Terraform Overview

Terraform is an open-source Infrastructure as Code (IaC) tool developed by HashiCorp. It enables users to define, provision, and manage infrastructure using a declarative configuration language. Terraform supports multiple cloud providers such as AWS, Azure, and Google Cloud, as well as on-premises environments.

Key Features:
1. **Declarative Syntax:** Define what you want to build instead of how to build it.
2. **Multi-Cloud Support:** Manage resources across various providers.
3. **Execution Plans:** Preview changes before applying them.
4. **Resource Graph:** Automatically determines resource dependencies.
5. **State Management:** Maintains infrastructure state for consistent deployments.

Terraform Architecture:
Terraform's core components include:
- **Configuration Files:** Written in HashiCorp Configuration Language (HCL) to describe resources.
- **Provider:** A plugin that enables Terraform to interact with APIs of cloud or service providers.
- **State File:** A JSON file that records the current state of the managed infrastructure.
- **Terraform CLI:** The command-line tool used for operations such as init, plan, apply, and destroy.

Common Terraform Commands:
- **terraform init:** Initialize a new or existing Terraform configuration.
- **terraform plan:** Show what actions Terraform will take to reach the desired state.
- **terraform apply:** Apply the planned changes to create or modify resources.
- **terraform destroy:** Remove all managed resources.

Best Practices:
1. Use remote backends for state storage (e.g., S3, Azure Blob).
2. Implement version control for Terraform configurations.
3. Use modules for reusable code.
4. Apply workspace separation for different environments.
5. Review execution plans before applying.

Sample Workflow:
1. Write configuration files using HCL.
2. Run *terraform init* to initialize providers.
3. Execute *terraform plan* to review changes.
4. Apply changes with *terraform apply*.
5. Destroy resources if needed using *terraform destroy*.

Terraform simplifies infrastructure provisioning and management through automation and standardization. By treating infrastructure as code, it enables consistency, repeatability, and better collaboration across teams.