INFRAESTRUCTURA COMO CÓDIGO CON

HashiCorp Terraform

Guillermo Alvarado Mejía
CTO Sentinella

@galvarado89

http://www.linkedin.com/in/guillermoalvarado

¿Quiénes somos?

sentinel.la
by sentinelle.labs

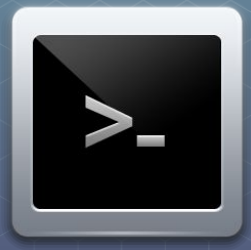
Brindamos soluciones y servicios de Cloud, Containers y DevOps en las plataformas de nube más populares como AWS, Azure, Google Cloud Platform.

Proveemos soluciones de monitoreo y APM, servicios de consultoría y migración de aplicaciones hacia la nube o su "Containerización".

Damos consultoría en todo el proceso para construir, implementar y ejecutar sus aplicaciones en tecnologías como Docker, Kubernetes, OpenShift, Jenkins, Ansible, GitLab, etc.

¿QUÉ ES INFRAESTRUCTURA COMO CÓDIGO?

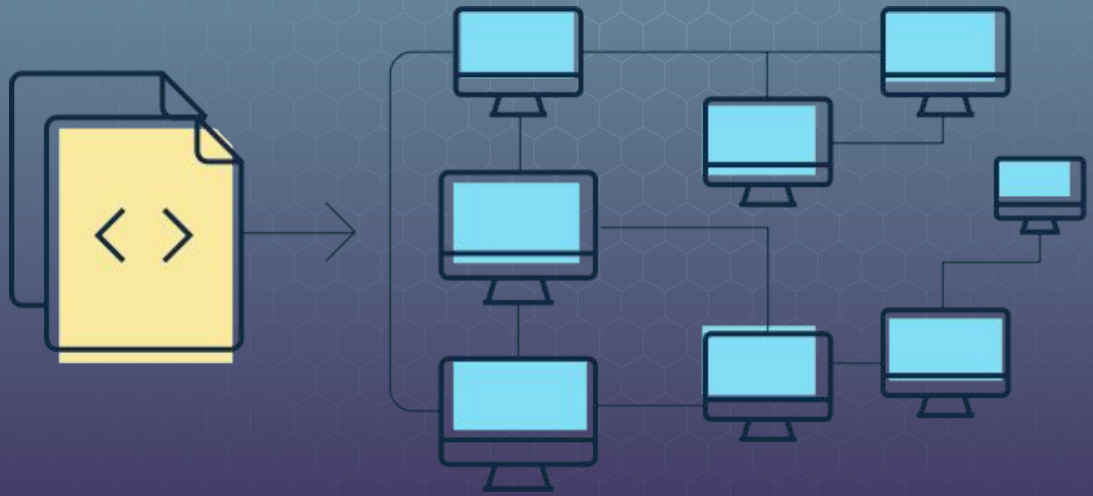
IaC - hace referencia a la práctica de utilizar scripts para configurar la infraestructura de una aplicación, en lugar de hacerlo de forma manual.



La infraestructura como código permite una construcción de infraestructura más consistente y de mayor calidad con mejores capacidades de administración.

¿QUÉ ES INFRAESTRUCTURA COMO CÓDIGO?

El resultado es una infraestructura **elástica, escalable y replicable** gracias a la capacidad de modificar, configurar y apagar cientos de máquinas en cuestión de minutos con solo presionar un botón.



Gestión de configuración vs Orquestación

Gestión de configuración

Chef, Puppet, Ansible y SaltStack son herramientas de "gestión de configuración" lo que significa que están diseñadas para instalar y administrar software en servidores existentes.

Orquestación de infra

Terraform, CloudFormation, OpenStack Heat y otros, son "herramientas de orquestación" lo que significa que están diseñadas para aprovisionar los servidores, dejando el trabajo de configurar esos servidores para otras herramientas.

