Revista de I+D TECNOLÓGICO
Portal de revistas: http://revistas.utp.ac.pa

Herramientas de infraestructura como código: ansible, terrafom, chef, puppet

Tool infrastructure as code: ansible, terrafom, chef, puppet

Luis Dominguez-Quintero [1], *Miguel Vargas-Lombardo* [2*]
[1] Maestría TICS-Senacyt, Universidad tecnológica de Panamá, Panamá
[2] Gises, Universidad Tecnológica de Panamá
***Autor de correspondencia:** miguel.vargas@utp.ac.pa

RESUMEN– Las herramientas de Infraestructura como código (IaC) permiten automatizar las tareas realizadas por los departamentos de IT de forma rápida y dinámica mediante el uso de lenguajes de programación de *scripts*, que permiten administrar, crear, manipular y distribuir múltiples recursos informáticos a gran escala dentro de una infraestructura de *Cloud Computing*. Las herramientas de Infraestructura como código *Ansible, Terrafom, Chef, Puppet* generan una representación virtual de toda la infraestructura física y escalable de una plataforma *Cloud Computing* y de Centro de Datos, facilitando que esta sea programable y dinámica.

Palabras clave– *Infraestructura como código, Ansible, Terrafom, Chef, Puppet.*

ABSTRACT- Infrastructure tools such as code (IaC) allow the tasks performed by IT departments to be automated quickly and dynamically through the use of scripting languages, which allow managing, creating, manipulating and distributing multiple large-scale computing resources within of Cloud Computing infrastructure. Infrastructure tools such as Ansible code, Terrafom, Chef, Puppet, generate a virtual representation of all the physical and scalable infrastructure of a Cloud Computing and Data Center platform, making it easy to program and dynamic.

Keywords– *Infrastructure as code, Ansible, Terrafom, Chef, Puppet.*

1. Introducción

La Infraestructura como Código o *Infrastructure as Code* (IaC) es la práctica de automatizar tareas de configuración y aprovisionamiento en una infraestructura de Computación en la Nube (*Cloud Computing*) [1] y Centro de Datos mediante el desarrollo de código [2][3][4]. Para ello, toda la infraestructura física es representada de forma virtual, como si fuera un *software* o un dato que se puede manejar de forma dinámica mediante la programación de rutinas *scripts*. Estas rutinas de *scripts* son ejecutadas por aplicaciones conocidas cono *runners* que se encargan de implementar las órdenes codificas en esos *scripts* [5]. En consecuencia, como la infraestructura es 'código programable', podemos administrarla de forma dinámica mediante el uso de herramientas de desarrollo de *software*, tales como los sistemas de control de versiones, pruebas automatizadas, orquestación de recursos, abriendo así, también la posibilidad del uso buenas prácticas de programación, como lo es la integración y despliegue continuo.

Algunas de las herramientas de IaC más reconocidas son: *Ansible* [6], *Terrafom* [2], *Chef* [7] y *Puppet* [8]. Son utilizadas en la configuración y aprovisionamiento de servicios en la nube, así como en los entornos de desarrollo de *software*. Organizaciones de Tecnologías de Información como *GitHub, Mozilla, Amazon* y *Netflix* están usando Iac para automatizar los procesos relacionados con la administración de sistemas escalables de base de datos, recursos de *hardware* bajo

Citación: L. Dominguez y M. Vargas, "Herramientas de infraestructura como código: ansible, terrafom, chef, puppet" *Revista de I+D Tecnológico*, vol. 17, no. 2, pp. (), 2021.
Tipo de artículo: Revisión. **Recibido:** 25 de mayo de 2020. **Recibido con correcciones:** 17 de junio de 2020. **Aceptado:** 12 de enero de 2021.
DOI

demanda, recursos de *software* bajo demanda, cuentas de usuario o generación de certificados digitales [9] o para crear máquinas virtuales para implementar un servidor *Web* o de base de datos distribuidas.

