

# TERRAFORM

---

## 1. Which Terraform command initializes working directory plugins, modules, and backend configuration?

- A. terraform refresh
- B. terraform init
- C. terraform providers
- D. terraform get

**Answer: B**

Explanation: 'terraform init' downloads provider plugins, sets up required modules, and configures the backend so the working directory is ready for plan and apply; the other commands operate on already-initialized directories.

## 2. What is the default state file name created in local backend when you run 'terraform apply'?

- A. terraform.state.json
- B. terraform.tfvars
- C. terraform.tfstate
- D. terraform.state

**Answer: C**

Explanation: Terraform stores local state in 'terraform.tfstate' by default, which tracks resource attributes; the other filenames are used for variable definitions or not used at all.

## 3. Which backend type allows storing Terraform state in Amazon S3 with state locking capabilities?

- A. local
- B. s3
- C. gcs
- D. remote

**Answer: B**

Explanation: The 's3' backend integrates with Amazon S3 buckets and can use DynamoDB for locking, providing remote storage and concurrency protection; local and GCS lack that AWS integration.

## 4. In Terraform 1.x, which block is used to declare support for specific Terraform versions?

- A. terraform\_version
- B. required\_providers
- C. terraform

D. provider

**Answer: C**

Explanation: The top-level 'terraform' block can include a 'required\_version' attribute that constrains which Terraform versions may run the configuration; other blocks cannot enforce the CLI version.

**5. How does Terraform determine the order in which resources are created?**

- A. Random order each run
- B. Alphabetical by resource type
- C. Reverse plan order
- D. Dependency graph analysis

**Answer: D**

Explanation: Terraform builds a dependency graph from references between resources and creates them in an order that satisfies those dependencies, ensuring prerequisites exist before dependents.

**6. Which command refreshes Terraform state to match remote objects without modifying infrastructure?**

- A. terraform output
- B. terraform apply -refresh-only
- C. terraform plan -destroy
- D. terraform destroy

**Answer: B**

Explanation: 'terraform apply -refresh-only' reads the current remote object states and updates the state file without changing infrastructure, serving as the refresh command in newer versions.

**7. What is the primary purpose of the 'terraform fmt' command?**

- A. Validate syntax
- B. Format HCL files to canonical style
- C. Show resource dependencies
- D. Force module re-download

**Answer: B**

Explanation: 'terraform fmt' automatically reformats configuration files to the standard Terraform style, improving readability while leaving semantics unchanged; it is not a validator or loader.

**8. Which block in Terraform is used to configure providers?**

- A. resource
- B. provider
- C. module
- D. backend

**Answer: B**

Explanation: A 'provider' block sets credentials, regions, and other options for a given provider, whereas 'resource' declares infrastructure and 'backend' is used within the 'terraform' block only.

**9. When using 'count' meta-argument, how can you reference the index of the current instance?**

- A. count.number
- B. count.index
- C. count.id
- D. count.iteration

**Answer: B**

Explanation: Within a resource using 'count', the 'count.index' value exposes the zero-based index of the current instance, useful for naming and referencing unique attributes.

**10. What happens if you run 'terraform apply' without first running 'terraform plan'?**

- A. Terraform fails with an error
- B. Apply executes, but Terraform auto-generates a plan internally
- C. Terraform ignores state updates
- D. Terraform loads last saved plan

**Answer: B**

Explanation: 'terraform apply' implicitly creates an execution plan and prompts for approval, so running 'plan' first is optional though helpful for review; there is no stored plan reuse by default.

**11. Which Terraform configuration element allows defining reusable infrastructure patterns?**

- A. Data sources
- B. Providers
- C. Modules
- D. Backends

**Answer: C**

Explanation: Modules package resources and logic so they can be reused and shared, whereas providers connect to APIs and data sources read existing infrastructure.

**12. Which file is not loaded automatically by Terraform?**

- A. terraform.auto.tfvars
- B. terraform.tfvars.json
- C. variables.tf
- D. production.tfvars

**Answer: D**

Explanation: Files named '\*.auto.tfvars' or 'terraform.tfvars[.json]' are auto-loaded, but environment-specific files like 'production.tfvars' must be passed with '-var-file'.