HashiCorp

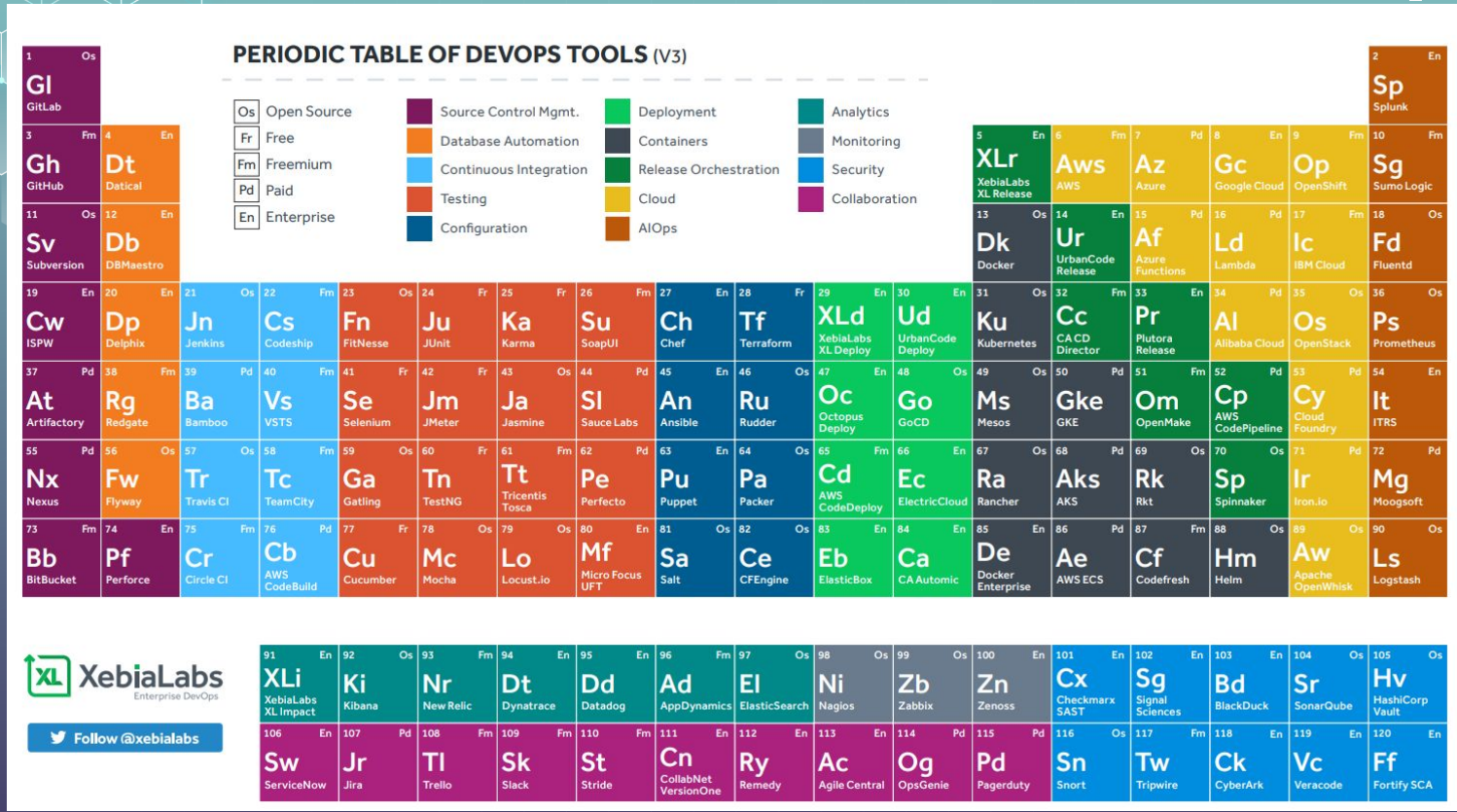
Terraform

Desarrollado por Hashicorp, creadores de Vagrant y Packer, Terraform es software que nos permite definir nuestra infraestructura como código. Terraform es una herramienta de código abierto para construir, combinar y poner en marcha la infraestructura. Desde servidores físicos a contenedores hasta productos SaaS, Terraform es capaz de crear y componer todos los componentes necesarios para ejecutar cualquier servicio o aplicación.

Desarrollador: **HashiCorp**
Programado en: **Go**
Sistema operativo: **Linux**
Licencia: **MPL 2.0**