Este trabajo de investigación básica está organizado de la siguiente manera. En la sección 2, se explican los principios de IaC; en la sección 3, se comenta el ciclo de vida IaC; en la 4, presentamos las plataformas dinámicas IaC; la sección 5 aborda el tema de las herramientas IaC, y, finalmente, las conclusiones de esta investigación.

2. Principios de la IaC

Para implementar una infraestructura dinámica y automatizada con herramientas IaC, se requiere que los departamentos de tecnología de información hagan un análisis de su actual infraestructura para elegir la estrategia de IaC adecuada, basándose en los siguientes principios:

- *Los sistemas son recreables*: Capacidad de poder construir y reconstruir una parte o toda una infraestructura las veces que sea necesarias mediante los scripts programados. Por ejemplo, escoger la versión de librerías que un software necesita o el aprovisionamiento de un servidor virtual en la Cloud Computing.
- *Los sistemas deben estar fácilmente disponibles*: Como sabemos, en la IaC, la infraestructura es dinámica por lo que debemos tener la capacidad de poder crearla, destruirla, reemplazarla, redimensionarla y moverla de un lugar a otro. Esto lo permiten solo los sistemas que son adaptable al cambio.
- *Los sistemas son consistentes*: Capacidad de reproducir de forma idéntica elementos de la infraestructura a partir de un mismo *script* de configuración, sin que ello represente ningún tipo de diferencia. Por ejemplo, si creamos 100 servidores virtuales de base de datos que varían entre 40gb – 2 TB y la base de datos *Postgresql* 6.3, cada vez que generemos un servicio nos presentará esta configuración.
- *Los procesos deben poder ser repetibles*: Los sistemas pueden ser reproducidos todas las veces que sea necesario a partir del mismo archivo de *scripts.* Por ejemplo, dentro de un entorno de programación, podemos crear un servidor de pruebas y, en caso de fallos, crear una versión idéntica del servidor de forma fácil y rápida.
- *Los sistemas deben ser diseñados para estar en constante cambio*: La infraestructura de *Cloud Computing* es física y hacer un cambio como aumentar la capacidad de disco duro o el ancho de banda ancha de internet, sin detener la producción, es complejo y caro. Pero una infraestructura virtual de *Cloud Computing* dinámica permite cambiar un sistema existente de forma fácil y rápida. Para que esto sea posible tanto los *softwares* como la infraestructura deben estar diseñados de forma simple.

3. Ciclo de vida IaC

Durante el ciclo de vida de una infraestructura virtual basada en IaC, se realiza las siguientes actividades:

- *Uso de Archivos de Definición (Scripts):* Una de las principales actividades de las herramientas de Infraestructura como Código es la codificación y uso de los archivos de definición. Un archivo de definición especifica qué elementos de la infraestructura serán utilizados y cómo deben ser configurados. Los archivos de definición son la guía que utilizan las herramientas de IaC para realizar aprovisionamiento y/o configuración de los elementos de la infraestructura. Un elemento puede ser uno o varios servidores virtuales, que tipos o versiones de sistemas operativos o cuanta memoria RAMs tendrán.
- *Documentación de los sistemas y los procesos:* En la IaC es necesario documentar los procedimientos realizados en el archivo de definición, en las rutinas programadas, así como capturar los eventos que acontecen durante la ejecución de los *scripts.*
- *Se crean versiones de todos los elementos de la IaC:* Un sistema de control de versiones (SCV) de los archivos *script* o rutinas es el elemento principal de una infraestructura de *Cloud Computing* administrada por código. Los SVC mantienen un control de los estados de los elementos de una infraestructura y, cualquier cambio en los SCV, se refleja en un cambio en la infraestructura virtual de *Cloud Computing*. Un sistema de SCV permite hacer un seguimiento de los cambios en los elementos y en las configuraciones de una infraestructura, hacer *rollback* en caso de un problema en el sistema, disparar *trigger* para realizar determinadas tareas en caso de algún problema en la infraestructura.

