Exam : **TA-002-P**

Title : HashiCorp Certified: Terraform Associate

Vendor : HashiCorp

Version : V14.35

QUESTION NO: 1
Which of the following is not true of Terraform providers?
(A). Providers can be written by individuals
(B). Providers can be maintained by a community of users
(C). Some providers are maintained by HashiCorp
(D). Major cloud vendors and non-cloud vendors can write, maintain, or collaborate on Terraform providers
(E). None of the above
Answer: D

QUESTION NO: 2
Examine the following Terraform configuration, which uses the data source for an AWS AMI. What value should you enter for the ami argument in the AWS instance resource?

```
data "aws_ami" "ubuntu" {
  ...
}

resource "aws_instance" "web" {
  ami = _____
  instance_type = "t2.micro"

  tags = {
    Name = "HelloWorld"
  }
}
```


(A). aws_ami.ubuntu
(B). data.aws_ami.ubuntu
(C). data.aws_ami.ubuntu.id
(D). aws_ami.ubuntu.id
Answer: C
resource "aws_instance" "web" {
ami = data.aws_ami.ubuntu.id

QUESTION NO: 3
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 _____.
(A). a private install
(B). disconnected
(C). air-gapped
(D). non-traditional
Answer: D
A Terraform Enterprise install that is provisioned on a network that does not have Internet access is generally known as an air-gapped install. These types of installs require you to pull

updates, providers, etc. from external sources vs. being able to download them directly.

QUESTION NO: 4
You have modified your Terraform configuration to fix a typo in the Terraform ID of a resource from aws_security_group.http to aws_security_group.http

Original configuration:

```
resource "aws_security_group" "htp" {
  name = "http"
  ingress {
    from_port   = "80"
    to_port     = "80"
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```


Updated configuration:

Which of the following commands would you run to update the ID in state without destroying the resource?
(A). terraform refresh
(B). terraform apply
(C). terraform mv aws-security-group.htp aws-security-group.http
Answer: C

QUESTION NO: 5
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
Answer: A,C,F
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/

QUESTION NO: 6
What does terraform refresh command do?
(A). terraform refresh can be used to selectively update sections of the state file, using terraform resource level addressing.
(B). terraform refresh command basically updates the configuration file with the current state of the actual infrastructure
(C). terraform refresh is use to change/modify the infrastructure based on the existing state

file, at that moment.
(D). terraform refresh can be used to selectively update sections of the state file, using terraform resource level addressing.
(E). terraform refresh syncs the state file with the real world infrastructure.
Answer: E

QUESTION NO: 7
Eric needs to make use of module within his terraform code. Should the module always be public and open-source to be able to be used?
(A). False
(B). True
Answer: A
Terraform module need not be public and open-source. Module can be placed in -
* Local paths
* Terraform Registry
* GitHub
* Bitbucket
* Generic Git, Mercurial repositories
* HTTP URLs
* S3 buckets
* GCS buckets
https://www.terraform.io/docs/modules/sources.html

QUESTION NO: 8
You decide to move a Terraform state file to Amazon S3 from another location. You write the code below into a file called\

backend.tf .

```
terraform {
  backend "s3" {
    bucket = "my-tf-bucket"
    region = "us-east-1"
  }
}
```


You immediately run terraform apply but don't see any changes. Your state file didn't move. Which command will migrate your current state file to the new S3 remote backend?
(A). terraform push
(B). terraform init
(C). terraform refresh
(D). terraform state
Answer: B

QUESTION NO: 9
Using multi-cloud and provider-agnostic tools provides which of the following benefits?
(A). Operations teams only need to learn and manage a single tool to manage infrastructure, regardless of where the infrastructure is deployed.

(B). Increased risk due to all infrastructure relying on a single tool for management.
(C). Can be used across major cloud providers and VM hypervisors.
(D). Slower provisioning speed allows the operations team to catch mistakes before they are applied.
Answer: A,C
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 NO: 10
State locking does not happen automatically and must be specified at run
(A). False
(B). True
Answer: A
State locking happens automatically on all operations that could write state.
https://www.terraform.io/docs/state/locking.html

QUESTION NO: 11
Which of the following is the correct way to pass the value in the variable num_servers into a module with the input servers?
(A). servers = num_servers
(B). servers = variable.num_servers
(C). servers = var(num_servers)
(D). servers = var.num_servers
Answer: A

QUESTION NO: 12
Which of the following best describes the default local backend?
(A). The local backend is where Terraform Enterprise stores logs to be processed by an log collector.
(B). The local backend stores state on the local filesystem, locks the state using system APIs, and performs operations locally.
(C). The local backend is the directory where resources deployed by Terraform have direct access to in order to update their current state.
(D). The local backend is how Terraform connects to public cloud services, such as AWS, Azure, or GCP.
Answer: B
The local backend stores state on the local filesystem, locks that state using system APIs, and performs operations locally.
terraform {
backend "local" {
path = "relative/path/to/terraform.tfstate"
}
}

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

QUESTION NO: 13
Terra form installs its providers during which phase?
(A). Man
(B). Init
(C). Refresh
(D). All of the above
Answer: B

QUESTION NO: 14
Terraform Cloud is more powerful when you integrate it with your version control system (VCS) provider. Select all the supported VCS providers from the answers below. (select four)
(A). GitHub
(B). CVS Version Control
(C). Azure DevOps Server
(D). Bitbucket Cloud
(E). GitHub Enterprise
Answer: A,C,D,E
Terraform Cloud supports the following VCS providers:
- https://www.terraform.io/docs/cloud/vcs/github.html
- https://www.terraform.io/docs/cloud/vcs/github.html
- https://www.terraform.io/docs/cloud/vcs/github-enterprise.html
- https://www.terraform.io/docs/cloud/vcs/gitlab-com.html
- https://www.terraform.io/docs/cloud/vcs/gitlab-eece.html
- https://www.terraform.io/docs/cloud/vcs/bitbucket-cloud.html
- https://www.terraform.io/docs/cloud/vcs/bitbucket-server.html
- https://www.terraform.io/docs/cloud/vcs/azure-devops-server.html
- https://www.terraform.io/docs/cloud/vcs/azure-devops-services.html
https://www.terraform.io/docs/cloud/vcs/index.html#supported-vcs-providers

QUESTION NO: 15
A Terraform provider is not responsible for:
(A). Provisioning infrastructure in multiple clouds
(B). Managing actions to take based on resource differences
(C). Exposing resources and data sources based on an API
(D). Understanding API interactions with some service
Answer: B

QUESTION NO: 16
HashiCorp Configuration Language (HCL) supports user-defined functions.
(A). True
(B). False
Answer: B

QUESTION NO: 17

What is a downside to using the Vault provider to read secrets from Vault?
(A). Secrets are persisted to the state file and plans.
(B). Terraform and Vault must be running on the same version.
(C). Terraform and Vault must be running on the same physical host.
(D). Terraform requires a unique auth method to work with Vault.
Answer: A
The Vault provider allows Terraform to read from, write to, and configure Hashicorp Vault. 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 NO: 18
Your security team scanned some Terraform workspaces and found secrets stored in a plaintext in state files.
How can you protect sensitive data stored in Terraform state files?
(A). Delete the state file every time you run Terraform
(B). Store the state in an encrypted backend
(C). Edit your state file to scrub out the sensitive data
(D). Always store your secrets in a secrets.tfvars file.
Answer: B

QUESTION NO: 19
If a module uses a local variable, you can expose that value with a terraform output.
(A). True
(B). False
Answer: A
Output values are like function return values.

QUESTION NO: 20
How can you trigger a run in a Terraform Cloud workspace that is connected to a Version Control System (VCS) repository?
(A). Only Terraform Cloud organization owners can set workspace variables on VCS connected workspaces
(B). Commit a change to the VCS working directory and branch that the Terraform Cloud workspace is connected to

(C). Only members of a VCS organization can open a pull request against repositories that are connected to Terraform Cloud workspaces
(D). Only Terraform Cloud organization owners can approve plans in VCS connected workspaces
Answer: B

QUESTION NO: 21
You can migrate the Terraform backend but only if there are no resources currently being managed.
(A). False
(B). True

Answer: A
If you need to migrate 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.
https://www.terraform.io/docs/cloud/migrate/index.html

QUESTION NO: 22
Which of the following terraform subcommands could be used to remove the lock on the state for the current configuration?
(A). Unlock
(B). force-unlock
(C). Removing the lock on a state file is not possible
(D). state-unlock
Answer: B
https://www.terraform.io/docs/commands/force-unlock.html

QUESTION NO: 23
Talal is a DevOps engineer and he has deployed the production infrastructure using Terraform. He is using a very large configuration file to maintain and update the actual infrastructure. As the infrastructure have grown to a very complex and large, he has started experiencing slowness when he run runs terraform plan. What are the options for him to resolve this slowness?
(A). Use -refresh=true flag as well as the -target flag with terraform plan in order to work around this.
(B). Run terraform refresh every time before running terraform plan.
(C). Break large configurations into several smaller configurations that can each be independently applied.
(D). Use -refresh=false flag as well as the -target flag with terraform plan in order to work around this.
Answer: C,D
For larger infrastructures, querying every resource is too 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. On top of this, cloud providers almost always have API rate limiting so Terraform can only request a certain number of resources in a period of time. Larger users of Terraform make heavy use of the -refresh=false flag as well as the -target flag in order to work around this. In these scenarios, the cached state is treated as the record of truth.
Although 'Use -refresh=false flag as well as the -target flag with terraform plan in order to work around this.' is a solution, but its not always recommended. Instead of using -target as a means to operate on isolated portions of very large configurations, prefer instead to break large configurations into several smaller configurations that can each be independently applied. Data sources can be used to access information about resources created in other configurations, allowing a complex system architecture to be broken down into more manageable parts that can be updated independently.
Option 'Run terraform refresh every time before running terraform plan.' and 'Use -refresh=true flag as well as the -target flag with terraform plan in order to work around this.' is

not correct because in both the cases terraform will query every resources of the infrastructure.

QUESTION NO: 24
Why should secrets not be hard coded into Terraform code? Choose two correct answers
(A). All passwords should be rotated on a quarterly basis.
(B). The Terraform code is copied to the target resources to be applied locally and could expose secrets if a target resource is compromised.
(C). Terraform code is typically stored in version control, as well as copied to the systems from h it's run. Any of those may not have robust security mechanisms.
(D). It makes the code less reusable.
Answer: C,D

QUESTION NO: 25
Select the operating systems which are supported for a clustered Terraform Enterprise: (select four)
(A). Unix
(B). Red Hat
(C). CentOS
(D). Amazon Linux
(E). Ubuntu
Answer: B,C,D,E
https://www.terraform.io/docs/enterprise/before-installing/index.html#operating-systemrequirements

QUESTION NO: 26
You have created 2 workspaces PROD and RQA.
You have switched to RQA and provisioned RQA infrastructure from this workspace. Where is your state file stored?
(A). terraform.tfstate.d
(B). terraform.d
(C). terraform.tfstate.RQA
(D). terraform.tfstate
Answer: A

QUESTION NO: 27
You have used Terraform to create an ephemeral development environment in the cloud and are now ready to destroy all the infrastructure described by your Terraform configuration. To be safe, you would like to first see all the infrastructure that will be deleted by Terraform. Which command should you use to show all of the resources that will be deleted? (Choose two.)
(A). Run terraform plan -destroy.
(B). This is not possible. You can only show resources that will be created.
(C). Run terraform state rm *.
(D). Run terraform destroy and it will first output all the resources that will be deleted before prompting for

Answer: C,D
approval.

QUESTION NO: 28
terraform state subcommands such as list are read-only commands, do read-only commands create state backup files?
(A). Yes
(B). No
Answer: B
Subcommands that are read-only (such as list) do not write any backup files since they aren't modifying the state.
All terraform state subcommands that modify the state write backup files. The path of these backup file can be controlled with -backup.
https://www.terraform.io/docs/commands/state/index.html#backups

QUESTION NO: 29
You want to know from which paths Terraform is loading providers referenced in your Terraform configuration (files). You need to enable debug messages to find this out.
Which of the following would achieve this?
(A). Set the environment variable TF_LOG=TRACE
(B). Set verbose logging for each provider in your Terraform configuration
(C). Set the environment variable TF_VAR_log=TRACE
(D). Set the environment variable TF_LOG_PATH
Answer: A

QUESTION NO: 30
Which Terraform collection type should you use to store key/value pairs?
(A). set
(B). tuple
(C). list
(D). map

Answer: D

QUESTION NO: 31
What is the result of the following terraform function call?
(A). True
(B). False
Answer: B
https://www.terraform.io/docs/configuration/functions/index.html

QUESTION NO: 32
What is the name of the default file where Terraform stores the state?
Type your answer in the field provided. The text field is not case-sensitive and all variations of the correct answer are accepted.
Answer:
Terraformtfstate

QUESTION NO: 33
Which statements best describes what the local variable assignment is doing in the following code snippet:
(A). Create a distinct list of route table name objects
(B). Create a map of route table names to subnet names
(C). Create a map of route table names from a list of subnet names
(D). Create a list of route table names eliminating duplicates
Answer: D

QUESTION NO: 34
Open source Terraform can only import publicly-accessible and open-source modules.
(A). True
(B). False
Answer: B

QUESTION NO: 35
You have created a custom variable definition file my_vars.tfvars. How will you use it for provisioning infrastructure?
(A). terraform apply -var-state-file ="my_vars.tfvars"
(B). terraform apply var-file="my_vars.tfvars"
(C). terraform plan -var-file="my_vars.tfvar"
(D). terraform apply -var-file="my_vars.tfvars"
Answer: D
To set lots of variables, it is more convenient to specify their values in a variable definitions file (with a filename ending in either .tfvars or .tfvars.json) and then specify that file on the command line with -var-file:
terraform apply -var-file="my_vars.tfvars"
https://www.terraform.io/docs/configuration/variables.html#variable-definitions-tfvars-files

QUESTION NO: 36
Your team uses terraform OSS . You have created a number of resuable modules for important , independent network components that you want to share with your team to enhance consistency . What is the correct option/way to do that?
(A). Terraform modules cannot be shared in OSS version . Each developer needs to maintain their own modules and leverage them in the main tf file.
(B). Upload your modules with proper versioning in the terraform public module registry . Terraform OSS is directly integrated with the public module registry , and can reference the modules from the code in the main tf file.
(C). Terraform module sharing is only available in Enterprise version via terraform private module registry , so no way to enable it in OSS version.
(D). Store your modules in a NAS/ shared file server , and ask your team members to directly reference the code from there. This is the only viable option in terraform OSS ,which is better than individually maintaining module versions for every developer.
Answer: B

Software development encourages code reuse through reusable artifacts, such as libraries, packages and modules. Most programming languages enable developers to package and publish these reusable components and make them available on a registry or feed. For example, Python has Python Package Index and PowerShell has PowerShell Gallery.
For Terraform users, the Terraform Registry enables the distribution of Terraform modules, which are reusable configurations. The Terraform Registry acts as a centralized repository for module sharing, making modules easier to discover and reuse.
The Registry is available in two variants:
* Public Registry houses official Terraform providers -- which are services that interact with an API to expose and manage a specific resource -- and community-contributed modules.
* Private Registry is available as part of the Terraform Cloud, and can host modules internally within an organization.
https://www.terraform.io/docs/registry/index.html

QUESTION NO: 37
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). Backends
(B). Data sources
(C). Named values
(D). Functions
Answer: D
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.
https://www.terraform.io/docs/configuration/functions.html

QUESTION NO: 38
How would you be able to reference an attribute from the vsphere_datacenter data source for use with the argument within the vsprere_folder resource in the following configuration?

