# Hashicorp Terraform — Certification MCQ + Scenarios + Mock Test!!

**LinkedIn:** Amit Singh

**Medium:** Amit Singh – Medium

**Contains recently asked terraform questions in Hashicorp terraform exam!! Covered MCQs, scenario based question**

## Section 1: Terraform Basics (1–10)

1. **What is the primary purpose of Terraform?**
   A. Manage containers
   B. Monitor infrastructure
   C. Provision infrastructure as code
   D. Manage databases

2. **What does terraform init do?**
   A. Validates code
   B. Starts provisioning
   C. Initializes backend and downloads providers
   D. Destroys infrastructure

3. **Which command shows a preview of changes before applying them?**
   A. terraform validate
   B. terraform show
   C. terraform plan
   D. terraform graph

4. **Which command is used to apply infrastructure changes?**
   A. terraform up
   B. terraform exec
   C. terraform plan
   D. terraform apply

5. **What does terraform destroy do?**
   A. Deletes configuration
   B. Deletes infrastructure managed by Terraform
   C. Removes providers

D. Reverts to a previous state

6. **What is Terraform state used for?**
   A. Store logs
   B. Track infrastructure resources
   C. Backup configuration
   D. Encrypt variables

7. **Why is terraform state critical?**
   A. It helps in reducing costs
   B. It contains the current status of infrastructure
   C. It validates HCL
   D. It provides encryption

8. **What is the .terraform directory for?**
   A. Store version history
   B. Store logs
   C. Cache providers and modules
   D. Save credentials

9. **What does .terraform.lock.hcl do?**
   A. Locks remote state
   B. Locks workspace
   C. Locks provider versions
   D. Locks variables

10. **Which file is most sensitive and should be secured?**
    A. terraform.tf
    B. terraform.tfvars
    C. terraform.tfstate
    D. main.tf.json

**ANS- 1:C 2:C 3:C 4:D 5:B 6:B 7:B 8:C 9:C 10:C**

## ◆ Section 2: Terraform Configuration Language

1. **How do you define a variable?**
   A. let variable = “value”
   B. input variable “name”
   C. variable "name" { default = "value" }
   D. var name = value

2. **How can you set default values for a variable?**
   A. With locals
   B. Using variable block
   C. Using .tfstate
   D. Through providers

3. **What are locals used for?**
   A. To connect providers
   B. Define temporary and reusable values
   C. Export state
   D. Encrypt secrets

4. **What is an output value?**
   A. A result of a function
   B. A printed log
   C. A value shown after terraform apply
   D. A temporary cache

5. **What is a resource in Terraform?**
   A. A module
   B. A file
   C. A real infrastructure component
   D. A provider

6. **How do you force dependency order between resources?**
   A. Using locals
   B. Using count
   C. Using depends_on
   D. Using output

7. **What is a data source used for?**
   A. To create resources
   B. To fetch existing resources
   C. To destroy resources
   D. To monitor resources

8. **How do you reference a data source?**
   A. data.name
   B. name.data.resource
   C. data.<TYPE>.<NAME>.<ATTRIBUTE>
   D. module.data

9. **What does count allow you to do?**
   A. Count logs
   B. Create multiple copies of a resource
   C. Validate variables

D. Loop over outputs

10. **What are conditional expressions used for?**
    A. To define local variables
    B. To write if-else logic
    C. To import data
    D. To encrypt files

**1:C 2:B 3:B 4:C 5:C 6:C 7:B 8:C 9:B 10:B**

## Section 3: Providers and Resources

1. **How do you define a provider in Terraform?**
    A. Using use block
    B. Using import statement
    C. Using provider block
    D. Using provision block

2. **How does Terraform handle multiple providers?**
    A. One at a time
    B. Through aliases
    C. Not allowed
    D. Using provider maps only

3. **What is a provider version constraint?**
    A. Terraform version restriction
    B. Module version restriction
    C. Limits allowed provider versions
    D. Backend locking

4. **What is the correct way to use third-party providers?**
    A. Only from AWS
    B. Through Terraform Registry
    C. Clone manually
    D. Using Docker

5. **How can you securely pass AWS credentials to Terraform?**
    A. Hardcode in main.tf
    B. Use .env
    C. Use environment variables
    D. Pass via output block

6. **Which provider is used to provision AWS resources?**
   A. aws.tf
   B. provider.aws
   C. terraform.aws
   D. module.aws

7. **Which file is used to lock provider versions?**
   A. versions.tf
   B. terraform.tfvars
   C. .terraform.lock.hcl
   D. provider.tf

8. **What is the function of the required_providers block?**
   A. Lists data sources
   B. Enforces usage of versions
   C. Prevents provider updates
   D. Applies all modules

9. **What happens if a provider is not found?**
   A. Plan still works
   B. Terraform uses latest automatically
   C. terraform init fails
   D. It skips the provider

10. **How can you override default provider configuration?**
    A. With local-exec
    B. Using count
    C. Using provider alias
    D. You can't