- *Se crean sistemas y procesos de prueba continua:* Las pruebas automatizadas son una práctica importante dentro de los procesos de desarrollo de *software*. Cuando la infraestructura de *Cloud Computing* está automatizada, podemos crear procedimientos para la realización de pruebas automatizadas por cambios realizados en el *software* o en sus configuraciones, permitiendo así recibir retroalimentación rápida del comportamiento del *software* ante determinados cambios o fallos que se presenten.
- *Es mejor implementar pequeños cambios, que grandes cantidades:* Cuando estamos realizando cambios en los archivos de *scripts* es aconsejable hacer cambios poco a poco, crear versiones del mismo e ir haciendo pruebas para ver el comportamiento del sistema.
- *Mantener los servicios continuamente disponibles*: Con las herramientas de IaC, se puede generar rutinas que detecten la situación de un servidor o un determinado servicio y, en el caso de que surjan eventualidades o fallos, las rutinas de IaC puedan restablecer los servicios o levantar el servidor o, en caso de un fallo grave, reconstruir toda o parte de la infraestructura de *Cloud Computing*.

4. Plataformas dinámicas IAC

Las herramientas de software de infraestructura como código generan una representación abstracta de los diferentes recursos con los que cuenta la plataforma de *Cloud Computing*. Estos recursos son, a nivel de *hardware* y *software*, el almacenamiento de espacio y las conexiones de red, los cuales deben ser programables, demandables y autoadministrables.

4.1. Características de las plataformas dinámicas

Para que una plataforma como infraestructura[10] sea dinámica y administrada por IaC, debe cumplir con las siguientes características:

- *Ser programable:* La plataforma debe tener herramientas de software de fácil uso o APIS. Algunas herramientas de IaC son: *Ansible* [6], *Terrafom* [2], *Chef* [7] y *Puppet* [8], que tienen sus propias nomenclaturas de *scripts* o lenguaje de programación.
- *Demandable:* Una plataforma dinámica tiene que permitir crear, cambiar o eliminar cualquier elemento de la infraestructura de forma rápida, ya sea en minutos o en milésimas de segundos.
- *Autoadministrable:* El usuario debe tener acceso ya por medio de las herramientas de *software* o APIS de programación y así administrar los recursos de la plataforma sin la necesidad de un asesor externo o departamento de IT, salvo para situaciones muy graves.

4.2. Tipos de recursos de una plataforma dinámica

Como hemos comentado, una plataforma dinámica de *Cloud Computing* está compuesta por diversos recursos que, por lo general, están agrupados de la siguiente manera:

- Computo: Estos recursos son instancias de servidores, conocidos también como servidores virtuales, los cuales deben poder ser creados, cambiados o eliminados con facilidad. Estos servidores virtuales son una representación en *software* de un servidor físico que pue puede tener instalado uno o varios servicios (*Cloud*).
- Almacenamiento: Una plataforma basada en una infraestructura dinámica debe poder asignar recursos de espacio de almacenamiento requeridos de forma fácil, sin que el usuario conozca todos los detalles técnicos de como la información es guardada.
- Redes: Las plataformas de infraestructura dinámica deben contar con la capacidad de administrar y controlar la conectividad de red interna y externa de los diferentes elementos que componen la infraestructura de forma fácil. Esto, por lo general, es a nivel del enrutamiento de redes, balanceo de carga, cambio dinámico de las reglas del cortafuego, agregar o restringir las conexiones de red de un servidor.

Los servicios que ofrecen las plataformas dinámicas se basan en los tres elementos anteriormente mencionados, pero también puede ofrecer variaciones de estos. Como, por ejemplo, un servidor virtual con espacio de almacenamiento asignado y que tenga balanceo de carga de las conexiones de red con otros servidores, servicios de autenticación de usuarios, etc.

5. Herramientas IaC

En el mercado, existen diversas herramientas de Infraestructura como código que permiten a los usuarios asignar y configurar, de forma dinámica, los recursos disponibles dentro de una infraestructura de *Cloud Computing* virtual de manera que podamos

documentarla, recrearla, probarla y reusarla todas las veces que sea necesario, por interface de usuario (GUI), línea de comandos o el uso de API Programables, o usando lenguajes de *scripts* los cuales se asemejan a los lenguajes de programación como es el caso de *Ansible* [6], *Chef* [7] y *Puppet* [8] y el uso de archivos de definición de configuraciones, cuya sintaxis es similar al XML o JSON, como es el caso de *Terrafom* [2].

