ÍNDICE

INTRODUCCIÓN

UN POCO DE HISTORIA

La constante evolución de los entornos de software no es ninguna sorpresa para nadie involucrado en todo este mundo. Llevo ya 12 años trabajando como profesional y en esta pequeña fracción de la historia de la tecnología me da mucha risa recordar el hardware con el que trabajaba en la universidad, recordar versiones de los sistemas operativos que usaba y con un nivel no menor de pena reconocer que fui un gran partidario de las aplicaciones entonces conocidas como *Rich Internet Applications* o RIA, productos como Flash, Flex, Java FX o Silverlight son nombres que ahora solo podemos ver como víctimas de la evolución tecnológica, desde ahí hasta ahora he visto un sinfín de productos que se van quedando en el camino de la constante mejora tecnológica (aunque quizá es bueno reconocer también que algunos productos ya extintos pudieron ser mejores de los que ahora se usan para las mismas tareas).

Sin embargo, he visto también que los productos van y vienen, pero los conceptos permanecen inamovibles y eso es lo que me mantiene completamente enamorado de este enorme ajetreo, permíteme ser más específico al hablar del tema que corresponde a este libro.

Hace años, para poder desplegar una aplicación web, el desarrollador debía mantener una constante pelea con el especialista en infraestructura. El primero culpaba al segundo de no proveer una capacidad amplia y cómoda para permitir que

el sitio web estuviera publicado mientras que el segundo culpaba al primero de que la aplicación estaba mal hecha y consumía muchos recursos haciendo de esto el motivo por el que el sitio web no funcionara.

Dando un gran y afortunado paso en la evolución, los proveedores de nube comenzaron a aparecer, al principio como meros prestadores de servicios para almacenamiento de los sitios (hosting) y sus nombres de dominio, poco a poco te invitaban a utilizar nuevos servicios como bases de datos hospedadas, instalación de cierto software con un solo clic, administración de correos corporativos y un largo etcétera. En este proceso de evolución, en la gestión y administración de la infraestructura de los sistemas empezó un gradual cambio dejando de lado la necesidad de que una compañía contara con hardware propio para simplemente pagar los recursos que necesitara sin preocuparse demasiado por su cuidado sino más bien pagando lo que consumiera en cierto periodo. De este mundo, grandes compañías comenzaron a fijar su atención en este tema y fue así como IBM, Google, Amazon y Microsoft destacaron como los principales jugadores en el servicio de cómputo en la nube. Hoy en día el especialista de infraestructura ha evolucionado sus tareas de gran manera, pero siempre velando por los recursos de su compañía. Antes, cuidaba el consumo de electricidad, el consumo de internet, el cuidado de todo el hardware involucrado en sus centros de datos y un sinfín de tareas con bajo nivel técnico, pero de gran importancia en la economía de las empresas. Ahora, sigue velando por esos mismos intereses, pero viendo que no se creen máquinas virtuales más grandes de las necesarias, que los equipos de desarrollo no creen recursos que los sistemas no necesitan, que todo lo que no ha sido utilizado por cierto tiempo se elimine por completo y estas tareas, así como muchas otras más fueron implementando poco a poco nuevas herramientas que permitieran un ahorro cuantioso de tiempo, así como también un mayor control sobre la infraestructura. Y fue así como la herramienta que poco a poco se iba viendo cada vez más reducida tomó un segundo aire y con mucha más fuerza que antes, la línea de comandos regresaba para no volver a irse nunca.

En la línea de comandos de mis amores (sí, permanezco usándola tanto como me sea posible, si hay una tarea que puede ser hecha de manera gráfica o por comandos sin dudarlo me voy por la segunda) se empezaron a sumar muchas nuevas herramientas a la ya de por sí natural capacidad de automatizar comandos para poder tener la capacidad de crear granjas de servidores enormes con un solo comando; ahora, solo basta escribir un buen *script* para poder tener cientos de máquinas virtuales con hardware al que difícilmente le podríamos sacar toda su capacidad, ya podemos crear un conjunto de sitios web enlazados por medio de un balanceador de cargas y una red privada para operar varios procesos de diferentes áreas en nuestra compañía. De momento, gracias a este gran paso, ahora en una

línea de comandos podemos tener todo el control de la infraestructura de nuestra compañía al alcance de un script.