**1:C 2:B 3:C 4:B 5:C 6:B 7:C 8:B 9:C 10:C**

## Section 4: State Management

1. **What file stores Terraform's state?**
   A. terraform.tfvars
   B. terraform.tf
   C. terraform.tfstate
   D. state.json

2. **What is a remote backend?**
   A. Remote storage for .tf files
   B. Remote compute
   C. Remote location for state file

D. Remote provider

3. **Difference between local and remote state?**
   A. Format
   B. Encryption
   C. Storage location
   D. It depends

4. **What is Terraform Cloud?**
   A. Cloud-based provider
   B. SaaS for remote operations
   C. Terraform UI tool
   D. GCP version of Terraform

5. **What does state locking do?**
   A. Encrypts state
   B. Prevents simultaneous changes
   C. Archives old state
   D. Locks .tf files

6. **How do you inspect state manually?**
   A. terraform inspect
   B. terraform state show
   C. terraform state browse
   D. terraform state check

7. **What is terraform state list used for?**
   A. Lists variables
   B. Lists outputs
   C. Lists resources in state
   D. Lists modules

8. **What does terraform state rm do?**
   A. Deletes infrastructure
   B. Removes resource from state only
   C. Uninstalls provider
   D. Reverts last apply

9. **When do you use terraform state mv?**
   A. To move files
   B. To rename provider
   C. To move state from one resource to another
   D. To archive state

10. **Where can you store remote state?**
    A. S3

B. Terraform Cloud
C. Consul
D. All of the above

**1:C 2:C 3:C 4:B 5:B 6:B 7:C 8:B 9:C 10:D**

## Section 5: Modules

1. **What is a module?**
   A. A command
   B. A reusable Terraform configuration
   C. A Terraform plugin
   D. A wrapper for HCL

2. **How do you create a module?**
   A. Use module.tf file
   B. Create a new directory with .tf files
   C. Use terraform create-module
   D. Install from CLI

3. **How do you call a module?**
   A. Using use block
   B. With provider
   C. Using module block
   D. Using import

4. **What is the root module?**
   A. The first provider
   B. The main state
   C. The directory where Terraform is run
   D. The oldest module

5. **How do you pass variables to a module?**
   A. Directly as environment vars
   B. Through outputs
   C. Using variables.tf
   D. Using module "name" { ... }

6. **Best way to organize modules?**
   A. One large module
   B. All in root
   C. Per service in separate directories
   D. In main.tf only

7. **How do you source a module from GitHub?**
   A. source = “github.com/owner/repo”
   B. import = “repo”
   C. git clone first
   D. Only from Registry

8. **What is terraform get used for?**
   A. Download remote state
   B. Fetch modules
   C. Apply plan
   D. Import resources

9. **How does Terraform handle module versioning?**
   A. It doesn't
   B. Through Git tags
   C. Through provider constraint
   D. Automatically updates

10. **What is registry.terraform.io?**
    A. Terraform's documentation
    B. Terraform CLI repo
    C. Official module and provider registry
    D. Plugin repo

**1:B 2:B 3:C 4:C 5:D 6:C 7:A 8:B 9:B 10:C**

## Section 6: Terraform Cloud & Workspaces

1. What are Terraform workspaces used for?
   A. Multi-region deployment
   B. Organizing variables
   C. Managing multiple state files
   D. Automating testing

2. How do you switch between workspaces?
   A. terraform use
   B. terraform workspace select
   C. terraform switch
   D. terraform change

3. What is the default workspace called?
   A. prod
   B. global
   C. default
   D. dev

4. What is Terraform Cloud?
   A. A cloud provider
   B. A Terraform GUI
   C. Remote execution and collaboration platform
   D. An AWS service

5. How do you connect CLI to Terraform Cloud?
   A. Add backend config in main.tf
   B. Create a workspace in UI
   C. Use Terraform login and configure backend
   D. Push code to GitHub

6. What is Sentinel?
   A. Logging tool
   B. Policy-as-code framework
   C. Provider manager
   D. Terraform backup

7. What is remote execution in Terraform Cloud?
   A. Running commands on target VM
   B. Running plan/apply on Terraform Cloud servers
   C. SSH-based provisioning
   D. Using external data

8. What's the main benefit of Terraform Cloud?
   A. Cheaper plans
   B. Supports only Azure

C. Centralized state management and RBAC
D. Plugin marketplace

9. What command shows current workspace?
   A. terraform get
   B. terraform workspace
   C. terraform workspace show
   D. terraform env

10. What is a common use case for multiple workspaces?
    A. Use multiple providers
    B. Separate environments like dev/stage/prod
    C. Run tests
    D. Change cloud accounts

**1:C 2:B 3:C 4:C 5:C 6:B 7:B 8:C 9:C 10:B**

## Section 7: Functions & Expressions

1. **What does join() do in Terraform?**
   A. Joins map
   B. Joins list with delimiter
   C. Combines resources
   D. Imports modules

