✅ Terraform Associate Certification Concepts (v1.3/003)

🧱 1. Understand Infrastructure as Code (IaC) concepts.

- Benefits of IaC (version control, automation, consistency)
- Declarative vs imperative programming
- Why Terraform vs other tools (e.g., CloudFormation, Ansible)
- Immutable infrastructure
- Idempotency
- Dependency management in IaC

⚙️ 2. Understand Terraform's purpose (vs. other IaC tools)

- What Terraform is (open-source IaC tool)
- Multi-cloud and provider-agnostic support
- Declarative syntax via HCL
- Use cases for Terraform (provisioning, compliance, CI/CD integration)
- Comparison with tools like Ansible, Pulumi, Chef

🧠 3. Understand Terraform basics

- Terraform workflow: init, plan, apply, destroy
- Providers and resources
- Modules (root, child, external)
- Input and output variables
- State management: terraform.tfstate
- Backends (local, remote, S3, Terraform Cloud)
- Configuration structure: .tf, .tfvars, terraform.tfstate
- Lifecycle rules (create_before_destroy, prevent_destroy)
- Meta-arguments: depends_on, count, for_each

🔐 4. Use Terraform CLI (outside of core workflow)

- terraform console
- terraform fmt
- terraform validate
- terraform taint, untaint
- terraform state subcommands (show, rm, mv, list)
- terraform output
- terraform providers
- terraform import

- terraform workspace
- CLI debugging: TF_LOG, TF_VAR_*
- Plan file usage: terraform plan -out=planfile, terraform apply planfile

---

🔍 5. Interact with Terraform modules

- Use publicly available modules from Terraform Registry
- Call child modules using module block
- Pass variables into modules
- Access module outputs in root module
- Create and structure a custom module (main.tf, variables.tf, outputs.tf)
- Version pinning for modules
- Use of local and remote modules (GitHub, Terraform Registry, file paths)

📁 6. Navigate Terraform workflow

- Order of operations: init → plan → apply
- Apply without prompt: -auto-approve
- Plan for changes: terraform plan
- Destroy infrastructure: terraform destroy
- Lock file (.terraform.lock.hcl)
- Provider constraints (required_providers)
- Upgrade providers: terraform init -upgrade

🗂️ 7. Implement and maintain state

- Purpose of Terraform state
- State file format: JSON
- Local vs remote state storage
- State locking (DynamoDB for S3 backend)
- State management commands (state list, state rm, state mv)
- Backups and .tfstate.backup
- State drift and refresh
- Sensitive data in state

☁️ 8. Read, generate, and modify configuration

- HCL (HashiCorp Configuration Language) syntax
- Variables (variable, default, type)
- Output values
- Data sources (data block)

- Locals (locals {})
- Functions (length, merge, lookup, join, split, etc.)
- Expressions: for, count, for_each, conditionals
- Dynamic blocks
- Resource addressing (aws_instance.myvm.id)
- Terraform files hierarchy and loading order

🧰 9. Understand Terraform Cloud and Enterprise capabilities

- Remote operations (Terraform Cloud CLI backend)
- Workspaces in Terraform Cloud
- Variable sets (environment vs Terraform variables)
- Terraform Cloud VCS integration
- Sentinel policies (Enterprise only)
- Remote backends: state storage, access control
- Cost estimation
- Remote state versioning
- Role-based access control (RBAC)

🛡️ 10. Understand security best practices

- Use sensitive = true for outputs
- Avoid storing state files in version control
- Use remote backends with encryption
- Secure provider credentials via environment variables
- Restrict access to state via backend (e.g., S3 + IAM)
- Avoid plaintext secrets in .tf files
- Use workspaces to separate environments

📚 Bonus: Frequently Covered Topics in Exams

- Importing existing infrastructure
- File structure of a Terraform project
- Handling secrets securely (e.g., with Vault or environment variables)
- CI/CD integration (GitHub Actions, Jenkins, etc.)
- Difference between count and for_each
- Troubleshooting failed applies
- Using the terraform graph command

🔍 1. Terraform Basics (1–25)

1. What command initializes a new Terraform working directory?
   ✅ B. terraform init
2. Which file stores your infrastructure's current state?
   ✅ C. terraform.tfstate
3. What does terraform plan do?
   ✅ C. Shows proposed changes without applying
4. Which best describes Infrastructure as Code (IaC)?
   ✅ C. Declarative configuration stored in version control
5. Terraform is primarily what type of tool?
   ✅ B. Infrastructure orchestration
6. What is the main configuration language used in Terraform?
   ✅ A. HCL
7. What does terraform apply do?
   ✅ C. Provisions resources
8. What command removes infrastructure created by Terraform?
   ✅ D. terraform destroy
9. What command verifies if the Terraform files are syntactically valid?
   ✅ C. terraform validate
10. How is state stored by default?
    ✅ B. In a local file
11. What does .terraform.lock.hcl do?
    ✅ C. Locks provider versions
12. Which of these is NOT a Terraform command?
    ✅ D. terraform config
13. How does Terraform identify resources uniquely?
    ✅ C. By type and name
14. Which of the following is NOT a Terraform file extension?
    ✅ C. .yaml
15. What happens if you rerun terraform apply without changes?
    ✅ A. No actions performed
16. What does terraform version show?
    ✅ C. The Terraform CLI and provider versions
17. What is a "resource" in Terraform?
    ✅ B. An object to manage infrastructure
18. What command shows all resources in the current state?
    ✅ A. terraform state list

19. What is Terraform Cloud?
    ✅ D. A hosted platform for remote state and collaboration
20. What command previews execution plans in human-readable format?
    ✅ B. terraform show
21. Which of the following is required in every Terraform configuration?
    ✅ C. Provider block
22. What is the default backend if none is configured?
    ✅ A. local
23. What command formats Terraform code?
    ✅ C. terraform fmt
24. How is sensitive data protected in Terraform output?
    ✅ D. Using sensitive = true
25. What file extension is used for variable definitions?
    ✅ B. .tfvars

💡 2. HCL Configuration Language (26–40)

26. What block defines a variable?
    ✅ B. variable
27. What HCL construct allows sharing logic across configs?
    ✅ A. Module
28. How do you reference a resource aws_vpc.main id?
    ✅ C. aws_vpc.main.id
29. What does the output block do?
    ✅ A. Displays values after apply
30. Which keyword defines a reusable configuration?
    ✅ C. module
31. What is a local value in Terraform?
    ✅ D. A named expression to simplify logic
32. What is the syntax to assign a string to a variable?
    ✅ B. variable = "value"
33. What keyword provides dynamic input at runtime?
    ✅ B. variable
34. What is a map?
    ✅ A. A collection of key-value pairs
35. How do you loop through a list to create multiple resources?
    ✅ B. Using count or for_each
36. What keyword is used to reference Terraform functions?
    ✅ D. They are used directly (no keyword)

37. How do you create a conditional resource?
    ✅ A. Use count = var.create ? 1 : 0
38. Which function merges two maps?
    ✅ C. merge()
39. What HCL feature lets you use dynamic content?
    ✅ B. dynamic blocks
40. What is the type of true or false in HCL?
    ✅ A. bool

🗃️ 3. State Management (41–60)

41. What is a Terraform state file used for?
    ✅ B. To track infrastructure resources managed by Terraform
42. What is the default location of the state file?
    ✅ C. terraform.tfstate in the current directory
43. How can you prevent simultaneous updates to state?
    ✅ A. Enable state locking
44. What backend feature supports locking with DynamoDB?
    ✅ D. S3 backend with DynamoDB table
45. How do you inspect detailed information about a specific resource in the state?
    ✅ C. terraform state show
46. What command is used to remove an item from state without destroying it?
    ✅ B. terraform state rm
47. Which backend is used for remote collaboration in Terraform Cloud?
    ✅ A. remote
48. Why is it not recommended to store state files in VCS?
    ✅ C. They may contain sensitive data
49. How can you migrate a local backend to a remote one?
    ✅ D. Configure and re-run terraform init
50. What command lets you manually import resources into the state?
    ✅ C. terraform import
51. What command lists resources currently in state?
    ✅ B. terraform state list
52. What does the terraform refresh command do?
    ✅ D. Updates state to match real infrastructure
53. What does a state lock file do?
    ✅ C. Prevents multiple concurrent changes
54. Which file holds the dependency graph used by Terraform?
    ✅ A. .terraform directory

55. How does Terraform detect changes to resources?
    ✅ B. By comparing state with provider data
56. How can you recover a lost state file?
    ✅ C. Use remote backend state versions or backup
57. What file extension is used for backup of state?
    ✅ D. .tfstate.backup
58. What command can be used to rename resources in state?
    ✅ B. terraform state mv
59. What kind of backend supports encrypted state and team collaboration?
    ✅ A. remote
60. Which command should be used after modifying the backend configuration?
    ✅ A. terraform init

📦 4. Modules & Reusability (61–80)

61. What is a module in Terraform?
    ✅ A. A container for multiple resources used together
62. What is the source of a module when pulling from Terraform Registry?
    ✅ C. terraform-<PROVIDER>-<NAME>
63. Where are modules stored by default when pulled from the registry?
    ✅ B. In the .terraform/modules directory
64. What does the source argument in a module block specify?
    ✅ A. Path or URL to module code
65. How do you pass variables to a module?
    ✅ C. By using input variable arguments inside the module block
66. What is the purpose of module outputs?
    ✅ D. To expose values from a module to the root config
67. What is a child module?
    ✅ B. A module used inside another module
68. How do you reference a module output in the root module?
    ✅ A. module.<MODULE_NAME>.<OUTPUT_NAME>
69. What does the for_each meta-argument enable?
    ✅ D. Iterating over maps and sets
70. What is the benefit of modules?
    ✅ B. Reusability and consistency
71. What is the recommended structure of a custom module?
    ✅ A. main.tf, variables.tf, and outputs.tf
72. How do you avoid module version drift?
    ✅ C. Use version = constraint in the module source

73. Can a module call another module?
    ✅ B. Yes, it's called a nested module
74. How do you use local modules?
    ✅ C. Use a relative path in the source
75. What happens if you omit required module input variables?
    ✅ A. Terraform throws an error
76. Which file in a module defines output values?
    ✅ C. outputs.tf
77. Which of the following allows optional module inputs?
    ✅ B. Default values in variables.tf
78. How do you reuse logic without copy-paste?
    ✅ C. Use modules
79. Where should variables be defined in a reusable module?
    ✅ A. variables.tf
80. What is the best way to share a reusable module across teams?
    ✅ D. Publish to Terraform Registry or shared Git repository

🛠️ 5. Terraform CLI & Workflow (81–100)

