"] where INDEX is an alphanumerical key index into a resource with multiple instances specified by the for_each meta-argument.

https://developer.hashicorp.com/terraform/language/meta-arguments/for_each#referring-to-instances

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 47Skipped

Which of the following is not a benefit of Terraform state?

provides a one-to-one mapping of the configuration to real-world resources

determines the dependency order for deployed resources

increases performance by reducing the requirement to query multiple resources at once

Correct answer

reduces the amount of outbound traffic by requiring that state is stored locally

Overall explanation

Terraform state is required and there are many benefits that are outlined in this documentation:

https://developer.hashicorp.com/terraform/language/state/purpose

Domain

Objective 2 - Understand Terraform's Purpose (vs other IAC)

Question 48Skipped

You are working with a cloud provider to deploy resources using Terraform. You've added the following data block to your configuration. When the data block is used, what data will be returned?

```
data "aws_ami" "amzlinux2" {
  most_recent = true
  owners      = ["amazon"]

  filter {
    name   = "name"
    values = ["amzn2-ami-hvm-*-x86_64-ebs"]
  }
}
```

```
resource "aws_instance" "vault" {
  ami                    = data.aws_ami.amzlinux2.id
  instance_type          = "t3.micro"
  key_name               = "vault-key"
  vpc_security_group_ids = var.sg
  subnet_id              = var.subnet
  associate_public_ip_address = "true"
  user_data              = file("vault.sh")

```

```
10.  tags = {
11.    Name = "vault"
12.  }
13. }
```

the latest AMI you have previously used for an Amazon Linux 2 image

the IP address of an EC2 instance running in AWS

Correct answer

all possible data of a specific Amazon Machine Image(AMI) from AWS

a custom AMI for Amazon Linux 2

Overall explanation

When you add a data block to your configuration, Terraform will retrieve all of the available data for that particular resource. It is then up to you to reference a specific attribute that can be exported from that data source. For example, if you include a data block for the aws_ami resource, Terraform will get a ton of attributes about that AMI that you can use elsewhere in your code - check out this link to see the list of attributes specific to the aws_ami, for example. https://registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/ami#attributes-reference

Within the block body (between { and }) are query constraints defined by the data source. Most arguments in this section depend on the data source, and indeed in this example most_recent, owners and tags are all arguments defined specifically for the aws_ami data source.

https://developer.hashicorp.com/terraform/language/data-sources#using-data-sources

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 49Skipped

What is the primary function of Terraform Cloud agents?

store and manage Terraform state files

provide remote access to Terraform workspaces

Correct answer

execute Terraform plans and apply changes to infrastructure

monitor and troubleshoot Terraform deployments

Overall explanation

Terraform Cloud agents are lightweight programs deployed within your target infrastructure environment. Their primary function is to receive Terraform plans from Terraform Cloud, execute those plans locally, and apply the desired infrastructure changes. This allows you to manage private or on-premises infrastructure with Terraform Cloud without opening your network to public ingress traffic.

****

WRONG ANSWERS:

- **Provide remote access to Terraform workspaces:** While agents can facilitate communication between your infrastructure and Terraform Cloud, they don't directly provide remote access to workspaces themselves.
- **Store and manage Terraform state files:** While agents can interact with state files during execution, their primary role is not state management. State files are typically stored in a separate, centralized location like Terraform Cloud or a cloud storage service.
- **Monitor and troubleshoot Terraform deployments:** While agents can be used to gather information for troubleshooting, their core function is not dedicated monitoring.

https://developer.hashicorp.com/terraform/cloud-docs/agents

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 50Skipped

Pam just finished up a new Terraform configuration file and has successfully deployed the configuration on Azure using Terraform open-source. After confirming the configuration on Azure, Pam changes to a new workspace and then heads to lunch. When she arrives back at her desk, Pam decides to destroy the resources to save on cost. When Pam executes a terraform destroy, the output indicates there are no resources to delete.

Why can't Pam delete the newly created resources in Azure?

1. $ terraform destroy
2.
3. An execution plan has been generated and is shown below.
4. Resource actions are indicated with the following symbols:
5.
6. Terraform will perform the following actions:
7.
8. Plan: 0 to add, 0 to change, 0 to destroy.

