Mock Interview Guide
Terraform

Instructions for Interviewer:

- You are playing the role of **interviewer**. Use this guide as a script.
- Ask each question one at a time. Follow the steps: **Definition → Details → Scenario → Follow-up.**
- If the interviewee struggles, use the **hint**.
- The goal is to keep it conversational and practical. Help the interviewee think and express their learning.
- **colors assigned:** Questions Answers Hint

Freshers - Level
Terraform
(10 Easy DevOps Interview Questions)

---

1. "What is Terraform used for?"

✓ Expected Answer: "Terraform is an open-source tool used to automate the provisioning of cloud infrastructure using code."

Hint: ***"Think of Terraform like writing blueprints for your cloud. Instead of clicking around in AWS or Azure, you write code that builds the setup for you."***

2. "What does IaC mean in Terraform?"

✓ Expected Answer: "IaC stands for Infrastructure as Code. It means managing and provisioning infrastructure through code instead of manually."

Hint: ***"Imagine describing your servers and networks the same way you write code for apps — then letting a tool set them up for you."***

3. "What file format does Terraform use?"

✅ Expected Answer: "Terraform uses files written in HashiCorp Configuration Language (HCL) with .tf extension."

Hint: ***"You're writing configuration, not code logic. It's stored in plain-text files with a specific extension."***

4. "What is terraform init?"

✅ Expected Answer: "It initializes a Terraform working directory and downloads required providers."

Hint: ***"The very first step before you apply any changes — it sets up your project folder and downloads cloud plugins."***

5. "What does terraform plan do?"

✅ Expected Answer: "It shows what changes Terraform will make to the infrastructure before applying them."

Hint: ***"Want to see what will happen before you actually build something? This command gives you that preview."***

6. "How do you apply the Terraform configuration?"

✅ Expected Answer: "By using the terraform apply command."

Hint: ***"Once you've planned your infrastructure, this command actually builds or updates it."***

7. "How do you destroy resources created by Terraform?"

✅ Expected Answer: "By running the terraform destroy command."

Hint: ***"Used when you want to clean up everything you've provisioned using your*** **.tf** ***files."***

8. "What is a Terraform provider?"

✅ Expected Answer: "A provider is a plugin that Terraform uses to interact with APIs of cloud platforms like AWS, Azure, GCP."

Hint: ***"This is how Terraform knows how to talk to AWS or Azure — think of it as the bridge between Terraform and the cloud."***

9. "Where are Terraform states stored?"

✅ Expected Answer: "By default, locally in a file called terraform.tfstate, but it can be stored remotely."

Hint: ***"This file keeps track of what's already built, so Terraform knows what needs changing."***

10. "What is a variable in Terraform?"

✅ Expected Answer: "A variable is a reusable value defined in configuration that can be customized."

Hint: ***"Think of it like a placeholder — instead of hardcoding values like AMI ID or region, you make it flexible."***

SCENARIO-BASED INTERVIEW QUESTIONS

---

1. Ask: *"You've run* terraform plan*, but nothing changed in your cloud environment. What do you think might have been missed?"*

✅ **Expected:** *The candidate likely forgot to run* **terraform apply** *after planning.*

Hint: *What command actually makes the changes?*

2. Ask: *"Suppose you updated a value in* terraform.tfvars*, but running* terraform plan *shows no difference. What could be the reason?"*

✅ **Expected:** *Maybe the variable isn't being referenced in the code, or* **terraform plan** *wasn't re-run properly.*

Hint: *Did the variable actually connect to anything in your config?*

3. Ask: *"You changed the backend configuration, but Terraform throws an error during init. Why does this happen?"*

✅ **Expected:** *Backend config is locked after first init — you must reinitialize with* **terraform init -reconfigure.**

Hint: *What's the safe way to tell Terraform you changed how it stores state?*

4.* Ask: *"Imagine you've cloned someone else's Terraform project and* terraform plan *fails immediately. What's your first step?"

✅ **Expected:** ***Run*** **terraform init** ***to initialize providers and setup.***

Hint: ***What command prepares the environment for use?***

5.* Ask: *"Your Terraform script tries to delete an S3 bucket but fails. What could be the reason?"

