Terraform 101

Infinity Works 101 Sessions
Tutors : Richard Harper

WiFi : WeWork

Password : P@ssw0rd

Slack : https://tinyurl.com/yar2yfpb

https://github.com/RichardHarperInfinityWorks/100-training-terraform

Why are we doing this ?

To spread the knowledge of what we are doing

To improve by teaching

To involve more people with what we are excited about

Offer something different to the community

INFINITYWORKS

Terraform-100

Session 1

Agenda

- Introduction to Terraform
  - HCL
  - Variables
  - Providers
  - Data Sources
  - Resources
- Structure of a Terraform Project
- Installing Terraform
- The Terraform CLI
- Getting the sample code
- Deploying an server with Terraform - Workshop

INFINITYWORKS

An introduction to Terraform

Infrastructure as code

Overview

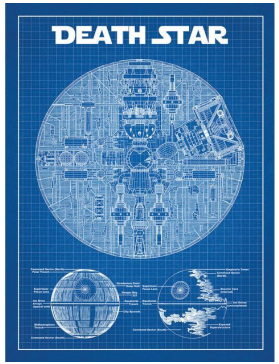
HashiCorp Terraform

"Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently." www.terraform.io

"Terraform generates an execution plan describing what it will do to reach the desired state, and then executes it to build the described infrastructure." www.terraform.io

Sequence of Deployment

Design



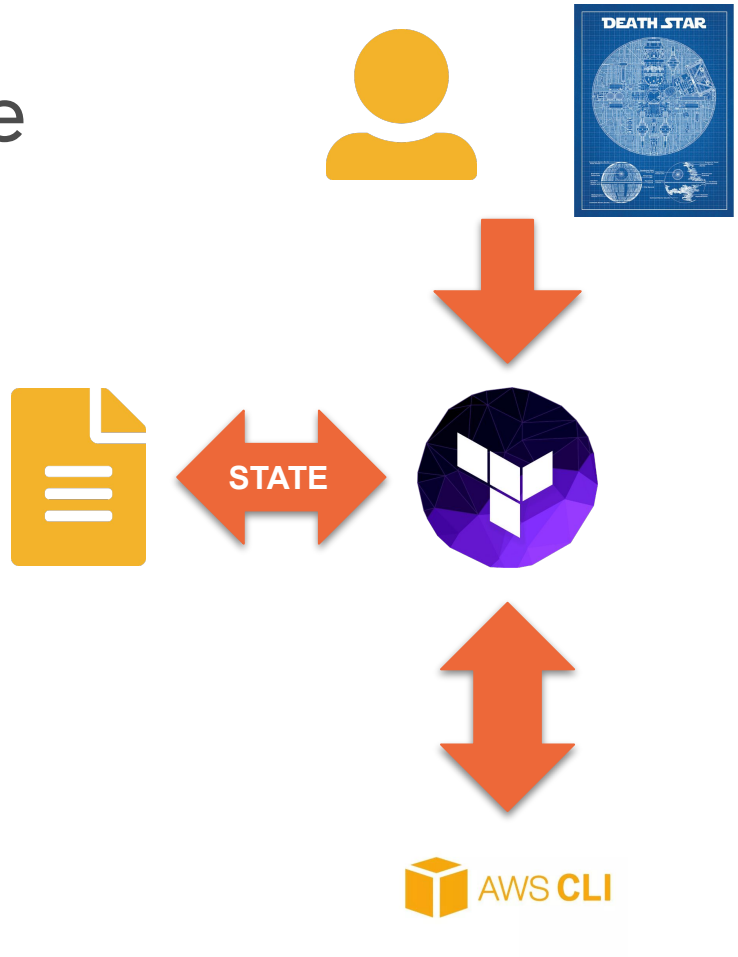
Plan Changes

Apply Changes

AWS CLI

State Persistence / Purpose

- State location can be defined
- Describes known artefacts only
- Local state by default
- Used to plan against
- Updated post apply
- Can be manipulated



HCL (HashiCorp Configuration Language)

HCL is designed to be both human and machine friendly

- Single line comments start with # or //
- Values are assigned with the syntax key = value # (whitespace doesn't matter).
- Values can be any primitive: a string, number, boolean, object, or list.
- Strings double-quoted can contain any UTF-8 characters: // “Hello, World”
- Interpolated Values can be accessed using ${scope.key} # ${var.myvariable}

Variables

```
variable "aws_region" {
 description = "EC2 Region for the VPC"
 default     = "eu-west-1"
}
```

Input variables serve as parameters for a Terraform module, each block declares a single variable, all arguments are optional.

Root Module can be set from CLI arguments & environment variables, child Modules allow values to pass from parent to child.

- Type # defines the type of the variable; string, list, & map.
- Default # defines a default value for the variable if not set.
- Description # should be a human-friendly description for the variable.

Provider Configuration

```
provider "aws" {
 # AWS provider region set in vars
 region = "${var.aws_region}"
}
```

Providers are responsible for understanding API interactions and exposing resources.

