Question 1Skipped

You need to enable logging for Terraform and persist the logs to a specific file. What two environment variables can be set to enable logs and write them to a file? (select two)

TF_ENABLE_LOG=true

TF_LOG_OUTPUT="<file_path>"

Correct selection

TF_LOG=TRACE

Correct selection

TF_LOG_PATH="<file_path>"

Overall explanation

Terraform has detailed logs which can be enabled by setting the TF_LOG environment variable to any value. This will cause detailed logs to appear on stderr. You can set TF_LOG to one of the log levels TRACE, DEBUG, INFO, WARN or ERROR to change the verbosity of the logs, with TRACE being the most verbose.

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.

WRONG ANSWERS:

The wrong answers provided in this question are not valid environment variables that you can use with Terraform.

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

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 2Skipped

Beyond storing state, what capability can an enhanced storage backend, such as the remote backend, provide your organization?

replicate your state to a secondary location for backup

allow multiple people to execute operations on the state file at the same time

provides versioning capabilities on your state file in the event it becomes corrupted

Correct answer

execute your Terraform on infrastructure either locally or in Terraform Cloud

Overall explanation

Using an enhanced storage backend allows you to execute your Terraform on infrastructure either locally or in Terraform Cloud. Note that this ***enhanced storage backend*** term has now been deprecated by Terraform but it's likely to show up in the test for a while. See the note below from this site:

Note: *In Terraform versions prior to 1.1.0, backends were also classified as being 'standard' or 'enhanced', where the latter term referred to the ability of the remote backend to store state and perform Terraform operations. This classification has been removed, clarifying the primary purpose of backends. Refer to Using Terraform Cloud for details about how to store state, execute remote operations, and use Terraform Cloud directly from Terraform.*

https://developer.hashicorp.com/terraform/language/settings/backends/configuration#what-backends-do

Domain

Objective 7 - Implement and Maintain State

Question 3Skipped

True or False? You can use a combination of Terraform Cloud's cost estimation feature and Sentinel policies to ensure your organization doesn't apply changes to your environment that would result in exceeding your monthly operating budget.

False

Correct answer

True

Overall explanation

Terraform Cloud provides cost estimates for many resources found in your Terraform configuration. For each resource, an hourly and monthly cost is shown, along with the monthly delta. The total cost and delta of all estimable resources is also shown.

Since Sentinel policies are run AFTER cost estimation, you can take cost into account when evaluating Sentinel policies before the run is executed.

https://developer.hashicorp.com/terraform/cloud-docs/cost-estimation#verifying-costs-in-policies

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 4Skipped

What is ***NOT*** a benefit of using Infrastructure as Code?

the reduction of misconfigurations that could lead to security vulnerabilities and unplanned downtime

Correct answer

reducing vulnerabilities in your publicly-facing applications

the ability to programmatically deploy infrastructure

your infrastructure configurations can be version controlled and stored in a code repository alongside the application code

Overall explanation

Although Infrastructure as Code (IaC) tools allow you to programmatically deploy and manage your applications, it does NOT ensure that your applications have a reduced number of vulnerabilities. This security feature is not the responsibility of IaC, and you would need to pair IaC with another tool to scan your code to identify security vulnerabilities.

WRONG ANSWERS:

All of the wrong answers in this question are actually the primary use cases of Infrastructure as Code tools.

Infrastructure as code (IaC) tools allow you to manage infrastructure with configuration files rather than through a graphical user interface. IaC allows you to build, change, and manage your infrastructure in a safe, consistent, and repeatable way by defining resource configurations that you can version, reuse, and share.

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

Domain

Objective 1 - Understand Infrastructure as Code Concepts

Question 5Skipped

You have a configuration file that you've deployed to one AWS region already but you want to deploy the same configuration file to a second AWS region without making changes to the configuration file. What feature of Terraform can you use to accomplish this?

terraform import

terraform get

Correct answer

terraform workspace

terraform plan

Overall explanation

Workspaces should be used in this scenario to create separate state files for each regional deployment. When you use a workspace in Terraform OSS, you get a brand new state file to work with that is completely separate from the original. Therefore, you can modify environment variables or other values and use the same Terraform without negatively impacting resources that were deployed in any other workspace.

To create a new workspace, you can use the command terraform workspace new <name>

WRONG ANSWERS:

terraform plan - this command compares the current infrastructure against the desired state (configuration file) and proposes changes to your infrastructure. This is also commonly referred to as a dry run.

terraform get - this command is used to download modules

terraform import - this command can be used to import existing resources and pull them under Terraform management

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

Question 6Skipped

Which of the following are true statements regarding Terraform? (select three)

Correct selection

Terraform can orchestrate large-scale, multi-cloud infrastructure deployments

Correct selection

A single configuration file can use multiple providers

Correct selection

Terraform is cloud-agnostic

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

Overall explanation

Terraform can indeed manage dependencies across multiple cloud providers. That is a huge benefit of using Terraform since it's cloud-agnostic, it doesn't care where the resources are deployed. It can still manage implicit or explicit dependencies between resources regardless of where they are deployed.

https://learn.hashicorp.com/tutorials/terraform/dependencies?in=terraform/0-13

Domain

Objective 2 - Understand Terraform's purpose (vs other IaC)

Question 7Skipped

A child module created a new subnet for some new workloads. What Terraform block type would allow you to pass the subnet ID back to the parent module?

data block

resource block

Correct answer

output block

terraform block

Overall explanation

The resources defined in a module are encapsulated, so the calling module cannot access their attributes directly. However, the child module can declare output values to selectively export certain values to be accessed by the calling module.

https://developer.hashicorp.com/terraform/language/modules/syntax#accessing-module-output-values

Domain

Objective 5 - Interact with Terraform Modules

Question 8Skipped

Which of the following Terraform versions would be permitted to run the Terraform configuration based on the following code snippet?

```
terraform {
  required_version = "~> 1.0.0"
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
    random = {
      source  = "hashicorp/random"
      version = "3.1.0"
    }
  }
}
```

Correct answer

Terraform v1.0.5

Terraform v1.4.9

Terraform v1.1.0

Terraform v1.2.0

Overall explanation

When setting Terraform required_version or provider constraints, the ~ specifies that only the right-most version number can be incremented - therefore only v1.0.5 will satisfy the requirements.

https://developer.hashicorp.com/terraform/language/providers/requirements#best-practices-for-provider-versions

Domain

Objective 3 - Understand Terraform Basics

Question 9Skipped

Your colleague provided you with a Terraform configuration file and you're having trouble reading it because parameters and blocks are not properly aligned. What command can you run to quickly update the file configuration file to make it easier to consume?

