Certified Tech Developer
The Ultimate Degree

Infraestrutura II

Terraform: nos bastidores

Como vimos, a função principal do Terraform é criar, modificar e destruir recursos de infraestrutura.

Mas como esse componente realmente funciona? Como você se comunica com nosso provedor de nuvem? Como está configurado?

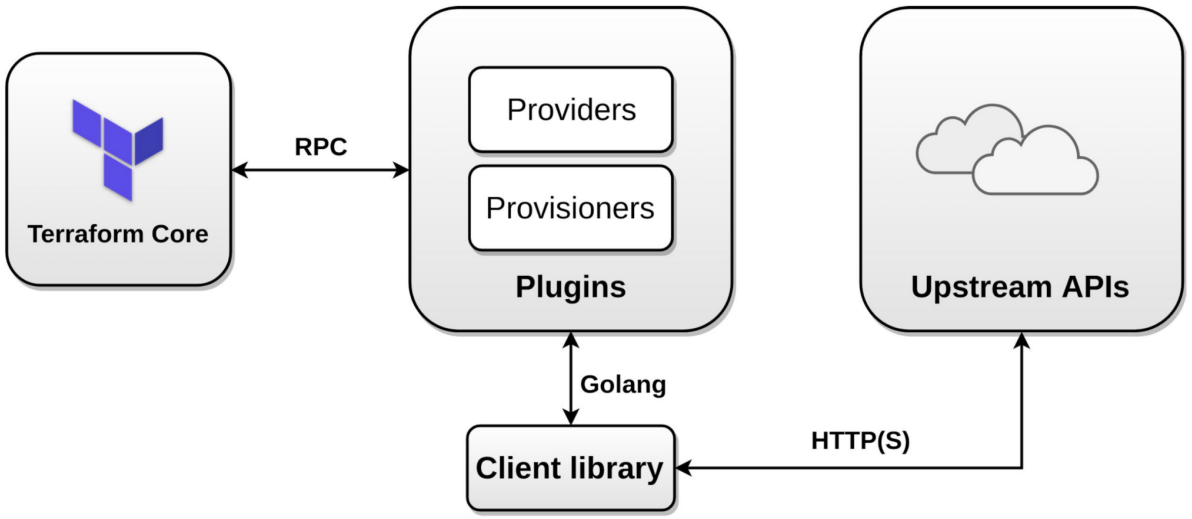
Vamos descobrir juntos!

Arquitetura

O núcleo do TF é composto de várias partes móveis que:

1. Fornecem uma camada de abstração sobre a API subjacente.
2. Eles são responsáveis por interpretar as interações da API e expor recursos.
3. Eles oferecem suporte a várias instâncias de provedores de nuvem.

Se pudéssemos tirar uma radiografia desse componente e como ele lida com seu fluxo de dados, poderíamos ver algo semelhante ao seguinte:



Vamos dar uma olhada em cada um desses elementos e, em seguida, tentar entender como ele se comunica com nosso provedor de dados nuvem.

Plugins

É um aplicativo complementar, geralmente pequeno, que serve para agregar funcionalidades extras ou adicionais (muito específicas) a algo que já existe. Os plug-ins usados estão divididos em: Provedores e Provisionadores.

Provedores

Um provedor é um plugin "específico" que permitirá ao nosso provedor de nuvem entender o idioma em que vamos falar. Por exemplo, para informar que queremos um novo servidor.

O uso do termo "específico" refere-se ao fato de que existem vários provedores, por exemplo: um provedor para AWS, outro para GCP, para Azure, Kubernetes, etc. Aqui podemos ver a lista completa.

Como você se comunica com a nuvem?

Do lado da nuvem, existe uma API especialmente desenhada para saber interpretar os comandos vindos do nosso computador. Em outras palavras, é ouvir nossos pedidos.

Se o “provedor” não existisse, não haveria comunicação entre ambas as partes.

Ao executar, por exemplo, o comando "plano de terraform", este binário irá procurar o "Provedor" que definimos em nosso módulo de terraform:

```
# ================= = ============================================================
# Declaramos o provedor de nuvem com o qual queremos trabalhar
terraform {
# Dizemos que queremos:
# a. a versão do binário do terrenoform maior ou igual a 0.12
      required_version = ">=0.12"
      required_providers {
      aws = {
# Especificamos de onde queremos fazer o download do binário:
            source = "hashicorp / aws"
# Dizemos que só permitirá:
# b. a versão binária do provedor 3.20.0 (com algumas restrições)
     version = "~> 3.20.0"
   }
 }
}
# ==================================== = ================================
```

Aqui podemos ver que a frase "required_providers" está definida como "aws", Em outras palavras, não nos interessa em trabalhar com o Google ou a Microsoft, mas especificamente com a AWS.

Então, por meio da instrução "source =" hashicorp / aws "", informamos de onde faremos o download (algo que ocorre automaticamente) desse provisionador.

Continuando com nossa ilustração, o termo "APIs Upstream" se refere ao método usado pelo protocolo HTTP para "fazer upload" ou "download" de dados de / ou para a fonte de origem.

Por que você usa a terminologia HTTP?

A API que a AWS nos oferece para entrar em contato com nossos plug-ins usa as operações CRUD básicas (criar, ler, atualizar, excluir). Este modelo é assumido por operações HTTP REST.

Provisionador

Um provisionador é um método escrito no próprio código HCL do Terraform e serve para preencher todas as lacunas que não podem ser cobertas pelos métodos padrão que o Terraform oferece. Por exemplo: executar comandos remotos em um servidor.

Observação: Da mesma forma, a Hashicorp, a empresa que possui o produto Terraform, recomenda o uso de provisionadores apenas em casos extremos.

Para esta tarefa, existem ferramentas de “Gerenciamento de Configuração”, como Ansible ou Puppet. Se por algum motivo essas ferramentas não puderem ser utilizadas, o Terraform nos oferece a possibilidade de utilizar este método em seu código programável.

Um exemplo do uso de provisionador seria o seguinte trecho de código HCL:

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

Neste trecho ou “snippet”, podemos ver o método “remote-exec” usado para executar comandos remotos.

Conclusão

Quando nos referimos à execução de "terraform", geralmente falamos de provisionamento para afetar objetos de infraestrutura reais. Lembre-se das classes anteriores de que o binário do Terraform tem outros subcomandos para uma ampla variedade de ações administrativas: planejar, aplicar, destruir, etc. Por trás de todos esses comandos, a arquitetura é a mesma.