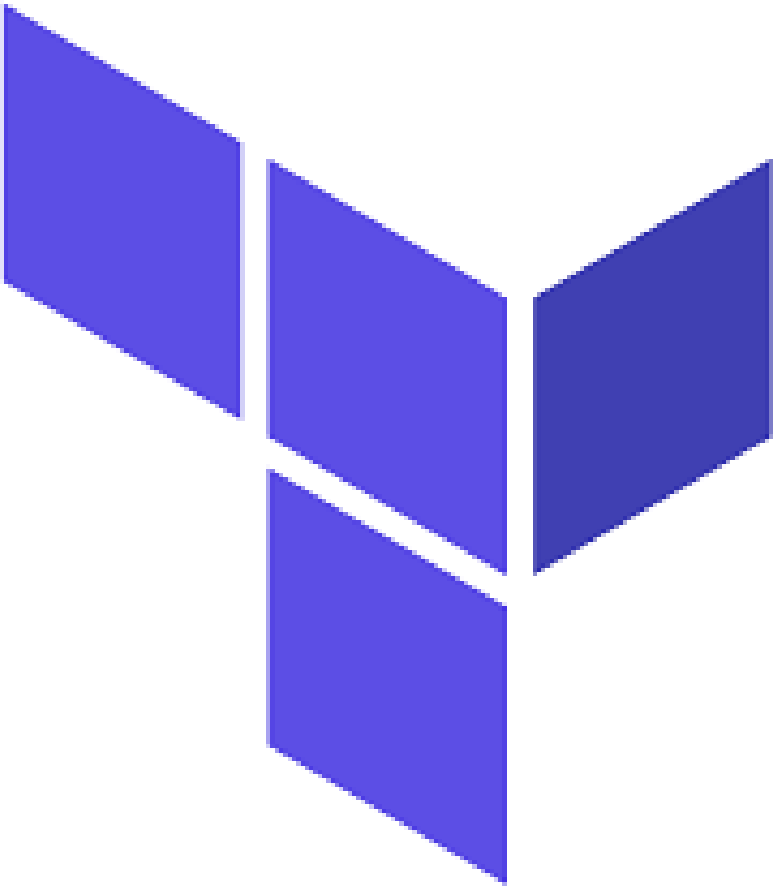

Bavithran
@bavicnative

Terraform



Top 30 Terraform Interview Questions & Answers


Bavithran
@bavicnative

1. What is Terraform and why is it used?

Terraform is an open-source Infrastructure as Code (IaC) tool by HashiCorp that allows you to define, provision, and manage infrastructure using declarative configuration files.

It is used for:

- Multi-cloud provisioning
- Infrastructure standardization
- Version-controlled infra
- Reproducible environments
- Automation and scalability

2. Explain HCL (HashiCorp Configuration Language).

HCL is a declarative domain-specific language used to define Terraform resources.

It is designed for:

- Human readability
- Machine-parsable execution
- Modular organization
- Reusability

3. What is the difference between Terraform and other IaC tools like CloudFormation/Bicep?

Terraform

- Supports multi-cloud environments (AWS, Azure, GCP, etc.)
- Uses HCL, a human-readable and modular language
- State tracked locally or remotely (S3, Azure Storage, Terraform Cloud)
- Has the largest provider ecosystem across clouds and SaaS platforms

AWS CloudFormation

- Works only with AWS
- Uses JSON/YAML templates
- State fully managed by AWS, no manual handling
- Ecosystem limited to AWS resources

Azure Bicep

- Azure-only IaC tool
- Uses the simplified Bicep DSL (easier than ARM JSON)
- No external state — Azure manages everything internally
- Limited ecosystem, but deeply integrated with Azure services

4. What is Terraform State? Why is it needed?

Terraform uses a state file (terraform.tfstate) to map real infrastructure to configuration.

It stores:

- Resource attributes
- Dependencies
- Metadata

Why needed:

- Tracks which resources exist
- Enables incremental updates
- Supports change detection

5. What are Terraform Providers?

Providers are plugins that allow Terraform to interact with APIs of cloud platforms and services.

Examples:

- azurerm
- aws
- google
- kubernetes
- vault

They define resources + data sources for the service.

6. What is the difference between Resources and Data Sources?

- Resource: Creates/updates/deletes actual infrastructure.
- Data Source: Reads existing infrastructure without modifying it.