Correct answer

terraform fmt

terraform workspace

terraform init

terraform state

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.

Other Terraform commands that generate Terraform configuration will produce configuration files that conform to the style imposed by terraform fmt, so using this style in your own files will ensure consistency.

The canonical format may change in minor ways between Terraform versions, so after upgrading Terraform we recommend to proactively run terraform fmt on your modules along with any other changes you are making to adopt the new version.

If you want to format ALL of your .tf files, you can use terraform fmt -recursive and it'll format all files in the current and all subdirectories.

WRONG ANSWERS:

terraform init - this is used to initialize and work with the Terraform backend terraform state - this command is for working with and viewing Terraform state

terraform workspace - this command is used to create and manage Terraform OSS workspaces

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

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 10Skipped

Both you and a colleague are responsible for maintaining resources that host multiple applications using Terraform CLI. What feature of Terraform helps ensure only a single person can update or make changes to the resources Terraform is managing?

local backend

version control

provisioners

Correct answer

state locking

Overall explanation

If supported by your backend, Terraform will lock your state for all operations that could write state. This prevents others from acquiring the lock and potentially corrupting your state.

State locking happens automatically on all operations that could write state. You won't see any message that it is happening. If state locking fails, Terraform will not continue. You can disable state locking for most commands with the -lock flag but it is not recommended.

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

Domain

Objective 7 - Implement and Maintain State

Question 11Skipped

Which statement below is ***true*** regarding using Sentinel in Terraform Enterprise?

Correct answer

Sentinel runs before a configuration is applied, therefore potentially reducing cost for public cloud resources

Sentinel can extend the functionality of user permissions in Terraform Enterprise

Sentinel policies can be developed using HCL, JSON, or YAML

Sentinel runs before each phase of the Terraform workflow, meaning a terraform init, terraform plan, and terraform apply

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.

When using Sentinel policies to define and enforce policies, it (Sentinel) runs after a terraform plan, but before a terraform apply. Therefore, you can potentially reduce costs on public cloud resources by NOT deploying resources that do NOT conform to policies enforced by Sentinel. For example, without Sentinel, your dev group might deploy instances that are too large, or too many of them, by accident or just because they can. Rather than being *REACTIVE* and shutting them down after they have been deployed, which it would cost you $, you can use Sentinel to prevent those large resources from being deployed in the first place.

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

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 12Skipped

You have infrastructure deployed with Terraform. A developer recently submitted a support ticket to update a security group to permit a new port. To satisfy the ticket, you update the Terraform configuration to reflect the changes and run a terraform plan. However, a co-worker has since logged into the console and manually updated the security group to the same configuration.

What will happen when you run a terraform apply?

the terraform apply command will require you to re-run the terraform plan command first

the security group will be changed back to the original configuration

Terraform will detect the drift and return an error.

Correct answer

Nothing will happen. Terraform will validate the infrastructure matches the desired state.

Overall explanation

A terraform apply will run its own state refresh and see the configuration matches the deployed infrastructure, so no changes will be made to the infrastructure.

WRONG ANSWERS:

Terraform will detect the drift and return an error - since terraform apply will refresh state, it will see that the configuration has changed and now meets the desired state, therefore it won't do anything.

the security group will be changed back to the original configuration - this won't happen because the Terraform configuration now states it should have the new port. If Terraform changed it back to the original configuration, the real-world resources would NOT match the desired state

***the terraform apply command will require you to re-run the terraform plan command first* -** terraform plan is not a requirement of the terraform apply command, so this won't happen

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

Domain

Objective 6 - Use the Core Terraform Workflow

Question 13Skipped

You have a number of different variables in a parent module that calls multiple child modules. Can the child modules refer to *any* of the variables declared in the parent module?

Correct answer

Not the variable, but it can only refer to values that are passed to the child module

Yes, child modules can refer to any variable in a parent module

No, child modules can never refer to any variables or values declared in the parent module

Overall explanation

Child modules can only access values that are passed in the calling module block.

The resources defined in a module are encapsulated, so the calling module cannot access its attributes directly. However, the child module can declare output values to selectively export certain values to be accessed by the calling module.

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

https://developer.hashicorp.com/terraform/language/modules/syntax#accessing-module-output-values

Domain

Objective 5 - Interact with Terraform Modules

Question 14Skipped

Where is the most secure place to store credentials when using a remote backend?

using an input variable defined in your variables.tf file

in the backend configuration block where the remote state location is defined

environment variables

Correct answer

defined outside of Terraform

Overall explanation

Anytime you can configure these credentials outside of Terraform is your best choice. Environment variables would be the second most-secure choice here. The primary focus is to ensure your credentials are not stored in plaintext and committed to a code repository. ***NOTE:*** *You could use an encrypted file to store credentials and that encrypted file could be accessed by Terraform to read the creds.*

WRONG ANSWERS:

environment variables - this is the SECOND best choice here, with storing outside of Terraform using a credential file being the best choice

in the backend configuration block where the remote state location is defined - this is exactly what we DO NOT want to do because the creds are now stored in cleartext, which isn't desirable. Also, backend config might also get committed to a code repo.

using an input variable defined in your variables.tf file - this is another example of what we DO NOT want - storing the credentials in a cleartext file

https://blog.gruntwork.io/a-comprehensive-guide-to-managing-secrets-in-your-terraform-code-1d586955ace1

Domain

Objective 7 - Implement and Maintain State

Question 15Skipped

True or False? Running a terraform apply will fail if you do not run a terraform plan first.

Correct answer

False

True

Overall explanation

You do NOT need to run a terraform plan before running terraform apply. When you execute a terraform apply, it will actually run its own "plan" to make sure it knows what resources to update.

