200 Terraform Interview Questions (Easy to Hard Level)

I. Easy / Basic Questions (Foundational Knowledge)

1. What is Infrastructure as Code (IaC)?
2. What is Terraform?
3. What are the key benefits of using Terraform?
4. What is HCL (HashiCorp Configuration Language)?
5. What is a Terraform provider? Give some examples.
6. What is a Terraform resource?
7. What are the three core Terraform commands in its workflow?
8. Explain `terraform init`. What does it do?
9. Explain `terraform plan`. What is its purpose?
10. Explain `terraform apply`.
11. Explain `terraform destroy`.
12. What is a Terraform state file (`terraform.tfstate`)? Why is it important?
13. Where is the state file stored by default? What are the risks of this default?
14. What are Terraform variables? How do you define them?
15. What are Terraform outputs? Why are they useful?
16. How does Terraform handle dependencies between resources?
17. What is `terraform validate`?
18. What is `terraform fmt`? Why is it useful?
19. What is a Terraform module?
20. How do you declare a module in your Terraform configuration?

II. Intermediate Questions (Core Concepts & Common Use Cases)

21. What is a Terraform backend? Give examples of remote backends.
22. Why is using a remote backend crucial for team collaboration?
23. Explain state locking. Why is it important in a team environment?
24. How does Terraform handle sensitive data (e.g., passwords, API keys)?
25. What are Terraform workspaces? When would you use them?
26. Describe the difference between `count` and `for_each` meta-arguments. When would you use each?
27. What are data sources in Terraform? Provide a use case.
28. How do you import existing infrastructure into Terraform? What are its limitations?
29. What is the `depends_on` meta-argument? When do you need to use it explicitly?
30. What are Terraform locals? When are they useful?
31. Explain the `lifecycle` block in a resource. What are `prevent_destroy`, `create_before_destroy`, and `ignore_changes`?
32. What are Terraform provisioners? When should you use them, and why are they generally discouraged for production infrastructure?
33. Describe the difference between `local-exec` and `remote-exec` provisioners.

34. How do you manage different environments (dev, staging, prod) using Terraform? Compare workspaces vs. separate directories.
35. How do you pass variables to your Terraform configuration at runtime?
36. What is a null resource? Provide a practical example of its use.
37. How can you query information from the Terraform state file?
38. What is drift detection in Terraform, and how can you address it?
39. How do you upgrade Terraform versions? What about provider versions?
40. How would you structure a Terraform project for a medium-sized application with multiple services?

III. Advanced Questions (Complex Scenarios, Best Practices, & Troubleshooting)

41. **Scenario:** You have a complex application with hundreds of resources, and `terraform plan` is taking an excessively long time. How would you optimize this?
42. **Scenario:** A `terraform apply` fails halfway through. How do you recover from a failed apply? What steps would you take to ensure consistency?
43. **Scenario:** You accidentally deleted the `terraform.tfstate` file. What's your recovery plan?
44. **Scenario:** You need to refactor a large, monolithic Terraform configuration into smaller, reusable modules. Outline your strategy and the challenges you might face.
45. **Scenario:** How would you implement a blue/green deployment strategy for an application using Terraform? Detail the resources and logic involved.
46. **Scenario:** You need to manage infrastructure across multiple AWS accounts/regions. How would you structure your Terraform code to handle this effectively?
47. **Scenario:** Your team wants to enforce specific naming conventions or tagging policies for all resources created by Terraform. How would you implement this?
48. **Scenario:** How do you integrate Terraform into a CI/CD pipeline (e.g., Jenkins, GitLab CI, GitHub Actions, Azure DevOps)? Describe the key stages and commands.
49. **Scenario:** How would you ensure that your Terraform configurations are secure? Discuss best practices for secrets management, IAM, and network security within Terraform.
50. **Scenario:** What is Policy as Code? How can you implement it with Terraform (e.g., Sentinel, OPA)? Provide a concrete example.
51. What is Terragrunt? What problems does it solve that Terraform alone doesn't address?
52. How would you implement a dynamic resource creation where the number of resources depends on an external data source or a list of inputs?
53. Explain the concept of `backend` configuration in `main.tf` and how it relates to state storage.
54. How do you handle a situation where a resource needs to be replaced (destroyed and recreated) rather than updated in place?
55. What are explicit and implicit dependencies, and when might you need to use `depends_on`?
56. Discuss common strategies for managing resource lifecycle (e.g., preventing accidental deletion of production resources).
57. How do you manage provider configurations when dealing with multiple regions or multiple accounts for the same provider?