81. What does `terraform fmt` do?
    ✅ C. Formats configuration files to HCL style
82. What does `terraform validate` do?
    ✅ A. Checks syntax and internal consistency
83. How do you preview changes before applying them?
    ✅ B. `terraform plan`
84. Which command saves a plan to a file?
    ✅ D. `terraform plan -out=planfile`
85. What command applies a saved plan file?
    ✅ A. `terraform apply planfile`
86. How do you apply changes without interactive approval?
    ✅ C. `terraform apply -auto-approve`
87. Which environment variable enables detailed logs?
    ✅ B. `TF_LOG`
88. How do you initialize a working directory for a new configuration?
    ✅ A. `terraform init`
89. What file locks the provider versions?
    ✅ D. `terraform.lock.hcl`
90. How do you upgrade provider plugins?
    ✅ B. `terraform init -upgrade`

91. Which command destroys managed infrastructure?
    ✅ C. `terraform destroy`
92. How do you get output values from the applied configuration?
    ✅ A. `terraform output`
93. How do you taint a resource to force its recreation?
    ✅ B. `terraform taint <resource>`
94. How do you untaint a resource?
    ✅ D. `terraform untaint <resource>`
95. What command removes unused provider plugins and modules?
    ✅ C. `terraform clean`
96. What command shows the current workspace?
    ✅ B. `terraform workspace show`
97. Which command lists available CLI commands?
    ✅ A. `terraform -help`
98. What does `terraform console` provide?
    ✅ D. An interactive shell for evaluating expressions
99. How can you supply values to variables?
    ✅ C. CLI flags, environment variables, or `.tfvars` file
100. What is the order of variable loading precedence?
    ✅ A. CLI > Environment vars > `terraform.tfvars` > Default values

🔐 6. Security, Provisioners & Functions (101–120)?

101. How do you mark an output as sensitive?
    ✅ B. `sensitive = true` in the output block
102. Why should you avoid committing the `terraform.tfstate` file to VCS?
    ✅ C. It may contain sensitive data
103. Which file extension is used to define variable values?
    ✅ A. `.tfvars`
104. How do you prevent a resource from being destroyed?
    ✅ D. Use `lifecycle { prevent_destroy = true }`
105. What is a provisioner used for in Terraform?
    ✅ B. To execute scripts after resource creation
106. When should you avoid using provisioners?
    ✅ C. When native provider functionality is available
107. What is the function of `null_resource`?
    ✅ A. Execute provisioners without creating infrastructure
108. What provisioner runs commands on remote instances?
    ✅ C. `remote-exec`

109. Which provisioner uploads files to a remote machine?
    ✅ B. `file`
110. How can you pass secure credentials to a provider?
    ✅ D. Use environment variables or secure backends
111. Which of these functions merges maps?
    ✅ A. `merge()`
112. What does the `coalesce()` function do?
    ✅ C. Returns the first non-null argument
113. Which function converts a list to a string?
    ✅ B. `join()`
114. What does `split()` do?
    ✅ D. Splits a string into a list
115. What is a for expression used for in Terraform?
    ✅ A. To transform a list or map
116. How do you define optional variables?
    ✅ C. With a default value
117. What does `terraform function lookup(map, key, default)` do?
    ✅ A. Returns a value from a map or a default if not found
118. How can you evaluate logic inline?
    ✅ B. Use a ternary operator `condition ? true_val : false_val`
119. What does the `length()` function return?
    ✅ C. Number of items in a list or map
120. How do you transform a map to a list of values?
    ✅ D. Use `values()` function

🧠 7. Workspaces, Terraform Cloud, & Misc (121–150)

121. What is a Terraform workspace used for?
    ✅ C. To manage multiple state files for the same configuration
122. What is the default workspace in Terraform?
    ✅ A. default
123. How do you create a new workspace?
    ✅ B. terraform workspace new <name>
124. What is a limitation of workspaces for environment management?
    ✅ C. They do not isolate configuration or variables
125. How do you switch between workspaces?
    ✅ D. terraform workspace select <name>
126. What is Terraform Cloud?
    ✅ B. A SaaS platform for running Terraform securely in the cloud

127. What is the purpose of Terraform Cloud workspaces?
✅ A. Isolate runs and states for configurations
128. What is VCS-driven run in Terraform Cloud?
✅ D. A Terraform run triggered by a commit to version control
129. What is a variable set in Terraform Cloud?
✅ C. A reusable group of variables shared across workspaces
130. How do you store sensitive variables in Terraform Cloud?
✅ A. Use the UI or CLI with sensitive = true
131. What is Sentinel in Terraform Cloud?
✅ D. A policy-as-code framework for governance
132. What does terraform login do?
✅ C. Authenticates CLI with Terraform Cloud
133. What is the cost estimation feature in Terraform Cloud?
✅ A. It predicts resource costs before apply
134. What backend type is used for Terraform Cloud?
✅ B. remote
135. How do you trigger a manual run in Terraform Cloud?
✅ C. Via the web UI or CLI
136. Can Terraform CLI be used with Terraform Cloud?
✅ A. Yes, with backend configured to remote
137. How do you restrict team access in Terraform Cloud?
✅ B. Use role-based access control
138. What feature helps ensure one team doesn't affect another's infrastructure?
✅ D. Workspace isolation
139. What file controls provider version constraints?
✅ C. required_providers in terraform {} block
140. What is the benefit of locking provider versions?
✅ B. Ensures consistent behavior across environments
141. What is the main use of the terraform console?
✅ C. Evaluate expressions and debug output
142. How do you debug Terraform errors?
✅ D. Use TF_LOG and terraform plan for context
143. What is the function of terraform graph?
✅ B. Visualizes resource dependencies
144. What causes a resource to be recreated during apply?
✅ A. Configuration drift or changes to immutable fields
145. What should you do before running terraform destroy in production?
✅ C. Review the execution plan carefully

146. How can you avoid applying accidental changes?

✅ D. Use terraform plan and peer reviews

147. What's the first step in troubleshooting a Terraform error?

✅ B. Review logs and the plan output

148. What happens if a remote backend is unavailable during apply?

✅ C. Terraform fails and does not proceed

149. What can be used to prevent unauthorized apply actions?

✅ A. Sentinel policies

150. What's the best way to share code and practices across teams?

✅ D. Create reusable modules and use version control

✅ Topic 1: Understand Infrastructure as Code (IaC) concepts

1. What is a primary benefit of Infrastructure as Code (IaC)?

A. Manual resource provisioning
B. Automated and consistent deployments
C. Infrastructure documentation only
D. Increased human interaction

✅ **Answer:** B

2. Which of the following best describes Terraform's programming model?

A. Imperative
B. Declarative
C. Procedural
D. Object-oriented

✅ **Answer:** B

3. What is the benefit of version-controlling infrastructure code?

A. Makes it harder to track changes
B. Enables team collaboration and auditing
C. Reduces code reuse
D. Improves runtime performance

✅ **Answer:** B

4. What is a key principle of immutable infrastructure?

A. Modify servers in place
B. Update resources manually
C. Replace infrastructure instead of changing it
D. Avoid using automation tools

✅ **Answer:** C

5. What does idempotency mean in IaC tools like Terraform?

A. The tool fails on repeated runs
B. Multiple runs produce the same result
C. The configuration must be unique every time
D. It creates different infrastructure each time

✅ **Answer:** B

6. Which of this best describes dependency management in Terraform?

A. Manual control of resource creation order
B. Ignoring relationships between resources

C. Automatic dependency graph resolution
D. Fixed ordering through bash scripts

✅ **Answer:** C

7. Which statement is true about declarative configuration?

A. It describes steps to reach a goal
B. It defines the desired end state
C. It uses shell scripts
D. It requires more manual effort

✅ **Answer:** B

8. Which tool focuses more on procedural (imperative) infrastructure automation?

A. Terraform
B. CloudFormation
C. Ansible
D. Packer

✅ **Answer:** C

9. Which of the following is NOT a benefit of IaC?

A. Repeatability
B. Inconsistent environments
C. Automation
D. Versioning

✅ **Answer:** B

10. How does Terraform ensure consistency across environments?

A. Uses randomized configuration
B. Ignores configuration changes
C. Applies the same code to multiple environments
D. Manually edits the resources

✅ **Answer:** C

11. Which of these is a real-world use case for Infrastructure as Code?

A. Manual server patching
B. Email communication automation
C. Rebuilding environments in staging and production
D. Writing application source code

✅ **Answer:** C

12. What happens when an IaC tool like Terraform is run multiple times with no changes in configuration?

A. It errors out
B. It destroys infrastructure
C. It makes unnecessary updates
D. It makes no changes

✅ **Answer:** D

13. Why might a team prefer declarative over imperative IaC?

A. Declarative is more error-prone
B. It allows describing the desired end state
C. It is harder to use
D. It runs shell commands directly

✅ **Answer:** B

14. What advantage does immutable infrastructure provide during deployment?

A. Faster patching
B. Reduced testing effort
C. Predictable and stable environments
D. More manual intervention

✅ **Answer:** C

15. Which of the following is most likely to be version-controlled in an IaC environment?

A. Production servers
B. Virtual machines
C. Terraform configuration files
D. Cloud provider credentials

✅ **Answer:** C

16. Why is idempotency important in IaC tools?

A. Prevents resource leaks
B. Ensures consistent outcomes across multiple runs
C. Avoids state management
D. Allows for manual changes

✅ **Answer:** B

17. Which of these is a Terraform advantage over CloudFormation?

A. Native AWS integration
B. Support for only JSON configuration
C. Multi-cloud support

D. More complex language

✅ **Answer:** C

18. What best defines Infrastructure as Code (IaC)?

A. Infrastructure managed using CLI commands only
B. Writing manual documentation for infrastructure
C. Managing infrastructure using machine-readable definition files
D. Using GUIs to manage resources

✅ **Answer:** C

19. Which of the following tools is not primarily focused on IaC?

A. Terraform
B. Ansible
C. Jenkins
D. Pulumi

✅ **Answer:** C

20. Which concept refers to deploying fresh infrastructure instead of modifying existing ones?

A. Mutable infrastructure
B. Imperative programming
C. Immutable infrastructure
D. Drift correction

✅ **Answer:** C

21. What challenge does IaC solve in team environments?

A. Manual configuration drift
B. Infrastructure silos
C. Lack of visibility
D. All of the above

✅ **Answer:** D

22. What is the output of terraform plan if no changes are detected?

A. Error message
B. Infrastructure destroyed
C. No changes
D. Skips validation

✅ **Answer:** C

23. Which of these best reflects idempotent behavior?

A. Code fails after the first run
B. Same input yields same infrastructure every run
C. Code randomizes resource names
D. Configuration changes every time