The most straightforward way to use terraform apply is to run it without any arguments at all, in which case it will automatically create a new execution plan (as if you had run terraform plan) and then prompt you to approve that plan, before taking the indicated actions.

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

Domain

Objective 6 - Use the Core Terraform Workflow

Question 16Skipped

True or False? In Terraform OSS, workspaces generally use the same code repository while workspaces in Terraform Enterprise/Cloud are often mapped to different code repositories.

False

Correct answer

True

Overall explanation

Workspaces in OSS are often used within the same working directory while workspaces in Enterprise/Cloud are often (but not required) mapped to unique repos.

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

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

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 17Skipped

True or False? Input variables that are marked as sensitive are NOT written to Terraform state.

True

Correct answer

False

Overall explanation

While the value is not shown in the Terraform CLI output, the value will still be written to state. This is why it's important to secure your state file wherever possible.

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

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 18Skipped

You have declared the variable as shown below. How should you reference this variable throughout your configuration?

```
variable "aws_region" {
  type        = string
  description = "region used to deploy workloads"
  default     = "us-east-1"
  validation {
    condition     = can(regex("^us-", var.aws_region))
    error_message = "The aws_region value must be a valid region in the USA, starting with \"us-\"."
  }
}
```

variable.aws_region.id

Correct answer

var.aws_region

variable.aws_region

var.aws_region.id

Overall explanation

Input variables (commonly referenced as just 'variables') are often declared in a separate file called `variables.tf`, although this is not required. Most people will consolidate variable declaration in this file for organization and simplification of management. Each variable used in a Terraform configuration must be declared before it can be used. Variables are declared in a variable block - one block for each variable. The variable block contains the variable name, most importantly, and then often includes additional information such as the type, a description, a default value, and other options.

The value of a Terraform variable can be set multiple ways, including setting a default value, interactively passing a value when executing a terraform plan and apply, using an environment variable, or setting the value in a `.tfvars` file. Each of these different options follows a strict order of precedence that Terraform uses to set the value of a variable.

A huge benefit of using Terraform is the ability to reference other resources throughout your configuration for other functions. These might include getting certain values needed to create other resources, creating an output to export a specific value, or using data retrieved from a data block. Most of these use dot-separated paths for elements of object values.

The following represents the kinds of named values available in Terraform:

* <RESOURCE TYPE>.<NAME> represents a managed resource of the given type and name.

* **var.<NAME> is the value of the input variable of the given name.**

* local.<NAME> is the value of the local value of the given name.

* module.<MODULE NAME> is a value representing the results of a module block.

* data.<DATA TYPE>.<NAME> is an object representing a data resource of a given type and name

* Additional named values include ones for filesystem and workspace info and block-local values

https://developer.hashicorp.com/terraform/language/expressions/references#input-variables

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 19Skipped

True or False? Infrastructure as code (IaC) tools allow you to manage infrastructure with configuration files rather than through a graphical user interface.

Correct answer

True

False

Overall explanation

This is true, although there are tools out there that have UIs to deploy IaC. However, the goal is to reduce or eliminate the need to use a UI to deploy infrastructure and applications.

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

Domain

Objective 1 - Understand Infrastructure as Code Concepts

Question 20Skipped

True or False? A provider block is required in every configuration file so Terraform can download the proper plugin.

True

Correct answer

False

Overall explanation

You don't have to specify a provider block since Terraform is smart enough to download the right provider based on the specified resources. That said, Terraform official documentation states that Terraform configurations must declare which providers they require so that Terraform can install and use them. Although Terraform CAN be used without declaring a plugin, you should follow best practices and declare it along with the required_version argument to explicitly set the version constraint.

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

Domain

Objective 3 - Understand Terraform Basics

Question 21Skipped

True or False? Under special circumstances, Terraform can be used without state.

True

Correct answer

False

Overall explanation

State is a hard requirement for Terraform - there's no getting around it. You can have the state stored locally or you can configure a remote backend to store it somewhere else. But overall, state is always required for Terraform.

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

Domain

Objective 2 - Understand Terraform's purpose (vs other IaC)

Question 22Skipped

Your organization has multiple engineers that have permission to manage Terraform as well as administrative access to the public cloud where these resources are provisioned. If an engineer makes a change outside of Terraform, what command can you run to detect drift and update the state file?

terraform get

terraform init

terraform state list

Correct answer

terraform apply -refresh-only

Overall explanation

To instruct Terraform to refresh the state file based on the current configuration of managed resources, you can use the terraform apply -refresh-only command. If Terraform discovers drift, it will update the state file with the changes.

Note that terraform refresh used to be the correct command here, but that command is deprecated. It might show up on the exam though.

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

Domain

Objective 7 - Implement and Maintain State

Question 23Skipped

You want to restrict your team members to specific modules that are approved by the organization's security team when using Terraform Cloud. What feature should you use?

Correct answer

Terraform Cloud Private Registry

Terraform Registry (public)

Terraform Cloud Organizations

Terraform Workspaces

Overall explanation

Private providers and private modules are hosted on an organization's private registry and are only available to members of that organization.

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

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 24Skipped

Rather than having to scan and inspect every resource on every run, Terraform relies on what feature to help manage resources?

Correct answer

state

environment variables

local variables

providers

Overall explanation

Terraform must store state about your managed infrastructure and configuration. This state is used by Terraform to map real-world resources to your configuration, keep track of metadata, and improve performance for large infrastructures.

This state is stored by default in a local file named "terraform.tfstate", but it can also be stored remotely, which works better in a team environment.

