

HASHICORP CERTIFIED TERRAFORM ASSOCIATE

Practice Questions - Intermediate

Question 1:

Which Terraform command will force a marked resource to be destroyed and recreated on the next apply?

- A. terraform destroy
- B. terraform refresh
- C. terraform taint
- D. terraform fmt

Correct: C

Explanation

The `terraform taint` command manually marks a Terraform-managed resource as tainted, forcing it to be destroyed and recreated on the next apply. 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. The next `terraform apply` will implement this change.

Question 2:

In regards to deploying resources in multi-cloud environments, what are some of the benefits of using Terraform rather than a provider's native tooling? (select three)

- A. Terraform simplifies management and orchestration, helping operators build large-scale, multi-cloud infrastructure
- B. Terraform can help businesses deploy applications on multiple clouds and on-premises infrastructure
- C. Terraform can manage cross-cloud dependencies
- D. Terraform is not cloud-agnostic and can be used to deploy resources across a single public cloud



Correct: A, B, C

Explanation

Terraform is a cloud-agnostic tool, and therefore isn't limited to a single cloud provider, such as AWS CloudFormation or Azure Resource Manager. Terraform supports all of the major cloud providers and allows IT organizations to focus on learning a single tool for deploying its infrastructure, regardless of what platform it's being deployed on

Question 3:

What happens when a terraform apply command is executed?

- A. Reconciles the state Terraform knows about with the real-world infrastructure
- B. The backend is initialized and the working directory is prepped
- C. Creates the execution plan for the deployment of resources
- D. Applies the changes required in the target infrastructure in order to reach the desired configuration

Correct: D

Explanation

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

Question 4:

What does the command terraform fmt do?

- A. deletes the existing configuration file
- B. rewrite Terraform configuration files to a canonical format and style
- C. updates the font of the configuration file to the official font supported by HashiCorp
- D. formats the state file in order to ensure the latest state of resources can be obtained

Correct: B



Explanation

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

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

Question 5:

Select two answers to complete the following sentence:

Before a new provider can be used, it must be ____ and ____.

- A. approved by HashiCorp
- B. uploaded to source control
- C. declared in the configuration
- D. initialized

Correct: D,C

Explanation

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

Question 6:

Which flag would be used within a Terraform configuration block to identify the specific version of a provider required?

- A. required_versions
- B. required-version
- C. required-provider
- D. required_providers



Correct: D

Explanation

For production use, you should constrain the acceptable provider versions via configuration file to ensure that new versions with breaking changes will not be automatically installed by terraform init in the future. When terraform init is run without provider version constraints, it prints a suggested version constraint string for each provider.

For example:

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

Question 7:

In the example below, where is the value of the DNS record's IP address originating from?

```
1. resource "aws_route53_record" "www" {  
2.   zone_id = aws_route53_zone.primary.zone_id  
3.   name     = "www.helloworld.com"  
4.   type     = "A"  
5.   ttl      = "300"  
6.   records = [module.web_server.instance_ip_addr]  
7. }
```

- A. value of the web_server parameter from the variables.tf file
- B. the regular expression named module.web_server
- C. the output of a module named web_server
- D. by querying the AWS EC2 API to retrieve the IP address

Correct: C



Explanation

In a parent module, outputs of child modules are available in expressions as `module.<MODULE NAME>.<OUTPUT NAME>`. For example, if a child module named `web_server` declared an output named `instance_ip_addr`, you could access that value as `module.web_server.instance_ip_addr`.

Question 8:

Given the Terraform configuration below, in which order will the resources be created?

```
1. resource "aws_instance" "web_server" {
2.     ami = "i-abdce12345"
3.     instance_type = "t2.micro"
4. }
5.
6. resource "aws_eip" "web_server_ip" {
7.     vpc = true
8.     instance = aws_instance.web_server.id
9. }
```

- A. resources will be created simultaneously
- B. aws_eip will be created first
aws_instance will be created second
- C. aws_instance will be created first
aws_eip will be created second

Correct: C

Explanation

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

Question 9:

True or False: You can migrate the Terraform backend but only if there are no resources currently being managed.

- A. True
- B. False



Correct: B. False

Explanation

If you are already using Terraform to manage infrastructure, you probably want to transfer to another backend, such as Terraform Cloud, so you can continue managing it. By migrating your Terraform state, you can hand off infrastructure without de-provisioning anything.

