Question 1Skipped

Which of the following is ***true*** about working with modules?

modules must be published to the Terraform registry before they can be used

every module that is called from a parent module must output values

Correct answer

a single module can be called many times in a single configuration file

a module can only contain a single resource to be deployed or managed

Overall explanation

Ok, so there's a lot to unpack for this question here. First, let's talk about the correct answer. Modules can be called one or more times by a parent module. The configuration file/module that calls a module is often called the parent, root,or calling module. The module that is called is the child module, or sometimes just "module". The whole point of using a module is to be able to call it one or many times to create resources without having to rewrite the same code over and over.

WRONG ANSWERS:

Where do modules live? While modules can be published to the Terraform registry, they don't have to be. They can be simply stored locally on your machine or in a private code repository. Publishing them to the public registry, or using a private registry, is completely optional.

Module outputs: While modules are often more valuable when they output values, they don't necessarily have to output values. They can be used to simply manage resources. If you do need values from the module, that's when you'd create outputs. Those outputs can be used just for informational purposes or they can be used as inputs for other modules. For example, you might create a subnet in a public cloud in one module and need to output the subnet ID so you can use it as an input on a second module to deploy application workloads.

Resources in Modules: Modules can be used to deploy and manage one or more resources within the module. For example, you might need to deploy multiple resources needs for a specific application or requirements.

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

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

Domain

Objective 5 - Interact with Terraform Modules

Question 2Skipped

If supported by your backend, Terraform will lock your state for all operations that could write state. What purpose does this serve?

Correct answer

This prevents others from acquiring the lock and potentially corrupting your state.

Ensures the state file cannot be moved after the initial terraform apply

Prevents others from committing Terraform code that could override your updates.

Locks colleagues from making manual changes to the managed infrastructure

Overall explanation

State locking prevents others from acquiring the lock and potentially corrupting your state. If Terraform didn't use state locking, multiple people could try to make changes to your infrastructure and corrupt the state file. At that point, Terraform would no longer understand how to manage the resources deployed since the state file wouldn't be consistent.

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.

WRONG ANSWERS:

Committing code? State locking wouldn't prevent somebody from committing code to your code repository.

Moving the state file!..Even after you run your first terraform apply, you can move the state file by modifying your state block and running a terraform init. State locking does not prevent you from moving state in the future.

Changing Infrastructure Even with state locking, somebody could make manual changes to your infrastructure using the API, CLI, or console if they wanted to. State locking doesn't prevent this.

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

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

Domain

Objective 7 - Implement and Maintain State

Question 3Skipped

Your organization has standardized on Microsoft Azure to run its applications on PaaS, SaaS, and IaaS offerings. The deployment quickly standardized on Azure ARM to provision these resources quickly and efficiently.

Which of the following is true about how the team currently deploys its infrastructure?

Correct answer

the adoption of another public cloud provider will prove to be more challenging since all of its codebase is based on ARM

the team would not be able to quickly adapt and integrate baseline security measures in its code to help standardize application deployments

the team would not be able to develop reusable code in order to reduce the time it takes to develop code for new applications

the team would not be able to use its existing skill set to develop code for newly announced services

Overall explanation

While each of the main public cloud providers has its own version of Infrastructure as Code (ARM, AWS CloudFormation, etc.), adopting the native solution can be limiting as the organization matures its cloud capabilities and offerings. If the organization only learns the native solution, what happens if they decide to use a different public cloud provider or they acquire a company that uses a different one? While the developer mindset would still apply, the skillset used to deploy ARM doesn't necessarily apply one-to-one for writing AWS CloudFormation, for example. Each of these solutions handles development and deployment differently, and the engineers have to learn a second solution.

By using Terraform, engineers and developers can focus their time on ***learning a single solution*** applicable to all the public cloud providers and other SaaS, PaaS, and IaaS offerings available on the market.

Wrong Answers:

The wrong answers are all benefits of using any infrastructure as code, such as standardization, reusability, and familiarity with using a cloud provider solution.

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

https://developer.hashicorp.com/terraform/intro/vs/cloudformation

Domain

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

Question 4Skipped

Your organization uses IaC to provision and manage resources in a public cloud platform. A new employee has developed changes to existing code and wants to push it into production.

What best practice should the new employee follow to submit the new code?

Make the change directly using the cloud provider's API to ensure the changes are valid. Store the code locally and email a copy of the code to a teammate so they have an extra copy.

Correct answer

Submit a merge/pull request of the proposed changes. Have a team member validate the changes and approve the request.

Submit the change to the change control board and wait for the approval. Commit the code directly to the main repository.

Execute the code locally on the developer's machine to make the changes directly to the infrastructure.

Overall explanation

Following best practices for code, the new changes should be submitted as a pull/merge request in the existing code repository. A teammate, or the security team, should validate the changes and approve the request, ultimately merging the new changes into the existing codebase

Wrong Answers:

None of these follow best practices for managing code. Please don't do any of these things :)

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

https://developer.hashicorp.com/terraform/intro/vs.

Domain

Objective 1 - Understand Infrastructure as Code Concepts

Question 5Skipped

Which of the following are advantages of using infrastructure as code (IaC) for your day-to-day operations? (select three)

Correct selection

provides the ability to version control the infrastructure and application architecture

Correct selection

enables self-service for developers and operators alike

ensures the security of applications provisioned on managed infrastructure

Correct selection

API-driven workflows

Overall explanation

Using Infrastructure as Code (IaC) like Terraform, CloudFormation, etc. enables organizations to completely change the way they deploy applications and the underlying infrastructure to support them. Rather than click around in a console, ***IaC enables API-driven workflows*** for deploying resources in public clouds, private infrastructure, and other SaaS and PaaS services.

When developing IaC, organizations can now use a Version Control System, such as GitHub, GitLab, Bitbucket, etc. ***to store and version its code***. When changes are needed, the existing code can be cloned, modified, and merged back into the main repository by way of a pull or merge request. These requests can follow a traditional workflow where approvals are needed before they are deployed to a production environment.

By moving your configurations to code and publishing them to a shared repository, or registry, different operators in the organization can now easily consume this code without even knowing how to write Terraform. They can simply provide the required values by way of variables and entire environments can be provisioned to support their application(s).

WRONG ANSWER:

While Terraform can indeed help with the security of your applications, it won't guarantee it. You can combine Terraform with other tools, such as LaunchDarkly, SonarQube, DeepScan, and more using a CI/CD pipeline, but again, it doesn't guarantee the security of your application. Security is the responsibility of everybody involved in the deployment of an application, starting with the developer(s) themselves all the way to the people responsible for the day-to-day operations of the application.

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

https://developer.hashicorp.com/terraform/intro

https://developer.hashicorp.com/terraform/intro/use-cases

Domain

Objective 1 - Understand Infrastructure as Code Concepts

Question 6Skipped

You need Terraform to destroy and recreate a single database server that was deployed with a bunch of other resources. You don't want to modify the Terraform code. What command can be used to accomplish this task?

terraform state rm aws_instance.database

terraform plan -destroy="aws_instance.database"

Correct answer

terraform apply -replace="aws_instance.database"

terraform state show aws_instance.database

Overall explanation

When working with resources, there may be times where a particular resource didn't deploy correctly, although Terraform thinks it did. An example of this might be a script that runs on a virtual machine in the background. The virtual came up fine, so Terraform believes it was successful, but the script didn't perform the tasks you needed it to, so you need Terraform to destroy and recreate the one resource. In this case, you can use terraform apply -replace="<resource_id>" to have Terraform replace this one resource on the next terraform apply.

IMPORTANT - PLEASE READ

This command was formally terraform taint, and you may or may not see terraform taint still on the exam. The taint command was deprecated in Terraform 0.15.2 and replaced with the terraform apply -replace command. Note that the resource is NOT immediately replaced when you run a terraform taint. It will only happen on the next terraform plan/apply. While HashiCorp does a great job updating their exams, sometimes commands can be a little slow to be removed or replaced from the test questions.

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

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 7Skipped

True or False? Official Terraform providers and modules are owned and maintained by HashiCorp.

False

Correct answer

True

Overall explanation

This is true. If a module or provider is marked as official, it is owned and maintained by HashiCorp themselves. *In fact, I copied the sentence in the question straight off the official Terraform registry page :)*

There are other modules/providers available in the registry that are maintained by third-party partners, or even individuals. This also means that not all of the modules published to the Terraform registry are validated or verified by HashiCorp. Many folks will use the public registry as a starting place to create their own custom modules needed to meet requirements.

https://registry.terraform.io/

https://registry.terraform.io/browse/modules

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

Domain

Objective 5 - Interact with Terraform Modules

Question 8Skipped

True or False? You can continue using your local Terraform CLI to execute terraform plan and terraform apply operations while using Terraform Cloud as the backend.

