**UNIVERSIDAD POLITÉCNICA SALESIANA**

**SEDE CUENCA**

**CARRERA DE INGENIERÍA DE SISTEMAS**

# DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UNA ARQUITECTURA GITOPS (GIT Y KUBERNETES) QUE INTEGRE INFRAESTRUCTURA BASADA EN CÓDIGO DESPLEGADAS EN PLATAFORMAS IAAS (INFRAESTRUCTURA COMO SERVICIO)

Trabajo de titulación previo a la obtención
del título de Ingeniera de Sistemas

AUTORA: LEIDY MAGALY CHACÓN CABRERA

TUTOR: ING. DIEGO FERNANDO QUISI PERALTA, MGT

Cuenca - Ecuador

2022

## CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Leidy Magaly Chacón Cabrera con documento de identificación N° 1400630677 manifiesto que:

Soy la autora y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 7 de abril del 2022

Atentamente,

Leidy Magaly Chacón Cabrera

1400630677

**CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA**

Yo, Leidy Magaly Chacón Cabrera con documento de identificación N° 1400630677, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Artículo Académico: "Diseño, desarrollo e implementación de una arquitectura Gitops (Git y Kubernetes) que integre infraestructura basada en código desplegadas en plataformas IaaS (Infraestructura Como Servicio)", el cual ha sido desarrollado para optar por el título de: Ingeniera de Sistemas, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 7 de abril del 2022

Atentamente,

Leidy Magaly Chacón Cabrera

1400630677

## CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Diego Fernando Quisi Peralta con documento de identificación N° 0104616461, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UNA ARQUITECTURA GITOPS (GIT Y KUBERNETES) QUE INTEGRE INFRAESTRUCTURA BASADA EN CÓDIGO DESPLEGADAS EN PLATAFORMAS IAAS (INFRAESTRUCTURA COMO SERVICIO), realizado por Leidy Magaly Chacón Cabrera con documento de identificación N° 1400630677, obteniendo como resultado final el trabajo de titulación bajo la opción Artículo Académico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 7 de abril del 2022

Atentamente,

Ing. Diego Fernando Quisi Peralta, MgS.

0104616461

## DEDICATORIA

*A Dios, por la vida y mi familia.*

*A mis padres Vicente Chacón e Hilda Cabrera, porque ellos son los cimientos de mi desarrollo profesional y les debo todo lo que soy, a mis hermanos Christian Ch., Michael Ch y Dennis Ch., que han creído en mí siempre para alcanzar este logro. Es para mí una gran satisfacción poder dedicarles a ellos, que con mucho esfuerzo, esmero y trabajo me lo he ganado.*

*A toda mi familia y amigos que fueron participes en este proceso a Romero T., Gerardo T., Diego Ch., Xavier R., Javier M., Mauricio M., Maicoly M., gracias por la amistad y el apoyo incondicional que siempre he recibido de su parte, los llevo siempre en mi corazón.*

*A YongQian Li, que llegó en los momentos difíciles y estuvo para brindarme motivación, apoyo y su amor. Deseo que sigas presente en todos mis logros.*

*Leidy Magaly Chacón Cabrera*

**AGRADECIMIENTOS**

*Agradezco a mis padres por haberme dado educación y un hogar donde desarrollarme adquiriendo valores que hoy me definen.*

*Agradezco a la Universidad por todos los conocimientos ofrecidos que me han formado como un buen cristiano y honrado ciudadano.*

*Agradezco a mis maestros que con su dedicación y esfuerzo han hecho de mi un buen profesional. De manera especial agradezco a mi tutor, maestro y amigo Diego Quisi, por haberme guiado a lo largo de este trabajo y darme ánimos para seguir adelante con todas mis metas.*

## RESUMEN

En una empresa el equipo de desarrollo y de operaciones debían pasar por diferentes pruebas (vulnerabilidad, integración, control calidad, etc.) para comercializar el producto final y enviar a producción. Esto con lleva a muchas desventajas como por ejemplo tiempo para preparar los servidores, recuperación en caso de algún incidente, lentitud, entre otros. En base a ello, poner en producción una aplicación o infraestructura utilizando la metodología GitOps se ganaría tiempo en cuestiones de entrega, resolución de problemas, recuperación ante una contingencia, agilidad, eficiencia y minorar los errores humanos. En conciencia, se expone el diseño e implementación de una arquitectura GitOps (Git y Kubernetes) utilizando infraestructura basada en código (IaC) para generar el aprovisionamiento automatizado de las tecnologías y aplicaciones, utilizando técnicas como la integración y el despliegue continuo (CI/CD).

Además, esta solución se implementó bajo tres proveedores de nube publica como son: Amazon, Azure y Google Cloud. La metodología empleada consta de 4 fases: La primera que es la construcción de los repositorios de control de versiones y configuración de WebHooks. La segunda la preparación de la herramienta de integración y despliegue continuo. La tercera que es Infraestructura como código mediante Terraform y Jenkins y por último el despliegue continuo de la Aplicación Odoo mediante Jenkins.

El resultado evidenció que por medio del aprovisionamiento se disminuyeron los tiempos en las pruebas para las entregas del producto final, así como la tasa de fallas en el despliegue e integración, asegurando que los cambios que surjan pueden funcionar correctamente. Las técnicas mencionadas y buenas prácticas en GitOps permitieron acelerar el desarrollo de aplicaciones sin comprometer la seguridad, facilitando los cambios en la infraestructura e involucrando, sin inconvenientes, a los nuevos miembros del equipo en su proceso de automatización, concluyendo que, por medio del proceso de aprovisionamiento y la adecuación de los recursos necesarios se consigue desplegar infraestructura, servicios o aplicaciones, logrando la gestión de múltiples recursos de forma organizada y centralizada.

Finalmente se realizó comparaciones entre los proveedores de nube publica, obteniendo un 14,83% para Azure, un 23,59% para Google y un 8.08% para AWS en pruebas de carga con JMeter.



**Palabras Claves:** Kubernetes, aprovisionamiento, CI/CD, GitOps, Terraform y Jenkins, WebHooks, Git, IaC, Odoo.

**ABSTRACT**

In a company, the development and operations team had to go through different tests (vulnerability, integration, quality control, etc.) to market the final product and send it to production. This leads to many disadvantages such as time to prepare the servers, recovery in case of an incident, slowness, among others. Based on this, putting an application or infrastructure into production using the GitOps methodology would save time in terms of delivery, problem resolution, recovery in the event of a contingency, agility, efficiency and reduce human errors. In conscience, the design and implementation of a GitOps architecture (Git and Kubernetes) using infrastructure based on code (IaC) to generate automated provisioning of technologies and applications, using techniques such as continuous integration and deployment (CI/CD) is exposed.

In addition, this solution was implemented under three public cloud providers such as: Amazon, Azure and Google Cloud. The methodology used consists of 4 phases: The first one which is the construction of the version control repositories and WebHooks configuration. The second phase is the preparation of the integration and continuous deployment tool. The third one which is Infrastructure as code through Terraform and Jenkins and finally the continuous deployment of the Odoo Application through Jenkins.

The result showed that through the provisioning, the testing times for the delivery of the final product were reduced, as well as the failure rate in the deployment and integration, ensuring that the changes that arise can work correctly. The mentioned techniques and good practices in GitOps allowed accelerating the development of applications without compromising security, facilitating changes in the infrastructure and involving, without inconveniences, new team members in the automation process, concluding that, through the provisioning process and the adequacy of the necessary resources, it is possible to deploy infrastructure, services or applications, achieving the management of multiple resources in an organized and centralized way.

Finally, comparisons were made between public cloud providers, obtaining 14.83% for Azure, 23.59% for Google and 8.08% for AWS in load tests with JMeter.

**Keywords:** Kubernetes, provisioning, CI/CD, GitOps, Terraform and Jenkins, WebHooks, Git, IaC, Odoo.

## ÍNDICE DE CONTENIDO

# I. Introducción

Actualmente es indiscutible que la evolución de las tecnologías de la información y la comunicación ha cambiado radicalmente la industria, delegando a un segundo plano las soluciones cliente-servidor, teniendo que adaptarse a entornos en la nube o cloud, reduciendo ampliamente la necesidad de adquirir y gestionar servidores (Blas et al., 2019). Adicionalmente, la incorporación de los nuevos esquemas arquitectónicos relacionados con máquinas virtuales repositorios Git y contenedores Kubernetes, han generado un cambio de paradigma en las empresas.

En base a ello, el Cloud Computing o Computación en la Nube, se encuentra dentro de los modelos tales como: Virtualización, Client Server Model y Peer to Peer, resaltando las características de elasticidad y escalabilidad. Sus ventajas se visualizan en la logística, la alta demanda de los recursos y servicios. Sin embargo, la ubicuidad representa una parte importante, pues no solo es un tópico controvertido en entornos corporativo y pedagógico, sino también es esencial debido a su uso a través de miles de aplicaciones, modificando las relaciones entre empresas, clientes, la manera de trabajar, conectarse, compartir y gestionar información de los empleados (Alejandre, 2018).

Considerando lo anterior, los repositorios Git son espacios de almacenamiento virtual que permiten coleccionar versiones de código, a las cuales se puede ingresar en cualquier momento y lugar (Moreira et al., 2017). Por otra parte, los contenedores Kubernetes permiten diseñar servicios de aplicaciones programados en clúster por contenedores, adoptando medidas puntuales para alcanzar una óptima seguridad de TI (Hernández y Camargo, 2017). Con relación a la plataforma IaaS, ésta permite a las compañías esgrimir sistemas de trabajo, aplicaciones y almacenamiento fundamentados en la web sin tener que adquirir, administrar u ofrecer soporte a la infraestructura (Taurus, 2021).

Es necesario mencionar que, la combinación de estos elementos, puede llegar a convertirse en la solución para muchas empresas en búsqueda de escalabilidad, seguridad, aplicaciones y servicios de alta calidad, orientadas a generar productividad sin tener que invertir en infraestructura física.

Por lo tanto, surge la interrogante del presente estudio: ¿los repositorios Git y orquestadores de contenedores Kubernetes desplegados en plataformas IaaS son eficientes? Es así que se empleó la técnica de la revisión sistemática para dar respuesta al objeto en estudio, bases de datos reconocidas y obtener los datos del respectivo análisis.

## 1.1 Estado del Arte

### 1.1.1. Marco de desarrollo GitOps.

Según GitOps (2017) este es un marco de desarrollo de software que permite a las organizaciones entregar continuamente aplicaciones mientas gestionan eficientemente la infraestructura de TI utilizando Git como única fuente. Adicionalmente, es un subconjunto de DevOps, que combina las mejores prácticas de construcción como código (IaC) y DevOps para crear un modelo operativo para gestionar la arquitectura y reproducir instantáneamente la infraestructura en la nube del sistema basándose en el estado de los repos de Git.

Por otra parte, existen varias herramientas y soluciones para el diseño de software; de igual manera, la gestión de la infraestructura siempre ha sido una tarea compleja que requiere un conocimiento experto para construir y mantener la arquitectura. DevOps revolucionó el panorama del desarrollo de software y potenció la gestión de la sistemas con el enfoque de código (IaC) (Johnston, 2020).

Adicionalmente, IaC permite a los desarrolladores declarar la configuración como código y automatizar la infraestructura del sistema, mantener la sincronización en vivo entre el código, las pruebas, el montaje y los entornos de producción que sigue representando una tarea compleja. GitOps amplía la funcionalidad de IaC permitiendo a los diseñadores declarar cada recurso en Git y mantener automáticamente el estado deseado en todo el sistema (Johnston, 2020).

***Ventajas de GitOps.***

Según Johnston (2020) entre las ventajas de GitOps se presentan las siguientes:

- **Fiabilidad:** GitOps permite revertir y deshacer cambios con el clic de un botón. Se puede experimentar con nuevas funciones rápidamente y deshacerlas en caso de un comportamiento inesperado del código. También aumenta los niveles de productividad a la vez que es confiable y seguro.
- **Experiencia de desarrollador mejorada:** GitOps está más focalizado en los diseñadores. Con la forma de trabajar con herramientas del lado del desarrollador como Git, los desarrolladores obtienen un buen control sobre la infraestructura de implementación y pueden administrar y monitorear fácilmente las implementaciones. Los bucles de retroalimentación continua ayudan a identificar errores, realizar cambios y enviar código a producción con más frecuencia,

adicionalmente, se pueden utilizar un único conjunto de herramientas en todo el sistema.

- **Fácil de Auditar:** En un entorno de GitOps, todos los cambios se realizan a través del repositorio. Como tal, se puede consultar el historial de la rama en cualquier momento para ver la evolución de implementación completo y el historial de cada cambio realizado en el código. La función de seguimiento de auditoría gratuita simplifica esencialmente las labores de auditoría. Además, cada miembro del equipo puede consultar el repositorio de Git y estar actualizado con lo que sucede a lo largo del proceso del desarrollo.
- **Operaciones estandarizadas en toda la infraestructura:** GitOps permite crear un modelo estándar para la gestión de cambios en las aplicaciones, implementaciones y complementos. Significa que tiene flujos de trabajo de un extremo a otro centrados en Git que son consistentes en toda la infraestructura. También proporciona una mayor visibilidad de las canalizaciones de CI / CD.

### 1.1.2. Canalización o Pipeline.

Según Poniszewska y Czechowska (2021) la canalización se refiere a la cola lógica que se llena con todas las instrucciones para que el procesador de la computadora procese en paralelo. En otras palabras, es el proceso de almacenar y poner en cola tareas que el procesador ejecuta simultáneamente de manera organizada. Asimismo, una distribución incluye labores e instrucciones del procesador en diferentes etapas. El procesador de la computadora trabaja en cada tarea en proceso. Funciona de forma diferente a la arquitectura informática FIFO y LIFO, pero asigna algo de tiempo de procesamiento a cada trabajo en la tubería. La canalización permite el procesamiento de cada tarea en paralelo en lugar de esperar a que termine una actividad y luego pasar a otra.

### 1.1.3. Canalización CI/CD.

Esta es una metodología empleada para la distribución de implementaciones, con el propósito de proporcionar una entrega continua según los requerimientos del negocio, garantizando una manera práctica de automatizado para el diseño, empaquetar y realizar pruebas de aplicaciones, en otras palabras, la integración constante (CI) y la entrega continua (CD) establece, un grupo de principios operativos y una recopilación de prácticas que apoyan a los equipos de diseño de aplicaciones realizar modificaciones de código con mayor seguridad y periodicidad (Buchanan et al., 2020).

### 1.1.4. Terraform.

Según Johnston (2020) es una herramienta de software de infraestructura de código abierto que ayuda a los ingenieros de DevOps aprovisionar de manera programada los recursos físicos que una aplicación necesita para funcionar. Es por ello, que este enfoque de la asignación de recursos permite a los desarrolladores gestionar, supervisar y aprovisionar recursos de forma lógica, en lugar de requerir que un equipo de operaciones configure manualmente cada recurso necesario.

Por otra parte, los usuarios de Terraform definen y aplican las configuraciones de la infraestructura mediante un lenguaje de configuración similar a JSON llamado HCL (HashiCorp Configuration Language). La sintaxis de HCL facilita a los equipos de DevOps el aprovisionamiento y reaprovisionamiento de la infraestructura en múltiples centros de datos en la nube y en las instalaciones (Havel, 2021a).

Terraform permite a los usuarios definir toda su infraestructura simplemente usando archivos de configuración y control de versiones. Cuando se da un comando para implementar y ejecutar un servidor, base de datos o equilibrador de carga, Terraform analiza el código y lo traduce en una llamada de interfaz de programación de aplicaciones API al proveedor de recursos. Dado que Terraform es de código abierto, los desarrolladores siempre pueden ampliar la utilidad de la herramienta escribiendo nuevos complementos o compilando diferentes versiones de complementos existentes (Johnston, 2020).

### 1.1.5. Jenkins.

Es una herramienta de sistematización de código abierto codificada en Java con complementos creados para propósitos de integración continua. Jenkins se emplea para diseñar y probar proyectos de software de forma constante, lo que favorece a los desarrolladores la incorporación de modificación en el proyecto y facilita a los usuarios la obtención de una nueva versión. También permite entregar continuamente software mediante la integración y con una gran cantidad de tecnologías de implementación y prueba (Poniszewska y Czechowska, 2021).

Adicionalmente, con Jenkins, las organizaciones pueden acelerar el proceso de elaboración de software por medio de la automatización, este integra procesos de ciclo de vida de desarrollo de todo tipo, incluidos compilación, documentación, prueba, empaquetado, etapa, implementación y análisis estático. También, logra la integración continua con la ayuda de complementos. Estos facilitan la integración de varias etapas de DevOps. Si se desea integrar una herramienta en particular, se deben instalar los complementos para esa herramienta. Por ejemplo, Git, proyecto Maven 2, Amazon

EC2, editor HTML, entre otros (Poniszewska y Czechowska, 2021).

***Las ventajas de Jenkins incluyen:***

Según Poniszewska y Czechowska (2021) indica las siguientes ventajas:

- Es una herramienta de código abierto con un gran apoyo de la comunidad.
- Es fácil de instalar.
- Tiene más de 1000 complementos para facilitar el trabajo. Si no existe un complemento, puede codificarlo y ser compartido con la comunidad.
- Gratis.
- Está construido con Java, por lo tanto, es portátil para todas las plataformas principales.

### 1.1.6. YAML.

Es un lenguaje de serialización de datos que se utilizar para escribir archivos de configuración es similar a XML y JSON. Sin embargo, es más legible por humanos. Adicionalmente, es un formato de datos que distingue entre mayúsculas y minúsculas. Utiliza espacios para definir la estructura del documento y las tabulaciones (\t) no están permitidas. Por lo general, los archivos YAML o también llamado YML, se usan para almacenar datos de configuración y comúnmente se nombran con extensiones ".ymlo .yaml, por ejemplo, config.yml" (Nocentino y Weissman, 2021).

Se emplean en el caso que un usuario o administrador de la aplicación, especifica datos en un archivo YAML, que la aplicación puede leer. Por ejemplo, YAML se utiliza para definir usuarios o instalar paquetes de software en servidores. Los pares clave-valor se definen en YAML y se separan con dos puntos. Las claves pueden ser cualquier componente, como una variable y el valor puede ser uno o varios valores asignados a esa clave. Estos pares clave-valor se especifican en listas, donde un elemento de la lista puede ir seguido de una configuración completa que debe crearse para ese aspecto y los elementos van desde partes de la configuración que deben definirse en un sistema informático hasta entradas que deben importarse a una base de datos, esto dependerá de la aplicación (Nocentino y Weissman, 2021).

Asimismo, algunas aplicaciones importantes que usan YAML son la herramienta de administración de configuración Ansible, Kubernetes para la orquestación de contenedores y OpenStack Heat, el motor de instrumentación para lanzar aplicaciones en la nube en la infraestructura de OpenStack como servicio. Python es un lenguaje de programación que se basa en la sintaxis YAML, pero la mayoría de los lenguajes de programación pueden interpretarlo (Gil, 2020).

### 1.1.7. Kubernetes.

Es una plataforma de organización de contenedores de código abierto que facilita la funcionalidad de un marco de servidores web elásticos para implementaciones en la nube. Este puede soportar la subcontratación de centros de datos a proveedores de nube pública o permite emplearse en el alojamiento web a escala. Las aplicaciones web y móviles con código complejo diseñado a medida pueden desplegarse utilizando Kubernetes en hardware básico para disminuir valores de aprovisionamiento de servidores web con hosts de nubes públicas y maximizar los métodos de desarrollo de software (Pérez, 2021).

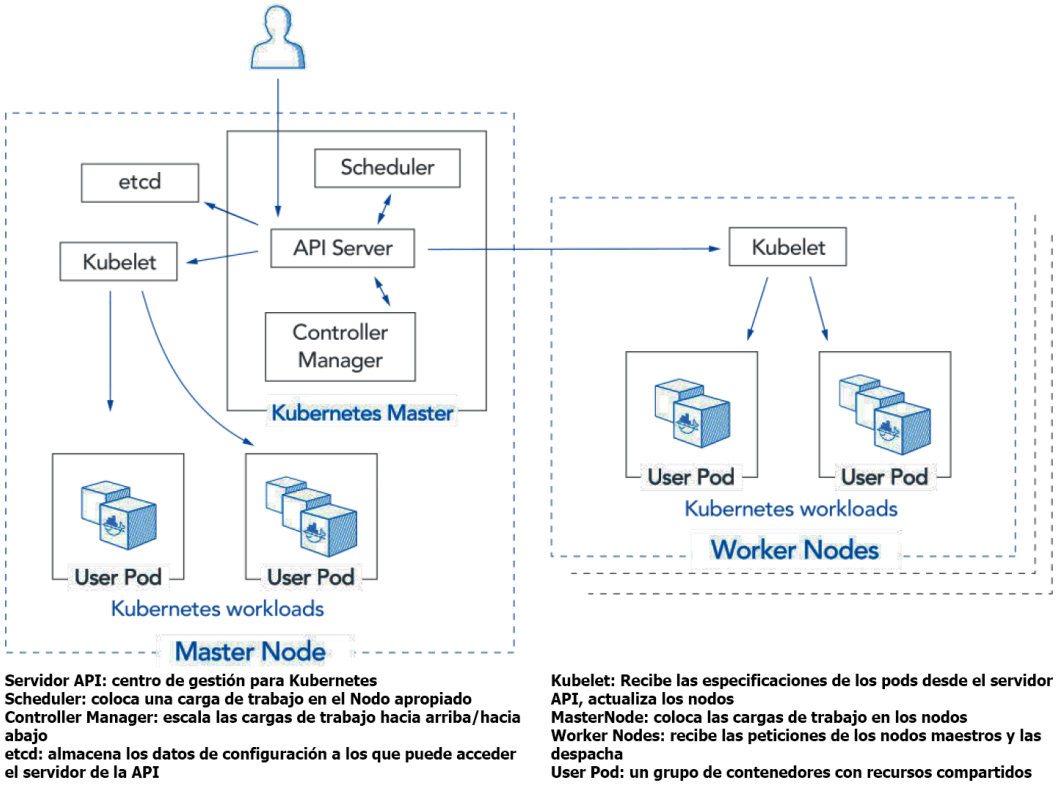


**Figura 1** Arquitectura de Kubernetes y sus componentes

A continuación, se detallan los conceptos clave dentro de la figura 1:

**Pod de usuario (User Pod):** grupo de contenedores con recursos compartidos.

**Servidor API (Server API):** centro de gestión para Kubernetes.

**Scheduler:** coloca una carga de trabajo en el Nodo apropiado.

**Controller Manager:** escala las cargas de trabajo hacia arriba/hacia abajo.

**Etcd:** almacena los datos de configuración a los que puede acceder el servidor de la API.

**Kubelet:** Recibe las especificaciones de los pods desde el servidor API, actualiza los

nodos.