✅ **Answer:** B

24. What does Terraform use to determine resource relationships?

A. Bash scripting
B. Indentation levels
C. Dependency graph
D. Manual sequencing

✅ **Answer:** C

25. What is the role of automation in IaC?

A. Increase manual oversight
B. Remove the need for configuration
C. Accelerate and standardize deployments
D. Avoid using scripts

✅ **Answer:** C

⚙️ Topic 2: Understand Terraform's purpose (vs. other IaC tools)

1. What is Terraform primarily used for?

A. Running application code
B. Managing infrastructure as code
C. Monitoring applications
D. Creating CI/CD pipelines

✅ **Answer:** B

---

2. What makes Terraform provider-agnostic?

A. It supports only AWS
B. It uses HCL for syntax
C. It can work with any cloud or service that has a provider plugin
D. It uses shell scripts

✅ **Answer:** C

---

3. Which language does Terraform use to define infrastructure?

A. YAML
B. JSON
C. HCL (HashiCorp Configuration Language)
D. XML
✅ **Answer:** C

---

4. Which of the following is a key feature of Terraform?

A. Push-based model
B. Declarative configuration
C. Imperative scripting
D. Stateful firewall
✅ **Answer:** B

---

5. Which tool is closest in functionality to Terraform but limited to AWS?

A. Ansible
B. Chef
C. CloudFormation
D. Jenkins
✅ **Answer:** C

---

6. What is one use case where Terraform excels?

A. Building application code
B. Continuous integration
C. Provisioning cloud infrastructure
D. Managing container runtime
✅ **Answer:** C

---

7. Compared to Ansible, Terraform is more focused on:

A. Configuration management
B. Application deployment
C. Infrastructure provisioning
D. Log analysis
✅ **Answer:** C

---

8. Which of the following tools uses an imperative style?

A. Terraform
B. Pulumi
C. Ansible
D. CloudFormation
✅ **Answer:** C

---

9. What does Terraform's provider plugin architecture enable?

A. Only AWS and Azure support
B. Dynamic runtime scripting
C. Integration with multiple cloud and SaaS platforms
D. Local-only infrastructure support
✅ **Answer:** C

---

10. Which tool uses Python or JavaScript for defining infrastructure?

A. Terraform
B. CloudFormation
C. Pulumi
D. Chef
✅ **Answer:** C

---

11. Why is Terraform preferred in multi-cloud scenarios?

A. It is a paid-only tool
B. It only supports one cloud
C. It supports multiple providers with a common syntax
D. It requires cloud-specific scripts
✅ **Answer:** C

---

12. Which of the following can Terraform do?

A. Run unit tests
B. Configure Kubernetes resources
C. Send application logs

D. Compile source code

✅ **Answer:** B

---

13. Which of the following is NOT a core use case of Terraform?

A. Deploying web applications
B. Infrastructure provisioning
C. Resource lifecycle management
D. Infrastructure compliance

✅ **Answer:** A

---

14. Terraform is best described as a:

A. Configuration management tool
B. Infrastructure automation tool
C. Log aggregation tool
D. Continuous delivery platform

✅ **Answer:** B

---

15. What makes Terraform unique compared to Chef or Puppet?

A. Uses declarative JSON only
B. Has no versioning capability
C. Builds and manages infrastructure statefully
D. Designed for application configuration

✅ **Answer:** C

---

16. Which tool would you use for provisioning VMs across AWS, Azure, and GCP with a single codebase?

A. Ansible
B. Terraform
C. Bash scripts
D. CloudFormation

✅ **Answer:** B

---

17. What is a key benefit of HCL over JSON in Terraform?

A. More verbose
B. Less human-readable
C. Easier to write and maintain
D. Only works with AWS
✅ **Answer:** C

---

18. What makes Terraform declarative?

A. It runs shell scripts
B. It defines how to reach a goal
C. It specifies the end state of infrastructure
D. It generates deployment logs
✅ **Answer:** C

---

19. What does it mean when we say Terraform is "open-source"?

A. Only the UI is open
B. It is licensed and not customizable
C. Its source code is publicly available and community-supported
D. It cannot be used in production
✅ **Answer:** C

---

20. Terraform's architecture includes:

A. Cookbooks and recipes
B. Agents and nodes
C. Providers and resources
D. Pipelines and containers
✅ **Answer:** C

---

21. How is Terraform different from Ansible?

A. Terraform uses Python
B. Ansible is used for infrastructure provisioning
C. Terraform provisions infrastructure; Ansible configures it
D. Both are used to monitor applications
✅ **Answer:** C

---

22. Which is NOT a benefit of using Terraform in a DevOps pipeline?

A. Repeatable infrastructure builds
B. Infrastructure versioning
C. Manual provisioning
D. Integration with CI/CD tools
✅ **Answer:** C

---

23. What enables Terraform to be cloud-agnostic?

A. It stores state locally
B. It has a REST API
C. Its provider model abstracts platform details
D. It is built into all cloud platforms
✅ **Answer:** C

---

24. Which Terraform feature allows automation in CI/CD workflows?

A. Bash scripts
B. Agent installation
C. CLI commands and modules
D. Manual provisioning
✅ **Answer:** C

---

25. When compared to Pulumi, Terraform differs mainly in:

A. Provider support
B. Programming language used
C. State management
D. Cloud capabilities
✅ **Answer:** B

🧠 Topic 3: Understand Terraform Basics

1. Which command initializes a working directory with Terraform configuration files?

A. terraform plan
B. terraform init
C. terraform apply
D. terraform refresh
✅ **Answer:** B

---

2. What is the purpose of terraform plan?

A. To destroy infrastructure
B. To initialize Terraform
C. To display the changes Terraform will make
D. To apply the configuration immediately
✅ **Answer:** C

---

3. What file contains the current state of infrastructure managed by Terraform?

A. main.tf
B. terraform.tfvars
C. terraform.tfstate
D. variables.tf
✅ **Answer:** C

---

4. What is a Terraform provider?

A. A managed cloud account
B. A plugin that interacts with APIs of supported platforms
C. A state file
D. A module for grouping resources
✅ **Answer:** B

---

5. Which block defines a resource in Terraform?

A. provider
B. module

C. resource
D. variable
✅ **Answer:** C

---

6. What does a module in Terraform do?

A. Stores the Terraform binary
B. Organizes and reuses configuration code
C. Defines provider-specific options
D. Generates JSON plans
✅ **Answer:** B

---

7. What file type typically contains variable definitions in Terraform?

A. .tfstate
B. .tfvars
C. .json
D. .log
✅ **Answer:** B

---

8. Which of the following can be used to reference an output value from a module?

A. module.output
B. module.<module_name>.output_name
C. output.<module_name>.value
D. var.output
✅ **Answer:** B

---

9. What does the terraform destroy command do?

A. Deletes the state file
B. Deletes Terraform installation
C. Destroys all resources in the state file
D. Removes the .terraform directory only
✅ **Answer:** C

---

10. What is the purpose of a backend in Terraform?

A. To define resource types
B. To provision compute infrastructure
C. To define where and how state is stored
D. To log CLI output
✅ **Answer:** C

---

11. What is create_before_destroy used for in Terraform lifecycle rules?

A. Forces deletion before replacement
B. Creates a new resource before deleting the old one
C. Ignores changes
D. Prevents all destruction
✅ **Answer:** B

---

12. What does count meta-argument do?

A. Groups resources
B. Loops through a resource list
C. Determines the number of instances to create
D. Limits logging
✅ **Answer:** C

---

13. What file holds provider version constraints?

A. terraform.lock.hcl
B. terraform.tfstate
C. provider.tf
D. main.tf
✅ **Answer:** A

---

14. Which Terraform CLI command shows the current state in human-readable form?

A. terraform show
B. terraform fmt
C. terraform output

D. terraform describe

✅ **Answer:** A

---

15. What is the purpose of terraform apply?

A. Validate Terraform code
B. Execute the planned infrastructure changes
C. Format code
D. Import existing resources

✅ **Answer:** B

---

16. What does depends_on do in a Terraform resource block?

A. Explicitly sets a dependency between resources
B. Skips validation
C. Loads output from other resources
D. Forces apply without confirmation

✅ **Answer:** A

---

17. Which file holds the actual infrastructure definitions?

A. terraform.lock.hcl
B. .tfstate.backup
C. .tf files like main.tf or network.tf
D. terraform.tfvars.json

✅ **Answer:** C

---

18. What are output values used for?

A. Replacing resources
B. Providing configuration to other modules or users
C. Formatting Terraform code
D. Logging runtime errors

✅ **Answer:** B

---

19. Which command shows a dependency graph in DOT format?

A. terraform dot
B. terraform show
C. terraform graph
D. terraform plan -dot
✅ **Answer:** C

---

20. What is the role of terraform.tfstate.backup?

A. Keeps a backup of the most recent successful state
B. Stores encrypted credentials
C. Is a remote state file
D. Manages module versions
✅ **Answer:** A

---

21. What does for_each allow that count does not?

A. Supports resource creation
B. Iterates over maps and sets with keys
C. Loops through one value only
D. Only works with AWS resources
✅ **Answer:** B

---

22. Which is NOT a valid backend for Terraform state?

A. Local
B. S3
C. MySQL
D. Terraform Cloud
✅ **Answer:** C

---

23. What happens when you run terraform init in a directory with a new provider?

A. Fails until provider is manually downloaded
B. Installs the provider plugin
C. Runs the configuration
D. Outputs a plan
✅ **Answer:** B

---

24. Which Terraform component connects your configuration to a specific cloud service?

A. State
B. Resource
C. Provider
D. Module
✅ **Answer:** C

---

25. Which of the following is NOT a valid use of a module?

A. Reduce duplication
B. Organize code
C. Store logs
D. Reuse components
✅ **Answer:** C

🔐 Topic 4: Use Terraform CLI (outside of core workflow)

1. What does terraform fmt do?

A. Destroys resources
B. Formats Terraform configuration files to canonical style
C. Validates Terraform plans
D. Applies configuration changes
✅ **Answer:** B

---

2. What is the purpose of terraform validate?

A. Check if provider plugins are up to date
B. Check syntax and internal consistency of configuration files
C. Verify cloud provider credentials
D. Display resource dependencies
✅ **Answer:** B

---

3. What does the terraform taint command do?

A. Permanently deletes a resource
B. Prevents a resource from being created

C. Marks a resource for recreation on the next apply
D. Creates a resource forcibly
✅ **Answer:** C

---

4. What is terraform console used for?

A. Creating new resources
B. Exploring state values and expressions interactively
C. Formatting Terraform code
D. Viewing provider documentation
✅ **Answer:** B