**13. What is the effect of setting 'lifecycle { prevent\_destroy = true }' on a resource?**

- A. Disables updates
- B. Ensures resource is always recreated
- C. Blocks deletion through Terraform
- D. Ignores configuration drift

**Answer: C**

Explanation: 'prevent\_destroy' tells Terraform to error out rather than destroy the resource, helping protect critical infrastructure from being removed accidentally.

**14. Which interpolation function merges multiple maps into one?**

- A. merge()
- B. concat()
- C. join()
- D. zipmap()

**Answer: A**

Explanation: 'merge()' takes several maps and returns a single map combining their keys, with later arguments overriding earlier ones, whereas functions like 'concat()' operate on lists.

**15. Which backend supports workspaces natively without remote execution?**

- A. artifactory
- B. consul
- C. s3
- D. local

**Answer: D**

Explanation: The 'local' backend stores state on disk and supports workspaces locally by naming files, even though it lacks remote execution features found in Terraform Cloud backends.

**16. In Terraform, what does the 'depends\_on' meta-argument do?**

- A. Sets resource count
- B. Forces explicit dependency
- C. Configures provider version
- D. Enables remote backend

**Answer: B**

Explanation: 'depends\_on' allows you to declare an explicit dependency when Terraform cannot infer it from references, ensuring dependency ordering in the graph.

**17. How can you override a variable value when running 'terraform apply'?**

- A. terraform apply --override var=value
- B. terraform apply -var="key=value"
- C. terraform apply var.key=value

D. terraform apply -set key=value

**Answer: B**

Explanation: Passing '-var="name=value"' on the CLI sets or overrides a variable for that run; the other syntaxes are not valid Terraform CLI options.

**18. Which command removes Terraform-managed infrastructure?**

- A. terraform clean
- B. terraform destroy
- C. terraform purge
- D. terraform rm

**Answer: B**

Explanation: 'terraform destroy' creates a plan that deletes all managed resources, whereas the other commands do not exist or serve different purposes.

**19. What is the purpose of a 'data' block in Terraform?**

- A. Define resource outputs
- B. Reference existing remote objects
- C. Create new resources
- D. Configure backend

**Answer: B**

Explanation: Data sources ('data' blocks) query external resources so their attributes can be used in configuration without managing them directly.

**20. Which expression best retrieves an output named 'db\_endpoint' from module 'database'?**

- A. module("database").output(db\_endpoint)
- B. database.db\_endpoint
- C. module.database.db\_endpoint
- D. var.database.db\_endpoint

**Answer: C**

Explanation: Module outputs are referenced via 'module.<name>.<output>', so 'module.database.db\_endpoint' reads the value computed by the child module.

**21. What will 'terraform state list' display?**

- A. Pending plan changes
- B. Current resources tracked in state
- C. Available modules
- D. Configured providers

**Answer: B**

Explanation: 'terraform state list' enumerates all resource addresses currently stored in the state file, helping inspect what Terraform manages.

**22. How do you import an existing resource into Terraform state?**

- A. terraform load <address> <id>

- B. terraform import <address> <id>
- C. terraform attach <address> <id>
- D. terraform bring <address> <id>

**Answer: B**

Explanation: 'terraform import' maps an existing infrastructure object (identified by ID) to a resource address in state so Terraform can manage it going forward.

**23. What does 'terraform taint' mark?**

- A. A variable for removal
- B. A backend for reconfiguration
- C. A resource instance for recreation
- D. A module for clean-up

**Answer: C**

Explanation: Marking a resource as tainted tells Terraform it must be destroyed and recreated during the next apply, useful when an object is faulty.

**24. Which statement about local values is correct?**

- A. They persist across runs in state
- B. They are evaluated once per workspace
- C. They provide named expressions for reuse in configuration
- D. They can only be set via CLI flags

**Answer: C**

Explanation: Local values store temporary named expressions within a module for readability and reuse, and do not appear in state or require CLI input.

**25. The 'terraform validate' command primarily checks for:**

- A. Provider connectivity
- B. Syntax and internal consistency
- C. State file corruption
- D. Plan drift

**Answer: B**

Explanation: 'terraform validate' parses configuration to detect syntax errors and some semantic issues without hitting providers; it doesn't contact cloud APIs.

