Hashicorp Terraform Associate (003) Exam Questions

Ashish Lahoti Last Modified: February 24, 2024 Cloud

PAGE CONTENT

Disclaimer:

Area 1: Infrastructure as Code (IaC)

Area 2: Terraform Cloud

Area 3: Terraform Configuration

Area 4: Terraform State

Area 5: Terraform Commands

Area 6: Terraform Backend

Area 7: Terraform Provisioners

Area 8: Terraform Providers

Area 9: Terraform Resources

Area 10: Terraform Variables and Outputs
Area 11: Terraform Module
Area 12: Terraform Security
Area 13: Terraform Workspace
Area 14: Terraform Version Constraint
Area 15: Terraform Types and Functions

A comprehensive list of Free 200+ unique practice exam questions for Hashicorp Terraform Associate Certification (003).

Disclaimer:

These are **not the Hashicorp Terraform official exam questions/dumps.** The questions are either created from the web content of the **official website** or consolidated from various sources. These **questions cover all the topics of the Hashicorp Terraform Associate (003) exam** and once you go through these questions and their concepts, you are more than ready to ace the exam with confidence.

Refer to the **link** for official exam details and objectives and this **link** for exam notes.

Please leave your feedback in the comment section.

Area 1: Infrastructure as Code (IaC)

Q1. What are the benefits of Infrastructure as Code tools like Terraform?

- [ ] Manage and track infrastructure
- [ ] Automate infrastructure changes
- [ ] Support Reusable configuration
- [ ] Collaboration using VCS (version control system)
- [ ] All of the above

Q2. Which one do you think is an advantage of using Terraform as the IaC tool?

- [ ] Terraform can manage infrastructure on multiple cloud platforms.
- [ ] Terraform's state allows you to track resource changes throughout your deployments.
- [ ] You can commit your configurations to version control to safely collaborate on infrastructure.
- [ ] All of the above

Q3. What are the benefits of using an "Infrastructure as Code" tool like Terraform? (select three)

- [ ] An imperative approach, offers flexibility to describe each step you want IaC to follow to reach the desired state
- [ ] You can commit your configurations to version control to safely collaborate on infrastructure
- [ ] The human-readable configuration language helps you write infrastructure code quickly.
- [ ] Terraform can manage infrastructure on multiple cloud platforms

Q4. Which is NOT a benefit of using an IaC tool like Terraform??

- [ ] You can commit your configurations to version control to safely collaborate on infrastructure
- [ ] You can manage infrastructure on multiple cloud platforms
- [ ] Infrastructure code is easy to maintain and understand without any Cloud knowledge
- [ ] The human-readable configuration language helps you write infrastructure code quickly

Q5. Does Terraform provide support for multiple cloud deployments?

- [ ] true
- [ ] false

Q6. How do you describe the Terraform precisely?

- [ ] A programming language
- [ ] An infrastructure as code (IaC) tool
- [ ] A cloud provider
- [ ] A containerization tool

Q7. Terraform is an Infrastructure as Code (IaC) tool that is? (select two)

- [ ] Imperative
- [ ] Declarative
- [ ] Cloud-agnostic
- [ ] Cloud-native

Q8. Which of the following describes the Terraform use case?

- [ ] Provisioning infrastructure in both Azure and AWS
- [ ] Enforce compliance and governance policies before any infrastructure changes
- [ ] Deploy a Kubernetes cluster and manage its resources
- [ ] All of the above

Q9. The activity of code provision and configuring your initial infrastructure in the Infrastructure Lifecycle is called?

- [ ] Day 0
- [ ] Day 1
- [ ] Sprint 0
- [ ] Sprint 1

Q10. Which characteristic of an IaC tool ensures the same infrastructure state, if the same code is applied multiple times?

- [ ] Consistent
- [ ] Repeatable
- [ ] Idempotent
- [ ] Predictable

Area 2: Terraform Cloud

Q11. What are the features of Terraform Cloud? (Choose 3 answers)

- [ ] Remote State Management
- [ ] Remote Terraform Execution
- [ ] Private Module Registry
- [ ] Terraform Linting

Q12. Which of the following features are available in Terraform Cloud Free Tier? (select three)

- [ ] Private Module Registry
- [ ] VCS Connection
- [ ] Team Management
- [ ] Application-level Logging
- [ ] Single Sign-On (SSO)

Q13. Which of the following features are NOT available in Terraform Cloud Free Tier? (select three)

- [ ] Audit Logging
- [ ] No-code provisioning
- [ ] Remote Terraform execution
- [ ] Private Module Registry
- [ ] Team Management

Q14. Which of the following Terraform features is only available in the Enterprise edition?

- [ ] Application-level Logging
- [ ] Single Sign-On (SSO)
- [ ] Drift Detection
- [ ] Audit Logging

Q15. Your company requires a self-hosted instance of Terraform Cloud with features like audit-logging and SAML single sign-on. Which Terraform plan you will choose?

- [ ] Terraform Cloud Free Tier
- [ ] Terraform Cloud Standard
- [ ] Terraform Cloud Plus
- [ ] Terraform Enterprise

Q16. VCS Connection in Terraform Cloud provides additional features such as automatically initiate Terraform runs when changes are committed. Which of the following VCS providers are supported by Terraform Cloud?

- [ ] Github.com
- [ ] Gitlab.com
- [ ] Bitbucket Cloud
- [ ] CVS Version Control

Q17. Which of the following is not a correct statement about version control system (VCS) in Terraform Cloud?

- [ ] Terraform Cloud can automatically initiate Terraform runs when changes are committed to the specified branch in VCS.
- [ ] Terraform Cloud makes code review easier by automatically predicting how pull requests will affect infrastructure.

- [ ] Github and Bitbucket VCS providers are supported by Terraform Cloud
- [ ] One VCS provider can be configured at a time in Terraform Cloud

Area 3: Terraform Configuration

Q18. How are the Core Terrform Workflow steps played out by an Individual Practitioner?

- [ ] Plan, Write, Apply
- [ ] Write, Plan, Apply
- [ ] Apply, Write, Plan
- [ ] Apply, Plan, Write

Q19. Which file is typically used to define resources in a Terraform configuration?

- [ ] main.tf
- [ ] terraform.tfvars
- [ ] variables.tf
- [ ] outputs.tf

Q20. What is the type of the block in below Terraform configuration?

```
resource "aws_instance" "example" {
  ami           = "abc123"
  instance_type = "t2.micro"
}
```

- [ ] `resource`
- [ ] `aws_instance`
- [ ] `t2.micro`
- [ ] `instance_type`

Q21. Which of the following is NOT a valid Terraform block type?

- [ ] provider
- [ ] resource
- [ ] output
- [ ] module
- [ ] data
- [ ] bucket

Q22. What is the workflow for deploying new infrastructure with Terraform?

- [ ] terraform plan to import the current infrastructure to the state file, make code changes, and terraform apply to update the infrastructure.
- [ ] Write a Terraform configuration, run terraform show to view proposed changes, and terraform apply to create new infrastructure.
- [ ] terraform import to import the current infrastructure to the state file, make code changes, and terraform apply to update the infrastructure.
- [ ] Write a Terraform configuration, run terraform init, run terraform plan to view planned infrastructure changes, and terraform apply to create new infrastructure.

Q23. Terraform configuration can be written in which language? (select two)

- [ ] XML
- [ ] JSON

- [ ] Hashicorp Configuration Language (HCL)
- [ ] YAML

Q24. Terraform language is widely expressed in Hashicorp Configuration Language (HCL) syntax. Which other syntax is supported?