---

5. How do you remove a resource from the state file without destroying it?

A. terraform destroy
B. terraform refresh
C. terraform state rm
D. terraform plan -destroy
✅ **Answer:** C

---

6. Which command imports existing infrastructure into Terraform state?

A. terraform fetch
B. terraform import
C. terraform get
D. terraform refresh
✅ **Answer:** B

---

7. What does terraform state list do?

A. Lists the files in the state directory
B. Lists all resources tracked in the current state
C. Lists resources to be destroyed
D. Lists all output variables
✅ **Answer:** B

---

8. What is the purpose of terraform output?

A. Formats Terraform files
B. Shows a list of providers
C. Displays values defined in the output blocks
D. Displays errors in configuration
✅ **Answer:** C

---

9. Which command would you use to move a resource from one module to another in the state file?

A. terraform mv
B. terraform rename
C. terraform shift
D. terraform copy
✅ **Answer:** A

---

10. Which Terraform CLI subcommand lists all providers used in the configuration?

A. terraform show
B. terraform validate
C. terraform providers
D. terraform list
✅ **Answer:** C

---

11. What does terraform untaint do?

A. Destroys the resource
B. Removes the taint from a resource
C. Marks all resources for destruction
D. Resets the provider version
✅ **Answer:** B

---

12. How can you specify input variable values via environment variables?

A. TF_INPUT=1
B. Prefixing the variable name with TF_VAR_
C. Using terraform apply -env
D. Using .terraformrc file
✅ **Answer:** B

---

13. What environment variable enables detailed CLI debugging?

A. DEBUG_TF
B. TF_LOG
C. TF_DEBUG_MODE
D. VERBOSE_TERRAFORM
✅ **Answer:** B

---

14. What is the use of terraform workspace?

A. Manages provider plugins
B. Manages multiple environments from a single configuration
C. Imports state
D. Creates Terraform modules
✅ **Answer:** B

---

15. How do you apply a saved plan file?

A. terraform plan
B. terraform execute planfile
C. terraform apply planfile
D. terraform commit
✅ **Answer:** C

---

16. What is the command to list all existing Terraform workspaces?

A. terraform workspaces
B. terraform workspace list
C. terraform list workspaces
D. terraform list
✅ **Answer:** B

---

17. Which command formats Terraform configuration files?

A. terraform format
B. terraform clean
C. terraform fmt

D. terraform tidy

✅ **Answer:** C

---

18. What does terraform state show <resource> display?

A. JSON plan output
B. The actual state data for a specific resource
C. The changes to be made
D. Output variable values

✅ **Answer:** B

---

19. Which of the following is NOT a valid TF_LOG level?

A. TRACE
B. INFO
C. NOTICE
D. ERROR

✅ **Answer:** C

---

20. Which terraform command is used to see current values stored in outputs after apply?

A. terraform print
B. terraform outputs
C. terraform output
D. terraform log

✅ **Answer:** C

---

21. What command creates a binary plan output file?

A. terraform apply -json
B. terraform plan -out=planfile
C. terraform output -plan
D. terraform run -o planfile

✅ **Answer:** B

---

22. After creating a plan file, which command do you use to execute it?

A. terraform run planfile
B. terraform build planfile
C. terraform apply planfile
D. terraform commit

✅ **Answer:** C

---

23. What does the terraform workspace new <name> command do?

A. Clones the existing workspace
B. Creates a new working directory
C. Creates a new isolated state environment
D. Deletes the default workspace

✅ **Answer:** C

---

24. What is a benefit of using workspaces in Terraform?

A. Running multiple Terraform versions
B. Organizing credentials
C. Managing multiple state files from the same config
D. Defining modules

✅ **Answer:** C

---

25. What happens if terraform apply is run without a planfile?

A. It fails immediately
B. It shows the plan and asks for confirmation
C. It destroys infrastructure
D. It runs the format check

✅ **Answer:** B

📁 Topic 5: Interact with Terraform Modules

1. What is a Terraform module?

A. A set of output variables
B. A collection of .tfstate files
C. A container for multiple resources used together
D. A CLI plugin
✅ **Answer:** C

---

2. Which files typically make up a custom Terraform module?

A. init.tf, deploy.tf, teardown.tf
B. provider.tf, state.tf, backup.tf
C. main.tf, variables.tf, outputs.tf
D. plan.tf, logs.tf, provider.tf
✅ **Answer:** C

---

3. What block is used to call a child module in Terraform?

A. use
B. module
C. resource
D. import
✅ **Answer:** B

---

4. How do you pass a variable into a child module?

A. Using data block
B. Using output block
C. By setting attributes inside the module block
D. Via environment variables only
✅ **Answer:** C

---

5. Where can you find public Terraform modules?

A. AWS Marketplace
B. Terraform Registry
C. GitHub Marketplace

D. Docker Hub

✅ **Answer:** B

---

6. What is the correct syntax to source a module from Terraform Registry?

A. source = "terraform.com/module/path"
B. source = "registry.terraform.io/module/name"
C. source = "docker.io/module"
D. source = "main.tf"

✅ **Answer:** B

---

7. How can you use a module stored on GitHub?

A. source = "github.com/user/repo"
B. source = "repo.git"
C. module = "github"
D. import = "github"

✅ **Answer:** A

---

8. What happens if you do not pin a module version in Terraform?

A. It uses the latest available version
B. It fails to run
C. It picks a random version
D. It uses version 1.0.0

✅ **Answer:** A

---

9. Why is module version pinning important?

A. To speed up Terraform CLI
B. To avoid provider authentication
C. To ensure consistent and predictable deployments
D. To allow state sharing

✅ **Answer:** C

---

10. Which command downloads modules referenced in the configuration?

A. terraform get
B. terraform fetch
C. terraform init
D. terraform import

✅ **Answer:** C

---

11. What does the outputs.tf file typically contain?

A. Local variables
B. Input variables
C. Output definitions to expose data from the module
D. Remote backend configuration

✅ **Answer:** C

---

12. What is the purpose of variables.tf in a module?

A. Define hardcoded output values
B. Import variables from other modules
C. Declare input variables for parameterization
D. Log state transitions

✅ **Answer:** C

---

13. Which of the following is a benefit of using modules in Terraform?

A. Larger configuration files
B. Reduced reusability
C. Simplified organization and reuse of infrastructure code
D. Slower plan and apply time

✅ **Answer:** C

---

14. What is a "root module"?

A. A module that only contains outputs
B. A module used inside other modules
C. The directory containing the Terraform configuration being applied
D. The main GitHub module

✅ **Answer:** C

---

15. Which of the following is a correct way to access an output from a child module?

A. output.module_name.value
B. module.module_name.output_name
C. child.output.value
D. terraform_output.module_name

✅ **Answer:** B

---

16. Can a module call another module?

A. No, only root modules can call modules
B. Yes, modules can be nested
C. Only Terraform Cloud supports this
D. Only in Enterprise version

✅ **Answer:** B

---

17. What happens if you change a module source without changing its version?

A. Terraform ignores it
B. You must re-run terraform get -update or init -upgrade
C. It causes immediate error
D. Terraform automatically removes resources

✅ **Answer:** B

---

18. Which module source path is used for local modules?

A. source = "local:/"
B. source = "./module_path"
C. source = "terraform.local"
D. source = "file://module"

✅ **Answer:** B

---

19. How do you ensure a module uses a specific provider configuration?

A. You cannot specify providers in modules
B. Use provider block within the module
C. Use providers argument in the module block

D. Export provider from root module

✅ **Answer:** C

---

20. Which Terraform file lists downloaded module versions and hashes?

A. main.tf
B. terraform.tfstate
C. terraform.lock.hcl
D. module.tf

✅ **Answer:** C

---

21. Which command can refresh or upgrade module versions?

A. terraform get -upgrade
B. terraform apply -module-update
C. terraform upgrade modules
D. terraform refresh-modules

✅ **Answer:** A

---

22. How can outputs from one module be used in another module?

A. By importing the state
B. Via output.tfvars
C. By passing them through root module variables
D. By including outputs in the main.tf directly

✅ **Answer:** C

---

23. How do you override default variable values in modules?

A. Redefine them in main.tf
B. Use a backend override file
C. Pass them explicitly in the module block
D. Use terraform env

✅ **Answer:** C

---

24. Can you call the same module multiple times with different configurations?

A. No
B. Only in Terraform Cloud
C. Yes, using different module blocks with different names
D. Only for public modules
✅ **Answer:** C

---

25. What is the correct method to call a remote module from Terraform Registry?

```
module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "3.5.0"
}
```

What does the source refer to?
A. Provider configuration
B. Output path
C. Terraform Registry module path
D. GitHub link
✅ **Answer:** C

🗂️ Topic 6: Navigate Terraform Workflow

1. What is the correct order of operations in a typical Terraform workflow?

A. apply → init → plan
B. init → plan → apply
C. plan → apply → init
D. init → apply → plan
✅ **Answer:** B

---

2. What does terraform init do?

A. Applies configuration
B. Initializes Terraform configuration and downloads providers/modules
C. Removes the state file
D. Formats Terraform files
✅ **Answer:** B

---

3. What is the purpose of terraform plan?

A. Validates backend configuration
B. Applies infrastructure changes
C. Displays execution plan without making changes
D. Shows all variables
✅ **Answer:** C

---

4. How can you apply a Terraform plan without user confirmation?

A. terraform plan -y
B. terraform run -auto
C. terraform apply -auto-approve
D. terraform execute
✅ **Answer:** C

---

5. Which command permanently deletes all infrastructure managed by Terraform?

A. terraform destroy
B. terraform rm
C. terraform clean
D. terraform format
✅ **Answer:** A

---

6. What is the purpose of .terraform.lock.hcl?

A. Stores logs
B. Defines module source
C. Locks provider versions for consistency
D. Controls resource creation order
✅ **Answer:** C

---

7. What does terraform init -upgrade do?

A. Upgrades Terraform CLI version
B. Upgrades provider versions and re-initializes modules
C. Re-applies configuration
D. Upgrades module versions only
✅ **Answer:** B

---

8. What file controls the minimum required provider versions?

A. terraform.lock.hcl
B. main.tf via required_providers
C. .tfstate
D. backend.tf
✅ **Answer:** B

---

9. What happens if you run terraform apply without running terraform init first?

A. It creates resources
B. It upgrades providers
C. It fails with an error
D. It skips the plan
✅ **Answer:** C

---

10. Which file is generated when initializing Terraform for the first time?

A. terraform.tfstate
B. .terraform.lock.hcl
C. .terraformignore
D. backend.tf
✅ **Answer:** B

---

11. What is the .terraform directory used for?

A. Holding environment variables
B. Storing provider binaries and module caches
C. Storing encrypted state
D. Logging Terraform output
✅ **Answer:** B

---

12. What happens during terraform apply?

A. It shows a plan and requires confirmation before proceeding
B. It always runs without confirmation
C. It destroys all infrastructure