Sabiendo ya de esta capacidad no pasó mucho tiempo para que nuevas herramientas surgieran para administrar recursos en la nube de una manera cada vez más sencilla y una de ellas que fue la primera que usé y me vino a la mente es **Ansible** de Red Hat. Si ya contaba con un buen ahorro de tiempo para poder trabajar con scripts, Ansible me ahorró muchísimo más para no solo crear nuevos recursos, sino que también podía controlar la configuración dentro de los servicios, lo que hacía a mi cabeza volar de la emoción. Paralelo a esto, comencé a adentrarme cada vez más en Azure, de Microsoft como el proveedor de nube por defecto para todos mis proyectos. En cuanto salió la herramienta de línea de comandos de Azure o **Azure CLI** como todos la conocemos, encontré una manera de mezclar mi experiencia son scripts y Ansible para hacer las cosas cada vez más veloces, después conocí una característica adicional de Azure, las plantillas **ARM** que me daban la oportunidad de crear recursos, lo malo es que ARM al crecer te va atrapando en sus kilométricas líneas de código y hace las cosas más confusas, al menos así fue como me sucedió a mí y fue con todo este gran avance que surgió la primera gran pregunta que debí responder.

¿Por qué meterme en todo esto?

Bueno, una parte fundamental de mi carrera siempre ha sido el encontrar la manera más rápida y simple de hacer las cosas. Soy enemigo de hacer las dos cosas de manera manual si hay una posibilidad de automatizar y la metodología de DevOps me enamoró por esto. Tan pronto como aprendí a controlar las versiones de mi software, a integrar y desplegar de manera continua, a controlar las pruebas y una larga lista de ventajas que DevOps te da. La única parte restante para cumplir con un proceso mucho más automatizado era lidiar con la infraestructura que mis soluciones requerían. Como te mencioné, probé con lo que mejor sabía usar, herramientas de automatización y una buena colección de scripts de PowerShell que me ayudó a obtener grandes tips de automatización, después de pasar por Bash, Ansible y uno que otro experimento que me encontraba, me di cuenta de qué quería de las herramientas que buscaba y qué era lo que no quería de ellas, por ejemplo, me llevó dos días descubrir de que las plantillas ARM de Azure no serían una opción que quisiera explorar o implementar por su complejidad y después de varias opciones, pensé que regresar a Bash era la única opción cuando me hablaron de Terraform, ahí las cosas cambiaron por completo.

¿Por qué Terraform?

Terraform es un producto de una compañía llamada HashiCorp que no se especializa en productos orientados a la administración de infraestructura, con media docena de muy buenos productos en su oferta Terraform ha destacado mucho y ha desempeñado un papel cada vez más relevante en cada proyecto vinculado a servicios en la nube. De hecho, HashiCorp cuenta con su propio servicio de nube que ofrece un montón de herramientas para gestionar desde ahí cualquier proyecto.

Una parte importante que me gustaría explicar es que Terraform es una herramienta diseñada para gestionar múltiples servicios en múltiples nubes, en este libro me orientaré principalmente a la creación de servicios dentro de Microsoft Azure y herramientas de Microsoft porque mi experiencia se ha orientado ahí pero no solo se puede trabajar con estos productos, de hecho, en el capítulo de proveedores veremos cómo es posible que puedas utilizar proveedores para servicios como Google Cloud o AWS y mucho más, he visto proveedores de todo tipo y eso no ha hecho más que ayudarme a confirmar la enorme utilidad que una herramienta como esta te puede ofrecer.

Desde ahora te agradezco que estés interesado en agregar una más a tu arsenal de herramientas de trabajo como profesional de software y más aún, que hayas invertido en este libro para aprender acerca de Terraform, espero ser capaz de llenar tus expectativas.

ACERCA DEL AUTOR

Amin Espinoza es un ingeniero de Software Senior en Microsoft Corporation y cuenta con 14 años de trayectoria profesional. Dentro de su historial de proyectos se pueden encontrar varios con algunas de las 500 compañías más grandes del mundo.