- [ ] XML
- [ ] JSON
- [ ] YAML
- [ ] Protobuf

Q25. How many Indent spaces are required at each nested level in the Terraform language style convention?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4

Q26. Which of the statements accurately describes the Terraform language? (select two)

- [ ] Terraform is declarative Infrastructure as Code provisioning language based on Hashicorp Configuration Language, or optionally YAML.
- [ ] Terraform is declarative Infrastructure as Code provisioning language based on Hashicorp Configuration Language, or optionally JSON.
- [ ] Code in the Terraform language is stored in plain text files with the `.tf` file extension, or optionally with the `.tf.json` file extension
- [ ] Code in the Terraform language is stored in plain text files with the `.tf` file extension, or optionally with the `.tf.yml` file extension

Q27. Terraform binary is available to download and install on different operating systems and platforms. Which of the following operating systems are supported?

- [ ] macOS
- [ ] Solaris
- [ ] FreeBSD
- [ ] Windows
- [ ] All of the above

Q28. Which Terraform files should be excluded from Git commit by adding them in `.gitingnore` file? (select two)

- [ ] `output.tf`
- [ ] `terraform.tfstate`
- [ ] `terraform.tfvars`
- [ ] `variables.tf`

Q29. You want to deploy multiple subnets within a VPC in AWS and looking for a `for` loop-like expression to iterate over a list of subnet CIDR blocks. Which feature of Terraform configuration you can use?

- [ ] terraform import
- [ ] dynamic blocks
- [ ] splat expression
- [ ] dynamic backend

Q30. How do you describe a `dynamic` block?

- [ ] It requests that Terraform read from a given data source and export the result under the given local name
- [ ] It declares a resource of a given type with a given local name
- [ ] It iterates over a given complex value, and generates a nested block for each element of that complex value
- [ ] It exports a value exported by a module or configuration

Q31. Which one of the following is considered as a Terraform plugin?

- [ ] Terraform provisioner
- [ ] Terraform module
- [ ] Terraform provider
- [ ] Terraform registry

Q32. Terraform remembers the compatible version of dependencies such as providers and modules through dependency lock file. What is the name of that file?

- [ ] `.terraform.lock.hcl`
- [ ] `.terraform.lock.tf`
- [ ] `.dependency.lock.hcl`
- [ ] `.dependency.lock.tf`

Q33. Terraform Core is a statically-compiled binary written in the _____ programming language.

- [ ] Java
- [ ] C#
- [ ] Python
- [ ] Go

Q34. Terraform builds a dependency graph from the Terraform configurations. Which is NOT a correct step of building a Graph?

- [ ] Resources are mapped to provisioners if they have any defined.
- [ ] Resources are mapped to providers.

- [ ] Resources nodes are added to the graph from the configuration.
- [ ] Resources are not added to the graph that are no longer present in the configuration but are present in the state file.

Area 4: Terraform State

Q35. Which of the following best describes a Terraform state file?

- [ ] A file that contains a list of available Terraform providers
- [ ] A file that stores the current state of infrastructure managed by Terraform
- [ ] A file that contains a list of Terraform modules used in a configuration
- [ ] A file that stores the output of a Terraform plan

Q36. The Terraform state always matches to the remote cloud resources defined in the configuration?

- [ ] true
- [ ] false

Q37. *Usernames and passwords referenced in the Terraform code, even as variables, will end up in plain text in the state file.

- [ ] true
- [ ] false

Q38. Terraform can inspect the real-world resources on every run to validate the desired state when the state file is missing.

- [ ] true
- [ ] false

Q39. You injected some secrets from variables into your Terraform configuration. What happens after you run the `terraform apply` command and they are loaded into state?

- [ ] They are shown in clear-text.
- [ ] They are shown as their referenced variables.
- [ ] They are shown as encrypted values.
- [ ] They are omitted from state.

Q40. 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.

- [ ] *terraform.tfstate*

Q41. Where is the default location that Terraform stores its state in?

- [ ] The current working directory in which Terraform was run.
- [ ] At the users root directory.
- [ ] In the same location that Terraform is installed. E.g. /usr/bin/terraform
- [ ] In ~/.terraform.d/plugins

Q42. What is the recommended way to implement Terraform's state for larger teams?

- [ ] By configuring a remote backend such that multiple teams can work in tandem and know which resources are being created and destroyed.
- [ ] By sticking the state in a cloud instance, and having team members SSH into the instance to work on their configuration files.
- [ ] Having your state synced to a github repo for members to compare to.
- [ ] By using the daily standup you are a part of sot that you can share changes to the state file.

Q43. You are part of a large DevOps team using the current version of Terraform, and there can be multiple changes going on to your terraform files across the company. What would you do to ensure that the state file is locked when you run `terraform apply`?

- [ ] Add the -lock=true flag to the command.
- [ ] Nothing, terraform will manage the locking by itself.
- [ ] First run terraform plan to lock in your proposed changes. Then run terraform apply to commit them.
- [ ] Add the -state-lock=true to the command.

Q44. If supported by your backend, Terraform automatically locks the state for all operations that could write state. What is the purpose of state locking?

- [ ] Prevents others from committing Terraform code that could override your updates.
- [ ] Locks colleagues from making manual changes to the managed infrastructure
- [ ] This prevents others from acquiring the lock and potentially corrupting your state.
- [ ] Ensures the state file cannot be moved after the initial `terraform apply`

Q45. Which of the following Terraform backend type supports state locking?

- [ ] consul
- [ ] kubernetes

- [ ] s3
- [ ] All of the above

Q46. You manage the AWS cloud resources using Terraform and want to destroy all dev resources to save cost. However, your team member request you to keep the Amazon Aurora dev instance running. How can you destroy all cloud resources without impacting the database instance?

- [ ] run `terraform state rm` command to remove the database instance from terraform state before running `terraform destroy` command
- [ ] take a snapshot of database, run `terraform destroy`, and then recreate database by restoring the snapshot
- [ ] run a `terraform destroy`, modify configuration file to include only database instance, and then run `terraform apply`
- [ ] manually delete the other resource from AWS

Q47. You manage the AWS cloud resources using Terraform and want to follow new naming standard for the local name within resource block. However, you don't want Terraform to replace the object after changing your configuration files. How can you change the local name from `data-bucket` to `prod-aws-s3-bucket` in the following resource block:-

```
resource "aws_s3_bucket" "data-bucket" {
  bucket = "prod-data-bucket"

  tags = {
    Name        = "prod-data-bucket"
    Environment = "prod"
  }
}
```

After renaming the local name of the resource block, what command would you run to update the local name while ensuring Terraform does not replace the existing resource?

- [ ] `terraform apply -refresh-only`
- [ ] `terrafrorm apply -replace aws_s3_bucket.data-bucket`
- [ ] `terraform state mv aws_s3_bucket.data-bucket aws_s3_bucket.prod-aws-s3-bucket`
- [ ] `terraform state rm aws_s3_bucket.data-bucket`

Q48. Which Terrafrom command only reads the state file and won't cause any refresh or modification of state file?

- [ ] `terraform state list`
- [ ] `terraform plan`
- [ ] `terraform apply`
- [ ] `terraform destroy`

Area 5: Terraform Commands

Q49. Which command need to run before Terraform plan or apply if you have added a new resource block in the configuration from different provider?

- [ ] `terraform refresh`
- [ ] `terraform eval`
- [ ] `terraform init`
- [ ] `terraform validate`

Q50. How can you check out the configuration from version control and initialize a directory?

- [ ] `terraform init -from-module={MODULE-SOURCE}`
- [ ] `terraform init -source={PATH}`
- [ ] `terraform init {PATH}`
- [ ] `terraform init -plugin-dir={PATH}`

Q51. Whenever you add a new module to a configuration, Terraform must install the module before it can be used. Which two commands will install and update modules? (select two)

- [ ] `terraform plan`
- [ ] `terraform refresh`
- [ ] `terraform init`
- [ ] `terraform get`

Q52. Which command is used to create an execution plan in Terraform?

- [ ] `terraform plan`
- [ ] `terraform apply`
- [ ] `terraform init`
- [ ] `terraform validate`

Q53. By default, when running `terraform plan`, what files are scanned?

- [ ] All *.tf files in the current directory.
- [ ] Only files in the .terraform directory
- [ ] Only files you specify with the -file-path flag.
- [ ] All files on your hard drive.