Question 10:

Why is it a good idea to declare the required version of a provider in a Terraform configuration file?

```
1 terraform {  
2   required_providers {  
3     aws = "~> 1.0"  
4   }  
5 }
```

- A. to remove older versions of the provider
- B. to match the version number of your application being deployed via Terraform
- C. providers are released on a separate schedule from Terraform itself; therefore a newer version could introduce breaking changes
- D. to ensure that the provider version matches the version of Terraform you are using

Correct: C

Explanation

Providers are plugins released on a separate rhythm from Terraform itself, and so they have their own version numbers. For production use, you should constrain the acceptable provider version via configuration. This helps to ensure that new versions with potentially breaking changes will not be automatically installed by terraform init in the future.



Question 11:

You want to use terraform import to start managing infrastructure that was not originally provisioned through infrastructure as code. Before you can import the resource's current state, what must you do in order to prepare to manage these resources using Terraform?

- A. shut down or stop using the resources being imported so no changes are inadvertently missed
- B. update the configuration file to include the new resources
- C. modify the Terraform state file to add the new resources
- D. run terraform refresh to ensure that the state file has the latest information for existing resources.

Correct: B

Explanation

The current implementation of Terraform import can only import resources into the [state](#). It does not generate a configuration. Because of this, and prior to running `terraform import`, it is necessary to manually write a `resource` configuration block for the resource to which the imported object will be mapped.

First, add the resources to the configuration file:

```
resource "aws_instance" "example" {  
    # ...instance configuration...  
}
```

Then run the following command:

```
$ terraform import aws_instance.example i-abcd1234
```

Question 12:

True or False: State is a requirement for Terraform to function.

- A. True
- B. False

Correct: A True



Explanation

Terraform requires some sort of database to map Terraform config to the real world. When you have a resource in your configuration, Terraform uses this map to know how that resource is represented. Therefore, to map configuration to resources in the real world, Terraform uses its own state structure.

Question 13:

What is a downside to using the Vault provider to read secrets from Vault?

- A. Terraform requires a unique auth method to work with Vault
- B. secrets are persisted to the state file and plans
- C. Terraform and Vault must be running on the same version
- D. Terraform and Vault must be running on the same physical host

Correct: B

Explanation

Interacting with Vault from Terraform causes any secrets that you read and write to be persisted in both Terraform's state file *and* in any generated plan files. For any Terraform module that reads or writes Vault secrets, these files should be treated as sensitive and protected accordingly.

Question 14:

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

- A. VCS integration
- B. roles and team management
- C. private module registry
- D. workspace management

Correct: B



Explanation

Information on the comparisons of Terraform Cloud can be [found at this link](#).

<https://www.hashicorp.com/products/terraform/pricing/>

Question 15:

You've been given requirements to create a security group for a new application. Since your organization standardizes on Terraform, you want to add this new security group with the fewest number of lines of code. What feature could you use to iterate over a list of required tcp ports to add to the new security group?

- A. dynamic backend
- B. dynamic block
- C. terraform import
- D. splat expression

Correct: B

Explanation

A `dynamic` block acts much like a `for` expression, but produces nested blocks instead of a complex typed value. It iterates over a given complex value and generates a nested block for each element of that complex value.

Question 16:

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

- A. The resource will be created
- B. Terraform can't determine how to proceed due to a problem with the state file
- C. the resource will be updated in place
- D. the resource will be destroyed and recreated



Correct: C

Explanation:

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.

Question 17:

In Terraform Enterprise, a workspace can be mapped to how many VCS repos?

- A. 1
- B. 2
- C. 5
- D. 3

Correct: A

Explanation

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](https://www.terraform.io/docs/cloud/workspaces/configurations.html).

<https://www.terraform.io/docs/cloud/workspaces/configurations.html>

Question 18:

When multiple engineers start deploying infrastructure using the same state file, what is a feature of remote state storage that is critical to ensure the state doesn't become corrupt?

- A. Encryption
- B. object storage
- C. workspaces
- D. state locking

Correct: D

Explanation

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

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

Question 19:

By default, where does Terraform store its state file?