**26. Which argument configures a module source hosted in a Git repository?**

- A. source = "git::https://example.com/repo.git"
- B. repo = "https://example.com/repo.git"
- C. module = "https://example.com/repo.git"
- D. git\_source = "https://example.com/repo.git"

**Answer: A**

Explanation: A module block's 'source' attribute can reference Git by prefixing with 'git::', allowing Terraform to fetch the module from version control.

**27. What is a Terraform workspace useful for?**

- A. Version pinning providers
- B. Isolating state files within the same configuration
- C. Enabling HCL formatting
- D. Sharing variables between modules

**Answer: B**

Explanation: Workspaces let you maintain separate state snapshots (e.g., dev, prod) while reusing the same configuration directory.

**28. Which feature detects configuration drift by comparing real infrastructure with state?**

- A. terraform drift
- B. terraform check
- C. terraform plan
- D. terraform graph

**Answer: C**

Explanation: 'terraform plan' refreshes state and compares desired configuration to actual resources, revealing drift before apply.

**29. How can you lock provider versions for reproducibility?**

- A. Required\_providers with version constraints
- B. terraform lock command
- C. Setting versions in variables.tf
- D. Using terraform freeze

**Answer: A**

Explanation: Inside the 'terraform' block, 'required\_providers' allows setting version constraints so the same provider versions are used across runs.

**30. When 'terraform plan' reports '+/-' on a resource, what does it mean?**

- A. Resource unaffected
- B. Resource will be created twice
- C. Resource will be destroyed and recreated
- D. Resource is locked

**Answer: C**

Explanation: The '+/-' indicator signals Terraform must replace the resource (destroy then create) because certain changes cannot be applied in place.

**31. Which file stores dependency lock information for providers in Terraform 0.14+?**

- A. terraform.lock.hcl
- B. provider.lock
- C. terraform.tfstate.lock
- D. lock.providers.hcl

**Answer: A**

Explanation: 'terraform.lock.hcl' is generated by 'terraform init' to record exact provider versions, ensuring reproducible installs in collaborative workflows.

**32. How do 'for\_each' and 'count' differ?**

- A. for\_each accepts maps/sets enabling stable keys per instance
- B. for\_each is limited to numbers only
- C. count can use strings as keys
- D. They behave identically

**Answer: A**

Explanation: 'for\_each' iterates over maps or sets, giving each instance a stable key so renumbering doesn't destroy resources, unlike 'count' which is index-based.

**33. Which command upgrades modules and providers to the latest allowed versions?**

- A. terraform refresh
- B. terraform init -upgrade
- C. terraform providers lock
- D. terraform apply --upgrade

**Answer: B**

Explanation: Re-running 'terraform init -upgrade' forces Terraform to re-check registry versions within allowed constraints and update local plugins/modules accordingly.

**34. How can you visualize the dependency graph of resources?**

- A. terraform viz
- B. terraform graph
- C. terraform topology
- D. terraform show --graph

**Answer: B**

Explanation: 'terraform graph' outputs the dependency graph in DOT format, which can be rendered to visualize relationships among resources.

**35. What is the effect of 'terraform apply -auto-approve'?**

- A. Skips backend initialization
- B. Runs apply without interactive approval prompt
- C. Forces locking off
- D. Enables test mode

**Answer: B**

Explanation: The '-auto-approve' flag tells Terraform not to prompt for confirmation, useful in automation pipelines where manual approval isn't possible.

**36. Which meta-argument limits a resource to a specific provider configuration alias?**

- A. provider
- B. alias
- C. depends\_on

D. provisioner

**Answer: A**

Explanation: You can specify 'provider = aws.us\_east' for example, to bind a resource to a particular aliased provider configuration; the 'alias' attribute is defined inside the provider block.

**37. What does Terraform use to track the mapping between configuration resources and real resources?**

- A. Workspaces
- B. State file
- C. Modules
- D. Variables

**Answer: B**

Explanation: Terraform's state file maintains the mapping from resource addresses to remote objects, holding attributes needed for subsequent plans and applies.

**38. In Terraform expressions, how do you denote string interpolation using variables?**

- A. \${var.name}
- B. @{var.name}
- C. #{var.name}
- D. \$(var.name)

**Answer: A**

Explanation: HCL interpolations use '\${...}' syntax, so '\${var.name}' inserts the variable value into a string.