58. What is the significance of the `.terraform.lock.hcl` file?
59. How would you debug a complex Terraform configuration that is not behaving as expected? (e.g., `TF_LOG` environment variable, `terraform console`)
60. What are the advantages and disadvantages of using Terraform Cloud/Enterprise compared to open-source Terraform with remote backends?

IV. Expert & Scenario-Based Questions (Architecture, Troubleshooting, Optimization)

61. **Scenario:** You are migrating an existing, unmanaged infrastructure (created manually) to Terraform. The infrastructure is critical, and downtime must be minimized. Outline a detailed, step-by-step migration strategy.
62. **Scenario:** Your team is expanding, and multiple engineers will be working on the same Terraform code. How would you set up a collaborative environment, minimize conflicts, and ensure state integrity?
63. **Scenario:** You discover that a critical production resource managed by Terraform has been manually modified outside of Terraform (drift). How would you detect, resolve, and prevent this from happening again?
64. **Scenario:** Design a highly available and fault-tolerant Terraform setup for a large enterprise, including considerations for state storage, CI/CD, and team collaboration.
65. **Scenario:** You need to deploy an application that requires a specific network topology (e.g., hub-and-spoke with VPC peering/VPNs). How would you model this in Terraform?
66. **Scenario:** How would you automate the generation of Terraform configuration files (e.g., from a database of applications or a service catalog)?
67. **Scenario:** You are using a custom Terraform provider. Describe the high-level steps involved in developing such a provider.
68. **Scenario:** A `terraform destroy` operation fails or gets stuck. What steps would you take to manually clean up the resources and reconcile the state?
69. **Scenario:** You need to implement a disaster recovery strategy for your entire infrastructure managed by Terraform. How would Terraform contribute to or be part of this strategy?
70. **Scenario:** How do you handle circular dependencies in Terraform? Is it always possible? What are the common solutions or workarounds?
71. Discuss the benefits and drawbacks of using `Terraform CLI` vs. `Terraform Cloud` for state management and collaboration.
72. How would you manage and version control a large number of Terraform modules? What's the role of a module registry?
73. Explain how Terraform's resource graph works and how it determines the order of operations.
74. What is the `target` flag (`-target=resource.name`) in `terraform plan` and `terraform apply`? When should it be used, and what are its dangers?
75. Describe a scenario where you would use the `terraform state mv` command. What are the precautions?
76. What are the implications of directly editing the `terraform.tfstate` file? When might it be necessary, and what are the risks?

77. How do you implement robust error handling and conditional logic within your Terraform configurations?
78. How would you enforce a "dry run" or approval process for Terraform changes in a production environment?
79. What are remote_state data sources? Provide a practical use case.
80. How can you ensure that Terraform configurations are compliant with organizational security and cost policies before applying?

V. Conceptual & Strategic Questions (Beyond the Code)

81. What is the philosophy behind declarative vs. imperative IaC, and where does Terraform fit in?
82. Compare and contrast Terraform with other IaC tools like CloudFormation, Ansible, Puppet, or Chef. When would you choose Terraform over the others?
83. How does Terraform contribute to the "shift left" security paradigm in DevOps?
84. What are the challenges of managing multi-cloud infrastructure with Terraform, and how do you overcome them?
85. How do you measure the success and efficiency of your IaC implementation using Terraform? What metrics would you track?
86. What is the role of a "platform team" or "infrastructure team" in an organization heavily utilizing Terraform?
87. How do you keep your Terraform skills updated with the rapid pace of cloud provider changes and new Terraform features?
88. Discuss the importance of a clear module hierarchy and naming conventions for large-scale Terraform adoption.
89. How does Terraform fit into a broader GitOps strategy?
90. What are the common anti-patterns in Terraform code, and how do you avoid them?