False

Correct answer

True

Overall explanation

If you have migrated or configured your state to use Terraform Cloud using the backend configuration, you can continue using your local Terraform CLI to execute operations while using Terraform Cloud. You can even specify the workspace you want to execute the operation in.

To configure the backend to use Terraform Cloud, you can add something like this:

1. terraform {

```
2.  cloud {
3.   organization = "bryan"
4.
5.   workspaces {
6.    tags = ["app"]
7.   }
8.  }
9. }
```

https://developer.hashicorp.com/terraform/language/settings/terraform-cloud

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 9Skipped

Steve is a developer who is deploying resources to AWS using Terraform. Steve needs to gather detailed information about an EC2 instance that he deployed earlier in the day. What command can Steve use to view this detailed information?

terraform state rm aws_instance.frontend

Correct answer

terraform state show aws_instance.frontend

terraform state list

terraform state pull

Overall explanation

All resources that are managed by Terraform are referenced in the state file, including detailed information about the resource. Terraform uses the state to map your configuration to the real-world resources that are deployed and managed on the backend platform (AWS, GCP, F5, Infoblox, etc.). You can use the terraform state commands to view and manipulate Terraform state if needed.

terraform state show <resource address> will show you a lot of details on the resource, including things like the ID, IP address, the state of the resource, and lots more.

WRONG ANSWERS:

terraform state list will just show you a list of the resources being managed by Terraform, but it won't show you details on each of those resources

terraform state rm aws_instance.frontend would remove the resource from state. This ***would not destroy the resource*** on the public cloud, but it would tell Terraform to stop managing it.

terraform state pull will download the state from its current location, upgrade the local copy to the latest state file version that is compatible with locally-installed Terraform, and output the raw format to stdout

https://developer.hashicorp.com/terraform/cli/commands/state/show#example-show-a-resource

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

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 10Skipped

You need to use multiple resources from different providers in Terraform to accomplish a task. Which of the following can be used to configure the settings for each of the providers?

Correct answer

```
provider "consul" {
  address = "https://consul.krausen.com:8500"
  namespace = "developer"
  token = "45a3bd52-07c7-47a4-52fd-0745e0cfe967"
}

provider "vault" {
  address = "https://vault.krausen.com:8200"
  namespace = "developer"
}
data "consul" {
  address = "https://consul.krausen.com:8500"
  namespace = "developer"
  token = "45a3bd52-07c7-47a4-52fd-0745e0cfe967"
}

data "vault" {
  address = "https://vault.krausen.com:8200"
  namespace = "developer"
```

```
}
terraform {
  providers {
    consul {
      address = "https://consul.krausen.com:8500"
      namespace = "developer"
      token = "45a3bd52-07c7-47a4-52fd-0745e0cfe967"
    }
    vault {
      address = "https://vault.krausen.com:8200"
      namespace = "developer"
    }
  }
}
required_providers {
  consul {
    address = "https://consul.krausen.com:8500"
    namespace = "developer"
    token = "45a3bd52-07c7-47a4-52fd-0745e0cfe967"
  }
  vault {
    address = "https://vault.krausen.com:8200"
    namespace = "developer"
  }
}
```

Overall explanation

To configure each provider, you need to define a provider block and provide the configuration within that block. You would need to do this for each provider that you need to configure. For example, if you needed to customize the aws, gcp, and vault provider, you'd need to create three separate provider blocks, one for each provider.

Additional Clarity: While you can configure parameters inside a provider block, the provider block is not needed to use Terraform successfully. The most common configurations within a provider block are **credentials** to access the platform, which should be placed in *environment*

variables rather than inside a provider block. In my examples above, I am providing custom configurations for my needs. But, if I were using the defaults, I wouldn't need to add a provider block for my project to be successfully deployed.

Don't forget that configurations for a provider go inside of a provider block, but any provider constraints go inside of the terraform --> required_providers block.

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

https://developer.hashicorp.com/terraform/language/providers/configuration#provider-configuration-1

Domain

Objective 3 - Understand Terraform Basics

Question 11Skipped

You are working on updating your infrastructure managed by Terraform. Before lunch, you update your configuration file and run a terraform plan to validate the changes. While you are away, a colleague manually updates a tag on a managed resource directly in the console (UI).

What will happen when you run a terraform apply?

Terraform will destroy the manually-changed resource and recreate it to ensure the infrastructure matches the desired state.

Correct answer

Before applying the new configuration, Terraform will refresh the state and recognize the manual change. It will update the resource based on the desired state as configured in the Terraform configuration. The manual change will no longer exist.

Terraform will recognize the manual change and return an error since the Terraform state no longer matches the real-world infrastructure.

Terraform will update the manually changed resource back to the original configuration. It will then apply the new changes defined in the updated configuration file.

Overall explanation

There's a lot to this question, but the reasoning is pretty basic. Since a resource was manually changed, it means that Terraform state is no longer accurate. However, before a terraform plan or terraform apply is executed, Terraform refreshes its state to ensure it knows the status of all its managed resources. During this process, Terraform would recognize the change, update state, and compare that to the new configuration file. Assuming the change defined in the configuration is identical to the manual change, Terraform would simply apply any changes (if any), update the state file, and complete the terraform apply.

WRONG ANSWERS:

Change the resource back? Terraform relies on state for everything and any changes are a result of comparing the current state of the resource to the desired configuration (the .tf files). Therefore, Terraform won't revert the resource back to the original configuration because the configuration has been updated for the new change, and that's the desired state.

Return an Error? Since Terraform performs a state refresh before executing a plan or apply, Terraform will recognize any configuration changes and then apply any changes. This will not result in an error being returned.

Will Terraform destroy my resources? It won't destroy the resource since the resource is still defined in the configuration file. The only way that TF would destroy your resource is if you actually remove that resource from your configuration file(s).

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

https://learn.hashicorp.com/tutorials/terraform/resource-drift?in=terraform/state

https://learn.hashicorp.com/tutorials/terraform/resource-lifecycle?in=terraform/state

Domain

Objective 6 - Use the Core Terraform Workflow

Question 12Skipped

You work for a retail organization that has multiple peak seasons throughout the year. During those peak seasons, your applications need to be scaled up quickly to handle the increased demand. However, the deployment of application servers is manual and new servers are only deployed when problems are reported by users.

How can you reduce the effort required to deploy new resources, increase the speed of deployments, and reduce or eliminate the negative experiences of your customers?

Deploy new IaC code that automatically shuts down existing application servers and scales the resources down during periods of high demand.

Develop a manual runbook that the developers and operations teams can follow during the peak seasons to reconfigure the compute resources used to serve the primary application.

Rather than wait on user reports, implement a ticketing system that alerts the operations team of poor performance or customer errors. Automatically assign the tickets to the on-call team to quickly resolve.

Correct answer

Develop code that provisions new application servers programmatically. Use monitoring software to trigger a pipeline that deploys additional servers during periods of increased demand.

Overall explanation

In this case, automation is key. And considering that this is a course/question focused on Infrastructure as Code, developing IaC to trigger automatically based on workloads is the best answer here.

Wrong Answers:

While the others sound like a great idea or an improvement in the troubleshooting process, they wouldn't resolve the errors with their customers.

https://developer.hashicorp.com/terraform/intro

https://developer.hashicorp.com/terraform/intro/use-cases

Domain

Objective 1 - Understand Infrastructure as Code Concepts

Question 13Skipped

Which of the following are true about Terraform providers? (select four)

Correct selection

some providers are community-supported

Correct selection

some providers are maintained by HashiCorp

Correct selection

providers can be written and maintained by an outside organization, such as AWS, F5, or Microsoft

