

# Terraform Associate Exam Dumps

## Practice Set #1

### 1. How does Terraform can help in failover to a different datacenter or a cloud environment?

- Terraform can be an essential part of your disaster recovery strategy because it helps you stand up new infrastructure very quickly and efficiently.
- Terraform allows cheaper and much more dynamic disaster recovery plans because you can literally build a DR environment on the fly; hence it may not make sense to keep it running all the time and pay a massive bill for it.

2. User "Stewie" will get deleted, and the user "Chris" will get renamed to "Brian"

### 5. Remote state allows teams to share infrastructure resources in a read-only way without relying on any additional configuration store.

6. If both the type and default arguments are specified, the given default value must be convertible to the specified type. In the example given below –

```
variable "amis" {  
  type = map  
  default = [] # This must be a valid map type instead of list.  
}
```

7. The **local-exec provisioner** invokes a local executable after a resource is created. This invokes a process on the machine running Terraform, not on the resource.

8. A new provider can be added to a configuration — either explicitly via a provider block or by adding a resource from that provider — Terraform must initialize it before it can be used. Initialization downloads and installs the provider's plugin so that it can later be executed.

terraform init command will download and initialize any providers that are not already initialized.

|                            |  |
|----------------------------|--|
| Terraform v0.12 or earlier | Cannot automatically download third-party providers. |
| Terraform v0.13 or later   | Can automatically download third-party providers.    |

9. Connection blocks don't take a block label, and can be nested within either a resource or a provisioner.

10. To specify a Branch

```
module "vpc" {  
  source = "git::https://code.quizexperts.com/vpc.git#ref=hotfix"  
}
```

11. Terraform cannot reason about what the provisioner does; hence if a creation-time provisioner fails for a resource, Terraform will plan to destroy and recreate resource upon the

next terraform apply.

**If a creation-time provisioner fails, the resource is marked as tainted. A tainted resource will be planned for destruction and recreation upon the next terraform apply.**

**12.** All arguments, including the username and password, will be stored in the raw state as plain-text.

**13.** Private registry modules have source strings of the form `///`. This is the same format as the public registry, but with an added hostname prefix.

Source = `<HOSTNAME>/<NAMESPACE>/<NAME>/<PROVIDER>`

**14.** Enable Terraform state locking for the Amazon S3 backend by using a DynamoDB table to avoid race around the condition and concurrent execution problem of the same Terraform configuration.

**15.** Terraform retains a copy of the most recent set of dependencies within the state. Even if you delete one or more items from the configuration, Terraform can still determine what to delete and the correct order for destruction from the state.

**16.** Which of the following variable type allows multiple values of several distinct types to be grouped together as a single value?

- Tuple
- Object

**17.** When you declare variables in the root module of your configuration, you can set their values using CLI options and environment variables. When you declare them in child modules, the calling module should pass values in the module block.

**18.** Every Terraform configuration has at least one module, known as its root module, which consists of the resources defined in the `.tf` files in the main working directory.

**19.** Terraform **tracks** resources **by their name**. If you change the name, you have created a new resource and deleted the old resource. There is no way for Terraform to tell you changed the name of an existing resource since its all just text. Hence even if the user has just changed the resource name, Terraform will delete the existing VM and create a new VM with a new name.

**21.** Which command automatically updates configurations in the current directory for easy readability and consistency?

- `terraform fmt`

**22.** Named **terraform-`<PROVIDER>-<NAME>`**. Module repositories must use this three-part name format, where `<NAME>` reflects the type of infrastructure the module manages and `<PROVIDER>` is the main provider where it creates that infrastructure. The `<NAME>` segment can contain additional hyphens. Examples: `terraform-google-vault` or `terraform-aws-ec2-instance`.

**where `<NAME>` segment can contain additional hyphens.**

## **22. Workspace**

A common use for multiple workspaces is to create a parallel, distinct copy of a set of infrastructure in order to test a set of changes before modifying the main production infrastructure. For example, a developer working on a complex set of infrastructure changes might create a new temporary workspace in order to freely experiment with changes without affecting the default workspace.

Workspaces are technically equivalent to renaming your state file. They aren't any more complex than that. Terraform wraps this simple notion with a set of protections and support for remote state.

For local state, Terraform stores the workspace states in a directory called `terraform.tfstate.d`. This directory should be treated similarly to local-only `terraform.tfstate`

So for non-default workspaces state will be stored in a file –  
`terraform.tfstate.d//terraform.tfstate`

**23.** 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.

#### **24. When To Use Local Values**


Local values can be helpful to avoid repeating the same values or expressions multiple times in a configuration, but 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.

## Declaring a Local Value

A set of related local values can be declared together in a single `locals` block:

```
locals {  
  service_name = "forum"  
  owner        = "Community Team"  
}
```

Copy 

Example:

```
locals {  
  # Common tags to be assigned to all resources  
  common_tags = {  
    Service = local.service_name  
    Owner   = local.owner  
  }  
}  
  
resource "aws_instance" "example" {  
  # ...  
  
  tags = local.common_tags  
}
```

25. The behavior of this lock is dependent on the backend being used. Local state files cannot be unlocked by another process.
26. All security measures you must take to protect sensitive data such as username and password?
  - Strictly control who can access your Terraform backend.
  - If you manage any sensitive data with Terraform (like database passwords, user passwords, or private keys), treat the state itself as sensitive data, always encrypt state at rest, and protect it with TLS in transit.
  - Storing state remotely can provide better security.
29. The terraform import command currently can only import **one resource at a time**.

The terraform import command currently can only import one resource at a time. This means you can't yet point Terraform import to an entire collection of resources such as an AWS VPC and import all of it. This workflow will be improved in a future version of Terraform.

**30.** The expression of a local value can refer to other locals, but as usual reference cycles are not allowed.

**31.** Sentinel is a **proactive** service that prevents the provisioning of out-of-policy infrastructure.

**32. Primitive type** constraint in Terraform?

- String
- Bool

**33.** Your manager has asked you to create a strong separation between development and production deployments to ensure users will only touch the intended infrastructure. Any change in one deployment should not impact another deployment. Which of the following, you think is the right approach?

**Use separate working-directories for each deployment. Each working directory should have its separate deployment-specific configuration and backend with different credentials and access controls.**

**34.** Why does the traditional approach of managing infrastructure not go well with large-scale, distributed systems architectures?

- Logging it into a management portal or an administrative console, and pointing and clicking to provision large infrastructure is slow and prone to human error.
- The traditional approach can't keep up with higher elasticity of infrastructure, where resources might live days to weeks instead of months to years.
- The traditional approach was okay if we didn't have to manage a lot of infrastructure. But it does not fit well where The scale of infrastructure is much larger, because instead of a handful of large instances, we might have many smaller instances, so there are many more things we need to provision.

**35.** Manually Tainting Resources

In cases where you want to manually destroy and recreate a resource, Terraform has a built-in taint function in the CLI. This command will not modify infrastructure, but does modify the state file in order to mark a resource as tainted. Once a resource is marked as tainted, the next plan will show that the resource will be destroyed and recreated and the next apply will implement this change.

- **terraform taint "aws\_instance.skillcertpro\_lab[1]"**
- **terraform apply**

**36.** If you are a customer of HashiCorp's Terraform Enterprise/ Terraform Cloud, you can have this same experience with a **Private Terraform Registry**; that is, **a registry that lives in your private Git repos**, and is only accessible to your team. This can be a great way to share modules within your company.

The Terraform open source project does not provide a server implementation, but community members can create their own private registries by following the published protocol.

**37.** What does the below-given command do? `export TF_LOG=`

**Disable detailed logs.**

To disable, either unset it or set it to empty.

```
# Set it to empty
export TF_LOG=

OR

# Unset it
unset TF_LOG
```

**38. How can you source module from a separate private git repository?**

Arbitrary Git repositories can be used by prefixing the address with the special `git::` prefix. After this prefix, any valid Git URL can be specified to select one of the protocols supported by Git. For example, to use HTTPS or

- `SSH:module "vpc" {source = "git::https://example.com/vpc.git"}`
- `module "storage" {source = "git::ssh://username@example.com/storage.git"}`

**39.** By default, Terraform states are stored in a local file named `terraform.tfstate` under the current working directory, but it can also be stored remotely, which works better in a team environment.

