



Universidad
Continental

FACULTAD DE INGENIERÍA

Escuela Académico Profesional de Ingeniería
de Sistema e Informática

Trabajo de Investigación

Influencia del enfoque de infraestructura como código en sistemas de información basados en la nube de Amazon Web Services

Angel Francisco Ybarhuen Manrique

Lima, 2018

Para optar el Grado Académico de
Bachiller en Ingeniería de Sistemas e Informática



Repositorio Institucional Continental

Trabajo de Investigación



Obra protegida bajo la licencia de [Creative Commons Atribución-NoComercial-SinDerivadas 2.5 Perú](https://creativecommons.org/licenses/by-nc-nd/2.5/peru/)

AGRADECIMIENTOS

Agradezco en primer lugar a Dios por haberme permitido vivir en este siglo donde el acceso al conocimiento está limitado por la voluntad del hombre. Así mismo, brindar mi muestra de gratitud a mi familia, centro laboral y asesor de tesis quienes han dispuesto de su bien más preciado que es su tiempo para que este trabajo pueda ser realizado.

DEDICATORIA

Este trabajo de investigación va dedicado mis padres por haberme brindarme su apoyo a lo largo de mi carrera profesional.

CONTENIDO

CAPÍTULO I	1
PLANTEAMIENTO DEL ESTUDIO	1
1.1 Planteamiento y formulación del problema	1
1.2 Objetivos	2
1.3 Justificación e importancia	2
1.4 Hipótesis	3
1.5 Descripción de las variables	4
CAPÍTULO II	5
MARCO TEÓRICO	5
2.1 Antecedentes del problema	5
2.2 Bases teóricas	6
2.3 Definición de términos básicos	9
CAPÍTULO III	12
METODOLOGÍA	12
3.1 Métodos de la investigación	12
3.2 Alcance de la investigación	12
3.3 Diseño de la investigación	12
3.4 Técnica e instrumento de recolección de datos	14
CAPÍTULO IV	17
RESULTADOS Y DISCUSIÓN	17
4.1 Resultados del tratamiento y análisis de la información	17
4.2 Discusión de resultados	18
CONCLUSIONES	20
REFERENCIAS BIBLIOGRÁFICAS	21
ANEXOS	22

RESUMEN

El objetivo del presente trabajo de investigación es identificar la influencia del enfoque de infraestructura como código en los procesos de aprovisionamiento y administración de la infraestructura tecnológica para un sistema de información basado en la nube de Amazon Web Services. Para ello se ha definido la arquitectura de un sistema de gestión de contenidos el cual será implementado sobre la nube de Amazon Web Services por medio del enfoque de infraestructura como código. Así mismo, para las actividades de recolección de datos se procedió a elaborar una batería de pruebas de rendimiento que permitan comprobar las hipótesis y medir la relación entre las variables propuestas.

INTRODUCCIÓN

La presente investigación busca identificar la influencia del uso del enfoque de infraestructura como código (IaC) en los sistemas de información basados en la nube de Amazon Web Services. Esto se logrará a través del uso de las herramientas de Terraform y Ansible para desarrollar y poner en marcha dicho enfoque en un Sistema de Gestión de Contenidos como objeto del estudio.

Por otro lado, la importancia de este trabajo es determinar de qué manera el enfoque IaC afecta dentro de los procesos de aprovisionamiento y mantenimiento de infraestructura tecnológica para un sistema de información basado en la nube de Amazon Web Services.

Complementariamente, el paradigma de computación en la nube ha permitido desarrollar nuevas formas de implementar las arquitecturas de sistemas de información en cuanto a sus componentes de infraestructura. Es por ello que la importancia de investigar uno de los enfoques que actualmente se utilizan dentro de las industrias, IaC, aportará un valor académico significativo al campo de la investigación aplicada de infraestructura tecnológica.

CAPITULO I

PLANTEAMIENTO DEL ESTUDIO

1.1 Planteamiento y formulación del problema

De acuerdo a la IEEE en su artículo titulado “The Case for Cloud Computing” (2009), los proveedores de computación en la nube, permiten la entrega bajo demanda de capacidad de almacenamiento, cómputo, servicios de aplicación y otros recursos de TI por medio de internet, generando impacto en la innovación y la economía de los negocios en general y que basándose en el modelo “pay-as-you-go”, brinda a sus clientes la capacidad de pagar solo por los recursos consumidos.

Ahora bien, en base a la documentación del proveedor Amazon Web Services, los recursos de TI provistos en la nube son provisionados y administrados en forma manual por medio de la consola web. Complementariamente, la capacidad de las APIs REST de AWS han generado la posibilidad de desarrollar herramientas que implementen el enfoque de Infraestructura como Código (IaC). En este sentido, los sistemas de información que implementan el enfoque de IaC se pueden beneficiar de aprovisionamiento y administración sobre los recursos de TI de manera más sencilla y rápida. Kief Morris (2016), consultor destacado de la empresa internacional Thoughtworks, en su libro Infrastructure as Code “La nueva generación de tecnologías de administración de infraestructura promete transformar la forma en que administramos la infraestructura de TI.”

Por otro lado, de acuerdo a Kief Morris (2016), el aprovisionar y administrar la infraestructura sobre la nube de manera manual implica lidiar con configuraciones inconsistentes en cada componente, problemas al replicar las arquitecturas de sistema en

diferentes entornos, dificultad de escalar determinados componentes (generalmente servidores) en corto tiempo y poder reconstruir la infraestructura con versiones anteriores para probar determinados comportamientos de las aplicaciones de software. Ahora bien, un ejemplo de esta problemática es documentada por Raymond E. Suorsa quien realizando un trabajo para la empresa Hewlett-Packard, Automated provisioning framework for internet site servers, indica que “Quizás la mas significativa limitación asociada con el aprovisionamiento manual de infraestructura es la falta de repetibilidad en las configuraciones. Particularmente, las operaciones manuales involucradas con la instalación del software, tienen siempre la posibilidad de error humano, como la falla al instalar uno de los componentes requeridos, o la carga de varios elementos de software en el orden incorrecto. De tal manera que los errores pueden provocar un mal funcionamiento o una falla total en un sistema que puede ser extremadamente lenta de descubrir y corregir.”

Finalmente, el problema radica en ¿Cómo mejorar el nivel de operabilidad y maniobrabilidad de los recursos de TI en sistemas de información que usan la nube de Amazon Web Services para su implementación a través del enfoque IaC?

