InfernoRed TECHNOLOGY

AUTOMATING YOUR INFRASTRUCTURE WITH TERRAFORM

ROBERTO HERNANDEZ

WHAT WE WILL COVER?

- What is DevOps?
- What is Terraform?
  - Toolset and Lifecycle
  - What are providers, resources and variables?
  - What does a Terraform file look like?
- Demos - Automating an Azure Infrastructure using Terraform

InfernoRed TECHNOLOGY

WHAT IS DEVOPS?

- A compound of development (Dev) and operations (Ops), DevOps is the union of people, process, and technology to continually provide value to customers.
- What does DevOps mean for teams? DevOps enables formerly siloed roles—development, IT operations, quality engineering, and security—to coordinate and collaborate to produce better, more reliable products. By adopting a DevOps culture along with DevOps practices and tools, teams gain the ability to better respond to customer needs, increase confidence in the applications they build, and achieve business goals faster.

* https://azure.microsoft.com/en-us/overview/what-is-devops/

WHAT IS TERRAFORM?

Terraform is an open-source infrastructure as code software tool that provides a consistent CLI workflow to manage hundreds of cloud services. Terraform codifies cloud APIs into declarative configuration files.

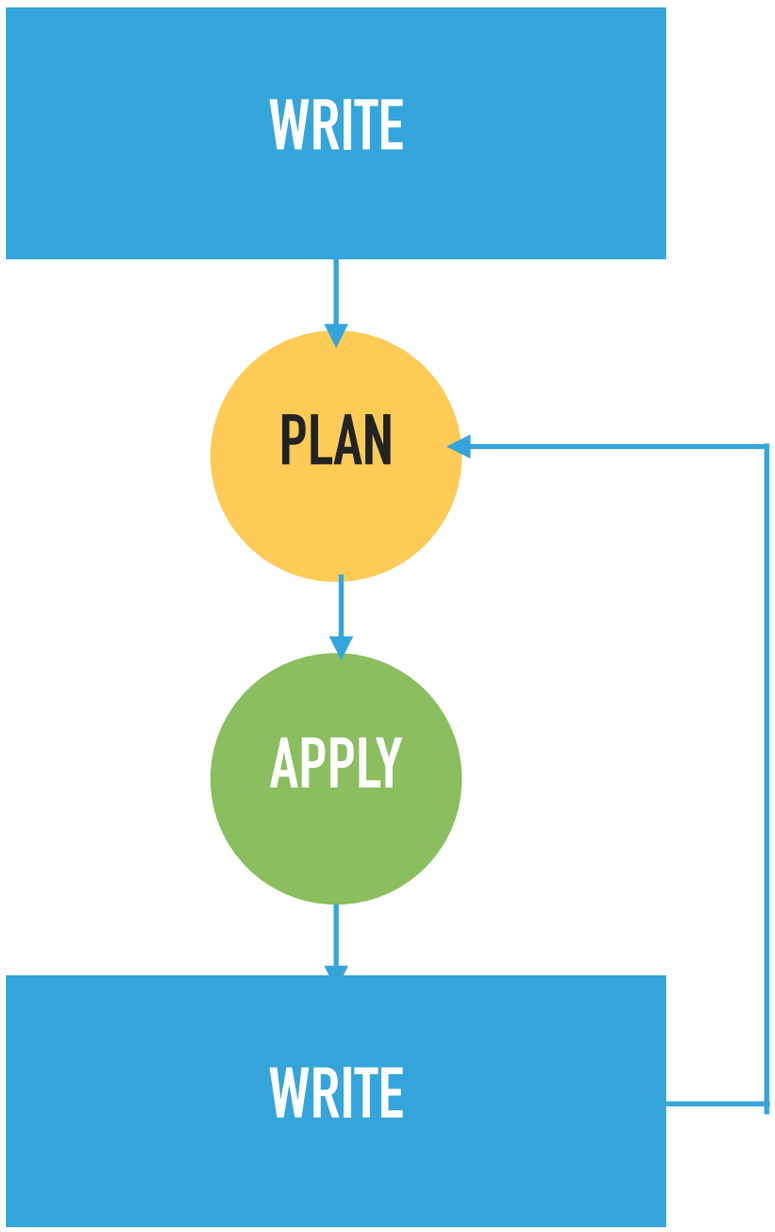
Terraform HashiCorp

```
# Configure the Azure provider
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 2.65"
    }
  }

  required_version = ">= 0.14.9"
}

provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "rg" {
  name     = "myTFResourceGroup"
  location = "westus2"
}
```

InfernoRed TECHNOLOGY

TERRAFORM TOOLSET & LIFECYCLE

- CLI
- Configuration Language
- Modules
- Provision
- State
- Terraform Cloud*



DEMO

- **00 - Write, Plan, Apply**
- 01 - Variables
- 02 - Modules
- 03 - Azure
- 04 - State (Remote)

TERRAFORM CONFIGURATION LANGUAGE

- Files and Directories
- Resources
- Providers (Terraform Registry)
- Variables and Outputs
- Modules
- Expressions (Built-in)
- Functions (Built-in)

```
# Configure the Azure provider
terraform {
  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 2.65"
    }
  }

  required_version = ">= 0.14.9"
}

provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "rg" {
  name     = "myTFResourceGroup"
  location = "westus2"
}
```

VARIABLES

- Local
- Input
- Output

```
variable "image_id" {
  type = string
}

variable "availability_zone_names" {
  type    = list(string)
  default = ["us-west-1a"]
}

variable "docker_ports" {
  type = list(object({
    internal = number
    external = number
    protocol = string
  }))
  default = [
    {
      internal = 8300
      external = 8300
      protocol = "tcp"
    }
  ]
}
```

DEMO

- 00 - Write, Plan, Apply
- **01 -Variables**
- 02 - Modules
- 03 - Azure
- 04 - State (Remote)

MODULES

- Modules are containers for multiple resources that are used together. A module consists of a collection of .tf and/or .tf.json files kept together in a directory.
- Modules are the main way to package and reuse resource configurations with Terraform.

DEMO

- 00 - Write, Plan, Apply
- 01 -Variables
- **02 - Modules**
- 03 - Azure
- 04 - State (Remote)

DEMO

- 00 - Write, Plan, Apply
- 01 -Variables
- 02 - Modules
- **03 - Azure**
- 04 - State (Remote)

DEMO

- 00 - Write, Plan, Apply
- 01 -Variables
- 02 - Modules
- 03 - Azure
- **04 - State (Remote)**

THE END!

DONE

- Roberto Hernandez
  roberto.hernandez@infernored.com
  twitter: @hernandezrobert

- Resources
  https://blog.infernored.com
  https://www.overridethis.com
  https://www.terraform.io/

- Code
  https://github.com/overridethis

InfernoRed TECHNOLOGY