WRONG ANSWERS:

environment variables - these can be used to provide values for variables or other Terraform configurations, but this is now how Terraform manages resources

providers - these are Terraform plugins that enable communication to a platform's API to allow Terraform to provision resources, but it's not the feature that it uses to manage the resources that it is managing.

local variables - these provide values to use for the deployment and management of deployed resources in Terraform. It doesn't use these to manage resources

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

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

Domain

Objective 2 - Understand Terraform's purpose (vs other IaC)

Question 25Skipped

Given the following code snippet, what is the managed resource name for this particular resource?

```
resource "aws_vpc" "prod-vpc" {
  cidr_block = var.vpc_cidr

  tags = {
    Name        = var.vpc_name
    Environment = "demo_environment"
    Terraform   = "true"
  }

  enable_dns_hostnames = true
}
```

demo environment

resource.aws_vpc

Correct answer

prod-vpc

aws_vpc

Overall explanation

prod-vpc is the managed resource name for this resource. More specifically, you'd use aws_vpc.prod-vpc address to refer to this resource in your code, like this:

```
output "vpc_id" {
  value = aws_vpc.prod-vpc.id
}
```

https://developer.hashicorp.com/terraform/cli/state/resource-addressing

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 26Skipped

Using multi-cloud and provider-agnostic tools like Terraform provides which of the following benefit?

increased risk due to all infrastructure relying on a single tool for management

Correct answer

can be used across major cloud providers and VM hypervisors

slower provisioning speed allows the operations team to catch mistakes before they are applied

forces developers to learn Terraform alongside their current programming language

Overall explanation

Terraform can be used across major cloud providers and VM hypervisors, which is a huge benefit. This is made possible by the thousands of providers/plugins that are written by HashiCorp, third-party partners, or individual contributors.

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

Domain

Objective 2 - Understand Terraform's purpose (vs other IaC)

Question 27Skipped

Based on the code snippet below, where is the module that the code is referencing?

```
module "server_subnet_1" {
  source        = "./modules/web_server"
  ami          = data.aws_ami.ubuntu.id
  key_name      = aws_key_pair.generated.key_name
  user          = "ubuntu"
  private_key   = tls_private_key.generated.private_key_pem
  subnet_id     = aws_subnet.public_subnets["public_subnet_1"].id
  security_groups = [aws_security_group.vpc-ping.id, aws_security_group.vpc-web.id]
}
```

stored in the Terraform public registry

the same working directory where Terraform is being executed

stored in a private registry managed by the organization

Correct answer

in the modules subdirectory in the current working directory where Terraform is being executed

Overall explanation

In this example, the user created a `modules` directory and then saved the module in that new directory. Therefore, the answer is in the modules subdirectory in the current working directory where Terraform is being executed.

Anytime you have a local path as the source, the module will be sourced from the referenced directory. You could also put the path of a VCS repository here or a reference to a private or public registry.

For example, if you wanted to reference a public module, you could use something like this:

```
module "network" {
  source  = "Azure/network/azurerm"
  version = "3.5.0"
  # insert the 1 required variable here
}
```

https://developer.hashicorp.com/terraform/internals/module-registry-protocol#module-addresses

Domain

Objective 5 - Interact with Terraform Modules

Question 28Skipped

After deploying a new virtual machine using Terraform, you find that the local script didn't run properly. However, Terraform reports the virtual machine was successfully created. How can you force Terraform to replace the virtual machine without impacting the rest of the managed infrastructure?

update the virtual machine resource and run a terraform init

execute a terraform debug command to see why the script failed to run

Correct answer

use terraform apply -replace to tag the resource for replacement

run a terraform destroy and then run terraform import to pull in the other resources under Terraform management

Overall explanation

Using terraform apply -replace is how you tag a resource for replacement.

IMPORTANT - PLEASE READ BELOW

The terraform taint command is deprecated. However, you may still see terraform taint on the real exam. Please note the information below and be prepared to understand both options.

For Terraform v0.15.2 and later, we recommend using the -replace option with terraform apply instead. Check out this link for more details.

https://developer.hashicorp.com/terraform/cli/commands/plan#replace-address

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 29Skipped

You've included two different modules from the official Terraform registry in a new configuration file. When you run a terraform init, where does Terraform OSS download and store the modules locally?

in the same root directory where the Terraform configuration files are stored

Correct answer

in the .terraform/modules folder in the working directory

Terraform stores them in memory on the machine running Terraform

in the /tmp directory of the machine executing Terraform

Overall explanation

When plugins and modules are downloaded, they are stored under their respective directory in the .terraform folder within the current working directory. For example, providers/plugins are downloaded to .terraform/providers and modules are downloaded to the .terraform/modules directory.

WRONG ANSWERS:

Terraform doesn't use any of these directories to store any downloaded content when running a terraform init

https://developer.hashicorp.com/terraform/language/providers#provider-installation

Domain

Objective 5 - Interact with Terraform Modules

Question 30Skipped

A coworker provided you with Terraform configuration file that includes the code snippet below. Where will Terraform download the referenced module from?

```
terraform {
  required_providers {
    kubernetes = {
      source = "hashicorp/kubernetes"
      version = "2.6.1"
    }
  }
}

provider "kubernetes" {
  # Configuration below
...
```

Correct answer

the official Terraform public registry

from the hashicorp/kubernetes directory where Terraform is executed

from the official Kubernetes public GitHub repo

from the configured VCS provider in the hashicorp/kubernetes repo

Overall explanation

When a module is located at hashicorp/<name>, Terraform download it from the official Terraform public registry. This is specified by the source argument within the module block. The module installer supports the following source types:

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

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

Domain

Objective 5 - Interact with Terraform Modules

Question 31Skipped

You are having trouble with executing Terraform and want to enable the *most verbose* logs. What log level should you set for the TF_LOG environment variable?

DEBUG

Correct answer

TRACE

ERROR

INFO

Overall explanation

Trace is the most verbose logging level. In order, they go TRACE, DEBUG, INFO, WARN and ERROR

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

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 32Skipped

You are managing multiple resources using Terraform running in AWS. You want to destroy all the resources except for a single web server. How can you accomplish this?

delete the web server resource block from the configuration file and run a terraform apply

change to a different workspace and run a terraform destroy

Correct answer

run a terraform state rm to remove it from state and then destroy the remaining resources by running terraform destroy

run a terraform import command against the web server and then execute a terraform destroy

Overall explanation

To accomplish this, you can delete the resource from state so Terraform no longer knows anything about it. Then you can run a terraform destroy to destroy the remaining resources. During the destroy, Terraform won't touch the web server since it is no longer managing it. This is similar to how Terraform won't impact existing resources that it does not know about when creating, modifying, and destroying resources in your local or public cloud infrastructure.