```
data "vsphere_datacenter" "dc" {}

resource "vsphere_folder" "parent" {
          path = "Production"
          type = "vm"
  datacenter_id = _________________
}
```


(A). vsphere_datacenter.dc.id
(B). data.vsphere_datacenter.dc

(C). data.dc,id
(D). data.vsphere_datacenter.dc.id
Answer: D

QUESTION NO: 39
Select all features which are exclusive to Terraform Enterprise. (Select Three)
(A). Sentinel
(B). Cost Estimation
(C). Audit Logs
(D). Clustering
(E). SAML/SSO
Answer: C,D,E
Sentinel and Cost Estimation are also available in Terraform Cloud
https://www.hashicorp.com/products/terraform/pricing/

QUESTION NO: 40
Terraform must track metadata such as resource dependencies. Where is this data stored?
(A). workspace
(B). backend
(C). state file
(D). metadata store
Answer: C
Terraform typically uses the configuration to determine dependency order. However, when you delete a resource from a Terraform configuration, Terraform must know how to delete that resource. Terraform can see that a mapping exists for a resource not in your configuration and plan to destroy. However, since the configuration no longer exists, the order cannot be determined from the configuration alone.
To ensure correct operation, Terraform retains a copy of the most recent set of dependencies within the state. Now Terraform can still determine the correct order for destruction from the state when you delete one or more items from the configuration.
https://www.terraform.io/docs/state/purpose.html#metadata

QUESTION NO: 41
What is terraform refresh intended to detect?
(A). Terraform configuration code changes
(B). Empty state files
(C). State file drift
(D). Corrupt state files
Answer: C

QUESTION NO: 42
Which one is the right way to import a local module names consul?
(A). module "consul" { source = "consul"}
(B). module "consul" { source = "./consul"}
(C). module "consul" { source = "../consul"}

(D). module "consul" { source = "module/consul"}

Answer: B,C

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

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

QUESTION NO: 43

A user has created three workspaces using the command line - prod, dev, and test. The user wants to create a fourth workspace named stage. Which command will the user execute to accomplish this?

(A). terraform workspace new stage

(B). terraform workspace -new stage

(C). terraform workspace -create stage

(D). terraform workspace create stage

Answer: A

The terraform workspace new command is used to create a new workspace.

https://www.terraform.io/docs/commands/workspace/new.html

QUESTION NO: 44

A Terraform provisioner must be nested inside a resource configuration block.

(A). True

(B). False

Answer: A

Most provisioners require access to the remote resource via SSH or WinRM, and expect a nested connection block with details about how to connect.

QUESTION NO: 45

During a terraform apply, a resource is successfully created but eventually fails during provisioning. What happens to the resource?

(A). The resource will be planned for destruction and recreation upon the next terraform appl y

(B). Terraform will retry to provision again.

(C). The failure of provisioner will be ignored and it will not cause a failure to terraform apply

(D). The resource will be automatically destroyed.

Answer: A

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. Terraform does this because a failed provisioner can leave a resource in a semi-configured state. Because Terraform cannot reason about what the provisioner does, the only way to ensure proper creation of a resource is to recreate it. This is tainting.

You can change this behavior by setting the on_failure attribute, which is covered in detail below.

https://www.terraform.io/docs/provisioners/index.html#creation-time-provisioners

https://www.terraform.io/docs/provisioners/index.html#destroy-time-provisioners
https://www.terraform.io/docs/provisioners/index.html#failure-behavior

QUESTION NO: 46
You have declared a variable called var.list which is a list of objects that all have an attribute id.
Which options will produce a list of the IDs? (Choose two.)
(A). { for o in var.list : o => o.id }
(B). var.list[*].id
(C). [var.list[*].id]
(D). [for o in var.list : o.id]
Answer: A,B

QUESTION NO: 47
If writing Terraform code that adheres to the Terraform style conventions, how would you properly indent each nesting level compared to the one above it?
(A). With four spaces
(B). With a tab
(C). With three spaces
(D). With two spaces
Answer: D

QUESTION NO: 48
After creating a new workspace "PROD" you need to run the command terraform select PROD to switch to it.
(A). False
(B). True
Answer: A
By default, when you create a new workspace you are automatically switched to it To create a new workspace and switch to it, you can use terraform workspace new <new_workspace_name>; to switch to a existing workspace you can use terraform workspace select <existing_workspace_name>; Example:
$ terraform workspace new example
Created and switched to workspace "example"!
You're now on a new, empty workspace. Workspaces isolate their state, so if you run "terraform plan" Terraform will not see any existing state for this configuration.

QUESTION NO: 49
In the example below, where is the value of the DNS record's IP address originating from?
1. resource "aws_route53_record" "www"
2. {
3. zone_id = aws_route53_zone.primary.zone_id
4. name = "www.example.com"
5. type = "A"
6. ttl = "300"
7. records = [module.web_server.instance_ip_address]

8. }
(A). The regular expression named module.web_server
(B). The output of a module named web_server
(C). By querying the AWS EC2 API to retrieve the IP address
(D). Value of the web_server parameter from the variables.tf file
Answer: B
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_address, you could access that value as module.web_server.instance_ip_address.

QUESTION NO: 50
Which argument(s) are required when declaring a Terraform variable?
(A). type
(B). default
(C). description
(D). All of the above
(E). None of the above

Answer: A

QUESTION NO: 51
Anyone can publish and share modules on the Terraform Public Module Registry, and meeting the requirements for publishing a module is extremely easy. Select from the following list all valid requirements. (select three)
(A). The module must be PCI/HIPPA compliant.
(B). Module repositories must use this three-part name format, terraform-- .
(C). The registry uses tags to identify module versions.
(D). Release tag names must be for the format x.y.z, and can optionally be prefixed with a v .
(E). The module must be on GitHub and must be a public repo.
Answer: C,D,E
https://www.terraform.io/docs/registry/modules/publish.html#requirements

QUESTION NO: 52
All standard backend types support state storage, locking, and remote operations like plan. apply and destroy.
(A). True
(B). False
Answer: A

QUESTION NO: 53
You want to use different AMI images for different regions and for the purpose you have defined following code block.
1. variable "images"
2. {
3. type = "map"

4.
5. default = {
6. us-east-1 = "image-1234"
7. us-west-2 = "image-4567"
8. us-west-1 = "image-4589"
9. }
10. }
What of the following approaches needs to be followed in order to select image-4589?
(A). var.images["us-west-1"]
(B). var.images[3]
(C). var.images[2]
(D). lookup(var.images["us-west-1"]
Answer: A

QUESTION NO: 54
Terraform provisioners that require authentication can use the _____ block.
(A). connection
(B). credentials
(C). secrets
(D). ssh
Answer: A

QUESTION NO: 55
The terraform.tfstate file always matches your currently built infrastructure.
(A). True
(B). False
Answer: B

QUESTION NO: 56
Choose the answer that correctly completes the sentence: _____backends support state locking.
(A). All
(B). No
(C). Only local
(D). Some
Answer: D

QUESTION NO: 57
You have declared an input variable called environment in your parent module. What must you do to pass the value to a child module in the configuration?
(A). Add node_count = var.node_count
(B). Declare the variable in a terraform.tfvars file
(C). Declare a node_count input variable for child module
(D). Nothing, child modules inherit variables of parent module
Answer: C

QUESTION NO: 58
Which of the following Terraform files should be ignored by Git when committing code to a repo? (select Three)
(A). Files named exactly terraform.tfvars or terraform.tfvars.json.
(B). Any files with names ending in .auto.tfvars or .auto.tfvars.json.
(C). input.tf
(D). terraform.tfstate
(E). output.tf
Answer: A,B,D
The .gitignore file should be configured to ignore Terraform files that either contain sensitive data or are not required to save.
Terraform state (terraform.tfstate) 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.
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 NO: 59
When using multiple configurations of the same Terraform provider, what meta-argument must be included in any non-default provider configurations?
(A). name
(B). alias
(C). depends_on
(D). id
Answer: B

QUESTION NO: 60
You have created a custom variable definition file testing.tfvars. How will you use it for provisioning infrastructure?
(A). terraform apply -var-state-file ="testing.tfvars"
(B). terraform plan -var-file="testing.tfvar"
(C). terraform apply -var-file="testing.tfvars"
(D). terraform apply var-file="testing.tfvars"

Answer: C
https://www.terraform.io/docs/configuration/variables.html

QUESTION NO: 61
A data block requests that Terraform read from a given data source and export the result under the given local name.
(A). False
(B). True
Answer: B

QUESTION NO: 62

State is a requirement for Terraform to function
(A). True
(B). False
Answer: A
State is a necessary requirement for Terraform to function. It is often asked if it is possible for Terraform to work without state, or for Terraform to not use state and just inspect cloud resources on every run.
Purpose of Terraform State
State is a necessary requirement for Terraform to function. It is often asked if it is possible for Terraform to work without state, or for Terraform to not use state and just inspect cloud resources on every run. This page will help explain why Terraform state is required.
As you'll see from the reasons below, state is required. And in the scenarios where Terraform may be able to get away without state, doing so would require shifting massive amounts of complexity from one place (state) to another place (the replacement concept).
1. Mapping to the Real World
Terraform requires some sort of database to map Terraform config to the real world. When you have a resource resource "aws_instance" "foo" in your configuration, Terraform uses this map to know that instance i- abcd1234 is represented by that resource.
For some providers like AWS, Terraform could theoretically use something like AWS tags. Early prototypes of Terraform actually had no state files and used this method. However, we quickly ran into problems. The first major issue was a simple one: not all resources support tags, and not all cloud providers support tags.
Therefore, for mapping configuration to resources in the real world, Terraform uses its own state structure.
2. Metadata
Alongside the mappings between resources and remote objects, Terraform must also track metadata such as resource dependencies.
Terraform typically uses the configuration to determine dependency order. However, when you delete a resource from a Terraform configuration, Terraform must know how to delete that resource. Terraform can see that a mapping exists for a resource not in your configuration and plan to destroy. However, since the configuration no longer exists, the order cannot be determined from the configuration alone.
To ensure correct operation, Terraform retains a copy of the most recent set of dependencies within the state. Now Terraform can still determine the correct order for destruction from the state when you delete one or more items from the configuration.
One way to avoid this would be for Terraform to know a required ordering between resource types. For example, Terraform could know that servers must be deleted before the subnets they are a part of. The complexity for this approach quickly explodes, however: in addition to Terraform having to understand the ordering semantics of every resource for every cloud, Terraform must also understand the ordering across providers.
Terraform also stores other metadata for similar reasons, such as a pointer to the provider configuration that was most recently used with the resource in situations where multiple aliased providers are present.
3. Performance
In addition to basic mapping, Terraform stores a cache of the attribute values for all

resources in the state. This is the most optional feature of Terraform state and is done only as a performance improvement.
When running a terraform plan, Terraform must know the current state of resources in order to effectively determine the changes that it needs to make to reach your desired configuration.
For small infrastructures, Terraform can query your providers and sync the latest attributes from all your resources. This is the default behavior of Terraform: for every plan and apply, Terraform will sync all resources in your state.
For larger infrastructures, querying every resource is too 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. On top of this, cloud providers almost always have API rate limiting so Terraform can only request a certain number of resources in a period of time. Larger users of Terraform make heavy use of the -refresh=false flag as well as the -target flag in order to work around this. In these scenarios, the cached state is treated as the record of truth.
4. Syncing
In the default configuration, Terraform stores the state in a file in the current working directory where Terraform was run. This is okay for getting started, but when using Terraform in a team it is important for everyone to be working with the same state so that operations will be applied to the same remote objects.
Remote state is the recommended solution to this problem. With a fully-featured state backend, Terraform can use remote locking as a measure to avoid two or more different users accidentally running Terraform at the same time, and thus ensure that each Terraform run begins with the most recent updated state.

QUESTION NO: 63
What kind of configuration block will create an infrastructure object with settings specified in the block?
(A). state
(B). provider
(C). resource
(D). data
Answer: C

QUESTION NO: 64
What is the name assigned by Terraform to reference this resource?

```
resource "azurerm_resource_group" "dev" {
  name = "test"
  location = "westus"
}
```


(A). dev
(B). azurerm_resource_group
(C). azurerm
(D). test
Answer: D

QUESTION NO: 65
You have deployed a new webapp with a public IP address on a clod provider. However, you did not create any outputs for your code.
What is the best method to quickly find the IP address of the resource you deployed?
(A). Run terraform output ip_address to view the result
(B). In a new folder, use the terraform_remote_state data source to load in the state file, then write an output for each resource that you find the state file
(C). Run terraform state list to find the name of the resource, then terraform state show to find the attributes including public IP address
(D). Run terraform destroy then terraform apply and look for the IP address in stdout
Answer: A

QUESTION NO: 66
Select all Operating Systems that Terraform is available for. (select five)
(A). Linux
(B). macOS
(C). Unix
(D). Solaris
(E). Windows
(F). FreeBSD
Answer: A,B,D,E,F
Terraform is available for macOS, FreeBSD, OpenBSD, Linux, Solaris, Windows
https://www.terraform.io/downloads.html

QUESTION NO: 67
Your company has been using Terraform Cloud for a some time now . But every team is creating their own modules , and there is no standardization of the modules , with each team creating the resources in their own unique way . You want to enforce a standardization of the modules across the enterprise . What should be your approach.
(A). Create individual workspaces for each team , and ask them to share modules across workspaces.
(B). Implement a Private module registry in Terraform cloud , and ask teams to reference them.
(C). Upgrade to Terraform enterprise , since this is not possible in terraform cloud.
(D). Upload the modules in the terraform public module registry , and ask teams to reference them
Answer: B
Terraform Cloud's private module registry helps you share Terraform modules across your organization. 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. By design, the private module registry works much like the public Terraform Registry. If you're already used the public registry, Terraform Cloud's registry will feel familiar.
Understand the different offerings in Terraform OS, Terraform Cloud and Terraform Enterprise. Terraform Cloud's private module registry helps you share Terraform modules across your organization.

https://www.terraform.io/docs/cloud/registry/index.html
https://www.terraform.io/docs/cloud/registry/publish.html

QUESTION NO: 68
In Terraform Enterprise, a workspace can be mapped to how many VCS repos?
(A). 5
(B). 2
(C). 3
(D). 1
Answer: D
A workspace can only be configured to a single VCS repo, however, multiple workspaces can use the same repo.
https://www.terraform.io/docs/cloud/workspaces/vcs.html

QUESTION NO: 69
terraform destroy is the only way to remove infrastructure.
(A). True
(B). False
Answer: B

QUESTION NO: 70
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). Sentinel
(B). Module registry
(C). Functions
(D). Workspaces
Answer: A
Sentinel is a language and framework for policy built to be embedded in existing software to enable fine-grained, logic-based policy decisions. A policy describes under what circumstances certain behaviors are allowed. Sentinel is an enterprise-only feature.
https://www.youtube.com/watch?v=Vy8s7AAvU6g&feature=emb_title

QUESTION NO: 71
You have multiple developers working on a terraform project (using terraform OSS), and have saved the terraform state in a remote S3 bucket . However ,team is intermittently experiencing inconsistencies in the provisioned infrastructure / failure in the code . You have traced this problem to simultaneous/concurrent runs of terraform apply command for 2/more developers . What can you do to fix this problem?
(A). Use terraform workspaces feature, this will fix this problem by default , as every developer will have their own state file , and terraform will merge them on server side on its own.
(B). Structure your team in such a way that only one individual will run terraform apply , everyone will just make changes and share with him. Then there will be no chance of any inconsistencies.
(C). Stop using remote state , and store the developer tfstate in their own machine . Once a

day , all developers should sit together and merge the state files manually , to avoid any inconsistencies.
(D). Enable terraform state locking for the S3 backend using DynamoDB table. This prevents others from acquiring the lock and potentially corrupting your state.
Answer: D
S3 backend support state locking using DynamoDB.
https://www.terraform.io/docs/state/locking.html

QUESTION NO: 72
A single terraform resource file that defines an aws_instance resource can simple be renamed to azurerm_virtual_machine in order to switch cloud providers
(A). True
(B). False
Answer: B
Providers usually require some configuration of their own to specify endpoint URLs, regions, authentication settings.
Providers Initialization can be done by either explicitly via a provider block or by adding a resource from that provide
https://www.terraform.io/docs/configuration/providers.html

QUESTION NO: 73
Which of the following variable definition files will terraform load automatically?
(A). terraform.tfvar
(B). Any files with names ending in .auto.tfvars.json
(C). terraform.tfvars
(D). terraform.tfvars.json
Answer: B,C,D
Terraform also automatically loads a number of variable definitions files if they are present:
Files named exactly terraform.tfvars or terraform.tfvars.json.
Any files with names ending in .auto.tfvars or .auto.tfvars.json.
https://www.terraform.io/docs/configuration/variables.html
https://www.terraform.io/docs/configuration/variables.html#variable-definitions-tfvars-files

QUESTION NO: 74
In contrast to Terraform Open Source, when working with Terraform Enterprise and Cloud Workspaces, conceptually you could think about them as completely separate working directories.
(A). True
(B). False
Answer: B

QUESTION NO: 75
terraform refresh will update the state file?
(A). True
(B). False

Answer: A
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.
This 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.

QUESTION NO: 76
Which of the following is not a way to trigger terraform destroy ?
(A). Passing ---destroy at the end of apian request
(B). Running terraform destroy from the correct directory and then typing "yes" when prompted in the CLI
(C). Using the destroy command with auto approve
(D). Delete the state file and run terraform apply
Answer: A

QUESTION NO: 77
Provisioners should only be used as a last resort.
(A). False
(B). True
Answer: B
Provisioners are a Last Resort
Terraform includes the concept of provisioners as a measure of pragmatism, knowing that there will always be certain behaviors that can't be directly represented in Terraform's declarative model.
However, they also add a considerable amount of complexity and uncertainty to Terraform usage. Firstly, Terraform cannot model the actions of provisioners as part of a plan because they can in principle take any action. Secondly, successful use of provisioners requires coordinating many more details than Terraform usage usually requires: direct network access to your servers, issuing Terraform credentials to log in, making sure that all of the necessary external software is installed, etc.
The following sections describe some situations which can be solved with provisioners in principle, but where better solutions are also available. We do not recommend using provisioners for any of the use-cases described in the following sections.
Even if your specific use-case is not described in the following sections, we still recommend attempting to solve it using other techniques first, and use provisioners only if there is no other option.
https://www.terraform.io/docs/provisioners/index.html

QUESTION NO: 78
Please identify the offerings which are unique to Terraform Enterprise, and not available in either Terraform OSS, or Terraform Cloud. Select four.
(A). Audit Logs
(B). Private Network Connectivity
(C). VCS Integration

(D). Sentinel
(E). Clustering
Answer: A,B,E
https://www.hashicorp.com/products/terraform/pricing/

QUESTION NO: 79
What is the workflow for deploying new infrastructure with Terraform?
(A). terraform plan to import the current infrastructure to the state file, make code changes, and terraform apply to update the infrastructure
(B). Write a Terraform configuration, run terraform show to view proposed changes, and terraform apply to create new infrastructure.
(C). terraform plan to import the current infrastructure to the state file, make code changes, and terraform apply to update the infrastructure
(D). Write a Terraform configuration, run terraform init, run terraform plan to view planned infrastructure
Answer: C
changes, and terraform apply to create new infrastructure.
Reference:
+run+terraform+plan+to+view+planned+infrastructure+changes%2C+and+terraform+apply+to+create+new
+infrastructure.&oq=Write+a+Terraform+configuration%2C+run+terraform+init%2C+run+terraform+plan+to
+view+planned+infrastructure+changes%2C+and+terraform+apply+to+create+new
+infrastructure.&aqs=chrome..69i57.556j0j7&sourceid=chrome&ie=UTF-8

QUESTION NO: 80
When running the command terraform taint against a managed resource you want to force recreation upon, Terraform will immediately destroy and recreate the resource.
(A). True
(B). False

Answer: B

QUESTION NO: 81
What is a key benefit of the Terraform state file?
(A). A state file represents a source of truth for resources provisioned with a public cloud console
(B). A state file represents a source of truth for resources provisioned with Terraform
(C). A state file represents the desired state expressed by the Terraform code files
Answer: C
D A state file can be used to schedule recurring infrastructure tasks

QUESTION NO: 82
Terraform can run on Windows or Linux, but it requires a Server version of the Windows operating system.
(A). True

(B). False
Answer: A

QUESTION NO: 83
Given the Terraform configuration below, in which order will the resources be created?
1. resource "aws_instance" "web_server"
2. {
3. ami = "ami-b374d5a5"
4. instance_type = "t2.micro"
5. }
6. resource "aws_eip" "web_server_ip"
7. {
8. vpc = true instance = aws_instance.web_server.id
9. }
(A). aws_eip will be created first
aws_instance will be created second
(B). aws_eip will be created first
aws_instance will be created second
(C). Resources will be created simultaneously
(D). aws_instance will be created first
aws_eip will be created second
Answer: D
Implicit and Explicit Dependencies
By studying the resource attributes used in interpolation expressions, Terraform can automatically infer when one resource depends on another. In the example above, the reference to aws_instance.web_server.id creates an implicit dependency on the aws_instance named web_server.
Terraform uses this dependency information to determine the correct order in which to create the different resources.
# Example of Implicit Dependency
resource "aws_instance" "web_server" {
ami = "ami-b374d5a5"
instance_type = "t2.micro"
}
resource "aws_eip" "web_server_ip" {
vpc = true
instance = aws_instance.web_server.id
}
In the example above, Terraform knows that the aws_instance must be created before the aws_eip.
Implicit dependencies via interpolation expressions are the primary way to inform Terraform about these relationships, and should be used whenever possible.
Sometimes there are dependencies between resources that are not visible to Terraform. The depends_on argument is accepted by any resource and accepts a list of resources to create explicit dependencies for.

For example, perhaps an application we will run on our EC2 instance expects to use a specific Amazon S3 bucket, but that dependency is configured inside the application code and thus not visible to Terraform. In that case, we can use depends_on to explicitly declare the dependency:
# Example of Explicit Dependency
# New resource for the S3 bucket our application will use.
resource "aws_s3_bucket" "example" {
bucket = "terraform-getting-started-guide"
acl = "private"
}
# Change the aws_instance we declared earlier to now include "depends_on" resource
"aws_instance" "example" { ami = "ami-2757f631" instance_type = "t2.micro"
# Tells Terraform that this EC2 instance must be created only after the
# S3 bucket has been created.
depends_on = [aws_s3_bucket.example]
}
https://learn.hashicorp.com/terraform/getting-started/dependencies.html

QUESTION NO: 84
When does terraform apply reflect changes in the cloud environment?
(A). Immediately
(B). However long it takes the resource provider to fulfill the request
(C). After updating the state file
(D). Based on the value provided to the -refresh command line argument
(E). None of the above
Answer: E

QUESTION NO: 85
If you enable TF_LOG = DEBUG, the log will be stored in syslog.log file in the currect directory.
(A). False
(B). True
Answer: A
https://www.terraform.io/docs/internals/debugging.html

QUESTION NO: 86
True or False? When using the Terraform provider for Vault, the tight integration between these HashiCorp tools provides the ability to mask secrets in the terraform plan and state files.
(A). False
(B). True
Answer: A
Currently, Terraform has no mechanism to redact or protect secrets that are returned via data sources, so secrets read via this provider will be persisted into the Terraform state, into any plan files, and in some cases in the console output produced while planning and applying.

These artifacts must, therefore, all be protected accordingly.

QUESTION NO: 87
Select the most accurate statement to describe the Terraform language from the following list.
(A). Terraform is an immutable, declarative, Infrastructure as Code provisioning language based on Hashicorp Configuration Language, or optionally JSON.
(B). Terraform is a mutable, declarative, Infrastructure as Code configuration management language based on Hashicorp Configuration Language, or optionally JSON.
(C). Terraform is an immutable, procedural, Infrastructure as Code configuration management language based on Hashicorp Configuration Language, or optionally JSON.
(D). Terraform is a mutable, procedural, Infrastructure as Code provisioning language based on Hashicorp Configuration Language, or optionally YAML.
Answer: A
Terraform is not a configuration management tool - https://www.terraform.io/intro/vs/chefpuppet.html Terraform is a declarative language - https://www.terraform.io/docs/configuration/index.html Terraform supports a syntax that is JSON compatible - https://www.terraform.io/docs/configuration/syntax-json.html Terraform is primarily designed on immutable infrastructure principles - https://www.hashicorp.com/resources/what-is-mutable-vs-immutable-infrastructure

QUESTION NO: 88
What is the purpose of a Terraform workspace in either open source of enterprise?
(A). Workspaces allow you to manage collections of infrastructure in state files.
(B). A logical separation of business units
(C). A method of grouping multiple infrastructure security policies
(D). Provides limited access to a cloud environment
Answer: B

QUESTION NO: 89
Terraform validate reports syntax check errors from which of the following scenarios?
(A). Code contains tabs indentation instead of spaces
(B). There is missing value for a variable
(C). The state files does not match the current infrastructure
(D). None of the above
Answer: B

QUESTION NO: 90
When using a module block to reference a module stored on the public Terraform Module Registry such as:

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


How do you specify version 1.0.0?
(A). Modules stored on the public Terraform Module Registry do not support versioning

(B). Append ?ref=v1.0.0 argument to the source path
(C). Add version = "1.0.0" attribute to module block
(D). Nothing - modules stored on the public Terraform Module Registry always default to version 1.0.0
Answer: A

QUESTION NO: 91
What value does the Terraform Cloud/Terraform Enterprise private module registry provide over the public Terraform Module Registry?
(A). The ability to share modules with public Terraform users and members of Terraform Enterprise Organizations
(B). The ability to tag modules by version or release
(C). The ability to restrict modules to members of Terraform Cloud or Enterprise organizations
(D). The ability to share modules publicly with any user of Terraform
Answer: D
Terraform Registry is an index of modules shared publicly using this protocol. This public registry is the easiest way to get started with Terraform and find modules created by others in the community.

QUESTION NO: 92
From the code below, identify the implicit dependency:
(A). The EIP with an id of ami-2757f631
(B). The AMI used for the EC2 instance
(C). The EC2 instance labeled web_server
(D). The S3 bucket labeled company_data
Answer: C

QUESTION NO: 93
Terraform can only manage resource dependencies if you set them explicitly with the depends_on argument.
(A). True
(B). False
Answer: A

QUESTION NO: 94
You've used Terraform to deploy a virtual machine and a database. You want to replace this virtual machine instance with an identical one without affecting the database. What is the best way to achieve this using Terraform?
(A). Use the Terraform taint command targeting the VMs then run Terraform plan and Terraform apply
(B). Delete the Terraform VM resources from your Terraform code then run Terraform plan and terraform apply
(C). Use the terraform apply command targeting the VM resources only
(D). Use the terraform state rm command to remove the VM from state file

Answer: A

QUESTION NO: 95
From the answers below, select the advantages of using Infrastructure as Code.
(A). Provide a codified workflow to develop customer-facing applications.
(B). Safely test modifications using a "dry run" before applying any actual changes.
(C). Easily integrate with application workflows (GitLab Actions, Azure DevOps, CI/CD tools).
(D). Easily change and update existing infrastructure.
(E). Provide reusable modules for easy sharing and collaboration.
Answer: B,C,D,E
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 NO: 96
terraform validate validate validates that your infrastructure matches the Terraform state file.
(A). True
(B). False
Answer: B

QUESTION NO: 97
Where does the Terraform local backend store its state?
(A). In the /tmp directory
(B). In the terraform.tfvars file
(C). In the terraform.tfstate file
(D). In the user's .terraformrc file
Answer: C
The local backend stores state on the local filesystem, locks that state using system APIs, and performs operations locally.

QUESTION NO: 98
What is one disadvantage of using dynamic blocks in Terraform?
(A). They cannot be used to loop through a list of values
(B). Dynamic blocks can construct repeatable nested blocks
(C). They make configuration harder to read and understand
(D). Terraform will run more slowly
Answer: A

QUESTION NO: 99
A provider configuration block is required in every Terraform configuration.
Example:

```
provider "provider_name" {
  ...
}
```

(A). True
(B). False
Answer: A

QUESTION NO: 100
You have created an AWS EC2 instance of type t2.micro through your terraform configuration file ec2.tf . Now you want to change the instance type from t2.micro to t2.medium. Accordingly you have changed your configuration file and and ran terraform plan. After running terraform plan you check the output and saw one instance will be updated from t2.micro --> t2.medium. After this you went to grab a coffee without running terraform apply and meanwhile a member of your team changed the instance type of that EC2 instance to t2.medium from aws console. After coming to your desk you run terraform apply. What will happen?
(A). No resource will be updated and you will see the message : Apply Complete ! Resources : 0 added, 0 changed, 0 destroyed.
(B). The instance type will be changed to t2.micro and again will be changed to t2.medium
(C). terraform apply will through an error.
(D). 1 resource will be updated and you will see the message : Apply Complete ! Resources : 0 added, 1 changed, 0 destroyed.
Answer: A

QUESTION NO: 101
Given the below resource configuration -
resource "aws_instance" "web" { # ... count = 4 }
What does the terraform resource address aws_instance.web refer to?
(A). It refers to all 4 web instances , together , for further individual segregation , indexing is required , with a 0 based index.
(B). It refers to the last web EC2 instance , as by default , if no index is provided , the last / N-1 index is used.
(C). It refers to the first web EC2 instance out of the 4 ,as by default , if no index is provided , the first / 0th index is used.
(D). The above will result in a syntax error , as it is not syntactically correct . Resources defined using count , can only be referenced using indexes.
Answer: A
A Resource Address is a string that references a specific resource in a larger infrastructure. An address is made up of two parts:
[module path][resource spec]
Module path:
A module path addresses a module within the tree of modules. It takes the form:
module.A.module.B.module.C...
Multiple modules in a path indicate nesting. If a module path is specified without a resource spec, the address applies to every resource within the module. If the module path is omitted, this addresses the root module.
Given a Terraform config that includes:
resource "aws_instance" "web" {

# ...
count = 4
}
An address like this:
aws_instance.web[3]
Refers to only the last instance in the config, and an address like this:
aws_instance.web
Refers to all four "web" instances.
https://www.terraform.io/docs/internals/resource-addressing.html

QUESTION NO: 102
When you initialize Terraform, where does it cache modules from the public Terraform Module Registry?
(A). On disk in the /tmp directory
(B). In memory
(C). On disk in the .terraform sub-directory
(D). They are not cached
Answer: C

QUESTION NO: 103
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 updated in place.
(B). The resource will be created.
(C). Terraform can't determine how to proceed due to a problem with the state file.
(D). The resource will be destroyed and recreated.
Answer: A
The prefix -/+ means that Terraform will destroy and recreate the resource, rather than updating it in-place.
The prefix ~ means that some attributes and resources can be updated in-place.
$ terraform apply
aws_instance.example: Refreshing state... [id=i-0bbf06244e44211d1]
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
-/+ destroy and then create replacement
Terraform will perform the following actions:
# aws_instance.example must be replaced
-/+ resource "aws_instance" "example" {
~ ami = "ami-2757f631" -> "ami-b374d5a5" # forces replacement
~ arn = "arn:aws:ec2:us-east-1:130490850807:instance/i-0bbf06244e44211d1" -> (known after apply)
~ associate_public_ip_address = true -> (known after apply)
~ availability_zone = "us-east-1c" -> (known after apply)
~ cpu_core_count = 1 -> (known after apply)
~ cpu_threads_per_core = 1 -> (known after apply)

```
- disable_api_termination = false -> null
- ebs_optimized = false -> null
get_password_data = false
+ host_id = (known after apply)
~ id = "i-0bbf06244e44211d1" -> (known after apply)
~ instance_state = "running" -> (known after apply)
instance_type = "t2.micro"
~ ipv6_address_count = 0 -> (known after apply)
~ ipv6_addresses = [] -> (known after apply)
+ key_name = (known after apply)
- monitoring = false -> null
+ network_interface_id = (known after apply)
+ password_data = (known after apply)
+ placement_group = (known after apply)
~ primary_network_interface_id = "eni-0f1ce5bdae258b015" -> (known after apply)
~ private_dns = "ip-172-31-61-141.ec2.internal" -> (known after apply)
~ private_ip = "172.31.61.141" -> (known after apply)
~ public_dns = "ec2-54-166-19-244.compute-1.amazonaws.com" -> (known after apply)
~ public_ip = "54.166.19.244" -> (known after apply)
~ security_groups = [
- "default",
] -> (known after apply)
source_dest_check = true
~ subnet_id = "subnet-1facdf35" -> (known after apply)
~ tenancy = "default" -> (known after apply)
~ volume_tags = {} -> (known after apply)
~ vpc_security_group_ids = [
- "sg-5255f429",
] -> (known after apply)
- credit_specification {
- cpu_credits = "standard" -> null
}
+ ebs_block_device {
+ delete_on_termination = (known after apply)
+ device_name = (known after apply)
+ encrypted = (known after apply)
+ iops = (known after apply)
+ snapshot_id = (known after apply)
+ volume_id = (known after apply)
+ volume_size = (known after apply)
+ volume_type = (known after apply)
}
+ ephemeral_block_device {
+ device_name = (known after apply)
+ no_device = (known after apply)
```

```
+ virtual_name = (known after apply)
}
+ network_interface {
+ delete_on_termination = (known after apply)
+ device_index = (known after apply)
+ network_interface_id = (known after apply)
}
~ root_block_device {
~ delete_on_termination = true -> (known after apply)
~ iops = 100 -> (known after apply)
~ volume_id = "vol-0079e485d9e28a8e5" -> (known after apply)
~ volume_size = 8 -> (known after apply)
~ volume_type = "gp2" -> (known after apply)
}
}
```

Plan: 1 to add, 0 to change, 1 to destroy.

QUESTION NO: 104

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

(A). Input Parameters
(B). Module
(C). Regular Expressions
(D). Output Value

Answer: A

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.
https://www.terraform.io/docs/configuration/variables.html

QUESTION NO: 105

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?

(A). Terraform registry
(B). custom VCS integration
(C). private module registry
(D). remote runs

Answer: C

QUESTION NO: 106

Which of the below commands will rename a EC2 instance without destroying and recreating it?

(A). terraform state mv
(B). terraform mv
(C). terraform plan
(D). terraform plan mv

Answer: A

QUESTION NO: 107
Which backend does the Terraform CLI use by default?
(A). Terraform Cloud
(B). Consul
(C). Remote
(D). Local
Answer: D

QUESTION NO: 108
Ric wants to enable detail logging and he wants highest verbosity of logs. Which of the following environment variable settings is correct option for him to select.
(A). Set TF_LOG = DEBUG
(B). Set VAR_TF = TRACE
(C). Set TF_LOG = TRACE
(D). Set VAR_TF_LOG = TRACE
Answer: C
https://www.terraform.io/docs/internals/debugging.html

QUESTION NO: 109
True or False: A list(...) contain a number of values of the same type while an object(...) can contain a number of values of different types.
(A). False
(B). True
Answer: B
Collection Types
A collection type allows multiple values of one other type to be grouped together as a single value. The type of value within a collection is called its element type. All collection types must have an element type, which is provided as the argument to their constructor.
For example, the type list(string) means "list of strings", which is a different type than list(number), a list of numbers. All elements of a collection must always be of the same type.
The three kinds of collection type in the Terraform language are:
* list(...): a sequence of values identified by consecutive whole numbers starting with zero.
The keyword list is a shorthand for list(any), which accepts any element type as long as every element is the same type. This is for compatibility with older configurations; for new code, we recommend using the full form.
* map(...): a collection of values where each is identified by a string label.
The keyword map is a shorthand for map(any), which accepts any element type as long as every element is the same type. This is for compatibility with older configurations; for new code, we recommend using the full form.
* set(...): a collection of unique values that do not have any secondary identifiers or ordering.
https://www.terraform.io/docs/configuration/types.html
Structural Types
A structural type allows multiple values of several distinct types to be grouped together as a

single value. Structural types require a schema as an argument, to specify which types are allowed for which elements.
The two kinds of structural type in the Terraform language are:
* object(...): a collection of named attributes that each have their own type.
The schema for object types is { <KEY> = <TYPE>, <KEY> = <TYPE>, ... } - a pair of curly braces containing a comma-separated series of <KEY> = <TYPE> pairs. Values that match the object type must contain all of the specified keys, and the value for each key must match its specified type. (Values with additional keys can still match an object type, but the extra attributes are discarded during type conversion.)
* tuple(...): a sequence of elements identified by consecutive whole numbers starting with zero, where each element has its own type.
The schema for tuple types is [<TYPE>, <TYPE>, ...] - a pair of square brackets containing a comma-separated series of types. Values that match the tuple type must have exactly the same number of elements (no more and no fewer), and the value in each position must match the specified type for that position.
For example: an object type of object({ name=string, age=number }) would match a value like the following:
{
name = "John"
age = 52
}
Also, an object type of object({ id=string, cidr_block=string }) would match the object produced by a reference to an aws_vpc resource, like aws_vpc.example_vpc; although the resource has additional attributes, they would be discarded during type conversion.
Finally, a tuple type of tuple([string, number, bool]) would match a value like the following:
["a", 15, true]
https://www.terraform.io/docs/configuration/types.html

QUESTION NO: 110
You want to define multiple data disks as nested blocks inside the resource block for a virtual machine. What Terraform feature would help you define the blocks using the values in a variable?
(A). Local values
(B). Dynamic blocks
(C). Count arguments
(D). Collection functions

Answer: B

QUESTION NO: 111
What does terrafom plan do ?
(A). Create an execution plan by evaluating the difference between configuration file and state file.
(B). Performs a refresh, unless explicitly disabled, and then apply the changes that are necessary to achieve the desired state specified in the configuration files.
(C). Create an execution plan by evaluating the difference between configuration file and actual infrastructure.

(D). Checks whether the execution plan for a set of changes matches your expectations by making changes to real resources or to the state.
Answer: A

QUESTION NO: 112
What Terraform feature is shown in the example below?
(A). conditional expression
(B). local values
(C). dynamic block
(D). data source
Answer: C

QUESTION NO: 113
Which parameters does terraform import require? Choose two correct answers.
(A). Provider
(B). Path
(C). Resource address
(D). Resource ID
Answer: C,D

QUESTION NO: 114
Which of the below configuration file formats are supported by Terraform? (Select TWO)
(A). Node
(B). JSON
(C). Go
(D). YAML
(E). HCL
Answer: B,E
Terraform supports both HashiCorp Configuration Language (HCL) and JSON formats for configurations.
https://www.terraform.io/docs/configuration/

QUESTION NO: 115
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). 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.
(B). Delete the existing resources and recreate them using new a Terraform configuration so Terraform can manage them moving forward.
(C). Resources that are manually deployed in the AWS console cannot be imported by Terraform.
(D). Using terraform import, import the existing infrastructure into your Terraform state.

Answer: D
Terraform is able to import existing infrastructure. This allows us take resources we've created by some other means (i.e. via console) and bring it under Terraform management. This is a great way to slowly transition infrastructure to Terraform.
The terraform import command is used to import existing infrastructure.
To import a resource, first write a resource block for it in our configuration, establishing the name by which it will be known to Terraform.
Example:
resource "aws_instance" "import_example" {
# ...instance configuration...
}
Now terraform import can be run to attach an existing instance to this resource configuration.
$ terraform import aws_instance.import_example i-03efafa258104165f
aws_instance.import_example: Importing from ID "i-03efafa258104165f"...
aws_instance.import_example: Import complete!
Imported aws_instance (ID: i-03efafa258104165f)
aws_instance.import_example: Refreshing state... (ID: i-03efafa258104165f) Import successful!
The resources that were imported are shown above. These resources are now in your Terraform state and will henceforth be managed by Terraform.
This command locates the AWS instance with ID i-03efafa258104165f (which has been created outside Terraform) and attaches its existing settings, as described by the EC2 API, to the name aws_instance.import_example in the Terraform state.

QUESTION NO: 116
Which of these is the best practice to protect sensitive values in state files?
(A). Blockchain
(B). Secure Sockets Layer (SSL)
(C). Enhanced remote backends
(D). Signed Terraform providers
Answer: C
Use of remote backends and especially the availability of Terraform Cloud, there are now a variety of backends that will encrypt state at rest and will not store the state in cleartext on machines running. Reference: https://www.terraform.io/docs/extend/best-practices/sensitive-state.html

QUESTION NO: 117
Multiple provider instances blocks for AWS can be part of a single configuration file?
(A). False
(B). True
Answer: B
You can optionally define multiple configurations for the same provider, and select which one to use on a per-resource or per-module basis. The primary reason for this is to support multiple regions for a cloud platform; other examples include targeting multiple Docker hosts, multiple Consul hosts, etc.

To include multiple configurations for a given provider, include multiple provider blocks with the same provider name, but set the alias meta-argument to an alias name to use for each additional configuration. For example:
# The default provider configuration
provider "aws" {
region = "us-east-1"
}
# Additional provider configuration for west coast region
provider "aws" {
alias = "west"
region = "us-west-2"
}
The provider block without alias set is known as the default provider configuration. When alias is set, it creates an additional provider configuration. For providers that have no required configuration arguments, the implied empty configuration is considered to be the default provider configuration.
https://www.terraform.io/docs/configuration/providers.html#alias-multiple-provider-instances

QUESTION NO: 118
Why is it a good idea to declare the required version of a provider in a Terraform configuration file?
1. terraform
2. {
3. required_providers
4. {
5. aws = "~> 1.0"
6. }
7. }
(A). To remove older versions of the provider.
(B). To ensure that the provider version matches the version of Terraform you are using.
(C). Providers are released on a separate schedule from Terraform itself; therefore a newer version could introduce breaking changes.
(D). To match the version number of your application being deployed via Terraform.
Answer: C

QUESTION NO: 119
Which of the following represents a feature of Terraform Cloud that is NOT free to customers?
(A). Roles and Team Management
(B). WorkSpace Management
(C). Private Module Registry
(D). VCS Integration
Answer: A
Role Based Access Controls (RBAC) for controlling permissions for who has access to what configurations within an organization and it is not free to customers.

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

QUESTION NO: 120
While using generic git repository as a module source, which of the below options allows terraform to select a specific version or tag instead of selecting the HEAD.
(A). Append ref argument as
module "vpc" { source = "git::https://example.com/vpc.git?ref=v1.2.0"}
(B). Append version argument as
module "vpc" { source = "git::https://example.com/vpc.git?version=v1.2.0"}
(C). Append ref argument as
module "vpc" { source = "git::https://example.com/vpc.git#ref=v1.2.0"}
(D). By default, Terraform will clone and use the default branch (referenced by HEAD) in the selected repository and you can not override this.

Answer: A
By default, Terraform will clone and use the default branch (referenced by HEAD) in the selected repository. You can override this using the ref argument:
module "vpc" {
source = "git::https://example.com/vpc.git?ref=v1.2.0"
}
The value of the ref argument can be any reference that would be accepted by the git checkout command, including branch and tag names.
https://www.terraform.io/docs/modules/sources.html

QUESTION NO: 121
You have 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). splat expression
(C). terraform import
(D). dynamic block
Answer: D
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.
https://www.terraform.io/docs/configuration/expressions.html#dynamic-blocks

QUESTION NO: 122
A colleague has informed you that a new version of a Terraform module that your team hosts on an Amazon S3 bucket is broken. The Amazon S3 bucket has versioning enabled. Your colleague tells you to make sure you are not using the latest version in your configuration. You have the following configuration block in your code that refers to the module:
module "infranet" { source = "s3::https://s3-us-west-2.amazonaws.com/infrabucket/infra_module.zip"} What is the best way to ensure that you are not using the latest version of the module?

(A). Add a module version constraint in your configuration's backend block and specify a previous version.
(B). Add a version key to the module configuration and specify a previous version.
(C). Delete the latest version of the module in S3 to rollback to the previous version.
(D). Add a version property to the module in Terraform's state file and specify a previous version.
Answer: C
Only Terraform Registries support module versioning by using the version key, one cannot configure a previous version of the module in the configuration. Deleting the latest version of the module in S3 is the only option of the available options that ensures you won't use the latest version. You could also modify the source URL to specify a versionId URL parameter for a previous version.
https://www.terraform.io/docs/configuration/modules.html#source

QUESTION NO: 123
You have been working in a Cloud provider account that is shared with other team members. You previously used Terraform to create a load balancer that is listening on port 80. After some application changes, you updated the Terraform code to change the port to 443.
You run terraform plan and see that the execution plan shows the port changing from 80 to 443 like you intended, and step away to grab some coffee.
In the meantime, another team member manually changes the load balancer port to 443 through the Cloud provider console before you get back to your desk.
What will happen when you terraform apply upon returning to your desk?
(A). Terraform will not make any changes to the Load Balancer and will update the state file to reflect any changes made.
(B). Terraform will change the port back to 80 in your code
(C). Terraform will change the load balancer port to 80, and) then change it back to 443
(D). Terraform will fail with in error because the state file is no longer accurate
Answer: A

QUESTION NO: 124
A terraform apply can not _____ infrastructure.
(A). change
(B). destroy
(C). provision
(D). import
Answer: A

QUESTION NO: 125
What is the name assigned by Terraform to reference this resource?