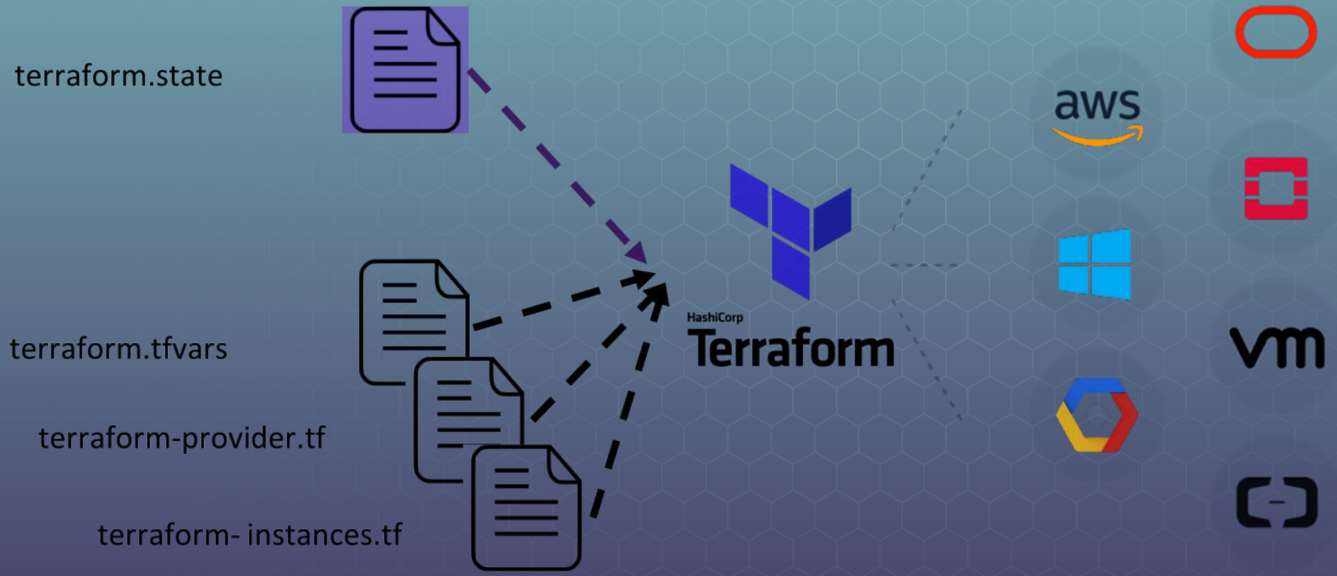
https://github.com/hashicorp/terraform

Tabla Periódica DevOps



https://xebialabs.com/periodic-table-of-devops-tools/download/

Terraform


terraform.state
terraform.tfvars
terraform-provider.tf
terraform- instances.tf
HashiCorp
Terraform
aws
vm

Terraform

La infraestructura que Terraform puede administrar incluye componentes de bajo nivel así como componentes de alto nivel como entradas de DNS, características de SaaS, etc.

https://www.terraform.io/docs/providers/index.html

¿Cómo funciona?

Archivos de configuración

Terraform core, escrito en go, utiliza archivos de texto para describir la infraestructura y establecer variables. El lenguaje de los archivos se llama HashiCorp Configuration Language (HCL)

Plugins

Los plugins de Terraform se escriben en Go y son binarios ejecutables invocados por Terraform Core sobre RPC. Cada plugin expone una implementación para un servicio específico, como AWS. Todos los proveedores y aprovisionadores utilizados en las configuraciones de Terraform son plugins.

Despliegue

Los blueprints descritos según la sintaxis de configuración se crea en los proveedores de infraestructura y permite ser versionada. Además, la infraestructura puede ser compartida y reutilizada.

Hashicorp Configuration Language

```
[root@zenbook terraform-example]# cat openstack-example.tf

# Configure the OpenStack Provider
provider "openstack" {
  user_name   = "admin"
  tenant_name = "admin"
  password    = "somestrongpass"
  auth_url    = "http://controller01:5000/v3"
  region      = "RegionOne"
}

# Create a RHEL server
resource "openstack_compute_instance_v2" "basic" {
  name            = "vm_from_terraform"
  image_id        = "567887bd-2635-4c2e-9feb-248a1b770745"
  flavor_id       = "i78478b4-2d58-42f6-940e-15bdea5a7849"

  metadata = {
    this = "that"
  }
  network {
    name = "Some_Network"
  }
}
```

Ventajas

- Infraestructura como código: Podremos versionar/revertir/redimensionar nuestra infraestructura de forma no destructiva. Esto permite desplegar una arquitectrura 100% idéntica a la principal de forma automatizada en una situación de emergencia o recrear nuestra infraestructura de forma parcial o total en un entorno de desarrollo o pruebas.

- Velocidad: Terraform es muy, muy rápido. Si el proveedor de infraestructura lo soporta, Terraform es capaz de generar los recursos especificados de forma paralela. En la práctica, esto posibilita que el tiempo de despliegue de 1 máquina sea el mismo que el requerido para 10.

- Agnóstico: A diferencia de Heat o CloudFormation, Terraform es capaz de trabajar de forma simultánea con todos sus diferentes proveedores de infraestructura.

Desventajas

- Depende enteramente de la API de cada proveedor: Aunque la herramienta intenta mitigar posibles fallos mediante reintentos, timeouts y el uso de una caché local que contiene el estado de la infraestructura, está a merced de la fiabilidad de la API.

- Terraform almacena el estado de los recursos en el fichero *terraform.state*, y esta es el punto de referencia a la hora de generar planes de ejecución. Si se modifica la infraestructura remota desde fuera de Terraform, el comportamiento de la herramienta puede no ser el deseado.

- La sintáxis declarativa de los ficheros no es agnóstica respecto al proveedor utilizado. Por tanto, debemos mantener diferentes versiones de nuestra infraestructura según dónde vayamos a desplegarla.

DEMO

Aprovisionamiento de una base de datos MYSQL

https://github.com/galvarado/terraform-examples

1

Despliegue de Wordpress en DigitalOcean

https://github.com/galvarado/terraform-ansible-DO-deploy-wordpress

2

GRACIAS