1.2 Objetivos

1.2.1 Objetivo general

Identificar la influencia que tiene el uso del enfoque de Infraestructura como Código en sistemas de información basados en la nube de Amazon Web Services durante el 2018.

1.2.2 Objetivo específicos

1. Determinar la influencia del enfoque de Infraestructura como Código a través del uso de las herramientas Terraform y Ansible en sistemas que empleen Amazon Web Services como proveedor de servicios.

2. Determinar el rendimiento del proceso de expansión de servidores y gestión de configuraciones de los componentes para sistema de información basadas en la nube de Amazon Web services a través del enfoque de Infraestructura como Código.
3. Determinar las diferencias de tiempo entre procesos de aprovisionamiento de infraestructura tecnología por medio de IaC y el aprovisionamiento de forma manual para sistemas de información basadas en la nube de Amazon Web services.

1.3 Justificación e importancia

Los recursos de cómputo sobre la nube y su uso para implementar sistemas de información, juegan un papel muy importante dentro de las organizaciones que alinean su valor de negocio con las TI hoy en día. Seguidamente, la transformación organizacional en las áreas de TI mediante el auge de los principios DevOps, han impulsado el enfoque de la Infraestructura como Código como una práctica dentro de las actividades de los equipos de desarrollo y operaciones para alcanzar el máximo beneficio.

David Linthicum (2015) en su artículo titulado *Beyond devops: Embrace infrastructure as code*. Indica que el beneficio del enfoque de la Infraestructura como Código aplicado a la administración de los sistemas de información genera dentro de las organizaciones la reducción de recursos de gestión, operaciones y personal de tecnología gracias a que los equipos de desarrollo y operaciones pueden compartir las capacidades de configurar y administrar dinámicamente las plataformas tecnológicas.

Andy Patrizio (2015) en su artículo *What is infrastructure as code; and why should you embrace it?*. Menciona que el uso de las herramientas IaC hacen que los procesos de aprovisionamiento y administración de infraestructura sobre la nube se realicen más rápido

eliminando los riesgos provocados por el factor humano, una vez que se consigue una maduración del enfoque mediante estructuras sólidas de código.

Por lo tanto, el presente trabajo de investigación tiene como motivación demostrar por medio de la investigación y experimentación, a través de la implementación de una arquitectura de sistemas de información desplegada sobre la nube de AWS, los beneficios que se pueden obtener al implementar el enfoque de Infraestructura como Código con la finalidad cuantificar de manera porcentual dichos beneficios.

1.4 Hipótesis

1.4.1 Hipótesis general

El enfoque de Infraestructura como Código afecta de manera positiva en el aprovisionamiento y la administración de los sistemas de información basados en la nube de Amazon Web Services.

1.4.2 Hipótesis específicas

- El uso del enfoque de Infraestructura como Código a través de las herramientas Terraform y Ansible en sistemas que empleen Amazon Web Services como proveedor de servicios influyen de manera positiva el despliegue de infraestructura.
- El enfoque de Infraestructura como Código influye de manera positiva al proceso de expansión de servidores y gestión de configuraciones de los componentes para sistema de información basadas en la nube de Amazon Web services.
- El enfoque de Infraestructura como Código influye de manera positiva en el tiempo empleado para el aprovisionamiento de infraestructura tecnología en sistemas de información basadas en la nube de Amazon Web services.

1.5 Descripción de las variables

1.5.1 Variable independiente

El enfoque de Infraestructura como Código. Se expresará a través del grado de implementación (bajo, medio y alto) dado por el uso de diversas herramientas tecnológicas. Seguidamente los indicadores estarán definidos por el cumplimiento de los principios del enfoque IaC que son modularidad, cooperación, composición, extensibilidad, flexibilidad, repetibilidad, declaración, abstracción, idempotencia y convergencia.

1.5.2 Variable dependiente

El rendimiento operacional de las tareas de infraestructura tecnológica sobre la nube. Se expresará a través de un grado porcentual (1% al 100%) de eficiencia de las tareas realizadas de por medio del enfoque IaC en contraste con las hechas de forma manual. Para la medición de indicadores se procede a elaborar una batería de pruebas relacionadas a las tareas de aprovisionamiento de infraestructura, configuración de aplicación, mantenimiento de infraestructura y replicación de ambientes.

CAPITULO II

MARCO TEÓRICO

2.1 Antecedentes del problema

- **Jens Segers (2013)** en su tesis titulada : “*Centralized management and monitoring of embedded devices/routers*” y luego de compartir un artículo en su blog personal acerca del mismo. Menciona que el uso de herramientas basadas en el paradigma de IaC mejora la relación entre el diseño y el trabajo de los equipos de operaciones con respecto a los recursos desplegados en la nube, ganando tiempo durante la ejecución de las tareas.

- **Yujuan Jiang (2016)** en su tesis titulada : “*Mining Software Repositories For Release Engineers - Empirical Studies On Integration And Infrastructure-as-code*”. Concluye que el enfoque IaC ayuda a automatizar el proceso de configuración de entornos donde serán desplegadas las aplicaciones de software que soportan los sistemas de información lo cual ayuda a simplificar la configuración, el despliegue y reducir los riesgos del trabajo manual con respecto a la gestión de la infraestructura.

- **Fabian Vloger (2015)** en su tesis titulada : “*Code Driven Infrastructure and Deployment*”. Concluye que el uso de la herramienta Ansible ayuda a implementar el enfoque IaC haciendo que la infraestructura sea más estructurada generando visibilidad a los involucrados con respecto a los cambios de la misma mientras se mantiene la flexibilidad para futuros desarrollos. Seguidamente, se señala que una de las características de este enfoque conocida como “*idempotencia*” está en función del tipo de herramientas que se usen.

2.2 Bases teóricas

2.2.1 DevOps

De acuerdo a Sam Guckenheimer (2016), es un acrónimo que significa Development and Operations y que traducido al español es Desarrollo y Operaciones. Definida como la combinación de filosofías, prácticas y herramientas culturales que aumentan la capacidad de trabajo de una organización TI con la finalidad de entregar aplicaciones y servicios a gran velocidad de manera tal que se mejoren los productos a un ritmo más rápido que las organizaciones que utilizan procesos tradicionales de desarrollo de software y gestión de infraestructura.