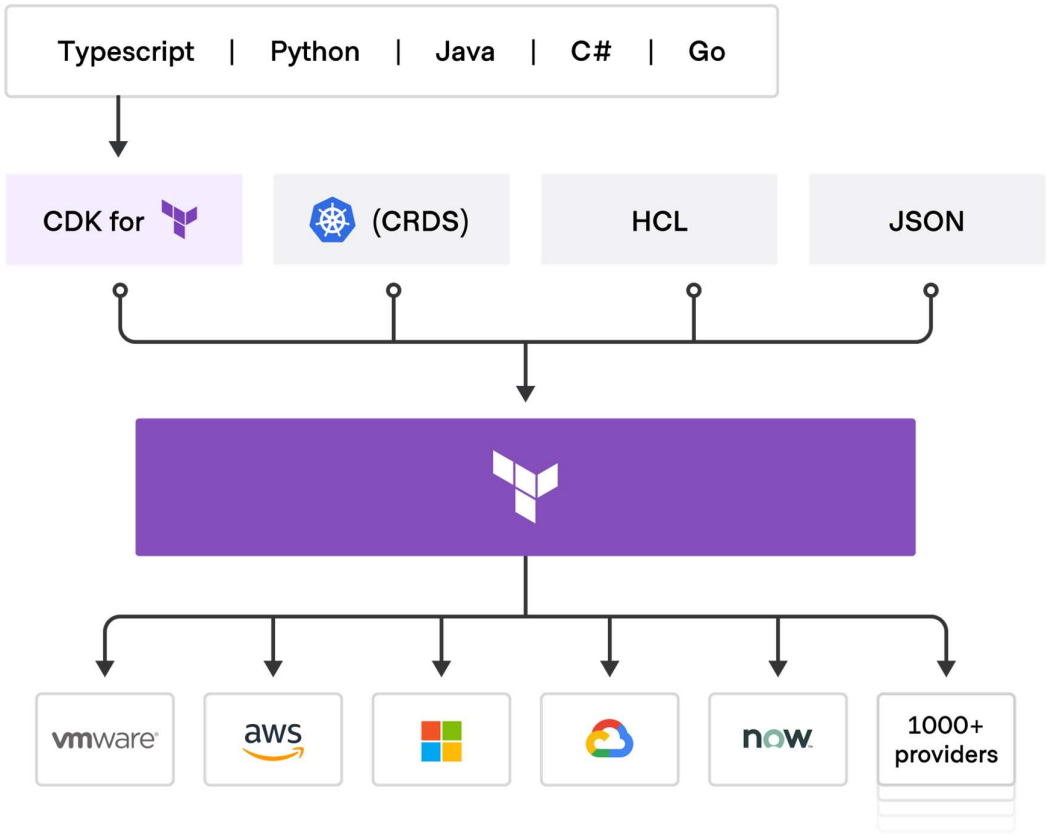
all providers are automatically included when downloading Terraform

Correct selection

they allow anybody to write a provider and publish it to the registry

Overall explanation

The cool part about providers is that anybody can write or contribute to them. This includes individuals would just want to contribute to open-source projects, manufacturers/platform vendors that want to ensure providers are up to date, or HashiCorp themselves. If you find that a provider doesn't provide the capabilities you need, you can develop the new capabilities and submit a PR for review. If approved, those changes would now become part of the Terraform provider that millions of people could use. Pretty cool!



Wrong Answer:

Providers are treated as plugins for Terraform, and during a terraform init process, the required providers are downloaded to the local machine that is executing Terraform so they can be used. Therefore, not all providers are included with Terraform when you download the latest version from terraform.io.

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

https://developer.hashicorp.com/terraform/plugin/how-terraform-works

Domain

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

Question 14Skipped

You have a module named prod_subnet that outputs the subnet_id of the subnet created by the module. How would you reference the subnet ID when using it for an input of another module?

subnet = prod_subnet.subnet_id

subnet = prod_subnet.outputs.subnet_id

subnet = module.outputs.prod_subnet.subnet_id

Correct answer

subnet = module.prod_subnet.subnet_id

Overall explanation

Using interpolation, you can reference the output of an exported value by using the following syntax: module.<module name>.<output name>

Don't forget that before you can reference data/values from a module, the module has to have an output declared that references the desired value(s).

WRONG ANSWERS:

None of the wrong answers are valid interpolation syntax to reference an output that originates from a module.

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

https://learn.hashicorp.com/collections/terraform/modules

Domain

Objective 5 - Interact with Terraform Modules

Question 15Skipped

True or False? When using Terraform Cloud, committing code to your version control system (VCS) can automatically trigger a speculative plan.

Correct answer

True

False

Overall explanation

When workspaces are linked to a VCS repository, Terraform Cloud can automatically initiate Terraform runs when changes are committed to the specified branch.

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

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 16Skipped

True or False? Running a terraform fmt will modify Terraform configuration files in the current working directory and all subdirectories.

True

Explanation

While it is true that running `terraform fmt` will modify Terraform configuration files in the current working directory to adhere to the standard formatting rules, it will not automatically affect files in all subdirectories. The command specifically targets the files in the directory where it is executed, ensuring consistent formatting within that specific location.

Use the -recursive flag to also process files in subdirectories. By default, only the given or current directory is processed.

Correct answer

False

Explanation

Running a `terraform fmt` command will only format the Terraform configuration files in the current working directory, not in all subdirectories. This command is used to standardize the formatting of Terraform code for consistency and readability, but it does not automatically apply to all directories within the project.

Use the -recursive flag to also process files in subdirectories. By default, only the given or current directory is processed.

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.

Use the -recursive flag to also process files in subdirectories. By default, only the given or current directory is processed.

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

Domain

Objective 3 - Understand Terraform Basics

Question 17Skipped

What Terraform command can be used to evaluate and experiment with expressions in your configuration?

terraform get

terraform plan

Correct answer

terraform console

terraform env

Overall explanation

The terraform console command provides an interactive command-line console for evaluating and experimenting with expressions. This is useful for testing interpolations before using them in configurations, and for interacting with any values currently saved in state.

Example from the Terraform documentation:

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

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

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 18Skipped

What are some of the benefits that Terraform *providers* offer to users? (select three)

Correct selection

abstracts the target platform's API from the end-user

enforces security compliance across multiple cloud providers

Correct selection

enables a plugin architecture that allows Terraform to be extensible without having to update Terraform core

Correct selection

enables the deployment of resources to multiple platforms, such as public cloud, private cloud, or other SaaS, PaaS, or IaaS services

Overall explanation

Terraform enables its users to interact with a platform's API without requiring the end-user to understand individual APIs for the targeted platform. This allows a user to easily provision and manage resources across many different platforms without having to understand the API for each individual backend. This benefit makes users more efficient and reduces the administrative burden for understanding and troubleshooting each one.

Terraform can support any platform that has an API, including public, private, and other offerings on the market today. If it has an API, a provider can be written to allow Terraform to manage it. Don't believe me? Check out the Spotify or Domino's Pizza Terraform provider :)

Lastly, by using providers, HashiCorp can enable the extensibility of Terraform without having to modify Terraform core for each supported platform. Each provider, or plugin, can be downloaded as needed to extend the functionality of Terraform itself.

Wrong Answer:

Now, while Terraform can help you standardize security configurations and settings across multiple clouds, it won't enforce it outside of your terraform apply. In other words, it doesn't act like a configuration management tool that can constantly watch for changes to change the configuration back to the desired state.

https://developer.hashicorp.com/terraform/intro/use-cases

https://developer.hashicorp.com/terraform/intro

Domain

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

Question 19Skipped

You have deployed your network architecture in AWS using Terraform. A colleague recently logged in to the AWS console and made a change manually and now you need to be sure your Terraform state reflects the new change.

What command should you run to update your Terraform state?

terraform get -update

terraform plan -out=refresh

Correct answer

terraform apply -refresh-only

terraform init -upgrade

Overall explanation

Terraform includes the ability to use the command terraform apply -refresh-only to refresh the local state based on the changes made outside of the Terraform workflow. Terraform will use the platform's API to query information about each known/managed resource and update any changes it finds.

IMPORTANT - READ THIS!!!!

terraform apply -refresh-only replaced the deprecated command terraform refresh. However, you still may find terraform refresh in the real exam until it gets updated. Keep this in mind when taking the real exam. HashiCorp does update the exams very often, and this could very well come out at the beginning of 2022 when they overhaul the existing exam as noted on the exam page that a new version would be released in early 2022.

WRONG ANSWERS:

terraform plan -out=refresh just runs a terraform plan and saves a plan called refresh. I was being a little tricky here but just know that this isn't how to refresh your state file

terraform init -upgrade is the command to use if you want Terraform to upgrade your existing downloaded providers

terraform get -update is used to download and update modules that are referenced in your Terraform configuration files

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

Oddly enough, the -refresh-only option doesn't currently exist as a valid option on the terraform apply documentation - *if it does show up, let me know so I can update this description*

Domain

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

Question 20Skipped

True or False? When developing Terraform code, you *must* include a provider block for each unique provider so Terraform knows which ones you want to download and use.