```
mainresource “google_compute_instance” “main” {
      name = “test”
}
```


(A). compute_instance
(B). main

(C). google
(D). test
Answer: B

QUESTION NO: 126
Environment variables can be used to set variables. The environment variables must be in the format "____"_<variablename>. Select the correct prefix string from the following list.
(A). TF_CLI_ARGS
(B). TF_VAR
(C). TF_VAR_
(D). TF_VAR_ENV
Answer: C
Environment variables can be used to set variables. The environment variables must be in the format TF_VAR_name and this will be checked last for a value. For example:
export TF_VAR_region=us-west-1
export TF_VAR_ami=ami-049d8641
export TF_VAR_alist='[1,2,3]'
export TF_VAR_amap='{ foo = "bar", baz = "qux" }'
https://www.terraform.io/docs/commands/environment-variables.html

QUESTION NO: 127
What does terraform refresh modify?
(A). Your cloud infrastructure
(B). Your Terraform plan
(C). Your state file
(D). Your Terraform configuration
Answer: C

QUESTION NO: 128
Which of the following is not a key principle of infrastructure as code?
(A). Versioned infrastructure
(B). Golden images
(C). Idempotence
(D). Self-describing infrastructure
Answer: A,B,D

QUESTION NO: 129
You have written a terraform IaC script which was working till yesterday , but is giving some vague error from today , which you are unable to understand . You want more detailed logs that could potentially help you troubleshoot the issue , and understand the root cause. What can you do to enable this setting? Please note , you are using terraform OSS.
(A). Terraform OSS can push all its logs to a syslog endpoint. As such, you have to set up the syslog sink, and enable TF_LOG_PATH env variable to the syslog endpoint and all logs will automatically start streaming.
(B). Detailed logs are not available in terraform OSS, except the crash message. You need to

upgrade to terraform enterprise for this point.
(C). Enable the TF_LOG_PATH to the log sink file location, and logging output will automatically be stored there.
(D). Enable TF_LOG to the log level DEBUG, and then set TF_LOG_PATH to the log sink file location. Terraform debug logs will be dumped to the sink path, even in terraform OSS.
Answer: D
Terraform has detailed logs which can be enabled by setting the TF_LOG environment variable to any value. This will cause detailed logs to appear on stderr.
You can set TF_LOG to one of the log levels TRACE, DEBUG, INFO, WARN or ERROR to change the verbosity of the logs. TRACE is the most verbose and it is the default if TF_LOG is set to something other than a log level name.
To persist logged output you can set TF_LOG_PATH in order to force the log to always be appended to a specific file when logging is enabled. Note that even when TF_LOG_PATH is set, TF_LOG must be set in order for any logging to be enabled.

QUESTION NO: 130
Which of the below options is the equivalent Terraform 0.12 version of the snippet which is written in Terraform 0.11?
"${var.instance_id}"
(A). variable.instance_id
(B). var.instance_ids
(C). var.instance_id
(D). None of the above
Answer: C

QUESTION NO: 131
When using remote state, state is only ever held in memory when used by Terraform.
(A). False
(B). True
Answer: B

QUESTION NO: 132
Which of the below command will upgrade the provider version to the latest acceptable one?
(A). terraform plan upgrade
(B). terraform provider -upgrade
(C). terraform init -upgrade
(D). terraform init -update
Answer: C
To upgrade to the latest acceptable version of each provider, run terraform init -upgrade. This command also upgrades to the latest versions of all Terraform modules.
https://www.terraform.io/docs/configuration/providers.html

QUESTION NO: 133
True or False? terraform init cannot automatically download Community providers.
(A). False

(B). True
Answer: B

QUESTION NO: 134
The canonical format may change in minor ways between Terraform versions, so after upgrading Terraform it is recommended to proactively run.
(A). terraform fmt
(B). terraform init
(C). terraform validate
(D). terraform plan
Answer: A

QUESTION NO: 135
Which Terraform command will check and report errors within modules, attribute names, and value types to make sure they are syntactically valid and internally consistent?
(A). terraform validate
(B). terraform format
(C). terraform fmt
(D). terraform show
Answer: A
The terraform validate command validates the configuration files in a directory, referring only to the configuration and not accessing any remote services such as remote state, provider APIs, etc.
Validate runs checks that verify whether a configuration is syntactically valid and internally consistent, regardless of any provided variables or existing state. It is thus primarily useful for general verification of reusable modules, including the correctness of attribute names and value types.
It is safe to run this command automatically, for example as a post-save check in a text editor or as a test step for a re-usable module in a CI system.

QUESTION NO: 136
Mary has created a database instance in AWS and for ease of use is outputting the value of the database password with the following code:
1. output "db_password"
2. {
3. value = local.db_password
4. }
Mary wants to hide the output value in the CLI after terraform apply? What is the best way?
(A). Use secure parameter
(B). Use sensitive parameter
(C). Use cryptographic hash
(D). Encrypt the value using encrypt() function
Answer: B

QUESTION NO: 137

Why would you use the terraform taint command?
(A). When you want to force Terraform to destroy a resource on the next apply
(B). When you want to force Terraform to destroy and recreate a resource on the next apply
(C). When you want Terraform to ignore a resource on the next apply
(D). When you want Terraform to destroy all the infrastructure in your workspace
Answer: B
The terraform taint command manually marks a Terraform-managed resource as tainted, forcing it to be destroyed and recreated on the next apply.

QUESTION NO: 138
terraform init initializes a sample main.tf file in the current directory.
(A). True
(B). False
Answer: B

QUESTION NO: 139
One remote backend configuration always maps to a single remote workspace.
(A). True
(B). False
Answer: A

QUESTION NO: 140
You have a Terraform configuration file where a variable itemNum is defined as follows:
variable "itemNum" { default = 3}
You also have a defined the following environment variables in your shell: TF_itemNum =6, TF_VAR_itemNum =9. You also have a terraform.tfvars file with the following contents itemNum = 7 When you run the following apply command, what is the value assigned to the itemNum variable?
terraform apply -var itemNum =4
(A). 10
(B). 6
(C). 1
(D). 4
(E). 3

Answer: D
The -var and -var-file methods of assigning variables have the highest precedence.
https://www.terraform.io/docs/configuration/variables.html

QUESTION NO: 141
When TF_LOG_PATH is set, TF_LOG must be set in order for any logging to be enabled.
(A). False
(B). True
Answer: B
TF_LOG_PATH specifies where the log should persist its output to. Note that even when TF_LOG_PATH is set, TF_LOG must be set in order for any logging to be enabled.
For example, to always write the log to the directory you're currently running terraform from:

export TF_LOG_PATH=./terraform.log
export TF_LOG=TRACE

QUESTION NO: 142
You have a simple Terraform configuration containing one virtual machine (VM) in a cloud provider. You run terraform apply and the VM is created successfully.
What will happen if you delete the VM using the cloud provider console, and run terraform apply again without changing any Terraform code?
(A). Terraform will remove the VM from state file
(B). Terraform will report an error
(C). Terraform will not make any changes
(D). Terraform will recreate the VM
Answer: C

QUESTION NO: 143
You would like to reuse the same Terraform configuration for your development and production environments with a different state file for each.
Which command would you use?
(A). terraform import
(B). terraform workspace
(C). terraform state
(D). terraform init
Answer: D

QUESTION NO: 144
Which task does teraform ini- not perform?
(A). Sources any modules and copies the configuration locally
(B). Validates all required variables are present
(C). Connects to the backend
(D). Sources all providers present in the configuration and ensures they are downloaded and available locally
Answer: B

QUESTION NO: 145
Your manager has instructed you to start using terraform for the entire infra provisioning of the application stack. There are 4 environments - DEV , QA , UAT , and PROD. The application team has asked for complete segregation between these environments including the backend , state , and also configurations ,since there will be unique resources in different environments . What is the possible way to structure the terraform code to facilitate that.
(A). Completely separate the working directories , keep one for each environment . For each working directory , maintain a separate configuration file , variables file , and map to a different backend.
(B). Completely separate the working directories , keep one for each environment . For each working directory , maintain a separate configuration file , variables file , and map to the same backend.
(C). Implement terraform workspaces , and map each environment with one workspace.

(D). Enable remote backend storage . Configure 4 different backend storages , one for each environment.

Answer: A

In particular, organizations commonly want to create a strong separation between multiple deployments of the same infrastructure serving different development stages (e.g. staging vs. production) or different internal teams. In this case, the backend used for each deployment often belongs to that deployment, with different credentials and access controls. Named workspaces are not a suitable isolation mechanism for this scenario.
https://www.terraform.io/docs/state/workspaces.html

QUESTION NO: 146

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?

(A). terraform delete
(B). terraform plan
(C). terraform init
(D). terraform apply

Answer: C

QUESTION NO: 147

Which of the following is not a valid Terraform collection type?

(A). list
(B). map
(C). tree
(D). set

Answer: C

QUESTION NO: 148

What is the default backend for Terraform?

(A). consul
(B). gcs
(C). local
(D). etcd

Answer: C

By default, Terraform uses the "local" backend, which is the normal behavior of Terraform you're used to.
https://www.terraform.io/docs/backends/index.html

QUESTION NO: 149

Which of the following can you do with terraform plan? Choose two correct answers.

(A). View the execution plan and check if the changes match your expectations
(B). Schedule Terraform to run at a planned time in the future
(C). Execute a plan in a different workspace
(D). Save a generated execution plan to apply later

Answer: A,D

QUESTION NO: 150
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

Answer: C,D
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 NO: 151
You run a local-exec provisioner in a null resource called null_resource.run_script and realize that you need to rerun the script.
Which of the following commands would you use first?
(A). terraform taint null_resource.run_script
(B). terraform apply -target=null_resource.run_script
(C). terraform validate null_resource.run_script
(D). terraform plan -target=null_resource.run_script
Answer: A

QUESTION NO: 152
The following is a snippet from a Terraform configuration file:
Which, when validated, results in the following error:
Fill in the blank in the error message with the correct string from the list below.
(A). version
(B). multi
(C). label
(D). alias
Answer: D
https://www.terraform.io/docs/configuration/providers.html#alias-multiple-providerinstances

QUESTION NO: 153
Every region in AWS has a different AMI ID for Linux and these are keep on changing. What is the best approach to create the EC2 instances that can deal with different AMI IDs based on regions?
(A). Use data source aws_ami.
(B). Create a map of region to ami id.
(C). Create different configuration file for different region.
(D). None of the above
Answer: A
https://www.terraform.io/docs/configuration/data-sources.html

QUESTION NO: 154
What advantage does an operations team that uses infrastructure as code have?
(A). The ability to delete infrastructure
(B). The ability to reuse best practice configurations and settings
(C). The ability to autoscale a group of servers
(D). The ability to update existing infrastructure
Answer: B

QUESTION NO: 155
You have recently started a new job at a retailer as an engineer. As part of this new role, you have been tasked with evaluating multiple outages that occurred during peak shopping time during the holiday season. Your investigation found that the team is manually deploying new compute instances and configuring each compute instance manually. This has led to inconsistent configuration between each compute instance.
How would you solve this using infrastructure as code?
(A). Implement a ticketing workflow that makes engineers submit a ticket before manually provisioning and configuring a resource
(B). Implement a checklist that engineers can follow when configuring compute instances
(C). Replace the compute instance type with a larger version to reduce the number of required deployments
(D). Implement a provisioning pipeline that deploys infrastructure configurations committed to your version control system following code reviews
Answer: A

QUESTION NO: 156
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?
(A). Terraform enterprise
(B). Consul
(C). Github
(D). S3
(E). Artifactory
Answer: C
Github is not a supported backend type.
https://www.terraform.io/docs/backends/types/index.html

QUESTION NO: 157
Terraform works well in Windows but a Windows server is required.
(A). False
(B). True
Answer: A
You may see this
Terraform does not require GO language to be installed as a prerequisite and it does not require a Windows Server as well.

QUESTION NO: 158
Which argument(s) is (are) required when declaring a Terraform variable?
(A). type
(B). default
(C). description
(D). All of the above
(E). None of the above
Answer: B
The variable declaration can also include a default argument.

QUESTION NO: 159
By default, a defined provisioner is a creation-time provisioner.
(A). True
(B). False
Answer: A
https://www.terraform.io/docs/provisioners/index.html

QUESTION NO: 160
ABC Enterprise has recently tied up with multiple small organizations for exchanging database information. Due to this, the firewall rules are increasing and are more than 100 rules. This is leading firewall configuration file that is difficult to manage. What is the way this type of configuration can be managed easily?
(A). Terraform Backends
(B). Terraform Functions
(C). Dynamic Blocks
(D). Terraform Expression

Answer: C

QUESTION NO: 161
Terraform providers are always installed from the Internet.
(A). True
(B). False
Answer: B
Terraform configurations must declare which providers they require, so that Terraform can install and use them.

QUESTION NO: 162
You have multiple team members collaborating on infrastructure as code (IaC) using Terraform, and want to apply formatting standards for readability.
How can you format Terraform HCL (HashiCorp Configuration Language) code according to standard Terraform style convention?
(A). Run the terraform fmt command during the code linting phase of your CI/CD process
(B). Designate one person in each team to review and format everyone's code
(C). Manually apply two spaces indentation and align equal sign "=" characters in every Terraform file (*.tf)

(D). Write a shell script to transform Terraform files using tools such as AWK, Python, and sed

Answer: C

Indent two spaces for each nesting level.

When multiple arguments with single-line values appear on consecutive lines at the same nesting level, align their equals signs.

QUESTION NO: 163

terraform refresh command will not modify infrastructure, but does modify the state file.

(A). True

(B). False

Answer: A

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. This does not modify infrastructure, but does modify the state file.

https://www.terraform.io/docs/commands/refresh.html

QUESTION NO: 164

Terraform import command can import resources into modules as well directly into the root of your state.

(A). True

(B). False

Answer: A

Import will find the existing resource from ID and import it into your Terraform state at the given ADDRESS. ADDRESS must be a valid resource address. Because any resource address is valid, the import command can import resources into modules as well directly into the root of your state.

Terraform is able to import existing infrastructure. This allows us take resources we've created by some other means (i.e. via console) and bring it under Terraform management. This is a great way to slowly transition infrastructure to Terraform.

The terraform import command is used to import existing infrastructure.

To import a resource, first write a resource block for it in our configuration, establishing the name by which it will be known to Terraform. For example:

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

Now terraform import can be run to attach an existing instance to this resource configuration:

```
$ terraform import aws_instance.import_example i-03efafa258104165f
aws_instance.import_example: Importing from ID "i-03efafa258104165f"...
aws_instance.import_example: Import complete!
Imported aws_instance (ID: i-03efafa258104165f)
aws_instance.import_example: Refreshing state... (ID: i-03efafa258104165f) Import successful!
```

The resources that were imported are shown above. These resources are now in your

Terraform state and will henceforth be managed by Terraform.
This command locates the AWS instance with ID i-03efafa258104165f (which has been created outside Terraform) and attaches its existing settings, as described by the EC2 API, to the name aws_instance.import_example in the Terraform state.
As a result of the above command, the resource is recorded in the state file. We can now run terraform plan to see how the configuration compares to the imported resource, and make any adjustments to the configuration to align with the current (or desired) state of the imported object.
https://www.terraform.io/docs/commands/import.html

QUESTION NO: 165
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
Answer: D
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 NO: 166
Which provisioner invokes a process on the resource created by Terraform?
(A). remote-exec
(B). null-exec
(C). local-exec
(D). file
Answer: A
The remote-exec provisioner invokes a script on a remote resource after it is created.
Reference: https://www.terraform.io/docs/language/resources/provisioners/remote-exec.html

QUESTION NO: 167
You cannot publish your own modules on the Terraform Registry.
(A). False
(B). True
Answer: A
https://www.terraform.io/docs/registry/modules/publish.html

QUESTION NO: 168
Which of the following value will be accepted for my_var?
1. variable "my_var"
2. {
3. type = string
4. }
(A). 15
(B). "15"
(C). Both A and B

(D). None of the above

Answer: C

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.

Example

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

Where possible, Terraform automatically converts values from one type to another in order to produce the expected type. If this isn't possible, Terraform will produce a type mismatch error and you must update the configuration with a more suitable expression.

https://www.terraform.io/docs/configuration/expressions.html#type-conversion

QUESTION NO: 169

Matt wants to import a manually created EC2 instance into terraform so that he can manage the EC2 instance through terraform going forward. He has written the configuration file of the EC2 instance before importing it to Terraform. Following is the code:

resource "aws_instance" "matt_ec2" { ami = "ami-bg2640de" instance_type = "t2.micro" vpc_security_group_ids = ["sg-6ae7d613", "sg-53370035"] key_name = "mysecret" subnet_id = "subnet-9e3cfbc5" } The instance id of that EC2 instance is i-0260835eb7e9bd40 How he can import data of EC2 to state file?

(A). terraform import aws_instance.id = i-0260835eb7e9bd40

(B). terraform import i-0260835eb7e9bd40

(C). terraform import aws_instance.i-0260835eb7e9bd40

(D). terraform import aws_instance.matt_ec2 i-0260835eb7e9bd40

Answer: D

https://www.terraform.io/docs/import/usage.html

QUESTION NO: 170

As a member of the operations team, you need to run a script on a virtual machine created by Terraform. Which provisioned is best to use in your Terraform code?

(A). local-exec

(B). file

(C). null-exec

(D). remote-exec

Answer: D

QUESTION NO: 171

Which of the following command can be used to view the specified version constraints for all providers used in the current configuration.

(A). terraform providers

(B). terraform state show

(C). terraform provider

(D). terraform plan

Answer: A
Use the terraform providers command to view the specified version constraints for all providers used in the current configuration.
https://www.terraform.io/docs/configuration/providers.html

QUESTION NO: 172
How is terraform import run?
(A). As a part of terraform init
(B). As a part of terraform plan
(C). As a part of terraform refresh
(D). By an explicit call
(E). All of the above
Answer: B

QUESTION NO: 173
A user runs terraform init on their RHEL based server and per the output, two provider plugins are downloaded: $ terraform init Initializing the backend...
Initializing provider plugins...
- Checking for available provider plugins...
- Downloading plugin for provider "aws" (hashicorp/aws) 2.44.0...
- Downloading plugin for provider "random" (hashicorp/random) 2.2.1...
:
Terraform has been successfully initialized! Where are these plugins downloaded to?
(A). The .terraform.plugins directory in the directory terraform init was executed in.
(B). The .terraform/plugins directory in the directory terraform init was executed in.
(C). /etc/terraform/plugins
(D). The .terraform.d directory in the directory terraform init was executed in.
Answer: B

QUESTION NO: 174
What type of block is used to construct a collection of nested configuration blocks?
(A). for_each
(B). repeated
(C). nesting
(D). dynamic
Answer: D

QUESTION NO: 175
As a developer, you want to ensure your plugins are up to date with the latest versions. Which Terraform command should you use?
(A). terreform providers- upgrade
(B). terreform apply -upgrade
(C). terreform refresh -upgrade
(D). terreformn lnit -upgrade
Answer: D

QUESTION NO: 176
Which one of the following command will rewrite Terraform configuration files to a canonical format and style.
(A). terraform graph -h
(B). terraform init
(C). terraform graph
(D). terraform fmt
Answer: D
The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style. This command applies a subset of the Terrra Terraform language style conventions, along with other minor adjustments for readability.

QUESTION NO: 177
Using the terraform state rm command against a resource will destroy it.
(A). True
(B). False
Answer: B

QUESTION NO: 178
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). Run terraform refresh to ensure that the state file has the latest information for existing resources.
(B). Update the configuration file to include the new resources.
(C). Shut down or stop using the resources being imported so no changes are inadvertently missed.
(D). Modify the Terraform state file to add the new resources.
Answer: B
The current implementation of Terraform import can only import resources into the state. It does not generate configuration. A future version of Terraform will also generate configuration.
Because of this, prior to running terraform import it is necessary to write manually a resource configuration block for the resource, to which the imported object will be mapped.
The terraform import command is used to import existing infrastructure.
To import a resource, first write a resource block for it in our configuration, establishing the name by which it will be known to Terraform.
Example:
resource "aws_instance" "import_example" {
# ...instance configuration...
}
Now terraform import can be run to attach an existing instance to this resource configuration.
$ terraform import aws_instance.import_example i-03efafa258104165f
aws_instance.import_example: Importing from ID "i-03efafa258104165f"...
aws_instance.import_example: Import complete!

Imported aws_instance (ID: i-03efafa258104165f)
aws_instance.import_example: Refreshing state... (ID: i-03efafa258104165f) Import successful!
The resources that were imported are shown above. These resources are now in your Terraform state and will henceforth be managed by Terraform.
This command locates the AWS instance with ID i-03efafa258104165f (which has been created outside Terraform) and attaches its existing settings, as described by the EC2 API, to the name aws_instance.import_example in the Terraform state.

QUESTION NO: 179
What command does Terraform require the first time you run it within a configuration directory?
(A). terraform import
(B). terraform init
(C). terraform plan
(D). terraform workspace
Answer: B
terraform init command is used to initialize a working directory containing Terraform configuration files. Reference: https://www.terraform.io/docs/cli/commands/init.html

QUESTION NO: 180
If a module declares a variable with a default, that variable must also be defined within the module.
(A). True
(B). False

Answer: B

QUESTION NO: 181
Which of the following best describes a Terraform provider?
(A). A plugin that Terraform uses to translate the API interactions with the service or provider.
(B). Serves as a parameter for a Terraform module that allows a module to be customized.
(C). Describes an infrastructure object, such as a virtual network, compute instance, or other components.
(D). A container for multiple resources that are used together.
Answer: A
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).
https://www.terraform.io/docs/providers/index.html

QUESTION NO: 182
If you delete a remote backend from the configuration, will you need to rebuild your state files locally?
(A). False
(B). True

Answer: A
You can change your backend configuration at any time. You can change both the configuration itself as well as the type of backend (for example from "consul" to "s3"). Terraform will automatically detect any changes in your configuration and request a reinitialization. As part of the reinitialization process, Terraform will ask if you'd like to migrate your existing state to the new configuration. This allows you to easily switch from one backend to another.
https://www.terraform.io/docs/backends/config.html#changing-configuration

QUESTION NO: 183
Which task does terraform init not perform?
(A). Sources all providers present in the configuration and ensures they are downloaded and available locally
(B). Connects to the backend
(C). Sources any modules and copies the configuration locally
(D). Validates all required variables are present
Answer: D

QUESTION NO: 184
When using constraint expressions to signify a version of a provider, which of the following are valid provider versions that satisfy the expression found in the following code snippet: (select two)
1. terraform
2. {
3. required_providers
4. {
5. aws = "~> 1.2.0"
6. }
7. }
(A). 1.3.1
(B). 1.2.3
(C). 1.2.9
(D). 1.3.0
Answer: B,C
As your Terraform usage becomes more advanced, there are some cases where you may need to modify the Terraform state. Rather than modify the state directly, the terraform state commands can be used in many cases instead. This command is a nested subcommand, meaning that it has further subcommands.
https://www.terraform.io/docs/commands/state/index.html

QUESTION NO: 185
You need to specify a dependency manually.
What resource meta-parameter can you use to make sure Terraform respects the dependency?
Type your answer in the field provided. The text field is not case-sensitive and all variations

of the correct answer are accepted.

Answer:

thelocalfiledatasource

QUESTION NO: 186

Provider dependencies are created in several different ways. Select the valid provider dependencies from the following list: (select three)

(A). Explicit use of a provider block in configuration, optionally including a version constraint.

(B). Use of any resource belonging to a particular provider in a resource or data block in configuration.

(C). Existence of any resource instance belonging to a particular provider in the current state.

(D). Existence of any provider plugins found locally in the working directory.

Answer: A,B,C

The existence of a provider plugin found locally in the working directory does not itself create a provider dependency. The plugin can exist without any reference to it in the terraform configuration. https://www.terraform.io/docs/commands/providers.html

QUESTION NO: 187

You are using a terraform operation that writes state. Unfortunately automatic state unlocking has failed for that operation. Which of the below commands can be used to remove the already acquired lock on the state?

(A). terraform unlock

(B). terraform force-unlock

(C). terraform state unlock

(D). None of the above

Answer: B

Command: force-unlock

Manually unlock the state for the defined configuration.

This will not modify your infrastructure. This command removes the lock on the state for the current configuration. The behavior of this lock is dependent on the backend being used. Local state files cannot be unlocked by another process.

https://www.terraform.io/docs/commands/force-unlock.html

https://www.terraform.io/docs/state/locking.html

Terraform has a force-unlock command to manually unlock the state if unlocking failed.

If you unlock the state when someone else is holding the lock it could cause multiple writers. Force unlock should only be used to unlock your own lock in the situation where automatic unlocking failed.

QUESTION NO: 188

You cannot publish your own modules on the Terraform Registry.

(A). False

(B). True

Answer: A

Anyone can publish and share modules on the Terraform Registry.

https://www.terraform.io/docs/registry/modules/publish.html

QUESTION NO: 189
Which of the following actions are performed during a terraform init?
(A). Initializes downloaded and/or installed providers
(B). Initializes the backend configuration
(C). Provisions the declared resources in your configuration
(D). Download the declared providers which are supported by HashiCorp
Answer: A,B,D
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.
This command is always safe to run multiple times, to bring the working directory up to date with changes in the configuration. Though subsequent runs may give errors, this command will never delete your existing configuration or state.
terraform init command does -
* Copy a Source Module
* Backend Initialization
* Child Module Installation
* Plugin Installation
https://www.terraform.io/docs/commands/init.html

QUESTION NO: 190
What kind of resource dependency is stored in terraform.tfstate file?
(A). Both implicit and explicit dependencies are stored in state file.
(B). Only explicit dependencies are stored in state file.
(C). Only implicit dependencies are stored in state file.
(D). No dependency information is stored in state file.

Answer: A
Terraform state captures all dependency information, both implicit and explicit. One purpose for state is to determine the proper order to destroy resources. When resources are created all of their dependency information is stored in the state. If you destroy a resource with dependencies, Terraform can still determine the correct destroy order for all other resources because the dependencies are stored in the state.
https://www.terraform.io/docs/state/purpose.html#metadata

QUESTION NO: 191
You have provisioned some virtual machines (VMs) on Google Cloud Platform (GCP) using the gcloud command line tool. However, you are standardizing with Terraform and want to manage these VMs using Terraform instead.
What are the two things you must do to achieve this? (Choose two.)
(A). Provision new VMs using Terraform with the same VM names
(B). Use the terraform import command for the existing VMs
(C). Write Terraform configuration for the existing VMs
(D). Run the terraform import-gcp command
Answer: B,D

The terraform import command is used to import existing infrastructure. Import existing Google Cloud resources into Terraform with Terraformer.

QUESTION NO: 192
Terraform will sync all resources in state by default for every plan and apply, hence for larger infrastructures this can slow down terraform plan and terraform apply commands?
(A). False
(B). True
Answer: B
For small infrastructures, Terraform can query your providers and sync the latest attributes from all your resources. This is the default behavior of Terraform: for every plan and apply, Terraform will sync all resources in your state.
For larger infrastructures, querying every resource is too 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. On top of this, cloud providers almost always have API rate limiting so Terraform can only request a certain number of resources in a period of time. Larger users of Terraform make heavy use of the -refresh=false flag as well as the -target flag in order to work around this. In these scenarios, the cached state is treated as the record of truth.
https://www.terraform.io/docs/state/purpose.html

QUESTION NO: 193
Terraform variable names are saved in the state file.
(A). True
(B). False
Answer: B

QUESTION NO: 194
Given the Terraform configuration below, in which order will the resources be created?
(A). Larger image
(B). resources will be created simultaneously
(C). aws_eip will be created first aws_instance will be created second
(D). aws_instance will be created first aws_eip will be created second
Answer: D
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 NO: 195
While attempting to deploy resources into your cloud provider using Terraform. you begin to see some odd behavior and experience sluggish responses. In order to troubleshoot you decide to turn on Terraform debugging. Which environment variables must be configured to make Terraform's logging more verbose?
(A). TF_10G_PATM
(B). TF_LOG
(C). TF_10G_LEVEL
(D). TF.LOG.FUE

Answer: B

QUESTION NO: 196
What happens when a terraform apply command is executed?
(A). Creates the execution plan for the deployment of resources.
(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). Reconciles the state Terraform knows about with the real-world infrastructure.
Answer: B
The terraform apply command is used to apply the changes required to reach the desired state of the configuration, or the pre-determined set of actions generated by a terraform plan execution plan.
https://www.terraform.io/docs/commands/apply.html

QUESTION NO: 197
A single terraform resource file that defines an aws_instance resource can simply be renamed to vsphere_virtual_machine in order to switch cloud providers.
(A). True
(B). False
Answer: B
Every provider has its own required and allowed declarations none of which match between cloud providers.

QUESTION NO: 198
When writing Terraform code, HashiCorp recommends that you use how many spaces between each nesting level?
(A). 0
(B). 1
(C). 2
(D). 4
Answer: C
The Terraform parser allows you some flexibility in how you lay out the elements in your configuration files, but the Terraform language also has some idiomatic style conventions which we recommend users always follow for consistency between files and modules written by different teams. Automatic source code formatting tools may apply these conventions automatically.
Indent two spaces for each nesting level.
When multiple arguments with single-line values appear on consecutive lines at the same nesting level, align their equals signs:
ami = "abc123"
instance_type = "t2.micro"
When both arguments and blocks appear together inside a block body, place all of the arguments together at the top and then place nested blocks below them. Use one blank line to separate the arguments from the blocks.

Use empty lines to separate logical groups of arguments within a block.
For blocks that contain both arguments and "meta-arguments" (as defined by the Terraform language semantics), list meta-arguments first and separate them from other arguments with one blank line. Place meta-argument blocks last and separate them from other blocks with one blank line.

```
resource "aws_instance" "example" {
count = 2 # meta-argument first
ami = "abc123"
instance_type = "t2.micro"
network_interface {
# ...
}
lifecycle { # meta-argument block last
create_before_destroy = true
}
}
```

Top-level blocks should always be separated from one another by one blank line. Nested blocks should also be separated by blank lines, except when grouping together related blocks of the same type (like multiple provisioner blocks in a resource).
Avoid separating multiple blocks of the same type with other blocks of a different type, unless the block types are defined by semantics to form a family. (For example: root_block_device, ebs_block_device and ephemeral_block_device on aws_instance form a family of block types describing AWS block devices, and can therefore be grouped together and mixed.)

QUESTION NO: 199
Jim has created several AWS resources from a single terraform configuration file. Someone from his team has manually modified one of the EC2 instance.
Now to discard the manual change, Jim wants to destroy and recreate the EC2 instance.
What is the best way to do it?
(A). terraform recreate
(B). terraform taint
(C). terraform destroy
(D). terraform refresh
Answer: B
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 and the next apply will implement this change.
Forcing the recreation of a resource is useful when you want a certain side effect of recreation that is not visible in the attributes of a resource. For example: re-running provisioners will cause the node to be different or rebooting the machine from a base image will cause new startup scripts to run.
Note that tainting a resource for recreation may affect resources that depend on the newly tainted resource. For example, a DNS resource that uses the IP address of a server may

need to be modified to reflect the potentially new IP address of a tainted server. The plan command will show this if this is the case.
This example will taint a single resource:
$ terraform taint aws_security_group.allow_all
The resource aws_security_group.allow_all in the module root has been marked as tainted.
https://www.terraform.io/docs/commands/taint.html

QUESTION NO: 200
The Terraform CLI will print output values from a child module after running terraform apply.
(A). True
(B). False

Answer: A

QUESTION NO: 201
Which flag would you add to terraform plan to save the execution plan to a file?
Type your answer in the field provided. The text field is not case-sensitive and all variations of the correct answer are accepted.
Answer:
outFILENAME

QUESTION NO: 202
You have already set TF_LOG = DEBUG to enable debug log. Now you want to always write the log to the directory you're currently running terraform from. what should you do to achieve this.
(A). Run the command export TF_LOG_FILE=./terraform.log.
(B). Run the command export TF_LOG_PATH=./terraform.log.
(C). Run the command export TF_DEBUG_PATH=./terraform.log.
(D). No explicit action required. Terraform will take care of this as you have enable TF_LOG.
Answer: B
https://www.terraform.io/docs/commands/environment-variables.html

QUESTION NO: 203
The current implementation of Terraform import can only import resources into the state. It does not generate configuration.
(A). False
(B). True
Answer: B
The current implementation of Terraform import can only import resources into the state. It does not generate configuration. A future version of Terraform will also generate configuration.
Because of this, prior to running terraform import it is necessary to write manually a resource configuration block for the resource, to which the imported object will be mapped.
While this may seem tedious, it still gives Terraform users an avenue for importing existing resources.
https://www.terraform.io/docs/import/index.html#currently-state-only

QUESTION NO: 204
terraform validate reports HCL syntax errors.
(A). True
(B). False
Answer: A

QUESTION NO: 205
Which of the following challenges would Terraform be a candidate for solving? (Select THREE)
(A). Enable self-service infrastructure to allocate resources on your proprietary private cloud.
(B). Reduce the number of workflows needed for managing infrastructure across each of the companies public and private clouds.
(C). Utilize a single tool for all of the infrastructure and configuration management needs.
(D). Have a single interoperable tool to manage the variety of services including GitHub repositories, MySQL database, and Kubernetes clusters.
Answer: A,B,D

QUESTION NO: 206
In order to reduce the time it takes to provision resources, Terraform uses parallelism. By default, how many resources will Terraform provision concurrently?
(A). 5
(B). 50
(C). 10
(D). 20
Answer: C

QUESTION NO: 207
Which of the below datatype is not supported by Terraform.
(A). Array
(B). List
(C). Object
(D). Map
Answer: A

QUESTION NO: 208
In the example below, the depends_on argument creates what type of dependency?
(A). implicit dependency
(B). internal dependency
(C). explicit dependency
(D). non-dependency resource
Answer: C

QUESTION NO: 209
In terraform, most resource dependencies are handled automatically. Which of the following statements describes best how terraform resource dependencies are handled?

(A). Resource dependencies are identified and maintained in a file called resource.dependencies. Each terraform provider is required to maintain a list of all resource dependencies for the provider and it's included with the plugin during initialization when terraform init is executed. The file is located in the terraform.d folder.
(B). The terraform binary contains a built-in reference map of all defined Terraform resource dependencies. Updates to this dependency map are reflected in terraform versions. To ensure you are working with the latest resource dependency map you much be running the latest version of Terraform.
(C). Resource dependencies are handled automatically by the depends_on meta_argument, which is set to true by default.
(D). Terraform analyses any expressions within a resource block to find references to other objects, and treats those references as implicit ordering requirements when creating, updating, or destroying resources.
Answer: D
https://www.terraform.io/docs/configuration/resources.html

QUESTION NO: 210
Which of the following Terraform commands will automatically refresh the state unless supplied with additional flags or arguments? Choose TWO correct answers.
(A). terraform state
(B). terraform apply
(C). terraform plan
(D). terraform validate
(E). terraform output

Answer: B,C

QUESTION NO: 211
Which of the following value will be accepted for var1?
variable "var1" {
type = string
}
(A). None of the above
(B). Both A and B
(C). "5"
(D). 5
Answer: B
Terraform automatically converts number and bool values to strings when needed.

QUESTION NO: 212
You have declared a variable name my_var in terraform configuration without a value associated with it.
variable my_var {}
After running terraform plan it will show an error as variable is not defined.
(A). True
(B). False
Answer: B

Input variables are usually defined by stating a name, type and a default value. However, the type and default values are not strictly necessary. Terraform can deduct the type of the variable from the default or input value.
Variables can be predetermined in a file or included in the command-line options. As such, the simplest variable is just a name while the type and value are selected based on the input.

```
variable "variable_name" {}
terraform apply -var variable_name="value"
```

The input variables, like the one above, use a couple of different types: strings, lists, maps, and boolean. Here are some examples of how each type are defined and used.
String
Strings mark a single value per structure and are commonly used to simplify and make complicated values more user-friendly. Below is an example of a string variable definition.

```
variable "template" {
type = string
default = "01000000-0000-4000-8000-000030080200"
}
```

A string variable can then be used in resource plans. Surrounded by double quotes, string variables are a simple substitution such as the example underneath.

```
storage = var.template
```

List
Another type of Terraform variables lists. They work much like a numbered catalogue of values. Each value can be called by their corresponding index in the list. Here is an example of a list variable definition.

```
variable "users" {
type = list
default = ["root", "user1", "user2"]
}
```

Lists can be used in the resource plans similarly to strings, but you'll also need to denote the index of the value you are looking for.

```
username = var.users[0]
```

Map
Maps are a collection of string keys and string values. These can be useful for selecting values based on predefined parameters such as the server configuration by the monthly price.

```
variable "plans" {
type = map
default = {
"5USD" = "1xCPU-1GB"
"10USD" = "1xCPU-2GB"
"20USD" = "2xCPU-4GB"
}
}
```

You can access the right value by using the matching key. For example, the variable below would set the plan to "1xCPU-1GB".

```
plan = var.plans["5USD"]
```

The values matching to their keys can also be used to look up information in other maps. For example, underneath is a shortlist of plans and their corresponding storage sizes.

```
variable "storage_sizes" {
type = map
default = {
"1xCPU-1GB" = "25"
"1xCPU-2GB" = "50"
"2xCPU-4GB" = "80"
}
}
```

These can then be used to find the right storage size based on the monthly price as defined in the previous example.

```
size = lookup(var.storage_sizes, var.plans["5USD"])
```

Boolean

The last of the available variable type is boolean. They give the option to employ simple true or false values. For example, you might wish to have a variable that decides when to generate the root user password on a new deployment.

```
variable "set_password" {
default = false
}
```

The above example boolean can be used similarly to a string variable by simply marking down the correct variable.

```
create_password = var.set_password
```

By default, the value is set to false in this example. However, you can overwrite the variable at deployment by assigning a different value in a command-line variable.

```
terraform apply -var set_password="true"
```

QUESTION NO: 213

Which of the below terraform commands do not run terraform refresh implicitly before taking actual action of the command?

(A). terraform apply
(B). terraform destroy
(C). terraform init
(D). terraform import
(E). terraform plan

Answer: C,D

https://www.terraform.io/docs/commands/refresh.html

QUESTION NO: 214

Complete the following sentence:

For local state, the workspaces are stored directly in a _____.

(A). a file called terraform.tfstate.backup
(B). directory called terraform.workspaces.tfstate
(C). a file called terraform.tfstate
(D). directory called terraform.tfstate.d

Answer: D
For local state, Terraform stores the workspace states in a directory called terraform.tfstate.d.
https://www.terraform.io/docs/state/workspaces.html#workspace-internals

QUESTION NO: 215
If a DevOps team adopts AWS Cloud Formation as their standardized method for provisioning public cloud resources, which of the following scenarios poses a challenge for this team?
(A). The team is asked to manage a new application stack built on AWS-natrve services
(B). The organization decides to expand into Azure and wishes to deploy new infrastructure using their existing codebase
(C). The team is asked to build a reusable code base that can deploy resources into any AWS region
(D). The DevOps team is tasked with automating a manual provisioning process
Answer: B

QUESTION NO: 216
Which of the below options is a valid interpolation syntax for retrieving a data source?
(A). ${google_storage_bucket.backend}
(B). ${azurerm_resource_group.test.data}
(C). ${aws_instance.web.id.data}
(D). ${data.google_dns_keys.foo_dns_keys.key_signing_keys[0].ds_record}
Answer: D
Data source attributes are interpolated with the general syntax data.TYPE.NAME.ATTRIBUTE. The interpolation for a resource is the same but without the data. prefix (TYPE.NAME.ATTRIBUTE).
https://www.terraform.io/docs/configuration-0-11/interpolation.html#attributes-of-a-data-source

QUESTION NO: 217
1. resource "aws_s3_bucket" "example" {
2. bucket = "my-test-s3-terraform-bucket"
3. ...} resource "aws_iam_role" "test_role" {
4. name = "test_role"
5. ...}
Due to the way that the application code is written, the s3 bucket must be created before the test role is created, otherwise there will be a problem. How can you ensure that?
(A). Add explicit dependency using depends_on . This will ensure the correct order of resource creation.
(B). This will already be taken care of by terraform native implicit dependency. Nothing else needs to be done from your end.
(C). This is not possible to control in terraform . Terraform will take care of it in a native way , and create a dependency graph that is best suited for the parallel resource creation.
(D). Create 2 separate terraform config scripts , and run them one by one , 1 for s3 bucket , and another for IAM role , run the S3 bucket script first.

Answer: A
Implicit dependency works only if there is some reference of one resource to another. Explicit dependency is the option here.

QUESTION NO: 218
A fellow developer on your team is asking for some help in refactoring their Terraform code. As part of their application's architecture, they are going to tear down an existing deployment managed by Terraform and deploy new. However, there is a server resource named aws_instance.ubuntu[1] they would like to keep to perform some additional analysis.
What command should be used to tell Terraform to no longer manage the resource?
(A). terraform apply rm aws_instance.ubuntu[1]
(B). terraform state rm aws_instance.ubuntu[1]
(C). terraform plan rm aws_instance.ubuntu[1]
(D). terraform delete aws_instance.ubuntu[1]
Answer: B

QUESTION NO: 219
All modules published on the official Terraform Module Registry have been verified by HashiCorp.
(A). True
(B). False
Answer: A

QUESTION NO: 220
A Terraform local value can reference other Terraform local values.
(A). True
(B). False

Answer: A

QUESTION NO: 221
Valarie has created a database instance in AWS and for ease of use is outputting the value of the database password with the following code. Valarie wants to hide the output value in the CLI after terraform apply that's why she has used sensitive parameter.
1. output "db_password" {
2. value = local.db_password
3. sensitive = true
4. }
Since sensitive is set to true, will the value associated with db password be available in plain-text in the state file for everyone to read?
(A). Yes
(B). No
Answer: A
Outputs can be marked as containing sensitive material by setting the sensitive attribute to true, like this:
output "sensitive" {

sensitive = true
value = VALUE
}
When outputs are displayed on-screen following a terraform apply or terraform refresh, sensitive outputs are redacted, with <sensitive> displayed in place of their value.
Limitations of Sensitive Outputs
The values of sensitive outputs are still stored in the Terraform state, and available using the terraform output command, so cannot be relied on as a sole means of protecting values. Sensitivity is not tracked internally, so if the output is interpolated in another module into a resource, the value will be displayed.

QUESTION NO: 222
Your developers are facing a lot of problem while writing complex expressions involving difficult interpolations . They have to run the terraform plan every time and check whether there are errors , and also check terraform apply to print the value as a temporary output for debugging purposes. What should be done to avoid this?
(A). Use terraform console command to have an interactive UI with full access to the underlying terraform state to run your interpolations , and debug at real-time.
(B). Add a breakpoint in your code, using the watch keyword , and output the value to console for temporary debugging.
(C). Use terraform zipmap function , it will be able to easily do the interpolations without complex code.
(D). Use terraform console command to have an interactive UI , but you can only use it with local state , and it does not work with remote state.
Answer: A
The terraform console command provides an interactive console for evaluating expressions. This is useful for testing interpolations before using them in configurations, and for interacting with any values currently saved in state.
https://www.terraform.io/docs/commands/console.html

QUESTION NO: 223
True or False? Each Terraform workspace uses its own state file to manage the infrastructure associated with that particular workspace.
(A). False
(B). True
Answer: B
The persistent data stored in the backend belongs to a workspace. Initially, the backend has only one workspace, called "default", and thus there is only one Terraform state associated with that configuration.

QUESTION NO: 224
Which one of the following will run echo 0 and echo 1 on a newly created host?
(A). provisioner "local-exec" { command = "echo 0"
command = "echo 1"
}
(B). provisioner "remote-exec" {