**Master Node:** coloca las cargas de trabajo en los nodos.

**Worker Nodes:** recibe las peticiones de los nodos maestros y las despacha.

**User Pod:** un grupo de contenedores con recursos compartidos.

***Características de Kubernetes.***

Kubernetes cuenta con la posibilidad de automatizar el aprovisionamiento de servidores web según el nivel de tráfico web en producción. El hardware del servidor web puede ubicarse en distintos centros de datos, en diferente hardware o por medio de diversos proveedores de alojamiento. Escala los servidores web en función de la demanda de las aplicaciones y luego degrada las instancias del servidor web durante los tiempos de inactividad. Kubernetes, igualmente, tiene capacidades avanzadas de equilibrio de carga para el enrutamiento del tráfico web a los servidores web en operaciones (Poniszewska y Czechowska, 2021).

***Arquitectura de Kubernetes.***

Esta evolucionó a partir del código Google que utilizó para gestionar sus centros de datos a escala con la plataforma "Borg". AWS introdujo los marcos de servidores web elásticos al público con el lanzamiento de la plataforma EC2. Kubernetes permite a las empresas orquestar contenedores como EC2, pero utilizando código abierto. Google, AWS, Azure y los demás principales anfitriones de nubes públicas ofrecen compatibilidad con Kubernetes para la orquestación de servidores web en la nube. Los clientes pueden utilizar Kubernetes para la externalización completa del centro de datos, aplicaciones web móviles, soporte SaaS, alojamiento web en la nube o computación de alto rendimiento (Martin, 2021).

***Terminología de Kubernetes.***

Kubernetes, el cual es con frecuencia mencionado como "K8s", es parte de la Cloud Native Computing Foundation, que apoya el desarrollo de estándares de red compartidos en el software de gestión de centros de datos en la nube. Docker es el estándar de virtualización de contenedores más popular empleado por Kubernetes. Docker brinda herramientas integradas de desarrollo del ciclo de vida del software para los equipos de programación. RancherOS, CoreOS y Alpine Linux son sistemas operativos creados concretamente para el uso de contenedores. La virtualización de contenedores es distinta a las herramientas de VM o VPS que emplean hipervisores y habitualmente, requiere una huella de sistema operativo más pequeña en la producción (Pérez, 2021).

***API Kubernetes.***

La API de Kubernetes proporciona las bases para la estructura de configuración declarativa del sistema y admite la ejecución de aplicaciones en contenedores, vincula implementaciones a través del despliegue de servicios, administra la infraestructura del clúster y relaciona almacenamiento persistente, así como la herramienta de línea de comandos Kubectl puede emplearse para generar, consultar, actualizar, eliminar objetos a través de la API (Kubernetes, 2021).

Las convenciones de la API de Kubernetes y las API relacionadas en el ecosistema están destinadas a facilitar el desarrollo del cliente y garantizar que se puedan implementar mecanismos de configuración que funcionen en un conjunto diverso de casos de uso de manera coherente. Por otra parte, el estilo general de la API de Kubernetes es RESTful: los clientes crean, actualizan, eliminan o recuperan una descripción de un objeto a través de los comandos en HTTP estándar, como, por ejemplo, (POST, PUT, DELETE y GET) y esas API aceptan y devuelven JSON preferentemente. Kubernetes también expone puntos finales adicionales para comandos no estándar y permite tipos de contenido alternativos. Todo el JSON aceptado y devuelto por el servidor tiene un esquema, identificado por los campos "kind" y "apiVersion". Cuando existan campos de encabezado HTTP relevantes, deben reflejar el contenido de los campos JSON, pero la información no debe representarse solo en el encabezado HTTP (Kubernetes, 2021).

***Nodos.***

Kubernetes ejecuta su carga de trabajo colocando contenedores en pods para que se ejecuten en nodos. Un nodo puede ser una máquina física o virtual, según el clúster. Cada nodo se gestiona por el plano de control y contiene los servicios requeridos para ejecutar Pods, normalmente, tiene varios nodos en un clúster, por ejemplo, es posible que solo tenga un nodo en un entorno de aprendizaje o de recursos limitados. Por otra parte, los componentes de un nodo incluyen el Kubelet, a tiempo de ejecución del contenedor y el proxy de Kube (Kubernetes, 2021).

***Gestión de Nodos.***

Según Kubernetes (2021) existen dos formas principales de agregar nodos al Servidor API:

- El Kubelet de un nodo se registra automáticamente en el plano de control.
- Agrega manualmente un objeto Nodo

Después de crear un objeto Node, o el Kubelet en un nodo este se auto-registra y el plano de control verifica si el nuevo objeto Node es válido. Posteriormente, Kubernetes crea un objeto Node internamente, esta comprueba que un Kubelet se haya registrado en el servidor de API que coincida con el metadata.name campo de Node. Si el nodo está en buen estado, es decir, se están ejecutando todos los servicios requeridos, por tanto, es elegible para ejecutar un Pod.

***Pods.***

Los pods son objetos que tienen como característica ser los más pequeños y básicos que se pueden implementar en Kubernetes. Un Pod representa una instancia única de un proceso ejecutado en un clúster (Kubernetes, 2021).

***Kubectl.***

Es herramienta de línea de comandos para comunicarse con un API de Kubernetes servidor. Se puede usar Kubectl para implementar aplicaciones, inspeccionar y administrar recursos del clúster y ver registros, también se puede instalar en una variedad de plataformas como Linux, macOS y Windows (Kubernetes, 2021).

***Contenedores.***

Un contenedor es un paquete de software que contiene todas las dependencias, bibliotecas del sistema, entorno en tiempo de ejecución, código y configuración para que pueda ejecutarse en cualquier sistema propietario. En el ambiente en tiempo de ejecución, el contenedor recoge su propia parte de recursos del sistema operativo, como disco, RAM, CPU y red. Su importancia radica en que estos son eficaces en aumentar la eficiencia de DevOps por medio de diversos códigos (Kubernetes, 2021).

***Imágenes.***

Según Kubernetes (2021) una imagen de contenedor es un archivo estático con código ejecutable que puede crear un contenedor en un sistema informático. Una imagen de contenedor es inmutable, lo que significa que no se puede cambiar y se puede implementar de forma coherente en cualquier entorno y es un componente central de una arquitectura en contenedores. Adicionalmente, las imágenes de contenedor incluyen todo lo que un contenedor necesita para ejecutarse: el motor del contenedor, como Docker o CoreOS, bibliotecas del sistema, utilidades, opciones de configuración y cargas de trabajo específicas que deben ejecutarse en el contenedor.

Una imagen de contenedor se compone de capas, agregadas a una imagen principal, también conocida como imagen base. Las capas permiten reutilizar componentes y configuraciones en imágenes. Por lo que, la construcción de capas de manera óptima

puede ayudar a reducir el tamaño del contenedor y mejorar el rendimiento (Kubernetes, 2021).

***Load balancer.***

El equilibrio de carga es el proceso de distribución del tráfico de red entre diferentes servidores. Esto garantiza que ningún servidor tolere excesivas solicitudes. Al repartir la carga con uniformidad, la carga se equilibra mejorando en la aplicación la capacidad de respuesta. Asimismo, incrementa el acceso de sitios web y aplicaciones para los usuarios. Las aplicaciones modernas no pueden ejecutarse sin balanceadores de carga. Adicionalmente, los equilibradores de carga de software han agregado capacidades adicionales, incluida la seguridad de las aplicaciones (Kubernetes, 2021).

Los balanceadores de carga administran el flujo de información entre el servidor y un dispositivo de punto final, como, por ejemplo, PC, computadora portátil, tableta o teléfono inteligente. El servidor puede estar en las instalaciones, en un centro de datos o en la nube pública, también puede ser físico o virtualizado. El equilibrador de carga ayuda a los servidores a mover datos de manera eficiente, optimiza el uso de los recursos de entrega de aplicaciones y evita las sobrecargas del servidor. Adicionalmente, estos realizan comprobaciones de estado continuas en los servidores para garantizar que puedan gestionar las solicitudes. Si es necesario, el equilibrador de carga elimina los servidores en mal estado del grupo hasta que se restauran. Algunos equilibradores de carga incluso desencadenan la creación de nuevos servidores de aplicaciones virtualizados para hacer frente al aumento de la demanda (Kubernetes, 2021).

### 1.1.8. Infraestructura como servicio o IaaS.

Describe un modelo de entrega común para servicios en la nube donde los clientes compran acceso a la infraestructura de TI administrada de un proveedor de servicios en la nube de terceros. Los proveedores de servicios en el espacio IaaS brindan infraestructura a los clientes que puede incluir hardware, servidores, almacenamiento y espacio del centro de datos, junto con los componentes de red necesarios para permitir el acceso y la administración de los activos arrendados (AZURE, 2021).


**Figura 2** IaaS Infraestructura como Servicio Fuente: AZURE (2021).

***Funcionamiento de IaaS.***

La IaaS funciona movilizando a un proveedor de servicios en la nube de terceros. Este proveedor de la nube aloja la infraestructura de TI. Esto significa servidores, almacenamiento en la nube, hardware y virtualización. Así pues, la entrega de IaaS está automatizada y el mantenimiento del sistema y el almacenamiento de los datos frecuentemente es cedido al proveedor de IaaS. En otras palabras, el proveedor de servicios aloja el equipamiento en nombre de su cliente en diferentes centros de datos que conforman la nube. El cliente accede a la arquitectura en línea y la utiliza para los mismos fines que una infraestructura informática interna. Se puede acceder a ella a través de ordenadores, dispositivos móviles o máquinas virtuales (AZURE, 2021).

***Ventajas de IaaS.***

Según AZURE (2021) entre las ventajas de IaaS se presentan las siguientes:

- **Pago por uso:** a diferencia de la TI tradicional, IaaS no requiere gastos de capital por adelantado y a los usuarios finales solo se les factura por lo que utilizan.
- **Velocidad:** con IaaS, los usuarios pueden aprovisionar pequeñas o grandes cantidades de recursos en cuestión de minutos, probando nuevas ideas rápidamente o escalando las ya probadas de forma aún

más rápida.

- **Disponibilidad:** a través de cosas como las regiones multizona, la disponibilidad y la resistencia de las aplicaciones en la nube pueden superar los enfoques tradicionales.
- **Escala:** con una capacidad aparentemente ilimitada y la capacidad de escalar recursos de forma automática o con cierta supervisión, es fácil pasar de una instancia de una aplicación o carga de trabajo a muchas.
- **Latencia y rendimiento:** dada la amplia huella geográfica de la mayoría de los proveedores de IaaS, resulta fácil colocar aplicaciones y servicios más cerca de sus usuarios, lo que reduce la latencia y mejora el rendimiento.

### 1.1.9. Azure Kubernetes Service (AKS).

AKS (Azure Kubernetes Service) es el servicio de contenedor dirigido por Azure y es un servicio de instrumentación de contenedores administrado y de código abierto. Administra su entorno de Kubernetes alojado, lo que facilita la implementación y la administración de aplicaciones en contenedores sin experiencia en orquestación de contenedores, y transfiere gran parte de esa responsabilidad a Azure. Este servicio ofrece aprovisionamiento, escalado y actualizaciones de recursos según los requisitos o la demanda sin ningún tiempo de inactividad en el clúster de Kubernetes (Poniszewska y Czechowska, 2021).

***Beneficios de AKS.***

Según Poniszewska y Czechowska (2021) refiere los siguientes beneficios:

- Ayuda a administrar una configuración de Kubernetes alojada que admite una preparación y administración versátiles y rápidas de aplicaciones de paquetes.
- Simplemente no desea tener una experiencia profunda en la orquestación de contenedores para usar AKS.
- Es compatible con el aprovisionamiento económico, las actualizaciones y el escalado de recursos en consonancia con la demanda, sin desconectar las aplicaciones.
- Este servicio ofrece a los desarrolladores una mayor flexibilidad, automatización y una reducción de la sobrecarga de gestión para administradores y desarrolladores.

### 1.1.10. Amazon Elastic Kubernetes Service (EKS).

Es un servicio administrado que se emplea para ejecutar Kubernetes en AWS sin necesidad de instalar, operar y mantener nodos de control, este es un sistema de código abierto para automatizar la implementación, administración y escalado de aplicaciones en contenedores (Malina, 2019).

***Beneficios de EKS.***

Según Malina (2019) indica los siguientes beneficios:

- Ejecuta y escala el plano de control de Kubernetes en varias zonas de disponibilidad de AWS para certificar una alta disponibilidad.
- Está integrado con muchos servicios de AWS para proporcionar escalabilidad y seguridad para sus aplicaciones, incluidas las siguientes capacidades:
    - Amazon ECR para imágenes de contenedor
    - Balanceo de carga elástico para distribución de carga
    - IAM para autenticación
    - Amazon VPC para aislamiento

### 1.1.11. Google Kubernetes Engine (GKE).

Este suministra un ambiente de administración para establecer, escalar, administrar las aplicaciones en contenedores a través de los servicios de Google. El entorno de GKE se basa de diversas máquinas, por ejemplo, instancias de *Compute Engine* que se organizan para conformar un clúster (Buchanan *et al*., 2020).

***Beneficios de GKE.***

Según Buchanan *et al*. (2020) muestran los siguientes beneficio:

- Balanceo de cargas de Google Cloud para instancias de *Compute Engine*.
- Grupos de nodos para designar subconjuntos de nodos dentro de un clúster con el propósito de alcanzar flexibilidad.
- Ajuste de escala automático del recuento de instancias de nodos del clúster.
- Actualización de forma automatizada para el software de nodo del clúster
- Reparaciones automáticas de los nodos para conservar la

disponibilidad y en buen estado

- Supervisión y registro con *Google Cloud's operations suite* para alcanzar visibilidad del clúster

***Costos***

A continuación, en la (**Tabla I**) se presentan los costos de Kubernetes, la base de datos Postgres según cada plan contratado para los 3 proveedores con el valor mensual.

| **Costos de Kubernetes** | | | | | | |
|---|---|---|---|---|---|---|
| **Proveedor** | Memoria | CPU | Nodos | Base de datos | Plan | Costo |
| **Azure Cloud** | 5GB | 1 | 2 | 2GB | Básico | $36.27/mes |
| | 10GB | 4 | 4 | 5GB | Avanzado | $43.25/mes |
| **Google Cloud** | 473.63 MiB | 1 | 2 | 1GB | Básico | $27.12/mes |
| | 15GB | 6 | 4 | 2GB | Avanzado | $38.74/mes |
| **AWS Cloud** | 5 GB | 1 | 2 | 2GB | Básico | $76.24/mes |
| | 18GB | 4 | 4 | 3GB | Avanzado | $138.12/mes |

**Tabla I:** Costos Kubernetes por proveedor

### 1.1.12. Trabajos relacionados.

A continuación, en la (**Tabla II**), se presenta un resumen de las investigaciones relacionadas al tema de esta investigación:

| TÍTULO | RESUMEN | HERRAMIENTAS A USAR | RESULTADO | AUTOR |
|---|---|---|---|---|
| GitOps: uma nova proposta para a infraestrutura | Propone desarrollar una metodología para luego, ejecutar una infraestructura y así automatizar los procesos de entrega continua de las aplicaciones a través de la metodología GitOps, como resultado demuestra el proceso adoptado desde su planificación hasta la creación de un entorno de infraestructura, donde, lo demuestran con realizando pruebas con un aplicativo. | • GITOPS<br>• AWS<br>• KUBERNETS<br>• DOCKER<br>• GITLAP | Como resultado de la implementación permitió dedicar menos tiempo a la resolución de conflictos<br>tiempo en la resolución de conflictos durante el ciclo de desarrollo de la infraestructura y la aplicación, haciéndolos disponibles para su uso en todo momento. | (Ventura da Silva, 2021) |
| Herramienta para despliegue y gestión de plataformas en la nube | Propone montar una arquitectura basado en herramientas de IaC, para desplegar una plataforma de Big Data y posterior a ello, aplicar un análisis de datos, que necesitaba ser desplegada en entornos de clientes, ajustando recursos, funcionalidades y su configuración por cada escenario. | • Servicios<br>• GitHub<br>• GitHub Actions<br>• CodeCov<br>• GoReport Card<br>• Go Doc<br>• Para el desarrollo de software<br>• Visual Studio Code<br>• Git<br>• Doker<br>• Consola de líneas de comandos ítem<br>• Browser Chrome<br>• Microsoft Word | Como resultado se presentó el caso de uso de una empresa de hosting. Consiste en la gestión de un conjunto de servidores de WordPress dedicados por cliente que utilizan una base de datos común, todo ello sobre un clúster de Kubernetes. Esto demostró las capacidades y posibilidades de la herramienta, que también puede aplicarse a casos más complejos. | (Rodríguez de la Cruz, 2020) |
| GitOps and ArgoCD: Continuous deployment and maintenance of a full stack application in a hybrid cloud Kubernetes environment | Desarrollar un ambiente a nivel de abstracción proporcionado por YAML para gestionar la implementación de aplicaciones, con ello permitirá automatiza, auditar y resultará más fácil entender su funcionalidad y ejecución. | • KUBERNETS<br>• GITOPS<br>• ARGOCD | Como resultado del trabajo de tesis en el usó de una estrategia de ramificación y una estrategia de etiquetas<br>que es gestionada automáticamente por los flujos de trabajo de GitHub Actions dentro del repositorio de infraestructura, los entornos Kubernetes son gestionados por<br>Kustomize, que es una herramienta de gestión de la configuración. Esta herramienta es muy interesante<br>por su característica de redefinir recursos escritos declarativamente en archivos YAML<br>dejando los archivos base sin cambios. La estructura de directorios también es muy legible. | (D'Amore, 2021) |
| Adaptive Application Scheduling under Interference in Kubernetes | Consiste en el desarrollo de un modelo de recursos basados en la red, mismo que es gestionado por Kubernets, con el objetivo de identificar los problemas de rendimiento. Este plan piloto usa datos obtenidos por medio de un micro-benchmarks, dentro de la implementación de Kubernets puede ser usado como una base para un diseño escalable (siendo potencialmente tolerante ante cualquier Buck).<br>El autor de este artículo presenta este modelo en el cual se basa en la gestión y rendimiento para Kubernets, donde identifica | • Kubernetes<br>• VM | Como resultado obtenido se evidencia que se puede utilizar para estimar la sobrecarga<br>introducida por las interferencias y también identificar las aplicaciones con baja/alta sensibilidad a las interferencias, lo que permite al programador asignar contenedores por<br>nodo. Además, el modelo puede utilizarse para realizar varios escenarios hipotéticos para investigar el comportamiento del SoI, pudiendo asignar una distribución de probabilidad aleatoria para el uso | (Medel et al., 2016) |