Si utilizamos una herramienta de IaC para la automatización de procesos basadas en *script,* se debe cumplir con los siguientes requerimientos:

- *Interface Scriptable*: Además de las interfaces de usuario (GUI) que dan muchos proveedores (*Hosting Cloud Computing* y *GPU*), es necesario que la IaC tenga s*oftware* en líneas de comandos, APIS Programables y acceso a código fuente para que los equipos de IT puedan personalizar determinadas funcionalidades o hacer labores de integración con otros tipos de sistemas.
- *Herramientas de comandos con modo desatendido* (Software que puede tener o no interface de usuario GUI): Mediante una interface de línea de comando, debemos poder crear *scripts* por medio de lenguaje de programación de *scripts*, que deben tener accesos a los elementos de la plataforma como variables del entorno, otros comandos vía parámetros, credenciales, licencias, etc.
- *Soporte para modo desentendido*: La mayoría de los administradores deben poder escribir sus *scripts* de rutinas sin la necesidad de intervención humana, durante la ejecución de una tarea o conjunto de ellas de forma automática.
- *Configuración externalizada:* Se debe ofrecer un conjunto de herramientas para la creación, cambio, configuración o eliminación de recursos virtuales en plataformas externas a la nuestra.

Como hemos mencionado, los *scripts* de la IaC deben funcionar en modo desentendido, silencioso o de consola; por lo cual, con el fin de rastrear el funcionamiento de estos *scripts* de programación, estos deben cumplir con los siguientes criterios:

- Ser improcedentes: Debe ser posible ejecutarlo varias veces, sin efectos negativos [11].
- Prerevisables: Se tiene que validar, demostrar si cumple con las condiciones antes de realizar la ejecución del script.
- Posrevisables: Se tiene que generar un mecanismo que notifique si los comandos se ejecutaron de forma correcta.
- Fallos visibles: Cuando una tarea falla esto tiene que ser visible de alguna manera para que el equipo de IT logre solucionarlo.

Por otro lado, al escoger herramientas *software* IaC basadas en archivos de definición de configuración (ADC), hay que considerar que estos son archivos que describen los elementos estáticos y parametrizables de los diversos recursos o elementos de una infraestructura de *Cloud Computing* virtual. Además, tienen su propia nomenclatura de cómo se especifican los elementos y recursos de la infraestructura, esta nomenclatura también conocida como Lenguajes de Especificación de Dominio (*Domain-specific Languages - DSL*) son implementados en *Terrafom, CFEngine, Chef* y *Puppet.* Otras de las ventajas de los ADC es que pueden ser reutilizables y, por medio de un sistema de control de versiones, se tiene acceso a diferentes versiones de una plataforma virtual. Dependiendo del tipo de tareas que serán automatizadas y de los recursos que serán administrados de forma dinámica, se escoge la herramienta de IaC adecuada a implementar, ya sea para creación, configuración, corrida de comandos en servidores o para el manejo de contenedores.

4.3. Configuración de servidores

Para la creación y configuración de servidores con *Ansible, Chef, Puppet*. Este tipo de herramientas usan archivos de definición de configuración para implementar la creación del servidor. Además, muchas de estas herramientas de configuración utilizan un agente instalado en cada servidor. El agente es un *software* que periódicamente busca en un repositorio central, la última versión de los ADC para aplicarlo a la infraestructura de *Cloud Computing* [1] .

4.4. Empaquetamiento de plantilla de servidores

Con el fin de ahorrar tiempo en la implementación de servidores para funciones específicas como Servidor *Web*, Base de Datos, Almacenamiento o Video *Streaming*, existen herramientas como *Aminator* [2] que tienen plantillas para la creación de servidores que empaquetan o agrupan todo el *software* y las configuraciones requeridas para creación de un determinado tipo de servidor.