✅ **Expected:** ***Terraform can't delete an S3 bucket unless it's empty. Manual cleanup or force-delete setup is needed.***

Hint: ***Did the bucket contain files during deletion?***

PROJECT-BASED INTERVIEW QUESTIONS

1.* Ask: *"If I ask you to provision a basic EC2 instance using Terraform, how would you go about it?"

✅ **Expected:** ***Use*** **aws_instance** ***with AMI ID and instance type in a*** **.tf** ***file, then run*** **init, plan,** ***and*** **apply.**

Hint: ***Think: one resource, three core commands.***

2.* Ask: *"I want you to create a simple S3 bucket using Terraform. What would your workflow look like?"

✅ **Expected:** ***Define*** **aws_s3_bucket,** ***provide a unique name and region, then apply the plan.***

Hint: ***What's the minimum info needed to make a bucket?***

3. **Ask:** ***"Let's say we need to allow HTTP and SSH traffic to a server. How would you create the security group for it in Terraform?"***

✅ **Expected:** ***Use*** **aws_security_group** ***with ingress rules for ports 22 and 80.***

Hint: ***Which block helps define allowed connections?***

4. **Ask:** ***"How would you structure your Terraform files for a small, single-module project?"***

✅ **Expected:** ***Use*** **main.tf** ***for resources,*** **variables.tf,** ***and*** **outputs.tf.** ***Use*** **terraform.tfvars** ***for values.***

Hint: ***How do you make your code readable and organized from day one?***

Medium - Level
Terraform
(DevOps Interview Questions - 1 to 2 Years Experience)

1. “What is the difference between terraform apply and terraform plan?”

✅ **Answer:**
terraform plan shows the changes Terraform will make without applying them.
terraform apply actually performs those changes on the infrastructure.

Hint: ***One is a dry run, the other makes it real.***

2. “What is the purpose of terraform state command?”

✅ **Answer:**
It lets you inspect, move, or remove items in the current Terraform state.
Useful for advanced debugging and resource tracking.

Hint: ***You're looking into or fixing the*** **.tfstate** ***file.***

3. “How do you manage multiple environments like dev, stage, and prod?”

✅ **Answer:**
Use Terraform workspaces, or separate directories/modules with

different variable files.
This ensures isolation and cleaner configuration.

Hint: ***Think reuse, but with separation.***

4. "What are Terraform modules and why are they useful?"

✅ Answer:
Modules are reusable components that help organize and simplify complex infrastructure.
They reduce duplication and enforce consistency.

Hint: ***Think of them like reusable functions in code.***

5. "How do you manage secrets securely in Terraform?"

✅ Answer:
Avoid hardcoding secrets; use environment variables, remote backends, or tools like Vault.
Sensitive values should never be stored in code.

Hint: ***Security 101: don't put passwords in*** **.tf** ***files.***

6. "What are data sources in Terraform?"

✅ Answer:
Data sources let Terraform fetch existing info from your provider (e.g., AMI IDs).
They're read-only and don't create resources.

Hint: ***You want to use data that already exists.***

7. “Difference between count and for_each?”

✅ **Answer:**
count is for simple replication using integers.
for_each gives more control using maps or sets with unique keys.

Hint: ***Want named resources tied to specific data?***

8. “How does remote state work?”

✅ **Answer:**
Remote state stores .tfstate files in services like S3 or Terraform Cloud.
It enables team collaboration and locking.

Hint: ***You need shared state across a team.***

9. “How to handle resource dependencies in Terraform?”

✅ **Answer:**
Terraform automatically detects dependencies using references.
You can manually specify them using depends_on.

Hint: ***Sometimes you need to control execution order.***

10. “What is the lifecycle block in Terraform?”

✅ **Answer:**
It lets you customize resource behavior, like forcing recreation or preventing deletion.
Useful for zero-downtime updates.

Hint: *Control how resources are destroyed or replaced.*

SCENARIO-BASED INTERVIEW QUESTIONS

1.* Ask: *"You're working in a team and both you and a teammate tried applying changes simultaneously. One apply failed with a lock error. Why?"