True

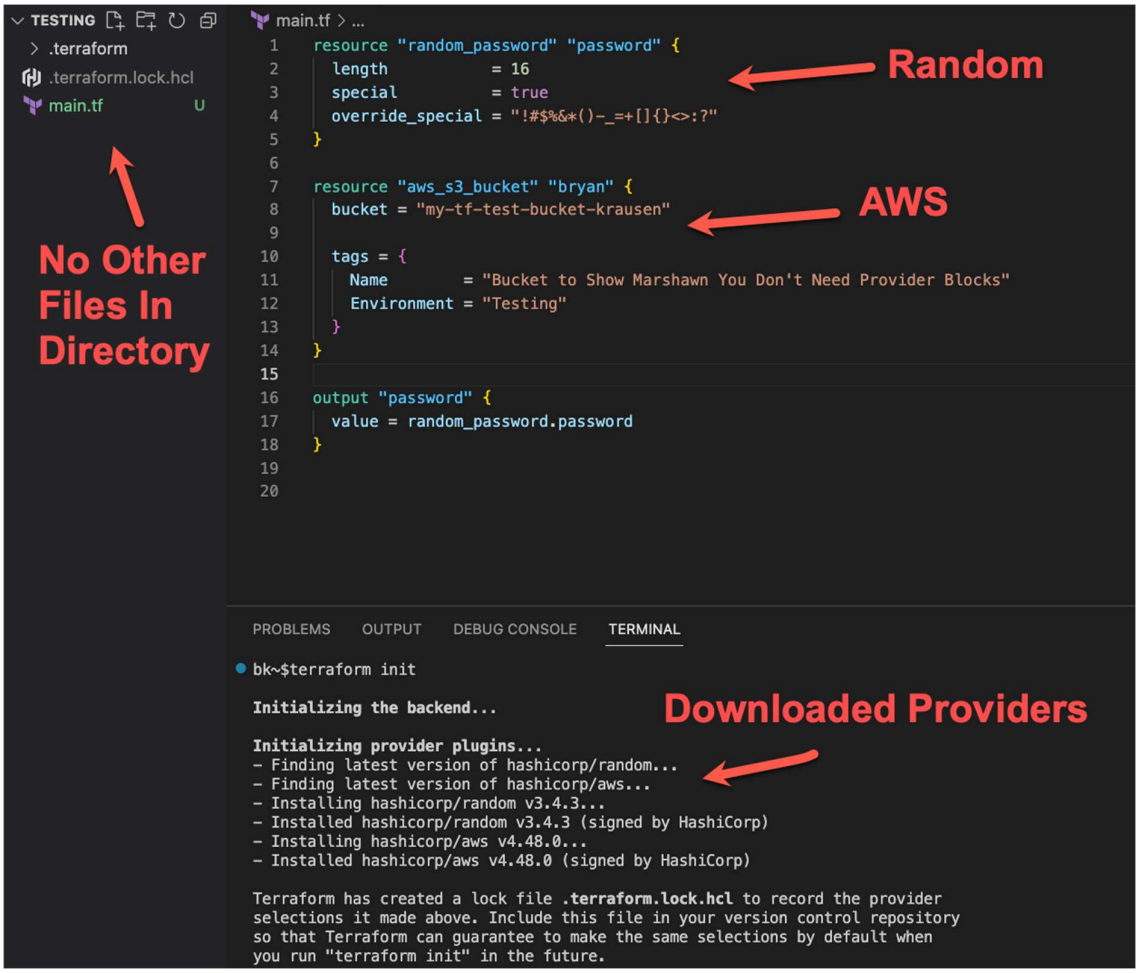
Correct answer

False

Overall explanation

Unlike many other objects in the Terraform language, a provider block may be omitted if its contents would otherwise be empty. Terraform assumes an empty default configuration for any provider that is not explicitly configured. In other words, if you don't have any specific configurations for your provider, you may indeed leave it out of your configuration.

To prove this out, I created a .tf file that includes a resource from the **RANDOM** provider as well as the **AWS** provider and omitted any provider blocks in my configuration. After running a terraform init, you can clearly see that Terraform understands what providers the resources are from and downloads the correct provider plugins. Thus proving that you do NOT need a Provider block to use Terraform.



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

Domain

Objective 3 - Understand Terraform Basics

Question 21Skipped

Which common action does not cause Terraform to refresh its state?

terraform apply

terraform destroy

terraform plan

Correct answer

terraform state list

Overall explanation

Running a terraform state list does not cause Terraform to refresh its state. This command simply reads the state file but it will not modify it.

terraform plan will refresh current state of any already-existing remote objects to make sure that the Terraform state is up-to-date --> see information here

WRONG ANSWERS:

When running a plan, apply, or destroy, Terraform needs to refresh state to ensure that it has the latest information about the managed resources so it understands what changes should be made when applying the desired state configuration.

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

Domain

Objective 7 - Implement and Maintain State

Question 22Skipped

You have recently cloned a repo containing Terraform that you want to test in your environment. Once you customize the configuration, you run a terraform apply but it immediately fails. Why would the apply fail?

Terraform needs to obtain authentication credentials using the terraform login command

Correct answer

Terraform needs to initialize the directory and download the required plugins

you need to run a terraform plan before you can apply the configuration

you can't run Terraform code that was cloned from another users code repository

Overall explanation

When you're learning the basics of Terraform, one of the critical requirements of executing any Terraform code is running terraform init to download all of the required plugins needed for the resources that will be deployed/managed. If you don't run a terraform init when running code in a new directory, a terraform apply/plan will immediately fail since it needs to download the plugins required to run.

WRONG ANSWERS:

While the traditional Terraform workflow is init --> plan --> apply, running a terraform plan is not required to execute a terraform plan. It's recommended to make changes to real environments, but when I'm testing or building labs for my other courses, I rarely run a terraform plan and go straight to terraform apply.

As for running Terraform cloned from another repo, you can absolutely do this. Many people use existing code as a starting point for their own environment. They will clone the repo, customize it however they need and then run it. This is a perfectly acceptable practice and it prevents you from constantly reinventing the wheel.

Regarding authentication, you only need to use the terraform login command when you are working with Terraform Cloud. You don't need authentication credentials outside of that use case unless you are deploying resources to some platform, like AWS, Azure, VMware, etc. There

are plenty of use cases for Terraform where you don't need authentication credentials at all, like using the TLS or random provider.

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

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

https://developer.hashicorp.com/terraform/cli/init#initialization

Domain

Objective 3 - Understand Terraform Basics

Question 23Skipped

You are worried about unauthorized access to the Terraform state file since it might contain sensitive information. What are some ways you can protect the state file? (select two)

replicate the state file to an encrypted storage device

Correct selection

store in a remote backend that encrypts state at rest

Correct selection

use the S3 backend using the encrypt option to ensure state is encrypted

enable native encryption in Terraform as configured in the terraform block

Overall explanation

If you manage any sensitive data with Terraform (like database passwords, user passwords, or private keys), treat the state itself as sensitive data.

Storing state remotely can provide better security. As of Terraform 0.9, Terraform does not persist state to the local disk when remote state is in use, and some backends can be configured to encrypt the state data at rest.

Terraform Cloud always encrypts state at rest and protects it with TLS in transit. Terraform Cloud also knows the identity of the user requesting state and maintains a history of state changes. This can be used to control access and track activity. Terraform Enterprise also supports detailed audit logging.

The S3 backend supports encryption at rest when the encrypt option is enabled. IAM policies and logging can be used to identify any invalid access. Requests for the state go over a TLS connection.

WRONG ANSWERS:

Replication? replicating the state file to another location won't prevent the original file from being accessed.

Encryption? As of today, Terraform doesn't support any type of native encryption capability when writing and managing state.

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

Domain

Objective 7 - Implement and Maintain State

Question 24Skipped

True or False? terraform validate will validate the syntax of your HCL files.

Correct answer

True

False

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 the correctness of attribute names and value types.

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

Domain

Objective 6 - Use the Core Terraform Workflow

Question 25Skipped

You have an existing resource in your public cloud that was deployed manually, but you want the ability to reference different attributes of that resource throughout your configuration without hardcoding any values. How can you accomplish this?

Run a terraform state list to find the *resource_id* of the resource you need the attributes from. Reference that *resource_id* throughout your configuration to get the exported attributes as needed.

Create a new variable block within your configuration. Add the *resource_id* as the default value and reference the variable using var.<name> throughout your configuration as needed.

Correct answer

Add a data block to your configuration to query the existing resource. Use the available exported attributes of that resource type as needed throughout your configuration to get the values you need.

Create a new resource block that matches the exact configuration of the existing resource. Run a terraform apply to import the resource. Use the available exported attributes of that resource throughout your configuration as needed.