- A. Amazon S3 bucket
- B. shared directory
- C. remotely using Terraform Cloud
- D. current working directory

Correct: D

Explanation

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

Question 20:

Which of the following actions are performed during a terraform init? (select three)

- A. initializes downloaded and/or installed providers
- B. download the declared providers which are supported by HashiCorp
- C. initializes the backend configuration
- D. provisions the declared resources in your configuration



Correct: A, B, C

Explanation

The `terraform init` command is used to initialize a working directory containing Terraform configuration files. This is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control. It is safe to run this command multiple times.

Question 21:

From the answers below, select the advantages of using Infrastructure as Code. (select four)

- A. Provide reusable modules for easy sharing and collaboration
- B. provide a codified workflow to develop customer-facing applications
- C. Safely test modifications using a "dry run" before applying any actual changes
- D. Easily integrate with application workflows (GitLab Actions, Azure DevOps, CI/CD tools)
- E. Easily change and update existing infrastructure

Correct: A, C, D, E

Explanation

Infrastructure as Code is **not** used to develop applications, but it can be used to help deploy or provision those applications to a public cloud provider or on-premises infrastructure.

All of the others are benefits to using Infrastructure as Code over the traditional way of managing infrastructure, regardless if it's public cloud or on-premises.

Question 22:

Using multi-cloud and provider-agnostic tools provides which of the following benefits? (select two)

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

- B. operations teams only need to learn and manage a single tool to manage infrastructure, regardless of where the infrastructure is deployed
- C. can be used across major cloud providers and VM hypervisors
- D. increased risk due to all infrastructure relying on a single tool for management

Correct: B, C

Explanation

Using a tool like Terraform can be advantageous for organizations deploying workloads across multiple public and private cloud environments. Operations teams only need to learn a single tool, single language, and can use the same tooling to enable a DevOps-like experience and workflows.

Question 23:

Which of the following best describes a Terraform provider?

- A. serves as a parameter for a Terraform module that allows a module to be customized
- B. describes an infrastructure object, such as a virtual network, compute instance, or other components
- C. a plugin that Terraform uses to translate the API interactions with the service or provider
- D. a container for multiple resources that are used together

Correct: C

Explanation

A provider is 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 (e.g., Terraform Cloud, DNSimple, CloudFlare).

Question 24:

What are some of the features of Terraform state? (select three)

- A. inspection of cloud resources
- B. determining the correct order to destroy resources
- C. mapping configuration to real-world resources
- D. increased performance

Correct: B, C, D

Explanation

See [this page](https://www.terraform.io/docs/state/purpose.html) on the purpose of Terraform state and the benefits it provides.

<https://www.terraform.io/docs/state/purpose.html>

Question 25:

When configuring a remote backend in Terraform, it might be a good idea to purposely omit some of the required arguments to ensure secrets and other important data aren't inadvertently shared with others. What are the ways the remaining configuration can be added to Terraform so it can initialize and communicate with the backend? (select three)

- A. directly querying HashiCorp Vault for the secrets
- B. command-line key/value pairs
- C. use the -backend-config=PATH to specify a separate config file
- D. interactively on the command line

Correct: B,C,D

Explanation

You do not need to specify every required argument in the backend configuration. Omitting certain arguments may be desirable to avoid storing secrets, such as access keys, within the main configuration. When some or all of the arguments are omitted, we call this a *partial configuration*.

With a partial configuration, the remaining configuration arguments must be provided as part of [the initialization process](#). There are several ways to supply the remaining arguments:

Interactively: Terraform will interactively ask you for the required values unless interactive input is disabled. Terraform will not prompt for optional values.

File: A configuration file may be specified via the `init` command line. To specify a file, use the `-backend-config=PATH` option when running `terraform init`. If the file contains secrets it may be kept in a secure data store, such as [Vault](#), in which case it must be downloaded to the local disk before running Terraform.

Command-line key/value pairs: Key/value pairs can be specified via the `init` command line. Note that many shells retain command-line flags in a history file, so this isn't recommended for secrets. To specify a single key/value pair, use the `-backend-config="KEY=VALUE"` option when running `terraform init`.

Question 26:

Your organization has moved to AWS and has manually deployed infrastructure using the console. Recently, a decision has been made to standardize on Terraform for all deployments moving forward.

