Certified Tech Developer
The Ultimate Degree

Infraestructura II

Terraform: Nos bastidores

Como já vimos, a função principal da Terraform é a criação, alteração e remoção de recursos de infraestrutura.

Mas... Como este componente trabalha realmente? Como se comunica com nosso fornecedor da Cloud? Como é configurado?

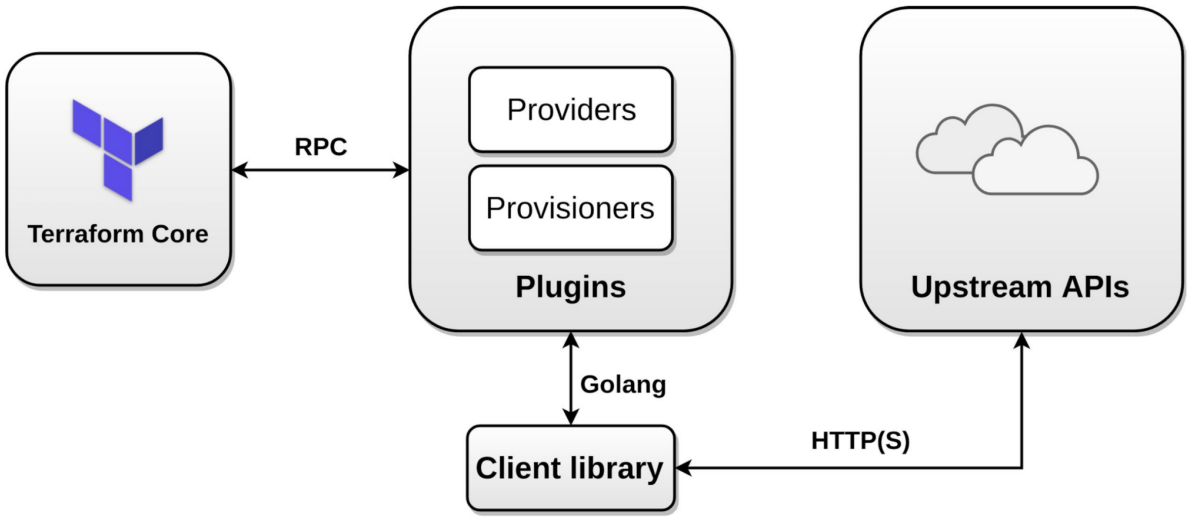
Vamos descobrir!

Arquitetura

O núcleo da TF é composto por várias partes móveis que:

1. Fornecem uma camada de abstração por cima da API subjacente.
2. São responsáveis pela interpretação das interações da API, e pela exibição de recursos.
3. Suportam múltiplas instâncias de fornecedores cloud.

Se fosse possível fazer uma radiografia deste componente e da forma como gerencia o fluxo de dados, iriamos ver algo parecido com:



Vamos dar uma olhada em cada um destes elementos e, após isso, tentaremos compreender como ele se comunica com nosso fornecedor da cloud.

Plugins

É uma aplicação complementar, normalmente pequena, que serve para adicionar uma funcionalidade extra ou adicional (muito específica) a algo já existente. Os plugins utilizados dividem-se em: Providers e Provisioners.

Providers

Um provider é um plugin "específico" que permitirá ao nosso fornecedor cloud compreender o idioma no qual iremos falar. Por exemplo, para dizer para ele que queremos dispor de um servidor novo.
O uso do termo "específico" refere-se ao fato de que existem vários providers, por exemplo: um provider para AWS, outro para GCP, para Azure, Kubernetes, etc. Aqui podemos visualizar a lista completa.

Como ela se comunica com a Nuvem?

Do lado da nuvem, existe uma API especialmente projetada para saber interpretar os comandos provenientes do nosso computador. Ou seja, está atenta às nossas requisições.
Se o "provider" não existir, não haverá comunicação entre ambas as partes.
Ao executar, por exemplo, o comando “terraform plan”, este binário irá procurar o “Provider” que definimos no nosso módulo da terraform:

```
# ================================================================
# Declaramos o Cloud Provider com o qual queremos trabalhar
terraform {
# Dizemos o que queremos:
# a. a versão do binário da terraform maior ou igual a 0.12
      required_version = ">=0.12"
      required_providers {
      aws = {
# Especificamos a partir de onde queremos baixar o binário:
            source = "hashicorp/aws"
# Dizemos que só permitirá:
# b. a versão do binário do provider 3.20.0 (com certas restrições)
     version = "~> 3.20.0"
   }
 }
}
# ================================================================
```

Aqui, podemos ver que a sentença “required_providers” está setada a “aws”, ou seja, eu não tenho interesse em trabalhar com o Google ou com a Microsoft, mas com a AWS especificamente.
Após isso, através da sentença "source = “hashicorp/aws””, lhe indicamos a partir de onde vamos baixar (algo que ocorre de forma automática) este provisioner.
Continuando com nossa ilustração, o termo “Upstream APIs” refere-se ao método utilizado pelo protocolo HTTP para “carregar” ou “descarregar” dados a partir da origem.

Por que ela faz uso da terminologia HTTP?

A API que a AWS nos oferece para se conectar aos nossos plugins utiliza as operações básicas CRUD (create, read, update, delete). Este modelo é recebido pelas operações HTTP REST.

Provisioner

Um provisioner é um método escrito no código mesmo da HCL da Terraform, e serve para pular qualquer quebra ou gap que não pode ser coberta pelos métodos padrão que a Terraform oferece. Por exemplo: executar comandos remotos em um servidor.
Nota: da mesma forma, a Hashicorp, empresa proprietária do produto Terraform, recomenda o uso de provisioners apenas em casos extremos.
Para fazer isso, existem ferramentas de "Configuration Management", por exemplo, a Ansible e a Puppet. Se por algum motivo não for possível fazer uso destas ferramentas, a Terraform nos possibilita o uso deste método no seu código programável.
Um exemplo do uso do provisioner seria esta peça de código HCL:

```
resource "aws_instance" "web" {
# ...
   provisioner "remote-exec" {
       inline = [
           "puppet apply",
           "consul join ${aws_instance.web.private_ip}",
               ]
   }
}
```

Nesta peça ou "snippet", podemos ver o método "remote-exec" sendo utilizado para executar comandos remotos.

Conclusão

Quando nos referimos à execução da "terraform", estamos normalmente falando em fornecimento para afetar os objetos de infraestrutura reais. Vamos nos lembrar das aulas anteriores, em que mencionamos que o binário da Terraform possui outros sub-comandos para uma ampla variedade de ações administrativas: plan, apply, destroy, etc. Porém, por trás desses comandos, a arquitetura é a mesma.