2. **What does split() do?**
   A. Split numbers
   B. Divide a resource
   C. Break string into list
   D. Join strings

3. **What is the lookup() function used for?**
   A. Reference variables
   B. Search data source
   C. Return value from a map
   D. Combine values

4. **What does merge() function do?**
   A. Merge state files
   B. Merge lists
   C. Merge maps
   D. Merge modules

5. **What does file() function do?**
   A. Create files
   B. Delete files
   C. Read file contents
   D. Import configuration

6. **What does length() return?**
   A. Character count
   B. Number of list/map elements
   C. Width of string
   D. Output size

7. **What does contains() do?**
   A. Checks if a key exists in state
   B. Checks membership in list/map
   C. Searches strings
   D. Returns last element

8. **What is element() used for?**
   A. Fetch an item by index
   B. Check resource type
   C. Loop over list
   D. Convert map to list

9. **What is a dynamic block in Terraform?**
   A. Runtime-defined resource
   B. Loopable resource configuration
   C. Static resource
   D. Output block

10. **What can you loop over in HCL?**
    A. Only variables
    B. Providers
    C. Lists and maps
    D. Workspaces

**1:B 2:C 3:C 4:C 5:C 6:B 7:B 8:A 9:B 10:C**

## Section 8: Lifecycle & Meta-Arguments

1. What is the lifecycle block used for?
   A. Manage module order
   B. Set resource lifecycle behavior
   C. Configure providers
   D. Control execution environment

2. What does create_before_destroy do?
   A. Runs destroy first
   B. Creates resource before deleting old one
   C. Prevents apply
   D. Enables auto-rollback

3. What does prevent_destroy do?
   A. Blocks deletion
   B. Forces delete
   C. Ignores change
   D. Skips validation

4. What does ignore_changes do?
   A. Disables outputs
   B. Prevents plan failures
   C. Ignores external changes to specified attributes
   D. Prevents state update

5. Why use depends_on?
   A. Group modules
   B. Delay execution
   C. Force resource order
   D. Export values

6. Which argument repeats a resource n times?
   A. count
   B. for_each

C. loop
D. replicate

7. Which meta-arg loops over a map or set of strings?
   A. count
   B. map
   C. for_each
   D. each_for

8. What is for expression used for in Terraform?
   A. Loop output
   B. Variable declaration
   C. Conditional statement
   D. Resource block

9. Which lifecycle setting helps prevent accidental deletion?
   A. ignore_changes
   B. depends_on
   C. prevent_destroy
   D. always_run

10. What does each.key return inside for_each?
    A. The index
    B. The name of the loop variable
    C. The key of the map item
    D. All items

**ANS → 1:B 2:B 3:A 4:C 5:C 6:A 7:C 8:A 9:C 10:C**

## ◆ Section 9: Debugging & Troubleshooting

1. How do you enable debug logs in Terraform?
   A. terraform --debug
   B. terraform log
   C. TF_LOG=DEBUG
   D. debug=true

2. What is TF_LOG?
   A. Variable for output
   B. Provider log
   C. Environment variable for logging
   D. Error handler

3. What does terraform validate do?
   A. Checks credentials
   B. Applies plan
   C. Syntax check of .tf files
   D. Compares plan to state

4. What is infrastructure drift?
   A. Incorrect module
   B. State is out of sync with real infra
   C. Provider version mismatch
   D. Terraform crash

5. **What causes “resource already exists” error?**
   A. Remote backend error
   B. Resource is not in state but exists in infra
   C. Syntax error
   D. Data source mismatch

6. **What happens if state file is deleted?**
   A. Plan fails
   B. All infra deleted
   C. Terraform loses track of managed resources
   D. Apply still works

7. **How to detect configuration drift?**
   A. terraform validate
   B. terraform refresh
   C. terraform graph
   D. terraform lock

8. **What does terraform console help with?**
   A. Logs output
   B. Runs shell
   C. Evaluate expressions and inspect values
   D. Debug API

9. **What is the effect of terraform plan -destroy?**
   A. Deletes state
   B. Shows what will be destroyed
   C. Deletes modules
   D. Skips plan

10. **What is a common fix for missing variables?**
    A. terraform clear
    B. terraform restart
    C. Pass -var or tfvars
    D. Re-import state

**ANS → 1:C 2:C 3:C 4:B 5:B 6:C 7:B 8:C 9:B 10:C**

## 🔹 Section 10–13: Real-World + Errors + Tips

1. **How to manage multiple environments in Terraform?**
   A. Multiple providers
   B. Workspaces or directories
   C. Change names
   D. Change versions

2. **What is the DRY principle in Terraform?**
   A. Debug Repeatably Yourself
   B. Don't Repeat Yourself - use modules
   C. Duplicate Resource YAML
   D. Developer Runtime YAML