What can you do to ensure that all existing is managed by Terraform moving forward without interruption to existing services?

- A. using terraform import, import the existing infrastructure into your Terraform state
- B. resources that are manually deployed in the AWS console cannot be imported by Terraform
- C. submit a ticket to AWS and ask them to export the state of all existing resources and use terraform import to import them into the state file
- D. delete the existing resources and recreate them using new a Terraform configuration so Terraform can manage them moving forward

Correct: A



Explanation

Terraform is able to import existing infrastructure. This allows you to 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 sure you're confident that you can use Terraform in the future if it currently doesn't support every AWS service or feature you need today.

Question 27:

Which of the following allows Terraform users to apply policy as code to enforce standardized configurations for resources being deployed via infrastructure as code?

- A. module registry
- B. functions
- C. workspaces
- D. sentinel

Correct: D

Explanation

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

Question 28:

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

Example:




```
# aws_instance.example:
resource "aws_instance" "example" {
  ami                = "ami-2757f631"
  arn                = "arn:aws:ec2:us-east-1:130490850807:instance/i-0
  associate_public_ip_address = true
  availability_zone   = "us-east-1c"
  cpu_core_count      = 1
  cpu_threads_per_core = 1
  disable_api_termination = false
  ebs_optimized        = false
  get_password_data     = false
  id                   = "i-0bbf06244e44211d1"
  instance_state       = "running"
  instance_type        = "t2.micro"
```

- A. terraform inspect
- B. terraform read
- C. terraform show
- D. terraform state

Correct: C

Explanation

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

Machine-readable output can be generated by adding the `-json` command-line flag.

Note: When using the `-json` command-line flag, any sensitive values in Terraform state will be displayed in plain text.

Question 29:

Select the answer below that completes the following statement:

Terraform Cloud can be managed from the CLI but requires _____?



- A. an API token
- B. a TOTP token
- C. a username and password
- D. authentication using MFA

Correct: A

Explanation

API and CLI access are managed with API tokens, which can be generated in the Terraform Cloud UI. Each user can generate any number of personal API tokens, which allow access with their own identity and permissions. Organizations and teams can also generate tokens for automating tasks that aren't tied to an individual user.

Question 30:

Which of the following is considered a Terraform plugin?

- A. Terraform logic
- B. Terraform provider
- C. Terraform tooling
- D. Terraform language

Correct: B

Explanation

Terraform is built on a plugin-based architecture. All providers and provisioners that are used in Terraform configurations are plugins, even the core types such as AWS and Heroku. Users of Terraform are able to write new plugins in order to support new functionality in Terraform.

Question 31:

What Terraform feature is shown in the example below?

```
1. resource "aws_security_group" "example" {
2.   name = "sg-app-web-01"
3.
4.   dynamic "ingress" {
5.     for_each = var.service_ports
6.     content {
7.       from_port = ingress.value
8.       to_port   = ingress.value
9.       protocol  = "tcp"
10.    }
11.  }
12. }
```

- A. conditional expression
- B. local values
- C. dynamic block
- D. data source

Correct: C

Explanation

You can dynamically construct repeatable nested blocks like `ingress` using a special `dynamic` block type, which is supported inside `resource`, `data`, `provider`, and `provisioner` blocks

Question 32:

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 four)



- A. SAML/SSO
- B. Sentinel
- C. Audit Logs
- D. Clustering
- E. Private Module Registry
- F. Private Network Connectivity

Correct: A, C, D, F

Explanation

While there are a ton of features that are available to open source users, many features that are part of the Enterprise offering are geared towards larger teams and enterprise functionality. To see what specific features are part of Terraform Cloud and Terraform Enterprise, [check out this link](https://www.hashicorp.com/products/terraform/pricing/).

<https://www.hashicorp.com/products/terraform/pricing/>

Question 33:

In regards to Terraform state file, select all the statements below which are correct: (select four)

- A. The Terraform state can contain sensitive data, therefore the state file should be protected from unauthorized access
- B. using the mask feature, you can instruct Terraform to mask sensitive data in the state file
- C. when using local state, the state file is stored in plain-text
- D. the state file is always encrypted at rest
- E. Terraform Cloud always encrypts state at rest
- F. storing state remotely can provide better security

Correct: A, C, E, F



Explanation

Terraform state can contain sensitive data, depending on the resources in use and your definition of "sensitive." The state contains resource IDs and all resource attributes. For resources such as databases, this may contain initial passwords.