Q54. Which command is used to apply changes to infrastructure in Terraform?

- [ ] `terraform destroy`
- [ ] `terraform apply`
- [ ] `terraform plan`
- [ ] `terraform validate`

Q55. Which command is used to destroy infrastructure in Terraform?

- [ ] `terraform destroy`
- [ ] `terraform apply`
- [ ] `terraform plan`
- [ ] `terraform validate`

Q56. Which Terraform command can be used to delete all of your managed infrastructure? (select two)

- [ ] `terraform init -destroy`
- [ ] `terraform apply -destroy`
- [ ] `terraform destroy`
- [ ] `terraform plan -destroy`

Q57. If different teams are working on the same configuration. How do you make files to have consistent formatting?

- [ ] `terraform fmt`
- [ ] `terraform apply`
- [ ] `terraform plan`
- [ ] `terraform validate`

Q58. What Terraform command modifies a HashiCorp Configuration Language (HCL) file to adhere to the recommended spacing rules for HCL files?

- [ ] `terraform fmt`
- [ ] `terraform apply`
- [ ] `terraform plan`
- [ ] `terraform validate`

Q59. Your teammate is worried that if they run the terraform fmt command on their current directory, it will change their configuration files too much. What flag do you tell them to pass into the command such that they can see the differences?

- [ ] `-diff`
- [ ] `-check`

- [ ] `-refresh`
- [ ] `-list=true`

Q60. Your organization want to ensure that all Terraform configuration files follows the Terraform language format and style convention. You have `main.tf` in the current directory, which calls modules stored in `module` directory. Which command can be used to format all the configuration files in the current directory and all subdirectories?

- [ ] `terraform fmt -check -recursive`
- [ ] `terraform fmt -diff`
- [ ] `terraform fmt -check`
- [ ] `terraform fmt -list=true`

Q61. Which command can be used to verify whether a configuration is syntactically valid and internally consistent?

- [ ] `terraform validate`
- [ ] `terraform apply`
- [ ] `terraform plan`
- [ ] `terraform fmt`

Q62. How would you get the JSON output of the `terraform validate` command?

- [ ] `terraform validate -json`
- [ ] `terraform validate json`
- [ ] `terraform validate -output=json`
- [ ] `terraform json validate`

Q63. Does the `terraform validate` command connect to remote APIs and state when being ran?

- [ ] No it does not.
- [ ] Only if configured to do so on the backend.
- [ ] If the -remote=true is set, yes it does.
- [ ] If there are providers set, it will attempt to.

Q64. Which command provides an interactive command-line console for evaluating and experimenting with expressions?

- [ ] `terraform show`
- [ ] `terraform eval`
- [ ] `terraform console`
- [ ] `terraform exec`

Q65. Which command is used to extract the value of an output variable from the state file?

- [ ] `terraform exec`
- [ ] `terraform show`
- [ ] `terraform output`
- [ ] `terraform state`

Q66. *You have defined the values for your variables in the file `terraform.tfvars`, and saved it in the same directory as your Terraform configuration. Which of the following commands will use those values when creating an execution plan?

- [ ] `terraform plan`
- [ ] `terraform plan -var-file=terraform.tfvars`

- [ ] All of the above
- [ ] None of the above

Q67. Which two steps are required to provision new infrastructure in the Terraform workflow? Choose *TWO* correct answers.

- [ ] `terraform init`
- [ ] `terraform import`
- [ ] `terraform apply`
- [ ] `terraform validate`
- [ ] `terraform destroy`

Q68. It is necessary to run `terraform plan` command before `terraform apply` in the Terraform Workflow.

- [ ] true
- [ ] false

Q69. Someone created few resources manually using the Azure console. You have company policy to manage all infrastructure using Terraform. How can you manage manually deployed resource using Terraform without impacting other resources?

- [ ] run a `terraform get` to get the manually deployed resources that are not under Terraform management
- [ ] delete the resources created manually using the Azure console and add these resource in terraform configuration, then run `terraform apply`
- [ ] use `terraform import` to import existing resources under Terraform management
- [ ] resources created outside Terraform cannot be managed by Terraform

Q70. How does `terraform import` run?

- [ ] As a part of `terraform init`
- [ ] As a part of `terraform plan`
- [ ] As a part of `terraform refresh`
- [ ] By an explicit call
- [ ] All of the above

Q71. What must be provided with the `terraform import` command for Terraform to successfully import resources?

- [ ] Resource ID, resource type, and the resource name.
- [ ] The resource name.
- [ ] The full resource ARN.
- [ ] Only resource Id

Q72. Your team is currently using Terraform to provision the infrastructure in AWS. Someone in your team has manually created an EC2 instance in AWS. You now want to bring this EC2 instance under Terraform management. What must you do before running the `terraform import` to manage these resource using Terraform?

- [ ] manually write a `resource` block in Terraform configuration file that match the resource you want to import
- [ ] manually modify the Terraform state file to add the new resources so Terraform will have a record of the resources to be managed
- [ ] shut down or stop using the resources being imported so no changes are inadvertently missed
- [ ] run `terraform apply -refresh-only` to ensure that the state file has the latest information for existing resources.

Q73. A user wants to list all resources which are deployed using Terraform. How can this be done?

- [ ] `terraform state show`
- [ ] `terraform state list`
- [ ] `terraform show`
- [ ] `terraform show list`

Q74. Which terraform state subcommand will give you all of the resources in your state?

- [ ] list
- [ ] show
- [ ] refresh
- [ ] apply

Q75. A user wants to see the resource block for resource `aws_instance` having name `foo` in state file. How can this be done?

- [ ] `terraform show aws_instance.foo`
- [ ] `terraform show aws_instance foo`
- [ ] `terraform state show aws_instance.foo`
- [ ] `terraform state show aws_instance foo`

Q76. Which of the following command provides the JSON reprentation of the state?

- [ ] `terraform state -json`
- [ ] `terraform state show -json`
- [ ] `terraform show -json`
- [ ] `terraform show state -json`

Q77. Why would you use the `terraform taint` command?

- [ ] When you want to force Terraform to destroy a resource on the next apply
- [ ] When you want to force Terraform to destroy and recreate a resource on the next apply
- [ ] When you want Terraform to ignore a resource on the next apply
- [ ] When you want Terraform to destroy all the infrastructure in your workspace

Q78. The command `terraform.taint` is deprecated in v0.15.2, which command you should use intead?

- [ ] `terraform apply -replace`
- [ ] `terraform plan -replace`
- [ ] `terraform apply -taint`
- [ ] `terraform plan -taint`

Q79. You want Terrafrorm to destory and recreate a particular AWS instance `test_123` that was deployed with a bunch of other AWS instances. Which command can be used to accomplish this task without modifying the Terraform code?

- [ ] `terraform apply -replace=aws_instance.test_123`
- [ ] `terraform apply -destroy=aws_instance.test_123`
- [ ] `terraform state recreate aws_instance.test_123`
- [ ] `terraform state destroy aws_instance.test_123`

Q80. You have an EC2 instance that is acting up in the cloud. It handles a relatively light ephemeral workload, so it can be restarted/destroyed with no repercussions. What full command would you use to target only this instance for recreation?

- [ ] `terraform apply -replace=aws_instance.{INSTANCE_NAME}`
- [ ] `terraform apply -replace aws_instance`

- [ ] `terraform apply -replace {INSTANCE_NAME}`
- [ ] `terraform destroy --target=aws.instance{INSTANCE_NAME}` and `terraform apply`

Q81. What is not processed when running a `terraform refresh`?

- [ ] State file
- [ ] Configuration file
- [ ] Credentials
- [ ] Cloud provider

Q82. *Which of the following Terraform commands will automatically refresh the state unless supplied with additional flags or arguments? Choose *TWO* correct answers.

- [ ] `terraform plan`
- [ ] `terraform state`
- [ ] `terraform apply`
- [ ] `terraform validate`
- [ ] `terraform output`

Q83. The command `terraform refresh` is deprecated in v0.15.4, which command is recommended to use instead? Choose *TWO* correct answers.

