Terraform Plugin Architecture

ow

Overview

- Terraform's plugin architecture allows it to interact with various infrastructure platforms and providers. Plugins enable Terraform to communicate with specific APIs and services, abstracting away the complexities of managing infrastructure resources.
  - Provider Plugins
  - Provisioner Plugins
  - State Backend Plugins
  - External Data Source Plugins

..

Plugin Type	Description
Provider	A Terraform provider plugin is responsible for managing and interacting with a specific service or infrastructure provider, such as AWS, Azure, or Kubernetes. Providers handle resource provisioning, state management, and CRUD operations. Multiple providers can be used within a Terraform configuration.
Provisioner	A Terraform provisioner plugin is used to execute scripts or commands on a resource after it has been created or destroyed. Provisioners are typically used for tasks like bootstrapping software, running configuration management tools, or initializing databases.
Backend	A Terraform backend plugin is responsible for storing and retrieving the Terraform state file. Backends define where the state is stored, allowing multiple team members to collaborate and share the same state. Examples of backends include local file storage, remote storage (like AWS S3 or HashiCorp Consul), or Terraform Cloud.
State	A Terraform state plugin is used to read, write, and manipulate the Terraform state file. State plugins are used internally by Terraform to manage the state data and handle operations like locking and versioning.

Provider Plugins

- Purpose: Provider plugins enable Terraform to interact with different infrastructure platforms (e.g., AWS, Azure, GCP). Each provider plugin implements the necessary functionality to manage resources on a specific platform.
- Example: Suppose you want to provision AWS EC2 instances using Terraform. To achieve this, you need the AWS provider plugin. You would configure the AWS provider in your Terraform files, and when you run terraform plan or terraform apply, Terraform uses the AWS provider plugin to communicate with the AWS API, create, modify, or delete EC2 instances.

Provisioner Plugins

- Purpose: Provisioner plugins allow Terraform to execute additional actions on resources during the provisioning process. These actions can include running scripts, executing commands, or performing configuration tasks on the created resources.
- Example: Let's say you want to install specific software on an EC2 instance after provisioning it with Terraform. You can use a provisioner plugin, such as the "remote-exec" provisioner, to execute shell commands on the EC2 instance once it's created. This plugin allows you to run scripts or commands remotely on the provisioned resource.

State Backend Plugins

- Purpose: State backend plugins handle the storage and retrieval of Terraform state. They define where the state file is stored and how to access it. Different backend plugins support various storage options like local file system, remote storage systems (e.g., Amazon S3, Azure Blob Storage), or remote state management services (e.g., Terraform Cloud, HashiCorp Consul).
- Example: If you want to store your Terraform state in an S3 bucket, you would use the S3 backend plugin. This plugin configures Terraform to store and retrieve the state file from the specified S3 bucket. When you run Terraform commands, the S3 backend plugin handles the interaction with the S3 API.

External Data Source Plugins

- Purpose: External data source plugins allow Terraform to fetch data from external sources and use it within the Terraform configuration. These plugins retrieve data from APIs, databases, or other external systems and provide the fetched data as input to Terraform resources.
- Example: Imagine you need to fetch information about existing AWS security groups to be used within your Terraform configuration. You can use an external data source plugin, such as the AWS Security Group data source plugin. This plugin communicates with the AWS API, retrieves the desired security group details, and exposes them as data that can be used in your Terraform resources.