VI. Specific HCL & Provider Feature Deep Dives (Advanced Syntax)

91. Explain dynamic blocks in HCL. Provide an example.
92. What are the try() and can() functions in HCL? When would you use them?
93. How do you use the lookup() function?
94. Explain the difference between map and object types in HCL.
95. How do you use the jsondecode() and jsonencode() functions?
96. How would you implement complex string manipulation or regex matching in HCL?
97. How do you use external data sources (e.g., external provider) to fetch data from custom scripts or APIs?
98. What is the purpose of terraform providers schema -json?
99. How do you work with secrets in tfvars files? (Hint: don't!)
100. How do you conditionally create resources or blocks within a resource based on a variable?

VII. More Advanced Scenarios & Troubleshooting

101. **Scenario:** You have a Terraform configuration that provisions a Kubernetes cluster. How would you then use Terraform to deploy applications *into* that Kubernetes cluster?
102. **Scenario:** You need to integrate Terraform with a custom internal API for provisioning specific resources not covered by existing providers. How would you approach this?
103. **Scenario:** Your Terraform `plan` output shows a large number of changes, some of which are unexpected or seem like "noise." How do you analyze and filter this output effectively?
104. **Scenario:** You are debugging a Terraform apply that fails with a generic API error from the cloud provider. What are your diagnostic steps?
105. **Scenario:** Your organization uses a private Git repository for Terraform modules. How do you configure Terraform to authenticate and fetch modules from this repository?
106. **Scenario:** You need to ensure that specific resources are always created or updated with zero downtime. How would you use `create_before_destroy` and other `lifecycle` arguments for this?
107. **Scenario:** Describe a robust backup and restore strategy for your Terraform state files, especially in a remote backend.
108. **Scenario:** How do you handle situations where a resource in your state file no longer exists in the cloud, but Terraform doesn't detect it? (e.g., manual deletion, `terraform state rm`)
109. **Scenario:** You need to run a post-deployment script on a newly provisioned VM. How would you use provisioners or alternative methods (e.g., cloud-init, configuration management tools) for this?
110. **Scenario:** How do you perform a "rolling update" of an instance group or other resource type using Terraform, minimizing downtime?

VIII. Security & Compliance Deep Dive

111. How do you ensure that Terraform itself is running with the principle of least privilege?
112. Discuss different approaches to managing cloud provider credentials for Terraform execution in a CI/CD pipeline. (e.g., OIDC, IAM roles, service principals, static keys).
113. How can you use `terraform validate` and static analysis tools (e.g., `tflint`, `checkov`, `terrascan`) to improve the security and quality of your Terraform code?
114. How would you prevent accidental `terraform destroy` operations in a production environment?
115. What are the considerations for storing and accessing remote state securely? (encryption at rest/in transit, access controls)
116. How do you manage secrets that need to be passed to modules or provisioners without exposing them in state files or logs?
117. What are the risks of using `local` state files in a team environment from a security perspective?
118. How can you use Terraform to enforce security groups/network ACLs that restrict ingress/egress only to necessary ports and IPs?

119. How would you use data sources to pull security group IDs or IAM role ARNs from a central security account?
120. How do you implement automated security scanning of your infrastructure *after* it's deployed by Terraform?