Correct answer

there is no Terraform state in the current workspace she is working in

an Azure administrator manually deleted the resources

the Terraform state was deleted when she created the new workspace

Terraform reached the maximum timeout while Pam was away from lunch, therefore the resources were automatically destroyed

Overall explanation

Workspaces isolate their state, so if Pam runs a terraform destroy, Terraform will not see any existing state for this configuration. Pam may use the command terraform workspace select <name> to choose the original workspace where the Azure resources were provisioned in order to properly destroy them in Azure.

https://developer.hashicorp.com/terraform/cli/workspaces

Domain

Objective 4 - Use Terraform Outside of Core Workflow

Question 51Skipped

Based on the Terraform code below, what block type is used to define the VPC?

1. vpc_id = aws_vpc.main.id
2. ...

Correct answer

resource block

data block

provider block

locals block

Overall explanation

Based on the Terraform code provided in the question, the VPC is defined in a resource block, meaning that there is a VPC resource being defined, such as:

1. resource "aws_vpc" "main" {
2. cidr_block = var.base_cidr_block
3. }

If it were locals, the resource would be referred to as local.aws_vpc

If it were in a data block, it would be referred to as data.aws_vpc.i.main.id

https://developer.hashicorp.com/terraform/language/resources

Domain

Objective 3 - Understand Terraform Basics

Question 52Skipped

True or False? Any sensitive values referenced in the Terraform code, even as variables, will end up in plain text in the state file.

False

Correct answer

True

Overall explanation

Any values that are retrieved in a data block or referenced as variables will show up in the state file.

https://developer.hashicorp.com/terraform/language/state/sensitive-data

Domain

Objective 7 - Implement and Maintain State

Question 53Skipped

Your organization requires that no security group in your public cloud environment includes "0.0.0.0/0" as a source of network traffic. How can you proactively enforce this control and prevent Terraform configurations from being executed if they contain this specific string?

Configure a rule in your public cloud provider to scan for security groups and alert you if a security group contains the string

Correct answer

Create a Sentinel or OPA policy that checks for the string and denies the Terraform apply if the string exists

Configure the user's TFC organization permissions to not allow any variables or Terraform configuration that contain this string

Perform a terraform validate on the local workstation before committing the code to the code repository linked to TFC workspace

Overall explanation

To prevent a Terraform configuration from being executed if it contains a specific string, you can use Sentinel or Open Policy Agent (OPA) to enforce policy checks. Both tools allow you to define custom policies to evaluate and control Terraform configurations before they are applied. Both offer powerful capabilities to enforce custom policies on your Terraform configurations, providing an additional layer of security and governance. By leveraging these tools, you can prevent sensitive information or undesired strings from being present in your infrastructure code, reducing the risk of accidental misconfigurations and potential security vulnerabilities.

Wrong Answers:

- there are no user settings in TFC that would prevent the use of a specific string in your Terraform configuration

- a terraform validate would not prevent specific strings from being used in Terraform

- while you could use another service to manage your environment and scan for security groups that permit "0.0.0.0/0" in your environment, it is a REACTIVE control, and not preventative.

https://developer.hashicorp.com/terraform/cloud-docs/policy-enforcement/sentinel

https://developer.hashicorp.com/terraform/cloud-docs/policy-enforcement/opa

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 54Skipped

Margaret is calling a child module to deploy infrastructure for her organization. Just as a good architect does (and suggested by HashiCorp), she specifies the module version she wants to use even though there are newer versions available. During a terrafom init, Terraform downloads v0.0.5 just as expected.

What would happen if Margaret removed the version parameter in the module block and ran a terraform init again?

```
1. module "consul" {
2.   source  = "hashicorp/consul/aws"
3.   version = "0.0.5"
4.
5.   servers = 3
6. }
```

Terraform would skip the module

Correct answer

Terraform would use the existing module already downloaded

Terraform would download the latest version of the module

Terraform would return an error, as the version parameter is required

Overall explanation

When using modules installed from a registry, HashiCorp recommends explicitly constraining the acceptable version numbers to avoid unexpected or unwanted changes.
The version argument accepts a version constraint string. Terraform will use the newest

installed version of the module that meets the constraint; if no acceptable versions are installed, it will download the newest version that meets the constraint.

A version number that meets every applicable constraint is considered acceptable.

Terraform consults version constraints to determine whether it has acceptable versions of itself, any required provider plugins, and any required modules. For plugins and modules, it will use the newest installed version that meets the applicable constraints.