To delete a resource from state, you can use the terraform state rm <address> command, which will effectively make Terraform "forget" the object while it continues to exist in the remote system.

WRONG ANSWERS:

change to new workspace and run terraform destroy - changing to a different workspace would have no impact on any resources in the current workspace. It could, however, impact any resources that were provisioned with the second workspace that you change to

terraform import & terraform destroy - using the import command here is basically the opposite of what we want to do. We want Terraform to "forget" about a particular resource, and the import command pulls existing resources under Terraform management

delete the block and run terraform apply - this would actually destroy ONLY the web server that we want to keep, so essentially doing the exact opposite of what we're trying to accomplish

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

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 33Skipped

Infrastructure as Code (IaC) provides many benefits to help organizations deploy application infrastructure much faster than manually clicking in the console. Which is NOT an additional benefit to IaC?

creates self-documenting infrastructure

allows infrastructure to be versioned

code can easily be shared and reused

Correct answer

eliminates API communication to the target platform

Overall explanation

Eliminating API communication to the target platform is NOT a benefit of IaC. In fact, Terraform likely increases communication with the backend platform since Terraform uses the platform's API to build and manage infrastructure.

Remember that Terraform providers/plugins are essentially the features that enable communication with the platform's API on your behalf.

WRONG ANSWERS:

All of the wrong answers are essentially direct benefits of using Terraform.

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

Domain

Objective 1 - Understand Infrastructure as Code Concepts

Question 34Skipped

When working with Terraform CLI/OSS workspaces, what command can you use to display the current workspace you are working in?

terraform workspace

terraform workspace new

Correct answer

terraform workspace show

terraform workspace select

Overall explanation

The terraform workspace show command is used to output the current workspace.

WRONG ANSWERS:

terraform workspace new will create a new workspace

terraform workspace select will tell Terraform what workspace to change to and use terraform workspace is just a container that requires additional subcommands

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

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 35Skipped

The command terraform destroy is actually just an alias to which command?

terraform apply -replace-all

terraform plan - destroy

terraform delete

Correct answer

terraform apply -destroy

Overall explanation

This command is just a convenience alias for the command terraform apply -destroy

For that reason, this command accepts most of the options that terraform apply accepts, although it does not accept a plan file argument and forces the selection of the "destroy" planning mode.

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

Domain

Objective 6 - Use the Core Terraform Workflow

Question 36Skipped

Terraform relies on state in order to create and manage resources. Which of the following is true regarding the state file? (select four)

Correct selection

the state file is formatted using JSON

Correct selection

remote state is required when more than one person wants to manage the infrastructure managed by Terraform

state should be modified by editing the file directly

Correct selection

it may contain sensitive data that you might not want others to view

Correct selection

by default, state is stored in a file named terraform.tfstate in the current working directory

Overall explanation

In order to properly and correctly manage your infrastructure resources, Terraform stores the state of your managed infrastructure. Terraform uses this state on each execution to plan and make changes to your infrastructure. This state must be stored and maintained on each execution so future operations can perform correctly.

During execution, Terraform will examine the state of the currently running infrastructure, determine what differences exist between the current state and the revised desired state, and indicate the necessary changes that must be applied. When approved to proceed, only the necessary changes will be applied, leaving existing, valid infrastructure untouched.

By default, Terraform will store the state in a JSON-formatted file named terraform.tfstate in the current working directory. The file should never be modified directly if state needs to be manually changed. You should use the terraform state command to modify state where possible.

The local state file is NOT encrypted (unless your local disk is encrypted or using something like Windows Bitlocker). This is why many organizations use a remote backend, which will store the state file in an encrypted store. This also allows multiple team members to work with the state file rather than storing it on an individual's laptop/desktop.

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

Domain

Objective 7 - Implement and Maintain State

Question 37Skipped

Which of the following code snippets will ensure you're using a specific version of the AWS provider?

```
terraform {
  required_version = ">= 3.0"
}
```

Correct answer

```
terraform {
  required_providers {
    aws = ">= 3.0"
  }
}
```

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

3. required_provider ">= 3.0"

4. }

1. provider "aws" {

2. region = "us-east-2"

3. required_version ">= 3.0"

4. }

Overall explanation

To specify the version of Terraform provider that is required, you need to use the required_providers block parameter under the terraform block. HashiCorp recommends that you explicitly set the version of both Terraform and the required providers/plugins to avoid issues when upgrading to the latest versions.

WRONG ANSWERS:

None of these wrong answers are valid configuration blocks for Terraform to set the specific version of Terraform.

https://developer.hashicorp.com/terraform/language/settings#specifying-a-required-terraform-version

Domain

Objective 3 - Understand Terraform Basics

Question 38Skipped

What feature does Terraform use to map configuration to resources in the real world?

resource blocks

Correct answer

state

parallelism

local variables

Overall explanation

Terraform requires some sort of database to map Terraform config to the real world. When you have a resource resource "aws_instance" "foo" in your configuration, Terraform uses this map to know that instance i-abcd1234 is represented by that resource.

For some providers like AWS, Terraform could theoretically use something like AWS tags. Early prototypes of Terraform actually had no state files and used this method. However, we quickly ran into problems. The first major issue was a simple one: not all resources support tags, and not all cloud providers support tags.

Therefore, for mapping configuration to resources in the real world, Terraform uses its own state structure.

WRONG ANSWERS:

Parallelism is the way Terraform can deploy many resources at the same time to speed up the deployment

Local variables are used to reduce repeating the same expressions or values over and over in your code

Resource blocks are the block type that deploys actual resources

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

Domain

Objective 2 - Understand Terraform's purpose (vs other IaC)

Question 39Skipped

Which of the following is an advantage of using Infrastructure as Code?

simplification of using a user interface to define your infrastructure

Correct answer

standardize your deployment workflow

elimination of security vulnerabilities in your application deployment workflow

increase the time to market for application deployment

Overall explanation

IaC helps organizations standardize their deployment workflows since they can codify and automate the deployment of applications and underlying infrastructure.