IX. More on Modules & Project Structure

121. What is the difference between a root module and a child module?
122. How do you handle module versioning and dependency management?
123. What are the best practices for designing reusable and composable Terraform modules?
124. How do you pass provider configurations to child modules?
125. When should you use a local module versus a remote module?
126. How do you manage module inputs and outputs effectively for complex modules?
127. Describe the "monorepo" vs. "multi-repo" approach for Terraform configurations. What are the pros and cons of each?
128. How would you structure a Terraform repository for a very large organization with many teams and applications?
129. What is a "module registry" and how does it benefit large organizations?
130. How do you test Terraform modules effectively? (e.g., `terratest`, kitchen-terraform)

X. Even More Advanced Scenarios & Niche Cases

131. **Scenario:** You need to conditionally provision a resource only if a specific feature flag is enabled in your application. How would you model this in Terraform?
132. **Scenario:** You have a custom configuration management system that needs to be triggered after Terraform provisions resources. How would you integrate this?
133. **Scenario:** How would you implement a "self-healing" infrastructure using Terraform and external monitoring?
134. **Scenario:** You need to manage a large number of identical environments for development teams. How would you use Terraform to efficiently provision and destroy these on demand?
135. **Scenario:** How do you handle state file migrations when changing the resource address or type in your Terraform code?
136. **Scenario:** You need to integrate Terraform with a service catalog or internal portal to allow non-DevOps users to provision pre-defined infrastructure patterns.
137. **Scenario:** How can you use `count` and `for_each` together in a nested structure?
138. **Scenario:** You've encountered `resource not found` errors during `terraform plan` for resources that actually exist. What could be the cause?
139. **Scenario:** How do you handle resource tags effectively across all your Terraform resources, ensuring consistency and compliance?
140. **Scenario:** You need to automate the cleanup of stale or unused resources managed by Terraform. How would you approach this?

XI. Error Handling, Debugging, and Best Practices (Deep Dive)

141. How do you define a `provider` block with multiple aliases? When is this useful?
142. Explain the purpose of `terraform graph` and how it can aid in understanding dependencies.
143. How do you write robust `variable` definitions, including `validation` rules and `description`?
144. What are implicit and explicit data conversions in HCL?
145. How do you use the `sensitive = true` attribute for output variables?
146. How do you handle `timeouts` in Terraform resources?
147. What is a "partial configuration" in Terraform?
148. How do you use `null_data_source`?
149. Explain the concept of `tainting` a resource. When would you use `terraform taint` and `terraform untaint`? Why is `terraform state replace-resource` often preferred?
150. How do you gracefully deprecate and remove old resources from your Terraform configurations without causing downtime?

XII. Cloud-Specific Integrations & Advanced Patterns

151. (AWS) How would you use IAM roles for Terraform execution instead of access keys?
152. (Azure) How do you manage Azure Service Principals and use them with Terraform?
153. (GCP) How do you configure Terraform to use Google Cloud service accounts?
154. How do you manage DNS records (e.g., Route 53, Azure DNS, Cloud DNS) using Terraform?
155. How do you provision and manage Kubernetes objects (deployments, services) using Terraform's Kubernetes provider?
156. How do you use Terraform to manage secrets in external secret managers like AWS Secrets Manager, Azure Key Vault, or HashiCorp Vault?
157. How do you implement auto-scaling groups or instance groups with Terraform?
158. How do you manage storage solutions (e.g., S3 buckets, Azure Blob Storage, GCS buckets) with specific access policies using Terraform?
159. How do you handle private networking and VPN/Direct Connect connections using Terraform?
160. How do you use Terraform to provision and configure serverless functions (e.g., AWS Lambda, Azure Functions, GCP Cloud Functions)?

XIII. Terraform Enterprise / Cloud Specifics

161. What are the key features of Terraform Cloud / Enterprise?
162. How do run environments work in Terraform Cloud?
163. How do you manage organizations, workspaces, and teams in Terraform Cloud?
164. What is the Private Module Registry in Terraform Cloud?
165. How does Sentinel (Policy as Code) integrate with Terraform Enterprise/Cloud?
166. How does remote operations execution (`terraform apply` in Terraform Cloud) work?
167. What are the benefits of VCS-driven workflows in Terraform Cloud?
168. How do you integrate Terraform Cloud with external CI/CD tools?

