 Introduction to Terraform

 Installation and Setup

 Terraform Configuration Language (HCL)

 Terraform CLI Commands

 Providers

 Resources

 Data Sources

 Variables and Outputs

 Terraform State

 Modules

 Provisioners

 Workspaces

 Remote Backends

17-12-2023:

Introduction to Terraform

**What does Terraform do?**

1. **Infrastructure as Code (IaC):** Terraform allows you to define your infrastructure in code using a declarative language called HashiCorp Configuration Language (HCL) or JSON. This code describes the desired state of your infrastructure.
2. **Multi-Cloud Support:** It supports various cloud providers like AWS, Azure, Google Cloud Platform, and many others, enabling you to manage diverse resources within these environments.
3. **Resource Graph:** Terraform builds a dependency graph of resources, allowing it to determine the order of provisioning and manage the entire infrastructure as a cohesive unit.

**How does Terraform work?**

1. **Configuration:** You write Terraform configurations defining the resources you need (servers, networks, databases, etc.) and their settings in .tf files.
2. **Initialization:** You initialize a Terraform project in a directory using **terraform init**. This downloads necessary plugins and sets up the working directory.
3. **Planning:** Run **terraform plan** to see what changes Terraform will apply before actually making them. It outlines the execution plan based on your configuration.
4. **Execution:** Apply the changes using **terraform apply**. Terraform then creates, modifies, or deletes resources to match the desired state defined in your configuration.
5. **Monitoring and Adjusting:** Terraform tracks the state of your infrastructure in a state file to manage and update resources. You can modify your configurations and reapply changes as needed.

**Key Benefits:**

* **Consistency:** Terraform ensures infrastructure is provisioned consistently every time the configuration is applied.
* **Versioning and Collaboration:** Configuration files can be version-controlled, facilitating collaboration and providing a history of changes.
* **Scalability:** It allows for managing both small-scale and complex infrastructure setups with ease.

**Best Practices:**

* **Modularity:** Organize configurations into reusable modules for better manageability.
* **Version Control:** Store configurations in version control systems like Git.
* **State Management:** Understand and manage Terraform state files properly to avoid issues in a team environment.