**40.** Your team uses Terraform to manage dev, staging, and production environments. For every update, you have to update each environment containing duplicate blocks of configuration separately. What Terraform features can you use to avoid this growing burden of configuration updates?

- **Modules**

Terraform recommend that every practitioner use modules by following these best practices:

- Start writing your configuration with modules in mind. Even for modestly complex Terraform configurations managed by a single person, you'll find the benefits of using modules outweigh the time it takes to use them properly.
- Use local modules to organize and encapsulate your code. Even if you aren't using or publishing remote modules, organizing your configuration in terms of modules from the beginning will significantly reduce the burden of maintaining and updating your configuration as your infrastructure grows in complexity.
- Use the public Terraform Registry to find useful modules. This way, you can more quickly and confidently implement your configuration by relying on the work of others to implement common infrastructure scenarios.
- Publish and share modules with your team. Most infrastructure is managed by a team of people, and modules are an important way for teams to work together to create and maintain infrastructure. As mentioned earlier, you can publish modules either publicly or privately.

**41.** Which of the following options can enable the most verbose logs and write those logs to a file?

- `export TF_LOG="TRACE" export TF_LOG_PATH="terraform.txt"`

- `export TF_LOG="RANDOM" export TF_LOG_PATH="terraform.txt"`

Terraform has detailed logs that 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. `TRACE` is the most verbose and it is the default if `TF_LOG` is set to something other than a log level name.

**42.** Providers can be passed down to descendent modules in two ways: either implicitly through inheritance, or explicitly via the providers argument within a module block. These two options are discussed in more detail in the following sections.

**44.** The Terraform language uses the following types for its values:

- **string:** a sequence of Unicode characters representing some text, like "hello".
- **number:** a numeric value. The numbertype 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}.

**45.** A "backend" in Terraform determines how state is loaded and how an operation such as apply is executed.

#### **46. What are the advantages of using Infrastructure as Code?**

- With Infrastructure as Code, users can write configuration once and then reuse it many times.
- Infrastructure as Code can be an essential part of your disaster recovery strategy. It helps you stand up new infrastructure quickly and efficiently.
- Infrastructure as Code provides the ability to gracefully roll the infrastructure back to a last known good state.
- Infrastructure as Code codifies manual pointing and clicking on management console and automate it.
- Infrastructure as Code provides version-control of infrastructure configuration to provide a single source of truth.

#### **47. Multiple Provider Configurations**

You can optionally define multiple configurations for the same provider, and select which one to use on a per-resource or per-module basis.

```
# Additional provider configuration for west coast region; resources can
# reference this as `aws.west`.

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

**48. Provisioners should only be used as a last resort.** For most common situations there are better alternatives like user\_data in AWS EC2 or tools like Chef/Ansible/Saltstack built specifically for configuration management.

**49. Initially, the backend has only one workspace, called "default",** and thus there is only one Terraform state associated with that configuration.

Terraform starts with a single workspace named "default". This workspace is special both because it is the default and also because **it cannot ever be deleted**. If you've never explicitly used workspaces, then you've only ever worked on the "default" workspace.

**Peter Griffin creates two workspaces named dev and prod:**

- Peter can delete dev workspace, but he can not delete the default workspace.
- Each workspace has a separate state file associated with it.
- Peter will have three workspaces named dev, prod, and default.

**50. In regards to Terraform List, Select all correct statements.**

- List is a sequence of values identified by consecutive whole numbers starting with zero.
- ["a", 15, true] is a valid Terraform list.
- All elements of a list must always be of the same type.

**Якщо аргумент** модуля вимагає значення типу list(string), а користувач надає кортеж ["a", 15, true], Terraform внутрішньо перетворить значення в ["a", "15", "true"], конвертуючи елементи до необхідного типу рядкового елемента, оскільки всі елементи колекції повинні мати однаковий тип.

**З іншого боку,** автоматичне перетворення не вдасться, якщо надане значення (включно зі значеннями елементів) несумісне з потрібним типом. Якщо аргумент вимагає тип list(string), а користувач надає кортеж ["a", [], "b"], то значення не може відповідати обмеженню типу, Terraform відхилить це значення, скаржачись на те, що всі елементи повинні мати однаковий тип, оскільки порожній кортеж не може бути перетворений в рядок.

**51. When it comes to Terraform 0.12, what does the terraform init command do?**

- Searches for module blocks and the source code for referenced modules is retrieved from the locations given in their source arguments.
- Searches the configuration for direct and indirect references to providers and load plugins distributed by HashiCorp.
- Sees the root configuration directory for backend configuration, and initialized the chosen backend.

**52. The TF\_LOG environment variable can enable detailed Terraform logs to appear on stderr.**

**53. Which of the following describes the right ways of using a local path as module sources?**

Локальний шлях повинен починатися з **./** або **../**, щоб вказати, що це локальний шлях, щоб відрізнити його від адреси реєстру модуля.

- module "consul" { source = "./consul" }
- module "consul" { source = "../consul" }

**54. In regards to Terraform Cloud, Select all correct statements.**



- Terraform Cloud always encrypts state at rest and protects it with TLS in transit.
- Terraform Cloud 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.

**55. Version constraints are supported only for modules installed from a module registry, such as the public Terraform Registry or Terraform Cloud's private module registry.**

Other module sources can provide their own versioning mechanisms within the source string itself, or might not support versions at all. In particular, modules sourced from local file paths do not support version; since they're loaded from the same source repository, they always share the same version as their caller.

**56.**

- Terraform can infer when one resource depends on another by analyzing the resource attributes used in interpolation expressions. From this, Terraform builds a dependency tree to determine the correct order to create each resource.
- The order of blocks and expressions in the Terraform configuration file doesn't matter.

**57.** A calling module can directly access child module resource attributes.

- **FALSE**

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.

**58.** You created a Terraform module to launch new VMs in Cloud. During your tests in a dev environment with a few VMs terraform plan and apply commands were running fine. You started using this module on production and have launched hundreds of VMs with it, and with time, you started observing slowness during terraform plan and apply commands execution. What could be the possible reason of slowness?

- **Cloud providers have API rate-limiting, and for a large infrastructure, Terraform has to make a lot of calls to querying every resource; hence API rate-limitation stops Terraform from making many concurrent API calls at once that make a plan and apply command slow.**
- **By default, for every plan and apply, Terraform will sync all resources in your state with the real world to know the current state of resources to effectively determine the changes that it needs to make to reach your desired configuration. For large infrastructures, this behavior of querying every resource makes plan and apply commands execution very slow.**
- **Many cloud providers do not provide APIs to query multiple resources at once, and the round trip time for each resource is hundreds of milliseconds; hence for a large infrastructure querying every resource causes slowness during the plan and apply command.**

**59.** If the given index is greater than the length of the list then the index is "wrapped around" by taking the index modulo the length of the list:

- > element(["a", "b", "c"], 1)  
b
- > element(["a", "b", "c"], 3)  
a

**60. What is the terraform state command used for?**

- Moving items in/out a Terraform state.
- Modifying/Updating Terraform state.
- **Advanced state management.**

**61.** 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.

For example, if the ./app-cluster module referenced in the example above has to export an output value named instance\_ids only then the calling module can reference that result using the expression module.instances.instance\_ids

**62.** A destroy-time provisioner within a resource that is tainted **will not run**. This includes resources that are marked tainted from a failed creation-time provisioner or tainted manually using terraform taint.

**63. The default workspace cannot ever be deleted!**

**64. Anyone can publish and share modules on the Public Terraform Registry.**

**65.** Which Terraform Enterprise feature allows users to create and confidentially share infrastructure modules within an organization?

- **Private Module Registry**

## Practice Set #2

**3.** Terraform has multiple mechanisms for setting variables. If the same variable is assigned multiple values, which value Terraform will use?

- Last

**7.** Sentinel is a policy as a code framework that's integrated into Hashicorp enterprise products. In which phase of a Terraform run Sentinel enforced user-defined policies against infrastructure?

- **Between the plan and apply phases of a Terraform run**

**8.** Workspaces are technically equivalent to renaming your state file. They aren't any more complex than that. Terraform wraps this simple notion with a set of protections and support for remote state.

**9.** To ensure that Terraform workspace names are stored correctly and safely in all backends, the workspace name must be valid to use in a URL path segment without escaping.

- TRUE

**10.** Peter created an EC2 with the configuration given below. DevOps team of his company noticed an unencrypted volume is attached to this EC2 and encrypts it. What will happen if Peter runs terraform apply again on the same configuration?#.....