- Current list of Terraform Providers: https://www.terraform.io/docs/providers/index.html
- Current list of Community Providers: https://www.terraform.io/docs/providers/type/community-index.html

Data Source Configuration

```
data "aws_ami" "amazon-linux-2" {
 most_recent = true
 filter {
   name   = "name"
   values = ["amzn2-ami-hvm*"]
 }
 filter {
   name   = "architecture"
   values = ["x86_64"]
 }
}
```

Data sources allow data to be fetched or computed for use elsewhere in Terraform configuration.

- Either, Present read-only views OR compute new values
- Configuration TYPES are specific to a Provider # AWS in this case
- Data instances have a “TYPE” & “NAME” # this combination must be unique
- Each exports one or more attributes # accessible by ${data.TYPE.NAME.ATTR}

Resource Configuration

```
resource "aws_instance" "myinstance" {
 ami = "${data.aws_ami.amazon-linux-2.id}"
 instance_type = "t2.micro"

 tags = {
   Name = "myinstance - 101 Training"
 }
}
```

Resources represent components of your infrastructure # Database, DNS record...

Current documentation for all resources can be found online:
https://www.terraform.io/docs/providers/aws/r/instance.html

- Resources have a “TYPE” & “NAME” # resource “PROVIDER_TYPE” “NAME”
- Resources have multiple arguments some of which are [optional]
- Values can be dynamic and are wrapped with ${scope.name.property}
- Some arguments accept arrays: [x,y,z] or collections: argument = { key = value }

INFINITYWORKS

Structure of a Terraform Project

Infrastructure as code

The bare minimum project

Terraform will read all *.tf files in a directory

- aws.tf # contains the Terraform provider details
- main.tf # contains the Terraform resources to plan against
- README.md # a guide in Markdown syntax that describes the project/repo
- terraform.tfvars # variable values to be available to *.tf as ${var.x}
- variables.tf # variable definitions and defaults that can be overridden in terraform.tfvars

INFINITYWORKS

Installing Terraform

Infrastructure as code

Installing the Terraform CLI (Command Line Interface)

Instructions: https://www.terraform.io/downloads.html for: macOS, FreeBSD, Linux, OpenBSD, Solaris & Windows.

To test your installation:

```
richard@groundhog:~$  terraform --version
Terraform v0.11.11
```

INFINITYWORKS

The Terraform command

Infrastructure as code

Terraform - Command Line Interface the basics….

- init # initialize a Terraform working directory, downloads providers and modules in the *.tf
- plan # generate and show an execution plan based upon the state vs design
- apply # builds or changes infrastructure, from a resource graph
- destroy # destroy Terraform-managed infrastructure

INFINITYWORKS

Getting the sample code

Infrastructure as code

Checkout the code using Git

Installing Git https://git-scm.com/downloads for: Mac OS X, Windows & Linux/Unix

To test your Git installation:

```
richard@groundhog:~$ git --version
git version 2.17.1
```

Repository: https://github.com/RichardHarperInfinityWorks/100-training-terraform

Git command to clone source code locally:

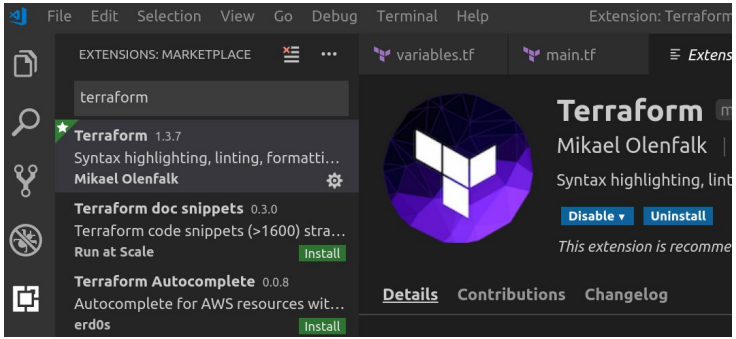
```
richard@groundhog:~$ git clone https://github.com/RichardHarperInfinityWorks/100-training-terraform.git
```

Open the code in VS Code

Download & install VS Code https://code.visualstudio.com/download
For: Mac OS X, Windows & Linux.

Open Folder…
select the location you cloned the repository

Install the Terraform Extension; reload IDE...



Configuring the AWS credentials for Terraform to use

Detailed instructions: https://www.terraform.io/docs/providers/aws/

```
# Static
provider "aws" {
 region     = "eu-west-1"
 access_key = "anaccesskey"
 secret_key = "asecretkey"
}
```

```
# Environment Vars
provider "aws" {
}
```

```
richard@groundhog:~$ export AWS_ACCESS_KEY_ID="anaccesskey"
richard@groundhog:~$ export AWS_SECRET_ACCESS_KEY="asecretkey"
richard@groundhog:~$ export AWS_DEFAULT_REGION="us-west-2"
```

```
# Shared Credentials file
provider "aws" {
 region     = "eu-west-1"
 shared_credentials_file  = "~/.aws/creds"
 profile = "customprofile"
}
```

INFINITYWORKS

Deploying an EC2 instance with Terraform

Infrastructure as code

THANK YOU

www.infinityworks.com