TERRAFORM AND PROVIDERS

TERRAFORM

A Terraform provider is a translator that helps Terraform talk to different systems, like a cloud service, an API, or a database. When you tell Terraform what you want to build or manage (e.g., a virtual machine), the provider figures out how to make that happen in the real world.

INFRASTRUCTURE AS CODE

Terraform is Infrastructure as Code, allowing you to write, manage, and version control your infrastructure just like you would with application code.

The declarative approach in Infrastructure as Code (IaC), like Terraform, focuses on defining the desired end state of infrastructure, making it simpler and easier to manage. It ensures idempotency, meaning configurations can be safely reapplied without unintended changes, reducing errors and manual effort. The clear, human-readable format improves understanding and maintenance

INFRASTRUCTURE AS CODE - CONTINUED

Declarative IaC also enhances consistency and scalability. Configurations can be versioned in tools like Git, enabling tracking, reviews, and reusability across environments. Automated change detection and planning provide visibility into updates, ensuring safer and more predictable infrastructure management.

TERRAFORM PROVIDER

A Terraform provider is like a translator that helps Terraform talk to different systems, like a cloud service, an API, or a database. When you tell Terraform what you want to build or manage (e.g., a virtual machine), the provider figures out how to make that happen in the real world.

UNIVERSAL REMOTE

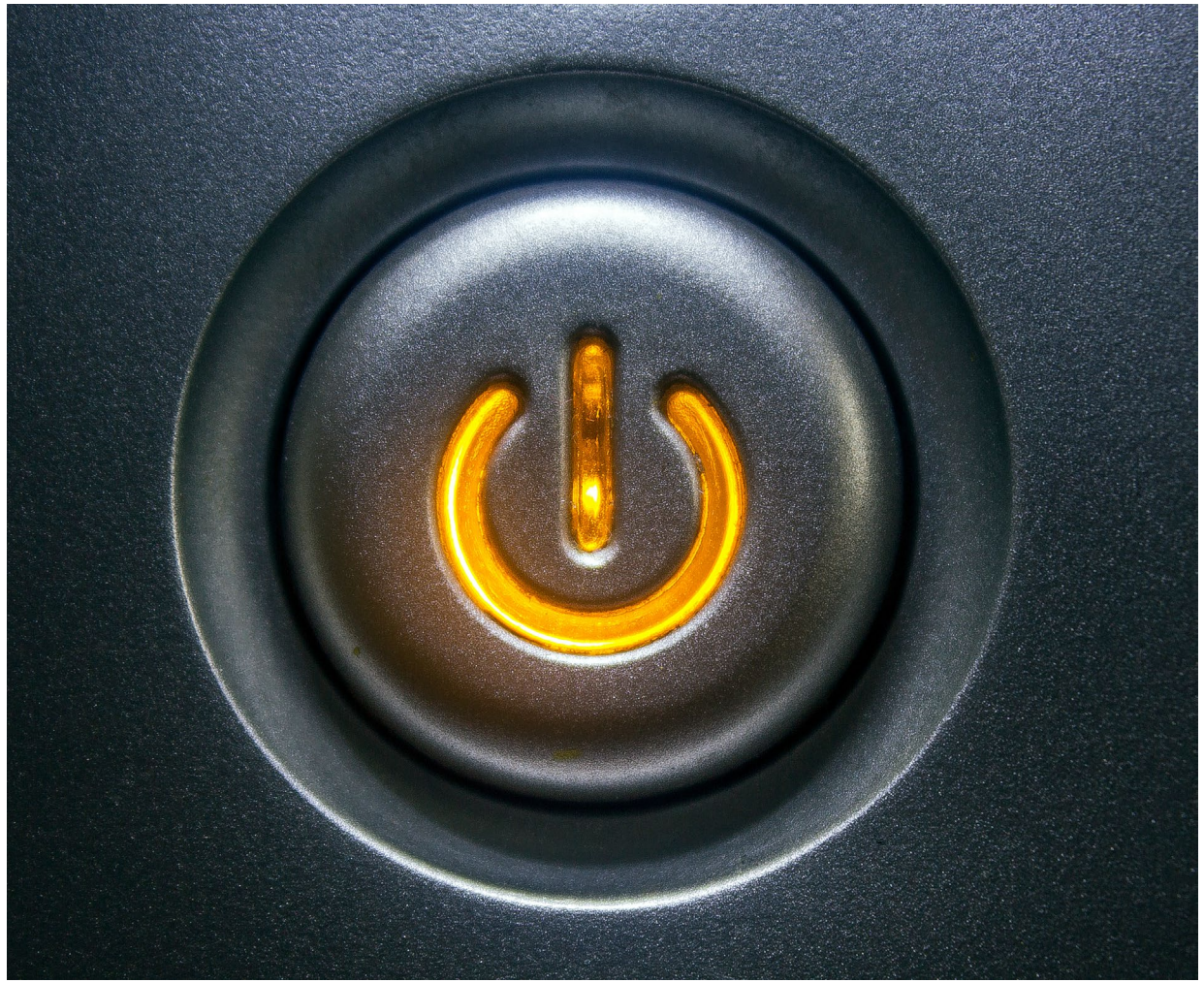
Terraform is equivalent to a universal remote control, where each television has a unique Terraform provider. The universal remote promotes a consistent standard that each television can rely upon, despite being different. For Terraform, this standard is the Terraform Plugin Framework, released in 2024.