```
resource "aws_instance" "example" {ami = "ami-2757f631"instance_type =  
"t2.micro"root_block_device {volume_size = 20delete_on_termination = true}}
```

Terraform узгоджує ресурси, що відслідковуються державним файлом, з реальним світом. Вона робить це шляхом запитів до постачальників інфраструктури, щоб з'ясувати, що насправді працює і поточну конфігурацію, і оновлює файл стану з цією новою інформацією. Terraform призначений для співіснування з іншими інструментами і ресурсами, що надаються вручну, тому він оновлює тільки ті ресурси, які знаходяться під його управлінням.

Обсяг даних для оновлення мінімальний. Terraform перераховує кожен ресурс, який він оновлює, разом з його внутрішнім ідентифікатором. Запуск оновлення не змінює інфраструктуру, але змінює файл стану. Якщо стан змінився з моменту останнього запуску Terraform, оновлення дозволяє виявити цей зсув.

План Terraform - це опис всього, що Terraform зробить для реалізації бажаної конфігурації, коли ви застосуєте план. План Terraform створюється автоматично під час застосування, але також може бути створений в явному вигляді.

В останньому прикладі, якщо ми оновили AMI або instance\_type EC2, план тераформи вкаже, що існуючий екземпляр буде знищено і створено заново. Однак, шифрування обсягу призведе до оновлення на місці, яке лише оновить стан.

**11.** Which of the following is right about Terraform?

- Customer teams can use shared Terraform configurations and use them as a black box tool to manage their services.
- Terraform is not limited to physical providers like AWS. Resource schedulers can be treated as a provider, enabling Terraform to request resources from them.
- Terraform can help team creating disposable parallel environments.

**12.** How can you export Terraform debug logs to a file /var/log/terraform.log?

- **export TF\_LOG\_PATH="/var/log/terraform.log"**

**13.** The Terraform language supports user-defined functions.

- **FALSE**

**14.** Why should you constrain the acceptable provider versions in Terraform configuration for production use?

- Providers are plugins released on a separate rhythm from Terraform, so providers have their own version numbers, and version constraint ensures that new versions with breaking changes will not be automatically installed by terraform init in the future.

**15.** Which of the following configuration block has Terraform 0.13 recommended way of provider version constraint?

- terraform { required\_providers { aws = "~> 3.0" } }
- The use of an object version constraint within required\_providers block is recommended in Terraform 0.13 or later.

- The use of explicit source addresses for all providers is recommended in Terraform 0.13 or later.

**14.** A module **can not access** all parent module variables; hence to pass variables to a child module, the calling module should pass specific values in the module block.

Example:

```
module "servers" {  
  source = "../app-cluster"  
  
  servers = 5  
}
```

**15.** What are the features exclusive to Terraform Enterprise. Select all that apply.

- Audit logging
- SAML single sign-on

**16.** What should be the right type for the variable `availability_zone_names` in the configuration given below?

type = \_\_\_\_\_

- list
- list(string)
- list(any)

**19.** If creation-time provisioner fails, Terraform marks the resource as **tainted**.

**20.** Due to some problem, you end up locking the Terraform state. Now your team members are not able to run `terraform apply` command to make any infrastructure changes. What will you do to fix this?

- **Use force-unlock command to unlock the state manually.**

**21.** Additional provider configurations (those with the `alias` argument set) are **never inherited automatically by child modules**, and so must always be passed explicitly using the `providers` map.

**22.** Which of the following command upgrades a Terraform provider to the latest acceptable version?

- **terraform init -upgrade**

**23.** Which command can be used to inspect a plan to ensure that the planned operations are expected?

- **terraform show**

**24.** If the Terraform backend does not have state lock enabled by default, we can use `-lock` flag to automatically enable state locking on all operations that could write state.

- **FALSE**

**25.** Terraform allows a single configuration to be used to manage resources in multiple providers, and even handle cross-cloud dependencies.

- **TRUE**

It's often attractive to spread infrastructure across multiple clouds to increase fault-tolerance. By using only a single region or cloud provider, fault tolerance is limited by the availability of that provider. Having a multi-cloud deployment allows for more graceful recovery of the loss of a region or entire provider.

Realizing multi-cloud deployments can be very challenging as many existing tools for infrastructure management are cloud-specific. Terraform is cloud-agnostic and allows a single configuration to be used to manage multiple providers, and to even handle cross-cloud dependencies. This simplifies management and orchestration, helping operators build large-scale multi-cloud infrastructures.

**26.** The Terraform provisioner block works only within the resource configuration block.

- **TRUE**

**27.** A module must be public to be able to be used in your terraform configuration.

- **FALSE**

The source argument in a module block tells Terraform where to find the source code for the desired child module.

A module can be public or private.

**28.** A user accidentally deleted remote backend from Terraform configuration how it will impact existing resources and state file?

- **terraform apply command will error out. You need to reinitialize Terraform again, and it will prompt you to migrate existing remote state to the local backend so that it can still manage resources managed by the remote state.**

**29.** How will you share state files between team members so that each team member can access the same Terraform state files to be able to use Terraform to update infrastructure?

Configure Amazon S3, Azure Storage, Google Cloud Storage, HashiCorp Consul as backend, or use Terraform Enterprise which works better in a team environment.

**30.** All the paid features:

- Roles/Team Management
- Encrypted Remote State

- Sentinel Policy as Code Management

| Collaborative Infrastructure as Code | OSS | FREE | TEAM & GOVERNANCE | BUSINESS | ENTERPRISE |
|--------------------------------------|-----|------|-------------------|----------|------------|
| Remote State                         |     | ✓    | ✓                 | ✓        | ✓          |
| VCS Connection                       |     | ✓    | ✓                 | ✓        | ✓          |
| Workspace Management                 |     | ✓    | ✓                 | ✓        | ✓          |
| Secure Variable Storage              |     | ✓    | ✓                 | ✓        | ✓          |
| Remote Runs                          |     | ✓    | ✓                 | ✓        | ✓          |
| Private Module Registry              |     | ✓    | ✓                 | ✓        | ✓          |
| Team Management & Governance         |     | FREE | TEAM & GOVERNANCE | BUSINESS | ENTERPRISE |
| Team Management                      |     |      | ✓                 | ✓        | ✓          |
| Sentinel Policy as Code Management   |     |      | ✓                 | ✓        | ✓          |
| Cost Estimation                      |     |      | ✓                 | ✓        | ✓          |

31. The format of the state files are **just JSON**, direct file editing of the state is discouraged.

32. Workspaces in Terraform Cloud and Terraform CLI are the same.

- FALSE

Terraform Cloud and Terraform CLI both have features called “workspaces,” **but they’re slightly different**. CLI workspaces are alternate state files in the same working directory; they’re a convenience feature for using one configuration to manage multiple similar groups of resources.

46. Let’s say your company uses Terraform to create and manage resources on the AWS cloud, and your colleague needs to create two EC2 instances in two different AWS regions and create a Terraform configuration that looks like the code given below and send it to you for review. Looking at the code, What do you think terraform will do? Choose TWO correct answers.