D. It initializes the backend

✅ **Answer:** A

---

13. What is the effect of using -auto-approve with terraform destroy?

A. Skips deletion
B. Requires confirmation
C. Deletes infrastructure without confirmation
D. Destroys only variables

✅ **Answer:** C

---

14. Where are provider constraints typically declared?

A. In .terraform directory
B. In the terraform.tfstate file
C. Inside required_providers block in configuration
D. In the shell profile

✅ **Answer:** C

---

15. What is the result of running terraform plan after making changes in .tf files?

A. Shows a summary of new, changed, or deleted resources
B. Deletes all resources
C. Resets variable values
D. Refreshes the backend

✅ **Answer:** A

---

16. What command displays changes between the current state and planned execution?

A. terraform fmt
B. terraform diff
C. terraform plan
D. terraform version

✅ **Answer:** C

---

17. Which command should be run after changing the provider version in the configuration?

A. terraform refresh
B. terraform update
C. terraform init -upgrade
D. terraform reapply
✅ **Answer:** C

---

18. Which file stores provider version constraints after init?

A. terraform.tfvars
B. .terraform.lock.hcl
C. provider.tf
D. versions.tf
✅ **Answer:** B

---

19. What happens if you change the provider block and run terraform init again?

A. Terraform downloads the new provider
B. It deletes the current state
C. It destroys all infrastructure
D. Nothing happens
✅ **Answer:** A

---

20. What is required to avoid manual approval for every apply operation?

A. Setting TF_AUTO_MODE=1
B. Using terraform validate
C. Adding -auto-approve to the command
D. Editing .terraform.tfstate
✅ **Answer:** C

---

21. Which command is responsible for creating infrastructure?

A. terraform init
B. terraform validate
C. terraform plan
D. terraform apply
✅ **Answer:** D

---

22. What happens if .terraform.lock.hcl is deleted?

A. Terraform fails permanently
B. Providers are reinstalled during next init
C. Modules are destroyed
D. Apply is blocked
✅ **Answer:** B

---

23. What does a plan file generated by terraform plan -out=planfile contain?

A. JSON version of the configuration
B. Actual logs
C. Binary file with the exact execution plan
D. Provider plugin data
✅ **Answer:** C

---

24. Why might you want to generate and apply a saved plan file?

A. To update the .tf files
B. To change the backend
C. For review and approval before execution in CI/CD
D. To skip validation
✅ **Answer:** C

---

25. What must you do after cloning a new Terraform configuration repository?

A. Run terraform apply
B. Run terraform destroy
C. Run terraform init
D. Run terraform output
✅ **Answer:** C

🗂️ Topic 7: Implement and Maintain State

1. What is the primary purpose of the Terraform state file (terraform.tfstate)?

A. To store variable definitions
B. To store remote backend configuration
C. To map real-world infrastructure to configuration
D. To define provider versions
✅ **Answer:** C

---

2. What format is the Terraform state file stored in?

A. YAML
B. Binary
C. JSON
D. XML
✅ **Answer:** C

---

3. Which command shows the resources tracked in the state file?

A. terraform plan
B. terraform state list
C. terraform output
D. terraform show
✅ **Answer:** B

---

4. What is state drift?

A. When backend changes without your knowledge
B. When resources are deleted from code
C. When real infrastructure differs from Terraform state
D. When Terraform can’t connect to the provider
✅ **Answer:** C

---

5. Which command updates state by reconciling it with real infrastructure?

A. terraform refresh
B. terraform plan
C. terraform state show

D. terraform validate

✅ **Answer:** A

---

6. How do you move a resource in the state file from one name to another?

A. terraform import
B. terraform mv
C. terraform rename
D. terraform copy

✅ **Answer:** B

---

7. What file is automatically created as a backup of state?

A. terraform.tfbackup.json
B. .terraformstate.old
C. terraform.tfstate.backup
D. backup.tfstate.json

✅ **Answer:** C

---

8. What is a benefit of remote state?

A. Slower deployments
B. Local-only visibility
C. Team collaboration and state locking
D. No provider support

✅ **Answer:** C

---

9. What does the S3 backend use for state locking?

A. SNS
B. CloudWatch
C. DynamoDB
D. EC2

✅ **Answer:** C

---

10. Which command removes a resource from the Terraform state file without destroying it?

A. terraform destroy
B. terraform state rm
C. terraform remove
D. terraform undeploy

✅ **Answer:** B

---

11. Which command displays detailed information about a resource in state?

A. terraform describe
B. terraform info
C. terraform state show
D. terraform list

✅ **Answer:** C

---

12. Why should you avoid committing the state file to version control?

A. It breaks the configuration
B. It exposes provider binaries
C. It may contain sensitive data
D. It's too large

✅ **Answer:** C

---

13. What happens if two team members try to apply changes to the same state at the same time without locking?

A. Only one can run apply
B. Terraform errors out
C. State file corruption or drift can occur
D. Terraform sends alerts

✅ **Answer:** C

---

14. What does a backend define in Terraform?

A. Resource location
B. Input variables
C. How and where Terraform stores state

D. Secret management

✅ **Answer:** C

---

15. Where is the backend configuration typically defined?

A. In .terraform.lock.hcl
B. In the terraform.tfstate
C. In a backend block inside a terraform block
D. In variables.tf

✅ **Answer:** C

---

16. What is the effect of changing the backend type (e.g., local to remote)?

A. Terraform automatically deletes old state
B. You must manually migrate state using terraform init with migration
C. You lose access to outputs
D. State file becomes encrypted

✅ **Answer:** B

---

17. What is the default backend used by Terraform if none is specified?

A. AWS S3
B. Azure Blob
C. Local backend
D. Terraform Cloud

✅ **Answer:** C

---

18. What feature does Terraform Cloud provide in terms of state?

A. Local-only state storage
B. No encryption
C. Automatic state versioning and locking
D. Direct AWS S3 integration

✅ **Answer:** C

---

19. Which of the following is a best practice when using remote backends?

A. Enable state locking
B. Disable access logging
C. Use public buckets
D. Store backend credentials in .tfvars
✅ **Answer:** A

---

20. Which of the following can lead to state drift?

A. Terraform destroy
B. terraform validate
C. Manually deleting infrastructure in the cloud console
D. Running terraform init
✅ **Answer:** C

---

21. What command initializes a backend for remote state?

A. terraform init
B. terraform deploy
C. terraform remote
D. terraform refresh
✅ **Answer:** A

---

22. What can happen if you share a local state file across a team?

A. Increased performance
B. Enhanced logging
C. Risk of state conflicts or corruption
D. No issues
✅ **Answer:** C

---

23. What should be done before manually editing a .tfstate file?

A. Backup the file
B. Remove the backend
C. Destroy all resources
D. Run terraform destroy
✅ **Answer:** A

---

24. What command can be used to list all addresses stored in the state?

A. terraform list
B. terraform state list
C. terraform inventory
D. terraform ls
✅ **Answer:** B

---

25. Where can Terraform store remote state securely with versioning and access control?

A. In main.tf
B. On your local machine
C. In AWS S3 with encryption and IAM
D. In GitHub
✅ **Answer:** C

📁 Topic 8: Read, Generate, and Modify Configuration

1. What language does Terraform use for its configuration files?

A. JSON
B. YAML
C. HCL (HashiCorp Configuration Language)
D. XML
✅ **Answer:** C

---

2. Which file typically holds variable definitions?

A. terraform.tfstate
B. variables.tf
C. main.tf
D. output.tf
✅ **Answer:** B

---

3. What is the correct way to define a variable with a default value?

A. variable = "region" { default = "us-west-1" }
B. variable "region" { default = "us-west-1" }
C. set variable "region" = "us-west-1"

D. input "region" { value = "us-west-1" }

✅ **Answer:** B

---

4. Which block is used to fetch information from external sources like AWS AMIs or IPs?

A. resource
B. module
C. data
D. variable

✅ **Answer:** C

---

5. What does the output block do in a Terraform configuration?

A. Writes to a log file
B. Passes variables to backend
C. Exposes values after apply
D. Imports resources

✅ **Answer:** C

---

6. Which file contains values passed to variables?

A. terraform.tfstate
B. main.tf
C. outputs.tf
D. .tfvars

✅ **Answer:** D

---

7. Which of the following functions merges two maps in Terraform?

A. join()
B. concat()
C. merge()
D. combine()

✅ **Answer:** C

---

8. Which Terraform function splits a string into a list?

A. join()
B. split()
C. explode()
D. list()
✅ **Answer:** B

---

9. What is the correct way to declare a local value in Terraform?

A. set local "x" = "value"
B. locals { x = "value" }
C. local x = "value"
D. define local "x" { value = "value" }
✅ **Answer:** B

---

10. What does count allow you to do in a resource block?

A. Create sequential names
B. Limit instances of variables
C. Create multiple resource instances dynamically
D. Run apply only once
✅ **Answer:** C

---

11. What is the difference between count and for_each in Terraform?

A. count is for strings only; for_each is for integers
B. count uses a list or integer; for_each uses a map or set
C. for_each is deprecated
D. Both are used only in modules
✅ **Answer:** B

---

12. What does a dynamic block do in Terraform?

A. Changes backend at runtime
B. Creates conditional resource types
C. Generates nested blocks programmatically
D. Delays execution
✅ **Answer:** C

---

13. Which expression type is used to filter or transform data in Terraform?

A. Static expressions
B. Regular blocks
C. For expressions
D. Variable blocks
✅ **Answer:** C

---

14. What does the terraform plan command do with configuration files?

A. Deletes them
B. Parses and formats them
C. Shows changes required to match the desired state
D. Imports external modules
✅ **Answer:** C

---

15. Which block is used to declare an AWS EC2 resource in Terraform?

A. module "aws_instance"
B. resource "aws_instance"
C. provider "aws_instance"
D. data "aws_instance"
✅ **Answer:** B

---

16. What happens if two .tf files define the same resource name in the same module?

A. Terraform merges them
B. One is ignored
C. A conflict/error occurs
D. They are both applied independently
✅ **Answer:** C

---

17. How are .tf files loaded by Terraform?

A. By alphabetical order
B. Randomly
C. All .tf files are loaded regardless of filename, in no guaranteed order

D. Only main.tf is used

✅ **Answer:** C

---

18. What is the correct way to reference a resource attribute in another resource?

A. aws_instance.id
B. aws_instance["myvm"]["id"]
C. aws_instance.myvm.id
D. ${aws_instance.id}

✅ **Answer:** C

---

19. Which keyword is used to make an output value sensitive?

A. hidden = true
B. encrypt = true
C. sensitive = true
D. secure = true

✅ **Answer:** C

---

20. What happens if you use terraform console?

A. A UI is launched
B. Configuration is validated
C. A REPL is opened to evaluate expressions
D. Plan file is created

✅ **Answer:** C

---

21. Which Terraform function checks for the presence of a key in a map?

A. lookup()
B. find()
C. exists()
D. has_key()

✅ **Answer:** D

---

22. What does the terraform validate command check?

A. Syntax and internal consistency of the configuration
B. If provider credentials are valid
C. Whether the cloud infrastructure is reachable
D. Backend configuration
✅ **Answer:** A