2.2.2 Infraestructura como Código (IaC)

De acuerdo a Kief Morris (2016), Acrónimo que significa Infrastructure as a Code que traducido al español es Infraestructura como Código. Definida como la gestión de la infraestructura (redes, máquinas virtuales, balanceadores de carga y topología de conexión) en un modelo descriptivo, realizada a través de código fuente por un equipo DevOps.

2.2.2.1 Desafíos del enfoque IaC

De acuerdo a Kief Morris (2016), se entiende como los problemas que se tratan de resolver por medio de este enfoque. Estos son los siguientes.

2.2.2.1.1 Expansión de servidores

Problema que surge cuando es necesario escalar miles de componentes (en especial servidores) en plataformas sobre la nube y no se puede controlar la uniformidad de sus configuraciones.

2.2.2.1.2 Derivaciones de configuración

Problema que surge cuando es necesario tener configuraciones consistentes de los componentes (en especial servidores) mientras estos van cambiando en el tiempo.

2.2.2.1.3 Servidores de copo de nieve

Problema que surge cuando es necesario replicar configuraciones de componentes (en especial servidores) para poder realizar pruebas de comparación y comportamiento de aplicaciones.

2.2.2.1.4 Infraestructura frágil

Problema que surge cuando surge la necesidad de migrar toda la arquitectura de infraestructura hacia un nuevo ambiente contemplando las configuraciones.

2.2.2.2 Principios del enfoque IaC

Adam Jacob, director de tecnología en la empresa Chef señala que los principios dentro de este enfoque son los siguientes.

2.2.2.2.1 Modularidad

Los componentes que implementan el enfoque deben ser pequeños, simples, independientes y útiles.

2.2.2.2.2 Cooperación

El diseño del enfoque debe fundamentarse en la colaboración de personas y servicios a utilizar en función a la mejora continua.

2.2.2.2.3 Composición

Los servicios que implementen el enfoque deben ser construidos como bloques de construcción que permitan crear complejidad sistémica.

2.2.2.2.4 Extenbilidad

Los servicios que implementen el enfoque deben ser fáciles de modificar y mejorar en el tiempo.

2.2.2.2.5 Flexibilidad

Se deben emplear herramientas que brindan versatilidad para garantizar la implementación de las bases teóricas del enfoque.

2.2.2.2.6 Repetibilidad

Los servicios que implementan el enfoque deben producir los mismos resultados, de la misma manera, con las mismas entradas, todo el tiempo.

2.2.2.2.7 Declaración

Los servicios deben de especificar las funciones que se deben cumplir descartando el cómo queremos hacerlo.

2.2.2.2.8 Abstracción

Al realizar el enfoque se deben contemplar los detalles de la implementación, y pensar a nivel del componente y su función.

2.2.2.2.9 Idempotencia

Los servicios que implementan el enfoque solo deben configurarse cuando sea necesario e idealmente la acción sólo debe tomarse una vez.

2.2.2.2.10 Convergencia

Los servicios que implementan el enfoque deben de responsabilizarse de su propio estado esté en el tiempo.

2.3 Definición de términos básicos

2.3.1 Cloud computing

Carl Hewitt (2008) *IEEE*. Es un paradigma en el que la información se almacena constantemente en servidores en Internet y se almacena en caché temporalmente en clientes que incluyen escritorios, centros de entretenimiento, computadoras de mesa, computadoras portátiles, computadoras de pared, dispositivos de bolsillo, sensores, monitores, etc.

2.3.2 AWS

Amazon Web Services. Es un proveedor mundial de recursos y servicios de cloud computing.

2.3.3 IAM

Servicio de AWS, Identity and Access Management (IAM). Permite controlar de forma segura el acceso de los usuarios a servicios y recursos de AWS.

2.3.4 EC2

Servicio de AWS, Amazon Elastic Compute Cloud (EC2). Proporciona capacidad informática segura y de tamaño variable en la nube.

2.3.5 AMI

Amazon Machine Image (AMI) proporciona la información necesaria para iniciar una instancia EC2.

2.3.6 RDS

Servicio de AWS, Relational Database Service (RDS). Permite la configuración, el funcionamiento y la ampliación de una base de datos relacional en la nube.

2.3.7 S3

Servicio de AWS, Simple Storage Service (S3). Proporciona un servicio de almacenamiento sobre la nube que permite almacenar y recuperar cualquier volumen de datos desde cualquier ubicación: sitios web y aplicaciones móviles, aplicaciones corporativas y datos de sensores.

2.3.8 VPC

Servicio de AWS, Virtual Private Cloud (VPC). Permite aprovisionar una sección aislada lógicamente de la nube de AWS donde puede iniciar recursos de AWS en una red virtual que se defina.

2.3.9 Cloudfront

Servicio de AWS de red de entrega de contenido (CDN) global que proporciona datos, vídeos, aplicaciones y API de forma segura a sus espectadores con baja latencia y altas velocidades de transferencia.

2.3.10 ELB

Servicio de AWS, Elastic Load Balancing (ELB). Permite automáticamente la distribución del tráfico de aplicaciones entrantes a través de múltiples destinos, como instancias EC2, contenedores y direcciones IP.

2.3.11 Route 53

Servicio de AWS que implementa el sistema de nombres de dominio (DNS) en la nube altamente disponible y escalable.

2.3.12 API REST

Interface de aplicación programable basada en el enfoque de Transferencia de Estado Representacional (REST) para invocar funcionalidades de servicios terceros a través de la web.

2.3.13 CLI

Interfaz de línea de comandos.

2.3.14 Git

Git es un sistema de control de versiones distribuidas de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia.

2.3.15 Nginx

Servidor web / proxy inverso, ligero y de alto rendimiento.

2.3.16 Wordpress

Es un sistema de gestión de contenidos o CMS (Content Management System) enfocado a la creación de cualquier tipo de página web dinámica.

2.3.17 Terraform

Terraform es una herramienta que permite crear, cambiar y mejorar la infraestructura de manera segura y predecible.

2.3.18 Ansible

Ansible es un motor de orquestación de TI que automatiza la administración de configuraciones, la implementación de aplicaciones y muchas otras necesidades de TI.

2.3.19 EBS

Bloque elástico de almacenamiento. Término acuñado por AWS para referirse a unidades de almacenamiento virtuales.

CAPÍTULO III

METODOLOGÍA

3.1 Métodos de la investigación

El siguiente trabajo de investigación usará el siguiente método investigación.

3.1.1 Método inductivo

Usado para analizar los datos y llegar a las conclusiones acerca del uso del enfoque de Infraestructura como Código en los sistemas de información basados sobre Amazon Web Services.