- [ ] `terraform apply -refresh-only`
- [ ] `terraform plan -refresh-only`
- [ ] `terraform apply -refresh`
- [ ] `terraform plan -refresh`

Q84. Which of the following command will give you an opportunity to review the changes that Terraform has detected during refresh? Choose *TWO* correct answers.

- [ ] `terraform apply -refresh-only -auto-approve`
- [ ] `terraform apply -refresh-only`
- [ ] `terraform refresh`
- [ ] `terraform plan -refresh-only`

Q85. What happens when you apply Terraform configuration? Choose *TWO* correct answers.

- [ ] Terraform makes any infrastructure changes defined in your configuration.
- [ ] Terraform gets the plugins that the configuration requires.
- [ ] Terraform updates the state file with any configuration changes it made.
- [ ] Terraform corrects formatting errors in your configuration.
- [ ] Terraform destroys and recreates all your infrastructure from scratch.

Q86. *Which flag is used to find more information about a Terraform command? For example, you need additional information about how to use the `plan` command. You would type: `terraform plan _____` .

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

- [ ] `-h`
- [ ] `-help`
- [ ] `--help`

Answers that would also receive full credit:

`--h`

`terraform plan -h`

`terraform plan --h`

`terraform plan -help`

`terraform plan --help`

`terraform -h plan`

`terraform -help plan`

`terraform --help plan`

`plan -h`

`plan --h`

`plan -help`

`plan --help`

`-h plan`

`-help plan`

`--help plan`

Q87. Which flag would you add to terraform plan to save the execution plan to a file? You would type: `terraform plan _____`

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

- [ ] `-out=FILENAME`

Q88. You just added a new set of resources to your configuration and would only like to see them when you run your `terraform plan` command. What flag do you specify when running the `terraform plan` command to only see their plans?

- [ ] `-target={resources}`
- [ ] `-refresh=true`

- [ ] `-state={new_state_file}`
- [ ] `-lock=true`

Q89. 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?

- [ ] Terraform will remove the VM from state file
- [ ] Terraform will report an error
- [ ] Terraform will not make any changes
- [ ] Terraform will recreate the VM

Q90. 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?

- [ ] Run the `terraform fmt` command during the code linting phase of your CI/CD process
- [ ] Designate one person in each team to review and format everyone's code
- [ ] Manually apply two spaces indentation and align equal sign "=" characters in every Terraform file (*.tf)
- [ ] Write a shell script to transform Terraform files using tools such as AWK, Python, and sed

Q91. You have deployed a new webapp with a public IP address on a cloud 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

Q92. Terraform uses parallelism to walk through the dependency graph during `terraform apply` to provision the resources. What is the default value of parallelism?

- [ ] 100
- [ ] 10
- [ ] 5
- [ ] 1

Q93. Say you wanted to increase the number of operations that terraform is concurrently using to create your resources. Which command would you run, with what specific flag, to accomplish this? (Choose 2 answers)

- [ ] `terraform apply`
- [ ] `-parallelism={NUMBER-OF-OPERATIONS}`
- [ ] `terraform init`
- [ ] `-concurrent={NUMBER-OF-OPERATIONS}`

Q94. Lately you noticed that your Terraform jobs are failing in your CI/CD pipeline. The error that is coming back mentions something about hitting a rate limit. Without altering the time that the builds are ran, what could you pass into the terraform apply command to slow your operations down?

- [ ] `-parallelism={NUMBER_OF_OPERATIONS}`
- [ ] `-concurrent={NUMBER_OF_OPERATIONS}`
- [ ] `-rate-limit={NUMBER_OF_OPERATIONS}`
- [ ] `-refresh=false`

Q95. Which Terraform command is used to manually unlock the state, if unlocking failed?

- [ ] `terraform unlock`
- [ ] `terraform force-unlock`
- [ ] `terraform manual-unlock`
- [ ] `terrafom state unlock`

Area 6: Terraform Backend

Q96. What does the default `local` Terraform backend store?

- [ ] `*.tfplan` files
- [ ] Terraform binary
- [ ] Provider plugins
- [ ] `terraform.tfstate` file

Q97. What two configuration variables are available to a default local backend? (Choose 2 answers)

- [ ] `path`
- [ ] `workspace_dir`
- [ ] `working_dir`
- [ ] `path_dest`

Q98. What is NOT true about the Terraform backend?

- [ ] A backend is where Terraform stores its state data files.
- [ ] By default, Terraform uses a backend called local, which stores state as a local file on disk.
- [ ] A terraform configuration can only provide one backend block.
- [ ] A backend block can refer to named values (like input variables, locals, or data source attributes).

Q99. How is the Terraform `remote` backend different than other state backends such as `s3`, `http` and `consul`, etc.?

- [ ] It can execute Terraform runs on dedicated infrastructure on premises or in Terraform Cloud
- [ ] It doesn't show the output of a terraform apply locally
- [ ] It is only available to enterprise customers
- [ ] All of the above

Q100. How do you supply remaining arguments to a partial backend configuration? (Choose 2 answers)

- [ ] Specify file `terraform init -backend-config=PATH`
- [ ] Specify key/value pairs `terraform init -backend-config="KEY=VALUE"`
- [ ] Environment variable `export TF_VAR_key=value`
- [ ] Set variable `terraform init -var="KEY=VALUE"`

Q101. You are a part of a growing Cloud Infrastructure team. Your boss asks you to transition the team off of local backends, and onto remote backends. Within Terraform, what do you do to use the S3 buckets as a remote backend? (Choose 2 answers)

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

- [ ] Specify the key to store state file inside the S3 bucket
- [ ] Make sure Terraform gets AWS IAM permission on target backend bucket and stored state file
- [ ] Export your AWS API key to TF_BACKEND_KEY
- [ ] Encrypt your AWS buckets with SSE.

Q102. Which of the following is a type of backend configurable in Terraform?

- [ ] local
- [ ] standard
- [ ] enhanced
- [ ] advanced

Q103. Which of the following is NOT a supported backend type?

- [ ] http
- [ ] bitbucket
- [ ] local
- [ ] s3

Q104. All standard backend types support remote state storage, state locking, and encryption at rest?

- [ ] true
- [ ] false

Q105. Which of the following backend type doesn’t support state locking?

- [ ] local
- [ ] s3
- [ ] remote
- [ ] artifactory

Q106. Which of the following backend type doesn't support remote state storage?

- [ ] remote
- [ ] Terraform Cloud
- [ ] github
- [ ] artifactory

Q107. Your team has decided to move the Terraform state to remote backend for collaboration so that multiple people can access it. Your co-worker has decided to migrate Terraform state to a remote backend. Accessing remote state requires access credentials, where should you store these credentials? (select two)

- [ ] in a variable
- [ ] in credentials file
- [ ] hardcode in the configuration
- [ ] in environment variables

Q108. If you want to migrate the backend configuration from "consul" to "s3", you have to remove all the resources managed by "consul" remote backend first.

- [ ] false
- [ ] true

Q109. You have decided to migrate the Terraform state to a remote s3 backend. You have added the `backend` block in the Terraform configuration. Which command you should run to migrate the state?

```
terraform {
  backend "s3" {
    bucket = "terraform-s3-bucket-name"
    key    = "s3 key path"
    region = "us-west-1"
  }
}
```

- [ ] `terraform init`
- [ ] `terraform push`
- [ ] `terraform apply`
- [ ] `terraform plan`

Area 7: Terraform Provisioners

Q110. Provisioners add a considerable amount of complexity and uncertainty to Terraform usage and should be used as a last resort. True or false?

- [ ] true
- [ ] false

Q111. You want to execute a script on the provisioned AWS instance. You can add a `provisioner` block inside which block type to achieve this?

- [ ] `terraform` block
- [ ] `data` block
- [ ] `provider` block
- [ ] `resource` block

Q112. Which option will you use to run provisioners that are not associated with any resources?