WRONG ANSWERS:

increase the time to market for application deployment - nope, who wants to use a tool that would INCREASE the time to market for application deployment? We want to find tools that DECREASE it.

simplification of using a user interface to define your infrastructure - Nah, we want to move quickly, and using a user interface for any tool likely slows us down and increases our chances of human error. We want to use a CLI or API for our changes so it's quick and predictable.

elimination of security vulnerabilities in your application deployment workflow - while Terraform could help with the security of your workflow, it doesn't guarantee it since Terraform just deploys what you tell it. It's not checking for security requirements or vulnerabilities or anything like that

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

Domain

Objective 1 - Understand Infrastructure as Code Concepts

Question 40Skipped

Which of the following are *collection* or *structural types* that can be used when declaring a variable in order to group values together? (select four)

Correct selection

map

Correct selection

list

Correct selection

tuple

string

bool

Correct selection

object

number

Overall explanation

As you continue to work with Terraform, you're going to need a way to organize and structure data. This data could be input variables that you are giving to Terraform, or it could be the result of resource creation, like having Terraform create a fleet of web servers or other resources. Either way, you'll find that data needs to be organized yet accessible so it is referenceable throughout your configuration. The Terraform language uses the following types for values:

* **string**: a sequence of Unicode characters representing some text, like "hello".

* **number**: a numeric value. The number type can represent both whole numbers like 15 and fractional values like 6.283185.

* **bool**: a boolean value, either true or false. bool values can be used in conditional logic.

* **list (or tuple)**: a sequence of values, like ["us-west-1a", "us-west-1c"]. Elements in a list or tuple are identified by consecutive whole numbers, starting with zero.

   * **map (or object)**: a group of values identified by named labels, like {name = "Mabel", age = 52}. Maps are used to store key/value pairs.

Strings, numbers, and bools are sometimes called primitive types. Lists/tuples and maps/objects are sometimes called complex types, structural types, or collection types.

https://developer.hashicorp.com/terraform/language/expressions/types#types-and-values

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 41Skipped

After hours of development, you've created a new Terraform configuration from scratch and now you want to test it. Before you can provision the resources, what is the first command that you should run?

terraform import

Correct answer

terraform init

terraform apply

terraform validate

Overall explanation

When you develop new Terraform code, and you're ready to test it out, the first thing you need to do is run terraform init in order to initialize the working directory and download any required providers or referenced modules. Even if you're in a directory that has some of these plugins, you should still run terraform init to make sure all the providers have been downloaded. You could even run a terraform init -upgrade to ensure you have the latest versions of the plugins that meet your requirements.

WRONG ANSWERS:

None of these commands would work if you haven't initialized the working directory yet. You would get an error similar to this:

```
Error: Could not load plugin


Plugin reinitialization required. Please run "terraform init".

```

```
│ Plugins are external binaries that Terraform uses to access and manipulate
│ resources. The configuration provided requires plugins which can't be located,
│ don't satisfy the version constraints, or are otherwise incompatible.
│
│ Terraform automatically discovers provider requirements from your
│ configuration, including providers used in child modules. To see the
│ requirements and constraints, run "terraform providers".
│
│ failed to instantiate provider "registry.terraform.io/hashicorp/aws" to obtain schema:
│ unknown provider "registry.terraform.io/hashicorp/aws"
```

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

Domain

Objective 6 - Use the Core Terraform Workflow

Question 42Skipped

You need to input variables that follow a key/value type structure. What type of variable would be used for this use case?

use a string variable to accomplish this task

use a list of strings for this use case

use an array to satisfy the requirement

Correct answer

use a map to satisfy this requirement

Overall explanation

Map is the best choice for this use case because it allows you to create a key/value structure that can easily be referenced in your Terraform configuration.

WRONG ANSWERS:

- use a list of strings for this use case - nope, this is just a list of strings that wouldn't create a key/value structure. '

- use a string variable to accomplish this task - nope, this would allow you to just create a single string value

- use an array to satisfy the requirement - array isn't a valid Type constraint in Terraform https://developer.hashicorp.com/terraform/language/expressions/types#maps-objects

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 43Skipped

You need to define a single input variable to support the IP address of a subnet, which is defined as a string, and the subnet mask, which is defined as a number. What type of variable variable should you use?

type = string

Explanation

Using a string type variable would only allow you to define a single value, such as the IP address, but not both the IP address and subnet mask. It does not support multiple data types within a single variable, so it would not be appropriate for this scenario.

Correct answer

type = object ()

Explanation

Using an object type variable allows you to define a single input variable that can hold multiple values, such as the IP address (string) and subnet mask (number). This type of variable is suitable for grouping related data together and maintaining the structure of the input data.

type = map ()

Explanation

Using a map type variable allows you to define key-value pairs, which could be used to store the IP address and subnet mask. However, an object type variable is more appropriate in this case as it explicitly defines the structure of the input data with different data types for each field.

type = bool

Explanation

Using a bool (boolean) type variable is used for true/false values, which are not suitable for storing both a string (IP address) and a number (subnet mask). It does not allow for the combination of different data types in a single variable.

Overall explanation

A *structural* type allows multiple values of *several distinct types* to be grouped together as a single value. Structural types require a *schema* as an argument, to specify which types are allowed for which elements.

object(...): a collection of named attributes that each have their own type.

The schema for object types is { <KEY> = <TYPE>, <KEY> = <TYPE>, ... } — a pair of curly braces containing a comma-separated series of <KEY> = <TYPE> pairs. Values that match the object type must contain *all* of the specified keys, and the value for each key must match its specified type. (Values with *additional* keys can still match an object type, but the extra attributes are discarded during type conversion.)

https://developer.hashicorp.com/terraform/language/expressions/type-constraints#optional-object-type-attributes

Domain

Objective 3 - Understand Terraform Basics

Question 44Skipped

What actions does a terraform init perform for you?

ensures that all Terraform files match the canonical formatting and style

ensures any configuration file that ends with a .tf file extension is syntactically valid and internally consistent

Correct answer

downloads plugins and retrieves the source code for referenced modules

compares the current configuration to the prior state and notes any differences

Overall explanation

One of the functions that a terraform init does for you is download any referenced modules/plugins so they can be used locally. This is required before you run most other terraform CLI commands.