---

23. Which of the following is NOT a valid file extension for Terraform configuration?

A. .tf
B. .tfvars
C. .terraform
D. .tf.json
✅ **Answer:** C

---

24. Which of the following allows the use of conditional expressions?

A. if else
B. condition {}
C. expression = var.enabled ? "yes" : "no"
D. optional {}
✅ **Answer:** C

---

25. Which data type allows complex nested structures?

A. String
B. List
C. Map
D. Object
✅ **Answer:** D

☁️ Topic 9: Understand Terraform Cloud and Enterprise Capabilities

---

1. What is the purpose of Terraform Cloud?

A. Storing sensitive data only
B. Hosting websites

C. Managing Terraform runs and state remotely
D. Creating virtual machines directly
✅ **Answer:** C

---

2. Which feature is unique to Terraform Cloud and not present in open-source Terraform?

A. terraform apply
B. CLI usage
C. Remote execution and state management
D. HCL syntax support
✅ **Answer:** C

---

3. What is a Terraform Cloud workspace?

A. A text editor
B. A provider block
C. A place to organize and execute Terraform configurations
D. An S3 bucket
✅ **Answer:** C

---

4. What allows Terraform Cloud to trigger runs from version control events?

A. API Gateway
B. CLI integration
C. VCS integration
D. Webhooks only
✅ **Answer:** C

---

5. Which file locks the provider versions used in Terraform?

A. terraform.tfstate
B. .terraform.lock.hcl
C. versions.tf
D. providers.tf
✅ **Answer:** B

---

6. What is the difference between CLI and remote backends in Terraform Cloud?

A. Remote backend supports interactive inputs
B. CLI backend allows VCS triggers
C. Remote backend executes Terraform runs in Terraform Cloud
D. CLI backend allows team collaboration

✅ **Answer:** C

---

7. Which capability is available only in Terraform Enterprise?

A. Cost estimation
B. Sentinel policies
C. State locking
D. Remote backends

✅ **Answer:** B

---

8. What is the purpose of Sentinel in Terraform Enterprise?

A. Logging outputs
B. Enforcing policy-as-code
C. Managing cloud credentials
D. Encrypting variables

✅ **Answer:** B

---

9. What feature allows Terraform Cloud to limit access based on user roles?

A. IAM
B. RBAC (Role-Based Access Control)
C. ACL
D. MFA

✅ **Answer:** B

---

10. How can Terraform Cloud store state securely?

A. In .zip files
B. On GitHub
C. Using encrypted remote state with access controls
D. Local-only state

✅ **Answer:** C

---

11. What is a variable set in Terraform Cloud?

A. A group of values passed to a function
B. A list of modules
C. A group of input variables shared across workspaces
D. A state snapshot
✅ **Answer:** C

---

12. How does Terraform Cloud handle secrets like AWS credentials?

A. In plain text files
B. In Terraform .tf files
C. Using environment variables or sensitive variables
D. By hardcoding in main.tf
✅ **Answer:** C

---

13. How can you estimate resource cost before provisioning using Terraform Cloud?

A. Enable cost estimation feature
B. Manually calculate with a calculator
C. Use AWS billing dashboard
D. Only Terraform Enterprise can do that
✅ **Answer:** A

---

14. How does Terraform Cloud manage versions of remote state?

A. By storing only the latest version
B. Through .tfvars backups
C. By keeping historical versions automatically
D. No support for state versioning
✅ **Answer:** C

---

15. What is the default method for applying changes in Terraform Cloud?

A. Manually using CLI
B. Automatically on plan success
C. Manual approval via the UI

D. Always automatic

✅ **Answer:** C

---

16. How are variables passed to Terraform Cloud workspaces?

A. Using hardcoded values only
B. Through .tf files only
C. Through variable sets and workspace-specific variables
D. Via Terraform plugins

✅ **Answer:** C

---

17. Which of the following supports VCS-driven runs in Terraform Cloud?

A. Bitbucket, GitLab, GitHub
B. Only GitHub
C. Only Terraform Enterprise
D. Azure DevOps only

✅ **Answer:** A

---

18. What feature prevents multiple users from editing state simultaneously in Terraform Cloud?

A. Version control
B. RBAC
C. State locking
D. Workspace forking

✅ **Answer:** C

---

19. Which of these is NOT a valid use case for Terraform Cloud?

A. Running Terraform in a central location
B. Storing state remotely
C. Sending logs to AWS CloudWatch
D. Collaborating via VCS integration

✅ **Answer:** C

---

20. What type of variable should be used to hide sensitive values in Terraform Cloud?

A. Public variable
B. Encrypted variable
C. Sensitive variable
D. Environment variable
✅ **Answer:** C

---

21. What does the Cost Estimation feature in Terraform Cloud depend on?

A. Terraform Enterprise license
B. VCS integration
C. Provider-specific pricing data
D. Local terraform plan
✅ **Answer:** C

---

22. In Terraform Cloud, what is the function of the UI workflow?

A. Manage AWS infrastructure
B. Execute all shell scripts
C. Review plans and apply with manual approval
D. Build Docker images
✅ **Answer:** C

---

23. What backend keyword enables use of Terraform Cloud?

A. remote
B. cloud
C. tfc
D. enterprise
✅ **Answer:** A

---

24. Can Terraform Cloud be used with terraform login for CLI access?

A. No
B. Yes, it saves an API token locally
C. Only in Enterprise
D. Only with GitHub
✅ **Answer:** B

---

25. What permission level in Terraform Cloud allows changing settings, plans, and applies?

A. Viewer
B. Owner
C. Contributor
D. Maintainer
✅ **Answer:** C

🛡️ Topic 10: Understand Security Best Practices

Aligned with the Terraform Associate (v1.3/003) exam.

---

1. Why should you avoid storing .tfstate files in version control systems like Git?

A. To prevent file corruption
B. Because they are not readable
C. They often contain sensitive information
D. Git doesn't support .tfstate files
✅ **Answer:** C

---

2. Which argument in the output block ensures sensitive values are not shown in the CLI?

A. hide = true
B. secret = true
C. sensitive = true
D. private = true
✅ **Answer:** C

---

3. What is a secure method for passing secrets into Terraform configurations?

A. Storing them in plain text in main.tf
B. Using environment variables like TF_VAR_
C. Including them directly in .tfvars committed to Git
D. Putting them in output blocks
✅ **Answer:** B

---

4. What feature can help restrict who accesses Terraform Cloud state files?

A. GitHub integration
B. RBAC (Role-Based Access Control)
C. SSH access
D. Auto-approval
✅ **Answer:** B

---

5. Which backend supports encryption for secure state storage?

A. Local backend
B. Terraform Cloud without encryption
C. S3 backend with server-side encryption
D. CLI backend
✅ **Answer:** C

---

6. How can you prevent secrets from being hardcoded in Terraform configuration files?

A. Store in main.tf
B. Use Vault, environment variables, or remote backends
C. Print them via terraform output
D. Store in .gitignore
✅ **Answer:** B

---

7. What does setting sensitive = true do in Terraform?

A. Encrypts the file
B. Prevents logging to screen or state
C. Prevents the value from showing in plan and apply outputs
D. Hides resource names
✅ **Answer:** C

---

8. Where is the best place to store provider credentials securely?

A. Inside the Terraform file
B. In Terraform Registry
C. In environment variables or a secrets manager

D. In terraform.tfvars

✅ **Answer:** C

---

9. What is a risk of storing .tfstate locally instead of using a remote backend?

A. Incompatibility with AWS
B. Slower apply operations
C. Loss of collaboration, lack of locking, and exposure of secrets
D. It doesn't work with CLI

✅ **Answer:** C

---

10. How can you ensure Terraform state is encrypted at rest when using AWS?

A. Use the local backend
B. Use S3 with SSE (Server-Side Encryption)
C. Store it in plain text and compress it
D. Use .zip file format

✅ **Answer:** B

---

11. What IAM policy should you use for state file security in S3?

A. Public-read
B. Read/write access restricted to specific users
C. Admin access for all users
D. No policy needed

✅ **Answer:** B

---

12. What is the role of state locking in security?

A. Speeding up Terraform runs
B. Preventing parallel operations that could corrupt state
C. Encrypting the state file
D. Backing up the state file

✅ **Answer:** B

---

13. How do you prevent a resource from being destroyed accidentally in Terraform?

A. Use force_destroy = true
B. Use prevent_destroy in the lifecycle block
C. Use auto_destroy
D. Mark it as read-only
✅ **Answer:** B

---

14. Why should .terraform directories be excluded from version control?

A. To avoid large file commits
B. They contain temporary data and provider plugins
C. They are required by Git
D. They increase speed
✅ **Answer:** B

---

15. What can you use to rotate credentials securely in Terraform pipelines?

A. Manual updates in .tf
B. Hardcoded strings
C. Secrets manager or dynamic credentials
D. Output values
✅ **Answer:** C

---

16. Why is plaintext secret storage in .tfvars files discouraged?

A. Terraform doesn’t support it
B. Secrets may get pushed to version control
C. Variables don’t work with secrets
D. They can’t be encrypted
✅ **Answer:** B

---

17. How can workspaces help with security and environment separation?

A. By using one state file for all environments
B. By isolating state files per environment
C. By avoiding the use of variable files
D. They don't affect security
✅ **Answer:** B

---

18. What is one benefit of using a remote backend like Terraform Cloud or S3 over local state?

A. No need to use terraform apply
B. You can see all your code in a UI
C. Enhanced security and team collaboration
D. Faster file reads
✅ **Answer:** C

---

19. What can you use to securely pass variables in CI/CD pipelines?

A. .tfvars committed to Git
B. Output blocks
C. Environment variables or encrypted secrets
D. Terraform logs
✅ **Answer:** C

---

20. Which option allows for secrets to be passed securely without being logged?

A. sensitive = true
B. secure_string = yes
C. encrypt = true
D. hidden = true
✅ **Answer:** A

---

21. Which of the following is a security risk when using Terraform?

A. Using remote backends
B. Exposing variables in terraform output
C. Version pinning providers
D. Using HCL
✅ **Answer:** B

---

22. When should you consider using Vault with Terraform?

A. To enable remote logging
B. To share configurations across teams
C. To manage and inject secrets securely

D. To increase performance

✅ **Answer:** C

---

23. Which of the following best protects sensitive output values?

A. Use terraform taint
B. Set sensitive = true in outputs
C. Store them in plaintext in outputs.tf
D. Run terraform state show