When using local state, state is stored in plain-text JSON files.

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

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

Question 34:

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

- A. Infrastructure as Code is relatively simple to learn and write, regardless of a user's prior experience with developing code
- B. Infrastructure as Code provides configuration consistency and standardization among deployments
- C. Infrastructure as Code is easily repeatable, allowing the user to reuse code to deploy similar, yet different resources
- D. Infrastructure as Code gives the user the ability to recreate an application's infrastructure for disaster recovery scenarios
- E. Infrastructure as Code easily replaces development languages such as Go and .Net for application development
- F. Infrastructure as Code allows a user to turn a manual task into a simple, automated deployment

Correct: A, B, C, D, F

Explanation

If you are new to infrastructure as code as a concept, it is the process of managing infrastructure in a file or files rather than manually configuring resources in a user interface.

A resource in this instance is any piece of infrastructure in a given environment, such as a virtual machine, security group, network interface, etc.

At a high level, Terraform allows operators to use HCL to author files containing definitions of their desired resources on almost any provider (AWS, GCP, GitHub, Docker, etc) and automates the creation of those resources at the time of application.

Question 35:

True or False: Provisioners should only be used as a last resort.

- A. True
- B. false

Correct: A True

Explanation

Provisioners are used to execute scripts on a local or remote machine as part of resource creation or destruction. Provisioners can be used to bootstrap a resource, cleanup before destroy, run configuration management, etc. Even if the functionality you need is not available in a provider today, HashiCorp suggests that you consider `local-exec` usage as a temporary workaround and to open an issue in the relevant provider's repo to discuss adding first-class support.

Question 36:

Which of the following best describes the default local backend?

- A. The local backend is the directory where resources deployed by Terraform have direct access to in order to update their current state
- B. The local backend is where Terraform Enterprise stores logs to be processed by an log collector
- C. The local backend is how Terraform connects to public cloud services, such as AWS, Azure, or GCP



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

Correct: D

Explanation

Information on the default local backend can be [found at this link](https://www.terraform.io/docs/backends/types/local.html).

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

Example:

```
terraform {  
  backend "local" {  
    path = "relative/path/to/terraform.tfstate"  
  }  
}
```

Question 37:

In order to make a Terraform configuration file dynamic and/or reusable, static values should be converted to use what?

- A. Module
- B. input parameters
- C. regular expressions
- D. output value

Correct: B

Explanation

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



Question 38:

What happens when a terraform plan is executed?

- A. reconciles the state Terraform knows about with the real-world infrastructure
- B. applies the changes required in the target infrastructure in order to reach the desired configuration
- C. the backend is initialized and the working directory is prepped
- D. creates an execution plan and determines what changes are required to achieve the desired state in the configuration files.

Correct: D

Explanation

The `terraform plan` command is used to create an execution plan. Terraform performs a refresh, unless explicitly disabled, and then determines what actions are necessary to achieve the desired state specified in the configuration files.

After a plan has been run, it can be executed by running a `terraform apply`

Question 39:

Which of the following connection types are supported by the remote-exec provisioner? (select two)

- A. smb
- B. winrm
- C. rdp
- D. ssh

Correct: B, D

Explanation

The remote-exec provisioner invokes a script on a remote resource after it is created. The remote-exec provisioner supports both ssh and winrm type connections.

More information on remote-exec can be [found at this link](https://www.terraform.io/docs/provisioners/remote-exec.html).

<https://www.terraform.io/docs/provisioners/remote-exec.html>

Question 40:

What are some of the problems of how infrastructure was traditionally managed before Infrastructure as Code? (select three)

- A. Requests for infrastructure or hardware required a ticket, increasing the time required to deploy applications
- B. Traditional deployment methods are not able to meet the demands of the modern business where resources tend to live days to weeks, rather than months to years
- C. Traditionally managed infrastructure can't keep up with cyclic or elastic applications
- D. Pointing and clicking in a management console is a scalable approach and reduces human error as businesses are moving to a multi-cloud deployment model

Correct: A, B, C

Explanation

Businesses are making a transition where traditionally-managed infrastructure can no longer meet the demands of today's businesses. IT organizations are quickly adopting the public cloud, which is predominantly API-driven.