169. How does Cost Estimation work in Terraform Cloud?
170. What are the advantages of using Terraform Enterprise for on-premise deployments?

XIV. Scenario-Based Design & Problem Solving

171. **Scenario:** Design a Terraform configuration for deploying a three-tier application (web, app, db) to a public cloud, ensuring high availability, scalability, and network isolation between tiers.
172. **Scenario:** Your team needs to manage a highly sensitive data analytics platform. How would you ensure data security, encryption, and access control are strictly enforced by Terraform?
173. **Scenario:** You're asked to set up a new sandbox environment for developers that can be spun up and down on demand. How would you automate this with Terraform, including cost optimization?
174. **Scenario:** Your organization requires audit trails for all infrastructure changes. How would you configure Terraform and its environment to log every `plan` and `apply` operation, including who performed it?
175. **Scenario:** You need to perform a major refactoring of your cloud networking using Terraform, which will involve changing many existing resource IDs. How would you plan and execute this with minimal service disruption?
176. **Scenario:** A new compliance requirement dictates that all S3 buckets must have encryption enabled and public access blocked. How would you enforce this using Terraform and policy as code?
177. **Scenario:** How would you design a Terraform configuration that can provision the *same* application infrastructure in multiple distinct customer environments, each with unique configurations?
178. **Scenario:** You are using Terraform to manage a large number of EC2 instances, and you need to apply a new security patch to all of them using a rolling update strategy. How would Terraform facilitate this?
179. **Scenario:** Describe a situation where using `data` sources is superior to using `resource` blocks to achieve a desired outcome.
180. **Scenario:** You need to integrate Terraform with a CMDB (Configuration Management Database) to automatically update resource information after provisioning. How would you achieve this?

XV. Miscellaneous & Advanced Concepts

181. What is the role of `.terraformignore`?
182. How do you use the `jsonnet` preprocessor with Terraform?
183. What is the `count.index` and `each.key`/`each.value`?
184. Explain the difference between `terraform console` and `terraform output`.
185. How do you manage external DNS records (e.g., domain registration) with Terraform?
186. What are the `depends_on` anti-patterns?
187. What is `terraform-exec`?

188. How do you write tests for your Terraform configurations? (e.g., unit tests, integration tests)
189. What is the concept of immutability in the context of Infrastructure as Code? How does Terraform support it?
190. What is `terraform validate` used for, and how does it differ from `terraform plan`?

XVI. Advanced Troubleshooting & Edge Cases

191. **Scenario:** A Terraform `plan` shows a resource will be `tainted` or `replaced`, but you expected an in-place update. How would you investigate why?
192. **Scenario:** You're working on a multi-cloud project, and one of the cloud providers is experiencing API rate limiting issues, causing Terraform to fail. How would you configure Terraform to be more resilient?
193. **Scenario:** You have a `data` source that fetches information about an existing resource, but that resource doesn't exist. How do you handle this gracefully without failing the `plan`?
194. **Scenario:** Your `terraform init` fails due to provider checksum mismatch. How do you resolve this?
195. **Scenario:** How do you recover from a `terraform state rm` that was performed incorrectly, removing a valid resource from state?
196. **Scenario:** Your Terraform apply is stuck or hanging, but not showing any errors. What are your diagnostic steps?
197. **Scenario:** You encounter an `InvalidParameterValue` or similar API-specific error. How do you efficiently debug these?
198. **Scenario:** You have a custom `provider` that is crashing during `terraform apply`. How would you approach debugging the provider itself?
199. **Scenario:** How do you handle the deprecation of a resource argument or an entire resource in a provider, especially for existing infrastructure?
200. **Scenario:** Your Terraform pipeline fails intermittently with network connectivity errors to the remote backend. What are the potential causes and solutions?

NAVNEET YADAV