✅ **Answer:** B

---

24. How can RBAC in Terraform Cloud improve security?

A. By allowing any user to apply changes
B. By enforcing state encryption
C. By controlling access to runs, variables, and state
D. By reducing plan file size

✅ **Answer:** C

---

25. What should you do before sharing a .zip of your Terraform project?

A. Run terraform plan
B. Compress the .terraform directory
C. Remove .tfstate and sensitive variables
D. Enable debug mode

✅ **Answer:** C

📚 Bonus: Frequently Covered Topics in Exams

1. What command is used to import existing infrastructure into Terraform?

A. terraform get
B. terraform import
C. terraform state add
D. terraform add

Answer: B

2. What is the default name of the Terraform state file?

A. terraform.tf
B. terraform.tfvars

C. terraform.tfstate
D. state.tf
Answer: C

3. Why should secrets not be stored in Terraform files directly?

A. They get encrypted automatically
B. To keep configurations shorter
C. To avoid exposure via version control
D. To reduce execution time
Answer: C

4. How can Terraform be integrated with CI/CD tools like GitHub Actions?

A. By using terraform destroy after every run
B. By manually copying files
C. By creating workflows that run terraform init, plan, and apply
D. By uploading plans as zip files
Answer: C

5. What is a difference between count and for_each in Terraform?

A. count works only with strings
B. for_each allows mapping over complex structures
C. count is slower than for_each
D. for_each cannot be used with modules
Answer: B

6. What does terraform graph generate?

A. Plan outputs in JSON
B. Resource dependency graphs in DOT format
C. Logs of resource creation
D. State backup
Answer: B

7. What type of file does terraform plan -out=planfile create?

A. JSON
B. Encrypted secret file
C. Binary plan file
D. Plain text plan
Answer: C

8. What is required to import an existing AWS resource into Terraform?

A. An output block
B. The exact resource ID
C. Running terraform plan
D. No configuration is required
Answer: B

9. Which file lists provider versions used in a project?

A. terraform.tfvars
B. main.tf
C. .terraform.lock.hcl
D. outputs.tf
Answer: C

10. What is a common use of terraform taint?

A. Reformat the code
B. Force Terraform to destroy and recreate a resource
C. Secure a resource
D. Unlock a locked state file
Answer: B

11. What does the terraform refresh command do?

A. Refreshes backend credentials
B. Synchronizes remote and local state
C. Re-runs module installation
D. Validates the configuration
Answer: B

12. Where should .terraform directory be placed in .gitignore?

A. Never
B. It must be included in version control
C. Always
D. Only on production environments
Answer: C

13. How does Terraform handle sensitive output values?

A. Ignores them
B. Prints them in cleartext
C. Obfuscates them in CLI output
D. Stores them in .env file
Answer: C

14. Which backend is recommended for state locking and versioning?

A. Local
B. Git
C. S3 with DynamoDB
D. Azure DevOps
Answer: C

15. What is the best practice for separating Terraform environments?

A. Use separate GitHub repos
B. Use Terraform Workspaces
C. Use different variable names
D. Use one state file
Answer: B

16. How do you pass credentials securely to Terraform?

A. Hardcode in variables.tf
B. Use environment variables like TF_VAR_
C. Upload in Git
D. Use plaintext in modules
Answer: B

17. What is the use of terraform output command?

A. Shows resources to be deleted
B. Displays current outputs from state
C. Deletes output blocks
D. Refreshes modules
Answer: B

18. What happens when a resource is removed from configuration but not from state?

A. Terraform deletes it
B. It's marked as tainted
C. It causes a drift
D. Nothing
Answer: C

19. What command removes a resource from state without destroying it?

A. terraform destroy
B. terraform delete
C. terraform state rm
D. terraform graph
Answer: C

20. How do you change the name of a resource in state?

A. terraform taint
B. terraform rename
C. terraform state mv
D. terraform plan
Answer: C

21. Which command lists all resources tracked in Terraform state?

A. terraform show
B. terraform graph
C. terraform state list
D. terraform plan
Answer: C

---

22. Where is .terraform.lock.hcl used?

A. Managing Terraform Cloud RBAC
B. Locking state changes
C. Ensuring consistent provider versions
D. Controlling backend access
Answer: C

---

23. What kind of data does a terraform.tfstate.backup file hold?

A. Backend configuration
B. Resource credentials
C. Previous known good state
D. Execution logs
Answer: C

---

24. Which option lets you skip user approval during apply?

A. --force
B. -var-file
C. -auto-approve
D. -skip-plan
Answer: C

---

25. What file contains runtime values captured after apply?

A. variables.tf
B. terraform.tfvars
C. terraform.tfstate
D. backend.tf
Answer: C

---

26. Which variable definition file format is supported by Terraform?

A. YAML
B. INI
C. .tfvars
D. .env
Answer: C

---

27. How do you format Terraform configuration files?

A. terraform format
B. terraform lint
C. terraform fmt
D. terraform validate
Answer: C

---

28. How do you test if Terraform configuration is syntactically valid?

A. terraform show
B. terraform validate
C. terraform inspect
D. terraform test
Answer: B

---

29. How can secrets be stored securely for Terraform use?

A. In plain text
B. In GitHub Secrets
C. In resource names
D. In the state file
Answer: B

---

30. What command moves state between modules or resources?

A. terraform state swap
B. terraform state transfer
C. terraform state mv
D. terraform rename
Answer: C

---

31. What is a drawback of using local state?

A. It's version controlled
B. It can't be encrypted
C. It's hard to collaborate in teams
D. It supports multiple backends
Answer: C

---

32. What command removes a taint from a resource?

A. terraform clean
B. terraform untaint
C. terraform remove
D. terraform reset
Answer: B

---

33. Which Terraform setting helps prevent accidental deletion?

A. lifecycle { count }
B. prevent_destroy
C. taint_protection
D. safe_apply
Answer: B

---

34. What output command argument shows values in JSON?

A. terraform output -format=json
B. terraform output -json
C. terraform show -json

D. terraform get -json
Answer: B

---

35. Why is state locking important?

A. For faster apply
B. To prevent concurrent changes
C. To avoid storing secrets
D. To backup files
Answer: B

---

36. What should you use to structure reusable code in Terraform?

A. Outputs
B. Locals
C. Modules
D. Data sources
Answer: C

---

37. How does Terraform identify existing resources in import?

A. Using tags
B. Resource type and ID
C. Output values
D. Names only
Answer: B

---

38. Which Terraform command removes all infrastructure?

A. terraform init
B. terraform destroy
C. terraform rm
D. terraform remove
Answer: B

---

39. What resource type does aws_instance.web refer to?

A. Module
B. Variable

C. Resource
D. Output
Answer: C

---

40. What is the best place to define common variables for multiple environments?

A. outputs.tf
B. terraform.tfstate
C. .tfvars files
D. terraform.lock.hcl
Answer: C

41. What is the purpose of terraform graph?

A. Display plan output in JSON
B. Visualize dependency graph
C. Show state differences
D. Render documentation
✅ **Answer:** B

42. What file should be excluded from version control?

A. main.tf
B. variables.tf
C. terraform.tfvars
D. terraform.tfstate
✅ **Answer:** D

43. What tool can securely inject secrets into Terraform?

A. Docker Secrets
B. AWS IAM
C. Vault
D. GitHub Pages
✅ **Answer:** C

44. What does the count meta-argument control?

A. Resource timeout
B. Number of instances to create
C. Variable types
D. Backend retries
✅ **Answer:** B

45. What does for_each allow that count does not?

A. Parallel resource creation
B. Use with data sources
C. Map and set iteration
D. Static resource naming

✅ **Answer:** C

46. How do you prevent sensitive outputs from showing in CLI?

A. hidden = true
B. encrypted = true
C. sensitive = true
D. secure = true

✅ **Answer:** C

47. What happens when you run terraform init?

A. Applies configuration
B. Downloads providers and modules
C. Creates output values
D. Imports resources

✅ **Answer:** B

48. What command helps you simulate what will be changed?

A. terraform show
B. terraform preview
C. terraform plan
D. terraform dry-run

✅ **Answer:** C

49. Where should a provider version constraint be defined?

A. variables.tf
B. terraform.tfvars
C. main.tf
D. required_providers block

✅ **Answer:** D

50. What environment variable is used to pass secrets securely?

A. TF_OUTPUT
B. TF_VAR_name
C. TF_SECRET

D. TF_CONFIG

✅ **Answer:** B

51. How can Terraform configurations be tested in pipelines?

A. Using terraform destroy
B. With Bash scripts only
C. With CI/CD tools like GitHub Actions
D. Using terraform import

✅ **Answer:** C

52. What file defines reusable outputs in a module?

A. inputs.tf
B. variables.tf
C. main.tf
D. outputs.tf

✅ **Answer:** D

53. What is the benefit of separating environments into workspaces?

A. Speed
B. Cost savings
C. Isolated state per environment
D. Syntactic validation

✅ **Answer:** C

54. How does Terraform identify the resource to import?

A. Name only
B. State
C. ID and resource type
D. .tfvars

✅ **Answer:** C

55. What type of file is backend.tf?

A. Data source config
B. Provider credentials
C. Remote state configuration
D. Module import definition

✅ **Answer:** C

56. What command helps fix formatting in .tf files?

A. terraform fix
B. terraform fmt
C. terraform validate
D. terraform layout

✅ **Answer:** B

57. Which command can show full state contents?

A. terraform fmt
B. terraform plan
C. terraform show
D. terraform validate

✅ **Answer:** C

58. Which CLI option applies changes without confirmation?

A. -confirm=false
B. -force
C. -y
D. -auto-approve

✅ **Answer:** D

59. What does terraform validate check?

A. Provider credentials
B. Code syntax and structure
C. State drift
D. IAM permissions

✅ **Answer:** B

60. What is .terraform directory used for?

A. State backups
B. Module and provider cache
C. Logs only
D. Remote plan output

✅ **Answer:** B

61–70: CI/CD, Secrets Handling, Import/Graph, Output

61. Which tool is commonly used for Terraform CI/CD?

A. Grafana
B. Jenkins
C. Terraform Cloud only

D. Docker

✅ **Answer:** B

62. Why avoid hardcoding secrets in .tf files?

A. For faster parsing
B. To reduce file size
C. Security best practice
D. Prevent duplication

✅ **Answer:** C

63. What feature tracks resource relationships visually?

A. terraform output
B. terraform list
C. terraform graph
D. terraform show

✅ **Answer:** C

64. What does terraform output do?

A. Shows plan logs
B. Displays output values
C. Imports resources
D. Lists variables

✅ **Answer:** B

65. Which command is used in automation pipelines to check file layout?

A. terraform validate
B. terraform scan
C. terraform plan
D. terraform fmt

✅ **Answer:** D

66. What feature enables reusing configurations across environments?

A. Terraform Cloud
B. Workspaces
C. Static backends
D. terraform destroy

✅ **Answer:** B

67. Which of the following supports VCS integration?

A. Terraform CLI
B. Terraform Cloud
C. State file
D. Locals

✅ **Answer:** B

68. Which file should never be committed to version control?

A. outputs.tf
B. terraform.tfstate
C. variables.tf
D. versions.tf

✅ **Answer:** B

69. When importing, which command format is correct?

A. terraform import resource_name
B. terraform import module.module_name
C. terraform import <address> <ID>
D. terraform plan -import