Overall explanation

Anytime you need to reference a resource that is NOT part of your Terraform configuration, you need to query that resource using a data block - assuming a data source is available for that resource_type. Once you add the data block to your configuration, you will be able to export attributes from that data block using interpolation like any other resource in Terraform. For example, if you had an AWS S3 bucket, you could get information using a data block that looked like this:

```
1. data "aws_s3_bucket" "data_bucket" {
2.   bucket = "my-data-lookup-bucket-btk"
3. }
```

Once you add the data block, you can refer to exported attributes like this: data.aws_s3_bucket.data_bucket.arn

WRONG ANSWERS:
None of the wrong answers would allow you to import or query information so that Terraform can use it through interpolation.

https://learn.hashicorp.com/tutorials/terraform/data-sources

https://registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/s3_bucket

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 26Skipped

After running terraform apply, you notice some odd behavior and need to investigate. Which of the following environment variables will configure Terraform to write more detailed logs to assist with troubleshooting?

TF_LOGS=ERROR

LOG_CONFIG=INFO

TF_LOG_CONFIG=WARN

Correct answer

TF_LOG=TRACE

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.

WRONG ANSWERS:

None of the incorrect answers are valid environment variables that you can use to configure Terraform logs.

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

https://developer.hashicorp.com/terraform/cli/config/environment-variables#tf_log

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 27Skipped

Which of the following best describes the primary use of Infrastructure as Code (IaC)?

combining disparate technologies and various tasks into a single workflow

ensuring that applications remain in the desired state configuration

Correct answer

the ability to programmatically deploy and configure resources

ensures that the operations team understands how to develop code

Overall explanation

The primary use case for IaC is to deploy and configure resources in almost any environment in a single, unified way that also abstracts the user from the APIs.

Wrong Answers:

Combing disparate technologies and tasks is really the job of a pipeline, such as GitLab CI/CD, Jenkins, or Azure DevOps. While Terraform CAN be used with multiple technologies within a single configuration file, it's not really the ideal job for Terraform. Terraform isn't an orchestrator like the aforementioned tools would be used for.

The goal of Terraform is NOT to ensure that operations folks know how to develop code, although I'd say that is somewhat of an end result in most organizations. While they are developing Java or Golang applications, operations folks tend to use Terraform as an opportunity to learn more developer-centric workflows, like using Git or learning how to develop code in a repeatable fashion.

When deploying Terraform, it's often a one-time thing, and Terraform doesn't actively monitor applications for changes. That's the job of a configuration management tool, such as Ansible, Chef, Puppet, or SaltStack.

https://developer.hashicorp.com/terraform/intro/use-cases

https://developer.hashicorp.com/terraform/intro/vs/chef-puppet

https://developer.hashicorp.com/terraform/intro/vs

Domain

Objective 1 - Understand Infrastructure as Code Concepts

Question 28Skipped

You are using modules to deploy various resources in your environment. You want to provide a "friendly name" for the DNS of a new web server so you can simply click the CLI output and access the new website. Which of the following code snippets would satisfy these requirements?

Add the following code to the web module:

```
1. output "website" {
2.   description = "Outputs the URL of the provisioned website"
3.   value       = "https://${aws_instance.web.public_dns}:8080/index.html"
4. }
```

Correct answer

Add the following code to the parent module:

```
1. output "website" {
2.   description = "Outputs the URL of the provisioned website"
3.   value       = "https://${module.web.public_dns}:8080/index.html"
4. }
```

Add the following code to the web module:

```
1. output "website" {
2.   description = "Outputs the URL of the provisioned website"
3.   value       = module.web.public_dns
4. }
```

Add the following code to the parent module:

```
1. output "website" {
2.   description = "Outputs the URL of the provisioned website"
```

3. value = aws_instance.web.public_dns
4. }

Overall explanation

When working with outputs, you need to determine where the value will be coming from and work your way backward from there. For example, if the resource was created inside of a module, then the module will require an output block to export that value. That said, output blocks that are created in a module aren't displayed on the Terraform CLI. Therefore, you need to create an output block in the parent/calling module to output the value while referencing the output in the module. Because of this, the correct answer requires you to create an output in the parent module and reference the output value from the module.

WRONG ANSWERS:

****

Add the following code to the web module:

1. output "website" {
2. description = "Outputs the URL of the provisioned website"
3. value = "https://${aws_instance.web.public_dns}:8080/index.html"
4. }

While this could be a way to get the proper URL, the output of a module wouldn't show up in the CLI output, therefore this is incorrect.

****

Add the following code to the parent module:

1. output "website" {
2. description = "Outputs the URL of the provisioned website"
3. value = aws_instance.web.public_dns
4. }

The resource was created inside of the web module, therefore you wouldn't be able to access their attributes directly from the parent module, making this an incorrect answer.

****

Add the following code to the web module:

1. output "website" {
2. description = "Outputs the URL of the provisioned website"
3. value = module.web.public_dns
4. }

Even if you could output the value from a module to the CLI, the resource ID for the module is incorrect because it is referring to another module, making this answer incorrect.

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

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 29Skipped

You need to ensure your Terraform is easily readable and follows the HCL canonical format and style. In the current directory, you have a main.tf that calls modules stored in a modules directory. What command could you run to easily rewrite your Terraform to follow the HCL style in both the current directory and all sub-directories?

terraform fmt -diff

terraform fmt -list=false

Correct answer

terraform fmt -recursive

terraform fmt -check

Overall explanation

By default, fmt scans the current directory for configuration files and formats them according to the HCP canonical style and format. However, if you need it to also scan and format files in sub-directories, you can use the -recursive flag to instruct terraform fmt to also process files in subdirectories.

WRONG ANSWERS:

None of the wrong answers would instruct terraform fmt to scan subdirectories. All of these are, however, other valid flags that you can use with terraform fmt.

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

https://developer.hashicorp.com/terraform/language/syntax/style

https://developer.hashicorp.com/terraform/cli/commands/fmt#command-fmt

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 30Skipped

True or False? min, max, format, join, trim, and length are examples of different expressions in Terraform.

True

Correct answer

False

Overall explanation

These are actually examples of Terraform functions, not expressions. Expressions would be something more in the line of string, number, bool, null, etc.

For the exam, you should go through these and understand what they do at a high level as you could get questions on using a few of them. Check out the built-in functions for Terraform here - https://developer.hashicorp.com/terraform/language/functions

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 31Skipped

Which of the following is the best description of a dynamic block?

declares a resource of a given type with a given local name

requests that Terraform read from a given data source and export the result under the given local name

Correct answer

produces nested configuration blocks instead of a complex typed value

exports a value exported by a module or configuration

Overall explanation

A dynamic block acts much like a for expression, but produces nested blocks instead of a complex typed value. It iterates over a given complex value and generates a nested block for each element of that complex value. You can dynamically construct repeatable nested blocks like setting using a special dynamic block type, which is supported
inside resource, data, provider, and provisioner blocks.

WRONG ANSWERS:

**** declares a resource of a given type with a given local name* =** this is the definition of a resource block

**** requests that Terraform read from a given data source and export the result under the given local name* =** this is a data block

**** exports a value exported by a module or configuration* =** this is an output block

https://developer.hashicorp.com/terraform/language/expressions/dynamic-blocks

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 32Skipped

Which of the following code snippets will properly configure a Terraform backend?

```
1.  backend "s3" {
2.    bucket = "krausen-bucket"
3.    key    = "terraform/"
4.    region = "us-west-2"
5.   }
6.  }
```

```
1.  data "terraform_remote_state" "btk" {
2.    backend = "etcd"
3.    config = {
4.      path     = "terraform/terraform.tfstate"
5.      endpoints = "http://server1:4001"
6.    }
7.  }
```

```
1.  provider "consul" {
2.    address = "consul.btk.com"
3.    scheme  = "https"
4.    path    = "terraform/"
5.    }
6.  }
```

Correct answer

```
1.  terraform {
2.    backend "remote" {
3.      hostname = "app.terraform.io"
4.      organization = "btk"
5.
6.    workspaces {
7.      name = "bryan-prod"
8.    }
```

9. }
10. }

Overall explanation

Backends are configured with a nested backend block within the top-level terraform block. There are some important limitations on backend configuration:

* A configuration can only provide one backend block.

* A backend block cannot refer to named values (like input variables, locals, or data source attributes).

WRONG ANSWERS:

None of the wrong answers are correct, since the state is ONLY configured inside of the terraform block.

https://developer.hashicorp.com/terraform/language/settings/backends/configuration#using-a-backend-block

Domain

Objective 7 - Implement and Maintain State

Question 33Skipped

Which of the following statements are ***true*** about using terraform import? (select three)

Correct selection

using terraform import will bring the imported resource under Terraform management and add the new resource to the state file

Correct selection

the resource address (example: *aws_instance.web*) and resource ID (example: *i-abdcef12345*) must be provided when importing a resource

Correct selection

you must update your Terraform configuration for the imported resource *before* attempting to import the resource

the terraform import command will automatically update the referenced Terraform resource block after the resource has been imported to ensure consistency

Overall explanation

terraform import can be used to import resources into Terraform so they can be managed by Terraform moving forward. Any resources that are imported will be added to Terraform state so

they can be managed like any other resource. Before you can use the terraform import command, you MUST develop the resource block for the resource that will be imported. For example, if you are planning to import an Azure virtual machine, you must add an azurerm_virtual_machine block with the proper configurations.