**39. Which provisioner runs commands on a local machine after resources are created?**

- A. remote-exec
- B. local-exec
- C. file
- D. null-provisioner

**Answer: B**

Explanation: 'local-exec' executes a command on the machine running Terraform (e.g., your workstation or CI runner), whereas 'remote-exec' runs on the remote resource.

**40. When using remote backends, how is state locking typically handled?**

- A. Not supported
- B. Through backend-specific mechanisms
- C. By locking the configuration files
- D. Using local .lock file only

**Answer: B**

Explanation: Remote backends such as S3+DynamoDB or Terraform Cloud provide their own locking implementations to prevent concurrent state modifications.

**41. Which Terraform block lets you define variables with default values?**

- A. variable
- B. locals
- C. data
- D. resource

**Answer: A**

Explanation: A 'variable' block defines input variables, and its 'default' argument provides a fallback value when the caller doesn't supply one.

**42. What does the 'terraform output -json' command do?**

- A. Shows outputs in machine-readable JSON format
- B. Imports outputs from JSON file
- C. Exports outputs to remote backend
- D. Converts outputs to YAML

**Answer: A**

Explanation: The '-json' flag prints outputs as JSON, making it easier to parse programmatically in scripts or CI pipelines.

**43. How can you target specific resources during 'terraform apply'?**

- A. terraform apply --resource <type>
- B. terraform apply -target=<resource\_address>
- C. terraform apply --only <resource\_type>
- D. terraform apply -scope=<name>

**Answer: B**

Explanation: The '-target' flag narrows the plan/apply to specific resource addresses, though it's recommended only for exceptional cases.

**44. Which statement about provisioners is recommended by HashiCorp?**

- A. Use them extensively for resource setup
- B. Avoid when possible; prefer declarative configuration
- C. Use only in modules
- D. They are mandatory for remote resources

**Answer: B**

Explanation: HashiCorp advises minimizing provisioner usage because they are imperative and can complicate lifecycle management compared to declarative configuration via providers.

**45. What command converts the current Terraform state file into a human-readable format?**

- A. terraform plan
- B. terraform fmt
- C. terraform show
- D. terraform graph

**Answer: C**

Explanation: 'terraform show' displays the state (or a plan file) in a readable format, letting you inspect resource attributes.

**46. What is the primary function of 'terraform state rm'?**

- A. Remove resources from configuration
- B. Delete resources in cloud
- C. Remove resource from state without destroying it
- D. Reset backend

**Answer: C**

Explanation: 'terraform state rm' forgets a resource so Terraform stops managing it, useful before importing under a different address or when something becomes unmanaged.

**47. Which Terraform argument ensures that changes to a resource attribute are ignored during apply?**

- A. ignore\_changes in lifecycle block
- B. skip\_changes property
- C. ignore attribute changes meta-argument
- D. attr\_ignore in provider

**Answer: A**

Explanation: 'lifecycle { ignore\_changes = [attribute] }' tells Terraform not to act on differences for that attribute, preventing unnecessary updates.

**48. What occurs when two users run 'terraform apply' concurrently against the same remote backend that supports locking?**

- A. Both applies succeed
- B. One apply waits or fails due to state lock
- C. The state merges automatically
- D. Terraform switches to local state

**Answer: B**

Explanation: With locking, the backend grants state access to one operation at a time so overlapping applies either wait for the lock or error out if they cannot obtain it.

**49. Why might you use 'terraform login'?**

- A. Authenticate with Terraform Cloud/Enterprise for remote operations
- B. Enable local backend
- C. Apply MFA to resources
- D. Initialize providers

**Answer: A**

Explanation: 'terraform login' obtains an API token and stores it for Terraform Cloud/Enterprise so remote backends and operations can authenticate securely.

**50. Which command will show the difference between the last saved plan and the configuration?**

- A. terraform show
- B. terraform compare
- C. terraform diff
- D. terraform inspect

**Answer: A**

Explanation: If you save a plan with '-out', 'terraform show' can display its contents, highlighting intended changes relative to current state and configuration.

