Build the World

with Terraform

Tim Olson

November 20, 2021

Outline

- History of Terraform
- What exactly is Terraform
- Comparing Terraform to other Tools
- Components of Terraform
- Phases of Terraform
- Demos!

History of Terraform

Let's take a journey to the 2010s

- We have a few newish cloud hosting solutions
- AWS, GCP, Azure
- For better or worse, in the cloud we have to solve problems in a new way

Consider the case of an AWS shop

- I'm not managing iptables, I'm using AWS Security Groups
- I'm not racking servers, I'm launching EC2 Instances
- I'm not configuring Unbound, I'm creating Route53 records
- You get the idea...

We're changing where the problem space from configuration management to state management.

Consider the case of an AWS shop (cont)

Some problems along our journey:

- Is the cloud choice even the right one?
- How are you going to create these resources?
    - ClickOps - Not reproducible
    - AWS CLI - A bit cumbersome
    - Python libraries - Highly complex

A lot of the knowledge we gain in learning AWS stays in AWS. Are we ok with the potential to vendor lock on on these solutions?

Enter Terraform

Aha! We might have a solution. It's 2015 and there is this new tool: Terraform

- CLI tool
- A mechanism for provisioning popular cloud providers
- Declare the state of your infrastructure
- Controller Pattern

The primary operation: Terraform performs a series of CRUD operations against a remote API.

- Important thought, is the remote API always a cloud provider?

Consider a controller pattern

kubernetes

Consider the fact Kuberentes hits an API with a payload. Our client fetches and displays the current state. Most importantly Kubernetes is auditing our desired state versus the current state, and it does whatever it can to schedule and achieve that.

¡tim open ./assets/kubedeploy/kubedeploy

Compare this pattern to another tool

Ansible

State Management

Config Management

- Terraform aims to abstract datacenter functions
- Strength of Terraform: bootstrap and initialize resources
- Getting software running
  - Config Management activated on system startup or later
  - Custom ISOs on boot

Terraform Primitives

- Provider
- Resource
- Data
- State

Terraform Primitives

Provider

Terraform relies on plugins called "providers" to interact with cloud providers, SaaS providers, and other APIs.

Every resource type is implemented by a provider; without providers, Terraform can't manage any kind of infrastructure.

Most providers configure a specific infrastructure platform (either cloud or self-hosted). Providers can also offer local utilities for tasks like generating random numbers for unique resource names.

```
provider "aws" {
  region = "us-east-1"
}
```

Terraform Primitives

Resource

Resources are the most important element in the Terraform language. Each resource block describes one or more infrastructure objects, such as virtual networks, compute instances, or higher-level components such as DNS records.

- Block, Behavior (CRUD), Metas (depends_on, count, etc), Provisioners (local-exec)

```
# Not a real record :)
resource "aws_route53_record" "wg" {
  zone_id = data.aws_route53_zone.primary.zone_id
  name    = "wireguard.tolson.io"
  type    = "A"
  ttl     = "300"
  records = [var.wan-ip]
}
```

Terraform Primitives
Data

Data sources allow Terraform use information defined outside of Terraform, defined by another separate Terraform configuration, or modified by functions.

```
data "google_iam_policy" "dns_admin" {
  binding {
    role = "projects/${local.project_id}/roles/${google_pro

    members = [
      "serviceAccount:${google_service_account.dns_sa.accou
    ]
  }
}
```

Terraform Primitives

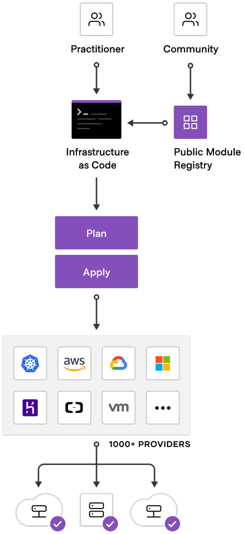
State

Terraform must store state about your managed infrastructure and configuration. This state is used by Terraform to map real world resources to your configuration, keep track of metadata, and to improve performance for large infrastructures.
This state is stored by default in a local file named ”terraform.tfstate”, but it can also be stored remotely, which works better in a team environment.

```
terraform {
  backend "s3" {
    bucket               = "terraform-state"
    workspace_key_prefix = "eks"
    key                  = "terraform.tfstate"
    region               = "us-east-2"
    dynamodb_table       = "terraform-state-lock"
  }
}
```

Terraform Phases

- Write it
- terraform init
- terraform plan
- terraform apply



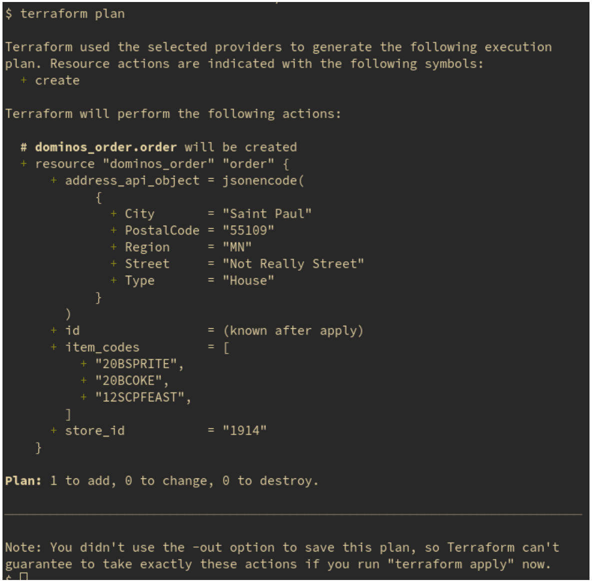
Shall we Demo?

I don't use AWS so none of this made sense.

Shall we Demo?

Remember, Terraform works with APIs not just cloud providers.

Plan



```
$ terraform plan

Terraform used the selected providers to generate the following execution
plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # dominos_order.order will be created
  + resource "dominos_order" "order" {
      + address_api_object = jsonencode(
            {
              + City       = "Saint Paul"
              + PostalCode = "55109"
              + Region     = "MN"
              + Street     = "Not Really Street"
              + Type       = "House"
            }
        )
      + id                 = (known after apply)
      + item_codes         = [
          + "20BSPRITE",
          + "20BCOKE",
          + "12SCPFEAST",
        ]
      + store_id           = "1914"
    }

Plan: 1 to add, 0 to change, 0 to destroy.

─────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't
guarantee to take exactly these actions if you run "terraform apply" now.
$
```

Demo DNS

```
watch -n 0 dig terraform-hello.tolson.io
```

Launch VMs

Launch some VMs

Quick action items!

- CICD?
  - Terraform is a CLI, wrappers exist but no library.
- Launching VMs without Config Management
  - Use an image from the cloud provider and tack on cloud-init
  - Packer!
- Modules
- Workspaces
- Browse the registry

Links and References

- Docs: https://www.terraform.io/docs/language/index.html
- Great tutorials: https://learn.hashicorp.com/terraform
- Browze providers: https://registry.terraform.io/