When you run the terraform import command, you will need to reference the resource address - like *azure_virtual_machine.web-server* - and the resource ID - like the ID of the virtual machine in Azure - as the two required parameters.

terraform import azurerm_virtual_machine.web-server 090556DA-D4FA-764F-A9F1-63614EDA019A

WRONG ANSWER:

IMPORTANT - terraform import will ***NOT*** create the resource block for you. You must create the resource block in your Terraform configuration *before* using the import command

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

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

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

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 34Skipped

Thomas has recently developed a new Terraform configuration in a new working directory and is very cost-conscious. After running a terraform init, how can Thomas perform a dry run to ensure Terraform will create the right resources without deploying real-world resources?

run terraform output

run terraform apply -refresh-only

run terraform show

Correct answer

run terraform plan -out=thomas

Overall explanation

To perform a dry-run of your Terraform configuration, you should run a terraform plan. The entire purpose of running a terraform plan is to validate the change(s) to your infrastructure before you apply the change. In this case, Thomas could see what resources would be created without actually deploying the resources and costing him money.

WRONG ANSWERS:

Running a terraform apply -refresh-only would not give you the desired output. This command is used to update the state file for existing resources deployed with Terraform. This would be

useful if somebody made a change outside of Terraform, and you needed to reflect that change in the state file.

Using terraform output wouldn't work because this command is used to view any outputs defined in your Terraform code. You can also use terraform output <output name> to view more detailed information about a particular output.

Running a terraform show would not give you what you're looking for here. This command displays the output from a state or plan file.

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

https://learn.hashicorp.com/tutorials/terraform/aws-build?in=terraform/aws-get-started

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

Domain

Objective 6 - Use the Core Terraform Workflow

Question 35Skipped

When using Terraform, where can you install providers from? (select four)

Correct selection

Terraform registry

Correct selection

official HashiCorp releases site

Correct selection

plugins directory

the provider's source code

Correct selection

Terraform plugin cache

Overall explanation

Providers can be installed using multiple methods, including downloading from a Terraform public or private registry, the official HashiCorp releases page, a local plugins directory, or even from a plugin cache. Terraform cannot, however, install directly from the source code.

Check out this site for more information on provider installation.

How to find a provider

Domain

Objective 3 - Understand Terraform Basics

Question 36Skipped

True or False? Terraform can only manage dependencies between resources if the depends_on argument is explicitly set for the dependent resources.

Correct answer

False

True

Overall explanation

The most common source of dependencies is an implicit dependency between two resources or modules. That means that Terraform builds a dependency map (aka resource graph) to help determine what resources it can create in parallel, and what resources are dependent on others based on interpolation used within the configuration.

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

https://www.youtube.com/watch?v=Ce3RNfRbdZ0

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 37Skipped

When initializing Terraform, you notice that Terraform's CLI output states it is downloading the modules referenced in your code. Where does Terraform cache these modules?

in the /tmp directory on the machine executing Terraform

in a /modules directory in the current working directory

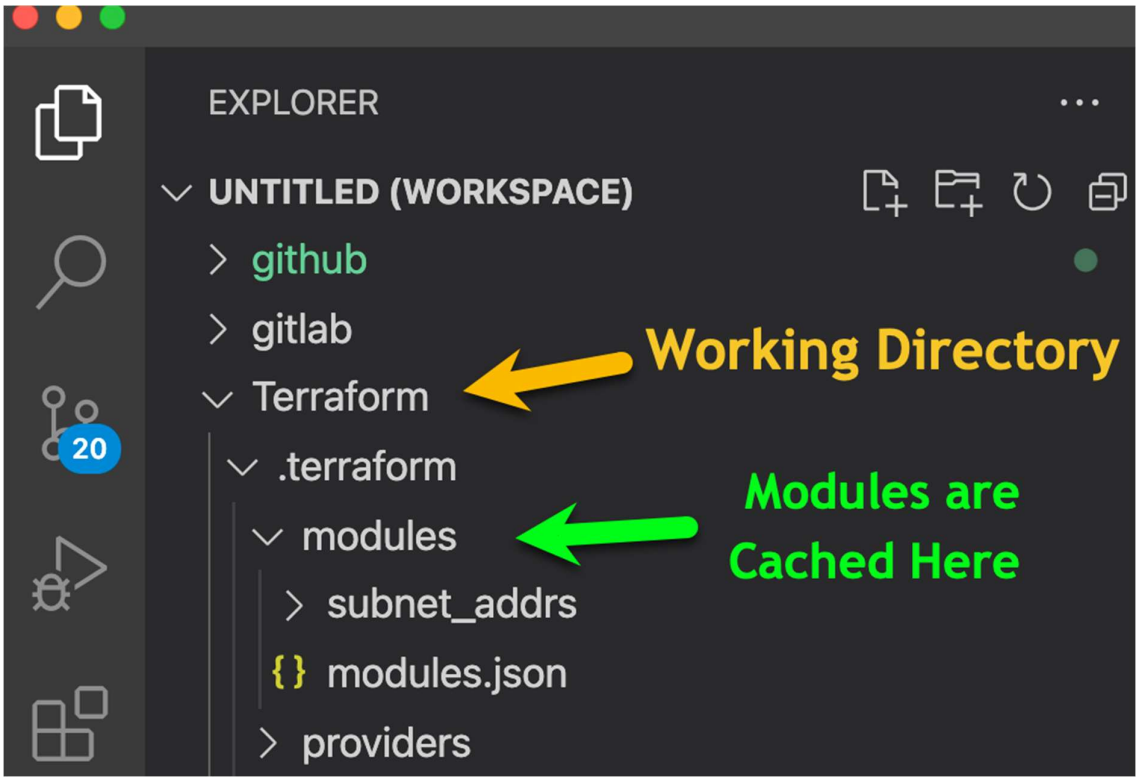
Correct answer

in the .terraform/modules subdirectory in the current working directory

in the /downloads directory for the user running the terraform init

Overall explanation

The .terraform directory contains the modules and plugins used to provision your infrastructure. These files are specific to a specific instance of Terraform when provisioning infrastructure, not the configuration of the infrastructure defined in .tf files.



https://learn.hashicorp.com/tutorials/terraform/module-create?in=terraform/modules

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

Domain

Objective 5 - Interact with Terraform Modules

Question 38Skipped

Which of the following are true regarding Terraform variables? (select two)

Correct selection

the default value will be found in the state file if no other value was set for the variable

the description of a variable will be written to state to help describe the contents of the state file

Correct selection

variables marked as sensitive are still stored in the state file, even though the values are obfuscated from the CLI output

the variable name can be found in the state file to allow for easy searching

Overall explanation

When it comes to working with variables, the value that is used in the Terraform configuration will be stored in the state file, regardless of whether the sensitive argument was set to true. However, the value will not be shown in the CLI output if the value was to be exported by an output block.

WRONG ANSWERS:

Beyond the value, you won't find the variable name or description in the state file because they are simply used on the development side of Terraform, and not the backend operational aspect of how Terraform works.

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

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

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

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 39Skipped

What CLI commands will completely tear down and delete all resources that Terraform is currently managing? (select two)

terraform apply -delete

terraform plan -destroy

Correct selection

terraform destroy

Correct selection

terraform apply -destroy

Overall explanation

The terraform destroy command is a convenient way to destroy all remote objects managed by a particular Terraform configuration.

While you will typically not want to destroy long-lived objects in a production environment, Terraform is sometimes used to manage ephemeral infrastructure for development purposes, in which case you can use terraform destroy to conveniently clean up all of those temporary objects once you are finished with your work.

This command is just a convenience alias for the following 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.

WRONG ANSWERS:

While terraform plan -destroy is a valid command, it only creates a speculative destroy plan to see what the effect of destroying would be

terraform apply -delete is not a valid Terraform command

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

Domain

Objective 6 - Use the Core Terraform Workflow

Question 40Skipped

After using Terraform locally to deploy cloud resources, you have decided to move your state file to an Amazon S3 remote backend. You configure Terraform with the proper configuration as shown below. What command should be run in order to complete the state migration while copying the existing state to the new backend?

```
terraform {
  backend "s3" {
    bucket = "tf-bucket"
    key = "terraform/krausen/"
    region = "us-east-1"
  }
}
```

Correct answer

terraform init -migrate-state

terraform apply -refresh-only

terraform state show

terraform plan -replace

Overall explanation

Whenever a configuration's backend changes, you must run terraform init again to validate and configure the backend before you can perform any plans, applies, or state operations. Re-running init with an already-initialized backend will update the working directory to use the new backend settings. Either -reconfigure or -migrate-state must be supplied to update the backend configuration.