| TÍTULO | RESUMEN | HERRAMIENTAS A USAR | RESULTADO | AUTOR |
|---|---|---|---|---|
| | diversos estados operacionales, mismos que pueden estar asociados a los "pods" y contenedores, siempre y cuando se tome en cuenta la competencia de estos para poder acceder a los recursos del sistema, basado en el consumo de recursos a nivel de hardware, entre otros. | | de un recurso asociado a cada SoI y medir la sobrecarga asociada a cada contenedor. | |
| Diseño e implementación del sistema de gestión de entornos para la oficina asesora de sistemas de la Universidad Distrital | Brindar una herramienta tecnológica que mejore los procesos de despliegue de la OAS (Oficina de Asesorías de Sistemas), misma que es la encargada del desarrollo, mantenimiento y soporte de sistemas de la información de dicha Universidad. El proyecto se basa en la explicación de la tecnología de Doker como CaaS (Container as a Service), cumple como meta replicar un microservicio varias veces que sea necesario. | • Doker (CaaS)<br>• Docker EE (Enterprise Edition):<br>• Docker CE (Community Edition) | Como resultado de la solución tecnológica logró reducir el tiempo empleado para el despliegue de entornos, brindando características resaltantes como lo es la facultad de los puertos de comunicación de cada una de los aplicativos o la versión de la misma que se va a utilizar dentro del proyecto, permita generar un estándar para el proyecto en su totalidad. | (Muñoz, 2017) |
| Implementación de una arquitectura de microservicios para una red de sensores IOT sobre Arduino. | Este trabajo de grado consiste en analizar la viabilidad de usar una arquitectura de microservicios, para administrar una red de sensores a una placa en Arduino. Para lograrlo, inicia el proyecto desarrollando una aplicación móvil de demótica y este a su vez es apoyada en una arquitectura de microservicios de spla en Kubernets para luego comunicarse con los sensores. | • Kubernetes<br>• Doker<br>• Arduino<br>• Cloud<br>• IoT<br>• Angular para el desarrollo de un Dashboard | Como resultado se ha logrado implementar una arquitectura de microservicios funcional y escalable, aprovechando las ventajas que ofrece este tipo de arquitectura con respecto a una arquitectura monolítica. Además, el uso de<br>Docker y Kubernetes ha supuesto un valor añadido en cuanto al despliegue de los microservicios, | (Gesteira, 2020) |
| Modelo para la orquestación de microservicios con Kubernets aplicado al servicio de control de versiones GIT. | consiste en realizar 4 configuraciones de redes para su correcto funcionamiento, donde, se llegó a aplicar tecnologías de microservicios, tales como: Kubernets y Doker, luego se implementó componentes de almacenamiento como entre otros. (aplicando un cifrado en reposo), PostgreSQL y Redis; este cifrado fue configurado por medio del uso de la criptografía asimétrica, alojamiento de GitLab, una aplicación codificada en Ruby, Javascript, HTML, CSS, configuración de DNS y de un servidor Web para acceder a los servicios dentro del Cluster. | • CoreOS<br>• ETCD<br>• Kubernetes<br>• RBAC | Como resultados del proyecto logró determinar una arquitectura basada en microservicios, que<br>soportara una demanda considerable de usuarios para el uso de una aplicación, en este caso GitLab. Kubernetes ofrece una amplia gama de componentes para la configuración y personalización en el orquestamiento de microservicios. Lo cual permite soportar diversos sistemas y aplicaciones. | (Delgado y Pineda, 2019) |
| A Kubernetes controller for managing the availability of elastic microservice based stateful applications | Identificación de posibles arquitecturas para implementar aplicaciones que se basan en microservicios, para ello proceder a analizar y evaluar los requisitos de HA. | • Kubernetes<br>• HA | Los resultados de los experimentos muestran que las acciones de reparación de Kubernetes no pueden satisfacer los requisitos de HA y en algunos casos, no pueden garantizar la recuperación del servicio. Por lo tanto, se propone un controlador de estado HA que se integra con Kubernetes y permite la replicación del estado de la aplicación y la redirección automática del servicio a las instancias de microservicio en buen estado al permitir la recuperación del servicio además de las acciones de reparación de Kubernetes y en base en experimentos, se evaluó la solución y se comparó las diferentes arquitecturas desde la perspectiva de la disponibilidad y la sobrecarga de escala. | (Vayghan et al., 2021) |
| Devops in e-commerce software development: demand for containerization. | Investigar tendencias para mejorar la entrega del software, además, la investigación decidió aplicar un conjunto de capacidades a una tubería CI /CD, basada en la tecnología de contenedorización. Con ello, se llevó a cabo una encuesta para comparar la canalización actual de CI/CD. | • Git VCS<br>• Kubernetes<br>• Magneto<br>• Docker | La investigación reveló un conjunto de capacidades y patrones culturales que los desarrolladores consideran más beneficiosos para el futuro de la entrega de software para un proveedor de soluciones de comercio electrónico. Estas capacidades incluyen los despliegues automáticos, el uso de Git VCS, el despliegue en el entorno de la nube utilizando contenedores. Además, los profesionales revelaron una actitud positiva a la hora de asumir la responsabilidad operativa y proporcionaron comentarios sobre las deficiencias de la tubería CI/CD | (Zakharenkov, 2019) |

| TÍTULO | RESUMEN | HERRAMIENTAS A USAR | RESULTADO | AUTOR |
|---|---|---|---|---|
| | | | actualmente implementada. Las conclusiones que se desprenden de la encuesta también fueron confirmadas por una correlación entre una determinada capacidad/solución y la competencia medida a través de la autoevaluación. Disponer de datos estadísticamente válidos y de una decisión concluyente a favor o en contra de una determinada capacidad tecnológica permite a los proveedores de comercio electrónico construir un canal de entrega de software que responda a las necesidades de la organización y de sus clientes. | |
| My Pension | Desarrollar una aplicación web progresiva, que calcula y simula seguro nacional, convenio colectivo, ahorro individual y presentar plan de ahorro. Una vez culminado el proyecto será implementado utilizando herramientas de servicios en la nube. | • Desarrollo de software<br>• NodeJS<br>• Apollo<br>• Servicios<br>• Azure Kubernetes Services<br>• Helm<br>• GitOps<br>• Bitbucket pipelines<br>• Flux<br>• Apollo GraphQL<br>• Vue JS framework | Los resultados obtenidos en este trabajo se han especificado los lineamientos requeridos para la construcción de un entorno de trabajo que facilite la tarea de diseño arquitectónico. La base de dicho entorno queda definida como un metamodelo, el cual establece los lineamientos de instanciación y verificación de patrones de diseño que serán desarrollados en trabajos futuros. | (Bakkland et al., 2019) |
| Kubernetes canary deployment controller řadič postupného nasazení software nad platformou kubernetes | Búsqueda e implementación de herramientas para habilitar y mejorar el desarrollo ágil. Los proyectos de software tienden a lidiar con nuevas estrategias de implementación para reducir el riesgo de que nuevos cambios rompan el sistema existe. | • Kubernetes<br>• GitOps<br>• Spinnaker<br>• Krane<br>• GitHub<br>• Canary | Los resultados obtenidos por medio de los múltiples experimentos realizados demostraron que Kubernetes canary pueden mejorar positivamente la estabilidad del despliegue y reducir el riesgo que suponen los nuevos cambios. | (Malina, 2019) |
| GitOps – Implementace CI/CD pipeline pro dokumentaci produktu při agilním vývoji | Consiste en crear y desplegar documentos utilizando los lotes de software GitOps, como meta tienden a estudiar tecnologías que sean necesarias para que la documentación sea administrada de manera ágil. El resultado obtenido es que se llega a obtener un sistema capaz de crear y publicar softwares | • Gitlap CI<br>• Argo CD<br>• IaC<br>• Kubernetes<br>• Git; DevOps<br>• CI/CD<br>• Doker | El resultado de este documento es un sistema de creación y publicación de documentación de software versionado reutilizable para otros proyectos. El pipeline de CI/CD se implementa utilizando GitLab CI y Argo CD. | (Havel, 2021b) |
| Modelo de composición de microservicios para la implementación de una aplicación web de comercio electrónico utilizando Kubernetes | Proponer un modelo de microservicios para implementar una aplicación web de comercio electrónico. Como resultado obtienen un producto significativo, en relación al rendimiento, disponibilidad y tiempo de respuesta en comparación a un aplicativo web basado en un modelo monolítico. | • Kubernetes<br>• Doker<br>• API Gateway<br>• Microservicios<br>• Servidor proxy NGINX<br>• Componente PVC | Como resultado se obtuvo una aplicación web funcional con una mejora significativa en un 104 %, en los indicadores de rendimiento, disponibilidad y tiempo de respuesta, en comparación con una aplicación web basado en el modelo monolítico. Por lo que el modelo de composición de microservicios presentado tiene un funcionamiento significativo. | (Ruelas, 2017) |
| Kubernetes Cluster for Automating Software Production Environment | Analizar y evaluar un clúster de Kubernetes un entorno de producción de software. Con ello, procede a realizar una comparación de dos métodos para implementar un clúster. | • Kubernetes: Kops y eksctl. Ambos métodos se refieren a la nube de AWS.<br>• Docker | Los resultados obtenidos evidenciaron que cada método se puede utilizar para diferentes casos de uso. Por ejemplo: kops permite más configuración, mientras que eksctl es más fácil de configurar. El método que se consideró más rápido en relación con las operaciones de clúster y más económico fue kops. | (Poniszewska y Czechowska, 2021) |

| TÍTULO | RESUMEN | HERRAMIENTAS A USAR | RESULTADO | AUTOR |
|---|---|---|---|---|
| | | • DevOps | | |
| GitOps basiertes Continuous Delivery für Serverless Anwendungen | Desarrollar y prototipar soluciones para aplicaciones sin servidor. | • FaaS<br>• DevOps<br>• CI/CD<br>• GitOps | El resultado del trabajo demostró que la tecnología es aún joven en cuanto a soporte de herramientas, normalización e investigación. Además, se espera que la brecha entre la orquestación sin servidor y la orquestación de contenedores acabe por cerrarse. | (Sahin, 2019) |
| Metodología GitOps para despliegue de aplicaciones basadas en microservicios | Demostrar la metodología para desplegar, esta propuesta es considerada más ágil porque se adapta bien a la integración CI/CD. Para lograrlo, se despliega un clúster de Kubernetes, en un entorno virtual, alojado en un ordenador portátil, quienes servirán como un marco para demostrar este despliegue con estas tecnologías en auge. | • CI/CD<br>• GitOps<br>• Kubernetes<br>• AWS (EKS) | Como resultado indicaron que estas prácticas, cada vez más demandadas por las empresas, están orientadas a minimizar en lo posible el error humano y aumentar la eficiencia, resiliencia y agilidad, tanto de las aplicaciones como de las infraestructuras. Por con esta implementación de solución en un entorno real productivo bajo un proveedor de nube pública eligiendo el modelo de Kubernetes gestionado de Amazon Web Services (EKS) para llevar a cabo esta tarea fue una forma eficiente y funcional ofreciendo excelentes resultados. | (Maderuelo, 2021) |
| Towards Network-Aware Resource Provisioning in Kubernetes for Fog Computing Applications | Kubernetes utiliza su propia herramienta llamada Kubelet para agrupar los nodos, donde todos los nodos son Docker Hosts. Utilizando Docker multinode mode, y la recientemente introducida herramienta Kubeadm es posible crear un clúster personalizado, añadir un nodo Fog al clúster y eliminar un nodo del clúster. Docker es una solución personalizada que es principalmente para aprender. | • Kubernetes<br>• Kubelet<br>• Docker Hosts<br>• Swarm<br>• Multinode<br>• Kubeadm | Los resultados muestran que el enfoque propuesto logra reducciones del 80% en términos de latencia de la red en comparación con el mecanismo de programación predeterminado. | (Santos et al., 2019) |
| Computación de Altas Prestaciones en entornos contenerizados | Kubeadm es una mejor característica y es muy similar a cómo funciona el modo Docker Swarm funciona. Pero Kubeadm no está listo por ahora con todas las características. Otra ventaja de Docker multinode y Kubeadm es que pueden ejecutar Kubernetes en Raspberry Pi. | • Kubernetes<br>• Quay.io<br>• Docker Hub | De los resultados obtenidos se puede indicar que Kubernetes sería una opción viable frente a la ejecución tradicional basadas en sistemas de colas. | (Pérez, 2021) |
| MASKDADOS.com, la web de ayuda a los juegos de rol desarrollada en Google Cloud Platform usando CICD con Jenkins y aplicando la escalabilidad de Kubernetes. | Artículo en el cual se refleja todo el proceso de desarrollo de una aplicación web utilizando las tecnologías actuales de Cloud Computing. | • Google Kubernetes Engine<br>• Google Cloud<br>• Cloud computing<br>• GCP<br>• Jenkins<br>• CICD<br>• Docker<br>• Spring Boot<br>• Angular<br>• JWT<br>• GCP<br>• MongoDB | El resultado del trabajo se basó en empleo de la metodología CICD a lo largo de todo el proceso y con el uso de la plataforma de Google Cloud Platform permitió crear toda la infraestructura y los servicios necesarios para operar en la nube y obteniendo excelentes resultados. | (Romero, 2020) |

| TÍTULO | RESUMEN | HERRAMIENTAS A USAR | RESULTADO | AUTOR |
|---|---|---|---|---|
| Unified Management of Applications on Heterogeneous. | Define la trans-cloud como una nueva extensión que agrupa la administración de diversos proveedores y escalas de servicios, IaaS y PaaS, en una misma API y empleo del estándar TOSCA para referir aplicativos portables y agnósticas, conteniendo procesos automatizados. | • IaaS<br>• PaaS<br>• Estándar TOSCA | El resultado obtenido por medio de la transnube proporciona funcionalidades y capacidades para hacer frente a muchos problemas relacionados con la dependencia del proveedor, proporcionando a los usuarios mecanismos robustos y flexibles para desplegar y gestionar sus aplicaciones, reaccionar ante los cambios y asegurar el comportamiento de sus sistemas en un entorno de nube heterogéneo, desvinculándolos de los detalles y la complejidad del proveedor. Por lo tanto, este desacoplamiento ofrece a los desarrolladores una forma de tratar las cuestiones de QoS y SLA y la flexibilidad de cambiar sus aplicaciones para llegar a nuevos proveedores. | (Carrasco, 2021) |
| Sistema de recomendación automático de servicios Multi-cloud. | Analiza y caracteriza los diferentes tipos de negocios involucrados en la creación de aplicaciones que podrían utilizar Multi-cloud. | • Azure<br>• Amazon Web Services<br>• Google Cloud Platform | Como resultado, se obtuvo un conjunto de características fundamentales para comparar los servicios Cloud de diferentes proveedores y recomendar los mejores servicios en función directamente de las características del usuario. | (Diaz y Matta, 2020) |
| Despliegue de SQL Server sobre Kubernetes | Realizar una extensa investigación sobre lo que Kubernetes puede ofrecer y qué problemáticas puede resolver. | • Kubernetes<br>• SQL | Como resultado, se menciona que el empleo de Kubernetes se convertirá en un estándar ya que es una herramienta que agiliza los procesos de producción que son tediosos y laboriosos. | (Gil, 2020) |
| Timing Channel en IaaS: cómo identificar e investigar | Investiga el comportamiento subyacente del canal de tiempo desde la perspectiva de los registros de actividad de la memoria y resumimos la firma del canal de tiempo en las actividades de memoria subyacentes. | • IaaS<br>• Amazon EC2 | Como resultado en este artículo, se presentó una varianza y∈ método de similitud del canal de red, donde ambos tuvieron un buen resultado. | (Fu et al., 2018) |
| Hacia la orquestación de contenedores en las infraestructuras de computación en la niebla. | Analiza como Kubernetes utiliza su propia herramienta llamada Kubelet para agrupar los nodos, donde todos los nodos son Docker Hosts. Utilizando Docker multinode mode, y la recientemente introducida herramienta Kubeadm es posible crear un clúster personalizado, añadir un nodo Fog al clúster y eliminar un nodo del clúster. Docker es una solución personalizada que es principalmente para aprender. Kubeadm es una mejor característica y es muy similar a cómo funciona el modo Docker Swarm funciona. | • Kubernetes<br>• Kubelet<br>• Docker multinode mode<br>• Kubeadm<br>• Nodo Fog<br>• Docker Swarm | Como resultado del desarrollo de plataformas de software productivas y atractivas y aplicaciones distribuidas aprovechando Kubernetes donde se engloban el aprovisionamiento de hardware, la instalación y gestión de clusters de servidores, gestión de la configuración, integración continua y el despliegue continuo, proporcionando excelentes resultados y flujos de trabajo organizados. | (Hoque et al., 2017) |
| Un modelo estocástico para investigar el rendimiento y la calidad del centro de datos en sistemas de computación en la nube de IaaS. | Presenta un modelo analítico, basado en redes de recompensa estocásticas (SRN), que es escalable para modelar sistemas compuestos por miles de recursos y flexible representando diferentes políticas y estrategias específicas de la nube. | • Redes de recompensa estocásticas (SRN)<br>• IaaS | Como resultado en este trabajo, se presenta un modelo estocástico para evaluar el rendimiento de un sistema de nube IaaS. Se han definido varias métricas de rendimiento, como la disponibilidad, la utilización y la capacidad de respuesta, lo que permitió investigar el impacto de diferentes estrategias tanto desde el punto de vista del proveedor como del usuario y se obtuvieron resultados excelentes. | (Bruneo, 2014) |
| Recomendación de arquitectura de servidor consciente del rendimiento y tecnología de verificación automática del rendimiento en la nube IaaS. | Propone una arquitectura de servidor y una tecnología de verificación automática del rendimiento, que recomienda y verifica la arquitectura de servidor adecuada en la nube de Infraestructura como servicio (IaaS) con servidores bare metal, servidores virtuales basados en contenedores y máquinas virtuales. Recientemente, los servicios en la nube se han extendido y los proveedores proporcionan no solo máquinas virtuales, sino también servidores bare metal y servidores | • IaaS<br>• Jenkis<br>• Contenedores Docker.<br>• Máquinas virtuales KVM<br>• OpenStack Heat<br>• CloudStack | Como resultado y una vez aplicadas las tecnologías propuestas, se logró confirmar el rendimiento y se concluye que sólo se necesita una hora para el aprovisionamiento y las verificaciones de rendimiento. Dado que los usuarios necesitan mucho más tiempo para diseñar la arquitectura del servidor por sí mismos, y la evolución de las tecnologías propuestas evidencio que son eficaces para reducir el esfuerzo de los usuarios. | (Yamato, 2016) |

| TÍTULO | RESUMEN | HERRAMIENTAS A USAR | RESULTADO | AUTOR |
|---|---|---|---|---|
| | virtuales basados en contenedores. | • Amazon Web Services (AWS) | | |
| Implementación y Evaluación de Plataformas en la Nube para Servicios de IoT. | En este modelo de servicio, el usuario no tiene control de la plataforma ni la infraestructura, pero sí de sus aplicaciones y datos. El proveedor dispone de control del software y la arquitectura. Referente a la instalación de escenarios en la nube para los servicios 5G-Iot existen respuestas comerciales como, por ejemplo, Microsoft Azure, Amazon Web Services, Google Cloud IoT, entre otros, y soluciones privadas que son diseñadas a través de un servidor o servidores vinculados que engloban hardware y software con el propósito de proporcionar de control y monitoreo. | • Google Cloud IoT<br>• Microsoft Azure<br>• Amazon Web Services. | Como resultado se realizaron mediciones en tiempo real en una aplicación de Smart Agriculture IoT donde se monitorizaron parámetros como la temperatura y humedad de una planta. A este respecto, se comprobó el correcto funcionamiento de cada uno de los elementos de la arquitectura del sistema IoT trabajando tanto de forma individual como junta. En la aplicación web, se verificaron exhaustivamente todas las funcionalidades, tales como, inicio de sesión, registro de usuarios, modificación de datos, alta y baja de servicios de IoT, consulta de datos en tiempo real a través de la visualización en gráficas y estadísticas lo que arrojo excelentes resultados de los mismos. | (Ortiz, 2019) |
| Evaluating Energy Consumption in a Different Virtualization within a Cloud System | Proporciona información sobre el impacto de la carga de trabajo en los servicios de computación en la nube al medir el uso de energía de tres tecnologías de virtualización diferentes, a saber, máquina virtual, Docker y Kubernete en un sistema Openstack, entorno de computación Opensource Cloud. | • Kubernetes<br>• Docker | El resultado del experimento muestra que<br>al medimos el uso de energía de diferentes virtualizaciones en un entorno de computación en nube con carga de trabajo. Docker, Virtual Machine y Kubernetes responden de forma diferente a una carga de trabajo que influyen en el consumo de energía. Por otra parte, la técnica de medición ha demostrado ser útil para medir la energía en diferentes virtualizaciones en la parte superior del sistema de computación en nube. Docker y Kubernetes consumen ligeramente una cantidad similar de energía cuando llegan al sistema 50 usuarios por segundo.<br>La máquina virtual consume la menor cantidad de energía y se determinó que las diferentes virtualizaciones tienen un comportamiento diferente en responder a una carga de trabajo similar. | (Murwantara y Yugopuspito, 2018) |
| Un estudio de los Contenedores Linux y su capacidad para ofrecer rápidamente escalabilidad para los servicios web Usando Kubernetes y Docker | Realizo una investigación de la capacidad de los contenedores Linux para permitir un rápido escalado de los servicios web, por lo que se realizó una comparación de este nuevo tipo de tecnología con otra más utilizada, que son las máquinas virtuales basadas en plantillas. Seguidamente, se realizó un primer experimento pretendía probar cuánto tiempo se tardaba en lanzar un nuevo servidor web, mientras que el segundo experimento se utilizó para probar cuánto tiempo se tardaba en sustituir un servidor web por un servidor de base de datos recién lanzado. Kubernetes y Docker fueron el software utilizado para los contenedores Linux y MLN y Openstack se utilizaron para las plantillas de máquinas virtuales. | • Kubernetes<br>• Docker<br>• Linux<br>• MLN<br>• Openstack<br>• Máquinas virtuales | Los resultados, permitieron evidenciar que Kubernetes debería ser elegido sobre MLN cuando se considera la escalabilidad rápida. Aunque la investigación y la teoría de los contenedores de Linux indican que son mucho más rápidos que las máquinas virtuales y las plantillas, no se puede concluir que todo el software que utiliza contenedores Linux tendría un mejor rendimiento que el software que utiliza máquinas virtuales y plantillas. Sin embargo, es probable asumir que sería el caso en la mayoría de las configuraciones. | Steinholt (2015) |
| Integración continua y despliegue de aplicaciones con la tecnología Kubernetes Student | Analiza la tecnología Kubernetes y las herramientas de despliegue comerciales y de código abierto disponibles adecuadas para esta tecnología. Así pues, se diseña una solución adecuada para el despliegue de aplicaciones utilizando algunas de estas herramientas de código abierto y proceder a implementar esta solución. | • Kubernetes<br>• Pipeline de código abierto | Los resultados de un repositorio. Indican que sigue siendo una arquitectura de microservicios. Cada desarrollador se integra en el repositorio de su microservicio. Ese microservicio se construye y se despliega independientemente en otros microservicios | Šmíd (2020) |
| Kubernetes como enfoque para resolver problemas bioinformáticos | Efectuaron una evaluación de la aplicabilidad de Kubernetes como enfoque para ejecutar flujos de trabajo bioinformáticos y se propusieron algunos ejemplos de cómo Kubernetes y los contenedores podrían utilizarse dentro del campo de las ciencias de la vida y cómo no deberían utilizarse, dicha propuesta resultante sirvió como prueba de concepto de cómo se realiza la | • Kubernetes<br>• Contenedores | | Markstedt (2017) |

| TÍTULO | RESUMEN | HERRAMIENTAS A USAR | RESULTADO | AUTOR |
|---|---|---|---|---|
| | implementación. | | | |
| Despliegue de una aplicación mediante Docker y Kubernetes | Se creó un Dockerfile, que forma la imagen base para el despliegue. La imagen se utiliza entonces para desplegar en Kubernetes ya que se necesitaba encontrar una forma rápida y eficiente de desplegar nuevas versiones de la aplicación en un entorno de pruebas y de producción. | • Docker<br>• Kubernetes<br>• Dockerfile | Los resultados indican que, mediante el uso de contenedores, el software no tendrá que instalarse en el entorno, sino que se pondrá en contenedores y seguirá siendo reutilizable por otros usuarios. Kubernetes proporciona varios servicios que son beneficiosos para llevar a cabo la investigación en ciencias de la vida. Proporciona una buena arquitectura para construir sobre ella debido a su naturaleza altamente personalizable y el objetivo del proyecto fue principalmente evaluar e ilustrar cómo Kubernetes y los contenedores pueden ser utilizados en el campo de la investigación en ciencias de la vida, ha demostrado ser una herramienta adecuada para este propósito. | Moilanen (2018) |

**Tabla II:** Resumen de fuentes bibliográficas

# II. Metodología

Seguidamente, se presenta en la figura 3 un diagrama de flujo con el proceso de canalización CI/CD, iniciando con el código declarativo, el cual se ejecuta dentro de Terraform, con la posterior construcción del repositorio GitHub y a través del WebHooks el cual es el encargado de realizar una constante verificación de cambios del código declarativo y generar la compilación, luego por medio de CI/CD Pipeline realizar el aprovisionamiento.