To meet customer demands and save costs, application teams are architecting their applications to support a much higher level of elasticity, supporting technology like containers and public cloud resources. These resources may only live for a matter of hours; therefore the traditional method of raising a ticket to request resources is no longer a viable option

Pointing and clicking in a management console is NOT scale and increases the change of human error



Question 41:

Which of the following Terraform files should be ignored by Git when committing code to a repo? (select two)

- A. terraform.tfstate
- B. variables.tf
- C. output.tf
- D. terraform.tfvars

Correct: A, D

Explanation

The .gitignore file should be configured to ignore Terraform files that either contain sensitive data or aren't required to save.

The terraform.tfstate file contains the terraform state of a specific environment and doesn't need to be preserved in a repo. The terraform.tfvars file may contain sensitive data, such as passwords or IP addresses of an environment that you may not want to share with others.

Question 42:

True or False: A list(...) may contain a number of values of the same type while an object(...) can contain a number of values of different types.

- A. True
- B. False

Correct: A True



Explanation

A *collection* type allows multiple values of *one* other type to be grouped together as a single value. This includes list, map, and set.

A *structural* type allows multiple values of *several distinct types* to be grouped together as a single value. This includes object and tuple.

Question 43:

True or False: Multiple providers can be declared within a single Terraform configuration file.

- A. True
- B. False

Correct: A True

Explanation

Multiple provider blocks can exist if a Terraform configuration is composed of multiple providers, which is a common situation. To add multiple providers in your configuration, declare the providers, and create resources associated with those providers.

Question 44:

When using parent/child modules to deploy infrastructure, how would you export a value from one module to import into another module.

For example, a module dynamically deploys an application instance or virtual machine, and you need the IP address in another module to configure a related DNS record in order to reach the newly deployed application.

- A. configure the pertinent provider's configuration with a list of possible IP addresses to use
- B. export the value using terraform export and input the value using terraform input



- C. preconfigure the IP address as a parameter in the DNS module
- D. configure an output value in the application module in order to use that value for the DNS module

Correct: D

Explanation

Output values are like the return values of a Terraform module and have several uses such as a child module using those outputs to expose a subset of its resource attributes to a parent module.

Question 45:

What is the best and easiest way for Terraform to read and write secrets from HashiCorp Vault?

- A. Vault provider
- B. API access using the AppRole auth method
- C. integration with a tool like Jenkins
- D. CLI access from the same machine running Terraform

Correct: A

Explanation

The Vault provider allows Terraform to read from, write to, and configure [Hashicorp Vault](#).

Question 46:

True or False: Workspaces provide identical functionality in the open-source, Terraform Cloud, and Enterprise versions of Terraform.



- A. True
- B. False

Correct: B False

Explanation

Workspaces, managed with the `terraform workspace` command, isn't the same thing as [Terraform Cloud's workspaces](#). Terraform Cloud workspaces act more like completely separate working directories.

[CLI workspaces](#) (OSS) are just alternate state files.

Question 47:

Select the feature below that best completes the sentence:

The following list represents the different types of _____ available in Terraform.

1. max
2. min
3. join
4. replace
5. list
6. length
7. range

- A. data sources
- B. functions
- C. backends
- D. named values

Correct: B



Explanation

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

Question 48:

What is the purpose of using the local-exec provisioner? (select two)

- A. to execute one or more commands on the machine running Terraform
- B. to invoke a local executable
- C. executes a command on the resource to invoke an update to the Terraform state
- D. ensures that the resource is only executed in the local infrastructure where Terraform is deployed

Correct: A, B

Explanation

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.

Note that even though the resource will be fully created when the provisioner is run, there is no guarantee that it will be in an operable state - for example, system services such as `sshd` may not be started yet on compute resources.

Question 49:

After running into issues with Terraform, you need to enable verbose logging to assist with troubleshooting the error. Which of the following values provides the MOST verbose logging?

- A. TRACE



- B. DEBUG
- C. WARN
- D. ERROR
- E. INFO

Correct: A

Explanation

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.

Question 50:

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

- A. data
- B. provider
- C. resource
- D. terraform

Correct: D

Explanation

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.

ALL THE BEST

