

Introducción a Infraestructura como Código (IaC)

Miguel Barajas

Version 1.0, 2021

Contenido

Dedicación	1
Prefacio	2
Por qué escribí este libro	2
Por qué debería de leer este libro	2
Convenciones usadas en este libro	2
Recursos en línea	2
Como contactarme	2
COPIA ESTE LIBRO	2
Capítulo 1. Introducción.....	3
Automatización de Infraestructura	3
Manejadores de configuración	4
El crecimiento de la popularidad de IaC	4
Capítulo 2. Caso de estudio.	5
Capítulo 3. Introducción a Terraform	6

Dedicación

A Valeria, Roberto y Carlota. Mi mundo

Prefacio

Por qué escribí este libro

Una gran parte de mi carrera profesional, la he dedicado a la automatización y orquestación. Es un tema que me apasiona, sin embargo, la automatización "tradicional" cada vez se ha convertido en un *break and fix*. Es muy estática y requiere mucho esfuerzo mantener los flujos de trabajo, integraciones y *scripts* para que la automatización se siga dando en un mundo tan dinámico como el que vivimos en las tecnologías de la información. Desde el primer momento que descubrí los manejadores de configuración, quedé enamorado. *INCOMPLETO*

Por qué debería de leer este libro

Convenciones usadas en este libro

Recursos en línea

Como contactarme

COPIA ESTE LIBRO

Sí, copia este libro, usalo para cualquier fin, auméntalo, regalalo. Este libro está bajo la licencia CC

Capítulo 1. Introducción.

Para entender la infraestructura como código (IaC por sus siglas en inglés) primero debemos entender qué es infraestructura y qué es código. Infraestructura en el contexto de informática, es el hardware, físico, virtual o lógico que soporta una aplicación o plataforma. Del mismo modo, el código son las instrucciones que permiten crear una aplicación o software. Luego entonces, la infraestructura como código, es la descripción del estado deseado de la configuración de esta infraestructura en un documento, normalmente escrito en texto plano. Al estar escrito en texto plano, esta descripción puede tratarse de la misma manera que se trata un código de software, donde podemos versionarlo, probarlo, compartirlo, etc.

Para hacer esto realidad, debemos contar con un intermediario o intérprete que comprenda esta descripción y la convierta en una configuración que sea entendida por la infraestructura. Esto nos permite automatizar la infraestructura de tal manera que esa configuración la tratemos como software, esto nos da varias ventajas que estaremos discutiendo durante la presente obra, algunas de ellas son:

- Automatización declarativa en vez de imperativa
- Versionamiento de la configuración
- Replicación sencilla
- **ESCRIBIR MAS**

Automatización de Infraestructura

La automatización de la infraestructura no es un tema nuevo, desde hace décadas, está práctica existe, dado a la necesidad de hacer más eficiente el despliegue de esta infraestructura, esto se hace mediante **scripts** o desarrollos que permiten que esta infraestructura se despliegue automáticamente, el problema con este acercamiento, es que no solo le tenemos que planear qué se va a desplegar, si no también cómo se desplegará, por lo que es una automatización **imperativa**, esto nos acarrea un problema dado que muchas veces la infraestructura no puede ser cambiada una vez desplegada, puesto que en la automatización debemos tener en cuenta todas las excepciones que pudieran ocurrir durante el ciclo de vida de la infraestructura. Por ende, la automatización imperativa funciona muy bien solo para el despliegue de nueva infraestructura, pero no para mantener el ciclo de vida completa de ella. Es decir, despliegue, cambios y decomisión.

Más allá de la automatización, existe el concepto de **orquestración** en el que normalmente existe un software llamado **orquestrador** (**orchestrator** en inglés) el cual cuenta con los conectores necesarios para automatizar el despliegue de las diferentes capas de la infraestructura: Almacenamiento, Red, Computo, Sistema Operativo, etc).