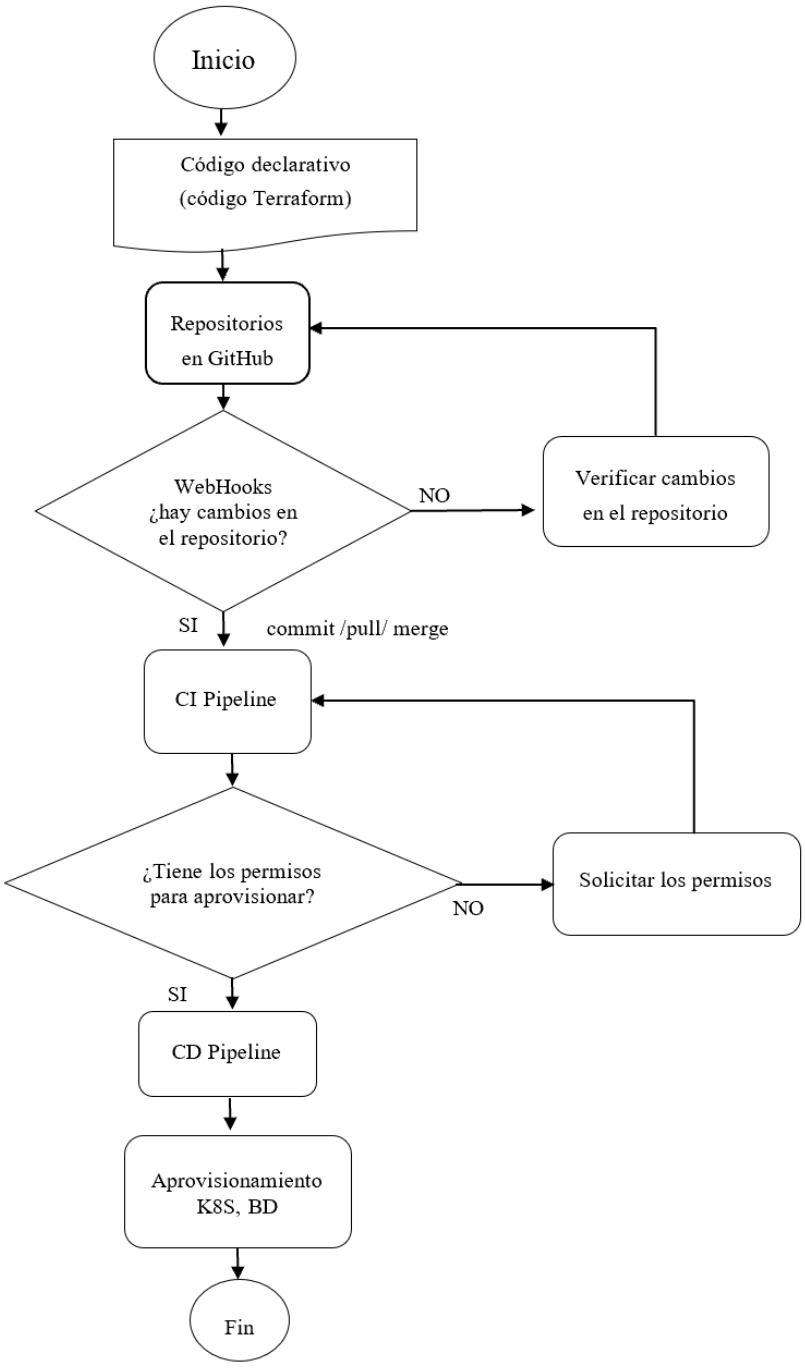


**Figura 3** Método propuesto para el aprovisionamiento K8S y BD

A continuación, se presenta el diagrama de la figura 4, donde se presentan los procesos para poder realizar la implementación Odoo. Una vez comprobados las credenciales y la integración (CI) se ejecuta el Pipeline el cual es el encargado de generar el contenedor Odoo y desplegar en la infraestructura Kubernetes.

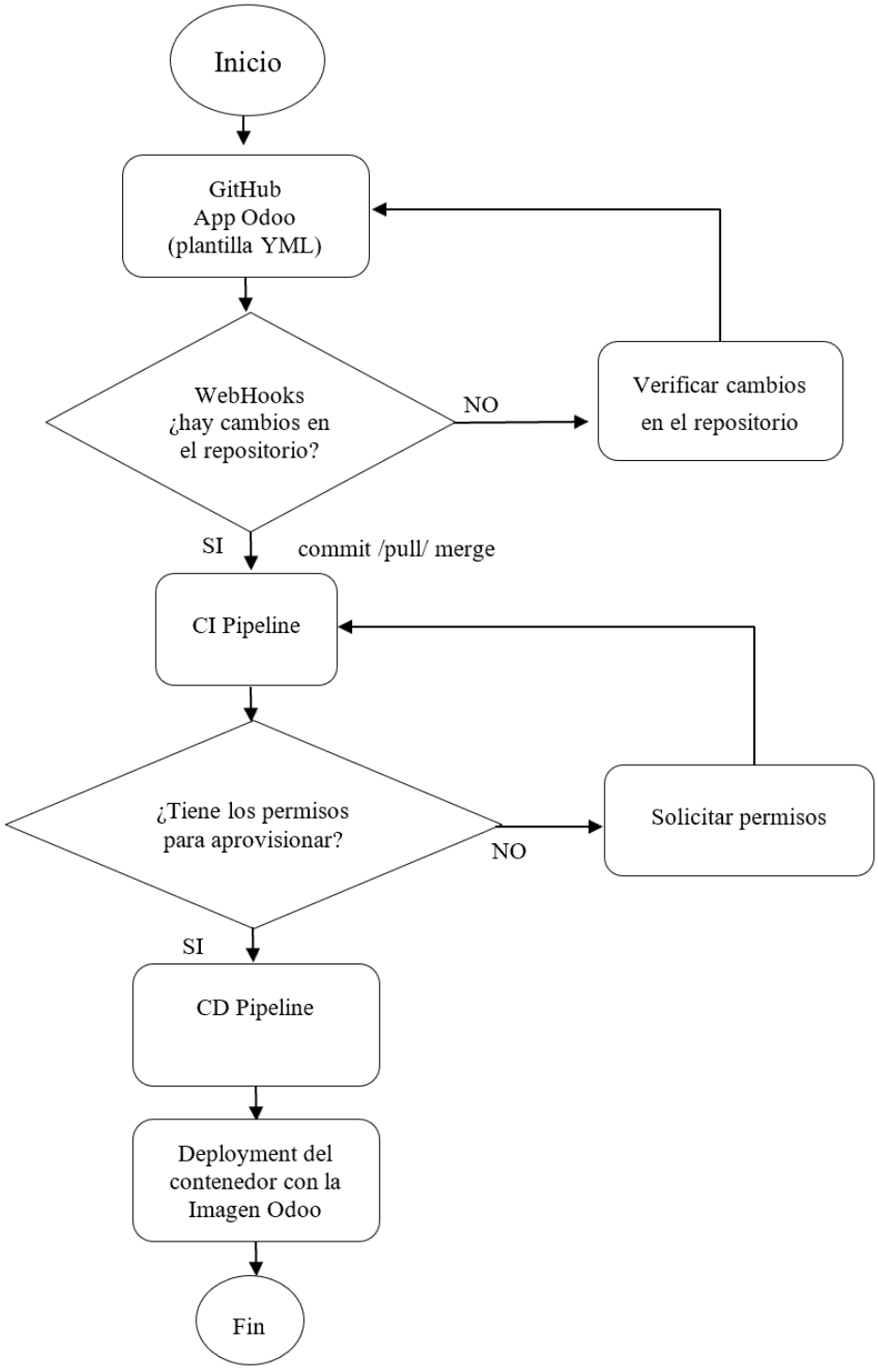


**Figura 4** Método propuesto para el despliegue del contenedor Odoo

## 2.1. Flujo de trabajo

El método propuesto en este trabajo consiste en 4 fases. En cada fase se define los pasos a ejecutar para llegar con éxito este proyecto. Una vez definido las fases se presenta dos diseños el primer diseño servirá para aprovisionar Kubernetes y la BD, y el segundo diseño permitirá ejecutar la aplicación Odoo sobre el Kubernetes correspondiente.

**Fase 1: Repositorio de control de versiones**

- Lo investigado es referente a GitHub, pero se puede aplicar para cualquier tecnología de control de versiones tales como BitBucket, Azure DevOps, GitLab, etc.
- El desarrollador crea los repositorios para cada proveedor Cloud (Azure, Google, Amazon) y carga el código de infraestructura para realizar el aprovisionamiento de Kubernetes, base de datos y algunos servicios propios según el proveedor de nube.
- Adicionalmente se crea otros repositorios para el despliegue de Odoo en cada proveedor de nube.
- Para iniciar el proceso de integración continua (CI), se procede a configurar WebHooks en GitHub que permitirá la ejecución automatizada conforme al estado de control de versión en los repositorios (commit, pull request, merge, etc.).

**Fase 2: Preparación de la herramienta para integración y despliegue continuo**

- En la investigación se trabajó con Jenkins, pero se puede utilizar otros productos similares tales como GitLab, Bamboo, Azure DevOps.
- Aprovisionar el servidor Jenkins en una máquina virtual en cualquier proveedor de nube. El presente trabajo se construyó y se configuró el servidor Jenkins en Microsoft Azure.
- Se instala Kubectl y Terraform en el servidor Jenkins.
- Se instala las llaves de seguridad para el acceso desde Jenkins al proveedor de nube que corresponde (IAM) y las variables de entorno para el aprovisionamiento de la infraestructura requerida para el proyecto.
- Las credenciales tienen una configuración de autorización que le permite a Jenkins tener los permisos mínimo necesario para la construcción de la infraestructura.

- Los secretos son almacenados en el vault de Jenkins que es un componente seguro que guarda las claves en formato hash.

**Fase 3: Infraestructura como código mediante Terraform y Jenkins**

- Jenkins crea los recursos de nube a través de la Infraestructura como código utilizando Terraform, se pudo haber trabajado con Azure Resource Template o Cloud Formation (AWS) sin embargo Terraform nos aporta la flexibilidad de ser independiente del proveedor de nube.
- Jenkins ejecuta un script Terraform el cual trabaja con un plan de ejecución, el plan contiene los detalles de los recursos que se desea crear y el versionamiento del mismo.
- Mediante el despliegue continuo (CD) el pipeline de Jenkins se conecta al repositorio de GitHub donde se encuentra el script Terraform de creación de recursos, construye el servicio AKS, EKS, GKE (según el proveedor de nube), la Base de datos PostgreSQL y recursos adicionales propios de cada nube (Resource group para Azure, VPC para AWS, Project para Google Cloud, etc.).
- El Script de Terraform contiene datos para identificar el proveedor de nube, credenciales de acceso, código de aprovisionamiento de recursos y finalmente la ejecución a través de los comandos Terraform Init y Terraform Apply.

**Fase 4: Despliegue continuo de la Aplicación Odoo mediante Jenkins**

- En Jenkins creamos un segundo pipeline que se conecta al repositorio de GitHub para ejecutar el archivo YML de Kubernetes y desplegar la aplicación Odoo al Cluster.
- Este archivo YML contiene las instrucciones para crear los diversos componentes que requiere un Cluster de Kubernetes tales como el Deploy, ReplicaSet, Pods, Services, Ingress, Balanceador, etc. y finalmente descargar la imagen Odoo desde DockerHub y crear los contenedores para la aplicación.

## 2.2. Diseño de la arquitectura GitOps

Una canalización de CI/CD automatiza su proceso de entrega de software. La canalización crea código, ejecuta pruebas (CI) e implementa de manera segura una nueva versión de la aplicación (CD). Las canalizaciones automatizadas eliminan errores manuales, brindan ciclos de retroalimentación estandarizados a los desarrolladores y permiten iteraciones rápidas de productos, Asimismo, CI abreviatura de Integración continua, es una práctica de desarrollo de software en la que todos los

desarrolladores fusionan los cambios de código en un repositorio central. CD significa entrega continua, que además de la Integración continua agrega la práctica de automatizar todo el proceso de lanzamiento de software. Con CI, cada cambio en el código desencadena una secuencia automatizada de compilación y prueba para el proyecto determinado. Por consiguiente, la entrega continua (CD) incluye el aprovisionamiento y la implementación de la infraestructura, que puede ser manual y constar de varias etapas. Lo importante es que todos estos procesos estén completamente automatizados, con cada ejecución totalmente registrada y visible para todo el equipo.

En la **Figura 5** se presenta el diseño para implementar AKS, EKS, GKE y la BD.

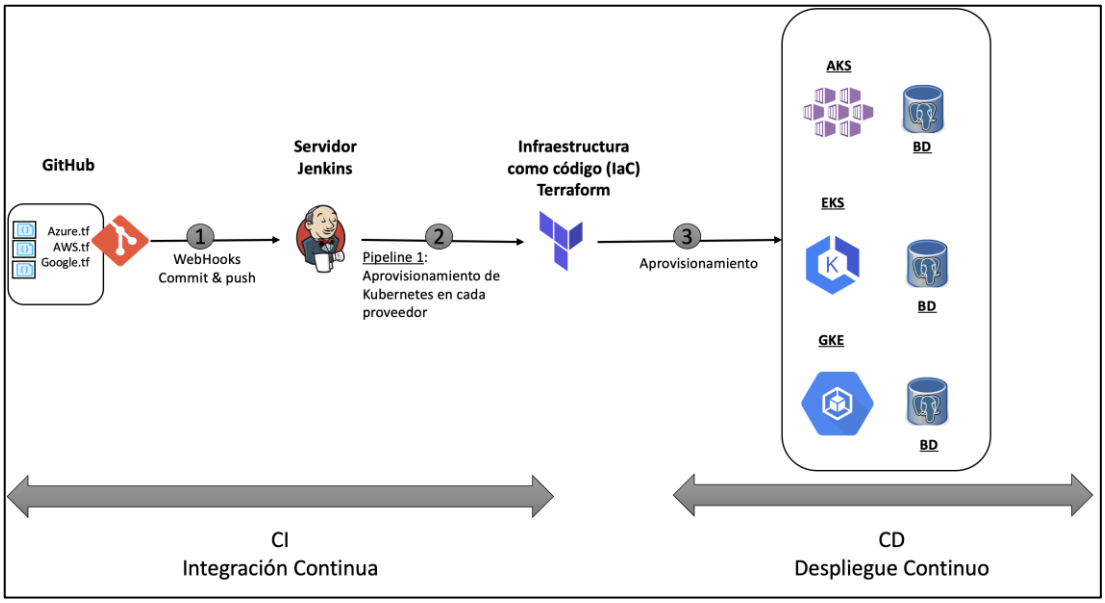


**Figura 5** Arquitectura de GitOps con infraestructura basada en código (elaboración propia)

### GitHub

En GitHub se procede a crear los repositorios para cada proveedor de nube publica de forma organizada y se define el código Terraform (.tf) para el aprovisionamiento de la Base de Datos, Kubernetes. (Ver Anexo 1 **Tabla IV**).

### WebHooks

Es un sistema que permite la comunicación automática entre aplicaciones o servidores. Se utilizará para crear la comunicación entre GitHub y el servidor Jenkins. WebHooks está en alerta constantes cuando surja algún evento en el repositorio GitHub (commit, pull, merge, etc). Esto permite ganar tiempo y mantener el proyecto actualizado. Su función es estar en alerta y escuchar el evento, para recopilar la información y enviar al servidor Jenkins. Si no existen errores en el código Terraform (.tf) o archivos YML, el servidor de Jenkins procede a realizar automáticamente la canalización CI/CD en la

nube correspondiente. Para configurar los eventos que escuchara WebHooks se debe especificar la dirección IP del servidor de Jenkins en GitHub (ver Anexo 1 **Figura 15**) y en el servidor Jenkins habilitar el trigger (ver Anexo 1 **Figura 16**).

**Servidor Jenkins**

A continuación, se construye una VM Linux y se instala el servidor Jenkins. Se crea un script llamado "code cloud-init-jenkins.txt" (ver **Figura 17**). En el script se implementa la instalación JDK, se agrega la clave del repositorio al sistema y la dirección del paquete Debian "echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list" y es incorporada en "source.list". También se realiza un update y finalmente se procede a la instalación de Jenkins y sus dependencias. En el CLI de Azure, se crea un grupo de recursos (RG) llamado "azrgdevopsd01" (ver **Figura 18** ), dentro del RG se crea una VM con los siguientes parámetros: el nombre del host, nombre de usuario, contraseña, claves de acceso SSH. El script es nombrado como "cloud-init-jenkins.txt" donde está la información personalizada de Jenkins, que permitirá la inicialización del servidor.

En el servidor Jenkins se crea los pipelines para realizar el despliegue del Cluster AKS, EKS, GKE y la implementación de la aplicación Odoo, especificando el token de GitHub y el nombre del repositorio. Se construyó en total 6 pipelines (ver **Figura 19**).

**Configuración para el despliegue Continuo (CD)**

Para establecer la seguridad durante la canalización CI entre el servidor Jenkins y el proveedor de nube correspondiente, se debe crear el Service principal y el usuario IAM. Se construye el usuario IAM y se especifica el rol que va a adoptar en el proyecto, se define las políticas o permisos para limitar y evitar el acceso a ciertos recursos. Las credenciales o claves que proporciona cada usuario IAM, será importante para realizar la canalización CI/CD desde el servidor Jenkins al cloud correspondiente. En la creación del usuario IAM para cada cloud (ver Tabla V).

El Service principal es una entidad de seguridad que es solicitado por un usuario IAM para acceder e implementar aplicaciones, servicios o herramientas automatizadas, esto mejora la seguridad en la canalización CI/CD. Cuando se crea el Service Principal, se obtiene las credenciales que permitirá acceder a los servicios y aprovisionar. En el anexo 1 se detalla la construcción del Service principal para Azure (ver **Figura 20**), AWS (ver **Figura 21**) y Google (ver **Figura 22**).

El usuario IAM obtiene la información del Service Principal y procede en el servidor Jenkins la configuración de las variables con la información sensible. El código Terraform (.tf) clonado en GitHub solicitará el valor de cada variable (ver **Figura 23**,

**Figura 25**, **Figura 27**) y establecerá una conexión segura con el Cloud respectivo. Este proceso verificará el acceso y permitirá la canalización CI/CD en el proveedor solicitado. En el servidor Jenkins se configura las variables con la información sensible para Azure (ver **Figura 24**), Google (ver **Figura 26**), AWS (ver **Figura 28**).

**Código fuente con Terraform para implementar AKS, EKS, GKE y BD**

En el anexo 1 del apartado 7 se detalla el código Terraform para proveer AKS, GKE y EKS, además se implementa el servicio de una BD PostgreSQL para cada Cluster Kubernetes, la BD es primordial para el funcionamiento de la aplicación Odoo. Código fuente para Azure (**Figura 29**), Google (**Figura 33**), Amazon (**Figura 37**).

**Integración continua y entrega continua en Odoo**

**Integración continua:**

La integración continua (CI) integrar los productos de trabajo de los desarrolladores individuales en un repositorio central de forma temprana y frecuente. El objetivo de la CI es convertir el proceso de integración en una tarea de desarrollo sencilla, fácilmente repetible y cotidiana para reducir los costes totales de construcción y revelar los defectos en las primeras fases del ciclo.

**Proceso**

- Con CI, los desarrolladores pueden integrar su código en un repositorio común.
- En lugar de construir funciones de forma aislada y enviar cada una de ellas al final del ciclo.
- Cada vez que se introduce el código, el sistema inicia el proceso de compilación, ejecuta pruebas unitarias y otras comprobaciones relacionadas con la calidad, según sea necesario.