GOLANG

With Golang, you have a toolbox for creating custom providers. Here are the required files:

- Main.go
- Provider.go
- Main.tf
- Terraform.rc

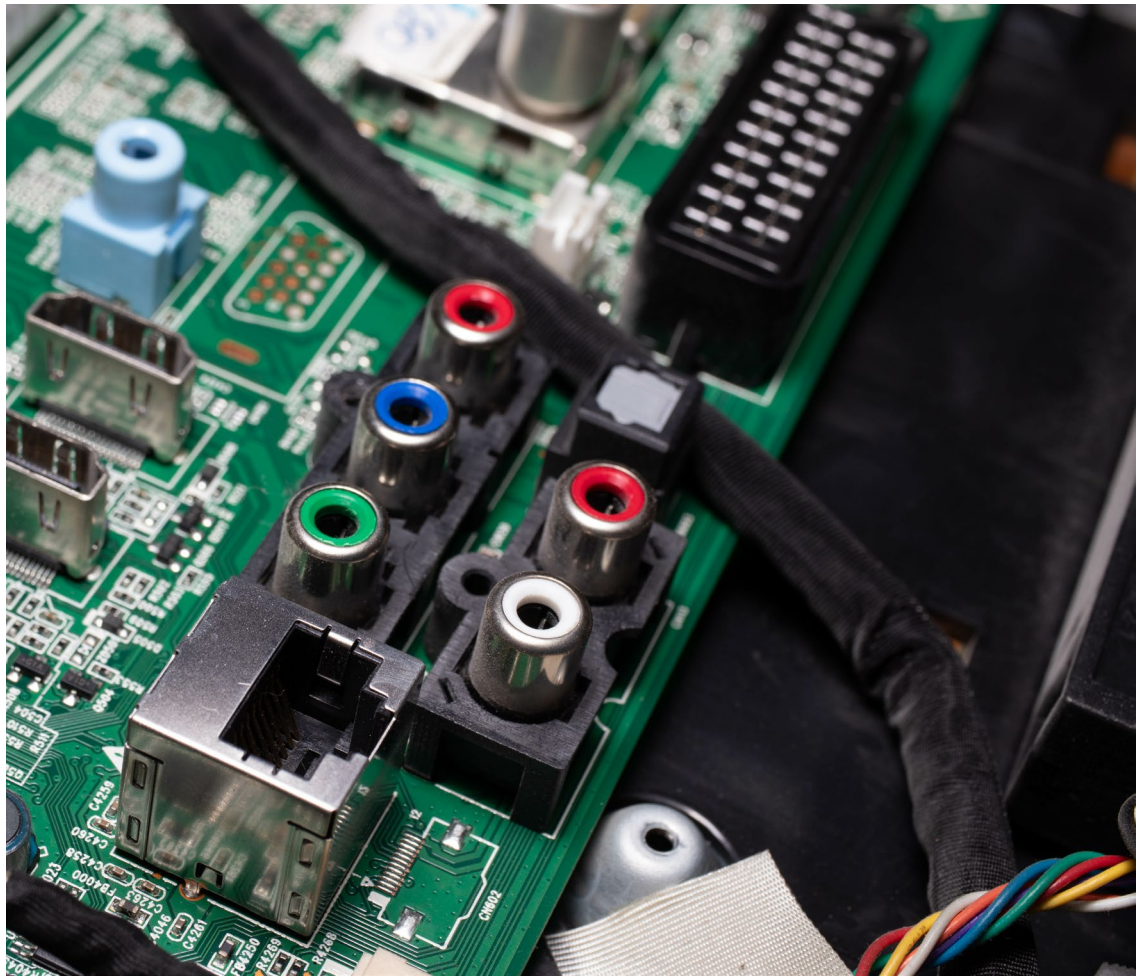
MAIN.GO



The main.go source file is the on / off switch that begins the conversation between the remote *widget*, whatever it is, and the custom provider.

It is responsible for creating an instance of the provider and passing configuration information to it, if necessary.

PROVIDER.GO



The provider.go source file implements the standard protocol (like an infrared receiver in the TV) for a remote control. In a television, the infrared receiver processes signals from the remote and tells the TV what to do.

Responsibilities include:

- Decodes Signals – Converts infrared pulses into electrical signals.
- Sends Commands – Passes instructions to the TV's control system.
- Confirms Execution – Ensures the TV responds correctly.

MAIN.TF

The main.tf configuration file controls the button presses on the remote. It represents the actions the user wants to perform.

Main.tf defines the desired state of infrastructure in Terraform. It specifies providers, resources, and settings that Terraform uses to create, manage, and destroy infrastructure.

RESOURCE FILES

resource_<name>.go – Defines Terraform resources and their CRUD operations : Create, Read, Update, and Delete. It manages infrastructure objects (e.g., databases, VMs, etc.).

datasource_<name>.go – Defines Terraform data sources, which retrieve existing infrastructure details without modifying them. It allows Terraform to read external resources, such as fetching customer data from a database.

TERRAFORM CUSTOMER PROVIDERS