3. **How do modules help with DRY?**
   A. Increase config
   B. Reduce security
   C. Create reusable configs
   D. Add complexity

4. **Can Terraform manage DNS, SaaS, etc.?**
   A. No
   B. Yes, via providers

C. Only AWS
D. With shell scripts

5. **What is a best practice for CI/CD with Terraform?**
   A. Use terraform destroy
   B. Use backend as S3
   C. Run init/plan/apply in stages
   D. Manual apply

6. **Error: “provider not found” — how to fix?**
   A. Remove backend
   B. Run terraform refresh
   C. Run terraform init
   D. Delete .terraform

7. **Error: "Cannot destroy resource, it is in use" — fix?**
   A. Re-run apply
   B. Force delete
   C. Check dependencies
   D. terraform clean

8. **Error: "resource already exists" — fix?**
   A. Remove manually and re-import
   B. Restart Terraform
   C. Delete state
   D. Plan again

9. **Error: terraform init fails — reason?**
   A. Workspace not set
   B. Missing provider config
   C. No .tfvars
   D. Output missing

10. **Best way to practice for exam?**
    A. Watch YouTube
    B. Do real cloud infra
    C. Use Terraform Learn and write hands-on modules
    D. Read code only

**ANS-> 1:B 2:B 3:C 4:B 5:C 6:C 7:C 8:A 9:B 10:C**

**50+ scenario-based questions** for the **latest HashiCorp Certified: Terraform Associate exam (including Terraform 1.x features)**. These are designed to reflect real-world use cases and frequently asked exam questions.

## Section 1: Terraform Core Concepts (1–10)

**1. Scenario:**
You're setting up Terraform for the first time in a new project. What command must you run first to prepare the working directory?
👉 **Answer:** terraform init — initializes backend and downloads providers.

**2. Scenario:**
You ran terraform plan and noticed no changes are proposed, but you've updated your configuration. What might be wrong?
👉 **Answer:** The change may not be referencing a managed resource or the value is unchanged from Terraform's perspective.

**3. Scenario:**
You want to validate the syntax of a .tf file without applying anything.
👉 **Answer:** Run terraform validate.

**4. Scenario:**
After modifying a variable's default value, no changes are seen during terraform plan. Why?
👉 **Answer:** If the variable was overridden by tfvars or CLI input, the default won't apply.

**5. Scenario:**
Your terraform apply fails due to a lock on the state file. What is happening?
👉 **Answer:** Another operation is holding a state lock; wait or manually unlock using terraform force-unlock.

**6. Scenario:**
You accidentally deleted your .terraform folder. What should you do?
👉 **Answer:** Re-run terraform init to reinitialize the project.

**7. Scenario:**
You want to see what infrastructure will be destroyed without making any changes.
👉 **Answer:** Run terraform plan -destroy.

**8. Scenario:**
How can you view the current values in your state file without opening the JSON file?
👉 **Answer:** Use terraform show or terraform state show <resource>.

**9. Scenario:**
You want to make sure that Terraform always replaces a resource instead of updating it.
👉 **Answer:** Use lifecycle { create_before_destroy = true }.

**10. Scenario:**
You are provisioning infrastructure on AWS and want Terraform to use credentials stored in environment variables.
👉 **Answer:** Use AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY.

## Section 2: State & Backends (11–20)

**11. Scenario:**
Two team members apply Terraform changes simultaneously. What can prevent state corruption?
👉 **Answer:** Use a remote backend with state locking (e.g., S3 + DynamoDB).

**12. Scenario:**
You need to move a resource to a new module structure but keep its existing state.
👉 **Answer:** Use terraform state mv.

**13. Scenario:**
You manually created a resource outside Terraform and now want Terraform to manage it.
👉 **Answer:** Use terraform import.

**14. Scenario:**
You have a resource in your configuration but want Terraform to stop tracking it.
👉 **Answer:** Use terraform state rm.

**15. Scenario:**
You're using a remote backend (S3) but see AccessDenied errors.
👉 **Answer:** Check IAM permissions on the S3 bucket and DynamoDB table (for locking).

**16. Scenario:**
You need to collaborate on infrastructure changes. What's required?
👉 **Answer:** Use a remote backend to share state (Terraform Cloud, S3, etc.).

**17. Scenario:**
You want to prevent sensitive values from being stored in plain text in your state file.
👉 **Answer:** Use sensitive = true in variable or output blocks.

**18. Scenario:**
State file is deleted accidentally from local disk. What's the consequence?
👉 **Answer:** Terraform loses track of managed resources.

**19. Scenario:**
Why is state required in Terraform?
👉 **Answer:** It maps configuration to real-world resources and tracks metadata.

**20. Scenario:**
How do you enable versioning and rollback for state?
👉 **Answer:** Use S3 versioning (for S3 backend).

## Section 3: Variables, Locals, and Expressions (21–30)

**21. Scenario:**
You want to define a variable with a default value but allow override via CLI.
👉 **Answer:** Use variable "name" { default = "value" } and pass via -var.