**Dependencias**

La IC depende en gran medida de los conjuntos de pruebas y de una ejecución de pruebas automatizada. Cuando se hace correctamente, permite a los desarrolladores realizar compilaciones frecuentes e iterativas y tratar los errores en una fase temprana del ciclo de vida.

**Entrega continua**

El objetivo de la entrega continua (CD) es automatizar el proceso de entrega de software para permitir un despliegue fácil y seguro en producción, en cualquier momento. El objetivo principal CD es producir software en ciclos cortos para que las

nuevas características y los cambios puedan ser liberados de forma rápida, segura y fiable en cualquier momento.

**Proceso**

- El CD ejecuta un conjunto progresivo de pruebas en cada compilación y alerta al equipo de desarrollo en caso de fallo para que este lo rectifique
- En las situaciones en las que no hay problemas, CD realiza las pruebas de forma secuencial.
- El resultado final es una compilación que puede desplegarse y verificarse en un entorno de producción real.

**Dependencias**

Dado que el objetivo de la CD es crear, probar y publicar software con rapidez y frecuencia, depende de un sistema automatizado que ayude al equipo de desarrollo a automatizar los procesos de prueba y despliegue.

En la **Figura 6** se presenta el siguiente diseño para realizar el despliegue continuo (CD) de la aplicación Odoo.

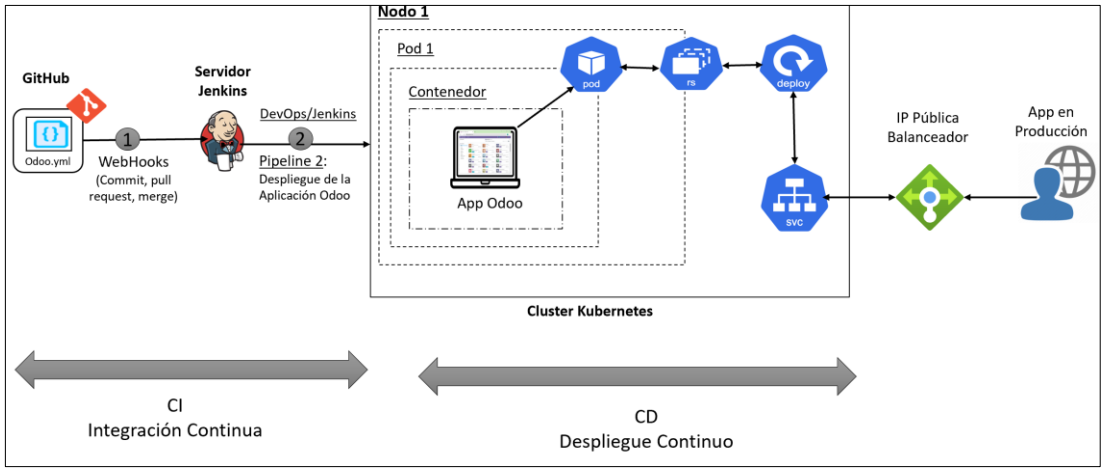


**Figura 6** Arquitectura de GitOps con Despliegue continuo para la aplicación Odoo (elaboración propia)

**Implementación del contenedor Odoo**

Odoo es una herramienta ERP y CRM de código abierto. Contiene una sólida infraestructura técnica que puede funcionar sin problemas con todas las aplicaciones. Lanzado por primera vez en 2005. El código fuente está escrito en Python. Aunque el sistema tiene la licencia GPL 3, también ofrece un SaaS (software como servicio). adicionalmente ofrece una variedad de aplicaciones fáciles de usar como CRP,

contabilidad, facturas, creador de sitios web, comercio electrónico, marketing por correo electrónico, gestión de proyectos, inventario, admite módulos útiles que incluyen gestión de proyectos, adicionalmente estos módulos pueden comunicarse entre sí para procesar todos los datos sin esfuerzo. Por otra parte, entre sus características claves se pueden mencionar:

- Proporcionar una interfaz de usuario fácil e intuitiva.
- Ofrece un diseño de interfaz de usuario intuitivo y moderno para trabaja
- Admite aplicaciones de terceros; desarrollado por los miembros de la comunidad de Odoo.

GitHub automatiza las actualizaciones de la infraestructura mediante un flujo de trabajo de Git con integración continua (CI) y entrega continua (CD). Cuando se fusiona código nuevo, la canalización de CI/CD publica el cambio en el entorno. Cualquier cambio de configuración, como cambios manuales o errores, se sobrescribe con la automatización de GitHub para que el entorno converja en el estado deseado definido en Git. GitHub usa canalizaciones de CI/CD para administrar e implementar la automatización.

Kubernetes utiliza los archivos YML para crear los objetos como Service, Deployment, ReplicaSet, Pods, etc. El archivo de configuración YML está determinada por 4 campos: La versión de API (apiVersion), el tipo (kind), metadatos y especificaciones (spec).

Por lo tanto, se define un archivo YML con los objetos y las especificaciones del Kubernetes (AKS, EKS, GKE), en la **Figura 43** se puede observar el script YML. El código YML utiliza la versión APPS/V1 para el API de Kubernetes que servirá para establecer los objetos. En el Kind se crea dos objetos la implementación (Deployment) y el servicio (Service). La metadata son los datos sobre cada objeto por ejemplo el nombre, las etiquetas, etc. El siguiente campo son las especificaciones (spec) que permitirá construir el contenedor con la imagen Odoo.

Además, se implementa los servicios necesarios del Cluster Kubernetes como son: Replica Set (RS), Deploy, Service (SVC), Load Balancer.

- **Replica Set (RS):** es la cantidad de Pods que se replicaran.
- **Deploy:** permite implementar un Pod o RS y se encarga de realizar actualizaciones declarativas para los Pods y ReplicaSets, ejecuta varias réplicas de la aplicación, además escala automáticamente la cantidad de Pods, asegurando que la aplicación siempre este activa.

- **Service (SVC):** se encarga de enrutar el tráfico de todos los Pods existentes, en caso de que los pods mueran se replicaran en el Kubernetes sin afectar la aplicación.
- **Load Balancer:** es un equilibrador de carga y su función es asignar una IP a la aplicación para que el usuario pueda utilizar los servicios.

Una vez creado el archivo YML se define la información del nodo, el pod, las etiquetas y el contenedor del Cluster AKS, EKS y GKE. (Ver **Figura 30**, **Figura 34**, **Figura 40**).

En la **Figura 7**, se visualiza el flujo de trabajo del pipeline para proveer la aplicación Odoo, el operador observa cada etapa con un mensaje de éxito en color verde y de fracaso en color rojo. En caso de que se presente algún error se puede visualizar cada registro (logs) para conocer en qué etapa se detuvo la implementación de los recursos y cuál fue el error. Si todo se ejecutó correctamente se implanta el contenedor Odoo en el contenedor correspondiente ya sea en AKS, GKE o EKS. En la **Figura 8** se indica el resultado de la implementación Odoo.

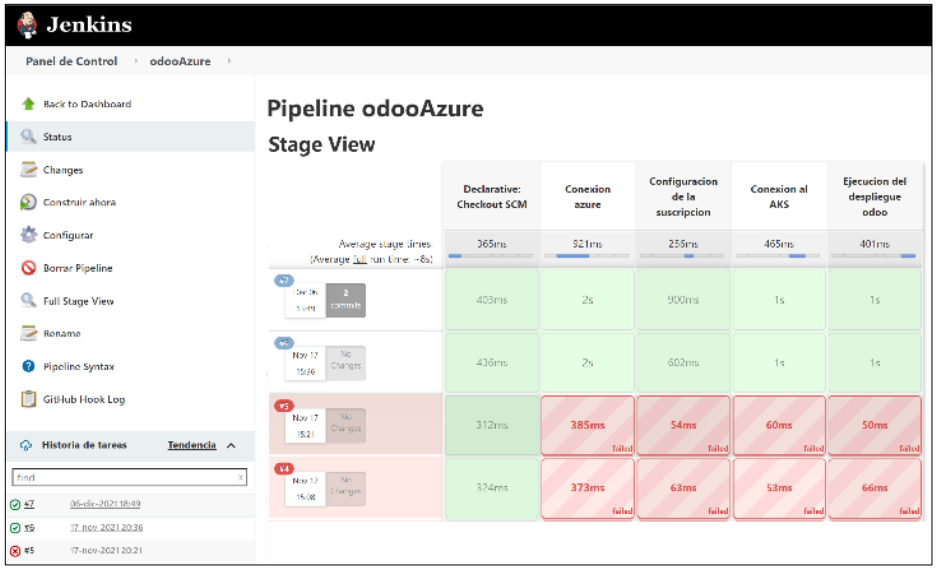


**Figura 7** Etapas del despliegue continuo de los scripts Terraform

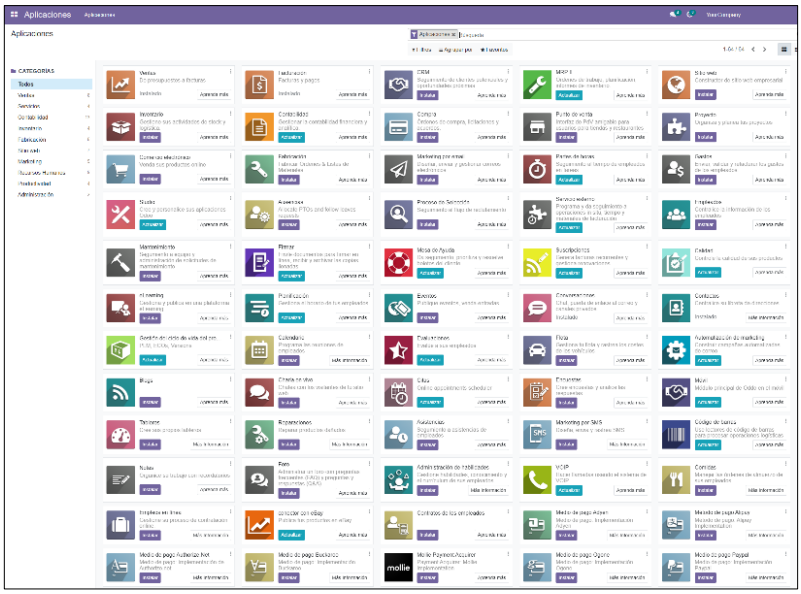

**Figura 8** Resultado de la App Odoo desplegado en AKS, EKS, GKE

# III. Resultados y discusión

Se utiliza la herramienta JMeter para realizar solicitudes HTTP y evaluar el rendimiento de la Base de datos, el Nodo y el porcentaje de error de las solicitudes de 100 muestras enviadas, por consiguiente, se obtienen los siguientes resultados. Para Azure se consigue un porcentaje de error de 14,83% es decir del total de peticiones 14 de las 100 no se ejecutaron y 86 se ejecutaron. Seguidamente con Google se alcanza un porcentaje de error de 23,59% es decir del total de peticiones 23 de las 100 no se ejecutaron y 77 se ejecutaron; por último, con AWS se obtiene un porcentaje de error de 8.08%, es decir, del total de peticiones, 8 de las 100 no se ejecutaron y 92 se ejecutaron.

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| /web/dataset/call_... | 100 | 42248 | 20441 | 67699 | 11042.17 | 11.00% | 16.9/min | 0.45 | 0.16 | 1639.8 |
| /web/dataset/call_... | 100 | 40225 | 20405 | 60073 | 10660.11 | 11.00% | 16.1/min | 0.42 | 0.13 | 1593.1 |
| /web/image/res.u... | 100 | 42765 | 21035 | 77908 | 11981.73 | 9.00% | 16.3/min | 1.65 | 0.08 | 6234.6 |
| /sales/sale_quotat... | 100 | 42892 | 21051 | 70971 | 9253.89 | 3.00% | 15.9/min | 0.39 | 0.14 | 1513.6 |
| /sale/static/src/im... | 100 | 4606 | 256 | 21077 | 6245.45 | 9.00% | 18.1/min | 9.79 | 0.10 | 33211.8 |
| /web/dataset/call_... | 100 | 38908 | 21023 | 70634 | 11898.54 | 20.00% | 15.6/min | 0.45 | 0.16 | 1757.4 |
| /web/dataset/call_... | 100 | 43136 | 17818 | 70994 | 11166.71 | 9.00% | 15.4/min | 0.40 | 0.17 | 1614.7 |
| /web/dataset/call_... | 100 | 44081 | 21019 | 89181 | 12968.50 | 13.00% | 14.4/min | 0.39 | 0.16 | 1643.3 |
| /web/dataset/call_... | 100 | 45339 | 21025 | 74628 | 12633.37 | 16.00% | 14.7/min | 0.37 | 0.29 | 1542.3 |
| /web/dataset/call_... | 100 | 45618 | 21025 | 95218 | 14002.39 | 14.00% | 14.9/min | 0.40 | 0.14 | 1632.9 |
| /web/dataset/call_... | 100 | 45117 | 18336 | 70962 | 13411.69 | 16.00% | 15.4/min | 0.39 | 0.16 | 1565.6 |
| /web/dataset/call_... | 100 | 45446 | 21018 | 65076 | 12047.60 | 17.00% | 16.0/min | 0.39 | 0.27 | 1485.3 |
| /web/dataset/call_... | 100 | 44577 | 21024 | 78348 | 12350.48 | 17.00% | 14.5/min | 0.36 | 0.15 | 1508.6 |
| /web/dataset/call_... | 100 | 42092 | 13840 | 71435 | 12814.46 | 14.00% | 14.6/min | 0.38 | 0.14 | 1609.6 |
| /web/dataset/call_... | 100 | 43489 | 21020 | 72543 | 12242.18 | 15.00% | 14.7/min | 0.39 | 0.13 | 1645.9 |
| /web/dataset/call_... | 100 | 42235 | 19776 | 82152 | 10908.24 | 8.00% | 14.6/min | 0.38 | 0.14 | 1578.4 |
| /web/dataset/call_... | 100 | 42527 | 20050 | 70210 | 11028.23 | 10.00% | 16.0/min | 0.41 | 0.16 | 1581.1 |
| /web/static/img/pl... | 100 | 2307 | 170 | 21044 | 4269.39 | 4.00% | 16.8/min | 1.71 | 0.09 | 6240.4 |
| /partner_autocom... | 100 | 2665 | 335 | 21085 | 5098.58 | 5.00% | 16.0/min | 11.81 | 0.08 | 45475.7 |
| /web/dataset/call_... | 100 | 41452 | 14273 | 70103 | 11107.02 | 7.00% | 15.0/min | 0.38 | 0.12 | 1565.5 |
| /mail/get_suggest... | 100 | 41426 | 14435 | 66864 | 10728.57 | 7.00% | 16.2/min | 0.41 | 0.12 | 1565.5 |
| /web/dataset/call_... | 100 | 38913 | 18626 | 77156 | 12837.00 | 10.00% | 17.1/min | 0.45 | 0.13 | 1627.7 |
| /web/dataset/call_... | 100 | 35064 | 13529 | 65932 | 12303.44 | 19.00% | 17.6/min | 0.49 | 0.15 | 1721.1 |
| /mail/read_followe... | 100 | 32425 | 11223 | 65224 | 10654.36 | 13.00% | 17.2/min | 0.47 | 0.11 | 1666.6 |
| /web/dataset/call_... | 100 | 28104 | 13991 | 52027 | 8259.24 | 35.00% | 18.6/min | 0.59 | 0.13 | 1952.0 |
| /web/static/img/fo... | 100 | 9842 | 172 | 21071 | 8801.41 | 32.00% | 20.5/min | 0.39 | 0.09 | 1156.2 |
| /mail/get_suggest... | 100 | 25669 | 7904 | 50254 | 7172.10 | 26.00% | 19.4/min | 0.58 | 0.11 | 1835.2 |
| /mail/read_followe... | 100 | 25873 | 15121 | 41143 | 6107.93 | 25.00% | 20.3/min | 0.60 | 0.12 | 1822.2 |
| /mail/thread/mess... | 100 | 25943 | 4447 | 50098 | 8161.86 | 24.00% | 22.0/min | 0.65 | 0.13 | 1809.3 |
| /web/dataset/call_... | 100 | 24371 | 2717 | 50122 | 8896.97 | 32.00% | 24.6/min | 0.77 | 0.15 | 1913.0 |
| /web/dataset/call_... | 100 | 22429 | 935 | 50215 | 10278.54 | 21.00% | 26.2/min | 0.75 | 0.18 | 1770.4 |
| /web/dataset/call_... | 100 | 19715 | 588 | 49424 | 11328.86 | 21.00% | 28.6/min | 0.82 | 0.22 | 1770.4 |
| /mail/static/src/im... | 100 | 6743 | 171 | 21070 | 7804.43 | 19.00% | 34.0/min | 1.42 | 0.15 | 2573.1 |
| TOTAL | 7600 | 34253 | 58 | 152788 | 24058.63 | 14.83% | 2.9/sec | 262.70 | 1.26 | 94219.3 |

**Figura 9** Resultados en JMeter para Azure con un porcentaje de error de 14,83%

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/... | Sent KB/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| /web/dataset/call_kw/crm.tag/read-45 | 100 | 2379 | 203 | 21055 | 4131.45 | 38.00% | 43.6/min | 0.82 | 0.46 | 1150.8 |
| /web/dataset/call_kw/res.users/has_group-46 | 100 | 1837 | 214 | 21041 | 3566.86 | 40.00% | 42.8/min | 0.79 | 0.37 | 1130.1 |
| /web/image/res.users/2/avatar_128-47 | 100 | 2089 | 211 | 21045 | 4176.11 | 43.00% | 41.9/min | 2.76 | 0.23 | 4045.9 |
| /sales/sale_quotation_onboarding_panel-48 | 100 | 1489 | 208 | 21049 | 2659.34 | 35.00% | 41.7/min | 0.79 | 0.36 | 1159.1 |
| /sale/static/src/img/sale_quotation_onboarding_bg.jp... | 100 | 2157 | 282 | 21027 | 3464.06 | 1.00% | 40.8/min | 23.83 | 0.24 | 35885.8 |
| /web/dataset/call_kw/mail.followers/load_views-50 | 100 | 1796 | 203 | 21060 | 3617.65 | 42.00% | 35.9/min | 0.65 | 0.44 | 1110.0 |
| /web/dataset/call_kw/mail.activity/load_views-51 | 100 | 1637 | 209 | 21025 | 2911.54 | 40.00% | 35.9/min | 0.65 | 0.45 | 1107.3 |
| /web/dataset/call_kw/mail.message/load_views-52 | 100 | 1874 | 211 | 21046 | 3508.10 | 39.00% | 35.9/min | 0.65 | 0.45 | 1117.7 |
| /web/dataset/call_kw/sale.order/read-53 | 100 | 1783 | 217 | 21026 | 3168.31 | 33.00% | 35.9/min | 0.69 | 0.78 | 1179.8 |
| /web/dataset/call_kw/crm.tag/read-56 | 100 | 2195 | 240 | 15318 | 3175.05 | 34.00% | 35.3/min | 0.66 | 0.38 | 1146.1 |
| /web/dataset/call_kw/mail.message/read-58 | 100 | 2307 | 212 | 21061 | 4314.56 | 33.00% | 35.1/min | 0.70 | 0.38 | 1226.5 |
| /web/dataset/call_kw/mail.activity/read-57 | 100 | 2317 | 205 | 21043 | 4408.05 | 37.00% | 35.1/min | 0.69 | 0.38 | 1208.4 |
| /web/dataset/call_kw/sale.order.line/read-54 | 100 | 1972 | 213 | 21059 | 4010.48 | 28.00% | 35.2/min | 0.72 | 0.63 | 1254.9 |
| /web/dataset/call_kw/mail.followers/read-55 | 100 | 2026 | 215 | 21050 | 4039.36 | 28.00% | 34.7/min | 0.72 | 0.38 | 1278.2 |
| /web/dataset/call_kw/res.partner/name_get-61 | 100 | 2108 | 214 | 21046 | 3763.94 | 31.00% | 31.9/min | 0.64 | 0.32 | 1223.8 |
| /web/dataset/call_kw/res.partner/name_get-59 | 100 | 2159 | 206 | 21060 | 4178.62 | 29.00% | 31.5/min | 0.65 | 0.34 | 1267.9 |
| /web/dataset/call_kw/res.partner/name_get-60 | 100 | 2793 | 205 | 21070 | 5208.51 | 31.00% | 31.3/min | 0.65 | 0.31 | 1270.5 |
| /web/static/img/placeholder.png-63 | 100 | 1903 | 187 | 21054 | 4152.49 | 3.00% | 31.3/min | 3.20 | 0.17 | 6276.4 |
| /partner_autocomplete/static/lib/jsvat.js-62 | 100 | 2029 | 372 | 21042 | 2972.05 | 1.00% | 30.8/min | 23.74 | 0.16 | 47272.8 |
| /web/dataset/call_kw/ir.attachment/search_read-72 | 100 | 1773 | 206 | 21055 | 3614.90 | 29.00% | 30.8/min | 0.62 | 0.32 | 1244.5 |
| /web/static/img/form_sheetbg.png-73 | 100 | 2113 | 186 | 21050 | 3472.59 | 1.00% | 30.6/min | 0.20 | 0.19 | 409.1 |
| /mail/get_suggested_recipients-71 | 100 | 2113 | 207 | 21055 | 4237.07 | 34.00% | 30.7/min | 0.61 | 0.23 | 1216.1 |
| /mail/read_followers-66 | 100 | 2034 | 206 | 21057 | 4717.41 | 33.00% | 30.7/min | 0.64 | 0.23 | 1273.2 |
| /web/dataset/call_kw/sale.order/read-69 | 100 | 2426 | 208 | 21047 | 4363.73 | 20.00% | 29.6/min | 0.64 | 0.25 | 1337.7 |
| /web/dataset/call_kw/ir.attachment/search_read-67 | 100 | 2058 | 207 | 21032 | 3416.55 | 25.00% | 29.9/min | 0.61 | 0.31 | 1262.6 |
| /web/dataset/call_kw/sale.order/read-65 | 100 | 2151 | 204 | 21032 | 4298.11 | 22.00% | 29.5/min | 0.64 | 0.25 | 1340.3 |
| /mail/get_suggested_recipients-68 | 100 | 2291 | 202 | 21056 | 4362.57 | 26.00% | 29.5/min | 0.61 | 0.23 | 1275.6 |
| /mail/read_followers-70 | 100 | 2184 | 204 | 21045 | 3782.37 | 19.00% | 29.1/min | 0.63 | 0.22 | 1324.7 |
| /mail/thread/messages-64 | 100 | 2466 | 203 | 21041 | 4623.51 | 23.00% | 28.7/min | 0.63 | 0.22 | 1353.3 |
| /web/dataset/call_kw/mail.activity/activity_format-74 | 100 | 2426 | 205 | 21044 | 4039.60 | 31.00% | 28.7/min | 0.57 | 0.25 | 1223.8 |
| /web/dataset/call_kw/mail.activity/activity_format-75 | 100 | 2315 | 207 | 21047 | 4067.85 | 25.00% | 28.7/min | 0.59 | 0.25 | 1262.6 |
| /mail/static/src/img/odoobot.png-77 | 100 | 2273 | 192 | 21051 | 4011.49 | 2.00% | 28.5/min | 1.17 | 0.15 | 2526.5 |
| /web/dataset/call_kw/mail.message/mark_all_as_rea... | 100 | 2222 | 209 | 21045 | 3537.61 | 23.00% | 28.3/min | 0.59 | 0.27 | 1283.3 |
| TOTAL | 7300 | 3338 | 58 | 70027 | 6945.54 | 23.59% | 21.9/sec | 2197.56 | 10.62 | 102842.0 |