- [ ] null_resource
- [ ] file
- [ ] local-exec
- [ ] remote-exec

Q113. Which provisioner copies files or directories from the machine running Terraform to the newly created resource?

- [ ] null_resource
- [ ] file
- [ ] local-exec
- [ ] remote-exec

Q114. Which type of connections supported by file provisioner? Select all valid options.

- [ ] ssh
- [ ] sftp
- [ ] winrm
- [ ] rdc

Q115. Which provisioner invokes a process on the machine running Terraform, not on the resource?

- [ ] null_resource
- [ ] file
- [ ] local-exec
- [ ] remote-exec

Q116. Where does the 'local-exec' provisioner execute its code provided in its block?

- [ ] On the remote resource specified.
- [ ] On the local machine running terraform.
- [ ] On a spot-instance on your cloud provider.
- [ ] In a container on your machine provided by the Terraform binary.

Q117. Which provisioner invokes a process on the resource created by Terraform?

- [ ] null_resource
- [ ] file
- [ ] local-exec
- [ ] remote-exec

Q118. What are the two accepted values for provisioners that have the “on_failure” key specified? (Choose 2 answers)

- [ ] continue
- [ ] fail
- [ ] abort
- [ ] retry

Q119. What does the following provisioner block specify?

```
provisioner "local-exec" {
  when    = destroy
  command = "echo 'Destroy-time provisioner'"
}
```

- [ ] Before the resource is destroyed, the provisioner will invoke “echo ‘Destroy-time provisioner’”
- [ ] If the resource receives a ‘destroy’ command locally, it will echo ‘Destroy-time provisioner’
- [ ] After the resource is destroyed, it will invoke “echo ‘Destroy-time provisioner’”
- [ ] On the next ‘terraform apply’ the resource will be destroyed

Area 8: Terraform Providers

Q120. How do you describe a Terraform provider?

- [ ] A collection of resources that can be used to define a specific piece of infrastructure
- [ ] A plugin that allows Terraform to interact with a specific cloud provider or service
- [ ] A tool for managing Docker containers
- [ ] A set of variables used to configure Terraform resources

Q121. Which of the following is NOT true of Terraform providers?

- [ ] Providers can be written by individuals
- [ ] Providers can be maintained by a community of users
- [ ] Some providers are maintained by HashiCorp
- [ ] Major cloud vendors and non-cloud vendors can write, maintain, or collaborate on Terraform providers
- [ ] None of the above

Q122. A provider configuration block is required in every Terraform configuration.

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

- [ ] true
- [ ] false

Q123. Official Terraform providers are owned and maintained by HashiCorp.

- [ ] true
- [ ] false

Q124. Which provider configuration can be used to define multiple aws provider with different regions?

- [ ] provider
- [ ] source
- [ ] region
- [ ] alias

Q125. What is a provider block without an `alias` meta argument?

- [ ] The default provider configuration.
- [ ] A broken provider configuration.
- [ ] A partial provider configuration.
- [ ] There must be an alias meta argument.

Q126. How do you select the alternate aws provider for us-west-2 region?

```
# 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"
}
```

- [ ] `resource "aws_instance" "foo" { provider = aws }`
- [ ] `resource "aws_instance" "foo" { provider = aws.west }`
- [ ] `resource "aws_instance" "foo" { provider = aws.us-west-2 }`
- [ ] `resource "aws_instance" "foo" { provider = west }`

Q127. Terraform uses a lock file to ensure predictable runs when using ambiguous provider version constraints. How do you update the lock file?

- [ ] `terraform providers lock`
- [ ] `terraform lock`
- [ ] `terraform apply lock`
- [ ] `terraform lock provider -provider={PROVIDER_NAME}`

Q128. Terraform is running in an isolated network without access to Terraform registry. How can you configure Terraform to consult only a local filesystem mirror to download plugins?

- [ ] `terraform providers mirror`
- [ ] `terraform mirror`
- [ ] `terraform providers local`
- [ ] `terraform plugins mirror`

Q129. While creating a terraform module, which configuration under `terraform` block can be used to specify the requirement of a specific version of a provider?

- [ ] `required_providers`
- [ ] `required_provider`
- [ ] `required_versions`
- [ ] `required_version`

Area 9: Terraform Resources

Q130. Who is the provider for the below resource?

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

- [ ] vpc
- [ ] main
- [ ] aws
- [ ] test

Q131. What is the name assigned by Terraform to reference this resource?

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

- [ ] computer_instance
- [ ] main

- [ ] google
- [ ] test

Q132. 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 = "t3.micro"
}
```

- [ ] aws_ami.ubuntu
- [ ] data.aws_ami.ubuntu
- [ ] data.aws_ami.ubuntu.id
- [ ] aws_ami.ubuntu.id

Q133. What's the correct syntax for referencing a resource within the configuration file?

- [ ] `<RESOURCE TYPE>.<NAME>`
- [ ] `<NAME>.<RESOURCE TYPE>`
- [ ] `<PROVIDER>.<RESOURCE TYPE>`
- [ ] `<LOCAL/REMOTE STATE>.<RESOURCE TYPE>`

Q134. Which of the following is an implicit dependency in the below example?

```
resource "aws_s3_bucket" "financial_report" { }

resource "aws_instance" "banking_service" {
  ami           = data.aws_ami.amazon_linux.id
  instance_type = "t2.micro"
  depends_on    = [aws_s3_bucket.financial_report]
}
```

- [ ] S3 Bucket `financial_report`
- [ ] EC2 Instance Type `t2.micro`
- [ ] AMI `amazon_linux`
- [ ] EC2 Instance `banking_service`

Q135. Which type of dependency is created in the below example using `depends_on` argument?

```
resource "aws_s3_bucket" "financial_report" { }

resource "aws_instance" "banking_service" {
  ami           = data.aws_ami.amazon_linux.id
  instance_type = "t2.micro"
  depends_on    = [aws_s3_bucket.financial_report]
}

module "example_sqs_queue" {
  source        = "terraform-aws-modules/sqs/aws"
```

```
  version       = "3.3.0"
  depends_on    = [aws_s3_bucket.financial_report, aws_instance.banking_service]
}
```

- [ ] internal dependency
- [ ] implicit dependency
- [ ] external dependency
- [ ] explicit dependency

Q136. What is the syntax to correctly reference a data source?

- [ ] `data.<DATA TYPE>.<NAME>`
- [ ] `data.<NAME>`
- [ ] `data.<NAME>.<DATA TYPE>`
- [ ] `<DATA TYPE>.<NAME>.data`

Q137. You want to use the `terraform state show` to see the attributes of a single resource created by the `for_each` in below resource block. What resource address should be used for the instance related to `vault` ?

```
resource "aws_instance" "demo" {
  # ...
  for_each = {
    "terraform": "infrastructure",
    "vault":     "security",
    "consul":    "connectivity",
    "nomad":     "scheduler",
  }
}
```

- [ ] `aws_instance.demo[1]`
- [ ] `aws_instance.demo["2"]`
- [ ] `aws_instance.demo.vault`
- [ ] `aws_instance.demo["vault"]`

Q138. How can you obtain a list of all of the `device_name` values from `ebs_block_device` nested blocks, that are created by this `resource` block?