```
inline = [
echo 0,
echo 1
]
}
(C). provisioner "remote-exec" {
command = "${echo 0}"
command = "${echo 1}"
}
(D). provisioner "remote-exec" {
inline = [
"echo 0",
"echo 1"
]
}
```

Answer: D

remote-exec Provisioner

Example usage

```
resource "aws_instance" "web" {
# ...
provisioner "remote-exec" {
inline = [
"puppet apply",
"consul join ${aws_instance.web.private_ip}",
]
}
}
```

QUESTION NO: 225

Which of the below backends support state locking?

(A). S3

(B). consul

(C). azurerm

(D). artifactory

Answer: A,B,C

QUESTION NO: 226

Any user can publish modules to the public Terraform Module Registry.

(A). True

(B). False

Answer: B

QUESTION NO: 227

Which statement describes a goal of infrastructure as code?

(A). An abstraction from vendor specific APIs

(B). Write once, run anywhere
(C). A pipeline process to test and deliver software
(D). The programmatic configuration of resources
Answer: D

QUESTION NO: 228
You want to share Terraform state with your team, store it securely and provide state locking. How would you do this? Choose three correct answers.
(A). Using the consul Terraform backend.
(B). Using the remote Terraform backend with Terraform Cloud / Terraform Enterprise.
(C). Using the local backend.
(D). Using the s3 terraform backend. The dynamodb_field option e not needed.
(E). Using an s3 terraform backend with an appropriate IAM policy and dynamodb_field option configured.
Answer: A,B,E

QUESTION NO: 229
Which of the following commands will launch the Interactive console for Terraform interpolations?
(A). terraform console
(B). terraform cli
(C). terraform
(D). terraform cmdline
Answer: B
https://www.terraform.io/docs/commands/console.html

QUESTION NO: 230
Named workspaces are not a suitable isolation mechanism for strong separation between staging and production?
(A). True
(B). False

Answer: A
Organizations commonly want to create a strong separation between multiple deployments of the same infrastructure serving different development stages (e.g. staging vs. production) or different internal teams. In this case, the backend used for each deployment often belongs to that deployment, with different credentials and access controls. Named workspaces are not a suitable isolation mechanism for this scenario.
https://www.terraform.io/docs/state/workspaces.html#when-to-use-multiple-workspaces

QUESTION NO: 231
You have decided to create a new Terraform workspace to deploy a development environment. What is different about this workspace?
(A). It uses a different branch of code
It uses a different backend
(B). It has its own state file

(C). It pulls in a different terraform.tvars file
Answer: B

QUESTION NO: 232
Complete the following sentence:
The terraform state command can be used to ____
(A). modify state
(B). view state
(C). refresh state
(D). There is no such command
Answer: A
https://www.terraform.io/docs/commands/state/index.html

QUESTION NO: 233
Which of the following connection types are supported by the remote-exec provisioner? (select two)
(A). WinRM
(B). UDP
(C). SMB
(D). RDP
(E). ssh
Answer: A,E
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.
remote-exec connection types -
* ssh on Linux
* winrm on Windows
https://www.terraform.io/docs/provisioners/remote-exec.html

QUESTION NO: 234
True or False: Workspaces provide identical functionality in the open-source, Terraform Cloud, and Enterprise versions of Terraform.
(A). True
(B). False
Answer: B
https://www.terraform.io/docs/cloud/workspaces/index.html
https://www.terraform.io/docs/state/workspaces.html

QUESTION NO: 235
You have created a terraform script that uses a lot of new constructs that have been introduced in terraform v0.12. However, many developers who are cloning the script from your git repo, are using v0.11, and getting errors. What can be done from your end to solve this problem?
(A). Force developer to use v0.12 by using terraform setting 'required_version' and set it to >=0.12.

(B). Refactor the code to support both v0.11, and v0.12. It might be a difficult process, but there is no other way.
(C). Add a condition in front of each such specific construct, to check whether the running terraform version id v0.11 or v0.12, and ,work accordingly.
(D). Add comments in your code to tell developers to use v0.12 . If they use v0.11 , that should be their problem , which they need to figure out.
Answer: A
https://www.terraform.io/docs/configuration/terraform.html

QUESTION NO: 236
Refer to the following terraform variable definition
variable "track_tag" { type = list default = ["data_ec2","integration_ec2","digital_ec2"]}
track_tag = { Name = element(var.track_tag,count.index)} If count.index is set to 2, which of the following values will be assigned to the name attribute of track_tag variable?
(A). integration_ec2
(B). digital_ec2
(C). track_tag
(D). data_ec2
Answer: B

QUESTION NO: 237
Which of the following is available only in Terraform Enterprise or Cloud workspaces and not in Terraform CLI?
(A). Secure variable storage
(B). Support for multiple cloud providers
(C). Dry runs with terraform plan
(D). Using the workspace as a data source
Answer: B

QUESTION NO: 238
Setting the TF_LOG environment variable to DEBUG causes debug messages to be logged into syslog.
(A). True
(B). False
Answer: A

QUESTION NO: 239
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
Answer: A
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 NO: 240
As a member of an operations team that uses infrastructure as code (IaC) practices, you are tasked with making a change to an infrastructure stack running in a public cloud. Which pattern would follow IaC best practices for making a change?
(A). Make the change via the public cloud API endpoint
(B). Make the change programmatically via the public cloud CLI
(C). Submit a pull request and wait for an approved merge of the proposed changes
(D). Use the public cloud console to make the change after a database record has been approved
(E). Clone the repository containing your infrastructure code and then run the code

Answer: E

QUESTION NO: 241
Which of these options is the most secure place to store secrets foe connecting to a Terraform remote backend?
(A). Defined in Environment variables
(B). Inside the backend block within the Terraform configuration
(C). Defined in a connection configuration outside of Terraform
(D). None of above
Answer: A

QUESTION NO: 242
You should store secret data in the same version control repository as your Terraform configuration.
(A). True
(B). False
Answer: B

QUESTION NO: 243
John wants to use two different regions to deploy two different EC2 instances. He has specified two provider blocks in his providers.tf file.
provider "aws" { region = "us-east-1" }
provider "aws" { region = "us-west-2" }
When he run terraform plan he encountered an error. How to fix this?
(A). Use another provider version
(B). Use alias for region = "us-west-2"
(C). Use default keyword with region = "us-east-1"
(D). It can not be fixed
Answer: B

QUESTION NO: 244
Which of the following is allowed as a Terraform variable name?
(A). count

(B). name
(C). source
(D). version
Answer: B

QUESTION NO: 245
Terraform init can indeed be run only a few times, because, every time terraform init will initialize the project , and download all plugins from the internet repository , regardless of whether they were present or not , and this increases the waiting time
(A). True
(B). False
Answer: B
Re-running init with modules already installed will install the sources for any modules that were added to configuration since the last init, but will not change any already-installed modules. Use -upgrade to override this behavior, updating all modules to the latest available source code.
https://www.terraform.io/docs/commands/init.html

QUESTION NO: 246
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
Answer: A,B,C
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 NO: 247
lookup retrieves the value of a single element from which of the below data type?
(A). map
(B). set
(C). string
(D). list
Answer: A

https://www.terraform.io/docs/configuration/functions/lookup.html

QUESTION NO: 248
The Security Operations team of ABC Enterprise wants to mandate that all the Terraform configuration that creates an S3 bucket must have encryption feature enabled. What is the best way to achieve it?
(A). Use Sentinel Policies.
(B). Use S3 bucket policy.
(C). Create a script that checks the encryption parameter is enabled on every git commit.
(D). Shared a SOP to engineers to mandate encryption feature on S3.
Answer: A
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.
Using Sentinel with Terraform Cloud involves:
* Defining the policies - Policies are defined using the policy language with imports for parsing the Terraform plan, state and configuration.
* Managing policies for organizations - Users with permission to manage policies can add policies to their organization by configuring VCS integration or uploading policy sets through the API. They also define which workspaces the policy sets are checked against during runs. (More about permissions.)
* Enforcing policy checks on runs - Policies are checked when a run is performed, after the terraform plan but before it can be confirmed or the terraform apply is executed.
* Mocking Sentinel Terraform data - Terraform Cloud provides the ability to generate mock data for any run within a workspace. This data can be used with the Sentinel CLI to test policies before deployment.
https://www.terraform.io/docs/cloud/sentinel/index.html

QUESTION NO: 249
When using Terraform in a team it is important for everyone to be working with the same state so that operations will be applied to the same remote objects. Which of the below option is a recommended solution for this?
(A). Remote State
(B). Module
(C). Use the cached state and treat this as the record of truth.
(D). Workspace
Answer: A
https://www.terraform.io/docs/state/remote.html

QUESTION NO: 250
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 _____.

(A). First Time Configuration
(B). Default Configuration
(C). Changing Configuration
(D). Partial Configuration
(E). Incomplete Configuration

Answer: D
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.
https://www.terraform.io/docs/backends/config.html#partial-configuration

QUESTION NO: 251
Refer to the below code where developer is outputting the value of the database password but has used sensitive parameter to hide the output value in the CLI.
output "db_password" { value = aws_db_instance.db.password description = "The password for logging in to the database." sensitive = true} Since sensitive is set to true, the value associated with db password will not be present in state file as plain-text?
(A). False
(B). True
Answer: A
Sensitive output values are still recorded in the state, and so will be visible to anyone who is able to access the state data.

QUESTION NO: 252
Where in your Terraform configuration do you specify a state backend?
(A). The terraform block
(B). The resource block
(C). The provider block
(D). The datasource block
Answer: A
Backends are configured with a nested backend block within the top-level terraform block.

QUESTION NO: 253

When does Sentinel enforce policy logic during a Terraform Enterprise run?
(A). Before the plan phase
(B). During the plan phase
(C). Before the a apply phase
(D). After the apply phase
Answer: B

QUESTION NO: 254
In regards to Terraform state file, select all the statements below which are correct?
(A). When using local state, the state file is stored in plain-text.
(B). The state file is always encrypted at rest.
(C). Storing state remotely can provide better security.
(D). Using the mask feature, you can instruct Terraform to mask sensitive data in the state file.
(E). The Terraform state can contain sensitive data, therefore the state file should be protected from unauthorized access.
(F). Terraform Cloud always encrypts state at rest.
Answer: A,C,E,F
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.
When using remote state, state is only ever held in memory when used by Terraform. It may be encrypted at rest, but this depends on the specific remote state backend.
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.
Recommendations
If you manage any sensitive data with Terraform (like database passwords, user passwords, or private keys), treat the state itself as sensitive data.
Storing state remotely can provide better security. As of Terraform 0.9, Terraform does not persist state to the local disk when remote state is in use, and some backends can be configured to encrypt the state data at rest.
For example:
* Terraform Cloud always encrypts state at rest and protects it with TLS in transit. Terraform Cloud also knows the identity of the user requesting state and maintains a history of state changes. This can be used to control access and track activity. Terraform Enterprise also supports detailed audit logging.
* The S3 backend supports encryption at rest when the encrypt option is enabled. IAM policies and logging can be used to identify any invalid access. Requests for the state go over a TLS connection.

QUESTION NO: 255
Terraform provisioners can be added to any resource block.
(A). True

(B). False
Answer: A

QUESTION NO: 256
If a Terraform creation-time provisioner fails, what will occur by default?
(A). The resource will not be affected, but the provisioner will need to be applied again
(B). The resource will be destroyed
(C). The resource will be marked as "tainted"
(D). Nothing, provisioners will not show errors in the command line
Answer: C

QUESTION NO: 257
Which of the following does terraform apply change after you approve the execution plan? Choose two correct answers.
(A). The execution plan
(B). Terraform code
(C). Cloud infrastructure
(D). State file
(E). The .terraform directory
Answer: C,D

QUESTION NO: 258
You need to specify a dependency manually. What resource meta-parameter can you use lo make sure Terraform respects thee dependency?
Type your answer in the field provided. The text field is not case-sensitive and all variations of the correct answer are accepted.
Answer:
dependson

QUESTION NO: 259
Terraform and Terraform providers must use the same major version number in a single configuration.
(A). True
(B). False
Answer: B

QUESTION NO: 260
What Terraform command can be used to inspect the current state file?
(A). terraform inspect
(B). terraform read
(C). terraform show
(D). terraform state

Answer: C

QUESTION NO: 261
Terraform configuration (including any module references) can contain only one Terraform

provider type.
(A). True
(B). False
Answer: B

QUESTION NO: 262
What does the default "local" Terraform backend store?
(A). tfplan files
(B). Terraform binary
(C). Provider plugins
(D). State file
Answer: D
The local backend stores state on the local filesystem, locks that state using system APIs, and performs operations locally.

QUESTION NO: 263
You want terraform plan and terraform apply to be executed in Terraform Cloud's run environment but the output is to be streamed locally. Which one of the below you will choose?
(A). Local Backends.
(B). Terraform Backends.
(C). This can be done using any of the local or remote backends.
(D). Remote Backends.
Answer: D
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. Remote plans and applies use variable values from the associated Terraform Cloud workspace.
Terraform Cloud can also be used with local operations, in which case only state is stored in the Terraform Cloud backend.
https://www.terraform.io/docs/backends/types/remote.html

QUESTION NO: 264
Terraform Cloud always encrypts state at rest and protects it with TLS in transit. Terraform Cloud also knows the identity of the user requesting state and maintains a history of state changes.
(A). False
(B). True
Answer: B
Terraform Cloud always encrypts state at rest and protects it with TLS in transit. Terraform Cloud also knows the identity of the user requesting state and maintains a history of state changes. This can be used to control access and track activity. Terraform Enterprise also supports detailed audit logging.
https://www.terraform.io/docs/state/sensitive-data.html#recommendations

QUESTION NO: 265
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 can help businesses deploy applications on multiple clouds and on-premises infrastructure.
(B). Terraform is not cloud-agnostic and can be used to deploy resources across a single public cloud.
(C). Terraform simplifies management and orchestration, helping operators build large-scale, multi-cloud infrastructure.
(D). Terraform can manage cross-cloud dependencies.
Answer: A,C,D
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.
https://www.terraform.io/intro/use-cases.html

QUESTION NO: 266
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?
(A). terraform workspace dev
(B). terraform workspace select dev
(C). terraform workspace -switch dev
(D). terraform workspace switch dev
Answer: B
The terraform workspace select command is used to choose a different workspace to use for further operations. https://www.terraform.io/docs/commands/workspace/select.html

QUESTION NO: 267
How can terraform plan aid in the development process?
(A). Validates your expectations against the execution plan without permanently modifying state
(B). Initializes your working directory containing your Terraform configuration files
(C). Formats your Terraform configuration files
(D). Reconciles Terraform's state against deployed resources and permanently modifies state using the current status of deployed resources
Answer: A

QUESTION NO: 268
Forcing the recreation of a resource is useful when you want a certain side effect of recreation that is not visible in the attributes of a resource. What command will do this?
(A). terraform taint
(B). terraform apply
(C). terraform graph
(D). terraform refresh
Answer: A

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 and the next apply will implement this change.
Forcing the recreation of a resource is useful when you want a certain side effect of recreation that is not visible in the attributes of a resource. For example: re-running provisioners will cause the node to be different or rebooting the machine from a base image will cause new startup scripts to run.
Note that tainting a resource for recreation may affect resources that depend on the newly tainted resource. For example, a DNS resource that uses the IP address of a server may need to be modified to reflect the potentially new IP address of a tainted server. The plan command will show this if this is the case.
This example will taint a single resource:
$ terraform taint aws_security_group.allow_all
The resource aws_security_group.allow_all in the module root has been marked as tainted.
https://www.terraform.io/docs/commands/taint.html

QUESTION NO: 269
What resource dependency information is stored in Terraform's state?
(A). Only implicit dependencies are stored in state.
(B). Both implicit and explicit dependencies are stored in state.
(C). Only explicit dependencies are stored in state.
(D). No dependency information is stored in state.
Answer: B
Terraform state captures all dependency information, both implicit and explicit. One purpose for state is to determine the proper order to destroy resources. When resources are created all of their dependency information is stored in the state. If you destroy a resource with dependencies, Terraform can still determine the correct destroy order for all other resources because the dependencies are stored in the state.
https://www.terraform.io/docs/state/purpose.html#metadata

QUESTION NO: 270
You need to constrain the GitHub provider to version 2.1 or greater.
Which of the following should you put into the Terraform 0.12 configuration's provider block?
(A). version >= 2.1
(B). version ~> 2.1
(C). version = "<= 2.1"
(D). version = ">= 2.1"

Answer: B

QUESTION NO: 271
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
Answer: B,C,D
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:
https://www.terraform.io/docs/backends/init.html#backend-initialization

QUESTION NO: 272
terraform apply will fail if you have not run terraform plan first to update the plan output.
(A). True
(B). False
Answer: B

QUESTION NO: 273
In the following code snippet, the block type is identified by which string?
(A). "aws_instance"
(B). resource
(C). "db"
(D). instance_type
Answer: B

QUESTION NO: 274
What is the standard workflow that a developer follows while working with terraform open source version?
(A). Run terraform refresh to update the terraform state , then write the terraform code , and finally run terraform apply.
(B). Run terraform destroy first since you need to start from fresh every time , before running terraform apply.
(C). Write terraform code , and run terraform push , to update the terraform state to the remote repo , which in turn will take care of the next steps.
(D). Write the terraform code on the developer machine , run terraform plan to check the changes , and run terraform apply to provision the infra.
Answer: D
You do not need to run terraform refresh as terraform plan implicitly will run terraform refresh.
https://www.terraform.io/guides/core-workflow.html

QUESTION NO: 275
resource "aws_s3_bucket" "example" { bucket = "my-test-s3-terraform-bucket" ...} resource "aws_iam_role" "test_role" { name = "test_role" ...} Due to the way that the application code is

written , the s3 bucket must be created before the test role is created , otherwise there will be a problem. How can you ensure that?
(A). This will already be taken care of by terraform native implicit dependency. Nothing else needs to be done from your end.
(B). Add explicit dependency using depends_on . This will ensure the correct order of resource creation.
(C). Create 2 separate terraform config scripts , and run them one by one , 1 for s3 bucket , and another for IAM role , run the S3 bucket script first.
(D). This is not possible to control in terraform . Terraform will take care of it in a native way , and create a dependency graph that is best suited for the parallel resource creation.
Answer: B
Use the depends_on meta-argument to handle hidden resource dependencies that Terraform can't automatically infer.
Explicitly specifying a dependency is only necessary when a resource relies on some other resource's behavior but doesn't access any of that resource's data in its arguments.

QUESTION NO: 276
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?
(A). source = "git::https://example.com/my_test_module.git@tag=v1.0.4"
(B). source = "git::https://example.com/my_test_module.git&ref=v1.0.4"
(C). source = "git::https://example.com/my_test_module.git#tag=v1.0.4"
(D). source = "git::https://example.com/my_test_module.git?ref=v1.0.4"
Answer: D
https://www.terraform.io/docs/modules/sources.html#selecting-a-revision

QUESTION NO: 277
What is the purpose of using the local-exec provisioner? (Select Two)
(A). To invoke a local executable.
(B). Executes a command on the resource to invoke an update to the Terraform state.
(C). To execute one or more commands on the machine running Terraform.
(D). Ensures that the resource is only executed in the local infrastructure where Terraform is deployed.
Answer: A,C
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.
Example usage
resource "aws_instance" "web" {
# ...
provisioner "local-exec" {

command = "echo ${aws_instance.web.private_ip} >> private_ips.txt"
}
}
Note: Provisioners should only be used as a last resort. For most common situations there are better alternatives.
https://www.terraform.io/docs/provisioners/local-exec.html

QUESTION NO: 278
Hanah is writing a terraform configuration with nested modules, there are multiple places where she has to use the same conditional expression but she wants to avoid repeating the same values or expressions multiple times in the configuration,. What is a better approach to dealing with this?
(A). Expressions
(B). Local Values
(C). Variables
(D). Functions
Answer: B
https://www.terraform.io/docs/configuration/locals.html

QUESTION NO: 279
A Terraform output that sets the "sensitive" argument to true will not store that value in the state file.
(A). True
(B). False
Answer: B

QUESTION NO: 280
How would you reference the "name" value of the second instance of this fictitious resource?