To test this, I ran a terraform init with the code as shown in the file:

1. $ terraform init
2. Initializing modules...
3. Downloading hashicorp/consul/aws 0.0.5 for consul...
4. - consul in .terraform\modules\consul
5. - consul.consul_clients in .terraform\modules\consul\modules\consul-cluster
6. - consul.consul_clients.iam_policies in .terraform\modules\consul\modules\consul-iam-policies
7. - consul.consul_clients.security_group_rules in .terraform\modules\consul\modules\consul-security-group-rules
8. - consul.consul_servers in .terraform\modules\consul\modules\consul-cluster
9. - consul.consul_servers.iam_policies in .terraform\modules\consul\modules\consul-iam-policies
10. - consul.consul_servers.security_group_rules in .terraform\modules\consul\modules\consul-security-group-rules

Then I removed the constraint from the configuration file and ran a terraform init again:

1. $ terraform init
2. Initializing modules...
3.
4. Initializing the backend...
5.
6. Initializing provider plugins...
7. - Reusing previous version of hashicorp/aws from the dependency lock file
8. - Reusing previous version of hashicorp/template from the dependency lock file

Terraform did not download a newer version of the module. It reused the existing one.

https://developer.hashicorp.com/terraform/language/modules/syntax#version

https://developer.hashicorp.com/terraform/language/expressions/version-constraints

Domain

Objective 5 - Interact with Terraform Modules

Question 55Skipped

Terraform Cloud provides organizations with many features not available to those running Terraform open-source to deploy infrastructure. Select the ADDITIONAL features that organizations can take advantage of by moving to Terraform Cloud. (select three)

Correct selection

VCS connection

providers

Terraform registry

Correct selection

remote runs

Correct selection

private registry

Overall explanation

Terraform Cloud offers many features, even in the free version, that organizations can quickly take advantage of. This is the best table that compares the features available in Terraform OSS vs. Terraform Cloud and Terraform Enterprise.

https://www.datocms-assets.com/2885/1602500234-terraform-full-feature-pricing-tablev2-1.pdf

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 56Skipped

Using the Terraform code below, where will the resource be provisioned?

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

provider "aws" {
  alias  = "west"
  region = "us-west-2"
}
```

```
9.
10. provider "aws" {
11.   alias  = "eu"
12.   region = "eu-west-2"
13. }
14.
15. resource "aws_instance" "vault" {
16.   ami                  = data.aws_ami.amzlinux2.id
17.   instance_type            = "t3.micro"
18.   key_name               = "krausen_key"
19.   vpc_security_group_ids     = var.vault_sg
20.   subnet_id              = var.vault_subnet
21.   user_data              = file("vault.sh")
22.
23.   tags = {
24.     Name = "vault"
25.   }
26. }
```

Correct answer

us-east-1

us-west-1

us-west-2

Overall explanation

The resource above will be created in the default region of us-east-1, since the resource does signify an alternative provider configuration. If the resource needs to be created in one of the other declared regions, it should have looked like this, where "aws" signifies the provider name and "west" signifies the alias name as such <PROVIDER NAME>.<ALIAS>:

```
1.  resource "aws_instance" "vault" {
2.    provider               = aws.west
3.    ami                  = data.aws_ami.amzlinux2.id
```

```
4.  instance_type           = "t3.micro"
5.  key_name                = "krausen_key"
6.  vpc_security_group_ids  = var.vault_sg
7.  subnet_id               = var.vault_subnet
8.  user_data               = file("vault.sh")
9.
10. tags = {
11.   Name = "vault"
12. }
13. }
```

https://developer.hashicorp.com/terraform/language/providers/configuration#selecting-alternate-provider-configurations

Domain

Objective 3 - Understand Terraform Basics

Question 57Skipped

Scenario: You are deploying a new application and want to deploy it to multiple AWS regions within the same configuration file. Which of the following features will allow you to configure this?

a provider with multiple versions defined

using the default provider along with a single defined provider

Correct answer

multiple provider blocks using an alias

one provider block that defines multiple regions

Overall explanation

You can optionally define multiple configurations for the same provider, and select which one to use on a per-resource or per-module basis. The primary reason for this is to support multiple regions for a cloud platform; other examples include targeting multiple Docker hosts, multiple Consul hosts, etc.

https://developer.hashicorp.com/terraform/language/providers/configuration#alias-multiple-provider-configurations

Domain

Objective 3 - Understand Terraform Basics