WRONG ANSWERS:

- ensures any configuration file that ends with a .tf file extension is syntactically valid and internally consistent - this is the job of terraform validate, not terraform init

- ensures that all terraform files match the canonical formatting and style - this is done by the CLI command terraform fmt

- compares the current configuration to the prior state and notes any differences - nope, that's what a terraform plan will do for you, not terraform init

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

Domain

Objective 6 - Use the Core Terraform Workflow

Question 45Skipped

Fill in the blank from the correct answer below:

A Terraform module (usually the root module of a configuration) can *call* other modules to include their resources into the configuration. A module that has been called by another module is often referred to as _______________.

sourced module

parent module

Correct answer

child module

called module

Overall explanation

A Terraform module (usually the root module of a configuration) can *call* other modules to include their resources into the configuration. A module that has been called by another module is often referred to as a *child module.*

Child modules can be called multiple times within the same configuration, and multiple configurations can use the same child module.

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

Domain

Objective 5 - Interact with Terraform Modules

Question 46Skipped

By default, where does Terraform CLI store its state?

in the default workspace in Terraform Cloud

in a temp directory on the local machine executing the Terraform configurations

Correct answer

in the terraform.tfstate file on the local backend

in the .terraform directory within the current working directory

Overall explanation

Terraform must store state about your managed infrastructure and configuration. This state is used by Terraform to map real-world resources to your configuration, keep track of metadata, and improve performance for large infrastructures.

Terraform will store its state in the terraform.tfstate file on the local backend. This is the default but you can always change it if you want.

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

Domain

Objective 7 - Implement and Maintain State

Question 47Skipped

Which of the following are tasks that terraform apply can perform? (select three)

Correct selection

update existing infrastructure with new configurations

Correct selection

provision new infrastructure

import existing infrastructure

Correct selection

destroy infrastructure previously deployed with Terraform

Overall explanation

The terraform apply command executes the actions proposed in a Terraform plan. This works the same regardless of whether you have existing infrastructure deployed and changes are needed, or if you are just deploying your infrastructure for the first time. The terraform apply command can also destroy infrastructure by passing the -destroy flag as well.

WRONG ANSWER:

Running a terraform apply cannot import infrastructure to pull under Terraform management. That's the job of the terraform import command

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

https://learn.hashicorp.com/collections/terraform/aws-get-started

Domain

Objective 6 - Use the Core Terraform Workflow

Question 48Skipped

As you are developing new Terraform code, you are finding that you are constantly repeating the same expression over and over throughout your code, and you are worried about the effort that will be needed if this expression needs to be changed. What feature of Terraform can you use to avoid repetition and make your code easier to maintain?

data block

terraform graph

Correct answer

locals

remote backend

Overall explanation

A local value assigns a name to an expression, so you can use it multiple times within a configuration without repeating it. The expressions in local values are not limited to literal constants; they can also reference other values in the configuration in order to transform or combine them, including variables, resource attributes, or other local values.

You can use local values to simplify your Terraform configuration and avoid repetition. Local values (locals) can also help you write a more readable configuration by using meaningful names rather than hard-coding values. If overused they can also make a configuration hard to read by future maintainers by hiding the actual values used.

Use local values only in moderation, in situations where a single value or result is used in many places and that value is likely to be changed in the future. The ability to easily change the value in a central place is the key advantage of local values.

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

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 49Skipped

What command can be used to display the resources that are being managed by Terraform?

terraform output

terraform state rm

Correct answer

terraform show

terraform version

Overall explanation

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.

Machine-readable output is generated by adding the -json command-line flag.

WRONG ANSWERS:

terraform state rm is used to remove resources from state

terraform output is used to get values from an output of a module or configuration

terraform version is used to display the current version of Terraform

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

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 50Skipped

Given the code snippet below, how would you identify the arn to be used in the output block that was retrieved by the data block?

```
data "aws_s3_bucket" "data-bucket" {
  bucket = "my-data-lookup-bucket-btk"
}
...

output "s3_bucket_arn" {
  value = ????
}
```

aws_s3_bucket.data-bucket

data.aws_s3_bucket.arn

Correct answer

data.aws_s3_bucket.data-bucket.arn

data.aws_s3_bucket.data-bucket

Overall explanation

To refer to a resource, you'd use <block type>.<resource type>.<name> . In this case, the <block type> is **data**, the <resource type> is **aws_s3_bucket**, and then you'd add the **arn** attribute at the end.

https://developer.hashicorp.com/terraform/cli/state/resource-addressing

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 51Skipped

What of the following are benefits of using Infrastructure as Code? (select three)

Correct selection

the ability to programmatically deploy infrastructure

Correct selection

your infrastructure configurations can be version controlled and stored in a code repository alongside the application code

Correct selection

the reduction of misconfigurations that could lead to security vulnerabilities and unplanned downtime

reducing vulnerabilities in your publicly-facing applications

Overall explanation

Reducing vulnerabilities in your publicly-facing applications is NOT a benefit of using IaC since IaC is geared towards deploying infrastructure and applications, but not determining whether your application is secure.

WRONG ANSWER:

Terraform does not reduce vulnerabilities in your applications. You CAN pair Terraform with other tools that do this through a CI?CD pipeline or something like that, but Terraform will not do this natively.

https://www.youtube.com/watch?v=l5k1ai_GBDE

Domain

Objective 1 - Understand Infrastructure as Code Concepts

Question 52Skipped

As part of a Terraform configuration, you are deploying a Linux-based server using a default image that needs to be customized based on input variables. What feature of Terraform can execute a script on the server once is has been provisioned?

provider

Correct answer

remote-exec provisioner

local-exec provisioner

data resource

Overall explanation

We can utilize Terraform provisioners to deploy a web app onto an instance we've created. In order to run these steps, Terraform needs a connection block along with our generated SSH key from the previous labs in order to authenticate into our instance. Terraform can utilize both the `local-exec` provisioner to run commands on our local workstation (that is executing Terraform) and the `remote-exec` provisioner to execute commands against a resource that has been provisioned with Terraform.