Esto permite "orquestrar" el despliegue de la pila completa de la infraestructura. Pero de la misma manera que se hace al automatizar de manera imperativa, capa por capa, la orquestración imperativa no permite manejar el ciclo de vida completo de la infraestructura.

Manejadores de configuración

Los manejadores de configuración, han existido, de igual manera, desde hace décadas, son piezas de software que nos permiten mantener la configuración de un sistema operativo o aplicación de manera declarativa. Es decir, en vez de indicar el qué y el cómo, se describe el qué, es decir se declara el estado deseado de la configuración del Sistema Operativo o aplicación y es el manejador de configuración quien hace la conciliación entre el estado declarado y el estado actual. Es decir, que solo modifica las partes de la configuración que no concuerdan con el estado deseado. Como se manifestó, los Manejadores de configuración (***Configuration Manager*** en inglés) fueron concebidos para mantener la configuración del Sistema Operativo y/o la aplicación. Con el avance de las tecnologías de virtualización y del concepto de ***Software Defined***, el despliegue y configuración de la infraestructura, también puede ser declarada y manejada como si de un sistema operativo o una aplicación se tratara.

El crecimiento de la popularidad de IaC

Sin duda, la adopción del ***Cloud Computing*** ha permitido que la Infraestructura como código se haya popularizado, la disponibilización de la interfaces de programabilidad (APIs), así como el cobro por tiempo utilizado de la infraestructura, ha hecho que IaC sea muy popular para este tipo de modelos de consumo. Sin embargo, también e influenciado los fabricantes de infraestructura física, y estos han empezado a disponibilizar APIs Así como también han separado el plano de control del plano de datos, que es el principio del *hardware* definido por *software*. Empresas como ***Hashicorp*** han creado soluciones de Infraestructura como código que estaremos usando en esta obra, que lleva por nombre ***Terraform***. Terraform es una herramienta de código abierto para infraestructura como código que provee un flujo de trabajo por linea de comandos consistente para manejar cientos de servicios de nube. Terraform codifica las APIs de las nubes en archivos de configuración declarativa. ^[1]

[1] <https://www.terraform.io/>

Capítulo 2. Caso de estudio.

Através de este libro, estaremos creando los bloques necesarios para mantener la operación de una aplicación mantenida en la nube, la cual cuenta con varios elementos de infraestructura que definiremos como código. La idea de este caso de estudio, será que podamos desplegar esta aplicación, modificarla, escalarla, decomisarla, cambiarla de una región a otra, etc.

El caso de estudio será de la empresa imaginaria de repostería "Galletería Carlota" la cual mantiene su sistema de tienda en línea en la **Amazon Web Service** o **AWS** como le llamaremos de aquí en adelante. La aplicación tienen varias capas o niveles como se muestra en la siguiente figura, esta será la arquitectura propuesta para la aplicación. El cual cuenta con lo siguiente:

- Tres (3) Balanceador de Carga nativo de AWS para la capas de presentación, aplicación y base de datos.
- Una capa de presentación *web* que permite ser escalada verticalmente
- Las imágenes de los productos a vender serán almacenadas en un *S3 Bucket*
- Una capa de Aplicación que será donde se procese la lógica de negocio la cual también debe ser verticalmente escalable
- Una capa de Base de datos, la cual es nativa a la nube y permite ser escalada verticalmente, en este caso usaremos *CoackroachDB*