Su interés por compartir su experiencia y conocimientos con comunidades lo ha llevado a tener un ritmo constante en su blog con más de 10 años escribiendo artículos de interés en varias tecnologías y tendencias en las que se ha especializado. Este blog solo se ha expandido usando nuevos canales como YouTube y una amplia presencia en redes sociales permitiendo a Amin consolidarse como un destacado generador de contenido en Latinoamérica.

Ha contribuido a una decena de proyectos Open Source que abarcan en general temas especializados en las prácticas de DevOps e Infraestructura en la nube, así como Fundamentos de Ingeniería de Software.

Con este, su segundo libro publicado, no verás nada más que el resultado de todos estos proyectos, experiencia y conversaciones que te permitirán allanar el camino en aras de conseguir mejores resultados en un menor tiempo.

Puedes contactar con Amin en sus medios sociales, así como visitar su sitio web en las direcciones siguientes:

- **Web:** http://aminespinoza.com
- **YouTube:** https://www.youtube.com/c/AminEspinoza/
- **Twitter:** @aminespinoza

1 TERRAFORM

CONCEPTOS BÁSICOS

Mi parte favorita de trabajar con Terraform es que solo requiere dos herramientas, una de ellas es completamente gratuita y de código abierto, Visual Studio Code. La segunda es simplemente parte esencial del sistema operativo que estés utilizando, tu terminal de comandos.

Para arrancar asumiré que ya tienes instalado Visual Studio Code, no requiere ningún esfuerzo relevante tenerlo, lo importante es que después de hacerlo instales una extensión, la extensión es HashiCorpTerraform.

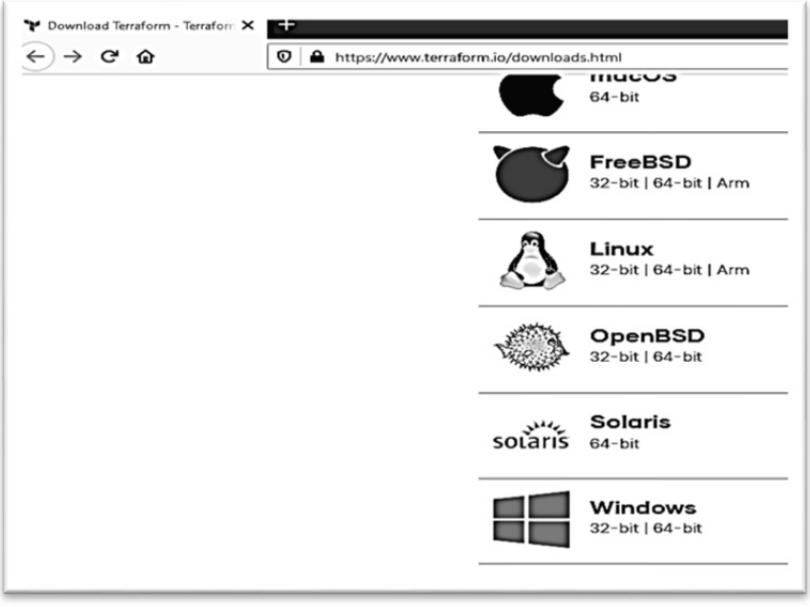
Instalar esta extensión te dará la posibilidad de trabajar con el lenguaje que Terraform utiliza, este lenguaje se llama HCL que viene de las siglas **Hashicorp Configuration Language**, por cierto, un dato muy interesante es que basado en los repositorios que utilizan este lenguaje, Github lo ha considerado como uno de los 10 lenguajes de mayor crecimiento en el año 2020, más que duplicando su base de usuarios con respecto al año pasado y aumentando. Eso es solo una consecuencia natural de la enorme adopción que Terraform ha ido teniendo.

Bien, continuando con la configuración, con que tengas la extensión ya instalada, ahora sigue el paso de instalar Terraform y aquí te podrás saltar unas páginas para que realices el proceso basándome en el sistema operativo que estés utilizando, me divierte mucho saber que estaremos usando un producto que únicamente en su instalación hará una diferencia, después de este paso no importará más tu sistema operativo, los comandos y las herramientas que utilices tendrán exactamente la misma apariencia. ¿No te divierte vivir en la época en donde el sistema operativo es lo de menos? ¡Yo francamente lo disfruto cada día!