When changing backends, Terraform will give you the option to migrate your state to the new backend. This lets you adopt backends without losing any existing state.

WRONG ANSWERS:

None of the wrong answers would allow you to migrate state. They are simply other CLI commands that are commonly used with Terraform.

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

https://developer.hashicorp.com/terraform/cli/commands/init#backend-initialization

Domain

Objective 7 - Implement and Maintain State

Question 41Skipped

You are using Terraform to manage resources in Azure. Due to unique requirements, you need to specify the version of the Azure provider so it remains the same until newer versions are thoroughly tested.

What block would properly configure Terraform to ensure it always installs the same Azure provider version?

1. required_providers {
2. azurerm = {
3. source = "hashicorp/azurerm"
4. version = "2.90.0"
5. }
6. }

Correct answer

1. terraform {
2. required_providers {
3. azurerm = {
4. source = "hashicorp/azurerm"
5. version = "2.90.0"
6. }
7. }
8. }

1. provider "azurerm" {
2. source = "hashicorp/azurerm"
3. version = "2.90.0"
4. }

1. data "azurerm" {
2. source = "hashicorp/azurerm"

3. version = 2.90.0
4. }

Overall explanation

When you need to constrain the provider to a specific version, you would do this under the terraform configuration block. Within that block, you would use the required_providers block to set certain configurations, including the version of each provider you want to lockdown.

Note that even though you would add the provider constraint under the terraform block, you may still indeed have a separate provider block to set certain configurations, like credentials, regions, or other settings specific to the provider. Just keep in mind that each distinct block is used for different settings.

Example:

Set the version of the Azure provider:

```
terraform {
 required_providers {
  azurerm = {
   source = "hashicorp/azurerm"
   version = "2.90.0"
  }
 }
}
```

Configure settings for the Azure provider:

```
provider "azurerm" {
 features {}
}
```

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

https://developer.hashicorp.com/terraform/language/providers/requirements#version-constraints

Domain

Objective 3 - Understand Terraform Basics

Question 42Skipped

Which of the following Terraform offerings provides the ability to use a private registry?

Terraform OSS/CLI

Correct answer

Terraform Cloud

Overall explanation

You can use Terraform Private Registry with Terraform Cloud (and Enterprise), but not when using Terraform OSS/CLI.

WRONG ANSWER:

You do not have the ability to use a private registry with Terraform OSS.

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

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 43Skipped

By default, Terraform OSS stores its state file in what type of backend?

shared backend

remote backend

Correct answer

local backend

encrypted backend

Overall explanation

The default local backend will be used if you don't specify a backend at all in your Terraform configuration. The local backend stores state on the local filesystem, locks that state using system APIs, and performs operations locally.

Note that you can define the backend to be local by using the backend "local" {} block.

WRONG ANSWERS:

While remote is a valid backend type, shared or encrypted is not a valid backend type. Local is the default and a remote backend must be explicitly configured in your configuration file.

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

Domain

Objective 7 - Implement and Maintain State

Question 44Skipped

A new variable has been created using the list type as shown below. How would you reference terraform in your configuration?

```
variable "products" {
  type = list(string)
  default = [
    "vault",
    "consul",
    "terraform",
    "boundary",
    "nomad"
  ]
}
```

var.list.products[2]

var.default.products["terraform"]

var.products[3]

Correct answer

var.products[2]

Overall explanation

For a collection type such as alist, the provided values are referenced using an index that starts with [0]. In this case, the strings would be represented as such:

1. [0] = vault
2. [1] = consul
3. [2] = terraform
4. [3] = boundary
5. [4] = nomad

When referencing a variable that contains a list, you reference it almost identically to a regular variable, except you just add the index on the end of the variable. So for terraform, it would simply be var.products[2].

WRONG ANSWERS:

var.products[3] = actually equals **boundary** since an index starts with 0

var.list.products[2] = this isn't a valid way to reference a variable in Terraform

var.default.products["terraform"] = this isn't a valid way to reference a list variable in Terraform. If you remove the default from the answer, it could be a valid option if the variable was of type map, though.

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

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 45Skipped

Which of the following is ***not true*** about the terraform.tfstate file used by Terraform?

Correct answer

it always matches the infrastructure deployed with Terraform

it is recommended not to modify the file directly

the file includes information about each resource managed by Terraform

the file can potentially contain sensitive values

Overall explanation

The one thing that cannot be guaranteed is that the terraform.tfstate file **ALWAYS** matches the deployed infrastructure since changes can easily be made outside of Terraform. For example, if you deploy a bunch of resources in GCP and nobody makes any changes, then yes, the terraform.tfstate file does match the current state of those resources. However, if an engineer makes a change in the GCP console or CLI, then the terraform.tfstate would **NOT** match the infrastructure deployed until you ran a terraform apply -refresh-only command.

This is why the only false statement in this question is: ***it always matches the infrastructure deployed with Terraform*.**

WRONG ANSWERS:

The following statements are TRUE about Terraform, which makes them the incorrect choice for this question.

Terraform uses the terraform.tfstate file to store everything it needs to manage the resources it is managing. This includes a ton of information about each resource it provisions and manages. Because of this, HashiCorp recommends that you DO NOT modify the file directly outside of using the Terraform workflow (terraform init, plan, apply, destroy) and terraform state CLI commands.

Many times, you'll need to provide sensitive values to deploy and manage resources, or Terraform may retrieve sensitive values at your request (like data blocks). In that case, these values may get saved to the state file, therefore you should limit who can access the state file to protect this sensitive data.

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

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

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

Domain

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

Question 46Skipped

Which of the following tasks does terraform init perform? (select three)

creates a sample Terraform configuration file in the working directory

updates your state file based on any new changes

Correct selection

downloads required providers used in your configuration file

Correct selection

caches the source code locally for referenced modules

Correct selection

prepares the working directory for use with Terraform

Overall explanation

The terraform init command performs several different initialization steps in order to prepare the current working directory for use with Terraform. Some of these steps include downloading any referenced providers (like AWS, Azure, GCP, etc.), caching the source code for modules in the local directory so they can be used, and other steps to prepare the working directory to be used with Terraform.

Note that there are quite a few options that you can use with terraform init to perform operations that you might need when using Terraform. These operations might include state migrations or upgrading providers.

WRONG ANSWERS:

terraform init does NOT create a sample Terraform configuration file. Actually, I don't know if there are any native Terraform commands that will create a .tf file for you.

You can run terraform init over and over again and it will not change/modify your state file.

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

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

Domain

Objective 6 - Use the Core Terraform Workflow

Question 47Skipped

True or False? In most cases, you can move Terraform state between supported backends at any time, even after running your first terraform apply.

False

Correct answer

True

Overall explanation

You can change your backend configuration at any time. You can change both the configuration itself as well as the type of backend (for example from "consul" to "s3"). However, there may be limitations when migrating certain backends to others. For example, you may not be able to move from TFC to a local backend as it might not be supported.

Terraform will automatically detect any changes in your configuration and request a reinitialization. As part of the reinitialization process, Terraform will ask if you'd like to migrate your existing state to the new configuration. This allows you to easily switch from one backend to another.

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

Domain

Objective 7 - Implement and Maintain State

Question 48Skipped

You have declared a variable named ***db_connection_string*** inside of the ***app*** module. However, when you run a terraform apply, you get the following error message:

1. Error: Reference to undeclared input variable
2.
3. on main.tf line 35:
4. 4: db_path = var.db_connection_string
5.
6. An input variable with the name "db_connection_string" has not been declared. This variable can be declared with a variable "db_connection_string" {} block.

Why would you receive such an error?

Correct answer

since the variable was declared within the module, it cannot be referenced outside of the module

an output block was not created in the module, and therefore the variable cannot be referenced

input variables are not referenced using the var prefix

the variable should be referenced as var.module.app.db_connection_string

Overall explanation

When using modules, it's common practice to declare variables outside of the module and pass the value(s) to the child module when it is called by the parent/root module. However, it's perfectly acceptable to declare a variable inside of a module if you needed. Any variables declared *inside* of a module are only directly referencable within that module. You can't directly reference that variable outside of the module. You can, however, create an output in the module to export any values that might be needed outside of the module.

WRONG ANSWERS:

Output block? While an output block would allow you to get information from within the module, creating an output block still wouldn't allow you to reference the variable directly using the var.<name> nomenclature.