3.2 Alcance de la investigación

Existen diversas arquitecturas de sistemas de información que pueden desplegarse en la nube de Amazon Web Services. Por lo tanto, el alcance del presente trabajo se centra en el diseño de una arquitectura web desplegada sobre la nube para un sistema de información de gestión de contenidos que permita desarrollar el aprovisionamiento, administración y evolución de la infraestructura por medio del enfoque de Infraestructura como Código mediante las herramientas de Terraform y Ansible.

3.3 Diseño de la investigación

El siguiente trabajo de investigación considera el siguiente diseño de investigación.

3.3.1 Población

Todas las arquitecturas de sistemas de información que se pueden implementar en la nube de Amazon Web Services.

3.3.2 Muestra

Arquitectura web para un sistema de información de gestión de contenidos implementado mediante Wordpress CMS.

3.3.3 Diseño de la muestra

Para el sistema propuesto se usará la arquitectura definida en la *figura 1*.

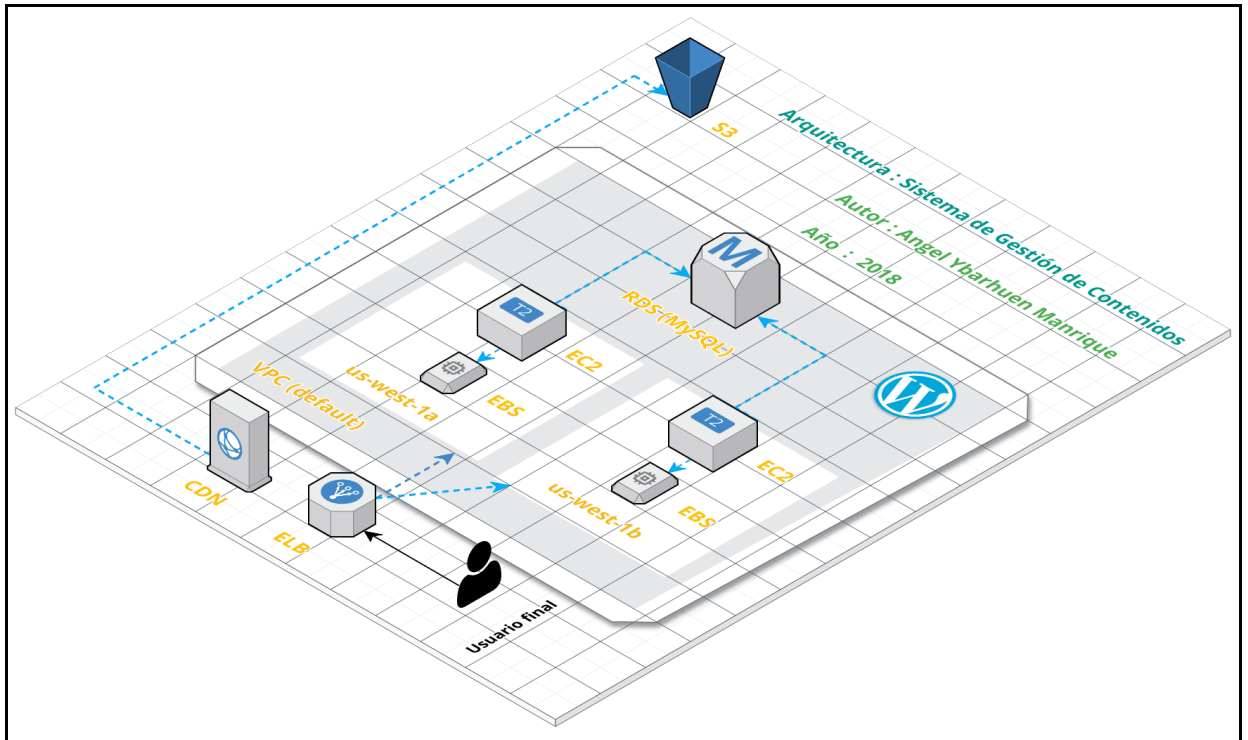


Figura 1. Arquitectura del Sistema de Gestión de Contenidos

Complementariamente, los costos involucrados para la implementación de la arquitectura planeada en la *figura 1* se detalla en la *tabla 1*.

Componente	Costo mensual (\$)	Costo aproximado x hora (\$)
Computación	16.70	0.023
Almacenamiento	10.00	0.013
Red	18.08	0.025
Base de datos	24.48	0.034
Costo total (\$)	69.26	0.095

Tabla 1. Costos para la arquitectura inicial del Sistema de Gestión de Contenidos

Por otro lado, tenemos el siguiente costo extra para los recursos de cómputo en pruebas de escalamiento.

Componente	Costo mensual (\$)	Costo aproximado x hora (\$)
Computación	41.75	0.057

Tabla 2. Costo adicional para pruebas de escalamiento

3.4 Técnica e instrumento de recolección de datos

El siguiente trabajo de investigación propone la obtención de información por medio de una batería de pruebas de rendimiento para elaborar los ensayos del enfoque IaC.

3.4.1 Tiempo de aprovisionamiento de infraestructura

Busca obtener resultados del tiempo que se emplea para aprovisionar la infraestructura sobre la nube de manera manual en contraste con el enfoque IaC.

Aprovisionamiento de componentes de infraestructura		
Componente aprovisionado	Tiempo empleado (minutos)	
	Manual	Enfoque IaC
IAM - acceso a servicios		
VPC - topología de red		
ELB - balanceo de carga		
EC2 (x2) - aplicación		
RDS - datos		
S3 - contenidos		
CDN - red de contenidos		
Tiempo total (minutos)		

Tabla 3. Tiempo de aprovisionamiento de infraestructura

3.4.2 Tiempo de configuración de aplicación

Busca obtener resultados del tiempo que se emplea para configurar la aplicación de software sobre la infraestructura disponible de manera manual en contraste con el enfoque IaC.

Aprovisionamiento de configuración de aplicación		
Actividad	Manual	Enfoque IaC
Gestión de archivos de configuración		
Desarrollo de variables de entorno		
Empaquetado de aplicación		
Sincronización de aplicación		
Tiempo total (minutos)		

Tabla 4. Tiempo de aprovisionamiento de aplicación

3.4.3 Tiempo de mantenimiento de infraestructura

Busca obtener resultados del tiempo que se emplea para realizar cambios en la arquitectura de los componentes de infraestructura sobre la nube de manera manual en contraste con el enfoque IaC.