```

provider "aws" {
  region = "us-east-1"
}
provider "aws" {
  alias = "west"
  region = "us-west-2"
}
# 1st VM in us-east-1 region
resource "aws_instance" "skillcertpro_lab_east" {
  ami = "some_ami_id_from_us-east-1"
  instance_type = "t3.micro"
  tags = {
    Name = "SkillCertPro"
  }
}
# 2nd VM in us-west-2 region
resource "aws_instance" "skillcertpro_lab_west" {
  ami = "some_ami_id_from_us-west-2"
  instance_type = "t3.micro"
  tags = {
    Name = "SkillCertPro"
  }
}

```

- Terraform will create the 1st VM in "us-east-1"
- Terraform will try to create the 2nd VM in "us-east-1"

### Correct

The condition given in the question user expects to create two VMs in two different regions; hence two separate provider blocks have been defined, each for a different [region](#). By default, resources use a default provider configuration (one without an alias argument). Hence Terraform will try to create both VMs in "us-east-1". It will create the 1st VM as given AMI will be present in "us-east-1," but it will fail while trying to create the 2nd VM as 2nd AMI will not be visible in the "us-east-1" region.

**Hence after modification configuration given in the question will look like this:**

```

provider "aws" {
  region = "us-east-1"
}
provider "aws" {
  alias = "west"
  region = "us-west-2"
}
# 1st VM in us-east-1 region
resource "aws_instance" "skillcertpro_lab_east" {
  ami = "some_ami_id_from_us-east-1"
  instance_type = "t3.micro"
  tags = {
    Name = "SkillCertPro"
  }
}

```

```

}
}
# 2nd VM in us-west-2 region
resource "aws_instance" "skillcertpro_lab_west" {
  provider = aws.west # Add this to the 2nd VM <-----
  ami = "some_ami_id_from_us-west-2"
  instance_type = "t3.micro"
  tags = {
    Name = "SkillCertPro"
  }
}

```

**48.** Which of the following can we use to make Terraform configuration dynamic and reusable?

- Input Variables

Input variables serve as parameters for a Terraform module, allowing aspects of the module to be customized dynamically without altering the module's own source code, and allowing modules to be shared between different configurations.

**49.** Provider dependencies are created in several different ways:

- Explicit use of a provider block in configuration, optionally including a version constraint.
- Use of any resource belonging to a particular provider in a resource or data block in configuration.
- Existence of any resource instance belonging to a particular provider in the current state.

For example, if a particular resource is removed from the configuration, it continues to create a dependency on its provider until its instances have been destroyed.

**51.** Select all the correct statements about using modules over a monolithic configuration. Choose FOUR correct answers.

- Allows you to share a piece of configuration you have written with your team or the general public, giving them the benefit of your hard work.
- Modules help to provide consistency in your configurations and make it easier to understand.
- Modules enable you to version your configuration. So production might be using version 1 of a module code, whereas dev and staging could be using version 2 or 3, depending on what level of testing you've done and how ready it is for production.
- Allows you to reuse a piece of proven, tested, documented infrastructure.

**52.** You want to use an archive stored in S3 as a module source. Should the archive on s3 always have to be public to be able to be used by Terraform?

```

module "vpc" {
  source = "s3::https://s3-eu-west-1.amazonaws.com/skillcertpro-lab-s3/vpc.zip"
}

```

- **No**

**58.** As given below, Glenn has created a variable of string type. Select all accepted input from the options given below.