**22. Scenario:**
You want to reuse a computed value in multiple places without repeating logic.
👉 **Answer:** Use locals.

**23. Scenario:**
You need to create N identical resources.
👉 **Answer:** Use count.

**24. Scenario:**
You need to create a set of resources based on keys in a map.
👉 **Answer:** Use for_each.

**25. Scenario:**
You want to use a conditional value inside a resource argument.
👉 **Answer:** Use a ternary expression: condition ? true_val : false_val.

**26. Scenario:**
You want to access a specific element from a list.
👉 **Answer:** Use element(list, index).

**27. Scenario:**
You want to conditionally define a block inside a resource.
👉 **Answer:** Use dynamic blocks.

**28. Scenario:**
You want to concatenate a list of strings with a comma.

👉 **Answer:** Use join(",", list).

**29. Scenario:**
You want to convert a string to a list.
👉 **Answer:** Use split(",", string).

**30. Scenario:**
You want to return a value from a map with a fallback.
👉 **Answer:** Use lookup(map, key, default).

## Section 4: Providers, Modules, and Reusability (31–40)

**31. Scenario:**
You want to reuse a configuration multiple times across environments.
👉 **Answer:** Use modules.

**32. Scenario:**
You want to lock the provider version to 4.x only.
👉 **Answer:** version = "~> 4.0".

**33. Scenario:**
You want to use multiple provider configurations (e.g., AWS in two regions).
👉 **Answer:** Use provider aliases.

**34. Scenario:**
You want to source a module from GitHub.
👉 **Answer:** source = "github.com/org/repo//path".

**35. Scenario:**
You want to pin a specific module version from the Terraform Registry.
👉 **Answer:** source = "terraform-aws-modules/vpc/aws" version = "3.0.0".

**36. Scenario:**
You want to expose a value from a module to the root.
👉 **Answer:** Use output.

**37. Scenario:**
You need to use different variable values for different environments (dev, prod).
👉 **Answer:** Use workspaces or directories with different .tfvars files.

**38. Scenario:**
You want to reference the region from your provider configuration.
👉 **Answer:** Use provider.aws.region.

**39. Scenario:**
You want to define required providers for a configuration.
👉 **Answer:** Use terraform.required_providers block.

**40. Scenario:**
You want to share Terraform modules across teams.
👉 **Answer:** Use private Git repositories or Terraform private registry.

## Section 5: Workspaces, Lifecycle, Debugging (41–50)

**41. Scenario:**
You want to separate state for dev and prod environments.
👉 **Answer:** Use Terraform workspaces.

**42. Scenario:**
You want to prevent a resource from being deleted accidentally.
👉 **Answer:** Use prevent_destroy in lifecycle block.

**43. Scenario:**
You want to ignore changes made outside Terraform.
👉 **Answer:** Use ignore_changes.

**44. Scenario:**
You want to replace a resource only if a specific attribute changes.
👉 **Answer:** Use lifecycle + ignore_changes to avoid triggering recreation unnecessarily.

**45. Scenario:**
You need to enforce creation order manually.
👉 **Answer:** Use depends_on.

**46. Scenario:**
You want to troubleshoot why a resource is being recreated.
👉 **Answer:** Use terraform plan and check diff; may also check lifecycle metadata.

**47. Scenario:**
You want to evaluate Terraform expressions interactively.
👉 **Answer:** Use terraform console.

**48. Scenario:**
You want to simulate destroying a resource without actually doing it.
👉 **Answer:** Run terraform plan -destroy.

**49. Scenario:**
You want to mark a resource for recreation in the next apply.

👉 **Answer:** Use terraform taint.

**50. Scenario:**
You're debugging Terraform and want detailed logs.
👉 **Answer:** Set TF_LOG=DEBUG.

## Bonus Recent Additions (Terraform 1.x) (51–55)

**51. Scenario:**
You want to check a plan before it is applied automatically in automation.
👉 **Answer:** Use terraform apply -auto-approve=false or manual approval.

**52. Scenario:**
You want to update a module version and verify no breaking changes.
👉 **Answer:** Use terraform plan after changing the module version and check the diff.

**53. Scenario:**
You want to make sure only a specific team can run apply in Terraform Cloud.
👉 **Answer:** Use RBAC and Sentinel policies.

**54. Scenario:**
You want to control apply based on policy logic (e.g., tag enforcement).
👉 **Answer:** Use Sentinel in Terraform Cloud.

**55. Scenario:**
You're using the CLI but want to connect it to a remote workspace.
👉 **Answer:** Configure remote backend with the Terraform Cloud workspace and run terraform login.