Mantenimiento de infraestructura		
Actividad	Manual	Enfoque IaC
Escalado de un servidor EC2 más		
Escalado de dos servidores EC2 más		
Escalado de tres servidores EC2 más		
Tiempo total (minutos)		

Tabla 5. Tiempo de mantenimiento de infraestructura

3.4.4 Tiempo de replicación de ambientes

Busca obtener resultados del tiempo que se emplea para realizar tareas de portabilidad de infraestructura sobre la nube de manera manual en contraste con el enfoque IaC.

Replicación de infraestructura		
Actividad	Manual	Enfoque IaC
Generar un nuevo ambiente en otra región		
Generar ambientes con versiones inferiores		
Generar ambientes de prueba		
Tiempo total (minutos)		

Tabla 6. Tiempo de replicación de infraestructura

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1 Resultados del tratamiento y análisis de la información

Luego de realizar la implementación de la muestra diseñada, recopilar los tiempos de las actividades involucradas y analizar los datos obtenidos (*ver anexo 03*) se tienen los siguientes resultados.

4.1.1 Análisis de datos y resultados

Aprovisionamiento de componentes de infraestructura	
Manual	Enfoque IaC
4.5 minutos	7 minutos
El enfoque IaC invierte un 55.55% más de tiempo que el proceso manual	
Aprovisionamiento de configuración de aplicación	
Manual	Enfoque IaC
6.3 minutos	12.9 minutos
El enfoque IaC invierte un 104.76% más de tiempo que el proceso manual	
Mantenimiento de infraestructura	
Manual	Enfoque IaC
22.3 minutos	2.25 minutos
El enfoque IaC disminuye en un 89.91% el tiempo de mantenimiento	
Replicación de infraestructura	
Manual	Enfoque IaC
89.6 minutos	16.6 minutos
El enfoque IaC disminuye en un 81.47% el tiempo de mantenimiento	

Tabla 7. Análisis de tiempo en actividades para la arquitectura propuesta

4.2 Discusión de resultados

Se realizaron dos implementaciones (*ver anexo 04 al 13*) para la arquitectura propuesta (revisar el punto 3.3.3) una de manera manual, por medio de la consola web de Amazon, y la otra por medio del enfoque de la infraestructura como código a través de las herramientas de Terraform y Ansible.

Por un lado, el aprovisionamiento de la infraestructura en la nube mediante Terraform considero, por medio de la herramienta, el almacenado del estado de la infraestructura de manera local (se puede configurar de manera distribuida) , es decir los metadatos de los componentes creados en archivos de texto plano (*ver anexo 02 y 15*). Seguidamente, se usó el concepto de módulos en terraform lo cual permitió re-utilizar las definiciones de infraestructura en otros entornos y regiones de Amazon.

Por otro lado, el aprovisionamiento de la aplicación (Wordpress) mediante Ansible considero el uso de variables (*ver anexo 01 y 14*) las cuales ayudaron a poder escoger diferentes versiones del sistema con la finalidad de realizar pruebas de regresión de estados. Así mismo, el uso de Ansible demostró gran flexibilidad al realizar los comandos necesarios que son usados por medio del sistema operativo.

Para validar el tiempo consumido por cada actividad se adoptó el uso de una aplicación móvil de cronómetro considerando solo los tiempos de ejecución ya que, Amazon Web Services toma un tiempo variable al crear los recursos, por ejemplo, las instancias EC2 demoran en crearse alrededor de 1 a 2 minuto, los contenedores S3 demoran en crearse alrededor de 1 a 3 segundos, la base de datos RDS toma alrededor de 5 a 10 minutos, etc.

Luego de ser verificado el correcto funcionamiento del sistema, se procedió a realizar los procesos de eliminación de componentes de la cuenta de Amazon Web Services para evitar

cargos por costos y salimos del presupuesto calculado. Cabe resaltar que aunque no se considero dentro de las tablas de medición (revisar el punto 3.3.3) la actividad de eliminar componentes de infraestructura, se observó que los procesos de eliminación manual demoraron más que los procesos lanzados por Terraform ya que, solo basto usar el comando *terraform destroy*.

Finalmente, se evidencia que el uso del enfoque IaC en el aprovisionamiento y administración de infraestructura influencia en los tiempos de las actividades relacionadas, esta influencia es positiva en tareas de mantenimiento y replicación de infraestructura. Sin embargo, es importante resaltar que el enfoque de IaC por medio de las herramientas señaladas en el presente trabajo añade una capa de complejidad reflejada en el tiempo empleado al aprovisionar los componentes de manera manual contra las del enfoque.

CONCLUSIONES

1. De las pruebas de rendimiento realizadas se concluye que la implementación del enfoque IaC a través de las herramientas Terraform y Ansible ha influenciado de manera positiva en sistemas que empleen Amazon Web Services como proveedor de servicios ayudando a disminuir los tiempos de mantenimiento y replicación de infraestructura tecnológica en más del 50% aproximadamente.
2. De las pruebas de rendimiento realizadas se concluye que la implementación del enfoque IaC ha influido de manera positiva al proceso de expansión de servidores y gestión de configuraciones de los componentes para sistemas de información basados en la nube de Amazon Web services ayudando a disminuir el tiempo de respuesta de dichas actividades hasta en un 50% aproximadamente.
3. De las pruebas de rendimiento realizadas se concluye que la implementación del enfoque IaC no ha influido de manera positiva en el tiempo empleado para el aprovisionamiento de infraestructura tecnología en sistemas de información basadas en la nube de Amazon Web services ya que, hubo un incremento de tiempo en las tareas de aprovisionamiento en contraste con las labores manuales aumentando en aproximadamente 50%.