✅ **Expected:** *Terraform state was locked. Only one apply can modify the state at a time.*

Hint: *Think about how Terraform avoids conflicting writes.*

2.* Ask: *"You're using a remote backend with S3 and DynamoDB for state locking. What happens if someone deletes the DynamoDB table?"

✅ **Expected:** *State locking would fail, risking concurrent updates to the state.*

Hint: *What part protects your state from being changed at the same time?*

3.* Ask: *"You used* count *to create multiple instances, but now want to delete one without affecting others. What challenge will you face?"

✅ **Expected:** count *relies on index, so removing one causes shifting. Use* for_each *for better resource targeting.*

Hint: *Count is index-based. What's better for selective removal?*

4. Ask: *"You changed a module but the plan didn't show any updates. What might be the reason?"*

✅ **Expected: *You may have forgotten to run* terraform get -update *to refresh the module.***

Hint: ***What command ensures module changes are recognized?***

5. Ask: *"You're using* terraform import *to bring an existing AWS resource under Terraform control. What's the limitation here?"*

✅ **Expected: *It only imports state — no configuration. You must manually write the matching* .tf *code.***

Hint: ***Import pulls in state, not code.***

PROJECT-BASED INTERVIEW QUESTIONS

---

1. Ask: *"You've been asked to create a reusable Terraform module for provisioning EC2 instances. How would you structure it?"*

✅ **Expected: *Use* main.tf, variables.tf, *and* outputs.tf *inside a separate module folder. Then call it using* module *block in root.***

Hint: ***Think modular: input → reusable logic → output.***

2.* Ask: *"How would you design a Terraform workflow that deploys both a VPC and EC2 instance — but only if the VPC was created successfully?"

✅ **Expected:** ***Define the EC2 instance with a dependency on VPC outputs. Terraform builds the dependency tree automatically.***

Hint: ***Let Terraform handle the dependency chain using references.***

3.* Ask: *"You're provisioning infra for two environments: dev and prod. How would you structure the Terraform project to support both?"

✅ **Expected:** ***Use workspaces or maintain separate folders for* dev/ *and* prod/ *with shared modules.***

Hint: ***Environments need separation — by workspace or folder.***

4.* Ask: *"Suppose you need to deploy a public and a private subnet using Terraform. What AWS resource will help you route traffic properly?"

✅ **Expected:** ***Use route tables: one public (with internet gateway), one private (with NAT gateway if needed).***

Hint: ***Routing rules are key to network segregation.***

Hard - Level
Terraform
(DevOps Interview Questions - 3+ Years Experience)

1. "How do you handle state locking in Terraform?"

✅ Answer:
State locking prevents concurrent modifications to the state file.
It's automatically handled in backends like S3 with DynamoDB or Terraform Cloud.

Hint: ***What if two people run*** **apply** ***at the same time?***

2. "What are the benefits of using Terraform Cloud or Terraform Enterprise over local execution?"

✅ Answer:
They provide remote state storage, policy enforcement, RBAC, collaboration, and cost estimation.
Ideal for teams working at scale.

Hint: ***Think collaboration, security, governance, and audit trails.***

3. "How do you manage secrets in Terraform when using S3 backend with version control?"

✅ Answer:
Avoid committing secrets; use sensitive = true, environment variables, or reference secrets from external tools like Vault or SSM.
Never store secrets in .tf files or state.

Hint: *Even state files can leak secrets — what's your strategy?*

4. "What is Terraform drift detection and how do you handle it?"

✅ **Answer:**
Drift detection is identifying changes in infrastructure that were made outside Terraform.
You detect it using terraform plan and resolve by reconciling or reapplying.

Hint: *What if someone manually modified a resource in AWS?*

5. "How can you modularize Terraform code for large-scale environments?"

✅ **Answer:**
Use well-structured modules with clear inputs/outputs, follow DRY principles, and organize reusable logic per resource/domain.
Use terragrunt or mono-repos for additional control.

Hint: *You're managing 200+ resources — how do you stay sane?*

6. "What are dynamic blocks and when should you use them?"

✅ **Answer:**
Dynamic blocks generate nested blocks programmatically in resources (e.g., security groups).
Useful when the block count or content is variable.

