1. **Basic Concept**

**Terraform** is an open-source infrastructure-as-code (IaC) tool that allows you to define, provision, and manage cloud resources using configuration files written in a high-level language called **HCL** (HashiCorp Configuration Language). With Terraform, you can automate the process of creating and managing infrastructure across multiple cloud providers (like AWS, Azure, Google Cloud) or on-premises environments. It allows you to describe your infrastructure as code, version it, and apply changes consistently and reliably. Terraform creates an execution plan to show what changes will be made to your infrastructure, applies the changes, and keeps track of the current state of your resources. It supports resource management, dependency handling, and the ability to roll back changes, making infrastructure management easier, more efficient, and repeatable.

A purple rectangular object with white text

Description automatically generatedConfiguration files and execution plan are the key components of terraform to manage and execute the infrastructure

1. **HCL(Terraform Configuration Language)**

**Terraform Configuration Language (HCL)** is a declarative language used to define the infrastructure resources that Terraform will manage. It allows users to describe the desired state of resources (such as virtual machines, storage, networks, etc.) using a simple and human-readable syntax.

The key components of HCL include **resources**, which define the infrastructure components; **variables**, which allow dynamic values to be passed into configurations; **outputs**, which provide information about the infrastructure once it is created or modified; and **data sources**, which fetch information from existing resources outside of Terraform's control.

HCL also supports **modules**, which are reusable blocks of configuration that help organize complex setups. The configuration files are typically stored with a .tf extension and are processed by Terraform to create or modify resources according to the defined plan. The language is designed for simplicity and readability, with a focus on declaratively specifying what infrastructure should look like rather than how to create it.

1. **Terms used in Terraform**

* **Provider**: A provider is a plugin that Terraform uses to interact with APIs of cloud platforms (like AWS, Azure, or Google Cloud) or other services. Providers define the resources that Terraform can manage, such as virtual machines, databases, and networking components.
* **Resource**: A resource represents an infrastructure component in Terraform (e.g., an EC2 instance, a storage bucket, or a load balancer). Resources are defined in Terraform configurations and are created, updated, or destroyed based on the desired state.
* **Configuration**: This is a set of .tf files where you define the infrastructure you want to create or manage. It consists of resources, variables, outputs, and other settings written in **HCL (HashiCorp Configuration Language)**.
* **State**: Terraform keeps track of the current state of your infrastructure in a state file (terraform.tfstate). This file contains the mapping between the real-world infrastructure and the configuration written in your .tf files. Terraform uses this file to compare the current state of your resources with the desired state.
* **Plan**: A Terraform plan is a preview of the changes Terraform will make to your infrastructure. Running terraform plan shows what actions Terraform will take to achieve the desired state, such as creating, updating, or deleting resources.
* **Apply**: After reviewing the plan, you run terraform apply to make the changes to your infrastructure. This command updates your resources according to the configuration and plan.
* **Variables**: Variables are used to make configurations dynamic. They allow you to pass different values (such as instance sizes, region names, etc.) into your configuration without hardcoding them. Variables are defined in .tf files or can be provided at runtime.
* **Outputs**: Outputs are values that Terraform generates and displays after applying a configuration. These can be used to provide useful information about the infrastructure, like IP addresses or instance IDs, and can be referenced in other parts of Terraform or scripts.
* **Module**: A module is a container for multiple resources that are used together. Modules help organize and reuse configurations. You can use modules to simplify complex infrastructure setups or share configurations across different projects.
* **Data Source**: A data source allows you to fetch information about existing infrastructure resources, either from the provider or external sources. This is useful when you need to reference a resource that is already created but not managed by Terraform.
* **Backend**: A backend is where Terraform stores the state file. By default, Terraform stores the state file locally, but you can configure it to store the state remotely (e.g., in AWS S3 or Terraform Cloud) for team collaboration and enhanced security.
* **Provisioner**: Provisioners allow you to execute scripts or commands on your infrastructure after it is created or modified. They are useful for tasks like setting up software or running post-deployment configurations.
* **Terraform Registry**: The Terraform Registry is a repository where providers, modules, and other reusable Terraform code can be shared. It contains pre-built modules for commonly used services that you can import into your configuration.

1. A screenshot of a computer program

   Description automatically generated**Create a instance using terraform**

Access key and secret key should be of iam user of terraform which have admin access.

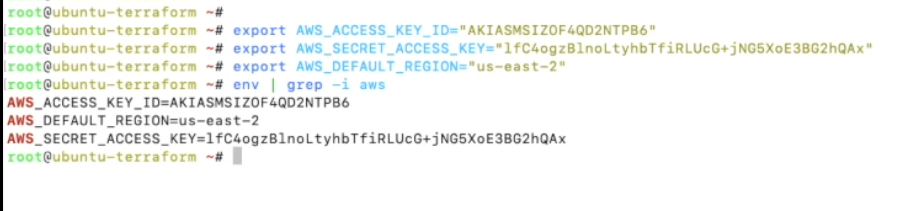
Ami can be found in the google Amazon ami locator

Terraform init: whenever you need to create a new directory or new file

After that : terraform apply

This will create the machine on your aw . to delete the machinw you need command: terraform destroy

Terraform plan: : A Terraform plan is a preview of the changes Terraform will make to your infrastructure.

1. **Provide credentials in environment variables**
2. **A screenshot of a computer

   Description automatically generatedCreate multiple instances**