REFERENCIAS BIBLIOGRÁFICAS

- **DAVID LINTHICUM.** *Beyond devops: Embrace infrastructure as code.* InfoWorld: 2015. Disponible en : ProQuest.
- **ANDY PATRIZIO.** *What is infrastructure as code; and why should you embrace it?.* CIO: 2015. Disponible en : ProQuest.
- **JENS SEGERS.** *Centralized management and monitoring of embedded devices/routers.* Belgica, 2013. Comentario disponible en : <https://jenssegers.com/44/infrastructure-as-code>
- **YUJUAN JIANG.** *Mining Software Repositories For Release Engineers - Empirical Studies On Integration And Infrastructure-as-code.* Canada, 2016. Disponible en : http://mcis.polymtl.ca/publications/2016/thesis_jojo.pdf
- **FABIAN VOGLER.** *Code Driven Infrastructure and Deployment.* Suiza, 2015. Disponible en : <http://fabian.github.io/code-driven-infrastructure/>
- **STEPHEN NELSON-SMITH.** *Test-Driven Infrastructure with Chef.* USA: O'Reilly, 2010. 978-1449372200
- **KIEF MORRIS.** *Infrastructure as Code: Managing Servers in the Cloud.* USA: O'Reilly, 2016. 978-1491924358

- **STAWEK LIGUS.** *Monitoring and Alerting*. USA: O'Reilly, 2013. 978-1449333522
- **JENNIFER DAVIS & KATHERINE DANIELS.** *Effective DevOps*. USA: O'Reilly, 2016. 978-1491926307
- **UC BERKELEY.** *A view of cloud computing*. USA, 2010. Disponible en :
<https://dl.acm.org/citation.cfm?id=1721672>

ANEXOS

ANEXO 01 : Instalación de Terraform en Ubuntu

```
angel@angel-desktop: ~  
angel@angel-desktop: ~ 80x24  
angel@angel-desktop:~$ cat hashicorp.sh  
#!/bin/sh  
TOOL_NAME="terraform"  
ZIP_URL="https://releases.hashicorp.com/terraform/0.11.3/terraform_0.11.3_linux  
amd64.zip?_ga=2.23495354.1620781409.1518540312-1203024832.1518540312"  
mkdir /tmp/$TOOL_NAME  
cd /tmp/$TOOL_NAME  
curl -sS $ZIP_URL > $TOOL_NAME.zip  
unzip $TOOL_NAME.zip  
sudo cp $TOOL_NAME /usr/local/bin/$TOOL_NAME  
rm -rf /tmp/$TOOL_NAME  
echo "$TOOL_NAME :: is installed successfully"  
angel@angel-desktop:~$ terraform --version  
Terraform v0.11.0
```

Fuente: Elaboración propia.

ANEXO 02 : Instalación de Ansible en Ubuntu

```
angel@angel-desktop: ~  
angel@angel-desktop: ~ 80x24  
angel@angel-desktop:~$ cat ansible.sh  
#!/bin/bash  
sudo apt-get update  
sudo apt-get install software-properties-common  
sudo apt-add-repository ppa:ansible/ansible  
sudo apt-get update  
sudo apt-get install ansible  
angel@angel-desktop:~$ ansible --version  
ansible 2.2.2.0  
  config file = /etc/ansible/ansible.cfg  
  configured module search path = Default w/o overrides  
angel@angel-desktop:~$
```

Fuente: Elaboración propia.

ANEXO 03 : Resultado de las pruebas

Aprovisionamiento de componentes de infraestructura		
Componente provisionado	Tiempo empleado (minutos)	
	Manual	Enfoque IaC
IAM - acceso a servicios	3.5	4.5
VPC - topología de red	4.5	7
ELB - balanceo de carga	2.5	2
EC2 (x2) - aplicación	3.4	5

RDS - datos	4	8.4
S3 - contenidos	0.8	1.5
CDN - red de contenidos	5	8.9
Tiempo total (minutos)	23.7	37.3

Fuente: Elaboración propia.

Aprovisionamiento de configuración de aplicación		
Actividad	Manual	Enfoque IaC
Gestión de archivos de configuración	1	2.5
Desarrollo de variables de entorno	1.4	3.4
Empaquetado de aplicación	2.5	5
Sincronización de aplicación	1.4	2
Tiempo total (minutos)	6.3	12.9

Fuente: Elaboración propia.

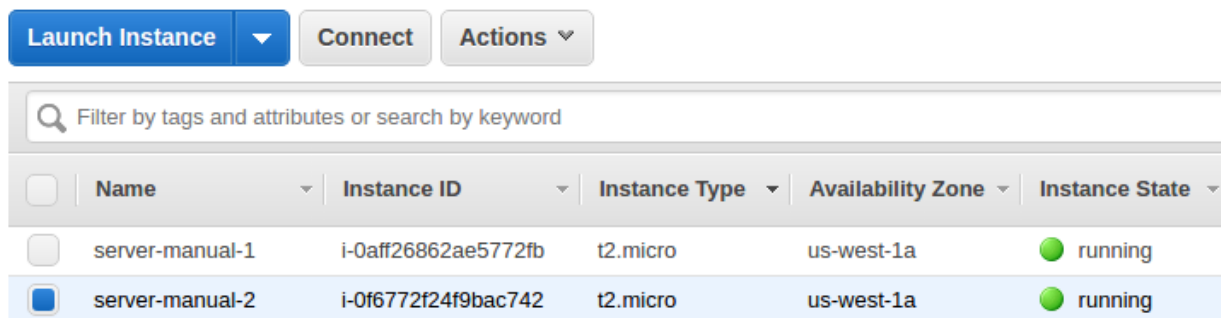
Mantenimiento de infraestructura		
Actividad	Manual	Enfoque IaC
Escalado de un servidor EC2 más	6	0.75
Escalado de dos servidores EC2 más	7.8	0.75
Escalado de tres servidores EC2 más	8.5	0.75
Tiempo total (minutos)	22.3	2.25

Fuente: Elaboración propia.

Replicación de infraestructura		
Actividad	Manual	Enfoque IaC
Generar un nuevo ambiente en otra región	23.7	4
Generar ambientes con versiones inferiores	40.2	7.5
Generar ambientes de prueba	25.7	5.1
Tiempo total (minutos)	89.6	16.6

Fuente: Elaboración propia.