**51. In Terraform, what does 'sensitive = true' on an output do?**

- A. Encrypts stored state
- B. Hides value from CLI output
- C. Blocks the output
- D. Forces variable encryption

**Answer: B**

Explanation: Marking an output as sensitive suppresses it from normal CLI display to avoid leaking secrets, though the value is still present in state files.

**52. What is required to use workspaces with the S3 backend?**

- A. Versioning disabled
- B. DynamoDB table is mandatory
- C. 'workspace\_key\_prefix' configuration
- D. Separate bucket per workspace

**Answer: C**

Explanation: You must set 'workspace\_key\_prefix' so each workspace's state file is stored under a unique key prefix in the same bucket.

**53. Which terraform block declares the providers required for a module?**

- A. terraform { required\_providers { ... } }
- B. providers { ... }
- C. provider\_declaration { ... }
- D. module { required\_providers { ... } }

**Answer: A**

Explanation: The 'terraform' block's 'required\_providers' map specifies which providers and versions a module depends on, letting Terraform fetch and verify them.

**54. How can you prevent Terraform from creating a plan file?**

- A. Use terraform plan -no-save
- B. Plans are only saved when explicitly output with -out flag
- C. Use terraform plan --dry-run
- D. Plans always create planfile by default

**Answer: B**

Explanation: Terraform does not write plan files unless you specify '-out'. Running 'plan' without it only displays the plan.

**55. What is a correct way to define a numeric variable with validation in Terraform?**

- A. variable "size" { default = 2 validation { condition = size > 0 error\_message = "positive" } }
- B. variable "size" { type = number validation { condition = var.size > 0 error\_message = "Must be positive" } }
- C. variable "size" { validation { constraint = number > 0 } }
- D. variable "size" { rule { size > 0 } }

**Answer: B**

Explanation: You declare the variable type, then within 'validation' reference 'var.size' in the condition; Terraform requires the full expression and an error message.

**56. Which function converts a list of strings into a single string separated by commas?**

- A. concat(list, ",")
- B. join(",", list)
- C. split(",", list)
- D. merge(",", list)

**Answer: B**

Explanation: 'join(delimiter, list)' concatenates list elements using the delimiter string; 'split' performs the opposite conversion.

**57. What is a recommended method to share modules internally?**

- A. Bundle modules into Terraform binary
- B. Publish to a private module registry
- C. Store modules in state file
- D. Email module files to team

**Answer: B**

Explanation: Hosting modules in a private registry (Terraform Cloud or an internal one) enforces versioning and access control better than ad hoc distribution.

**58. What happens when you run 'terraform destroy -target=aws\_instance.app'?**

- A. Destroys entire configuration
- B. Destroys only targeted resource along with dependencies
- C. Does nothing without plan
- D. Updates targeted resource

**Answer: B**

Explanation: Targeted destroy removes the specified resource and anything that depends on it, leaving unrelated infrastructure untouched.

**59. How can you output the raw state file as JSON?**

- A. terraform state show -json
- B. terraform state pull
- C. terraform show -json
- D. terraform output state

**Answer: B**

Explanation: 'terraform state pull' downloads the full state in JSON format from the backend, useful for scripting or inspection.

**60. When using 'for\_each' on a map, how do you reference the key within the resource?**

- A. each.key
- B. each.index
- C. each.name
- D. each.id

**Answer: A**

Explanation: 'each.key' returns the current map key when iterating with 'for\_each', enabling you to refer to meaningful identifiers rather than numeric indexes.

**61. Which block allows you to conditionally include expressions?**

- A. dynamic
- B. optional
- C. selective
- D. filter

**Answer: A**

Explanation: 'dynamic' blocks generate nested configuration only when a collection is non-empty, giving conditional control over nested arguments.

**62. Which Terraform Cloud workspace execution mode provides remote run capabilities?**

- A. Local execution mode
- B. Agent execution mode
- C. Remote execution mode
- D. Manual mode

**Answer: C**

Explanation: Remote execution mode runs Terraform on Terraform Cloud's infrastructure after you push configuration or trigger runs.

**63. What is the purpose of the 'terraform providers' command?**

- A. List providers required by configuration and their versions
- B. Install providers
- C. Validate provider credentials
- D. Compare provider versions

**Answer: A**

Explanation: 'terraform providers' surfaces the dependency graph of providers, showing which modules require which providers and version constraints.