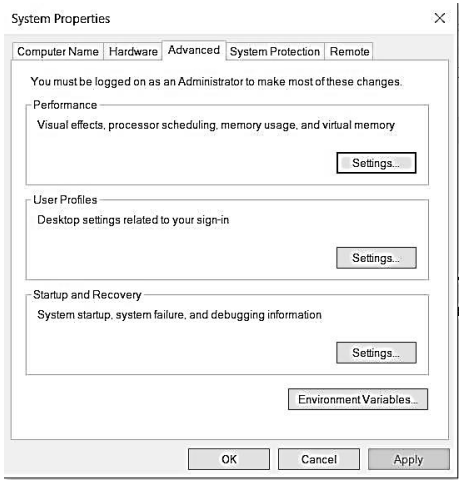
Instalación de Terraform

Comienza por descargar el paquete compilado del sistema operativo que estés usando desde https://www.terraform.io/downloads.html_donde hay una enorme oferta de opciones.

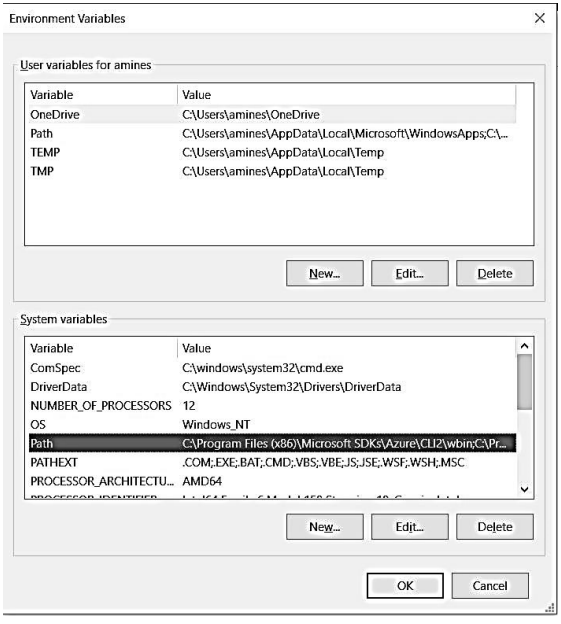
En el momento de escribir este libro, la versión estable es: **1.0.1**



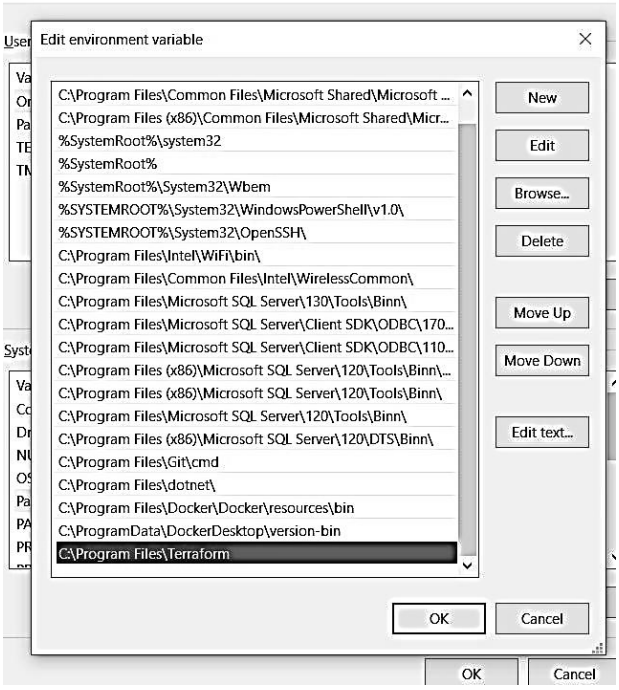
Con el archivo ya descargado en Windows (que es el sistema operativo que estoy utilizando), solo necesitas dar un doble clic y la instalación se ejecuta. Después de ello, puedes ir a tu **Panel de Control**, selecciona la opción de **Sistema** y abre **Configuración avanzada de sistema**.