## Mock Test 1: Terraform Basics & Configuration Language

**1.** You initialized a project with terraform init, but terraform apply fails due to missing credentials. What should you do?
A. Add credentials block inside the .terraform directory
B. Run terraform plan again
C. Configure provider credentials using environment variables
D. Delete .terraform.lock.hcl and re-run apply

**2.** What happens if you run terraform apply on a configuration with count = 0 on a resource?
A. The resource will be created but inactive
B. Terraform will throw a syntax error
C. No resource will be created
D. A null resource will be added to state

**3.** What is the main difference between terraform plan and terraform apply?
A. plan updates the infrastructure; apply checks it
B. plan executes changes; apply previews
C. plan shows changes; apply applies them
D. They both do the same thing

**4.** You defined a variable in variables.tf but didn't assign a value. What happens?
A. Terraform crashes
B. Terraform uses null by default
C. Terraform prompts for the value during apply
D. It uses the first resource block as default

**5.** Where is Terraform's lock file stored, and what is its purpose?
A. .terraform.lock.hcl; locks provider versions
B. terraform.tfstate; locks state updates
C. .lock.json; locks modules
D. .tfvars; locks input variables

## Mock Test 2: Providers, State, Modules, and Workspaces

**6.** You see an error: provider not found. What is the most likely cause?
A. Missing AWS credentials
B. Provider block syntax is invalid
C. You forgot to run terraform init
D. Incorrect use of terraform apply

**7.** You want to store your state file remotely and allow locking. What should you use?
A. Local backend with S3 bucket
B. Remote backend with DynamoDB for locking
C. GitHub as backend
D. No backend, use local state only

**8.** Which command shows all resources in the state file?
A. terraform show
B. terraform list
C. terraform state list
D. terraform plan

**9.** You deleted a resource manually from AWS but it still exists in your state. What can you do?
A. Use terraform state rm
B. Re-run terraform init
C. Use terraform plan
D. Nothing, Terraform will recreate it automatically

**10.** What is the correct syntax to call a module?
A. resource "module" "name" {}
B. module "name" { source = "./module_dir" }
C. include "module" { path = "./mod" }
D. import module "mod_name" {}

**11.** What's the difference between count and for_each?
A. count only works with maps
B. for_each only with lists
C. count uses index-based logic, for_each uses keys
D. No difference

## Mock Test 3: Lifecycle, Debugging, Security, Advanced Concepts

**12.** You want to make sure a resource is always replaced before destroying the old one. What should you use?
A. depends_on
B. prevent_destroy
C. create_before_destroy
D. force_replace

**13.** Which lifecycle setting can protect critical resources from accidental deletion?
A. prevent_destroy
B. ignore_changes
C. replace_triggered_by
D. depends_on

**14.** You want to log Terraform's internal behavior for debugging. What should you do?
A. Enable logging in the .terraformrc file
B. Set TF_LOG=DEBUG environment variable
C. Run terraform --debug
D. Install the debug plugin

**15.** You need to import an existing EC2 instance into Terraform state. Which command is correct?

A. terraform add aws_instance.id i-12345
B. terraform import aws_instance.my_instance i-1234567890
C. terraform register ec2 i-123456
D. terraform scan ec2 --import

**16.** What does terraform taint do?
A. Destroys resource immediately
B. Ignores resource during apply
C. Marks a resource for recreation
D. Blocks future updates

**17.** You want to store secrets (like AWS keys) securely. Which is a best practice?
A. Hardcode in main.tf
B. Pass via environment variables
C. Store in state file
D. Print them in output

## Answers Key:

1. C
2. C
3. C
4. C
5. A
6. C
7. B
8. C
9. A
10. B
11. C
12. C
13. A
14. B
15. B
16. C
17. B

## Mock Test 4: Real-World AWS Scenarios (Advanced)

**1.** You are managing an auto-scaling group and want Terraform to **ignore changes** made to the desired_capacity outside of Terraform (via AWS Console). What should you do?
A. Remove desired_capacity from your configuration
B. Use ignore_changes in the lifecycle block

C. Use prevent_destroy = true
D. Use terraform taint

**2.** A team member modifies an S3 bucket manually from the AWS Console. You want to reconcile state without recreating the resource. What should you do?
A. Delete the bucket and apply again
B. Use terraform taint
C. Run terraform refresh
D. Use terraform destroy

**3.** You are provisioning EC2 instances in two different regions with the **same configuration**. What is the best way to structure this in Terraform?
A. Use count on region block
B. Create two separate modules with different providers
C. Hardcode regions inside one file
D. Use Terraform Cloud for each region

**4.** You want to deploy an RDS instance with **multi-AZ support** and retain its snapshots after destroy. Which of the following is essential?
A. multi_az = true and skip_final_snapshot = false
B. Use replica = true
C. Add prevent_destroy lifecycle
D. Set publicly_accessible = true

**5.** You use for_each on a security group rule set. Later, you switch to count for the same resource. What is the risk?
A. Resources get updated in-place
B. All security rules are tainted
C. Terraform fails because of conflicting resource addresses
D. Nothing changes