✅ **Answer:** C

70. What happens if you re-apply without changes?

A. Terraform errors out
B. Configuration is re-created
C. Nothing changes (idempotency)
D. Outputs disappear

✅ **Answer:** C

71–80: Error Handling, Secrets, Dynamic Infra

71. What kind of error might terraform apply show when a resource doesn't exist anymore?

A. PlanError
B. DriftError
C. ResourceNotFound
D. StateMismatch

✅ **Answer:** C

72. Which feature ensures repeatable infrastructure changes?

A. terraform destroy
B. terraform drift

C. Idempotency
D. plan suppression
✅ **Answer:** C

73. How can you dynamically build blocks in HCL?

A. count
B. depends_on
C. dynamic
D. lookup
✅ **Answer:** C

74. What tool encrypts secrets for use in Terraform workflows?

A. Docker
B. GitHub Pages
C. HashiCorp Vault
D. Terraform Cloud
✅ **Answer:** C

75. How do you define environment variables in CI pipelines for Terraform?

A. export TF_VAR_*
B. Define in locals
C. Use variables.tf
D. Commit in .tf files
✅ **Answer:** A

76. How do you output sensitive values securely?

A. output "value" { sensitive = true }
B. Use encrypted.tf
C. Put in variable name
D. Use data "secret"
✅ **Answer:** A

77. What does state drift indicate?

A. Syntax errors
B. CLI issues
C. Infra changed outside Terraform
D. Backend misconfiguration
✅ **Answer:** C

78. How to fix state drift?

A. Delete the state file
B. Run terraform plan
C. Run terraform refresh
D. Re-import resources

✅ **Answer:** C

79. What type of data can locals store?

A. Sensitive secrets
B. Constant computed values
C. Module outputs
D. Provider names

✅ **Answer:** B

80. Why use terraform plan in CI/CD?

A. To avoid using CLI
B. To ensure code compiles
C. To preview changes before apply
D. To edit variables

✅ **Answer:** C

81. What is the purpose of .terraform.lock.hcl?

A. Track sensitive data
B. Lock workspace configurations
C. Ensure consistent provider versions
D. Control backend settings

✅ **Answer:** C

82. What happens when you change a resource manually outside Terraform?

A. Terraform auto-detects and updates code
B. State drift occurs
C. Terraform removes the resource
D. Nothing, Terraform ignores it

✅ **Answer:** B

83. Which resource argument lets you wait for one resource before another?

A. wait_on
B. depends_on
C. delay_with
D. after_resource

✅ **Answer:** B

84. Which expression creates multiple instances based on a list?

A. resource "aws_instance" "web" {}
B. for_each = ["a", "b"]
C. count = 2
D. count = length(list)
✅ **Answer:** D

85. Where is the preferred place to define reusable default values?

A. terraform.tfvars
B. backend.tf
C. variables.tf
D. outputs.tf
✅ **Answer:** C

86. How does Terraform know if a resource must be replaced?

A. Resource has drifted
B. Lifecycle policy
C. Incompatible attribute change
D. Count has changed
✅ **Answer:** C

87. What is the output of terraform plan?

A. Applied resources
B. JSON logs
C. Execution plan showing changes
D. Final state
✅ **Answer:** C

88. Which block is used to fetch existing data from outside Terraform?

A. module {}
B. data {}
C. output {}
D. local {}
✅ **Answer:** B

89. Which command restores a previous state from backup?

A. terraform rollback
B. terraform plan -refresh
C. terraform state restore

D. Manually copy .tfstate.backup

✅ **Answer:** D

90. What kind of data structure can be passed with for_each?

A. Single string
B. List or Map
C. Output block
D. Resource reference

✅ **Answer:** B

---

91. Which meta-argument ensures a resource is not deleted?

A. depends_on
B. lifecycle { prevent_destroy = true }
C. sensitive = true
D. skip_delete = true

✅ **Answer:** B

92. What does terraform taint do?

A. Deletes the resource
B. Marks a resource for recreation
C. Encrypts the resource
D. Locks the state

✅ **Answer:** B

93. What command initializes the working directory?

A. terraform get
B. terraform init
C. terraform setup
D. terraform plan

✅ **Answer:** B

94. What is the use of terraform state mv?

A. Moves configuration files
B. Renames resources
C. Changes resource addressing in state
D. Moves plan file

✅ **Answer:** C

95. Why use terraform import?

A. Migrate state to cloud
B. Add resources to configuration
C. Bring existing infrastructure under Terraform control
D. Validate modules

✅ **Answer:** C

96. What does the terraform workspace command manage?

A. CI/CD environments
B. Backend configs
C. Logical state partitions
D. Git branches

✅ **Answer:** C

97. Which Terraform feature supports code reuse and encapsulation?

A. Resource blocks
B. Data sources
C. Modules
D. Locals

✅ **Answer:** C

98. What does terraform refresh do?

A. Reloads CLI
B. Updates remote backend
C. Syncs state with real infrastructure
D. Resets output values

✅ **Answer:** C

99. What file format is Terraform state stored in?

A. YAML
B. JSON
C. XML
D. HCL

✅ **Answer:** B

100. What is a benefit of separating .tfvars files by environment?

A. Faster plan time
B. Variable reusability and environment-specific values
C. Smaller state size
D. Easier backend configuration

✅ **Answer:** B

🧰 Terraform Core Workflow Commands

Command	Description
terraform init	Initializes a working directory with Terraform configuration files
terraform plan	Shows an execution plan of what Terraform will do
terraform apply	Applies the changes required to reach the desired state
terraform destroy	Destroys the managed infrastructure
terraform refresh	Updates the state file with real infrastructure changes

---

🧪 Validation, Formatting & Inspection

Command	Description
terraform validate	Checks the configuration syntax and internal consistency
terraform fmt	Formats Terraform configuration files to canonical style
terraform providers	Lists the provider requirements of the configuration
terraform show	Displays details from a Terraform state or plan file
terraform output	Extracts output values from state or plan
terraform console	Interactive REPL to evaluate expressions and inspect values

---

🔄 State Management Commands

Command	Description
terraform state list	Lists resources tracked in state
terraform state show <resource>	Shows attributes of a specific resource
terraform state rm <resource>	Removes resource from the state (does NOT delete the infra)

Command	Description
terraform state mv <old> <new>	Moves or renames items in the state
terraform taint <resource>	Marks a resource to be recreated during next apply
terraform untaint <resource>	Removes the taint from a resource

---

📦 Modules and Configuration

Command	Description
terraform get	Downloads and installs modules mentioned in the config
terraform init -upgrade	Reinitializes and updates modules and providers
terraform plan -var-file="file.tfvars"	Uses variables defined in an external file
terraform plan -out=planfile	Outputs the plan to a file for later use
terraform apply planfile	Applies the previously saved plan
terraform import	Brings existing infrastructure under Terraform management

---

🌐 Workspace Management

Command	Description
terraform workspace list	Lists all Terraform workspaces
terraform workspace new <name>	Creates a new workspace
terraform workspace select <name>	Switches to a different workspace
terraform workspace show	Shows the current workspace

🔍 Debugging and Troubleshooting

Command / Env Var	Description
TF_LOG=DEBUG	Enables verbose logging for troubleshooting
TF_LOG_PATH=./tf.log	Writes debug logs to a file
TF_VAR_<variable>	Defines a variable from the environment
terraform plan -detailed-exitcode	Returns 2 when there are changes (helpful in CI/CD)

📈 Visualization and Analysis

Command	Description
terraform graph	Generates a DOT format graph of resource relationships

---

📜 Lock File and Dependency Handling

Command	Description
.terraform.lock.hcl	Lock file managed automatically, ensures provider versions don’t drift
terraform providers lock	(In some versions) explicitly update lock file

---

🧩 Terraform Cloud / Backend Specific (if used)

Command	Description
terraform login	Authenticates to Terraform Cloud
terraform logout	Removes credentials
terraform state pull	Downloads latest remote state
terraform state push	Manually uploads a local state file (use with caution)

🧭 Common Terraform CLI Flags

✅ terraform init

Flag	Description
-upgrade	Upgrade modules and providers to latest allowed versions
-backend=false	Skip backend configuration initialization
-reconfigure	Reconfigure the backend, ignoring saved settings

---

📋 terraform plan

Flag	Description
-out=planfile	Saves the generated execution plan to a file
-var='key=value'	Sets a variable inline
-var-file=filename.tfvars	Loads variables from a file
-input=false	Disables prompting for input variables
-detailed-exitcode	Returns exit code 2 if there are changes (useful in CI/CD)

🚀 terraform apply

Flag	Description
-auto-approve	Skips the confirmation prompt
planfile	Applies changes from a saved plan file
-input=false	Disables interactive input
-lock=false	Skips state locking during apply (use with caution)

🔥 terraform destroy

Flag	Description
-auto-approve	Skips confirmation before destroying resources
-target=resource	Destroy specific resource(s) only

🔎 terraform show

Flag	Description
-json	Outputs the state or plan in JSON format

📤 terraform output

Flag	Description
-json	Outputs result in JSON
-raw	Outputs value as raw string (useful for scripts)

🧪 terraform validate

Flag	Description
-json	Outputs validation messages in JSON (for tooling integrations)

📁 terraform workspace

Flag	Description
-no-color	Disables colored output (useful in CI logs)

🌎 Environment Variables (often used as flags)

Variable	Description
TF_LOG=DEBUG	Enables detailed logging (TRACE, DEBUG, INFO, WARN, ERROR)

Variable	Description
TF_LOG_PATH=path/to/file.log	Writes logs to a specified file
TF_VAR_<NAME>	Sets a variable for Terraform via environment (e.g., TF_VAR_region)
TF_INPUT=false	Disables prompting for user input
TF_CLI_ARGS	Default flags appended to every Terraform command (not recommended for sensitive changes)

🎯 -target=resource

This flag can be used with plan, apply, or destroy to focus on a specific resource:

terraform plan -target=aws_instance.web