```
resource "aws_instance" "web" {
  count = 2
  name = "terraform-${count.index}"
}
```


(A). element(aws_instance.web, 2)
(B). aws_instance.web[1].name
(C). aws_instance.web[1]
(D). aws_instance.web[2].name
(E). aws_instance.web.*.name
Answer: A

QUESTION NO: 281
Once a resource is marked as tainted, the next plan will show that the resource will be _____ and _____ and the next apply will implement this change.
(A). recreated and tainted
(B). destroyed and not recreated
(C). tainted and not destroyed

(D). destroyed and recreated
Answer: D

QUESTION NO: 282
You want to get involved in the development of Terraform. As this is an open source project, you would like to contribute a fix for an open issue of Terraform. What programming language will need to use to write the fix?
(A). It depends on which command issue related to.
(B). Python
(C). Go
(D). Java
Answer: C
Basic programming knowledge. Terraform and Terraform Plugins are written in the Go programming language, but even if you've never written a line of Go before, you're still welcome to take a dive into the code and submit patches. The community is happy to assist with code reviews and offer guidance specific to Go.

QUESTION NO: 283
How is the Terraform remote backend different than other state backends such as S3, Consul, etc.?
(A). It can execute Terraform runs on dedicated infrastructure on premises or in Terraform Cloud
(B). It doesn't show the output of a terraform apply locally
(C). It is only available to paying customers
(D). All of the above
Answer: A
If you and your team are using Terraform to manage meaningful infrastructure, we recommend using the remote backend with Terraform Cloud or Terraform Enterprise.

QUESTION NO: 284
Which of the below features of Terraform can be used for managing small differences between different environments which can act more like completely separate working directories.
(A). Repositories
(B). Workspaces
(C). Environment Variables
(D). Backends
Answer: B
workspaces allow conveniently switching between multiple instances of a single configuration within its single backend. They are convenient in a number of situations, but cannot solve all problems.
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.
Non-default workspaces are often related to feature branches in version control. The default workspace might correspond to the "master" or "trunk" branch, which describes the intended state of production infrastructure. When a feature branch is created to develop a change, the developer of that feature might create a corresponding workspace and deploy into it a temporary "copy" of the main infrastructure so that changes can be tested without affecting the production infrastructure. Once the change is merged and deployed to the default workspace, the test infrastructure can be destroyed and the temporary workspace deleted.
https://www.terraform.io/docs/state/workspaces.html
https://www.terraform.io/docs/state/workspaces.html#when-to-use-multiple-workspaces

QUESTION NO: 285
Which of the following is the right substitute for static values that can make Terraform configuration file more dynamic and reusable?
(A). Output value
(B). Input parameters
(C). Functions
(D). Modules
Answer: B
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 NO: 286
In Terraform 0.13 and above, outside of the required_providers block, Terraform configurations always refer to providers by their local names.
(A). True
(B). False
Answer: A
Outside of the required_providers block, Terraform configurations always refer to providers by their local names.

QUESTION NO: 287
What information does the public Terraform Module Registry automatically expose about published modules?
(A). Required input variables
(B). Optional inputs variables and default values
(C). Outputs
(D). All of the above
(E). None of the above
Answer: E

QUESTION NO: 288
Which of the following is considered a Terraform plugin?
(A). Terraform language
(B). Terraform tooling

(C). Terraform logic
(D). Terraform provider
Answer: D
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.
https://www.terraform.io/docs/plugins/basics.html

QUESTION NO: 289
The terraform init command is always safe to run multiple times, to bring the working directory up to date with changes in the configuration. Though subsequent runs may give errors, this command will never delete your existing configuration or state.
(A). False
(B). True
Answer: B
https://www.terraform.io/docs/commands/init.html

QUESTION NO: 290
When should you use the force-unlock command?
(A). You see a status message that you cannot acquire the lock
(B). You have a high priority change
(C). Automatic unlocking failed
(D). Your apply failed due to a state lock

Answer: C
Manually unlock the state for the defined configuration.

QUESTION NO: 291
Your team lead does not trust the junior terraform engineers who now have access to the git repo . So , he wants you to have some sort of a checking layer , whereby , you can ensure that the juniors will not create any non-compliant resources that might lead to a security audit failure in future. What can you do to efficiently enforce this?
(A). Create a design /security document (in PDF) and share to the team , and ask them to always follow that document , and never deviate from it.
(B). Since your team is using Hashicorp Terraform Enterprise Edition , enable Sentinel , and write Policy-As-Code rules that will check for non-compliant resource provisioning , and prevent/report them.
(C). Use Terraform OSS Sentinel Lite version , which will save cost , since there is no charge for OSS , but it can still check for most non-compliant rules using Policy-As-Code.
(D). Create a git master branch , and implement PR . Every change needs to be reviewed by you , before being merged to the master branch.
Answer: B
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.
https://www.terraform.io/docs/cloud/sentinel/index.html

QUESTION NO: 292
Workspaces in Terraform provides similar functionality in the open-source, Terraform Cloud, and Enterprise versions of Terraform.
(A). True
(B). False
Answer: B
https://www.terraform.io/docs/cloud/migrate/workspaces.html
Workspaces, managed with the terraform workspace command, aren't the same thing as Terraform Cloud's workspaces. Terraform Cloud workspaces act more like completely separate working directories; CLI workspaces are just alternate state files.

QUESTION NO: 293
Your DevOps team is currently using the local backend for your Terraform configuration. You would like to move to a remote backend to begin storing the state file in a central location. Which of the following backends would not work?
(A). Amazon S3
(B). Artifactory
(C). Git
(D). Terraform Cloud
Answer: A

QUESTION NO: 294
John is writing a module and within the module, there are multiple places where he has to use the same conditional expression but he wants to avoid repeating the same values or expressions multiple times in a configuration,. What is a better approach to dealing with this?
(A). Local Values
(B). Expressions
(C). Functions
(D). Variables
Answer: A
A local value assigns a name to an expression, allowing it to be used multiple times within a module without repeating it.
https://www.terraform.io/docs/configuration/locals.html

QUESTION NO: 295
What is the command you can use to set an environment variable named "var1"of type String?
(A). export TF_VAR_VAR1
(B). set TF_VAR_var1
(C). variable "var1" { type = "string"}
(D). export TF_VAR_var1
Answer: D
The environment variable must be in the format TF_VAR_name, so for the
https://www.terraform.io/docs/commands/environment-variables.html#tf_var_name

QUESTION NO: 296
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
Answer: A

QUESTION NO: 297
Multiple configurations for the same provider can be used in a single configuration file.
(A). False
(B). True
Answer: B
You can optionally define multiple configurations for the same provider, and select which one to use on a per-resource or per-module basis. The primary reason for this is to support multiple regions for a cloud platform; other examples include targeting multiple Docker hosts, multiple Consul hosts, etc.
To include multiple configurations for a given provider, include multiple provider blocks with the same provider name, but set the alias meta-argument to an alias name to use for each additional configuration. For example:
# The default provider configuration
provider "aws" {
region = "us-east-1"
}
# Additional provider configuration for west coast region
provider "aws" {
alias = "west"
region = "us-west-2"
}
The provider block without alias set is known as the default provider configuration. When alias is set, it creates an additional provider configuration. For providers that have no required configuration arguments, the implied empty configuration is considered to be the default provider configuration.
https://www.terraform.io/docs/configuration/providers.html#alias-multiple-provider-instances

QUESTION NO: 298
Only the user that generated a plan may apply it.
(A). True
(B). False
Answer: A
The optional -out argument can be used to save the generated plan to a file for later execution with terraform apply, which can be useful when running Terraform in automation.

QUESTION NO: 299

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). Object Storage
(B). State Locking
(C). WorkSpaces
(D). Encryption
Answer: B
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.
If acquiring the lock is taking longer than expected, Terraform will output a status message. If Terraform doesn't output a message, state locking is still occurring if your backend supports it.
Not all backends support locking. Please view the list of backend types for details on whether a backend supports locking or not.
https://www.terraform.io/docs/state/locking.html

QUESTION NO: 300
Terraform plan updates your state file.
(A). True
(B). False

Answer: B

QUESTION NO: 301
You have never used Terraform before and would like to test it out using a shared team account for a cloud provider. The shared team account already contains 15 virtual machines (VM). You develop a Terraform configuration containing one VM, perform terraform apply, and see that your VM was created successfully.
What should you do to delete the newly-created VM with Terraform?
(A). The Terraform state file contains all 16 VMs in the team account. Execute terraform destroy and select the newly-created VM.
(B). The Terraform state file only contains the one new VM. Execute terraform destroy.
(C). Delete the Terraform state file and execute Terraform apply.
(D). Delete the VM using the cloud provider console and terraform apply to apply the changes to the Terraform state file.
Answer: B

QUESTION NO: 302
The Terraform language does not support user-defined functions, and so only the functions built in to the language are available for use.
(A). False
(B). True
Answer: B

https://www.terraform.io/docs/configuration/functions.html

QUESTION NO: 303
Multiple providers can be declared within a single Terraform configuration file.
(A). True
(B). False
Answer: A
You can optionally define multiple configurations for the same provider, and select which one to use on a per-resource or per-module basis. The primary reason for this is to support multiple regions for a cloud platform; other examples include targeting multiple Docker hosts, multiple Consul hosts, etc.
To include multiple configurations for a given provider, include multiple provider blocks with the same provider name, but set the alias meta-argument to an alias name to use for each additional configuration.
For Example
# The default provider configuration
provider "aws" {
region = "us-east-1"
}
# Additional provider configuration for west coast region
provider "aws" {
alias = "west"
region = "us-west-2"
}
The provider block without alias set is known as the default provider configuration. When alias is set, it creates an additional provider configuration. For providers that have no required configuration arguments, the implied empty configuration is considered to be the default provider configuration.
https://www.terraform.io/docs/configuration/providers.html

QUESTION NO: 304
What does the command terraform fmt do?
(A). Rewrite Terraform configuration files to a canonical format and style.
(B). Deletes the existing configuration file.
(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.
Answer: A
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.
https://www.terraform.io/docs/commands/fmt.html

QUESTION NO: 305

What is the provider for this fictitious resource?

```
resource "aws_vpc" "main" {
    name = "test"
}
```


(A). vpc
(B). main
(C). aws
(D). test
Answer: C

QUESTION NO: 306
How can you ensure that the engineering team who has access to git repo will not create any non-compliant resources that might lead to a security audit failure in future. your team is using Hashicorp Terraform Enterprise Edition.
(A). Use Terraform OSS Sentinel Lite version , which will save cost , since there is no charge for OSS , but it can still check for most non-compliant rules using Policy-As-Code.
(B). Implement a review process where every code will be reviewed before merging to the master branch.
(C). Since your team is using Hashicorp Terraform Enterprise Edition , enable Sentinel , and write Policy-As-Code rules that will check for non-compliant resource provisioning , and prevent/report them.
(D). Create a design /security document (in PDF) and share to the team , and ask them to always follow that document , and never deviate from it.
Answer: C
https://www.terraform.io/docs/cloud/sentinel/index.html

QUESTION NO: 307
Which option can not be used to keep secrets out of Terraform configuration files?
(A). A Terraform provider
(B). Environment variables
(C). A -var flag
(D). secure string
Answer: A

QUESTION NO: 308
Your team has started using terraform OSS in a big way , and now wants to deploy multi region deployments (DR) in aws using the same terraform files . You want to deploy the same infra (VPC,EC2 ...) in both us-east-1 ,and us-west-2 using the same script , and then peer the VPCs across both the regions to enable DR traffic. But , when you run your script , all resources are getting created in only the default provider region. What should you do?
Your provider setting is as below -
The default provider configuration provider "aws" { region = "us-east-1" }
(A). No way to enable this via a single script . Write 2 different scripts with different default providers in the 2 scripts , one for us-east , another for us-west.
(B). Create a list of regions , and then use a for-each to iterate over the regions , and create

the same resources ,one after the one , over the loop.
(C). Use provider alias functionality , and add another provider for us-west region . While creating the resources using the tf script , reference the appropriate provider (using the alias) .
(D). Manually create the DR region , once the Primary has been created , since you are using terraform OSS , and multi region deployment is only available in Terraform Enterprise.
Answer: C
You can optionally define multiple configurations for the same provider, and select which one to use on a per-resource or per-module basis. The primary reason for this is to support multiple regions for a cloud platform; other examples include targeting multiple Docker hosts, multiple Consul hosts, etc.
To include multiple configurations for a given provider, include multiple provider blocks with the same provider name, but set the alias meta-argument to an alias name to use for each additional configuration. For example:
# The default provider configuration
provider "aws" {
region = "us-east-1"
}
# Additional provider configuration for west coast region
provider "aws" {
alias = "west"
region = "us-west-2"
}
https://www.terraform.io/docs/configuration/providers.html

QUESTION NO: 309
You are reviewing Terraform configurations for a big project in your company. You noticed that there are several identical sets of resources that appear in multiple configurations. What feature of Terraform would you recommend to use to reduce the amount of cloned configuration between the different configurations?
(A). Packages
(B). Backends
(C). Provisioners
(D). Modules
Answer: D
Modules are reusable configuration packages that Terraform can share through a variety of sources including Terraform Registries, GitHub, and Amazon S3 buckets.
A module is a container for multiple resources that are used together. Modules can be used to create lightweight abstractions, so that you can describe your infrastructure in terms of its architecture, rather than directly in terms of physical objects.
Modules are reusable configuration packages that Terraform can share through a variety of sources including Terraform Registries, GitHub, and Amazon S3 buckets.
https://www.terraform.io/docs/modules/index.html

QUESTION NO: 310

Where can Terraform not load a provider from?
(A). Plugins directory
(B). Provider plugin cache
(C). Official HashrCorp distribution on releases, hashicorp.com
(D). Source code
Answer: D

QUESTION NO: 311
Which of the following statements best describes the Terraform list(...) type?
(A). a collection of values where each is identified by a string label.
(B). a sequence of values identified by consecutive whole numbers starting with zero.
(C). a collection of unique values that do not have any secondary identifiers or ordering.
(D). a collection of named attributes that each have their own type.
Answer: B
A terraform list is a sequence of values identified by consecutive whole numbers starting with zero. https://www.terraform.io/docs/configuration/types.html#structural-types

QUESTION NO: 312
Command terraform refresh will update state file?
(A). False
(B). True
Answer: B
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.
This 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.
https://www.terraform.io/docs/commands/refresh.html

QUESTION NO: 313
The terraform state command can be used to ____
(A). Update current state
(B). Refresh existing state file
(C). Print the current state file in console
(D). It is not a valid command
Answer: A
The terraform state command is used for advanced state management. Rather than modify the state directly, the terraform state commands can be used in many cases instead.
https://www.terraform.io/docs/commands/state/index.html

QUESTION NO: 314
Which of the below are paid features of Terraform Cloud?
(A). Full API Coverage
(B). Secure variable Storage
(C). Roles/ Team management

(D). Cost Estimation
(E). Private Module Registry
(F). Sentinel policies
Answer: C,D,F
https://www.hashicorp.com/products/terraform/pricing/

QUESTION NO: 315
Which of the following statements about local modules is incorrect:
(A). Local modules are not cached by terraform init command
(B). Local modules are sourced from a directory on disk
(C). Local modules support versions
(D). All of the above (all statements above are incorrect
(E). None of the above (all statements above are correct)
Answer: C

QUESTION NO: 316
You run a local-exec provisioner in a null resource called null_resource.run_script and realize that you need to rerun the script.
Which of the following commands would you use first?
(A). terraform validate null_resource.run_script
(B). terraform plan -target=null_resource.run_script
(C). terraform apply -target=null_resource.run_script
(D). terraform taint null_resource.run_script
Answer: D

QUESTION NO: 317
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). Export the value using terraform export and input the value using terraform input.
(B). Configure the pertinent provider's configuration with a list of possible IP addresses to use.
(C). Configure an output value in the application module in order to use that value for the DNS module.
(D). Preconfigure the IP address as a parameter in the DNS module.
Answer: C
Output values are like the return values of a Terraform module, and have several uses:
* A child module can use outputs to expose a subset of its resource attributes to a parent module.
* A root module can use outputs to print certain values in the CLI output after running terraform apply.
* When using remote state, root module outputs can be accessed by other configurations via a terraform_remote_state data source.

https://www.terraform.io/docs/configuration/outputs.html

QUESTION NO: 318
Which of the following state management command allow you to retrieve a list of resources that are part of the state file?
(A). terraform state list
(B). terraform state view
(C). terraform view
(D). terraform list
Answer: A
The terraform state list command is used to list resources within a Terraform state.
Usage: terraform state list [options] [address...]
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.
https://www.terraform.io/docs/commands/state/list.html

QUESTION NO: 319
You need to deploy resources into two different cloud regions in the same Terraform configuration. To do that, you declare multiple provider configurations as follows:

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

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


What meta-argument do you need to configure in a resource block to deploy the resource to the "us-west-2" AWS region?
(A). alias = west
(B). provider = west
(C). provider = aws.west
(D). alias = aws.west
Answer: A

QUESTION NO: 320
Which feature of Terraform allows multiple state files for a single configuration file depending upon the environment?
(A). Terraform Modules
(B). Terraform Enterprise
(C). Terraform Workspaces
(D). Terraform Remote Backends
Answer: C

QUESTION NO: 321

Which two steps are required to provision new infrastructure in the Terraform workflow? (Choose two.)
(A). Destroy
(B). Apply
(C). Import
(D). Init
(E). Validate
Answer: B,D

QUESTION NO: 322
Which of the following is not a valid string function in Terraform?
(A). split
(B). join
(C). slice
(D). chomp
Answer: C

QUESTION NO: 323
What features does the hosted service Terraform Cloud provide? (Choose two.)
(A). Automated infrastructure deployment visualization
(B). Automatic backups
(C). Remote state storage
(D). A web-based user interface (UI)
Answer: B,C
Reference:
https://www.terraform.io/docs/enterprise/admin/automated-recovery.html
https://www.terraform.io/docs/language/state/remote.html

QUESTION NO: 324
Your manager has instructed you to start using terraform for your day-to-day operations, but your security team is concerned about the terraform state files. They have heard it contains confidential information, and are worried that it will not be securely protected. What should be your response to the security team in this regard?
(A). Inform the security team that using terraform state is optional . There are ways to avoid it , and you will do the same.
(B). Ensure that the state is managed in a remote backend , preferably an enterprise grade state management system like Terraform Cloud.
(C). Mask the confidential entries in the terraform state file , using Vault Enterprise, another Hashicorp product , while keeping it locally.
(D). Keep the state file locally on each developer machine , and ensure that there is a local protection software like KeyPass protecting it.
Answer: B
https://www.terraform.io/docs/state/index.html
State is very important topic for exam. Please read all of the below subtopics Purpose Import Existing Resources Locking Workspaces Remote State Sensitive Data

QUESTION NO: 325
In a Terraform Cloud workspace linked to a version control repository, speculative plan runs start automatically when you merge or commit changes to version control.
(A). True
(B). False
Answer: B

QUESTION NO: 326
How would you reference the attribute "name* of this fictitious resource in HCL?