**64. To prevent storing state locally when using Terraform Cloud, what should you configure?**

- A. CLI-driven run
- B. Remote backend pointing to Terraform Cloud workspace
- C. Additional state encryption
- D. Local backend with sync

**Answer: B**

Explanation: Configuring the 'remote' backend with a Terraform Cloud workspace ensures state is stored and locked remotely rather than on disk.

**65. Which CLI flag outputs plan results to a file for later apply?**

- A. terraform plan -save plan.out
- B. terraform plan -out=planfile
- C. terraform plan --write planfile
- D. terraform plan -plan planfile

**Answer: B**

Explanation: Using '-out' with 'terraform plan' writes the plan to a file that can be passed to 'terraform apply' later, guaranteeing consistency.

**66. What does the 'terraform console' command do?**

- A. Launches Terraform Cloud UI
- B. Provides interactive evaluation of expressions
- C. Streams logs from apply
- D. Opens provider shells

**Answer: B**

Explanation: 'terraform console' lets you evaluate interpolation expressions against the current state/variables, handy for debugging complex expressions.

**67. Which state locking mechanism is supported by the Consul backend?**

- A. PostgreSQL row locks
- B. DynamoDB locks
- C. Consul sessions
- D. Redis locks

**Answer: C**

Explanation: The Consul backend uses Consul sessions to acquire and release locks, preventing concurrent modification of the state stored in Consul KV.

**68. If a provider requires specific credentials, where is the best practice location to store them?**

- A. Hard-coded in configuration files

- B. Terraform variables checked into version control
- C. Environment variables or secure secrets manager
- D. Within outputs for reuse

**Answer: C**

Explanation: Credentials should be kept out of version control and stored securely, commonly via environment variables or secret stores that the provider can read at runtime.

**69. What is the effect of 'terraform state mv'?**

- A. Move resources between modules or addresses within state
- B. Rename backend
- C. Duplicate resource state
- D. Migrate state to new backend

**Answer: A**

Explanation: 'terraform state mv' reassigns state entries to a new resource address, useful for refactoring module structure without destroying resources.

**70. Which attribute of 'resource' is used to reference output attributes of created resources?**

- A. resource.name.attribute
- B. resource.type.attribute
- C. resource["name"].attribute
- D. resource.attribute

**Answer: A**

Explanation: Resource attributes are accessed as '<type>.<name>.<attribute>', for example 'aws\_instance.app.id', combining type and local name defined in configuration.

**71. How do you run a destroy plan without applying it?**

- A. terraform plan -destroy
- B. terraform destroy --plan-only
- C. terraform check -destroy
- D. terraform preview destroy

**Answer: A**

Explanation: Adding '-destroy' to 'terraform plan' shows what would be removed without actually destroying anything, allowing you to review impact.

**72. Which expression merges two lists while removing duplicates?**

- A. concat(list1, list2)
- B. union(list1, list2)
- C. distinct(concat(list1, list2))
- D. merge(list1, list2)

**Answer: C**

Explanation: 'concat' simply combines lists, so wrapping it with 'distinct()' filters duplicates, which provides a merged unique list.

**73. In Terraform Cloud, what does a Sentinel policy enforce?**

- A. Provider version upgrades
- B. Custom governance rules before apply
- C. Automatic module creation
- D. Backend migration

**Answer: B**

Explanation: Sentinel policies let organizations codify governance checks that must pass before Terraform Cloud applies a run, preventing policy violations.

**74. What will 'terraform apply plan.out' do?**

- A. Ignore plan and create a new one
- B. Apply previously saved plan file
- C. Print plan without applying
- D. Delete plan file

**Answer: B**

Explanation: Passing the saved plan file ensures the exact changes reviewed earlier are applied, guarding against drift that might result from re-planning.

**75. Which backend option enables state encryption at rest using Azure?**

- A. remote
- B. azurearm
- C. s3
- D. consul

**Answer: B**

Explanation: The 'azurearm' backend stores state in Azure Blob Storage and can leverage Azure's encryption features, providing native integration with Azure services.