Referencing a Variable You can't reference a variable declared inside of a module, therefore the name shown in this incorrect answer wouldn't work. Ideally, you would create an output inside the module and reference the output rather than the variable inside of the module itself.

Using Interpolation in Terraform This incorrect answer is just plain wrong. In fact, you would reference an accessible variable using the var prefix. Interpolation is the ability to reference data or values within your Terraform code using specific formats. For variables, that format is var.<name>.

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

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

Domain

Objective 5 - Interact with Terraform Modules

Question 49Skipped

Your co-worker has decided to migrate Terraform state to a remote backend. They configure Terraform with the backend configuration, including the type, location, and credentials. However, you want to secure this configuration better.

Rather than storing them in plaintext, where should you store the credentials for the remote backend? (select two)

Correct selection

environment variables

on the remote system

use a variable

Correct selection

credentials file

Overall explanation

Some backends allow providing access credentials directly as part of the configuration for use in unusual situations, for pragmatic reasons. However, in normal use, HashiCorp ***does not*** recommend including access credentials as part of the backend configuration. Instead, leave those arguments completely unset and provide credentials via the credentials files or environment variables that are conventional for the target system, as described in the documentation for each backend.

WRONG ANSWERS:

Use a variable? Well, you could use a variable but that wouldn't really improve security here, since variable defaults or configurations are also stored in plaintext.

On the remote system? I don't think this is even a viable option. The creds would need to be read by the local system that is executing Terraform.

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

Domain

Objective 7 - Implement and Maintain State

Question 50Skipped

You are using Terraform Cloud to manage a new data analytics environment for your organization. You have decided to use Sentinel to enforce standardization and security controls. At what step are the Sentinel policies enforced during a run?

before **the plan phase has started to compare the changes to the existing infrastructure**

after **the apply phase has completed any required changes**

Correct answer

after **the plan, run tasks, cost estimation phases but *before* the apply phase**

before **the OPA policies have been evaluated**

Overall explanation

Sentinel policy evaluations occur ***after Terraform completes the plan and after both run tasks and cost estimation***. This order lets you write Sentinel policies to restrict costs based on the data in the cost estimates.

OPA policy evaluations are slightly different and occur after Terraform completes the plan and after any run tasks. Unlike Sentinel policies, Terraform Cloud evaluates OPA policies immediately before cost estimation.

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

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 51Skipped

True or False? If you have properly locked down access to your state file, it is safe to provide sensitive values inside of your Terraform configuration.

True

Correct answer

False

Overall explanation

Ok, so this was sort of a trick question because locking down your state file really has nothing to do with storing sensitive values inside of your Terraform configuration. Remember that most, if not all, of your configuration, will likely be committed to a code repository. Anybody, or any machine, with access to that code repo would now be able to read the sensitive values that were hardcoded in your Terraform configuration.

Best practice here is to provide your sensitive values OUTSIDE of Terraform, like storing and retrieving them from a secrets management platform like Vault, or using environment variables.

https://learn.hashicorp.com/tutorials/terraform/sensitive-variables?in=terraform/configuration-language

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 52Skipped

You are using Terraform OSS and need to spin up a copy of your GCP environment in a second region to test some new features. You create a new workspace. Which of the following is true about this new workspace? (select four)

Correct selection

it has its own state file

Correct selection

it uses the same Terraform code in the current directory

Correct selection

you can use a different variables file for this workspace if needed

Correct selection

changes to this workspace won't impact other workspaces

it uses a different Terraform backend

Overall explanation

Terraform workspaces (OSS) allow you to create a new workspace to execute the same Terraform but with a ***different state file***. This feature will enable you to run the same Terraform with different configurations without modifying Terraform code or impacting any existing

workspaces. Terraform states out with the default workspace, and that's the workspace you are using unless you create and switch to a new workspace.

IMPORTANT - PLEASE READ:

Remember that Terraform Cloud and Enterprise also have Workspaces, but they behave slightly differently. In Cloud and Ent, each workspace is still isolated from others, meaning it has its own state. Still, often these workspaces point to different code repositories and use completely different Terraform configuration files.

To create a new workspace, you'd run:

1. $ terraform workspace new btk
2.
3. Created and switched to workspace "btk"!
4.
5. You're now on a new, empty workspace. Workspaces isolate their state, so if you run "terraform plan" Terraform will not see any existing state for this configuration.

To list all of the existing workspaces, you can run (note the * indicates the workspace you are using):

1. $ terraform workspace list
2.
3. default*
4. btk
5. bryan-dev
6. temp-workspace

WRONG ANSWER:

When using workspaces, you're essentially using the same Terraform configuration files. Therefore the backend will remain the same for all of your workspaces. That makes this answer incorrect.

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

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

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

Domain

Objective 4 - Use the Terraform Outside Core Workflow

Question 53Skipped

True or False? In both Terraform OSS and Terraform Cloud, workspaces provide similar functionality of using a separate state file for each workspace.

False

Correct answer

True

Overall explanation

This is true. When you create a new workspace using Terraform OSS/CLI using the terraform workspace new command, you will be working with a separate state file when working with that workspace. You can easily change between workspaces and their respective state file using the terraform workspace select command.

The same is true in Terraform Cloud. When you create a new workspace, you'll be working with a dedicated state file for that particular workspace. It doesn't share a state file with any other workspace.

https://developer.hashicorp.com/terraform/cli/workspaces#the-purpose-of-workspaces

https://developer.hashicorp.com/terraform/cli/workspaces#interactions-with-terraform-cloud-workspaces

Domain

Objective 9 - Understand Terraform Cloud Capabilities

Question 54Skipped

When using collection types for variables in Terraform, which of the following ***two*** statements are true? (select two)

maps are defined inside of square brackets, like this: [name = "John" age = 52]

Correct selection

lists are defined inside of square brackets, like this: ["value1", "value2", "value3"]

lists are defined inside of curly braces, like this: {"value1", "value2", "value3"}

Correct selection

maps are defined inside of curly braces, like this: { name = "John" age = 52 }

Overall explanation

Lists/tuples are represented by a pair of ***square brackets*** containing a comma-separated sequence of values, like ["a", 15, true].

Maps/objects are represented by a pair of ***curly braces*** containing a series of <KEY> = <VALUE> pairs.

https://developer.hashicorp.com/terraform/language/expressions/types#lists-tuples

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 55Skipped

Your team is using Terraform, and multiple team members need to be able to manage the infrastructure. You need to support state locking to reduce the chance of corrupting the state file. What backends can you use to meet these requirements? (select three)

Correct selection

kubernetes backend

Correct selection

s3 backend (with DynamoDB)

Correct selection

consul backend

local backend

Overall explanation

Not all Terraform backends are created equal. Some backends act like plain "remote disks" for state files; others support *locking* the state while operations are being performed, which helps prevent conflicts and inconsistencies.

Kubernetes, Consul, and S3 backends all support state locking. S3 supports state locking with the help of DynamoDB.

WRONG ANSWER:

While the local backend does support locking via system APIs, you can't use the local backend to share the state across your team.

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

Domain

Objective 7 - Implement and Maintain State

Question 56Skipped

Given the code snippet below, how would you refer to the value of ip of an environment when using a for_each argument in a resource block?

1. variable "env" {

```
2.  type = map(any)
3.  default = {
4.   prod = {
5.    ip = "10.0.150.0/24"
6.    az = "us-east-1a"
7.   }
8.   dev  = {
9.    ip = "10.0.250.0/24"
10.   az = "us-east-1e"
11.  }
12. }
13. }
```

Correct answer

each.value.ip

var.env["dev.ip"]

var.env.dev.ip

each.dev.ip

Overall explanation

Sort of testing two different things here - a complex map variable plus the for_each argument.

A for_each argument will iterate over a map or set of strings and create a similar instance/resource for each item in the map or set. In our case, the map is the input variable and the "each" would be the higher-level map, so prod and dev. Underneath each value, there are two arguments, both az and ip that you can choose from.

The input variable that is shown in this example is essentially a map of maps.

WRONG ANSWERS:

None of the wrong answers are valid ways to reference the values provided by the input variable.

https://developer.hashicorp.com/terraform/language/meta-arguments/for_each

https://learn.hashicorp.com/tutorials/terraform/for-each

Domain

Objective 8 - Read, Generate, and Modify Configuration

Question 57Skipped

True or False? The terraform graph command can be used to generate a visual representation of a configuration or execution plan.

False

Correct answer

True

Overall explanation

The terraform graph command is used to generate a visual representation of either a configuration or execution plan. The output is in the DOT format, which can be used by GraphViz to generate charts.

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

Domain

Objective 8 - Read, Generate, and Modify Configuration