Terraform Interview Questions & Answers (3 Years Experience)

1. What is Terraform?

Terraform is an open-source Infrastructure as Code (IaC) tool by HashiCorp used to provision, manage, and version infrastructure across various cloud providers.

2. What is Infrastructure as Code (IaC)?

IaC means managing and provisioning infrastructure using code, allowing automation, version control, and repeatability.

3. Which language is used in Terraform?

HashiCorp Configuration Language (HCL), which is declarative and human-readable.

4. What are Terraform providers?

Providers are plugins that allow Terraform to interact with APIs of cloud platforms like AWS, Azure, GCP, etc.

5. What is a Terraform module?

A module is a container for multiple resources that can be reused. It helps in organizing code and improving reusability.

6. What is the Terraform state file?

`terraform.tfstate` keeps track of the current state of the infrastructure. It's critical for change tracking and updates.

7. What is the purpose of `terraform init`?

It initializes the working directory, downloads the provider plugins, and sets up the backend.

8. What does `terraform plan` do?

It shows what actions Terraform will perform to reach the desired state, without making any actual changes.

9. What does `terraform apply` do?

It applies the planned changes and updates the infrastructure as defined in the code.

10. What is `terraform destroy`?

It removes all the infrastructure managed by Terraform.

11. What is the difference between `terraform plan` and `apply`?

`plan` shows what will be changed; `apply` actually makes the changes.

12. How do you manage sensitive data in Terraform?

By using `terraform.tfvars`, environment variables, or tools like Vault. Never hard-code sensitive values in `.tf` files.

13. What is a backend in Terraform?

A backend determines how state is loaded and how operations are performed. Example: local, S3, Azure Blob, etc.

14. What is remote state?

Storing the state file in a remote backend like AWS S3 to share it across teams and enable collaboration.

15. What are data sources in Terraform?

Data sources allow Terraform to fetch read-only information from providers to be used in configurations.

16. What is `terraform refresh`?

Updates the state file to match the real infrastructure without changing it.

17. How do you use Terraform in a team?

Use remote backends with state locking, version control for `.tf` files, and create reusable modules.

18. How do you avoid conflicts in Terraform state?

By enabling state locking using backends like S3 with DynamoDB, or Terraform Cloud.

19. What are workspaces in Terraform?

Workspaces allow you to use the same code for multiple environments (e.g., dev, prod) with different states.

20. Have you used Terraform with AWS/Azure/GCP? Give an example.

Yes, I used Terraform to provision EC2 instances, S3 buckets, and IAM roles in AWS.

21. How do you manage different environments in Terraform?

By using workspaces, environment-specific variable files, or directory structures like `dev`, `prod`.

22. How do you troubleshoot Terraform errors?

By using verbose mode (`TF_LOG=DEBUG`), checking syntax, verifying provider versions, and ensuring state consistency.

23. How do you upgrade Terraform modules or providers?

By updating the version in `.tf` files and running `terraform init -upgrade`.

24. What is your directory structure in a typical Terraform project?

Common structure: main.tf, variables.tf, outputs.tf, terraform.tfvars, modules/, environments/.

25. How do you handle Terraform drift?

By running `terraform plan` regularly to identify and reconcile differences.

26. What is state locking in Terraform and why is it important?

It prevents concurrent runs of Terraform that could corrupt the state file.

27. How do you manage Terraform state securely?

Use remote backends (like S3) with encryption and controlled access.

28. What happens if the Terraform apply is interrupted?

Terraform may leave resources in partial state. Re-run apply or fix the state manually.

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

`count` is for indexed identical resources, `for_each` is for map/set-based resource creation.

30. Can you explain lifecycle rules in Terraform?

`create_before_destroy`, `prevent_destroy` control resource update and delete behavior.

31. How do you ensure idempotency in Terraform?

By writing declarative code and keeping state consistent.

32. How do you handle provider versioning?

Lock versions in `required_providers` block.

33. How do you reuse Terraform code across projects?

Using modules stored locally or in VCS repositories.

34. Do you test your Terraform code before applying?

Yes, using `terraform validate`, `plan`, `fmt`, and tools like `checkov` or `terratest`.

35. How do you integrate Terraform into a CI/CD pipeline?

Steps: validate plan approval apply with proper credentials.

36. Have you faced any Terraform errors in production? How did you resolve them?

Yes. Example: state conflict, resolved using state lock and manual corrections.

37. What if someone manually changes infrastructure outside of Terraform?

Terraform plan will show drift; either import or overwrite.

38. What happens when you delete a resource from the `.tf` file?

Terraform will destroy it during apply.

39. Can Terraform manage resources in multiple clouds at once?

Yes, by configuring multiple providers.

40. Have you used Terraform Cloud or Terraform Enterprise?

Yes, for remote state, Sentinel policies, and VCS-based workflows.