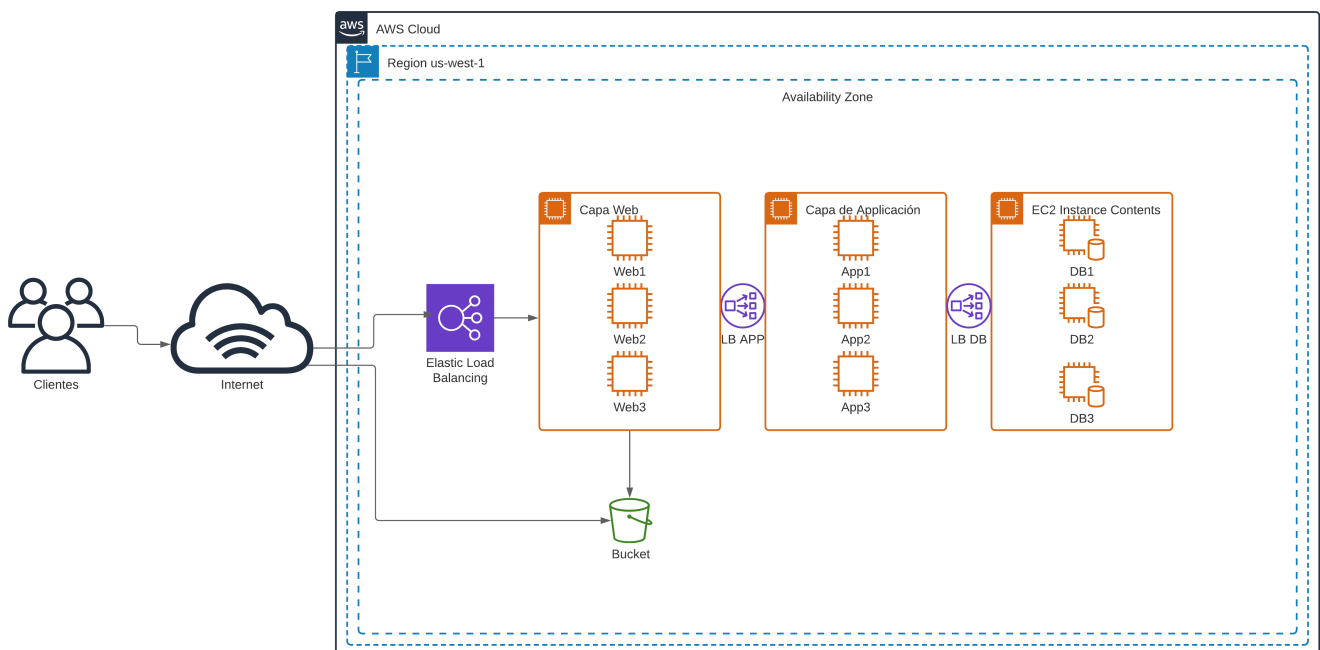


Imagen 1. Arquitectura de la Aplicación en AWS

Capítulo 3. Introducción a Terraform

Como ya se platicó en el capítulo uno, durante el desarrollo de esta obra estaremos utilizando la versión abierta y gratuita de **Hashicorp Terraform**, la cual ha tomado mucha popularidad entre ingenieros que hacen **DEVOPS** y **CI/CD**, temas que trataremos más adelante. Y no es casualidad, **Terraform** ofrece un entorno bien documentado y con cientos de integraciones que permiten a las organizaciones tratar su infraestructura como código al escribir archivos con la declaración del estado deseado de las configuraciones. **Terraform** usa su propio language llamado **Hashicorp Configuration Language (HCL)** que permite una descripción consisa de los recursos usando bloques de código, argumentos y expresiones.

De la misma manera, **Terraform** permite correr las verificaciones necesarias antes de aplicar las configuraciones deseadas para asegurarse que los cambios a realizar son factibles y el operador tiene oportunidad de entender qué cambios sucederán y como afectará el ambiente desplegado. Una vez hecho esto el operador puede proceder a aplicar los cambios, para poder llegar al estado deseado solo modificando el delta entre el estado actual y el deseado.

Código Ejemplo escrito en HCL para AWS

```
variable "ami_id" {
  type      = string
  description = "AMI ID to use"
  default    = "ami-09d95fab7fff3776c"
}

variable "instance_type" {
  type      = string
  description = "Instance type to use"
  default    = "t3.micro"
}

variable "availability_zone" {
  type      = string
  description = "Availability Zone to use"
  default    = "us-east-1a"
}
```