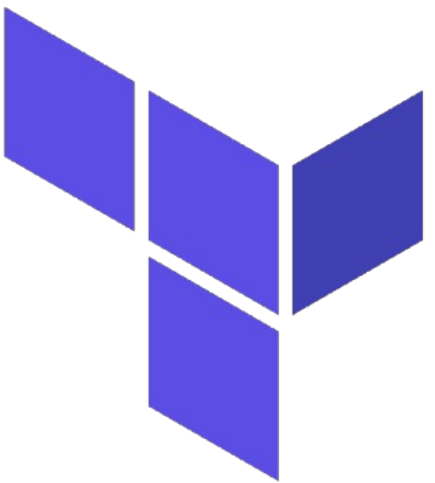
Terraform

William Harrell
8/8/20
LUG @ NC State



Cloud Infrastructure

- **Many modern software companies don’t own their servers, and instead rent them from cloud providers**
- **Many resources are “serverless” or are less focused on a specific piece of hardware**
  - Networking elements (e.g. load balancers)
  - Event driven executors (e.g. AWS Lambda)
  - Storage (e.g. AWS S3)
- **Hundreds of cloud providers, each with pros and cons**
  - Google Cloud Platform
  - Azure
  - AWS
  - DigitalOcean
  - Linode
  - … and more, each with their own web consoles, CLIs, and APIs

Infrastructure as Code (IaC)

- **Describe your infrastructure in a source file**
- **Keep all the infrastructure for a project in one file or folder, instead of having it scattered around pages in a web console**
- **Keep it stored in version control as a backup, revert to an old configuration when something goes wrong**
- **Platform specific IaC solutions already exist**
  - AWS CloudFormation (YAML)
  - GCP Deployment Manager (Python)
  - … but they can only manage resources on that platform

Terraform

- **Write IaC for multiple providers in the same file/folder**
- **Wrapper on top of provider APIs**
- **Describe what you want – Terraform will figure out how to get there**
- **Written in Hashicorp Configuration Language**
- **Terraform itself is written in Go**
- **Available as a web interface, or through a CLI**
  - I'll be focusing on the CLI

```
10 resource "digitalocean_vpc" "vpc" {
11   name     = "LUG-demo-vpc"
12   region   = "nyc1"
13 }
14
15 resource "digitalocean_droplet" "express_server" {
16   image     = "ubuntu-20-04-x64"
17   name      = "LUG-demo-droplet"
18   region    = "nyc1"
19   size      = "s-1vcpu-1gb"
20   ssh_keys  = [var.ssh_fingerprint]
21   vpc_uuid  = digitalocean_vpc.vpc.id
22
23   provisioner "file" {
24     source      = "./project"
25     destination = "/root"
26
27     connection {
28       type = "ssh"
29       user = "root"
30       host = self.ipv4_address
31       private_key = file("./id_rsa")
32     }
33   }
```

Project Structure

- ***.tf – HCL specifying resources**
  - Terraform will combine all .tf files in current directory when executing
- **.tfvars – Variable definitions**
- **.terraform/ - Downloaded providers and modules**
- **.tfstate – Most recent copy of the project’s state in the cloud**
  - Not used if a remote backend is used instead (more on this later)

The Four Commandments

- **Plan – Generate a plan of what to do, but don’t execute it**
- **Apply – Generate a plan of what to do and execute it**
- **Show – View the current state of your resources**
- **Destroy – Delete all resources**
  - Apply and Destroy both ask for confirmation before making changes

Reuse

- **Variables can be declared and referenced throughout the project, to reuse a set of resources for multiple, similar projects**
- **Variables can be defined through**
  - Project properties in Terraform Cloud
  - Environment variables
  - .tfvars files
  - On the CLI as parameters or during execution (not recommended)

Modules

- If you want to share your configuration, you can turn it into a module
- Just move all your .tf files into a subdirectory
- Add input and output variables to really allow reuse
- Publish modules to Terraform Registry

Using existing resources

- **Sometimes you want to reference a resource that already exists, without giving Terraform control of it**
  - Use a data block, and Terraform will grab information about it to be used throughout your script as though it was any other resource
- **Other times, you want to import a resource into Terraform without having to destroy and rebuild**
  - Use “terraform import <HCL id> <provider id>” to bring the resource into the state

Terraform Online Co-op Mode

- **Terraform files are easy to keep in source control and share with peers**
- **… but this is only the configuration, not the state of the resources which are also needed**
- **By default state is kept in .tfstate**
- **A remote backend can be used to store and share state**
  - e.g. AWS S3 bucket with DynamoDB table for locking