```
resource "aws_instance" "example" {
  ami           = "ami-abc123"
  instance_type = "t2.micro"

  ebs_block_device {
    device_name = "sda2"
    volume_size = 16
  }
  ebs_block_device {
    device_name = "sda3"
    volume_size = 20
  }
}
```

- [ ] `aws_instance.example.ebs_block_device[*].device_name`
- [ ] `aws_instance.example.ebs_block_device[0,1].device_name`
- [ ] `aws_instance.example.*.device_name`
- [ ] `aws_instance.*.*.device_name`

Q139. Terraform only manage the dependencies between resources when the `depends_on` argument is specified.

- [ ] true
- [ ] false

Area 10: Terraform Variables and Outputs

Q140. How can you set the value to a variable “region” declared in the configuration file?

- [ ] Using command line `terraform apply -var="region=us-east-1"`
- [ ] Using variable file `terraform apply -var-file="variables.tfvars"` where the file contains: region=us-east-1
- [ ] Using environment variable `export TF_VAR_region=us-east-1`
- [ ] All of the above

Q141. Which one of the following takes higher precedence in loading variable in Terraform?

- [ ] Command line flag - `terraform apply -var="region=us-east-1"`
- [ ] Configuration file - set in your `terraform.tfvars` file
- [ ] Environment variable - `export TF_VAR_region=us-east-1`
- [ ] Default Config - default value in `variables.tf`

Q142. Which of the following is an invalid argument for defining input variable in Terraform?

- [ ] default
- [ ] type
- [ ] description
- [ ] validation
- [ ] sensitive

- [ ] nullable
- [ ] depends_on

Q143. How would you configure your input variable to fallback to a pre-declared value in your variable block?

- [ ] By specifying the default meta-argument.
- [ ] By specifying the fallback meta-argument.
- [ ] Terraform has a list of fallbacks that it will always implement if nothing is specified. E.g. aws_instance will fall back to a t2.micro if the size is not specifed.
- [ ] Terraform will ask you to set a fallback when you run the terraform apply command.

Q144. You defined a variable and would like to reference it in your terraform configuration file. What is the syntax required to do so?

- [ ] `var.<VARIABLE_NAME>`
- [ ] `<VARIABLE_NAME>.var`
- [ ] `var.<VARIABLE_NAME>.<RESOURCE_NAME>`
- [ ] `<RESOURCE_NAME>.var.<VARIABLE_NAME>`

Q145. *Consider the following configuration snippet: How would you define the `cidr_block` for `us-east-1` in the aws_vpc resource using a variable?

```
variable "vpc_cidrs" {
  type = map
  default = {
    us-east-1 = "10.0.0.0/16"
    us-east-2 = "10.1.0.0/16"
```

```
    us-west-1 = "10.2.0.0/16"
    us-west-2 = "10.3.0.0/16"
  }
}

resource "aws_vpc" "shared" {
  cidr_block = _____________
}
```

- [ ] `var.vpc_cidrs["us-east-1"]`
- [ ] `var.vpc_cidrs.0`
- [ ] `vpc_cidrs["us-east-1"]`
- [ ] `var.vpc_cidrs[0]`

Q146. A new variable `fruits` has been created of type `list` as shown below. How would you reference `banana` in your configuration?

```
variable "fruits" {
    type = list(string)
    default = [
        "mango",
        "apple",
        "banana",
        "orange",
        "grapes"
    ]
}
```

- [ ] `var.fruits[2]`
- [ ] `var.fruits.banana`
- [ ] `var.list.fruits[2]`
- [ ] `var.fruits[3]`

Q147. A Terraform local value can reference other Terraform local values?

- [ ] true
- [ ] false

Q148. When are output variables ran and sent to stdout?

- [ ] Only with terraform apply.
- [ ] Only on terraform plan or apply.
- [ ] With any terraform command.
- [ ] Only if you specify the -outputs flag on apply.

Q149. You can set which of the environment variable in Terraform to enable detailed logs?

- [ ] `TF_LOG`
- [ ] `TF_LOG_LEVEL`
- [ ] `TF_TRACE`
- [ ] `TF_VAR_LOG`

Q150. You have encountered an issue with Terraform and want to enable logging to find the root cause of the error. You should set `TF_LOG` to which log level for the MOST verbose logging?

- [ ] `TRACE`
- [ ] `DEBUG`
- [ ] `WARN`
- [ ] `INFO`

Q151. You want to know from which paths Terraform is loading providers referenced in your Terraform configuration (*.tf files). You need to enable detailed logging to find this out. Which of the following would achieve this?

- [ ] Set the environment variable `TF_LOG=TRACE`
- [ ] Set the environment variable `TF_INPUT=1`
- [ ] Set the environment variable `TF_VAR_LOG=TRACE`
- [ ] Set the environment variable `TF_LOG_PATH=./terraform.log`

Q152. You are required to set up Terraform logs. Your boss asks you to make sure they always end up in one location such that they can be collected, and that they be set to the informational level. How would you accomplish this? (Choose 2 answers)

- [ ] Set and export the environment variable `TF_LOG=INFO`
- [ ] Set and export the environment variable `TF_LOG_PATH` to the requested path location.
- [ ] Only invoke the terraform apply command in the location your boss wants the logs, because terraform automatically saves a *.log* file in the working directory.
- [ ] Set and export the environment variable `TF_PATH_LOG` to the requested path location.

Q153. Environment variable `<?>_region=us-west-1` can be used to set the value of `region` input variable. What is `<?>` here?

- [ ] `TF_VAR`
- [ ] `TF_ENV`
- [ ] `TF_VAL`
- [ ] `TF_VARIABLE`

Q154. You have a Terraform variable that is declared as follows:

```
variable "num" {
 default = 3
}
```

You have also defined the following environment variables in your BASH shell:-

```
export TF_VAR_num=10
```

You also have a *terraform.tfvars* file with the following contents:-

```
num = 7
```

When you run the following apply command, what is the value assigned to the num variable?

```
terraform apply -var num=4
```

- [ ] 4
- [ ] 7
- [ ] 3
- [ ] 10

Q155. Which of the following is a valid variable name in Terraform?

- [ ] 1234
- [ ] 1_aws_vpc
- [ ] invalid
- [ ] count

Q156. What are Data Sources in terraform?

- [ ] Data to be fetched or computed for use elsewhere in terraform configuration.
- [ ] Similar to resources, they specify data to be created in the corresponding provider.
- [ ] A binary set of operators that tell resources how to behave with certain meta-arguments.
- [ ] Data sources are a way for terraform to keep track of all resources created in the provider’s infrastructure.

Area 11: Terraform Module

Q157. Which of the following best describes a Terraform module?

- [ ] A collection of resources that make up a specific piece of infrastructure
- [ ] A plugin that allows Terraform to interact with a specific cloud provider or service
- [ ] A set of variables used to configure Terraform resources
- [ ] A tool for managing Docker containers

Q158. In Terraform, what is a module?

- [ ] A group of related resources
- [ ] A singular, non-abstractive, resource.
- [ ] Essentially a comment, it doesn't do anything except to describe a set of resources.
- [ ] Similar to programming functions, modules are used to write code in Golang for direct interaction with Terraform.

Q159. In Terraform, What are modules used for?

- [ ] Organize configuration
- [ ] Encapsulate configuration
- [ ] Re-use configuration
- [ ] All of the above

Q160. Which of the following is true for installing provider when `terraform init` command runs?

- [ ] If any acceptable versions are installed, Terraform uses the newest installed version that meets the constraint (even if the Terraform Registry has a newer acceptable version)
- [ ] If no acceptable versions are installed and the plugin is one of the providers distributed by HashiCorp, Terraform downloads the newest acceptable version from the Terraform Registry and saves it in a subdirectory under `.terraform/providers/`
- [ ] If no acceptable versions are installed and the plugin is not distributed in the Terraform Registry, initialization fails and the user must manually install an appropriate version.
- [ ] All of the above

Q161. When you initialize Terraform from CLI `terraform init`, it starts downloading all the necessary modules referenced in the code and cache it to which directory?

- [ ] in the `/temp` directory on the machine executing Terraform
- [ ] in a `/modules` directory in the current working directory
- [ ] in the `/downloads` directory for the user running the terraform init
- [ ] in the `.terraform/modules` subdirectory in the current working directory

Q162. When you initialize a Terraform workspace, it installs all the provides and modules referred in the configuration in which directory?

- [ ] directly in the current working directory
- [ ] in the `downloads` directory in the current working directory
- [ ] in the directory where terraform is installed
- [ ] in the `.terraform` subdirectory in the current working directory

Q163. Which one of the following is the required argument for calling a child module?

- [ ] version
- [ ] source
- [ ] providers
- [ ] depends_on

Q164. What are three meta-arguments, along with `source` and `version`, that a module can use? (Choose 3 answers)

- [ ] `for_each`
- [ ] `count`
- [ ] `max`
- [ ] `depends_on`

Q165. A module that has been called by another module is often referred to as a child module. Where is the child module stored in below module block?