**6.** You are managing an IAM policy document using Terraform. You want to avoid manual string concatenation for JSON. What's the best solution?
A. Use heredoc syntax
B. Use jsonencode() function
C. Use external data source
D. Store the policy in a file and read it with file()

**7.** You need to access secrets from AWS Secrets Manager in Terraform. What is the best approach?
A. Hardcode them in a .tfvars file
B. Use a data block with AWS secretsmanager
C. Store them in the state file
D. Use terraform output to fetch values

**8.** You want to manage a team's IAM roles using a centralized Terraform module. What is the best way to **parametrize role policies** per team?
A. Use separate state files
B. Pass different maps of policies to the module using for_each
C. Use conditional locals in the root config
D. Use different provider aliases

## 🔷 Mock Test 5: Real-World GCP & Multi-Cloud Scenarios

**9.** You are provisioning GCP compute instances using Terraform. You want to reuse the same module in different projects. What's the best pattern?
A. Hardcode project ID inside module
B. Use locals to reference fixed project ID
C. Pass project_id as input variable
D. Write a different module per project

**10.** You want to deploy a GKE cluster using Terraform, and you must rotate the credentials regularly. Which option helps you avoid drift?
A. Use ignore_changes on credentials
B. Exclude GKE auth from the config
C. Use google_container_cluster.master_auth.0.password with a default
D. Rotate credentials manually and re-import every time

**11.** You have 3 Terraform workspaces: dev, staging, prod. You want to deploy the same resources with different values. How can you achieve this?
A. Use if-else conditions in main.tf
B. Use terraform.workspace inside a locals block
C. Use 3 different modules
D. Create 3 provider blocks with aliases

**12.** You're managing cloud infrastructure for multiple GCP projects. You want to separate billing and access controls. How should you architect your Terraform code?
A. Use workspaces per project
B. Use modules with variables and backends scoped per project
C. Use one shared state
D. Use a loop inside main.tf

**13.** You have a Terraform-managed BigQuery dataset and a developer manually updates access policy. How can you detect and sync this change?
A. Use terraform import
B. Run terraform plan to detect drift
C. Use terraform taint
D. Use terraform apply --force

**14.** You want to manage resources in AWS and GCP from the same Terraform root module. What must you configure?
A. Use separate terraform init for each
B. Use provider aliases and specify them in each resource
C. Use a wrapper script
D. Use different backend blocks in same module

**15.** You are provisioning a GCP Cloud Function with Terraform. The deployment fails due to missing IAM role binding. What should be your fix?
A. Retry terraform apply
B. Add IAM binding manually
C. Add google_project_iam_member in the same plan
D. Set depends_on for role

## Answers Key

| **Q** | **Ans** | **Q** | **Ans** | Q | Ans |
|---|---|---|---|---|---|
| 1 | B | 6 | B | 11 | B |
| 2 | C | 7 | B | 12 | B |
| 3 | B | 8 | B | 13 | B |
| 4 | A | 9 | C | 14 | B |
| 5 | C | 10 | A | 15 | C |

## Mock Test 6: Azure & Hybrid Cloud Scenarios

**1.** You're deploying an Azure App Service with Terraform and notice delays during apply. You want to **limit the retry timeout**. What should you do?
A. Use depends_on
B. Set timeouts block with custom values
C. Add lifecycle { create_before_destroy = true }
D. Use a conditional module

**2.** You need to provision an Azure resource group and assign role-based access. Which combination is correct?
A. Use azurerm_role_assignment after azurerm_resource_group
B. Create a null_resource with provisioner
C. Use Azure CLI inside a local-exec
D. Add role inside provider block

**3.** You're managing Azure VNets in multiple regions using a single module. Which feature allows different naming per region?
A. locals with conditional expressions
B. terraform.workspace
C. count with list of maps
D. for_each with a map keyed by region

**4.** You are provisioning Azure resources using a **custom service principal**. What must be explicitly declared in your provider block?
A. use_msi = true
B. subscription_id, client_id, client_secret, tenant_id
C. environment = "public"
D. key_vault_id and cert_thumbprint

**5.** You are provisioning across Azure and AWS from one repo. How do you ensure resources map to their cloud?
A. Separate .tf files by cloud
B. Use provider aliases and assign them to resources/modules
C. Create two state files manually
D. Run Terraform from two machines

**6.** You have a manually created Azure Key Vault. You want to **bring it under Terraform control** without changes. What do you do?
A. Write config and use terraform apply
B. Use terraform taint
C. Run terraform import followed by terraform plan
D. Use remote-exec to sync

## Mock Test 7: CI/CD Pipeline + GitOps with Terraform

**7.** Your team runs Terraform via GitHub Actions. They want to store plan output for review before apply. What should you do?
A. Pipe output to a log file
B. Use terraform plan -out=tfplan and upload as artifact
C. Store output in environment variable
D. Use terraform graph

**8.** In your Terraform GitOps pipeline, a contributor accidentally commits a .tfstate file. How can you prevent this?
A. Use .terraformignore
B. Add *.tfstate to .gitignore
C. Enable backend encryption
D. Use workspace isolation