```

variable "my_variable" {
  type = string
}

```



```
}
```

- **Both of these**

The Terraform language will automatically convert number and bool values to string values when needed, and vice-versa as long as the string contains a valid representation of a number or boolean value.

- true converts to "true", and vice-versa
- false converts to "false", and vice-versa
- 15 converts to "15", and vice-versa

**59.** An output containing confidential data like a password can be marked as sensitive.

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

Since the sensitive argument is set to true, can the value associated with the password still be available in plain-text in the state file for everyone to read?

- Yes, Setting an output value in the root module as sensitive prevents Terraform from showing its value in the list of outputs at the end of terraform apply . Sensitive output values are still recorded in the state, and so will be visible to anyone who is able to access the state data.

**60.** Each workspace in Terraform Cloud retains backups of its previous state files.

- **TRUE**

Correct

Each workspace in Terraform Cloud retains backups of its previous state files. Although only the current state is necessary for managing resources, the state history can be useful for tracking changes over time or recovering from problems

**61.** By default, a provisioner defined in a resource is a **creation-time** provisioner. If when = destroy is specified, the provisioner will run when the resource it is defined within is destroyed. Creation-Time Provisioners By default, provisioners run when the resource they are defined within is created. Creation-time provisioners are only run during creation, not during updating or any other lifecycle. They are meant as a means to perform bootstrapping of a system.

**64.** The taint command can be used to taint specific resources within a module:

- **terraform taint module.salt\_master\_cluster.module.instance.aws\_instance.salt\_master**

---

### Practice Set #3

**1.** For automation in a CI environment, Lois needs to disable the colored output of the terraform plan command; however, other terraform commands can have colored output. Is there any easier way to modify this default behavior without using -no-color option every time in the command-line?

- **export TF\_CLI\_ARGS\_plan="-no-color"**

2. How does Terraform treat a resource if it was successfully created but failed during provisioning?

- **Terraform does not automatically roll back and destroy the resource during the apply when the failure happens, because that would go against the execution plan.**
- Resource failed during provisioning has been physically created but can't be considered safe to use; hence Terraform will remove this resource and will create new again during next apply.
- Terraform will error and mark the resource as "tainted".

3. Which of the following is the right way to use the min function?

- min(12, 54, 3)
- min([12, 54, 3]...)

4. Which of the following Terraform function will result in an error for the below-given value of variable ami\_id?

Function 1: lookup(var.ami\_id, 'prod')

Function 2: lookup(var.ami\_id, "stage", "ami-xyz")

- **Function 1**

5. Can you run scripts inside a provisioner block without being attached to any "real" resource?

- **Yes, by defining provisioners on a null\_resource, you can run your scripts as part of the Terraform life cycle without being attached to any "real" resource.**

6. Terraform can be used to codify the configuration for software-defined networks.

- **TRUE**

8. Is it compulsory to provide a version constraint string to pull module from the Terraform registry?

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

- **FALSE**

9. What is the primary responsibility of providers in Terraform?

- A provider is responsible for understanding API interactions with infrastructure providers.

10. Can we create multiple instances of a module from a single module block defined in your Terraform configuration?

- Yes, this feature was added to Terraform 0.13.
- Yes, use the for\_each or the count argument to create multiple instances of a module from a single module block similar to the resource block.

11. The idempotent characteristic provided by IaC tools ensures that, even if the same code is applied multiple times, the result remains the same.

- TRUE

12. What programming language will you need to use to contribute to the Terraform Open Source Project?

- Go

13. Workspace Internals

Workspaces are technically equivalent to renaming your state file. They aren't any more complex than that. Terraform wraps this simple notion with a set of protections and support for remote state. For local state, Terraform stores the workspace states in a directory called `terraform.tfstate.d`. This directory should be treated similarly to local-only `terraform.tfstate`.

15. Terraform loads variables in the following order, with later sources taking precedence over earlier ones:

- \* Environment variables
  - \* The `terraform.tfvars` file, if present.
  - \* The `terraform.tfvars.json` file, if present.
  - \* Any `*.auto.tfvars` or `*.auto.tfvars.json` files, processed in lexical order of their filenames.
  - \* Any `-var` and `-var-file` options on the command line, in the order they are provided.
- (This includes variables set by a Terraform Cloud workspace.)

16. Failure Behavior

By default, provisioners that fail will also cause the Terraform apply itself to fail. The `on_failure` setting can be used to change this. The allowed values are:

- \* **continue** – Ignore the error and continue with creation or destruction.
- \* **fail** – Raise an error and stop applying (the default behavior). If this is a creation provisioner, taint the resource.

17. The `terraform state list` command is used to list resources within a Terraform state.

18. Does the `terraform taint` command modify infrastructure as soon as you run it?

**No, this command will not modify infrastructure, but does modify the state file in order to mark a resource as tainted and the next apply will implement this change.**

19. Select all the correct statements about providers below.

- **Provider configurations can be defined only in a root Terraform module.**

- **A child module automatically inherits default (un-aliased) provider configurations from its parent.**
- Each module must declare its own provider requirements so that Terraform can ensure that there is a single version of the provider that is compatible with all modules in the configuration.

**20.** If a backend supports, state locking happens automatically on all write operations on a state.

- **TRUE**

State Locking

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.

**21.** State is a necessary requirement for Terraform to function?

- **TRUE**

**22.** Read the statements given carefully and choose the correct options that define the purpose of the Terraform state.

- Terraform uses the state to track metadata, such as resource dependencies.
- Terraform uses the state to map configuration to resources in the real world.
- Terraform stores a cache of the attribute values for all resources in the state for performance improvement.

**24.** Declaring a variable without any value or type associated with it will cause an error at run time as the variable is not defined.

variable "variable\_name" {}

- **FALSE**

**25.** State Backups

All terraform state subcommands that modify the state write backup files. The path of these backup file can be controlled with -backup.

**26.** Terraform can manage includes low-level components such as compute instances, storage, and networking, as well as high-level components such as DNS entries, SaaS features, etc.

- **TRUE**

**27.** Terraform can manage includes low-level components such as compute instances, storage, and networking, as well as high-level components such as DNS entries, SaaS features, etc.

- **TRUE**

**28.** During which phase Terraform reads from data sources?

- **terraform plan**

**29.** What are all connection types supported by remote-exec provisioner to invoke a script on a remote resource after creating it? Choose TWO correct answers.

- ssh
- winrm

**30.** To manage manually created infrastructure resources with Terraform, you must first write a resource block for it in your configuration, establishing the name by which it will be known to Terraform.

- TRUE

**31.** For local state, Terraform stores the workspace states in a directory called **terraform.tfstate.d**.

**34.** When using remote state, state is only ever held in memory when used by Terraform.

- True

**35.** Provisioners should only be used as a last resort.

- TRUE

**36.** 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.

**39.** What does terraform plan command do?

- **Creates an execution plan and then determines what actions are necessary to achieve the desired state by evaluating the difference between the configuration file and state file.**

**40.** The **terraform state show** command is used to show the attributes of a single resource in the Terraform state.

**41. terraform plan -destroy**

- Plan command with -destroy option generates a plan to destroy all the known resources.

**terraform destroy**

- This generates a plan to destroy all the known resources and ask for confirmation to go further to destroy resources.

**42.** If a resource successfully creates but fails during provisioning, Terraform will error and mark the resource as “tainted”. A resource that is tainted has been physically created, but can’t be considered safe to use since provisioning failed.

**43.** Valid arguments to manage several similar objects, such as a fixed pool of compute instances:

- count
- for\_each

**44.** Syntax for referencing a registry module:

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

**45.** Purpose of the local-exec provisioner in terraform:

- to execute one or more commands on the machine running Terraform
- to invoke a local executable

The local-exec provisioner invokes a local executable after a resource is created. This invokes a process on the machine running Terraform, **not on the resource**.

**46.** If your state file is too big and you want to list the resources from your state:

- **terraform state list**

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

**47.** Which command will launch the interactive console for terraform interpolations?

- `terraform console` ([Usage: terraform console \[options\] \[dir\]](#))

This 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.

**48.** The **terraform workspace delete** command is used to delete an existing workspace.

- Usage: `terraform workspace delete [NAME]`

This command will delete the specified workspace.

To delete a workspace, it must already exist, it must have an empty state, and it must not be your current workspace. If the workspace state is not empty, Terraform will not allow you to delete it unless the `-force` flag is specified.

If you delete a workspace with a non-empty state (via `-force`), then resources may become "dangling". These are resources that physically exist but that Terraform can no longer manage. This is sometimes preferred: you want Terraform to stop managing resources so they can be managed some other way. Most of the time, however, this is not intended and so Terraform protects you from getting into this situation.

The command-line flags are all optional. The only supported flag is:

`-force` – Delete the workspace even if its state is not empty. Defaults to false.

**49.** Some of the features of the Terraform state:

- mapping configuration to real-world resources
- determining the correct order to destroy resources

- increased performance

**50.**Is defining the module version argument mandatory while pulling code from Terraform Registry?

- FALSE

**51.**Multiple provider blocks can exist if a Terraform configuration is composed of multiple providers, which is a common situation.

**52.**Different benefits of backends in terraform:

- Working in Team
- Remote Operations
- Keeping sensitive information off disk

**53.**After creating an EC2 instance, Jacob wants to automatically install certain software packages like Jboss inside that EC2 instance. What is the way to achieve it?

- Make use of Remote provisioner

The remote-exec provisioner invokes a script on a remote resource after it is created.

**54.**Following is not a valid Terraform string function:

- tostring

Below are **available string function** in Terraform?

- chomp
- format
- formatlist
- indent
- join
- lower
- regex
- regexall
- replace
- split
- strrev
- substr
- title
- trim
- trimprefix
- trimsuffix
- trimspace
- upper

**55.** All features which are exclusive to Terraform Enterprise:

- Clustering
- Audit Logs
- SAML/SSO

**56.** Operating systems which are supported for a clustered Terraform Enterprise:

- Amazon Linux
- CentOS
- Red Hat

**57.** Valid syntax for specifying the required terraform version in terraform

- **terraform { required\_version = "> 0.12.00" }**

**58.** What happens when a terraform plan is executed?

- creates an execution plan and determines what changes are required to achieve the desired state in the configuration files.

**60.** **terraform validate** command will check and report errors within modules, attribute names, and value types to make sure they are syntactically valid and internally consistent

**61.** When Terraform needs to be installed in a location where it does not have internet access to download the installer and upgrades, the installation is generally known as to be **air-gapped**.

Run the Installer – Airgapped

If the instance cannot reach the Internet, follow these steps to begin an Airgapped installation.

»Prepare the Instance

Airgap installations require Docker to be pre-installed. Double-check that your instance has a supported version of Docker (see Pre-Install Checklist: Software Requirements for details).

Download the .airgap file using the information given to you in your setup email and place that file somewhere on the the instance. If you use are using wget to download the file, be sure to use `wget -content-disposition ""` so the downloaded file gets the correct extension. The url generated for the .airgap file is only valid for a short time, so you may wish to download the file and upload it to your own artifacts repository.

**62.** The purpose of command "terraform plan -refresh=true":

- Update the state prior to checking for differences

By default, plan requires no flags and looks in the current directory for the configuration and state file to refresh.

The command-line flags are all optional. The list of available flags are:



- compact-warnings – If Terraform produces any warnings that are not accompanied by errors, show them in a more compact form that includes only the summary messages.
- destroy – If set, generates a plan to destroy all the known resources.
- detailed-exitcode – Return a detailed exit code when the command exits. When provided, this argument changes the exit codes and their meanings to provide more granular information about what the resulting plan contains:
  - 0 = Succeeded with empty diff (no changes)
  - 1 = Error
  - 2 = Succeeded with non-empty diff (changes present)
- input=true – Ask for input for variables if not directly set.
- lock=true – Lock the state file when locking is supported.
- lock-timeout=0s – Duration to retry a state lock.
- no-color – Disables output with coloring.
- out=path – The path to save the generated execution plan. This plan can then be used with terraform apply to be certain that only the changes shown in this plan are applied. Read the warning on saved plans below.
- parallelism=n – Limit the number of concurrent operation as Terraform walks the graph. Defaults to 10.
- refresh=true – Update the state prior to checking for differences.
- state=path – Path to the state file. Defaults to "terraform.tfstate". Ignored when remote state is used.
- target=resource – A Resource Address to target. This flag can be used multiple times. See below for more information.
- var 'foo=bar' – Set a variable in the Terraform configuration. This flag can be set multiple times. Variable values are interpreted as HCL, so list and map values can be specified via this flag.
- var-file=foo – Set variables in the Terraform configuration from a variable file. If a terraform.tfvars or any .auto.tfvars files are present in the current directory, they will be automatically loaded. terraform.tfvars is loaded first and the .auto.tfvars files after in alphabetical order. Any files specified by -var-file override any values set automatically from files in the working directory. This flag can be used multiple times.

**63.** lookup retrieves the value of a single element from a map, given its key. If the given key does not exist, the given default value is returned instead.

**lookup(map, key, default)**