```
resource "kubernetes_namespace" "example" {
    name = "test"
}
```


(A). resource.kubrnetes_namespace>example.name
(B). kubernetes_namespace.test.name
(C). kubernetes_namespace.example,name
(D). data kubernetes_namespace.name
(E). None of the above
Answer: C

QUESTION NO: 327
You just scaled your VM infrastructure and realized you set the count variable to the wrong value. You correct the value and save your change.
What do you do next to make your infrastructure match your configuration?
(A). Run an apply and confirm the planned changes
(B). Inspect your Terraform state because you want to change it
(C). Reinitialize because your configuration has changed
(D). Inspect all Terraform outputs to make sure they are correct
Answer: A

QUESTION NO: 328
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
Answer: C,D

QUESTION NO: 329
Dawn has created the below child module. Without changing the module, can she override the instance_type from t2.micro to t2.large form her code while calling this module?
1. resource "aws_instance" "myec2"
2. {
3. ami = "ami-082b5a644766e0e6f"
4. instance_type = "t2.micro
5. }

(A). YES
(B). No
Answer: B
As the instance_type is hard-coded in source module, you will not be able to change its value from destination module. Instead of hard-coding you should use variable with default values.

QUESTION NO: 330
Which of the following is not a benefit of adopting infrastructure as code?
(A). Automation
(B). Versioning
(C). Reusability of code
(D). Interpolation
Answer: D

QUESTION NO: 331
Taint the resource "aws_instance" "baz" resource that lives in module bar which lives in module foo.
(A). terraform taint module.foo.module.bar.baz
(B). terraform taint module.foo.bar.aws_instance.baz
(C). terraform taint module.foo.module.bar.aws_instance.baz
(D). terraform taint foo.bar.aws_instance.baz
Answer: C
Check resource addressing
https://www.terraform.io/docs/internals/resource-addressing.html

QUESTION NO: 332
You want terraform plan and apply to be executed in Terraform Cloud's run environment but the output is to be streamed locally. Which one of the below you will choose?
(A). Local Backends
(B). This can be done using any of the local or remote backends
(C). Remote Backends
(D). Terraform Backends
Answer: C
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.
Remote plans and applies use variable values from the associated Terraform Cloud workspace.
https://www.terraform.io/docs/backends/types/remote.html

QUESTION NO: 333
Refer below code where pessimistic constraint operator has been used to specify a version of a provider.

terraform { required_providers { aws = "~> 1.1.0" }}
Which of the following options are valid provider versions that satisfy the above constraint. (select two)
(A). 1.1.1
(B). 1.2.9
(C). 1.1.8
(D). 1.2.0
Answer: A,C
Pessimistic constraint operator, constraining both the oldest and newest version allowed. For example, ~> 0.9 is equivalent to >= 0.9, < 1.0, and ~> 0.8.4, is equivalent to >= 0.8.4, < 0.9

QUESTION NO: 334
When using Terraform to deploy resources into Azure, which scenarios are true regarding state files? (Choose two.)
(A). When a change is made to the resources via the Azure Cloud Console, the changes are recorded in a new state file
(B). When a change is made to the resources via the Azure Cloud Console, Terraform will update the state file to reflect them during the next plan or apply
(C). When a change is made to the resources via the Azure Cloud Console, the current state file will not be updated
(D). When a change is made to the resources via the Azure Cloud Console, the changes are recorded in the current state file
Answer: B

QUESTION NO: 335
You have created two workspaces PROD and DEV. You have switched to DEV and provisioned DEV infrastructure from this workspace. Where is your state file stored?
(A). terraform.d
(B). terraform.tfstate
(C). terraform.tfstate.DEV
(D). terraform.tfstate.d
Answer: D
Terraform stores the workspace states in a directory called terraform.tfstate.d. This directory should be treated similarly to default workspace state file terraform.tfstate

```
├── main.tf
├── provider.tf
├── terraform.tfstate.d
│ ├── DEV
│ │ └── terraform.tfstate # DEV workspace state file
│ └── PROD
│ └── terraform.tfstate # PROD workspace state file
├── terraform.tfvars # Default workspace state file
└── variables.tf
```

QUESTION NO: 336

Consider the following Terraform 0.12 configuration snippet:
1. variable "vpc_cidrs" {
2. type = map
3. default = {
4. us-east-1 = "10.0.0.0/16"
5. us-east-2 = "10.1.0.0/16"
6. us-west-1 = "10.2.0.0/16"
7. us-west-2 = "10.3.0.0/16"
8. }
9. }
10.
11. resource "aws_vpc" "shared" {
12. cidr_block = _____
13. }
How would you define the cidr_block for us-east-1 in the aws_vpc resource using a variable?
(A). var.vpc_cidrs.0
(B). vpc_cidrs["us-east-1"]
(C). var.vpc_cidrs["us-east-1"]
(D). var.vpc_cidrs[0]
Answer: C

QUESTION NO: 337
While Terraform is generally written using the HashiCorp Configuration Language (HCL), what other syntax can Terraform are expressed in?
(A). JSON
(B). YAML
(C). TypeScript
(D). XML
Answer: A
The constructs in the Terraform language can also be expressed in JSON syntax, which is harder for humans to read and edit but easier to generate and parse programmatically.

QUESTION NO: 338
You have provisioned some aws resources in your test environment through Terraform for a POC work. After the POC, now you want to destroy the resources but before destroying them you want to check what resources will be getting destroyed through terraform. what are the options of doing that? (Select TWO)
(A). Use terraform destroy command
(B). This is not possible
(C). Use terraform plan command
(D). Use terraform plan -destroy command.
Answer: A,D
https://learn.hashicorp.com/terraform/getting-started/destroy

QUESTION NO: 339

Which flag would be used within a Terraform configuration block to identify the specific version of a provider required?
(A). required-provider
(B). required-version
(C). required_providers
(D). required_versions
Answer: C
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.
Example
terraform {
required_providers {
aws = ">= 2.7.0"
}
}

QUESTION NO: 340
Why might a user opt to include the following snippet in their configuration file?
(A). Terraform 0.12 introduced substantial changes to the syntax used to write Terraform configuration
(B). The user wants to ensure that the application being deployed is a minimum version of 0.12
(C). this ensures that all Terraform providers are above a certain version to match the application being deployed
(D). versions before Terraform 0.12 were not approved by HashiCorp to be used in production

Answer: A

QUESTION NO: 341
terraform init retrieves the source code tot all referenced modules
(A). True
(B). False
Answer: A

QUESTION NO: 342
Terraform has detailed logs which can be enabled by setting the _____ environmental variable.
(A). TF_TRACE
(B). TF_DEBUG
(C). TF_LOG
(D). TF_INFO
Answer: C
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.
https://www.terraform.io/docs/internals/debugging.html

QUESTION NO: 343
Terraform-specific settings and behaviors are declared in which configuration block type?
(A). provider
(B). terraform
(C). resource
(D). data
Answer: B
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.
Example
terraform {
required_version = "> 0.12.0"
}
https://www.terraform.io/docs/configuration/terraform.html

QUESTION NO: 344
You have configured an Auto Scaling group in AWS to automatically scale the number of instances behind a load balancer based on the instances CPU utilization. The instances are configured using a Launch Configuration. You have observed that the Auto Scaling group doesn't successfully scale when you apply changes that require replacing the Launch Configuration. Why is this happening?
(A). You need to configure an explicit dependency for the Auto Scaling group using the depends_on meta-parameter.
(B). You need to configure an explicit dependency for the Launch Configuration using the depends_on meta-parameter.
(C). You need to configure the Auto Scaling group's create_before_destroy meta-parameter.
(D). You need to configure the Launch Configuration's create_before_destroy meta-parameter.
Answer: D
https://www.terraform.io/docs/providers/aws/r/launch_configuration.html#using-withautoscaling-groups

QUESTION NO: 345
Which of the following is not a valid Terraform string function?
(A). replace
(B). format
(C). join
(D). tostring
Answer: D
https://www.terraform.io/docs/configuration/functions/tostring.html

QUESTION NO: 346
Which of the following clouds does not have a provider maintained HashiCorp?
(A). IBM Cloud
(B). DigitalOcean

(C). OpenStack
(D). AWS
Answer: A
IBM Cloud does not have a provider maintained by HashiCorp, although IBM Cloud does maintain their own Terraform provider.
https://www.terraform.io/docs/providers/index.html

QUESTION NO: 347
How does Terraform handle working with so many providers?
(A). Terraform ships with all of the plugins embedded in the Terraform binary.
(B). Terraform uses a plugin architecture for providers and only installs the provider plugins required by your configuration in the configuration's working directory.
(C). Terraform uses a plugin architecture for providers and only installs the provider plugins required by your configuration in a shared, system-wide plugins directory.
(D). Terraform allows you to select the providers you want to support during the Terraform installation process.
Answer: B
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 NO: 348
Your company has a lot of workloads in AWS , and Azure that were respectively created using CloudFormation , and AzureRM Templates. However , now your CIO has decided to use Terraform for all new projects , and has asked you to check how to integrate the existing environment with terraform code. What should be your next plan of action?
(A). Tell the CIO that this is not possible . Resources created in CloudFormation , and AzureRM templates cannot be tracked using terraform.
(B). Use terraform import command to import each resource one by one .
(C). This is only possible in Terraform Enterprise , which has the TerraformConverter exe that can take any other template language like AzureRM and convert to Terraform code.
(D). Just write the terraform config file for the new resources , and run terraform apply , the state file will automatically be updated with the details of the new resources to be imported.
Answer: B

QUESTION NO: 349
Which of the following are string functions? Select three
(A). tostring
(B). tonumber
(C). Chomp
(D). format
(E). join
Answer: C,D,E
tonumber and tostring are Type Conversion function
https://www.terraform.io/docs/configuration/functions.html

QUESTION NO: 350
Which Terraform command will force a marked resource to be destroyed and recreated on the next apply?
(A). terraform fmt
(B). terraform destroy
(C). terraform taint
(D). terraform refresh

Answer: C
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 and the next apply will implement this change.
Forcing the recreation of a resource is useful when you want a certain side effect of recreation that is not visible in the attributes of a resource. For example: re-running provisioners will cause the node to be different or rebooting the machine from a base image will cause new startup scripts to run.
Note that tainting a resource for recreation may affect resources that depend on the newly tainted resource. For example, a DNS resource that uses the IP address of a server may need to be modified to reflect the potentially new IP address of a tainted server. The plan command will show this if this is the case.
https://www.terraform.io/docs/commands/taint.html

QUESTION NO: 351
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). ERROR
(B). INFO
(C). WARN
(D). TRACE
(E). DEBUG
Answer: D
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.
Examples:
export TF_LOG=DEBUG
export TF_LOG=TRACE

QUESTION NO: 352
A Terraform provider is not responsible for:
(A). Understanding API interactions with some service
(B). Provisioning infrastructure in multiple clouds

(C). Exposing resources and data sources based on an API
(D). Managing actions to take based on resource differences
Answer: D

QUESTION NO: 353
Which of the following is not an action performed by terraform init?
(A). Create a sample main.tf file
(B). Initialize a configured backend
(C). Retrieve the source code for all referenced modules
(D). Load required provider plugins
Answer: A

QUESTION NO: 354
During a terraform plan, a resource is successfully created but eventually fails during provisioning. What happens to the resource?
(A). Terraform attempts to provision the resource up to three times before exiting with an error
(B). the terraform plan is rolled back and all provisioned resources are removed
(C). it is automatically deleted
(D). the resource is marked as tainted
Answer: D
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. Terraform also does not automatically roll back and destroy the resource during the apply when the failure happens, because that would go against the execution plan: the execution plan would've said a resource will be created, but does not say it will ever be deleted.

QUESTION NO: 355
You write a new Terraform configuration and immediately run terraform apply in the CLI using the local backend.
Why will the apply fail?
(A). Terraform needs you to format your code according to best practices first
(B). Terraform needs to install the necessary plugins first
(C). The Terraform CLI needs you to log into Terraform cloud first
(D). Terraform requires you to manually run terraform plan first
Answer: C

QUESTION NO: 356
Module variable assignments are inherited from the parent module and do not need to be explicitly set.
(A). True
(B). False
Answer: B

QUESTION NO: 357
What is not processed when running a terraform refresh?
(A). State file
(B). Configuration file
(C). Credentials
(D). Cloud provider
Answer: C,D

QUESTION NO: 358
What does this code do?

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


(A). Requires any version of the AWS provider > = 3.0 and < 4.0
(B). Requires any version of the AWS provider > = 3.0
(C). Requires any version of the AWS provider after the 3.0 major release like 4.1
(D). Requires any version of the AWS provider > 3.0
Answer: A

QUESTION NO: 359
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)
Answer: A,C,D,F
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 NO: 360

terraform validate validates the syntax of Terraform files.
(A). True
(B). False
Answer: A
The terraform validate command validates the syntax and arguments of the Terraform configuration files. Reference: https://www.terraform.io/docs/cli/code/index.html

QUESTION NO: 361
You're building a CI/CD (continuous integration/ continuous delivery) pipeline and need to inject sensitive variables into your Terraform run.
How can you do this safely?
(A). Pass variables to Terraform with a -var flag
(B). Copy the sensitive variables into your Terraform code
(C). Store the sensitive variables in a secure_vars.tf file
(D). Store the sensitive variables as plain text in a source code repository
Answer: A

QUESTION NO: 362
Which of the following type of variable allows multiple values of several distinct types to be grouped together as a single value?
(A). Map
(B). Object
(C). Tuple
(D). List
Answer: B,C
Structural type of variable allows multiple values of several distinct types to be grouped together as a single value. They require a schema as an argument, to specify which types are allowed for which elements.
https://www.terraform.io/docs/configuration/types.html

QUESTION NO: 363
True or False. The terraform refresh command is used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure. If drift is detected between the real-world infrastructure and the last known-state, it will modify the infrastructure to correct the drift.
(A). False
(B). True
Answer: A
https://www.terraform.io/docs/commands/refresh.html

QUESTION NO: 364
Terraform Enterprise currently supports running under which the following operating systems?
(A). Ubuntu
(B). Amazon Linux

(C). Debian
(D). CentOS
(E). Red Hat Enterprise Linux
(F). Oracle Linux
Answer: A,B,C,D,E,F
Terraform Enterprise runs on Linux instances, and you must prepare a running Linux instance for Terraform Enterprise before running the installer. You will start and manage this instance like any other server.
Terraform Enterprise currently supports running under the following operating systems:
Standalone deployment:
Debian 7.7+
Ubuntu 14.04.5 / 16.04 / 18.04
Red Hat Enterprise Linux 7.4 - 7.8
CentOS 6.x / 7.4 - 7.8
Amazon Linux 2014.03 / 2014.09 / 2015.03 / 2015.09 / 2016.03 / 2016.09 / 2017.03 / 2017.09 / 2018.03 / 2.0 Oracle Linux 7.4 - 7.8
https://www.terraform.io/docs/enterprise/before-installing/index.html

QUESTION NO: 365
What features stops multiple admins from changing the Terraform state at the same time?
(A). Version control
(B). Backend types
(C). Provider constraints
(D). State locking
Answer: D

QUESTION NO: 366
What command should you run to display all workspaces for the current configuration?
(A). terraform workspace
(B). terraform workspace show
(C). terraform workspace list
(D). terraform show workspace
Answer: C
terraform workspace list
The command will list all existing workspaces.

QUESTION NO: 367
Terraform variables and outputs that set the "description" argument will store that description in the state file.
(A). True
(B). False
Answer: B

QUESTION NO: 368
Running terraform fmt without any flags in a directory with Terraform configuration files will

check the formatting of those files without changing their contents.
(A). True
(B). False
Answer: B

QUESTION NO: 369
A terraform apply can not_____infrastructure.
(A). import
(B). provision
(C). destroy
(D). change
Answer: A

QUESTION NO: 370
By default, provisioners that fail will also cause the Terraform apply itself to error. How can you change this default behavior within a provisioner?
(A). provisioner "local-exec" { on_failure = "next" }
(B). provisioner "local-exec" { when = "failure" terraform apply }
(C). provisioner "local-exec" { on_failure = "continue" }
(D). provisioner "local-exec" { on_failure = continue }

Answer: C
https://www.terraform.io/docs/provisioners/index.html

QUESTION NO: 371
Terraform Enterprise (also referred to as pTFE) requires what type of backend database for a clustered deployment?
(A). PostgreSQL
(B). Cassandra
(C). MySQL
(D). MSSQL
Answer: A
External Services mode stores the majority of the stateful data used by the instance in an external PostgreSQL database and an external S3-compatible endpoint or Azure blob storage. There is still critical data stored on the instance that must be managed with snapshots. Be sure to check the PostgreSQL Requirements for information that needs to be present for Terraform Enterprise to work. This option is best for users with expertise managing PostgreSQL or users that have access to managed PostgreSQL offerings like AWS RDS.

QUESTION NO: 372
A variable az has the following default value. What will be the datatype of the variable?
az=["us-west-1a","us-east-1a"]
(A). Object
(B). List
(C). Map
(D). String

Answer: B

QUESTION NO: 373
If you manually destroy infrastructure, what is the best practice reflecting this change in Terraform?
(A). Run terraform refresh
(B). It will happen automatically
(C). Manually update the state fire
(D). Run terraform import
Answer: A

QUESTION NO: 374
When you use a remote backend that needs authentication. HashrCorp recommends that you:
(A). Push your Tefraform configuration to an encrypted git repository
(B). Write the authentication credentials in the Terraform configuration files
(C). Use partial configuration to load the authentication credentials outside of the Terraform code
(D). Keep the Terraform configuration files in a secret store
Answer: B

QUESTION NO: 375
Terraform requires the Go runtime as a prerequisite for installation.
(A). True
(B). False
Answer: B

QUESTION NO: 376
In the below configuration, how would you reference the module output vpc_id ?

```
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"
  cidr   = "10.0.0.0/16"
  name   = "test-vpc"
}
```


Type your answer in the field provided. The text field is not case-sensitive and all variations of the correct answer are accepted.
Answer:
modulevpcid

QUESTION NO: 377
Module version is required to reference a module on the Terraform Module Registry.
(A). True
(B). False
Answer: B

QUESTION NO: 378
Which of the following is an invalid variable name?
(A). count
(B). web
(C). var1
(D). instance_name
Answer: A
https://www.terraform.io/intro/examples/count.html

QUESTION NO: 379
True or False? By default, Terraform destroy will prompt for confirmation before proceeding.
(A). False
(B). True
Answer: B

QUESTION NO: 380
What is the result of the following terraform function call?
(A). hello
(B). what?
(C). goodbye
Answer: B
https://www.terraform.io/docs/configuration/functions/lookup.html

QUESTION NO: 381
You have to initialize a Terraform backend before it can be configured.
(A). True
(B). False
Answer: A

QUESTION NO: 382
What allows you to conveniently switch between multiple instances of a single configuration within its single backend?
(A). Local backends
(B). Providers
(C). Remote backends
(D). Workspaces
Answer: D
Named workspaces allow conveniently switching between multiple instances of a single configuration within its single backend. ... 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.
Workspaces, allowing multiple states to be associated with a single configuration. The configuration still has only one backend, but multiple distinct instances of that configuration to be deployed without configuring a new backend or changing authentication credentials.
https://www.terraform.io/docs/state/workspaces.html

QUESTION NO: 383
You wanted to destroy some of the dependent resources from real infrastructure. You choose to delete those resources from your configuration file and run terraform plan and then apply. Which of the following way your resources would be destroyed?
(A). Terraform can still determine the correct order for destruction from the state even when you delete one or more items from the configuration.
(B). Those would be destroyed in the order in which they were written in the configuration file previously before you have deleted them from configuration file.
(C). The resource will be destructed in random order as you have already deleted them from configuration.
(D). You can not destroy resources by deleting them from configuration file and running plan and apply.
Answer: A
Terraform typically uses the configuration to determine dependency order. However, when you delete a resource from a Terraform configuration, Terraform must know how to delete that resource. Terraform can see that a mapping exists for a resource not in your configuration and plan to destroy. However, since the configuration no longer exists, the order cannot be determined from the configuration alone.
To ensure correct operation, Terraform retains a copy of the most recent set of dependencies within the state. Now Terraform can still determine the correct order for destruction from the state when you delete one or more items from the configuration.

QUESTION NO: 384
Terraform can import modules from a number of sources - which of the following is not a valid source?
(A). FTP server
(B). GitHub repository
(C). Local path
(D). Terraform Module Registry
Answer: A