Interview Questions:

1. What is Terraform, and how does it differ from other infrastructure as code tools?

- Terraform is an open-source infrastructure as code tool used to provision and manage cloud resources.
- Unlike other tools, Terraform is cloud agnostic and supports multiple cloud providers, making it flexible and suitable for hybrid cloud environments.

2. What are the main differences between Terraform and Ansible?

- Terraform is a declarative tool used for infrastructure provisioning, while Ansible is an imperative tool used for configuration management.
- Terraform focuses on creating and managing resources in cloud providers, whereas Ansible focuses on automating the configuration of servers and applications.

3. Explain the architecture of Terraform.

- Terraform follows a client-server architecture. The client, which runs on the user's machine, interprets Terraform configurations.
- It communicates with cloud providers' APIs to create and manage resources. The state file, stored locally or remotely, maintains the state of the infrastructure.

4. What is a Terraform configuration file, and what does it contain?

- A Terraform configuration file (usually named main.tf) is written in HashiCorp Configuration Language (HCL).
- It contains resource blocks that define the infrastructure resources to be created or managed, along with their configurations.

5. List some essential Terraform commands and explain their purposes.

- Common Terraform commands include init (initializes the working directory), plan (generates an execution plan), apply (creates or modifies resources), destroy (destroys resources), and output (displays output values).

6. How does Terraform manage resources and handle changes?

- Terraform uses the state file to keep track of the resources it manages.
- When changes are made to the configuration, Terraform compares the desired state to the current state and determines the actions required to achieve the desired state.

7. Describe an end-to-end Terraform project.

- An end-to-end Terraform project includes defining infrastructure as code, configuring providers, creating resources (e.g., EC2 instances, VPCs), managing dependencies, handling variables, and implementing state management.
- It covers the entire process from resource provisioning to destruction.

8. Can Terraform work with multiple cloud providers? How does it achieve this?

- Yes, Terraform is cloud agnostic and can work with multiple cloud providers like AWS, Azure, Google Cloud, etc.
- Terraform abstracts the interactions with cloud APIs using provider plugins, making it easy to manage resources across different providers.

9. How do Terraform variables and modules contribute to code reusability and maintainability?

- Terraform variables allow parameterization of configurations, making them reusable across different environments.
- Modules help package sets of resources and configurations into a reusable component, enabling easier management and sharing of complex infrastructure setups.

10. Explain the Terraform ecosystem and its role in extending Terraform's functionality.

- The Terraform ecosystem includes various third-party plugins, providers, and modules contributed by the community.
- These extensions expand Terraform's capabilities to manage additional resources, support new cloud providers, and enhance overall functionality.