```
> lookup({a="ay", b="bee"}, "a", "what?")
```

```
ay
```

```
> lookup({a="ay", b="bee"}, "c", "what?")
```

```
what?
```

---

## Practice Set #4

1. The remote backend stores Terraform state and **may be used to run operations in Terraform Cloud**. When using full remote operations, operations like terraform plan or terraform apply can be executed in Terraform Cloud's run environment, with log output streaming to the local terminal.
4. Which of the following best describes a Terraform provider?
  - a plugin that Terraform uses to translate the API interactions with the service or provider
6. Which of the following are correct statements about the private module registry?
  - It includes support for module versioning, a searchable and filterable list of available modules, and a configuration designer to help you build new workspaces faster
  - Terraform Cloud's private module registry helps you share Terraform modules across your organization.
7. What is the purpose of command "terraform plan -destroy"?
  - Generates a plan to destroy all the known resources
8. **Only constants are allowed inside the terraform block?**
  - **TRUE**
9. What is the command that shows the attributes of a single resource in the state file?
  - terraform state show 'resource name'
10. When you run terraform init command, all the providers are installed in the current working directory.
  - TRUE
11. Which of the following is the mandatory argument for the module?
  - Source
12. Which of the following command is used to generate a visual representation of either a configuration or execution plan.
  - Terraform graph
13. What are the different types of dependencies supported in terraform? (Select 02)
  - Implicit
  - Explicit

**14.** Bob is new to terraform. He is creating an EC2 instance. Matthew wants to IP address of the instance to show in output automatically once the EC2 instance is deployed? Which resource in terraform can he use?

- output

**16.** # Additional provider configuration for west coast region; resources can  
# reference this as `aws.west`.

**17.** Which of the following are correct statement about "terraform force-unlock" command ?

- This command can't remove the lock on local state file
- This command removes the lock on the state for the current configuration.
- This command can be used to manually unlock the state for the defined configuration

**18.** What is default Limit of number of concurrent operation in terraform while executing terraform apply command (-parallelism=n)

- 10

**23.** By default, only verified modules are shown in terraform search results.

- TRUE

**24.** Which of the following are supported backend type in terraform?

- Standard
- Enhanced

**26.** The following command can be used to automatically obtain and save an API token for Terraform Cloud, Terraform Enterprise:

- **Terraform login**

**27.** What does this symbol version = "~> 1.0" mean when defining versions?

- Any version more than 1.0 and less than 2.0

**28.** Which of the following is a valid statement about terraform untaint command?

- The terraform untaint command manually unmarks a Terraform-managed resource as tainted, restoring it as the primary instance in the state.
- This command will not modify infrastructure but does modify the state file in order to unmark a resource as tainted.
- This reverses either a manual terraform taint or the result of provisioners failing on a resource.

**30.** Which of the following are valid scenarios when terraform init command must be called?

- On any new environment that configures a backend

- On removing backend configuration completely
- On any change of the backend configuration (including type of backend)

**31.** Terraform offering support SAML/SSO feature - in **Terraform Enterprise – Self Hosted**

**33.** What are the meta-arguments that are defined by terraform itself and available for all provider blocks?

- version
- alias

**34.** Sentinel (Policy as Code) policies are checked when a run is performed, after the terraform plan but before it can be confirmed or the terraform apply is executed.

- TRUE

**35.** In regards to Terraform state file, select all the statements below which are correct:

- Storing state remotely can provide better security
- Terraform state can contain sensitive data, therefore the state file should be protected from unauthorized access
- Terraform Cloud always encrypts state at rest

**36.** При конфігуруванні віддаленого бекенду в Терраформі може бути гарною ідеєю навмисно опустити деякі з необхідних аргументів, щоб гарантувати, що секрети та інші відповідні дані не будуть ненавмисно передані іншим особам. Якими способами можна додати решту конфігурації до Terraform, щоб вона могла ініціалізуватись і взаємодіяти з внутрішньою частиною?

- interactively on the command line
- command-line key/value pairs
- use the -backend-config=PATH to specify a separate config file

**39.** Which of the following are correct statements about terraform workspace?

- default workspace can not ever deleted
- To create a new workspace and switch to it, you can use terraform workspace new
- Terraform starts with a single workspace named "default".

**40.** The Terraform language supports a number of different syntaxes for comments:

- /\*. \*/
- //
- #

**45.** How do you configure Multiple Provider Instances in Terraform?

- Alias

**46.** You want to create a parallel, distinct copy of a set of infrastructure in order to test a set of changes before modifying the main production infrastructure. How do you achieve that?

- terraform workspace

**48.** What is the simple way for Terraform to read and write secrets from HashiCorp Vault?

- Vault provider

**49.** Which feature allows multiple state files for a single configuration file depending upon the environment?

- Terraform Workspace

**51.** Which of the following is the correct syntax of remote backend?

- terraform { backend "s3" { bucket = "mybucket" key = "path/to/my/key" region = "us-east-1" } }

**52.** David has joined a Security Architect in Enterprise Corp. He has mandated that all the Terraform configuration that creates an S3 bucket must have encryption feature enabled. What is the best way to achieve it?

#### Sentinel and Policy as Code

Sentinel fully embraces policy as code in a number of ways:

Language. All Sentinel policies are written using the Sentinel language. This language is made to inputted directly to text files. As an additional benefit, all Sentinel-enabled applications share the same policy language.

Development. Sentinel provides a CLI for development and testing. This local CLI can be used to verify policies before deploying them to a system.

Testing. Sentinel provides a test framework designed specifically for automation. This allows developers and CI systems to further verify policies.

**54.terraform import aws\_instance.myec2 i-1212345678**

Terraform is able to import existing infrastructure. This allows you take resources you've created by some other means and bring it under Terraform management.

This is a great way to slowly transition infrastructure to Terraform, or to be able to be confident that you can use Terraform in the future if it potentially doesn't support every feature you need today.

**55.** The core Terraform workflow has three steps:

- Write – Author infrastructure as code.
- Plan – Preview changes before applying.
- Apply – Provision reproducible infrastructure.

**58.** Terraform-specific settings and behaviors are declared in which configuration block type?

- **terraform { # ... }**

Terraform-specific settings are gathered together into terraform blocks: Each terraform block can contain a number of settings related to Terraform's behavior. Within a terraform block, only constant values can be used; arguments may not refer to named objects such as resources, input variables, etc, and may not use any of the Terraform language built-in functions.

**59.** Which of the following is the right file where the local backend saves a backup state as a file?

- terraform.tfstate.backup

**60.** Which of the following are correct statements about terraform state?

- When working with Terraform in a team, use of a local file makes Terraform usage complicated
- Terraform supports storing state in Terraform Cloud, HashiCorp Consul, Amazon S3, Alibaba Cloud OSS, and more.
- With remote state, Terraform writes the state data to a remote data store, which can then be shared between all members of a team.

**61.** The current implementation of Terraform import can only import resources into the state. It does not generate configuration

- TRUE

**65.** Provider dependencies are created in several different ways. Select the valid provider dependencies from the following list:

- Existence of any provider plugins found locally in the working directory.
- Existence of any resource instance belonging to a particular provider in the current state.
- Use of any resource belonging to a particular provider in a resource or data block in the configuration.

**66.** Which of the below command can be used to manually write state in terraform?

- terraform state push

---

## **Pactice Set #5**

**1.** A local value assigns a name to an expression, allowing it to be used multiple times within a module without repeating it.

Comparing modules to functions in a traditional programming language: if input variables are analogous to function arguments and outputs values are analogous to function return values, then local values are comparable to a function's local temporary symbols.

Note: For brevity, local values are often referred to as just “locals” when the meaning is clear from context.

»Declaring a Local Value

A set of related local values can be declared together in a single locals block:

```
locals {  
  service_name = "forum"  
  owner = "Community Team"  
}
```

**10.** While Terraform is generally written using the HashiCorp Configuration Language (HCL), what another syntax can Terraform be expressed in?

- JSON

**12.** Donald is writing a module and within the module, there are multiple places where she has defined the same conditional expression. What is a better approach to dealing with this? (Select One)

- local Values

**13.** Terraform plan validates the overall syntax of terraform code and will error if aspects like an undefined variable, missing arguments are part of the code?

- TRUE

**15.** The **local backend** stores state on the local filesystem, locks the state using system APIs, and performs operations locally.

**21.** Terraform performs a refresh, unless explicitly disabled, and then determines what actions are necessary to achieve the desired state specified in the configuration files.

- TRUE

**23.** What is true about **terraform refresh** command? (Select 02)

- The command refresh does not modify infrastructure, but does modify the state file.
- If the state is changed, this may cause changes to occur during the next plan or apply

**30.** Which of the following are correct statements about provisioners? (Select 03)

- Remote-exec can be used to run a configuration management tool, bootstrap into a cluster, etc.
- The local-exec provisioner executing command locally on your machine running Terraform.
- We use local-exec provisioner to do something on our local machine without needing any external URL

**34.** HashiCorp offers multiple versions of Terraform, including Terraform open-source, Terraform Cloud, and Terraform Enterprise. Which of the following Terraform features are only available in the Enterprise edition? (Select 03)

- SAML/SSO
- Audit Logs
- Clustering

35. Select the answer below that completes the following statement: Terraform Cloud can be managed from the CLI but requires **an API token**

36. Which of the following are valid statement for "**terraform refresh**" command ?

- This does not modify infrastructure, but does modify the state file
- The terraform refresh command is used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure.
- If the state is changed, this may cause changes to occur during the next plan or apply.
- This can be used to detect any drift from the last-known state, and to update the state file.

40. Which of the following are **correct statements about terraform provider**? (Select 03)

- Terraform is used to create, manage, and update infrastructure resources such as physical machines, VMs, network switches, containers, and more. Terraform Cloud, DNSimple, Cloudflare)
- A provider is not responsible for understanding API interactions and exposing resources.
- Providers generally are an IaaS (e.g. Alibaba Cloud, AWS, GCP, Microsoft Azure, OpenStack), PaaS (e.g. Heroku), or SaaS services
- Almost any infrastructure type can be represented as a resource in Terraform.

41. The behavior of any terraform destroy command can be previewed at any time with an equivalent terraform apply -destroy command.

- **FALSE**

43. **terraform refresh** will update the state file?

- TRUE

**This does not modify infrastructure but does modify the state file!!!**

48. The AWS provider offers a flexible means of providing credentials for authentication. Which of the following methods are supported authentication methods? (select 03)

- Environment variables
- CodeBuild, ECS, and EKS Roles
- Static credentials

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

51. Backends are responsible for supporting state locking if possible. **Not all backend types support state locking.** In the list of supported backend types we explicitly note whether locking is supported.

- FALSE



**53.** When you are working with the workspaces how do you access the current workspace in the configuration files?

- `${terraform.workspace}`

**55.** If terraform crashes where should you see the logs?

- **crash.log**

**60.** Which of the following is the correct syntax for remote backend file ?

- `terraform { backend "remote" { hostname = "app.terraform.io" organization = "company" workspaces { name = "my-app-prod" } } }`

**64.** Expressions in provisioner blocks cannot refer to their parent resource by name

- TRUE

**65.** What is correct syntax of destroy provisioner ?

- `whenprovisioner "remote-exec" { when = "destroy" # <...snip...> }`

If **when = destroy** is specified, the provisioner will run when the resource it is defined within is destroyed.

---

## Practice Set #6

**11.** The prefix `-/+` means that Terraform will destroy and recreate the resource, rather than updating it in-place. Some attributes and resources can be updated in-place and are shown with the `~` prefix.

**14.** Which of the following are supported Standard Backends in Terraform? (Select 03)

- Artifactory
- Terraform Enterprise
- Consul

List of supported Standard Backends in Terraform • artifactory • azurerm • consul • cos • etcd • etcdv3 • gcs • http • manta • oss • pg • s3 • swift • terraform enterprise

**27.** Which of the following terraform offering support Sentinel (Policy as Code)? (Select 02)

- Terraform Enterprise – Self Hosted
- Terraform Cloud – Team & Governance

**32.** Workspaces provide identical functionality in the open-source, Terraform Cloud, and Enterprise versions of Terraform.

- FALSE

**38.** A user has created a module called "my\_test\_module" and committed it to GitHub. Over time, several commits have been made with updates to the module, each tagged in GitHub with an incremental version number. Which of the following lines would be required in a module configuration block in terraform to select tagged version v1.0.4?

- `source = "git::https://example.com/my_test_module.git?ref=v1.0.4"`

**46.** A user creates three workspaces from the command line – prod, dev, and test. Which of the following commands will the user run to switch to the dev workspace?

- `terraform workspace select dev`

**55.** Which of the following are correct CLI configuration files in terraform ? (Select 02)

- `.terraformrc`
- `terraform.rc`

**56.** When writing the Terraform code, HashiCorp recommends that you use how many spaces between each nesting level?

- 2

**57.** How do you access output variables from the modules?

- `module..`

**60.** By default, terraform fmt scans all the directory and subdirectories for configuration files.

- FALSE

**64.** In order to reduce the time it takes to provision resources, Terraform uses parallelism. By default, how many resources will Terraform provision concurrently?

- 10

**67.** Which of the following are the benefit of Sentinel(policy as code)? (Select 04)

- Automation
- Codification
- **Autoscaling - NO!!!!**
- Version Control
- Sandboxing

**69.** What are the advantages of Terraform as Infra as Code technology (Select 03)

- Platform Agnostic
- State Management
- Operator Confidence

**70.** Which of the following snippet you will choose if you want to set both a lower and upper bound on versions for each provider?

- `terraform { required_providers { aws = ~> "2.7.0" } }`

---

## Practice Set #7

### 1. Question

What feature of Terraform Cloud and/or Terraform Enterprise can you publish and maintain a set of custom modules which can be used within your organization?

- private module registry

Correct

You can use modules from a private registry, like the one provided by Terraform Cloud. Private registry modules have source strings of the form `///`. This is the same format as the public registry, but with an added hostname prefix.

6. When using providers that require the retrieval of data, such as the HashiCorp Vault provider, in what phase does Terraform actually retrieve the data required, assuming you are following the standard workflow of write, plan, and apply?

- terraform plan

14. Terraform Enterprise offers the ability to use Terraform to deploy infrastructure in your local on-premises datacenter as well as a public cloud platform, such as AWS, Azure, or GCP.

- TRUE

26. A “backend” in Terraform determines how state is loaded and how an operation such as apply is executed. Which of the following is not a supported backend type?

- github

GitHub is not a supported backend type. Check out the supported backends using the link below. Remember there is the “local” backend and then there are remote backends that store state elsewhere. Remote backends (and locking) are needed when more than one person is interacting with the same state file.

The **terraform state** command and subcommands can be used to modify the current state, such as removing items.

Terraform Cloud supports the following VCS providers as of January 2022:

- GitHub
- GitHub.com (OAuth)
- GitHub Enterprise
- GitLab.com
- GitLab EE and CE
- Bitbucket Cloud
- Bitbucket Server
- Azure DevOps Server
- Azure DevOps Services

By default, terraform init downloads plugins into a subdirectory of the working directory, `.terraform/providers` so that each working directory is self-contained.

---

## Practice Set #8

### 6. Question

Terraform Cloud can be managed from the CLI but requires **an API token**?

8. The list function is deprecated. From Terraform v0.12, the Terraform language has built-in syntax for creating lists using the [ and ] delimiters.

10. The aws\_instance will be created first, and then aws\_eip will be created second due to the aws\_eip's resource dependency of the aws\_instance id

17. Which Terraform command will force a resource to be destroyed and recreated even if there are no configuration changes that would require it?

- **terraform apply -replace=**

26. A workspace can only be configured to a **single VCS repo**, however, multiple workspaces can use the same repo, if needed. A good explanation of how to configure your code repositories can be found [here](#).

### 30. Question

HashiCorp offers multiple versions of Terraform, including Terraform open-source, Terraform Cloud, and Terraform Enterprise. Which of the following Terraform feature is only available in the Enterprise edition? (select one)

- Clustering

While there are a ton of features that are available to open source and Cloud users, there are still a few features that are part of the Enterprise offering which is geared towards enterprise requirements. With the introduction of Terraform Cloud for Business, almost all features are now available for a hosted Terraform deployment. To see what specific features are part of Terraform Cloud and Terraform Enterprise, check out [this link](#).

Information about Answers:

- Private Network Connectivity is available for Terraform Enterprise, since it is installed locally in your data center or cloud environment. However, it is also available in Terraform Cloud for Business since you can use a self-hosted agent to communicate with local/private hosts in your environment.
- SAML/SSO is available in BOTH Terraform Enterprise and Terraform Cloud for Business, therefore it is NOT exclusive to just Terraform Enterprise.
- Private Module Registry is available in every version of Terraform except for Open-Source. Therefore it is NOT exclusive to Terraform Enterprise

- Locally hosted installation is available for Terraform Enterprise, but technically you install Terraform open-source on your local machine or server, therefore it is NOT exclusive to Terraform Enterprise.
- Clustering is the ONLY answer that is available ONLY in Terraform Enterprise. You can't cluster Terraform open-source, and the other options are hosted solutions. This makes clustering the only correct answer. Note that Clustering was available for Enterprise for a while, then HashiCorp removed it. As of January 15, 2021, it's back and you can read more about it at this link.

### 36. Question

Which of the following represents a feature of Terraform Cloud that is NOT free to customers?

- team management and governance

37. The special **terraform configuration block** type is used to configure some behaviors of Terraform itself, such as requiring a minimum Terraform version to apply your configuration.

### 43. Question

What Terraform command can be used to inspect the current state file?

- terraform show

### 45. Question

After executing a terraform plan, you notice that a resource has a tilde (~) next to it. What does this mean?

- the resource will be updated in place

53. A solution to create and publish Terraform modules that only its engineers and architects can use - **private module registry!**

### 58. Question

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

- multiple provider blocks using an alias

60. IaC makes changes **idempotent**, **consistent**, **repeatable**, and **predictable**.

61. The resource above will be created in the default region of us-east-1, since the resource does not signify an alternative provider configuration. If the resource needs to be created in one of the other declared regions, it should have looked like this, where "aws" signifies the provider name and "west" signifies the alias name as such .:

```
resource "aws_instance" "vault" {  
  provider = aws.west
```

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

#### 64. Question

Terraform Cloud Agents are a feature that allows Terraform Cloud to communicate with private infrastructure, such as VMware hosts running on-premises. Which version of Terraform Cloud supports this feature?

- Terraform Cloud for Business

---

### Practice Set #9

2. If there is **no state file** associated with a Terraform configuration file, a terraform apply will create the resources defined in the configuration file. This is a normal workflow during the first terraform apply that is executed against a configuration file.

- Terraform will scan the VMware infrastructure, create a new state file, and compare the state to the configuration file to determine what resources should be created.

3. Best describes a "data source":

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

4. When using modules installed from a module registry, HashiCorp recommends explicitly constraining the acceptable version numbers to avoid unexpected or unwanted changes. The version argument accepts a version constraint string. Terraform will use the newest installed version of the module that meets the constraint; if no acceptable versions are installed, it will download the newest version that meets the constraint.

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

- **Terraform would use the existing module already downloaded**

8. Which of the following is true about this new workspace? (select four)

- changes to this workspace won't impact other workspaces
- you can use a different variables file for this workspace if needed
- it has its own state file
- it uses the same Terraform code in the current directory

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

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

- enables the deployment of resources to multiple platforms, such as public cloud, private cloud, or other SaaS, PaaS, or IaaS services
- enables a plugin architecture that allows Terraform to be extensible without having to update Terraform core

**13.** 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 show aws_instance.frontend`

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` 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

**14.** 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.

**15.** 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.

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

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.

**16.** 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 😊

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

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

**18.** 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.

**19.** how would you refer to the value of ip for the dev environment if you are using a for\_each argument?

- each.value.ip

**23.** Terraform Cloud (all versions) and Terraform Enterprise offer the use of a private module registry. Among all the features you get with Terraform OSS, all other versions get these features, at a minimum:

- State management
- Remote operations
- Private module registry
- Community support

WRONG ANSWER:

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

**24.** You want to use Terraform to deploy resources across your on-premises infrastructure and a public cloud provider. However, your internal security policies require that you have full control over both the operating system and deployment of Terraform binaries. What versions of Terraform can you use for this? (select two)

- Terraform Enterprise
- Terraform OSS/CLI



Terraform OSS and Terraform Enterprise are versions of Terraform that can be installed locally on your own servers, therefore giving you the ability to manage both the Terraform binary and the underlying operating system where Terraform runs.

**WRONG ANSWERS:**

Although **Terraform Cloud for Business** does offer Cloud Agents that could be used to provision resources on your local infrastructure on-premises, it is a hosted solution and you would NOT have full control over the operating system that runs the Terraform platform.

**Terraform Cloud (free)** does not meet either of these use cases since you can't deploy to on-premises nor can you manage the underlying operating system since it's a hosted service.

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

- TRUE

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

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

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

**36.** The **-upgrade** 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.

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

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

```

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

```

- data\_processing

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

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

- use environment variables
- integrated services, such as AWS IAM or Azure Managed Service Identity
- directly in the provider block by hardcoding or using a variable

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

- remote runs
- VCS connection
- private module registry

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

```

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

```

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

1. 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.

## 5. Question

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

- it always matches the infrastructure deployed with Terraform
- 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.

## 6. terraform apply -replace="aws\_instance.database"

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.

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

- using terraform import will bring the imported resource under Terraform management and add the new resource to the state file
- the resource address (example: aws\_instance.web) and resource ID (example: i-abdcef12345) must be provided when importing a resource
- you must update your Terraform configuration for the imported resource before attempting to import the resource

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

- caches the source code locally for referenced modules
- downloads required providers used in your configuration file
- prepares the working directory for use with Terraform

## 18. 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.

- FALSE

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

- terraform destroy
- terraform apply -destroy

## 34. In order to use the terraform console command, the CLI must be able to lock state to prevent changes.

- TRUE

## 58. Question

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

- terraform workspace show

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

WRONG ANSWERS:

**terraform workspace list** will list out all available workspaces

**terraform workspace select** will instructs Terraform what workspace to change to and use

**terraform workspace** is just a container that requires additional subcommands

63. 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

## 65. Question

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

- they allow anybody to write a provider and publish it to the registry
- providers can be written and maintained by an outside organization, such as AWS, F5, or Microsoft
- some providers are community-supported
- some providers are maintained by HashiCorp

---

## Practice Set #11

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

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

**8.** To upgrade an existing provider that you have already downloaded, you need to run **terraform init -upgrade**.

**13.** When a module is located at **hashicorp/**, **Terraform download it from the official Terraform public module 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

**20.** Which of the features below is available in the free version of Terraform Cloud? (select three)

- Private Module Registry
- Remote Operations
- State Management

Incorrect

**Single Sign-On** is a feature of **Terraform Enterprise** and **Terraform Cloud for Business**. It is NOT available in Terraform Cloud (free tier)

**21.** The **local-exec provisioner** invokes a local executable after a resource is created. **This invokes a process on the machine running Terraform, not on the resource.**

**24.** WRONG ANSWER:

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

**38.** The terraform refresh command is used to reconcile the state Terraform knows about (via its **state file**) with the real-world infrastructure.

**40.** All **plan**, **apply** and **destroy** commands **run refresh first**, prior to any other work unless explicitly disabled. Detect drift with terraform plan, which reconciles desired configuration with real-

world state and tells you what Terraform will do during terraform apply in a similar way reconciled state will tell Terraform what to destroy during terraform destroy.

Terraform provides a flag to avoid refreshing the state during plan/apply/destroy, **-refresh=false**

**41.**It is true, **not all providers and resources support Terraform import.**

---

**THE END**