4.5. Correr comando en un *cluster* de servidores

Correr comandos de forma remota en múltiples servidores es sumamente importante. Para la realización de este tipo de tareas existen herramientas como *MCollective, Fabric* y *Capistrano* [12] que pueden ser usadas para verificar, reparar problemas y también utilizar cualquier lenguaje de *scripts* que vienen con los sistemas operativos.

4.6. Manejo de contenedores

Los contenedores son una alternativa que permite correr aplicaciones a los servidores. Los sistemas de contenerización como *Docke*r [13], *Rocket* [14] y *Windows Containers* son usados para definir y empaquetar todo lo necesario para que un proceso funcione en un archivo de imagen, de esta manera, podemos usar una aplicación determinada sin tener que instalar todo un sistema operativo completo. El contenedor puede ser distribuido, creado y correr diversas instancias a partir de una imagen del contenedor. Cada contenedor aísla los procesos y conexiones de redes hasta el sistema de archivo.

5. Conclusiones

Las herramientas de Infraestructura como Código (IaC) facilitan la administración de los centros de datos y de *Cloud Computing* de diversos proveedores de servicios en *Cloud*. La infraestructura como código tiene como objetivo desarrollar rutinas de *scripts* adaptables a los diversos servicios que una infraestructura como servicios en la *cloud computing* requiere al ofrecer a millones de empresas y usuarios. Con la IaC en la *Cloud Computing* se han construido herramientas de *software* como *Ansible*, *Chef*, *Puppet,* las cuales facilitan la gestión de los grandes esquemas de *cluster* de servidores, contenedores, almacenamientos y conexiones a redes distribuidas.

REFERENCIAS

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-degree compared," *Grid Computing Environments Workshop, GCE 2008*, 2008.

[2] K. Morris, "Infrastructure as code: managing servers in the cloud," *O'Reilly Media, Inc*, p. 362, Jan. 2016.

[3] S. Johann, "Kief Morris on Infrastructure as Code," *IEEE Software*, vol. 34, no. 1, pp. 117–120, 2017.

[4] Y. Jiang and B. Adams, "Co-evolution of infrastructure and source code - An empirical study," *IEEE International Working Conference on Mining Software Repositories*, vol. 2015-Augus, pp. 45–55, 2015.

[5] J. Wettinger, U. Breitenbücher, and F. Leymann, "DevOpSlang - Bridging the gap between development and operations," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8745 LNCS, pp. 108–122, 2014.

[6] N. K. Singh, S. Thakur, H. Chaurasiya, and H. Nagdev, "Automated provisioning of application in IAAS cloud using Ansible configuration management," *Proceedings on 2015 1st International Conference on Next Generation Computing Technologies, NGCT 2015*, no. September, pp. 81–85, 2016.

[7] W. Hummer, F. Rosenberg, F. Oliveira, and T. Eilam, "Automated testing of chef automation scripts," *Proceedings of the Demo and Poster Track of ACM/IFIP/USENIX International Middleware Conference, MiddlewareDPT 2013*, 2013.

[8] S. Dustdar, F. Leymann, and M. Villari, "Service Oriented and Cloud Computing: 4th European Conference, ESOCC 2015 Taormina, Italy, September 15–17, 2015 Proceedings," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9306, no. September, pp. V–VI, 2015.

[9] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, "A systematic mapping study of infrastructure as code research," *Information and Software Technology*, vol. 108, no. November, pp. 65–77, 2019.

[10] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, 2018.

[11] W. Hummer, F. Rosenberg, F. Oliveira, and T. Eilam, "Testing idempotence for infrastructure as code," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8275 LNCS, no. May 2014, pp. 368–388, 2013.

[12] Diomidis Spinellis and Stephanos Androutsellis-Theotokis, "Software Development Tooling," no. December, pp. 21–23, 2014.

[13] C. Boettiger, "An introduction to Docker for reproducible research," *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, Jan. 2015.

[14] Z. Kozhirbayev and R. O. Sinnott, "A performance comparison of container-based technologies for the Cloud," *Future Generation Computer Systems*, vol. 68, pp. 175–182, 2017.