Hint: *Need to create N similar nested blocks with loops?*

7. “Explain the difference between depends_on and implicit dependencies in Terraform.”

✅ **Answer:**
Implicit dependencies come from references; explicit ones are declared using depends_on.
Use depends_on when Terraform can’t infer order.

Hint: *Sometimes Terraform doesn’t know what should come first — help it out.*

8. “How do you migrate Terraform state between backends?”

✅ **Answer:**
Use terraform init -migrate-state to move state to a new backend.
Ensure backups and access before migration.

Hint: *You’re moving from local to S3 — how to avoid data loss?*

9. “What is the purpose of terraform import?”

✅ **Answer:**
terraform import maps existing resources to Terraform code without recreating them.
It helps bring unmanaged resources under control.

Hint: *Your infra already exists — now bring it under Terraform.*

10. “How do you handle breaking changes when upgrading Terraform provider versions?”

✅ **Answer:**
Read changelogs, use version constraints in required_providers, and test upgrades in isolated environments.
Use terraform state replace-provider if needed.

Hint: *You upgraded the AWS provider and nothing works — now what?*

SCENARIO-BASED INTERVIEW QUESTIONS

---

1. Ask: *"Your team reported that a recent Terraform apply caused service downtime. How do you prevent this in future runs?"*

✅ **Expected:** *Use* **lifecycle** *blocks like* **create_before_destroy**, *set* **depends_on**, *and use* **terraform plan** *to review changes before apply.*

Hint: *What lifecycle setting ensures zero downtime when replacing resources?*

2. Ask: *"How do you manage Terraform state in a team with multiple contributors working on different modules?"*

✅ **Expected:** *Split state using separate backends per module or use workspaces. Use remote backend with locking (e.g., S3 + DynamoDB).*

Hint: *Think: isolation, locking, and concurrent safety.*

3. Ask: *"You have multiple modules with dependencies. Terraform is trying to destroy a dependent resource first. How do you fix it?"*

✅ **Expected:** ***Use*** **depends_on** ***to explicitly control the destroy/create order.***

Hint: ***Implicit dependencies not enough? Time to be explicit.***

4. **Ask:** ***"You're upgrading a provider version and the plan shows unexpected changes. How do you handle provider upgrades safely?"***

✅ **Expected:** ***Read changelogs, pin versions with*** **required_providers,** ***and test in isolated branches or workspaces before merging.***

Hint: ***Upgrades ≠ safe by default. How do you stay protected?***

5. **Ask:** ***"You've been asked to detect manual infra changes made outside of Terraform. What's your strategy?"***

✅ **Expected:** ***Use*** **terraform plan** ***to detect drift or use tools like*** **terraform plan -detailed-exitcode** ***in CI/CD.***

Hint: ***Terraform can't fix what it doesn't track. Plan is your detective.***

PROJECT-BASED INTERVIEW QUESTIONS

---

1. **Ask:** ***"How would you design a Terraform solution to deploy infrastructure across multiple AWS accounts using a single repo?"***

✅ **Expected:** ***Use modules with account-specific variables, assume***

roles for each account, and structure directories per environment/account.

Hint: ***Think: cross-account auth + reusability + separation.***

2. Ask: *"You need to provision Kubernetes clusters across GCP and AWS using Terraform. How would you design this setup?"*

✅ **Expected:** ***Use separate providers and workspaces per cloud, with reusable modules per cluster type.***

Hint: ***Multi-cloud needs clean separation and provider switching.***

3. Ask: *"Suppose you want to automate destroying old test environments every 7 days. How would you build this with Terraform?"*

✅ **Expected:** ***Use a CI/CD pipeline with a script that checks resource age (via tags) and triggers*** **terraform destroy** ***for old workspaces.***

Hint: ***Tag it, script it, automate it.***

4. Ask: *"Your Terraform plan applies 100+ resources and takes too long. How would you optimize execution time?"*

✅ **Expected:** ***Use parallelism, break infra into smaller modules with isolated states, and avoid unnecessary refreshes.***

Hint: ***Smaller chunks = faster runs. Don't refresh what's untouched.***