**Figura 10** Resultados en JMeter en Google con un porcentaje de error de 23,59%

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| /web/image/res... | 200 | 9510 | 200 | 42340 | 11602.57 | 3.50% | 2.9/min | 0.06 | 0.02 | 1212.4 |
| /sales/sale_quotat... | 200 | 9728 | 226 | 42123 | 12920.28 | 7.50% | 2.9/min | 0.07 | 0.03 | 1611.8 |
| /sale/static/src/im... | 200 | 9309 | 300 | 42222 | 11647.25 | 4.00% | 2.8/min | 1.61 | 0.02 | 34891.8 |
| /web/dataset/call_... | 200 | 8802 | 227 | 42230 | 11671.13 | 4.50% | 2.8/min | 0.07 | 0.04 | 1566.3 |
| /web/dataset/call_... | 200 | 10676 | 226 | 42199 | 12815.75 | 8.50% | 2.8/min | 0.07 | 0.04 | 1626.9 |
| /web/dataset/call_... | 200 | 9353 | 225 | 42191 | 11239.58 | 3.50% | 2.8/min | 0.07 | 0.04 | 1551.1 |
| /web/dataset/call_... | 200 | 8758 | 225 | 42447 | 11575.16 | 5.00% | 2.8/min | 0.07 | 0.07 | 1573.8 |
| /web/dataset/call_... | 200 | 9982 | 226 | 42258 | 12174.20 | 6.00% | 2.8/min | 0.07 | 0.04 | 1589.0 |
| /web/dataset/call_... | 200 | 9369 | 227 | 42214 | 12561.61 | 6.50% | 2.8/min | 0.07 | 0.04 | 1596.6 |
| /web/dataset/call_... | 200 | 10387 | 225 | 42116 | 13078.11 | 7.50% | 2.8/min | 0.07 | 0.04 | 1611.8 |
| /web/dataset/call_... | 200 | 11540 | 225 | 42115 | 13749.77 | 9.50% | 2.8/min | 0.07 | 0.04 | 1642.1 |
| /web/dataset/call_... | 200 | 9648 | 226 | 42095 | 11676.64 | 5.00% | 2.7/min | 0.07 | 0.06 | 1573.8 |
| /web/dataset/call_... | 200 | 10874 | 227 | 42117 | 13728.62 | 9.00% | 2.7/min | 0.07 | 0.03 | 1634.5 |
| /web/dataset/call_... | 200 | 12368 | 227 | 42363 | 14310.07 | 11.00% | 2.7/min | 0.07 | 0.03 | 1664.9 |
| /web/dataset/call_... | 200 | 11685 | 224 | 42310 | 13470.95 | 9.50% | 2.7/min | 0.07 | 0.03 | 1642.1 |
| /web/static/img/pl... | 200 | 12881 | 199 | 42109 | 14076.94 | 11.50% | 2.7/min | 0.26 | 0.02 | 5996.6 |
| /partner_autocom... | 200 | 14741 | 399 | 42114 | 15343.68 | 15.50% | 2.7/min | 1.76 | 0.02 | 40792.4 |
| /web/dataset/call_... | 200 | 13992 | 226 | 42116 | 14702.06 | 13.50% | 2.7/min | 0.07 | 0.03 | 1702.8 |
| /web/dataset/call_... | 200 | 14459 | 225 | 42131 | 15536.76 | 15.50% | 2.7/min | 0.07 | 0.03 | 1733.1 |
| /mail/read_followe... | 200 | 15515 | 226 | 42188 | 15757.86 | 18.50% | 2.7/min | 0.08 | 0.02 | 1778.6 |
| /mail/get_suggest... | 200 | 16531 | 226 | 42121 | 16443.67 | 22.50% | 2.7/min | 0.08 | 0.02 | 1839.3 |
| /web/dataset/call_... | 200 | 16038 | 227 | 42114 | 15349.80 | 17.50% | 2.7/min | 0.08 | 0.03 | 1763.5 |
| /web/dataset/call_... | 200 | 15168 | 224 | 42161 | 15612.59 | 18.00% | 2.7/min | 0.08 | 0.03 | 1771.1 |
| /web/static/img/fo... | 198 | 15858 | 198 | 42112 | 15860.00 | 18.69% | 2.6/min | 0.04 | 0.02 | 876.5 |
| /mail/thread/mess... | 197 | 16591 | 226 | 42223 | 14917.87 | 15.23% | 2.6/min | 0.07 | 0.02 | 1729.0 |
| /mail/get_suggest... | 195 | 16223 | 223 | 42199 | 15243.97 | 16.41% | 2.6/min | 0.07 | 0.02 | 1746.9 |
| /mail/read_followe... | 194 | 15050 | 226 | 42121 | 15150.69 | 17.01% | 2.6/min | 0.07 | 0.02 | 1756.0 |
| /web/dataset/call_... | 193 | 17042 | 223 | 42155 | 15837.43 | 20.73% | 2.6/min | 0.08 | 0.02 | 1812.4 |
| /mail/static/src/im... | 193 | 15170 | 199 | 42113 | 15134.27 | 15.03% | 2.6/min | 0.11 | 0.02 | 2595.2 |
| /web/dataset/call_... | 192 | 15188 | 227 | 42179 | 15557.13 | 16.67% | 2.6/min | 0.07 | 0.03 | 1750.8 |
| /web/dataset/call_... | 191 | 16049 | 226 | 42181 | 16288.40 | 21.99% | 2.5/min | 0.08 | 0.03 | 1831.6 |
| /canonical.html-96 | 191 | 224 | 117 | 2499 | 251.63 | 0.00% | 2.5/min | 0.01 | 0.01 | 302.0 |
| /success.txt-98 | 191 | 138 | 57 | 3503 | 322.94 | 0.00% | 2.5/min | 0.01 | 0.01 | 220.0 |
| /success.txt-97 | 191 | 106 | 58 | 1441 | 127.00 | 0.00% | 2.5/min | 0.01 | 0.01 | 220.0 |
| TOTAL | 18726 | 9226 | 57 | 214095 | 14895.06 | 8.08% | 3.9/sec | 299.80 | 2.05 | 78523.1 |

**Figura 11** Resultados en JMeter en AWS con un porcentaje de error de 8.08%

En consiguiente, se presenta (**Tabla III**) el resumen de los resultados obtenidos por cada proveedor de nube y el porcentaje de error del Summary Report tras una simulación con Jmeter al procesar 100 ejecuciones concurrentes. Además, en el Anexo 2 se muestra el comportamiento de diversos parámetros de la CPU, RAM tanto en la BD y el nodo de los diferentes escenarios.

| | **Azure** | **Google** | **AWS** |
|---|---|---|---|
| Tamaño BD | 5GB | 473.63 MiB | 5GB |
| Métricas del Nodo entrada | 1.7 MB | 60 KiB/s | 116Kbs |
| Métricas del Nodo salida | 3.2 MB | 155 KiB/s | 35Kbs. |
| BD CPU | 30.91% | 13.73% | 7.43% |

**Tabla III:** Resumen de resultados obtenidos. Detalles adicionales a los mostrados en la tabla pueden observarse en el anexo 2.

Seguidamente, en la **Figura 12**, se muestra el porcentaje de error para cada una de las empresas empleadas y en la cual se puede observar para Azure un porcentaje de error del 13,83%; para Google, un porcentaje de error del 23,59%; y AWS mostró un 8,08%.

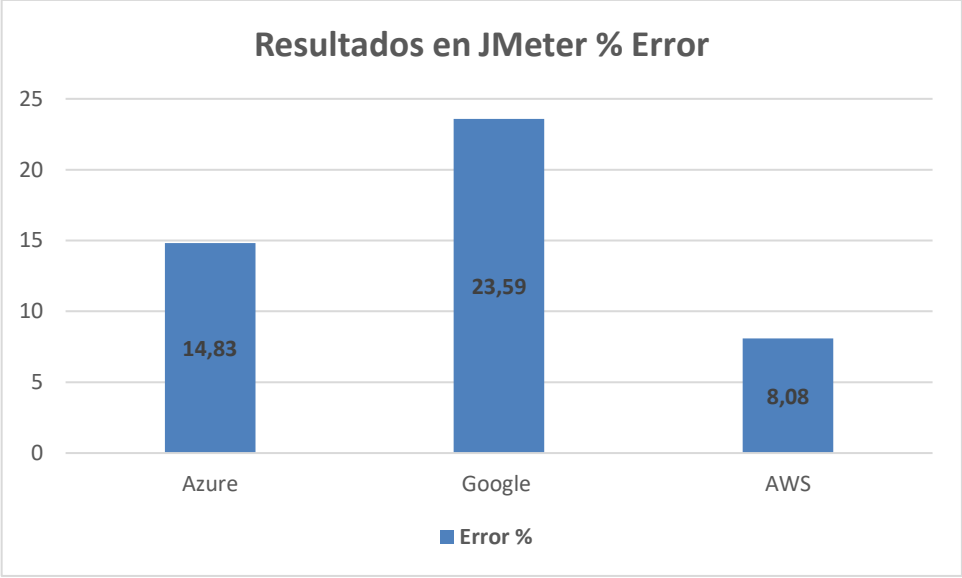


**Figura 12** Resultados en JMeter CPU y % Error

En la **Figura 13**, se presenta las métricas en relación a los nodos de entrada y de salida, expresados en MB/s que representan el tráfico de la red.

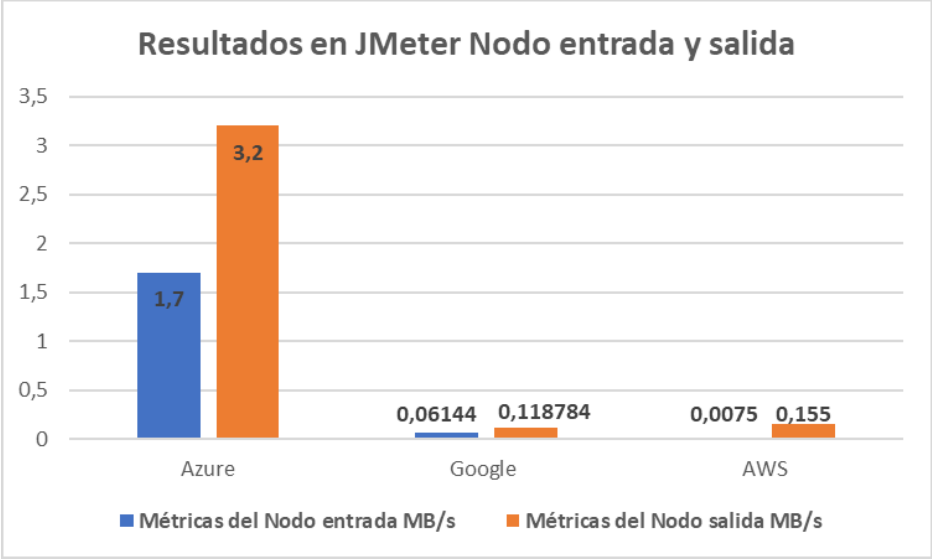


**Figura 13** Resultados en JMeter Nodo entrada y salida

En la **Figura 14** se presentan las métricas de la tasa de transferencia efectiva o el volumen de trabajo o de información neto (Throughput) exhibida por cada una de las empresas empleadas y en la cual los resultados obtenidos muestran a Azure con una tasa del 2,9sec, AWS con una tasa de 3,9sec y Google reflejo una tasa de 21,9sec.

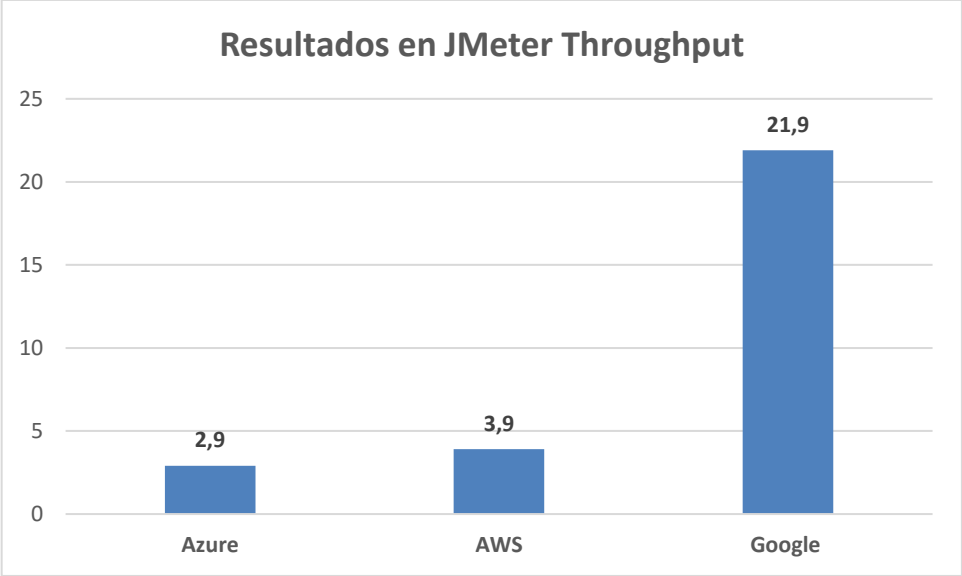


**Figura 14** Resultados en JMeter Throughput

Finalmente, los resultados obtenidos indican que es necesario poner énfasis en las etapas iniciales del diseño de la solución, componentes de arquitectura, alta disponibilidad, correcto escalamiento de la aplicación y un buen control de monitoreo y prevención. Estas premisas tienen soporte en los diversos proveedores de nube.

# IV. Conclusiones

Dentro del aprovisionamiento de la infraestructura con Terraform, en la automatización de los despliegues con pipelines de Jenkins se logra la implementación de un Cluster de Kubernetes y la ejecución de una aplicación Software ERP Odoo dentro de dicho Cluster, logrando así la gestión de múltiples recursos de forma organizada y centralizada.

El trabajo fue desarrollado en 3 proveedores de nube significativos (Amazon, Azure y Google Cloud). En los cuales se concluye que: se logró definir la arquitectura de la plataforma mediante código para generar una infraestructura en los proveedores, consiguiendo una gran versatilidad al ofrecer funcionalidades de computación, almacenamiento y redes a través de Internet.

De igual forma, se consiguió que al implementar el ciclo de integración y despliegue continuo dentro de un repositorio de código Git, se facilite la realización de mejoras y la colaboración entre equipos de trabajo, facilitando el ingreso a los servicios a varios usuarios al mismo tiempo, efectuando modificaciones y teniendo un control de cambio y versionado. Igualmente, el código empleado se encuentra en GitHub, lo que facilita

la realización de mejoras y la colaboración entre equipos de trabajo, facilitando el ingreso a los servicios a varios usuarios al mismo tiempo, efectuar modificaciones y tener un control de cambio y versionado.

Por otro lado, se logró validar y probar la arquitectura GitOps mediante métricas de rendimiento y escalabilidad, permitiendo realizar el comparativo de los proveedores Cloud respecto a tener una infraestructura onpremise. Esto entregó como resultado que Amazon tuvo un menor porcentaje de errores que los demás proveedores con respecto a la optimización de recursos de hardware.

En cuanto a los servicios que ofrecen las compañías Amazon, Azure y Google Cloud, estos son muy similares, pues solo existen pequeñas variaciones en el proceso de aprovisionamiento y uso de sus recursos. La principal diferencia radica en los costos que varían en función de ciertos parámetros establecidos por cada proveedor tales como tipo de servicio, región y plan tarifario según el tipo de cliente, y que al ser comparados en función a porcentajes de costos se señalan los siguientes datos: Google Cloud representa un 18%, Azure Cloud representa un 22%, y por ultimo AWS Cloud representa un 60%, siendo el porcentaje más bajo el que ofrece un servicio más económico y el más elevado el más costoso.

# V. Recomendaciones

Para el sistema es fundamental establecer el estado de los servicios de forma continua y mejorar la habilidad de escalabilidad, ya que debido a variaciones o cambios que puedan verse afectados a futuro, es recomendable el empleo de software adaptables a dichas circunstancias. Para este caso puede emplearse, por ejemplo, el software Grafana que permitirá realizar cuadros de mando, a través de los indicadores que este refleje para mejorar la capacidad de adaptación y respuesta del sistema.

Para el aseguramiento de la continuidad y escalabilidad del desarrollo una vez que los recursos necesarios del proyecto lo requieran, poder migrar a planes con mayores capacidades de servicio que estén acordes a los requerimientos y necesidades de rendimiento y capacidades.

Para mantener un esquema de seguridad adecuado y un correcto manejo de los procesos, datos y permisologías se recomienda un manejo riguroso de los usuarios IAM, en el cual se proporcionen las prioridades especificas a los diferentes actores dentro del sistema, tales como administradores, operadores y desarrolladores, y que les permita acceder solo a los datos y procesos necesarios para asegurar los más estrictos estándares de seguridad dentro de los procesos.

Se recomienda en el caso de querer cambiar la infraestructura con otro proveedor por

razones de costos, mejoras de rendimiento, entre otros factores, el uso de herramientas estandarizadas que simplifiquen los procesos de configuración y adecuación de credenciales de la plataforma Terraform, ya que a través de esta se puede realizar el aprovisionamiento por medio de Kubernetes y este último emplear en distintas nubes disponibles en el mercado para realizar este proceso.

# VI. Trabajo futuro

Realizar una revisión del estado del arte sobre el análisis de algoritmos de Machine Learning, que permita a través de la lectura de las métricas de Kubernetes y los nodos, predecir posibles fallos; y una vez se determinen cuáles pueden ser los mecanismos más adecuados de aprendizaje automático, estos puedan ser aplicados en los correctivos necesarios de forma eficiente, evitando gastos sobre incidentes. Esto contribuirá en la protección ante cualquier eventual fallo en el sistema.

Finalmente, para automatizar la creación de los diferentes objetos de Kubernetes, deployment, secrets, servicios, entre otros, utilizar la herramienta Helm para Kubernetes, que permita de forma automatizada la descarga, instalación y deployment de la aplicación, por medio de plantillas que pueden ser empleadas dentro del desarrollo anteriormente descrito y el cual aportará flexibilidad, evitando realizar tareas manuales, minimizando posibles errores humanos y realizando los despliegues de la forma más segura y sencilla posible.

# VII. Referencias


Alejandre, J. (2018). *Buenas prácticas en la docencia universitaria con apoyo de TIC. Experiencias en 2018* (1st ed.). Universidad de Zaragoza.