Note: Provisioners should only be used as a last resort. For most common situations there are better alternatives.

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

Domain

Objective 3 - Understand Terraform Basics

Question 53Skipped

Given the definition below, what Terraform feature is being described?

"helps you share Terraform providers and Terraform modules across your organization. It includes support for versioning, a searchable list of available providers and modules, and a configuration designer to help you build new workspaces faster."

CDK for Terraform

Correct answer

Private Registry

HashiCorp Sentinel

Terraform Workspaces

Overall explanation

This definition is describing the Private Registry...

Terraform Cloud's private registry works similarly to the public Terraform Registry and helps you share Terraform providers and Terraform modules across your organization. It includes support for versioning, a searchable list of available providers and modules, and a configuration designer to help you build new workspaces faster.

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

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 54Skipped

You need to set the value for a Terraform input variable. Which of the following allows you to set the value using an environment variable?

export TF_VARIABLE_app=eCommerce01

export VAR_database=prodsql01

export TF_db-pass=P@ssw0rd01!

Correct answer

export TF_VAR_user=dbadmin01

Overall explanation

As a fallback for the other ways of defining variables, Terraform searches the environment of its own process for environment variables named TF_VAR_ followed by the name of a declared variable.

WRONG ANSWERS:

None of the other environment variables shown as answers will successfully set a value for a Terraform variable.

https://developer.hashicorp.com/terraform/language/values/variables#environment-variables

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 55Skipped

True or False? When referencing a module, you must specify the version of the module in the calling module block.

True

Correct answer

False

Overall explanation

While it's not required, we recommend explicitly constraining the acceptable version numbers to avoid unexpected or unwanted changes. Use the version argument in the module block to specify versions.

```
module "consul" {
  source  = "hashicorp/consul/aws"
  version = "0.11.0"

  servers = 3
}
```

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

Domain

Objective 5 - Interact with Terraform Modules

Question 56Skipped

True or False? In order to use the terraform console command, the CLI must be able to lock state to prevent changes.

Correct answer

True

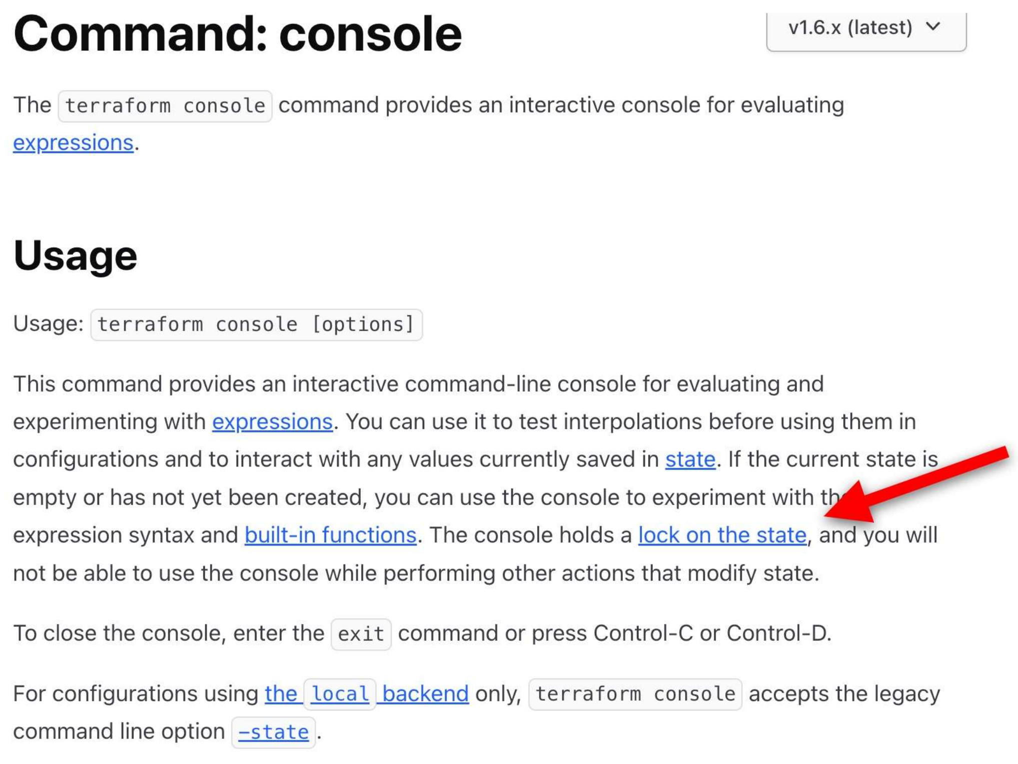
False

Overall explanation

The terraform console command will read the Terraform configuration in the current working directory and the Terraform state file from the configured backend so that interpolations can be tested against both the values in the configuration and the state file.

When you execute a terraform console command, you'll get this output:

1. $ terraform console
2. Acquiring state lock. This may take a few moments...
3. >



Command: console

v1.6.x (latest)

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

Usage

Usage: `terraform console [options]`

This command provides an interactive command-line console for evaluating and experimenting with expressions. You can use it to test interpolations before using them in configurations and to interact with any values currently saved in state. If the current state is empty or has not yet been created, you can use the console to experiment with the expression syntax and built-in functions. The console holds a lock on the state, and you will not be able to use the console while performing other actions that modify state.

To close the console, enter the `exit` command or press Control-C or Control-D.

For configurations using the `local` backend only, `terraform console` accepts the legacy command line option `-state`.

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

Domain

Objective 7 - Implement and Maintain State

Question 57Skipped

You are using Terraform to manage some of your AWS infrastructure. You notice that a new version of the provider now includes additional functionality you want to take advantage of. What command do you need to run to upgrade the provider?

terraform providers

terraform get hashicorp/aws

terraform plan

Correct answer

terraform init -upgrade

Overall explanation

To upgrade an existing provider that you have already downloaded, you need to run terraform init -upgrade . This command will upgrade all previously-selected plugins 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

Domain

Objective 3 - Understand Terraform Basics