```
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"

  name = "my-vpc"
  cidr = "10.0.0.0/16"

  azs             = ["eu-west-1a", "eu-west-1b", "eu-west-1c"]
  private_subnets = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  public_subnets  = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]
}
```

- [ ] in a local directory named `.terraform/terraform-aws-modules/vpc/aws`
- [ ] in a remote code repository
- [ ] in terraform cloud private module registry
- [ ] in terraform public module registry

Q166. You have written the below code snippet in `main.tf` file. Which of the following statements are true? (select two)

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

- [ ] `app-cluster` is a child module
- [ ] `app-cluster` is a calling module
- [ ] `main.tf` is a child module
- [ ] `main.tf` is the calling module

Q167. Which one of the following is a valid source type to download the source code of a module?

- [ ] Local Paths
- [ ] Terraform Registry
- [ ] Github
- [ ] Bitbucket
- [ ] HTTP URLs
- [ ] S3 buckets
- [ ] All of the above

Q168. Which one of the following file extension recognized by terraform while fetching archived module over HTTP?

- [ ] zip
- [ ] tar.bz2 and tbz2
- [ ] tar.gz and tgz
- [ ] tar.xz and txz
- [ ] All of the above

Q169. How do you download a module configured in your Terraform code?

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

- [ ] `terraform get module consul`
- [ ] `terraform install modules consul`
- [ ] `terraform init`
- [ ] `terraform module init`

Q170. What feature of Terraform Cloud allows you to share Terraform providers and modules only with the members of the organization?

- [ ] Remote Runs
- [ ] Remote Terraform Execution
- [ ] Private Registry
- [ ] Private VCS Connection

Q171. How do you correctly reference a private registry module source?

- [ ] `<HOSTNAME>/<NAMESPACE>/<NAME>/<PROVIDER>`
- [ ] `<NAMESPACE>/<NAME>/<PROVIDER>`
- [ ] `<HOSTNAME>/<NAMESPACE>/<PROVIDER>`
- [ ] `<NAMESPACE>/<NAME>/<PROVIDER>/<HOSTNAME>`

Q172. How do you reference module source from public terraform registry?

- [ ] `<NAMESPACE>/<NAME>/<PROVIDER>`
- [ ] `<NAMESPACE>/<PROVIDER>/<NAME>`
- [ ] `<NAMESPACE>/<PROVIDER>`
- [ ] `<HOSTNAME>/<NAMESPACE>/<NAME>/<PROVIDER>`

Q173. When specifying a module, what is the best practice for the implementation of the meta-argument version?

- [ ] The best practice is to explicitly set the version argument as a version constraint string from the Terraform registry.
- [ ] The best practice is to use no version and accept the latest version.
- [ ] The best practice is to download the module, place it in your working directory, then source that module, and specify the version that was downloaded.
- [ ] The best practice is to always ensure you append beta to the end of the version. This allows you and your team to always be working on the latest and greatest features for that module.

Q174. How do you access module attributes?

- [ ] Through the child module, by declaring an output value to selectively export certain values to be accessed by the calling module.
- [ ] Through the parent module, by declaring an output value to selectively export certain values to be accessed by the calling module.

- [ ] By specifying the outputs block.
- [ ] When apply is ran, you must pass in -resource-output={ATTRIBUTE.NAME}.

Q175. Who can publish and share modules on the Terraform Registry?

- [ ] Anyone
- [ ] Only specific providers
- [ ] Those who have passed the Hashicorp Terraform Associate exam
- [ ] Only those who have contributed to Open Source Terraform

Q176. What are the requirements to publish a module to the Terraform Public Module Registry? (select three)

- [ ] Release tag is optional for initial module publishing and mandatory for subsequent releases
- [ ] The module must be on GitHub and must be a public repo
- [ ] The module must adhere to the standard module structure
- [ ] Module repositories must use this three-part name format, `terraform-<PROVIDER>-<NAME>` e.g. `terraform-google-vault`

Q177. What are some of the requirements for publishing Private Modules to the Terraform Cloud Private Registry? (select three)

- [ ] The module must be PCI/HIPPA compliant
- [ ] The module must be on your configured VCS providers, and Terraform Cloud's VCS user account must have admin access to the repository
- [ ] The module must adhere to the standard module structure
- [ ] Module repositories must use this three-part name format, `terraform-<PROVIDER>-<NAME>` e.g. `terraform-google-vault`

Q178. In a parent module, outputs of child modules are available in expressions as?

- [ ] `module.<MODULE NAME>.<OUTPUT NAME>`
- [ ] `<MODULE NAME>.<OUTPUT NAME>`
- [ ] `module.<OUTPUT NAME>`
- [ ] `output.<MODULE NAME>.<OUTPUT NAME>`

Q179. You created a module named `web_server` that outputs the `instance_ip_addr`, which is the IP Address of the web server instance created by the module. How would you reference the IP Address when using it for an input of another module?

- [ ] `ip_addr = module.outputs.web_server.instance_ip_addr`
- [ ] `ip_addr = module.web_server.instance_ip_addr`
- [ ] `ip_addr = web_server.instance_ip_addr`
- [ ] `ip_addr = web_server.outputs.instance_ip_addr`

Area 12: Terraform Security

Q180. You want to ensure that your S3 buckets provisioned by Terraform are securely encrypted. What is the best way to achieve this?

- [ ] Create a Git hook that checks if the encryption parameter is enabled.
- [ ] Use AWS KMS to store a security key.
- [ ] Create a lambda function triggered on a “create bucket CloudTrail” event.
- [ ] Create a security policy using Sentinel policies.

Q181. Which Terraform feature can be used to apply policy as code to enforce compliance and governance policies before any infrastructure changes?

- [ ] Resources
- [ ] Functions
- [ ] Sentinel
- [ ] Workspaces

Q182. HashiCorp Sentinel is a(n) ____ framework.

- [ ] platform as a service
- [ ] function as a service
- [ ] infrastructure as code
- [ ] policy as code

Q183. Terraform Cloud provides imports to define Sentinel Policy Rules. Which of the following is not a valid import?

- [ ] `tfplan`
- [ ] `tfconfig`
- [ ] `tfstate`
- [ ] `tfapply`

Q184. You want to create a sentinel policy to ensure that naming convention is being followed in Terraform Configuration as per organization-wide standard. Which sentinel import can be used to access Terraform Configuration?

- [ ] `tfplan`
- [ ] `tfconfig`
- [ ] `tfstate`
- [ ] `tflan`

Q185. Which is NOT a valid sentinel policy enforcement level?

- [ ] advisory
- [ ] soft mandatory
- [ ] warning
- [ ] hard mandatory

Q186. You have enabled Sentinel Policy in Terraform Cloud. When Terraform Cloud evaluates policies?

- [ ] On every Terraform Run
- [ ] After successful `terraform plan`

- [ ] During `terraform plan`
- [ ] Before `terraform apply`

Q187. 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?

- [ ] Delete the state file every time you run Terraform
- [ ] Store the state in an encrypted backend
- [ ] Edit your state file to scrub out the sensitive data
- [ ] Always store your secrets in a secrets.tfvars file.

Q188. If you manage any sensitive data with Terraform (like database passwords, user passwords, or private keys), treat the state itself as sensitive data. What is recommended to protect the state file? (select two)

- [ ] use the S3 bucket using the `encrypt` option to ensure state is encrypted
- [ ] enable native encryption in Terraform as configured in the `terraform` block
- [ ] use Terraform Cloud which always encrypts state at rest
- [ ] replicate the state file to an encrypted storage device

Area 13: Terraform Workspace

Q189. Each Terraform CLI Workspace uses its own state file to manage the infrastructure associated with that particular workspace.

- [ ] true
- [ ] false

Q190. What Terraform feature is most applicable for managing small differences between different environments, for example development and production?

- [ ] Workspaces
- [ ] States
- [ ] Repositories
- [ ] Versions

Q191. Where are Terraform Workspace local state files stored?

- [ ] a directoy called `terraform.tfstate.d`
- [ ] a file called `terraform.tfstate`
- [ ] a temp directory called `.tfstate*`
- [ ] a directory called `terraform.workspaces.tfstate`

Q192. 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?

- [ ] `terraform import`
- [ ] `terraform workspace`
- [ ] `terraform state`
- [ ] `terraform init`

Q193. One of your colleagues is new to Terraform and wants to add a new workspace named new-hire. What command he should execute from the following?

- [ ] `terraform workspace –new –new-hire`
- [ ] `terraform workspace new new-hire`
- [ ] `terraform workspace init new-hire`
- [ ] `terraform workspace new-hire`

Q194. As a prestigious Sr. Cloud Engineer, your colleague comes up to you and asks for a new Development workspace. What's the fastest way to accomplish this?

- [ ] Through CLI `terraform workspace new dev`
- [ ] Head to the Terraform Enterprise console and create a new workspace there.
- [ ] Specify in the configuration block the new workspace to be created.
- [ ] Have them submit a Jira ticket and tell them you'll get around to it in the next Sprint.

Q195. You have created `prod` and `test` workspaces from the command line. Which of the following commands you can run to switch to the `test` workspace?

- [ ] `terraform workspace switch test`
- [ ] `terraform workspace select test`
- [ ] `terraform workspace test`
- [ ] `terraform workspace -switch test`

Q196. Which is NOT true about Terraform Cloud and Terraform CLI Workspaces?

- [ ] Each Terraform Cloud workspace has its own Terraform configuration, variables, state file, backup of previous state files, run history, credentials & secrets, and settings.
- [ ] Each Terraform CLI workspace is a persistent working directory, which may contains a configuration, state data, and variables.
- [ ] You cannot manage resources in Terraform Cloud without creating at least one workspace.
- [ ] You must create a local working directory using Terraform CLI to manage resources in local.

Area 14: Terraform Version Constraint

Q197. Which version constraint should use to set both a lower and upper bound on versions for each provider. Also known as pessimistic constraint operator?

- [ ] `>=`
- [ ] `~>`
- [ ] `!=`
- [ ] `<>`

Q198. What does the specified contraint version = “~> 1.0.4” means in required_providers block?