AZURE. (2021). *¿Qué es IaaS? Infraestructura como servicio | Microsoft Azure*. https://azure.microsoft.com/es-es/overview/what-is-iaas/#overview

Bakkland, K., Haukebøe, P., & Engelsen, F. (2019). *My Pension* (Vol. 06) [Høgskulen på Vestlandet]. https://hvlopen.brage.unit.no/hvlopen-xmlui/bitstream/handle/11250/2602712/Kalliainen_Haukeboe_Flatekval.pdf?sequence=1&isAllowed=y

Blas, M. J., Leone, H., & Gonnet, S. (2019). Modelado y Verificación de Patrones de Diseño de Arquitectura de Software para Entornos de Computación en la Nube. *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*, *35*, 1–17. https://doi.org/10.17013/RISTI.35.1-17

Bruneo, D. (2014). A stochastic model to investigate data center performance and qos in IaaS cloud computing systems. *IEEE Transactions on Parallel and Distributed Systems*, *25*(3), 560–569. https://doi.org/10.1109/TPDS.2013.67

Buchanan, S., Rangama, J., & Bellavance, N. (2020). CI/CD with Azure Kubernetes Service. *Introducing Azure Kubernetes Service*, 191–219. https://doi.org/10.1007/978-1-4842-5519-3_9

Carrasco, J. (2021). *Unified Management of Applications on Heterogeneous Clouds*. https://riuma.uma.es/xmlui/handle/10630/22490

D'Amore, C. (2021). GitOps and ArgoCD: Continuous deployment and maintenance of a full stack application in a hybrid cloud Kubernetes environment [(Tesis de Maestría) Politécnico Di Torino]. In *Webthesis Biblioteche d'ateneo*. https://webthesis.biblio.polito.it/18142/1/tesi.pdf

Delgado, L., & Pineda, O. (2019). *Modelo para la orquestación de Microservicios con Kubernetes aplicado al servicio de control de versiones Git* [Universidad Distrital Francisco José De Caldas]. https://repository.udistrital.edu.co/handle/11349/22424

Diaz, J., & Matta, M. (2020). *Sistema de recomendación automático de servicios Multi-cloud* [Universidad Icesi]. https://doi.org/10.6084/m9.figshare.13661018.v2

Fu, X., Yang, R., Du, X., & Luo, B. (2018). Timing Channel in IaaS: How to Identify and Investigate. *IEEE Access*, *7*, 1–11. https://doi.org/10.1109/ACCESS.2018.2876146

Gesteira, R. (2020). *Implementación de una arquitectura de microservicios para una red de sensores IoT sobre Arduino* [Comillas Universidad Pontificia]. https://repositorio.comillas.edu/xmlui/handle/11531/43411

Gil, C. (2020). *Despliegue de SQL Server sobre Kubernetes* [(Tesis de pregrado) Universidad Complutense de Madrid]. https://eprints.ucm.es/id/eprint/61970/

GitOps. (2017). *GitOps is Continuous Deployment for cloud native applications*. https://www.gitops.tech/

Havel, R. (2021a). *GitOps - Implementace CI/CD pipeline pro dokumentaci produktu*

*při agilním vývoji ; GitOps - CI/CD pipeline implementation for product documentation in the agile process.* http://ezproxy.unal.edu.co/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edsbas&AN=edsbas.86E84266&lang=es&site=eds-live

Havel, R. (2021b). GitOps – Implementace CI/CD pipeline pro dokumentaci produktu při agilním vývoji. *ČVUT DSpace*. https://dspace.cvut.cz/bitstream/handle/10467/92803/F3-BP-2021-Havel-Richard-GitOps - Implementace CICD pipeline pro dokumentaci produktu pri agilnim vyvoji.pdf?sequence=-1&isAllowed=y

Hernández, A., & Camargo, Á. (2017). Autorregulación del aprendizaje en la educación superior en Iberoamérica: una revisión sistemática. *Revista Latinoamericana de Psicología*, *49*(2), 146–160. https://doi.org/10.1016/j.rlp.2017.01.001

Hoque, S., Brito, M. S. De, Willner, A., Keil, O., & Magedanz, T. (2017). Towards Container Orchestration in Fog Computing Infrastructures. *Proceedings - International Computer Software and Applications Conference*, *2*, 294–299. https://doi.org/10.1109/COMPSAC.2017.248

Johnston, C. (2020). In-Platform CI/CD. In *Advanced Platform Development with Kubernetes* (4th ed., Vol. 1). Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-5611-4_4

Kubernetes. (2021). *Kubernetes*. https://kubernetes.io/

Maderuelo, F. (2021). Metodología GitOps para despliegue de aplicaciones basadas en microservicios. *Archivo Digital UPM*. https://oa.upm.es/67264/

Malina, P. (2019). *Kubernetes Canary Deployment Controller* [University of Hradec Králové]. https://www.fit.vut.cz/study/thesis-file/21857/21857.pdf

Markstedt, O. (2017). *Kubernetes as an approach for solving bioinformatic problems* [Uppsala Universitet]. http://www.teknat.uu.se/student

Martin, P. (2021). Kubernetes. In *Kubernetes* (2nd ed., Vol. 4). Apress. https://doi.org/10.1007/978-1-4842-6494-2

Medel, V., Rana, O., Bañares, J. Á., & Arronategui, U. (2016). Adaptive application scheduling under interference in Kubernetes. *Proceedings - 9th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2016*, 426–427. https://doi.org/10.1145/2996890.3007889

Moilanen, M. (2018). *Deploying an application using Docker and Kubernetes* [La Universidad de Ciencias Aplicadas de Oulu]. https://www.theseus.fi/bitstream/handle/10024/146845/Moilanen_Miika_Opinnaytetyo.pdf?sequence=1&isAllowed=y

Moreira, J. M., Laranjeira, C., Carvalho, J., Ribeiro, F., Lopes, P., & Graça, P. (2017). Integrating a National Network of Institutional Repositories into the National/International Research Management Ecosystem. *Procedia Computer Science*, *106*, 146–152. https://doi.org/10.1016/J.PROCS.2017.03.010

Muñoz, M. (2017). *Diseño e Implementación del Sistema de Gestión de Entornos para la Oficina Asesora de Sistemas de la Universidad Distrital* [Universidad Distrital Francisco José de Caldas]. https://repository.udistrital.edu.co/handle/11349/6711

Murwantara, I. M., & Yugopuspito, P. (2018). Evaluating Energy Consumption in a Different Virtualization within a Cloud System. *Proceedings - 2018 4th International Conference on Science and Technology, ICST 2018*. https://doi.org/10.1109/ICSTC.2018.8528695

Nocentino, A. E., & Weissman, B. (2021). SQL Server on Kubernetes. *SQL Server on Kubernetes*. https://doi.org/10.1007/978-1-4842-7192-6

Ortiz, M. (2019). *Implementación y evaluación de plataformas en la nube para servicios de IoT* [Universitat Politècnica de València]. https://riunet.upv.es/handle/10251/127825

Pérez, O. (2021). *Computación de Altas Prestaciones en entornos contenerizados* [Tesis doctoral, Universidad de la Laguna]. https://riull.ull.es/xmlui/handle/915/22678

Poniszewska, A., & Czechowska, E. (2021). Kubernetes Cluster for Automating Software Production Environment. *Sensors (Basel, Switzerland)*, *21*(5), 1–24. https://doi.org/10.3390/S21051910

Rodríguez de la Cruz, A. (2020). *Herramienta para despliegue y gestión de plataformas en la nube* [Universidad de Sevilla]. https://idus.us.es/handle/11441/108955

Romero, R. (2020). *MASKDADOS.com, la web de ayuda a los juegos de rol desarrollada en Google Cloud Platform usando CICD con Jenkins y aplicando la escalabilidad de Kubernetes* [Universitat Autònoma de Barcelona]. https://ddd.uab.cat/record/232639

Ruelas, D. (2017). Modelo de composición de microservicios para la implementación de una aplicación web de comercio electrónico utilizando Kubernetes [Universidad Nacional del Altiplano]. In *Universidad Nacional del Altiplano*. http://repositorio.unap.edu.pe/handle/UNAP/6886

Sahin, M. (2019). *GitOps basiertes Continuous Delivery für Serverless Anwendungen* [Universität Stuttgart]. https://elib.uni-stuttgart.de/bitstream/11682/10332/1/MA_MuesluemSahin.pdf

Santos, J., Wauters, T., Volckaert, B., & De Turck, F. (2019). Towards network-Aware resource provisioning in kubernetes for fog computing applications. *Proceedings of the 2019 IEEE Conference on Network Softwarization: Unleashing the Power of Network Softwarization, NetSoft 2019*, 351–359. https://doi.org/10.1109/NETSOFT.2019.8806671

Šmíd, R. (2020). *Continuous integration and application deployment with the Kubernetes technology Student*. https://gitlab.fit.cvut.cz/smidrad1/ci-cd-pipeline-with-kubernetes

Steinholt, R. (2015). *A study of Linux Containers and their ability to quickly offer scalability for web services Using Kubernetes and Docker* [(Tesis de maestría) University Of Oslo]. https://core.ac.uk/download/pdf/30815428.pdf

Taurus. (2021). *El año en que todo cambió*. https://www.fundaciontelefonica.com/cultura-digital/publicaciones/sociedad-digital-en-espana-2020-2021/730/

Vayghan, L. A., Saied, M. A., Toeroe, M., & Khendek, F. (2021). A Kubernetes

controller for managing the availability of elastic microservice based stateful applications. *Journal of Systems and Software*, *175*, 110924. https://doi.org/10.1016/J.JSS.2021.110924

Ventura da Silva, R. (2021). *GitOps: uma nova proposta para a infraestrutura* [Florianópolis, SC]. https://repositorio.ufsc.br/handle/123456789/223676

Yamato, Y. (2016). Performance-aware server architecture recommendation and automatic performance verification technology on IaaS cloud. *Service Oriented Computing and Applications 2016 11:2*, *11*(2), 121–135. https://doi.org/10.1007/S11761-016-0201-X

Zakharenkov, R. (2019). *DevOps in E-commerce software development: Demand for Containerization* [(Tesis de Pregrado) Oulu University of Applied Sciences]. http://www.theseus.fi/handle/10024/171099

# Anexos

**Anexo 1.**

Aprovisionamiento de GitOps

**Fase 1. Repositorio de control de versiones**

| **Construcción de los repositorios en GitHub** | | |
|---|---|---|
| Archivos Terraform y YML para aprovisionar K8S, BD y la aplicación Odoo | | |
| **Azure** | leidychaconc / TerraformAzure<br>devops<br>.DS_Store<br>az-resources.tf<br>variables.tf | leidychaconc / odooAzure<br>devops first commit<br>.DS_Store first commit<br>odoo.yml Update odoo.yml |
| **Google** | leidychaconc / terraformGoogle<br>devops<br>.DS_Store<br>main.tf<br>provider.tf<br>variables.tf | leidychaconc / odooGoogle<br>devops Update Jenkinsfile<br>.DS_Store first commit<br>odoo.yml first commit |
| **AWS** | leidychaconc / TerraformAWS<br>devops first commit2<br>.DS_Store first commit2<br>eks-cluster.tf first commit2<br>kubernetes-d... first commit2<br>kubernetes.tf first commit2<br>rds.tf Update rds.tf<br>security-grou... first commit2<br>variables.tf first commit2<br>vpc.tf first commit2 | leidychaconc / odooAWS<br>devops first commit<br>.DS_Store first commit<br>odoo.yml first commit |

**Tabla IV:** Creación de los repositorios con el Código Terraform

**Configuración de WebHooks en GitHub**

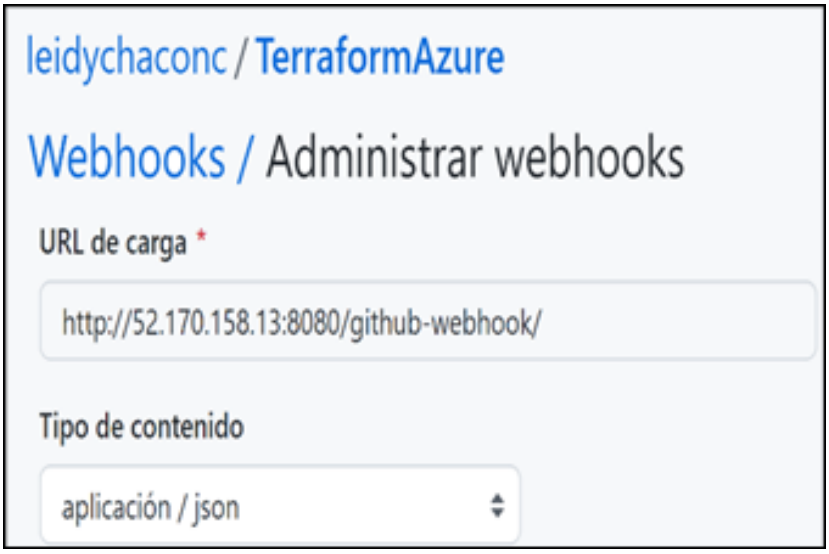


**Figura 15** Configuración WebHooks en GitHub


**Figura 16** Activación de WebHooks en Jenkins

## Fase 2: Preparación de la herramienta para CI/CD

### 1. Instalación del Servidor Jenkins en el Cloud Azure

```
#cloud-config
package_upgrade: true
runcmd:
  - apt install openjdk-8-jdk -y
  - wget -qO - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
  - sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
  - apt-get update && apt-get install jenkins -y
  - service jenkins restart
```

**Figura 17** Código para instalar el Servidor Jenkins

### 2. Script para crear el servidor Jenkins con Linux

```
az vm create \
--resource-group azrgdevopsd01 \
--name jenkins-vm \
--image UbuntuLTS \
--admin-username "azureuser" \
--generate-ssh-keys \
--custom-data cloud-init-jenkins.txt
```

**Figura 18** Implementación de una VM Linux con el servidor Jenkins

### 3. Construcción y ejecución del Pipeline en el servidor Jenkins.

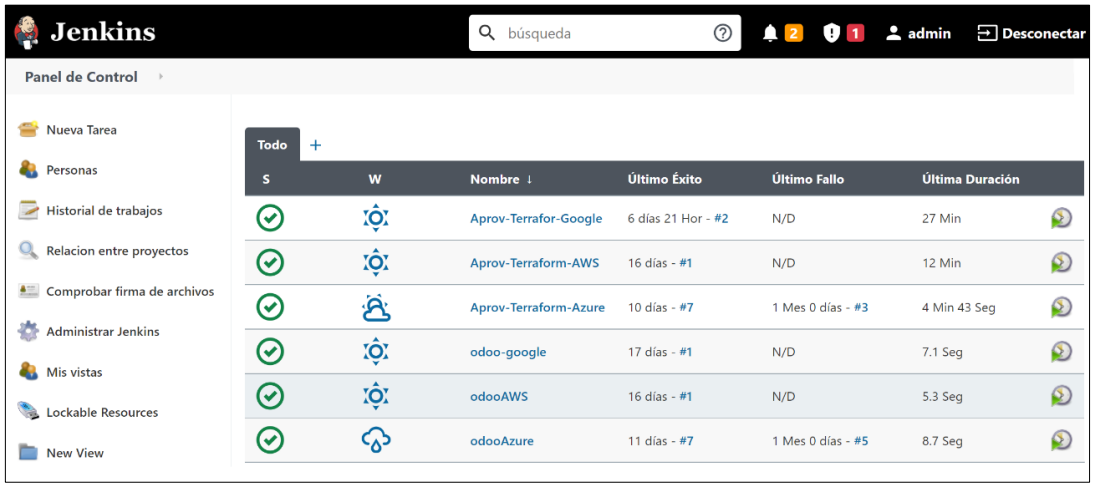


**Figura 19** Pipelines para la orquestación de recursos

## 4. Administración de identidades y accesos (IAM)

En la siguiente (**Tabla V**) se asigna los roles de suscripción necesarios a la entidad de servicio o SP.

| **Creación del usuario IAM para cada Cloud** | |
|---|---|
| Azure | Nombre / Tipo / Rol<br>azure-cli-2021-11-17-15-19-15 / Aplicación / Propietario |
| Google Cloud | Type / Principal / Name / Role<br>593492995065-compute@developer.gserviceaccount.com / Compute Engine default service account / Owner<br>593492995065@cloudservices.gserviceaccount.com / Google APIs Service Agent / Owner |
| AWS | Usuario / ID de clave de acceso / Clave de acceso secreta<br>myterraform / AKIAWALXN43MP7CUOIH5 / ********* Mostrar<br>Se ha creado el usuario myterraform<br>Política AdministratorAccess asociada al usuario myterraform<br>Se ha creado una clave de acceso para el usuario myterraform |

**Tabla V** Creación del usuario IAM para cada proveedor Cloud

## 5. Creación del Service Principal o entidad de servicio

### a) Cloud Azure

```
leidy@Azure:~$ az ad sp create-for-rbac --skip-assignment
{
  "appId": "[redacted]",
  "displayName": "azure-cli-2021-10-06-20-29-04",
  "name": "[redacted]",
  "password": "[redacted]",
  "tenant": "856b1c2d-3a70-4245-af4e-4b087ccc6c21"
}
```

**Figura 20** Service Principal en Azure

### a) Cloud AWS

El usuario IAM permitirá controlar de forma segura el acceso a los recursos de AWS,

la información sensible que proporciona IAM será necesario para autenticar con la cuenta AWS y realizar las diferentes solicitudes, además ayuda a que diferentes sistemas puedan comunicarse, en este caso se requiere una conexión segura con el servidor de Jenkins y cloud AWS.

```
User name,Password,Access key ID,Secret access key,Console login link
myterraform,,AKIAWALXN43MP7CUOIH5,v/3qxKXIy56HmCEfsn9aGhS69+5cNY0MKzxtg
```

**Figura 21** Service Principal en AWS

### b) Cloud Google

A diferencia de otros proveedores el SP de Google se llama Cuenta de servicio. Entonces, se debe crear una cuenta de servicio habilitando la API de Compute Engine para generar las credenciales y que el usuario IAM pueda crear y administrar los recursos.

```
azureuser@jenkins-vm:/var/lib/jenkins$ cat secretsgoogle.json
{
  "type": "service_account",
  "project_id": "weighty-utility-332922",
  "private_key_id": "[redacted]",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANBgkq
e91vO\nV6+EHCEMr9eA8hdt5TYw4wRIIG1x0unZvnKSm8zre1jJco3jzdNtfLk/
yPOLQq+uLOcSdFzhWNaxKBbwDRScVNi\ne75ycHto8uHr1xaKn1JRV+o/wfYW7F
sXXyAl1V9eQyic5sVczKWD380mSfG2isBLgeRGl6VFRUU8LCEqg5b01ag\n2wF(
iyXhyMWebaxrqyslC\nbCocCsgt8h5NREstToimDYxrxUfd4QqXA+cvYqakM6bE
5tS+NSH8cGCK6YTLTxaCqaeFcrZnIdj6vUEFNlA4sSu\nPMHqD2cIN/iVeZY9YM
K+F\nmLUM0Yr+h72kLgLt+XR5tjwUtELXSkpeHIBI0uMjI/MgnP8hkFzL8I687J
8Qi5p662M+/S1cjyPv51sPq0ad5cY\n9sNpTaKlBE9w9/TmXA+LKM8O5WCA2gV+
UX/qIKabt/Invhp/G3oRRc4QSVbK6mhmppfJrEL5PSRW1rNMghvFbnt\nKIitSA
dp4SkAb9fCry1nv\nalYupmR5/CSVLyOnoLiKEGO7QfRMvxVSbrYOW6+1VO67k+
=\n-----END PRIVATE KEY-----\n",
  "client_email": "[redacted]-compute@developer.gserviceaccou
  "client_id": "[redacted]",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oa
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/
}
```

**Figura 22** Service Principal en Google Cloud

## 6. Variables de entorno para el aprovisionamiento de la infraestructura

En el servidor Jenkins se configura las variables con la información sensible que proporciona el usuario IAM. El código clonado en GitHub solicitará el valor de cada variable y establecerá una conexión con el Cloud respectivo. Este proceso verificará el acceso y permitirá CI/CD en el proveedor solicitado.