ANEXO 04 : Servidores de aplicación (EC2) creados manualmente



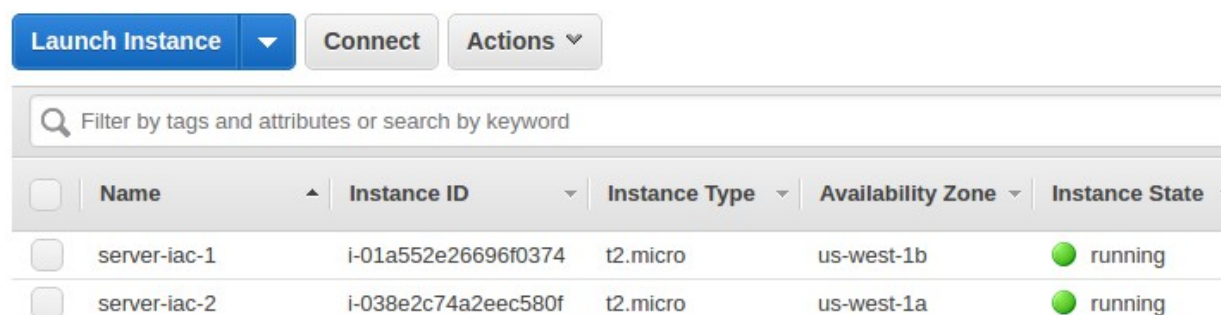
Launch Instance ▾ Connect Actions ▾

Filter by tags and attributes or search by keyword

<input type="checkbox"/>	Name ▾	Instance ID ▾	Instance Type ▾	Availability Zone ▾	Instance State ▾
<input type="checkbox"/>	server-manual-1	i-0aff26862ae5772fb	t2.micro	us-west-1a	● running
<input checked="" type="checkbox"/>	server-manual-2	i-0f6772f24f9bac742	t2.micro	us-west-1a	● running

Fuente: Elaboración propia.

ANEXO 05 : Servidores de aplicación (EC2) creados por IaC mediante Terraform



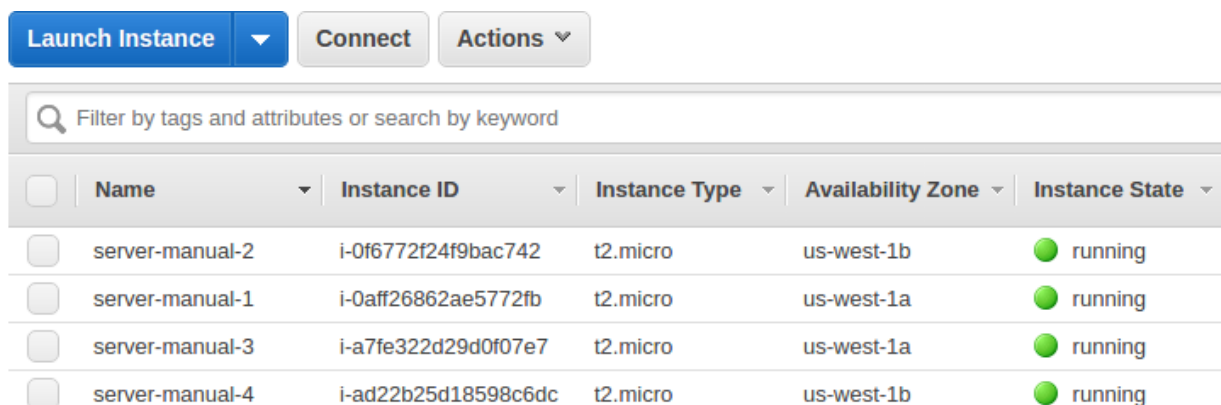
Launch Instance ▾ Connect Actions ▾

Filter by tags and attributes or search by keyword

<input type="checkbox"/>	Name ▾	Instance ID ▾	Instance Type ▾	Availability Zone ▾	Instance State ▾
<input type="checkbox"/>	server-iac-1	i-01a552e26696f0374	t2.micro	us-west-1b	● running
<input type="checkbox"/>	server-iac-2	i-038e2c74a2eec580f	t2.micro	us-west-1a	● running

Fuente: Elaboración propia.

ANEXO 06 : Escalamiento de servidores de aplicación (EC2) manualmente



Launch Instance ▾ Connect Actions ▾

Filter by tags and attributes or search by keyword

<input type="checkbox"/>	Name ▾	Instance ID ▾	Instance Type ▾	Availability Zone ▾	Instance State ▾
<input type="checkbox"/>	server-manual-2	i-0f6772f24f9bac742	t2.micro	us-west-1b	● running
<input type="checkbox"/>	server-manual-1	i-0aff26862ae5772fb	t2.micro	us-west-1a	● running
<input type="checkbox"/>	server-manual-3	i-a7fe322d29d0f07e7	t2.micro	us-west-1a	● running
<input type="checkbox"/>	server-manual-4	i-ad22b25d18598c6dc	t2.micro	us-west-1b	● running

Fuente: Elaboración propia.

ANEXO 07 : Escalamiento de servidores de aplicación (EC2) por IaC con Terraform

Launch Instance

Filter by tags and attributes or search by keyword

<input type="checkbox"/>	Name	Instance ID	Instance Type	Availability Zone	Instance State
<input type="checkbox"/>	server-iac-1	i-01a552e26696f0374	t2.micro	us-west-1b	● running
<input type="checkbox"/>	server-iac-2	i-038e2c74a2eec580f	t2.micro	us-west-1a	● running
<input type="checkbox"/>	server-iac-3	i-05b2b25d18598c6dc	t2.micro	us-west-1b	● running
<input type="checkbox"/>	server-iac-4	i-07fe369d29d0f07e7	t2.micro	us-west-1a	● running

Fuente: Elaboración propia.

ANEXO 08 : Balanceador clásico (ELB) manualmente

Create Load Balancer

search : iac-balanceador

<input checked="" type="checkbox"/>	Name	DNS name	State	VPC ID
<input checked="" type="checkbox"/>	manual-elb	manual-elb-58904634.us-we...		vpc-b98355dd

Load balancer: **iac-balanceador**

Description

Basic Configuration

Name: manual-elb

* DNS name: manual-elb-58904634.us-west-1.elb.amazonaws.com (A Record)

Type: Classic (Migrate Now)

Scheme: internet-facing

Availability Zones: subnet-79c17021 - us-west-1a, subnet-b5a8c9d1 - us-west-1b

Fuente: Elaboración propia.

ANEXO 09 : Balanceador clásico (ELB) por IaC con Terraform

The screenshot displays the AWS Management Console interface for a Classic Load Balancer. At the top, there are buttons for 'Create Load Balancer' and 'Actions'. Below is a search bar with the text 'search : iac-balanceador' and an 'Add filter' option. A table lists the load balancer details:

Name	DNS name	State	VPC ID
iac-elb	iac-elb-8590455.us-west-1.elb...		vpc-b98355dd

Below the table, the 'Load balancer: iac-balanceador' is selected. There are tabs for 'Description', 'Instances', 'Health Check', 'Listeners', 'Monitoring', 'Tags', and 'Migration'. The 'Basic Configuration' section shows the following details:

- Name: manual-elb
- * DNS name: iac-elb-8590455.us-west-1.elb.amazonaws.com (A Record)
- Type: Classic (Migrate Now)
- Scheme: internet-facing
- Availability Zones: subnet-79c17021 - us-west-1a, subnet-b5a8c9d1 - us-west-1b

Fuente: Elaboración propia.