**9.** You are running Terraform in a CI pipeline and need to **automate secrets injection securely**. What's the recommended approach?
A. Store secrets in .tfvars
B. Use environment variables managed by the CI tool
C. Commit encrypted files to Git
D. Read from a public S3 bucket

**10.** You are deploying infrastructure to different environments (dev, qa, prod) with a CI/CD pipeline. What Terraform feature supports this?
A. Terraform Enterprise
B. Workspaces and variable files
C. Module versioning
D. Resource aliases

**11.** In your Jenkins pipeline, Terraform fails with: Error locking state. What should you investigate first?
A. Check access to backend
B. Delete all modules
C. Use terraform destroy
D. Change provider version

**12.** In a Terraform CI pipeline, you want to **lint and format** your code before deployment. What command helps?
A. terraform validate
B. terraform show
C. terraform fmt -check
D. terraform apply -auto-approve

**13.** You want to apply changes only if the Terraform plan file has **no drift or change**. Which sequence ensures safety?
A. Plan > Manual review > Apply with plan file
B. Plan > Apply without review
C. Use terraform destroy > Apply
D. Plan twice then apply

## Answer Key

| Q | Ans | Q | Ans | Q | Ans |
|---|---|---|---|---|---|
| 1 | B | 6 | C | 11 | A |
| 2 | A | 7 | B | 12 | C |
| 3 | D | 8 | B | 13 | A |
| 4 | B | 9 | B | | |
| 5 | B | 10 | B | | |

## Mock Test 8: Advanced HCL – dynamic, count vs for_each, lifecycle, functions

**1.** You want to conditionally create multiple subnets in a module. Which approach is best to handle optional resources cleanly?
A. Use count with length() of subnet list
B. Use for_each directly
C. Use dynamic block inside resource
D. Use depends_on with null_resource

**2.** When should you prefer for_each over count in Terraform?
A. When dealing with a known fixed number
B. When each instance needs a unique key
C. When resources are conditional
D. To avoid dynamic blocks

**3.** You want to dynamically generate multiple security rules inside a single azurerm_network_security_group. How can this be done in Terraform?
A. Use for_each on the resource
B. Use multiple resource blocks
C. Use dynamic blocks inside the resource
D. Use count with inline map

**4.** You defined a count on a resource, then changed to for_each. Terraform fails on plan. What's the cause?
A. Module caching
B. For_each doesn’t accept list

C. Terraform sees this as different resources — needs taint or import
D. Provider version mismatch

**5.** You want to prevent a resource from being destroyed during terraform destroy. Which config block do you use?
A. lifecycle { ignore_changes = [...] }
B. count = 0
C. prevent_destroy = true inside lifecycle
D. depends_on = []

**6.** You're using terraform console to test expressions. What does flatten([["a", "b"], ["c"]]) return?
A. [["a", "b", "c"]]
B. ["a", "b", "c"]
C. ["ab", "c"]
D. [[a], [b], [c]]

## Mock Test 9: Terraform Cloud, Remote State, Importing, and Workspaces

**7.** You're using **Terraform Cloud**. Where is the state file stored by default?
A. In S3 bucket
B. In your workspace directory
C. Remotely managed by Terraform Cloud
D. On your local backend

**8.** How do you allow multiple users to collaborate and not conflict on state in Terraform Cloud?
A. Use -lock=false
B. Store state locally
C. Enable remote backend with state locking
D. Push state manually

**9.** You've created an S3 backend and want to **migrate your local state** to it. What command is used?
A. terraform push
B. terraform remote init
C. terraform init -migrate-state
D. terraform init (with confirmation)

**10.** You want to bring an existing cloud resource under Terraform management. What is the correct approach?

A. Copy config and run apply
B. Use terraform taint
C. Use terraform import and then define matching .tf code
D. Use remote-exec and recreate

**11.** Your S3 backend is throwing “AccessDenied” during plan. What should you check first?
A. terraform apply -auto-approve
B. Check bucket region
C. IAM policy attached to the user/role
D. Delete the lock file

**12.** You are using multiple workspaces (dev, qa, prod). How do you reference workspace-specific variables?
A. terraform.workspace inside locals
B. Use count to manage resources
C. Set via environment variables only
D. Use backend block with variables

**13.** You run terraform workspace select qa, then terraform apply. What is the impact?
A. Applies to all workspaces
B. Applies changes specific to qa's state
C. Deletes the default workspace
D. Destroys and re-creates all resources

## Answer Key

| Q | Ans | Q | Ans | Q | Ans |
|---|---|---|---|---|---|
| 1 | A | 6 | B | 11 | C |
| 2 | B | 7 | C | 12 | A |
| 3 | C | 8 | C | 13 | B |
| 4 | C | 9 | D | | |
| 5 | C | 10 | C | | |

## Thanks Everyone!

**Connect with me: Amit Singh**