```
terraform {
  required_providers {
    mycloud = {
      source  = "mycorp/mycloud"
      version = "~> 1.0.4"
    }
  }
}
```

- [ ] >= 1.0.4 and <= 1.1.0
- [ ] >= 1.0.4 and < 1.1.0

- [ ] > 1.0.4 and < 2.0.0
- [ ] >= 1.0.5 and < 1.1.0

Q199. What does this symbol version = “~> 1.0” mean when defining versions?

- [ ] > 1.0 and < 2.0
- [ ] >= 1.0 and < 2.0
- [ ] >= 1.0 and <= 2.0
- [ ] > 1.0.0 and < 2.0.0

Q200. What is the provider version of Google Cloud being used in Terraform? Select all valid options.

```
provider "google" {
  version = "~> 1.9.0"
}
```

- [ ] 1.9.1
- [ ] 1.10.0
- [ ] 1.8.0
- [ ] 1.9.9

Q201. How do you force users to use a particular version of required providers in your terraform code?

- [ ] `terraform { required_providers { aws = { source = “hashicorp/aws” version =”3.74.1” } } }`
- [ ] `terraform { aws = { source = “hashicorp/aws” version = “~>3.74.1” } }`

- [ ] `aws = { source = “hashicorp/aws” version = “3.74.1” }`
- [ ] `terraform { required_providers { aws = { source = “hashicorp/aws” version =”~>3.74.1” } } }`

Q202. What is more accurate description of the below version constraint?

```
terraform {
  required_version = ">= 1.4.7"
}
```

- [ ] All the terraform providers are should have a minimum 1.4.7 version
- [ ] Terraform version older than 1.4.7 are not supported by Hashicorp
- [ ] Terraform version older than 1.4.7 will produce an error running this configuration
- [ ] The minimum version of the application using Terraform should be 1.4.7

Area 15: Terraform Types and Functions

Q203. You are adding input variables in the Terraform module for customization. Which of the following is NOT a valid primitive variable type in Terraform?

- [ ] string
- [ ] number
- [ ] float
- [ ] bool

Q204. What are two complex types in terraform? (Choose 2 answers)

- [ ] A Collection Type
- [ ] A Structural Type
- [ ] A String Type
- [ ] A float64 type

Q205. What are complex types in terraform?

- [ ] A type that groups multiple values into a single value.
- [ ] A variation of a string type.
- [ ] A variance of a data source.
- [ ] A type that derives its value from RegEx logic.

Q206. If an input variable has no type value set, what type does it accept?

- [ ] Any type.
- [ ] None, it has to have a type value set.
- [ ] Terraform infers the type when it is referenced.
- [ ] Type string. As strings can be interpreted in a number of ways by Terraform.

Q207. Which of the following is not a valid Terraform Collection type?

- [ ] list
- [ ] map
- [ ] tree
- [ ] set

Q208. Which of the followings are valid Terraform Structural types? (Choose 2 answers)

- [ ] optional
- [ ] object
- [ ] pair
- [ ] tuple

Q209. What are the similar kind of complex types in Terraform? (select three)

- [ ] list
- [ ] map
- [ ] set
- [ ] tuple

Q210. You want to define a single input variable to store information about servers mainly server-name of type string and memory-size of type number. Which variable type should you choose?

- [ ] list
- [ ] map
- [ ] object
- [ ] set

Q211. Which variable type can be used in Terraform for key/value pairs?

- [ ] list
- [ ] tuple
- [ ] map
- [ ] set

Q212. Which of the following is not a valid string function in Terraform?

- [ ] split()
- [ ] join()
- [ ] slice()
- [ ] chomp()

Q213. What are some built-in functions that terraform provides? (Choose 3 answers)

- [ ] max()
- [ ] regex()
- [ ] alltrue()
- [ ] delete()

TERRAFORM CERTIFICATION

See Also

- Hashicorp Terraform Associate (003) Exam Guide
- Leading SAFe Agilist 6.0 (Scaled Agile) Exam Notes
- Apache Kafka CCDAK Exam Notes

About Ashish Lahoti

Ashish Lahoti is a Software Engineer with 12+ years of experience in designing and developing distributed and scalable enterprise applications using modern practices. He is a technology enthusiast and has a passion for coding & blogging.

« PREVIOUS
Hashicorp Terraform Associate (003) Exam Guide

Thanks Callai Chan for pointing out the Q114 incorrect answer. I have updated the blog post with correct answer.

thingamajig540 commented on Dec 7, 2023

For question 188: the answer is d;
question 189 :the answer is a

ashishlahoti commented on Dec 9, 2023 Owner

@thing
The answers to the questions 188 (b) and 189 (b) are correct. I have verified from various sources.

One of the hashicorp member also mentioned how pessimistic version works in below link:-
hashicorp/terraform#25762

The pessimistic operator is intended to apply to the last 2 segments of the provided version. The documented example for this is

~> 1.2.0: any non-beta version >= 1.2.0 and < 1.3.0, e.g. 1.2.X
~> 1.2: any non-beta version >= 1.2.0 and < 2.0.0, e.g. 1.X.Y

Andrewcena95 commented on Jan 12, 2024

Hello, could you please verify the answers to these practice questions: https://www.study4exam.com/hashicorp/free-terraform-associate-003-questions
A friend of mine, who recently passed the exam, recommended this source to me. Thanks.

Write Preview

Sign in to comment

Styling with Markdown is supported

Sign in with GitHub