**76. What is the recommended way to handle secrets in Terraform configurations?**

- A. Store them in version control
- B. Use Terraform outputs
- C. Leverage external secret management (e.g., Vault) and variables
- D. Embed in module source URL

**Answer: C**

Explanation: Sensitive values should be sourced from secret managers or environment variables and not committed or exposed via outputs to reduce risk.

**77. When using Terraform with Kubernetes provider, why might you set 'load\_config\_file = false'?**

- A. Enable interactive mode

- B. Skip kubeconfig loading and rely on explicit credentials
- C. Force provider upgrade
- D. Enable dynamic blocks

**Answer: B**

Explanation: Setting 'load\_config\_file = false' tells the Kubernetes provider to use credentials set in the provider block rather than reading kubeconfig files.

**78. Which of the following is true about 'terraform refresh' in 1.1+?**

- A. Deprecated; use 'terraform apply -refresh-only'
- B. Mandatory before every plan
- C. Only available for local backend
- D. Requires Terraform Enterprise

**Answer: A**

Explanation: HashiCorp deprecated 'terraform refresh' in favor of 'apply -refresh-only', consolidating refresh functionality into the apply workflow.

**79. How do you specify conditional expressions in Terraform?**

- A. condition ? true\_val : false\_val
- B. if condition then true\_val else false\_val
- C. (condition ? true\_val, false\_val)
- D. condition ? true\_val / false\_val

**Answer: A**

Explanation: Terraform adopts the ternary operator syntax 'condition ? true\_val : false\_val' for conditional expressions inside HCL.

**80. What does the 'terraform version' command output?**

- A. Current state schema
- B. Installed Terraform version and provider requirements
- C. Module versions
- D. Workspace list

**Answer: B**

Explanation: 'terraform version' prints the CLI version and any required provider versions recorded in the lock file.

**81. When using dynamic blocks, what does 'iterator' keyword customize?**

- A. Provider alias
- B. The name used within the block to reference each item
- C. The number of iterations
- D. The block type

**Answer: B**

Explanation: By default 'dynamic' uses 'each', but 'iterator' lets you rename the iteration variable to something clearer when referencing attributes.

**82. Which Terraform feature allows referencing the resource count in naming conventions?**

- A. format()
- B. count.index within expressions
- C. sprintf()
- D. resource.index

**Answer: B**

Explanation: Inside a counted resource, 'count.index' can be interpolated into names to produce unique identifiers, such as 'name = "app-\${count.index}"'.

**83. What is the effect of the 'retain\_on\_delete' lifecycle argument on S3 buckets?**

- A. It's not a valid lifecycle argument
- B. Retains resource when targeted destruction occurs
- C. Deletes bucket but retains objects
- D. Keeps bucket but empties objects

**Answer: A**

Explanation: 'retain\_on\_delete' is not a recognized Terraform lifecycle setting; valid options include 'create\_before\_destroy', 'prevent\_destroy', and 'ignore\_changes'.

**84. How do you apply a partial configuration using modules?**

- A. Use module 'count' and 'for\_each' to include modules based on conditions
- B. Use terraform partial apply
- C. Use provider filters
- D. Not possible

**Answer: A**

Explanation: Wrapping modules with conditionals such as 'count = var.enabled ? 1 : 0' allows you to include or skip infrastructure chunks dynamically.

**85. Which command prints all variables required without default values?**

- A. terraform variables
- B. terraform config-inspect -var-required
- C. terraform plan -var-check
- D. terraform fmt -var

**Answer: B**

Explanation: 'terraform config-inspect' with '-var-required' lists variables lacking defaults, helping ensure required inputs are provided.

**86. What is the correct syntax to read a value from a map variable?**

- A. var.map["key"]
- B. var.map.key()
- C. var.map.key[]
- D. var.map->key

**Answer: A**

Explanation: Terraform supports both index and dot notation, but for keys with punctuation or dynamic values, `'var.map["key"]'` is the canonical syntax.

**87. Which backend is recommended for collaboration with Terraform Cloud?**

- A. artifactory
- B. remote
- C. json
- D. virtual

**Answer: B**

Explanation: The 'remote' backend is designed to connect to Terraform Cloud/Enterprise workspaces, supporting collaboration, locking, and remote runs.