Al abrir las opciones de las variables de ambiente, debes editar una variable de sistema llamada **Path**.



Agrega la ubicación de Terraform aquí, la misma en donde pusiste el instalable.



Si todo sale ya en orden, entonces abre una terminal y verifica que ya tienes Terraform instalado con el icónico comando de versión.

```
C:\Users\amines>terraform --version
Terraform v0.14.2

Your version of Terraform is out of date! The latest version
is 0.14.3. You can update by downloading from https://www.terraform.io/downloads.html
```

Ahora, ya estás listo para poder trabajar sin mayor problema, solo ten en cuenta un punto importante que, afortunadamente, durante la edición de este libro sucedió: la actualización de Terraform. En Windows lo único que debes hacer para actualizar a la nueva versión es descargar el nuevo archivo ejecutable desde el mismo punto de descarga y colocarlo en la misma ubicación. Haz esto y consulta la nueva versión en tu terminal.

```
C:\Users\amines>terraform --version
Terraform v1.0.1
on windows_amd64
```

Así que podrá ser que la primera configuración en Windows sea un poco compleja, pero es un paso de una sola vez, las actualizaciones por venir únicamente necesitarán que reemplaces el archivo ejecutable en lo sucesivo.

En el caso de MacOS o Linux el proceso es esencialmente el mismo, no hay gran dificultad aquí, vamos a ver los pasos que estos SO requieren.

Instalar Terraform en OS X

La instalación en OS X es muy similar a la de Linux gracias a **Homebrew**. Puedes instalar la fórmula oficial de Terraform desde la terminal.

```
$ brew tap hashicorp/tap
```

Con la fórmula ya hecha, ahora ya puedes instalar Terraform.

```
$ brew install hashicorp/tap/terraform
```

Algo muy interesante es que al instalar Terraform de esta manera automáticamente se actualizará, eso es muy cómodo para evitar tener que estar siguiendo las actualizaciones. De cualquier manera, si quieres hacerlo manualmente puedes hacerlo.

```
$ brew upgrade hashicorp/tap/terraform
```

Ya sabes, solo verifica que todo esté instalado.

```
$ terraform -help
Usage: terraform [-version] [-help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.
##...
```

Instalar Terraform en Ubuntu

Por último, pero no menos importante veamos cómo lo puedes hacer en Ubuntu o esencialmente en cualquier distribución basada en Debian.

Comienza por instalar la llave GPG.

```
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
```

Agrega el repositorio oficial de HashiCorp para Linux.

```
sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"
```

Actualiza e instala Terraform.

```
sudo apt update && sudo apt install terraform
```

En el caso de Mac la instalación hará que la actualización sea automática. En el caso de Ubuntu, la gran ventaja de instalar es que con el repositorio de HashiCorp ya agregado puedes instalar los demás productos de esta compañía. Puedes añadir **Vault, Consul, Nomad** y **Packer**. Así que quizá vale la pena que explores un poco más de otros productos.

Termina verificando que esté todo instalado.

```
aminespinoza@MININT-GQTFFCU:/$ terraform -help
Usage: terraform [-version] [-help] <command> [args]

The available commands for execution are listed below.
The most common, useful commands are shown first, followed by
less common or more advanced commands. If you're just getting
started with Terraform, stick with the common commands. For the
other commands, please read the help and docs before usage.
```

Lo que sigue, después de tener ya Terraform instalado sin importar el sistema operativo, es hacer una pequeña pausa para continuar con toda la configuración restante.

UNA CUENTA DE MICROSOFT AZURE

Trabajar con Terraform es trabajar definitivamente con herramientas de nube. Para este libro, la gran mayoría de ejercicios que harás serán ejecutados dentro de Microsoft Azure. Para ello se requiere que crees una cuenta, no es necesario que se trate de una cuenta de pago, puede ser una de pruebas. La realidad es que si te tomas tu tiempo para leer el libro o simplemente olvidas eliminar las cosas que creaste, entonces quizá sí será una buena idea cambiar al modo de pago. Dejando de lado la decisión que tomes con respecto a esto, entra en el sitio de Azure y verifica que tu cuenta esté ya lista, verás algo así.