ANEXO 10 : Gestor de contenidos (S3) manualmente

The screenshot shows the Amazon S3 console interface. At the top, there is a search bar labeled 'Search for buckets'. Below it are three buttons: '+ Create bucket', 'Delete bucket', and 'Empty bucket'. A 'Bucket name' input field is visible with a list icon. Below the input field, a bucket named 'tesina.manual.s3.45832232' is listed.

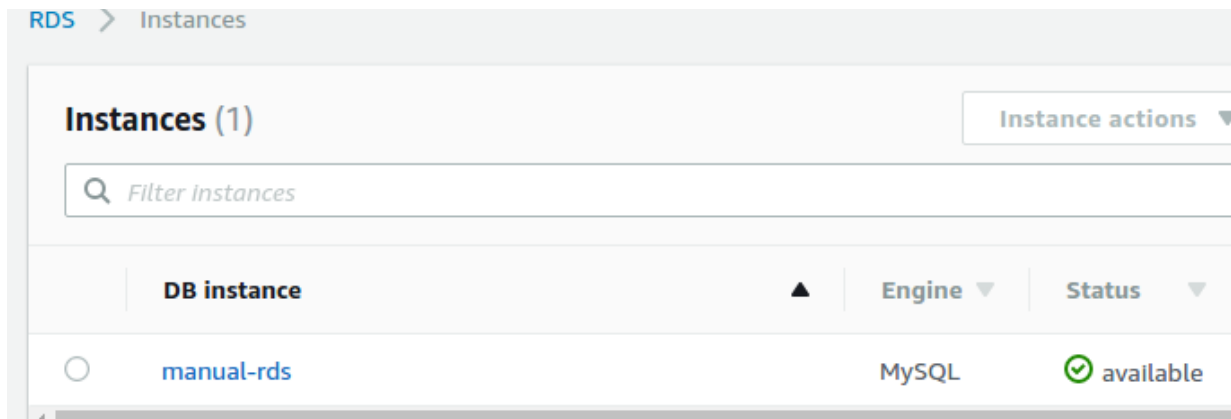
Fuente: Elaboración propia.

ANEXO 11 : Gestor de contenidos (S3) por IaC mediante Terraform

The screenshot shows the Amazon S3 console interface, similar to the previous one. It features a search bar 'Search for buckets', buttons for '+ Create bucket', 'Delete bucket', and 'Empty bucket', and a 'Bucket name' input field. Below the input field, a bucket named 'tesina.iac.s3.45832232' is listed.

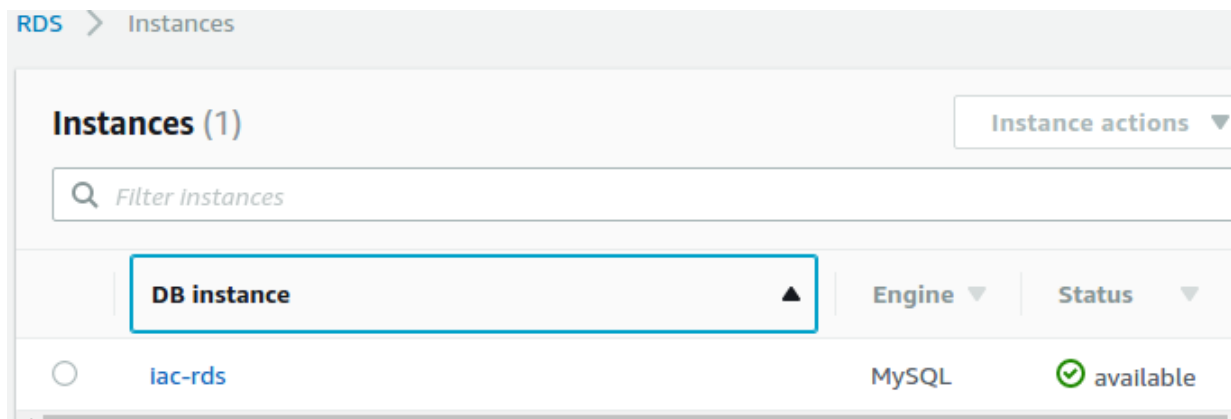
Fuente: Elaboración propia.

ANEXO 12 : Base de datos MySQL (RDS) manualmente



Fuente: Elaboración propia.

ANEXO 13 : Base de datos MySQL (RDS) por IaC mediante Terraform



Fuente: Elaboración propia.

ANEXO 14 : Ejemplo de variables en Ansible para la configuración de Wordpress

```
---
- name: Descargando Wordpress
  get_url: url=http://wordpress.org/wordpress-{{ wp_version }}.tar.gz dest=/srv/wordpress
          sha256sum="{{ wp_sha256sum }}"

- name: Descomprimiendo el archivo
  command: chdir=/srv/ /bin/tar xvf wordpress-{{ wp_version }}.tar.gz creates=/srv/wordpress

- name: Agregando el grupo Wordpress
  group: name=wordpress
```

Fuente: Elaboración propia.

```

- name: Creando la base de datos para el sistema
  mysql_db: name={{ wp_db_name }} state=present

- name: Creando usuario de la base de datos
  mysql_user: name={{ wp_db_user }} password={{ wp_db_password }} priv={{ wp_db_name }}.*:A

- name: Copiando configuraciones del sitio
  template: src=wp-config.php dest=/srv/wordpress/

```

Fuente: Elaboración propia.

ANEXO 15 : Estado de infraestructura con Terraform

```

└─ .terraform
  └─ plugins
    terraform.tfstate
    terraform.tfstate.backup

```

Fuente: Elaboración propia.

```

{
  "version": 3,
  "terraform_version": "0.11.0",
  "serial": 15,
  "lineage": "db886c42-59b6-4c0a-bb44-801accaa693b",
  "backend": {
    "type": "s3",
    "config": {
      "bucket": "tesina.manual.s3.45832232",
      "key": "accessaws/terraform.tfstate",
      "region": "us-east-1"
    },
    "hash": 6054750045309802459
  },
},

```

Fuente: Elaboración propia.