Example:
data "aws_ami" → fetch AMI
resource "aws_instance" → deploy EC2

7. Explain Terraform Modules.

Modules are reusable units of Terraform configuration that group resources.

Benefits:

- Reusability
- Standardization
- Cleaner code
- Best practice enforcement

Types:

- Root module
- Child/local module
- Public module registry

8. What is a Remote Backend? Why use it?

- A backend determines where Terraform stores its state.

Remote backends:

- Azure Storage
- AWS S3 + DynamoDB
- GCS
- Terraform Cloud

Benefits:

- State locking
- Collaboration
- Secure storage
- Versioning

9. Explain State Locking.

Prevents multiple users from running terraform apply simultaneously, avoiding corruption.

Example backends with locking:

- S3 + DynamoDB
- Terraform Cloud
- Consul

10. What is Terraform Plan?

terraform plan shows what actions Terraform will take before applying changes:

- Add
- Change
- Destroy

It provides a safe preview for review.

11. What is the difference between terraform apply and terraform destroy?

- apply: Creates or updates resources.
- destroy: Deletes all resources defined in config.

12. What are Variables and Outputs?

Variables → Input parameters
Outputs → Export resource attributes

Used for:

- Passing values between modules
- Displaying useful info
- CI/CD pipeline integration

13. What are Variable Types in Terraform?

Types include:

- string
- number
- bool
- list
- map
- object
- tuple

14. What are Terraform Workspaces?

Workspaces allow multiple state files for the same configuration.

Used for:

- dev
- staging
- prod

Workspace = separate state = isolated environments.

15. How does Terraform handle dependencies?

- Implicit (resource references)
- Explicit (depends_on)

Terraform automatically builds a dependency graph.

16. What is the Terraform Registry?

A public catalog of modules and providers.

You can download ready-made modules for AWS/Azure/GCP/K8s.

17. How do you import existing infrastructure into Terraform?

Using terraform import:

terraform import aws_instance.example i-123456789

After import, you must write the corresponding configuration manually.

18. What is the difference between terraform refresh and terraform plan?

- refresh: Updates state file from actual infrastructure.
- plan: Shows changes required to move infra to desired state.

19. Explain Terraform Lifecycle Rules.

Used to optimize provisioning:

```
lifecycle {
  create_before_destroy = true
  prevent_destroy = true
  ignore_changes = [tags]
}
```

20. What is Tainting?

Marks a resource for recreation.

terraform taint aws_instance.example

21. What is Drift?

When real infrastructure changes outside Terraform, leading to:

- Inconsistent state
- Unexpected apply results

Detected via:

- terraform plan
- terraform refresh

22. How do you test Terraform code?

Using:

- Terratest
- Checkov
- OPA (Open Policy Agent)
- Conftest
- tfsec

23. What are Terraform Provisioners?

Used for last-mile configuration (rarely recommended):

- remote-exec
- local-exec

Best practice: avoid provisioners.

24. What is Terragrunt?

A wrapper for Terraform providing:

- DRY folder structures
- Better state management
- Dependency orchestration
- Multi-environment workflows

Used in large enterprises.

25. What is Sensitive Data Handling?

Use:

- sensitive = true
- Vault / Secrets Manager
- Avoid variables.tf for passwords
- Use CI/CD secret context

26. How do you manage multiple environments?

Methods:

- Workspaces
- Separate repos
- Folder structure
- Terragrunt
- Remote backends

27. How does Terraform ensure idempotency?

Because Terraform is declarative:

- Same config = same infra
- Only required changes applied
- State reconciles differences

28. What is a Null Resource?

A resource used to run scripts or triggers.

```
resource "null_resource" "deploy" {
  triggers = { app = timestamp() }
}
```

29. What is the difference between local-exec and remote-exec?

- local-exec: runs on the machine executing Terraform
- remote-exec: runs inside the created VM/container

30. What are Terraform Best Practices in 2026?

- Use modules for everything
- Implement CI/CD pipeline for Terraform
- Use remote backend with locking
- Avoid long root modules
- Enforce policies using OPA/Checkov
- Use version pinning
- Naming conventions & tagging standards
- Always run terraform validate + format
- Automate security scans
- Use least-privilege IAM for Terraform

Found this useful?

Follow

Let's connect

💬 Found it useful? Drop your thoughts below and share it with your fellow DevOps engineers!