**88. How does Terraform treat unknown values during plan?**

- A. Always assumes defaults
- B. Represents them as 'known after apply'
- C. Treats as zero values
- D. Blocks the plan

**Answer: B**

Explanation: Until apply time, Terraform shows placeholders like '(known after apply)' for attributes determined by the provider only after resource creation.

**89. What is the effect of applying 'terraform apply -refresh=false'?**

- A. Prevents Terraform from checking real infrastructure before apply
- B. Disables plan execution
- C. Clears state cache
- D. Enables CLI-driven runs

**Answer: A**

Explanation: The '-refresh=false' flag skips the state refresh step, useful when API limits or known drift should be ignored temporarily.

**90. Which statement about modules is correct?**

- A. Modules cannot be nested
- B. Modules can expose outputs to parent modules
- C. Modules must reside in the root
- D. Modules run after resources

**Answer: B**

Explanation: Modules can output values that the calling module can reference, allowing data to flow upward; they can also be nested arbitrarily.

**91. How do you upgrade only a single module's dependencies?**

- A. `terraform init -upgrade -module=module.name`
- B. `terraform get -update=module.name`
- C. `terraform init -upgrade -target=module.name`

D. terraform apply --upgrade module.name

**Answer: A**

Explanation: Using 'terraform init -upgrade -module=module.name' refreshes that module's source, leaving others untouched.

**92. What is the purpose of 'terraform force-unlock'?**

- A. Reset provider caching
- B. Release a stuck state lock using lock ID
- C. Delete remote workspaces
- D. Force destroy protected resources

**Answer: B**

Explanation: If a previous run crashed leaving a lock, 'terraform force-unlock <ID>' clears it so subsequent operations can proceed.

**93. Which interpolation function builds a map from two lists?**

- A. zipmap(keys, values)
- B. merge(keys, values)
- C. combine(keys, values)
- D. pair(keys, values)

**Answer: A**

Explanation: 'zipmap' pairs keys and values lists into a map, ideal for constructing lookup tables from two parallel lists.

**94. How can you run a subset of tests using 'terraform test'?**

- A. terraform test -run=<regex>
- B. terraform test -select=<name>
- C. terraform test --module=<name>
- D. terraform test -filter=<label>

**Answer: A**

Explanation: The '-run' flag filters which tests execute based on regex matching of test names, similar to Go's testing framework.

**95. Which CLI flag allows specifying a different state file location?**

- A. terraform apply -state=path
- B. terraform plan -state=path
- C. terraform state -state=path
- D. terraform init -state=path

**Answer: C**

Explanation: Commands under 'terraform state' accept '-state' to operate on an alternate state file, which is useful for advanced state editing tasks.

**96. In Terraform, a 'null\_resource' is typically used for:**

- A. Provisioning physical hardware

- B. Placeholder for provisioners or triggers without real resources
- C. Backend migration
- D. Variable defaults

**Answer: B**

Explanation: 'null\_resource' lets you run provisioners or use 'triggers' to execute logic even when no provider-managed resource is involved.

**97. What does 'terraform env list' do in Terraform 0.11 and earlier?**

- A. List environment variables
- B. List available workspaces
- C. Create new environments
- D. Delete old state

**Answer: B**

Explanation: Prior to workspaces being renamed, 'terraform env list' displayed the available environments (now known as workspaces), maintaining compatibility.

**98. How do you access attributes of an object variable?**

- A. var.object.attribute
- B. var.object["attribute"]
- C. Both dot and index notation
- D. Neither is allowed

**Answer: C**

Explanation: Object variables support both dot and bracket syntax, letting you choose whichever best matches the attribute naming.

**99. When configuring remote exec provisioner, which connection type allows using SSH?**

- A. connection { type = "ssh" }
- B. connect { protocol = "ssh" }
- C. ssh { enabled = true }
- D. remote { type = "ssh" }

**Answer: A**

Explanation: Setting 'connection { type = "ssh" ... }' defines how 'remote-exec' and 'file' provisioners connect to the target resource, enabling SSH sessions.

**100. Which statement about Terraform providers is accurate?**

- A. Providers map Terraform resources to external APIs
- B. Providers only manage local files
- C. Providers are optional for resource creation
- D. Providers cannot be versioned

**Answer: A**

Explanation: Providers implement resource and data source logic by interacting with external APIs, and Terraform requires them to manage infrastructure.