### a) Azure

El script a continuación solicitará las credenciales del proveedor Azure para establecer una conexión segura y proceder con el aprovisionamiento de los recursos.

```
provider "azurerm" {
  features {}

  subscription_id = var.subscriptionId
  client_id       = var.appId
  client_secret   = var.appId_password
  tenant_id       = var.tenantId
}
```

**Figura 23** Script de las variables del usuario IAM de Azure

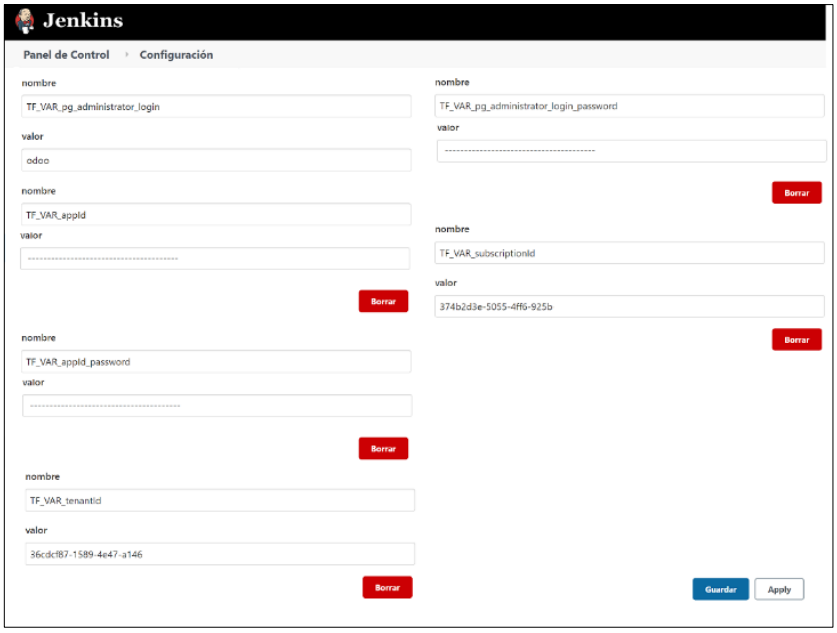


**Figura 24** Configuración de las credenciales de Azure en el servidor Jenkins

### b) Google

El siguiente script solicita las credenciales del usuario IAM de Google, para establecer la conexión.

```
provider "google" {
    project = "weighty-utility-332922"
    region = "us-west1"
    credentials = "/var/lib/jenkins/secretsGoogle.json"
}
```

**Figura 25** Definición de las variables del usuario IAM de Google

```
jenkins@jenkins-vm:~$ cat secretsGoogle.json
{
  "type": "service_account",
  "project_id": "weighty-utility-332922",
  "private_key_id": "bf4fe4208d0429b/474d6c5",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEvQIBADANB
ot0E0+u2DxEJgMS/d12I\nuZVdolPJ2XsGpjLwyggTLyrjfJ27ZXUY6PDIlm
u3KQXrmU4/ze+1IzEoeT2fnQWhp5wresRkoJ2m03ncSAbD7/umQPlJ/DF5bE
0BbB0I/1wwNyeVC8J1Iy2V2uTtMD08n/Njay\nQy7fMBWikDutRr6P0RF7Nf
zDz/pdSOauo\n2tWf8lPkH89PBqnOO/k6o+KuQ85ri8C8qQ2hi/Zayy96zEO
UwUH3wxXlkDhNyHlm/LrjlUAKfDfxQKBgQDpGn3JCI2D+P0En4sA\nFKPocV
v80VKx3NtLUC7mpP8ovVa5INxiV\nwJWH3nzpyH4DVvpVwsuCnj2zvwKBgQD
p0\nt3OUt/10rEFy65DDerKf7dy51UCXPhkJlRrjQC1NNdh7SpHJMWlxGczg
Ti3NBpelBqNc4ijlOxqFzGcdllDZDkgpFvJryuCBzVq\nhyB4snpj8VRWH3m
WiEYVNDTx3SfBuCpCE\n+5KPLEPVbvMCc+h2hBvvkedJvZYpI9eQJmQBZnax
DbkEQz3UKhRPLcuOLzuC/HSCpq8rNmQuLYRj/WSQHiqcgwiJyYTlt8uhd7S\
-END PRIVATE KEY-----\n",
  "client_email": "593492995065-compute@developer.gserviceaccount.com",
  "client_id": "10570237360 ",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadat
}
```

**Figura 26** Configuración de las credenciales de Google en el servidor Jenkins

### c) AWS

El script solicita las credenciales del usuario IAM de AWS, para establecer la conexión.

```
variable "aws_access_key" {
  description = "AWS IAM access_key"
  sensitive   = true
}

variable "aws_secret_key" {
  description = "AWS IAM secret_key"
  sensitive   = true
}


variable "aws_pg_administrator_login" {
  description = "Postgres admin login"
  sensitive   = true
}

variable "aws_pg_administrator_password" {
  description = "Postgres admin login password"
  sensitive   = true
}
```

**Figura 27** Definición de las variables del usuario IAM de AWS

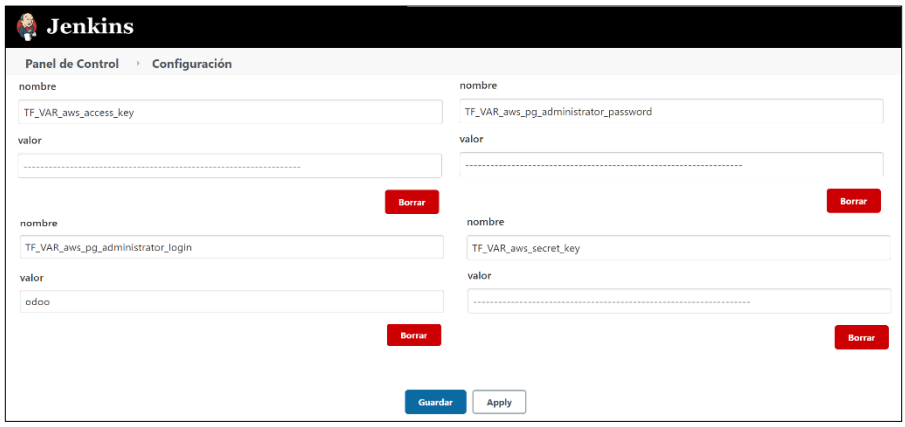


**Figura 28** Configuración de las credenciales de AWS en el servidor Jenkins

## 7. Infraestructura como código mediante Terraform

### a) Código para implementar AKS

Se plantea el siguiente código (.tf) para implementar el AKS y se otorga el siguiente nombre "azkseuappd01". Además, otros parámetros como la solicitud de un nodo con las características que el Operador requiera según la necesidad. El nombre del Nodo por default se le otorga como "aks-default-26787434". En la **Figura 30** se visualiza el resultado del código Terraform de la construcción del AKS en Azure.

```
provider "azurerm" {
  features {}

  subscription_id = var.subscriptionId
  client_id       = var.appId
  client_secret   = var.appId_password
  tenant_id       = var.tenantId
}

resource "azurerm_resource_group" "default" {
  name     = "azrgeuappd01"
  location = "East US"

  tags = {
    environment = "Development"
  }
}

resource "azurerm_kubernetes_cluster" "default" {
  name                = "azkseuappd01"
  location            = azurerm_resource_group.default.location
  resource_group_name = azurerm_resource_group.default.name
  dns_prefix          = "dns-k8s"

  default_node_pool {
    name            = "default"
    node_count      = 1
    vm_size         = "Standard_D2_v2
    os_disk_size_gb = 30
  }

  identity {
    type = "SystemAssigned"
  }

  role_based_access_control {
    enabled = true
  }

  tags = {
    environment = "Development"
  }
}
```

**Figura 29** Código para aprovisionar AKS en Azure

```
leidy@Azure :~$ kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-default-26787434-vmss000000     Ready    agent   30d   v1.20.9

leidy@Azure :~$ kubectl get all --all-namespaces
NAMESPACE     NAME                                          READY   STATUS    RESTARTS
default       pod/odoo-deployment-75dfd8cdd8-mgd75          1/1     Running   0
kube-system   pod/azure-ip-masq-agent-q9dr5                 1/1     Running   0
kube-system   pod/coredns-58567c6d46-hmrm8                  1/1     Running   0
kube-system   pod/coredns-58567c6d46-l2sdj                  1/1     Running   0
kube-system   pod/coredns-autoscaler-54d55c8b75-hth4q       1/1     Running   0
kube-system   pod/kube-proxy-2bhm8                          1/1     Running   0
kube-system   pod/metrics-server-569f6547dd-fsjkp           1/1     Running   1
kube-system   pod/omsagent-lhsnz                            2/2     Running   0
kube-system   pod/omsagent-rs-74dd7c4579-zkxnr              1/1     Running   0
kube-system   pod/tunnelfront-59965668df-qj97q              1/1     Running   0

NAMESPACE     NAME                                     TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)
default       service/kubernetes                       ClusterIP      10.0.0.1       <none>         443/TCP
default       service/serviceodoo                      LoadBalancer   10.0.187.116   20.84.18.34    80:32347/TCP
kube-system   service/healthmodel-replicaset-service    ClusterIP      10.0.99.147    <none>         25227/TCP
kube-system   service/kube-dns                         ClusterIP      10.0.0.10      <none>         53/UDP,53/TCP
kube-system   service/metrics-server                   ClusterIP      10.0.248.138   <none>         443/TCP

NAMESPACE     NAME                                  DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE
kube-system   daemonset.apps/azure-ip-masq-agent    1         1         1       1            1
kube-system   daemonset.apps/kube-proxy             1         1         1       1            1
kube-system   daemonset.apps/omsagent               1         1         1       1            1
kube-system   daemonset.apps/omsagent-win           0         0         0       0            0
NAMESPACE     NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
default       deployment.apps/odoo-deployment       1/1     1            1           29d
kube-system   deployment.apps/coredns               2/2     2            2           30d
kube-system   deployment.apps/coredns-autoscaler    1/1     1            1           30d
kube-system   deployment.apps/metrics-server        1/1     1            1           30d
kube-system   deployment.apps/omsagent-rs           1/1     1            1           29d
kube-system   deployment.apps/tunnelfront           1/1     1            1           30d

NAMESPACE     NAME                                            DESIRED   CURRENT   READY   AGE
default       replicaset.apps/odoo-deployment-75dfd8cdd8      1         1         1       29d
kube-system   replicaset.apps/coredns-58567c6d46              2         2         2       30d
kube-system   replicaset.apps/coredns-autoscaler-54d55c8b75   1         1         1       30d
kube-system   replicaset.apps/metrics-server-569f6547dd       1         1         1       30d
kube-system   replicaset.apps/omsagent-rs-74dd7c4579          1         1         1       29d
kube-system   replicaset.apps/tunnelfront-59965668df          1         1         1       30d
```

**Figura 30** Servicios implementados en el AKS

## b) Código para proveer una BD en Azure

```
resource "azurerm_postgresql_server" "default" {
  name                = "azdbpgodoo"
  location            = azurerm_resource_group.default.location
  resource_group_name = azurerm_resource_group.default.name

  sku_name = "B_Gen5_2"

  storage_mb                   = 5120
  backup_retention_days        = 7
  geo_redundant_backup_enabled = false
  auto_grow_enabled            = true

  administrator_login          = var.pg_administrator_login
  administrator_login_password = var.pg_administrator_login_password
  version                      = "10"
  ssl_enforcement_enabled      = false
#  allow_azure_services_access  = true
}
```

**Figura 31** Código para aprovisionar una BD en Azure

Se obtiene el siguiente resultado del código Terraform para la construcción de una BD PostgreSQL en Azure.

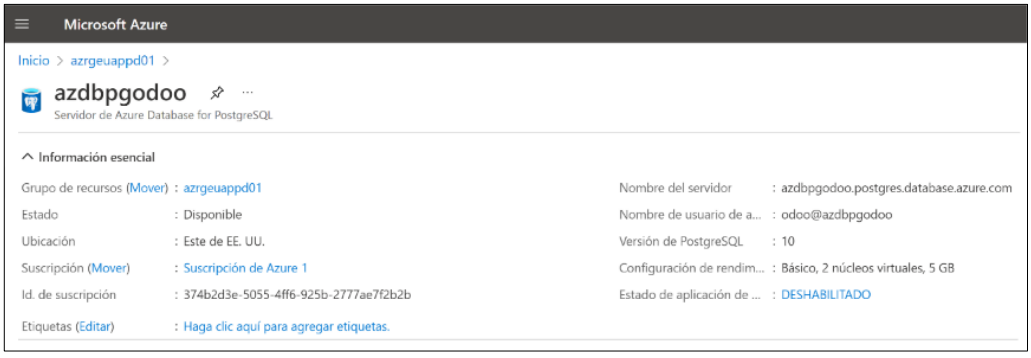


**Figura 32** Propiedades de la BD en Azure

### c) Código para implementar GKE

El código a continuación permite el aprovisionamiento del servicio de Kubernetes con el siguiente nombre "gke-test-1", además otros parámetros como la solicitud de un nodo llamado "default-node-pool" y dependiendo el tipo de cuenta que se contrata en el Cloud, para este proyecto se utiliza el plan de prueba y proporciona automáticamente un máximo de 110 Pods por nodo.

```
module "gke" {
  source                     = "terraform-google-modules/kubernetes-engine/google"
  project_id                 = "weighty-utility-332922"
  name                       = "gke-test-1"
  region                     = "us-central1"
  zones                      = ["us-central1-a", "us-central1-b", "us-central1-f"]
  network                    = "default"
  subnetwork                 = "default"
  ip_range_pods              = ""
  ip_range_services          = ""
  http_load_balancing        = false
  horizontal_pod_autoscaling = true
  network_policy             = false

  node_pools = [
    {
      name               = "default-node-pool"
      machine_type       = "n1-standard-2"
      min_count          = 1
      max_count          = 2  # quota errors if the number is too high
      disk_size_gb       = 10
      disk_type          = "pd-standard"
      image_type         = "COS"
      auto_repair        = true
      auto_upgrade       = true
      preemptible        = false
      initial_node_count = 1
    },
  ]
```

**Figura 33** Código para aprovisionar Kubernetes en Google

Se obtiene el siguiente resultado del código Terraform para la construcción del Kubernetes en Google Cloud.

```
jenkins@jenkins-vm:~$ kubectl get nodes
NAME                                          STATUS   ROLES    AGE   VERSION
gke-gke-test-1-default-node-pool-39dc45bf-2w5k   Ready    <none>   16d   v1.21.5-gke.1802
gke-gke-test-1-default-node-pool-b4abe26e-t318   Ready    <none>   16d   v1.21.5-gke.1802
gke-gke-test-1-default-node-pool-eac1b035-0jqs   Ready    <none>   16d   v1.21.5-gke.1802

jenkins@jenkins-vm:~$ kubectl get all --all-namespaces
NAMESPACE     NAME                                                      READY   STATUS    RESTARTS
default       pod/odoo-deployment-75dfd8cdd8-8hwjp                      1/1     Running   0
kube-system   pod/event-exporter-gke-5479fd58c8-mg749                   2/2     Running   0
kube-system   pod/fluentbit-gke-5hhrj                                   2/2     Running   0
kube-system   pod/fluentbit-gke-swdt7                                   2/2     Running   0
kube-system   pod/fluentbit-gke-znfjc                                   2/2     Running   0
kube-system   pod/gke-metadata-server-ctlhj                             1/1     Running   0
kube-system   pod/gke-metadata-server-f6c8b                             1/1     Running   0
kube-system   pod/gke-metadata-server-spmps                             1/1     Running   0
kube-system   pod/gke-metrics-agent-kgzdf                               1/1     Running   0
kube-system   pod/gke-metrics-agent-n87zs                               1/1     Running   0
kube-system   pod/gke-metrics-agent-t4rkg                               1/1     Running   0
kube-system   pod/konnectivity-agent-577df7599f-5cqzc                   1/1     Running   0
kube-system   pod/konnectivity-agent-577df7599f-jBvcx                   1/1     Running   0
kube-system   pod/konnectivity-agent-577df7599f-vntps                   1/1     Running   0
kube-system   pod/konnectivity-agent-autoscaler-5c49cb58bb-64qnd        1/1     Running   0
kube-system   pod/kube-dns-697dc8fc8b-cf45d                             4/4     Running   0
kube-system   pod/kube-dns-697dc8fc8b-m4212                             4/4     Running   0
kube-system   pod/kube-dns-autoscaler-844c9d9448-654pw                  1/1     Running   0
kube-system   pod/kube-proxy-gke-gke-test-1-default-node-pool-39dc45bf-2w5k   1/1     Running   0
kube-system   pod/kube-proxy-gke-gke-test-1-default-node-pool-b4abe26e-t318   1/1     Running   0
kube-system   pod/kube-proxy-gke-gke-test-1-default-node-pool-eac1b035-0jqs   1/1     Running   0
kube-system   pod/metrics-server-v0.4.4-857776bc9c-l6xqf                2/2     Running   0
kube-system   pod/netd-4cp61                                            1/1     Running   0
kube-system   pod/netd-f7m24                                            1/1     Running   0
kube-system   pod/netd-rs8hz                                            1/1     Running   0
kube-system   pod/pdcsi-node-4cwxt                                      2/2     Running   0
kube-system   pod/pdcsi-node-5lxxs                                      2/2     Running   0
kube-system   pod/pdcsi-node-rg5pj                                      2/2     Running   0

NAMESPACE     NAME                  TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)
default       service/kubernetes    ClusterIP      10.12.0.1     <none>         443/TCP
default       service/serviceodoo   LoadBalancer   10.12.6.82    34.70.22.134   80:32634/TCP
kube-system   service/kube-dns      ClusterIP      10.12.0.10    <none>         53/UDP,53/TCP
kube-system   service/metrics-server   ClusterIP   10.12.1.150   <none>         443/TCP

NAMESPACE     NAME                                      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
kube-system   daemonset.apps/fluentbit-gke              3         3         3       3            3           kubernetes.io/os=linux
kube-system   daemonset.apps/gke-metadata-server        3         3         3       3            3           beta.kubernetes.io/os=linux,iam.gke.io/gke-metadata-server-enabled=true
kube-system   daemonset.apps/gke-metrics-agent          3         3         3       3            3           kubernetes.io/os=linux
kube-system   daemonset.apps/gke-metrics-agent-windows  0         0         0       0            0           kubernetes.io/os=windows
kube-system   daemonset.apps/kube-proxy                 0         0         0       0            0           kubernetes.io/os=linux,node.kubernetes.io/kube-proxy-ds-ready=true
kube-system   daemonset.apps/metadata-proxy-v0.1        0         0         0       0            0           cloud.google.com/metadata-proxy-ready=true,kubernetes.io/os=linux
kube-system   daemonset.apps/netd                       3         3         3       3            3           cloud.google.com/gke-netd-ready=true,kubernetes.io/os=linux
kube-system   daemonset.apps/nvidia-gpu-device-plugin   0         0         0       0            0           <none>
kube-system   daemonset.apps/pdcsi-node                 3         3         3       3            3           kubernetes.io/os=linux
kube-system   daemonset.apps/pdcsi-node-windows         0         0         0       0            0           kubernetes.io/os=windows

NAMESPACE     NAME                                      READY   UP-TO-DATE   AVAILABLE
default       deployment.apps/odoo-deployment           1/1     1            1
kube-system   deployment.apps/event-exporter-gke        1/1     1            1
kube-system   deployment.apps/konnectivity-agent        3/3     3            3
kube-system   deployment.apps/konnectivity-agent-autoscaler   1/1     1            1
kube-system   deployment.apps/kube-dns                  2/2     2            2
kube-system   deployment.apps/kube-dns-autoscaler       1/1     1            1
kube-system   deployment.apps/metrics-server-v0.4.4     1/1     1            1

NAMESPACE     NAME                                                DESIRED   CURRENT   READY
default       replicaset.apps/odoo-deployment-75dfd8cdd8          1         1         1
kube-system   replicaset.apps/event-exporter-gke-5479fd58c8       1         1         1
kube-system   replicaset.apps/konnectivity-agent-577df7599f       3         3         3
kube-system   replicaset.apps/konnectivity-agent-autoscaler-5c49cb58bb   1         1         1
kube-system   replicaset.apps/kube-dns-697dc8fc8b                 2         2         2
```

**Figura 34** Resultados de los servicios implementados en el GKE

### d) Código para proveer una BD en Google

```
locals {
  rds_name = "odoo-${random_string.suffix.result}"
}
resource "random_string" "suffix" {
  length  = 4
  special = false
  lower = true
  upper = false
  number = true
}
# Create Database
resource "google_sql_database_instance" "gcp_database" {
    name = local.rds_name
    region = "${var.db_region}"
    database_version = "${var.database_version}"
    deletion_protection = false

    settings {
        tier = "${var.tier}"
        disk_size = "${var.disk_size}"
        replication_type = "${var.replication_type}"
        activation_policy = "${var.activation_policy}"

    }
}

# Create User

resource "google_sql_user" "admin" {
    count = 1
    name = "${var.user_name}"
    password = "${var.user_password}"
    instance = "${google_sql_database_instance.gcp_database.name}"
}
```

**Figura 35** Código para aprovisionar la BD en GKE

Se obtiene el resultado del código Terraform para la construcción de una BD PostgreSQL en Google cloud.

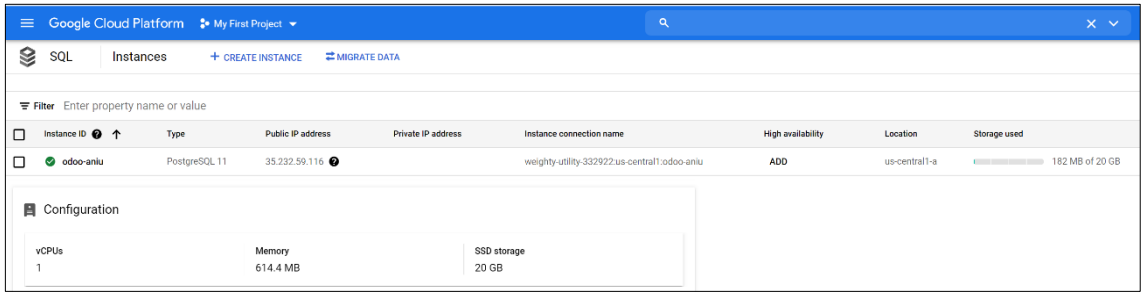


**Figura 36** Propiedades de la BD en GKE

### e) Código para implementar EKS

Para aprovisionar el Cluster EKS, se tiene varios archivos Terraform (.tf) que servirá para el funcionamiento del Kubernetes. En la siguiente figura el Cluster es llamado "education-eks-odoo" y se especifica la red.

```
locals {
  cluster_name = "education-eks-odoo"
}


module "vpc" {
  source  = "terraform-aws-modules/vpc/aws"
  version = "2.66.0"

  name                 = "education-vpc"
  cidr                 = "10.0.0.0/16"
  azs                  = data.aws_availability_zones.available.names
  private_subnets      = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  public_subnets       = ["10.0.4.0/24", "10.0.5.0/24", "10.0.6.0/24"]
  enable_nat_gateway   = true
  single_nat_gateway   = true
  enable_dns_hostnames = true

  tags = {
    "kubernetes.io/cluster/${local.cluster_name}" = "shared"
  }
```

**Figura 37** Código para crear una nube privada virtual (VPC) en AWS

Cuando lanza una instancia en una VPC, debe especificar un grupo de seguridad que se crea para esa VPC. Se crea un grupo de seguridad para las instancias EC2 para controlar el tráfico entrante y saliente. Los clústeres pueden contener más de un tipo de instancia de Amazon EC2.

```
resource "aws_security_group" "worker_group_mgmt_one" {
  name_prefix = "worker_group_mgmt_one"
  vpc_id      = module.vpc.vpc_id

  ingress {
    from_port = 22
    to_port   = 22
    protocol  = "tcp"

    cidr_blocks = [
      "10.0.0.0/8",
    ]
  }
}

resource "aws_security_group" "worker_group_mgmt_two" {
  name_prefix = "worker_group_mgmt_two"
  vpc_id      = module.vpc.vpc_id

  ingress {
    from_port = 22
    to_port   = 22
    protocol  = "tcp"

    cidr_blocks = [
      "192.168.0.0/16",
    ]
  }
}

resource "aws_security_group" "all_worker_mgmt" {
  name_prefix = "all_worker_management"
  vpc_id      = module.vpc.vpc_id

  ingress {
    from_port = 22
    to_port   = 22
    protocol  = "tcp"

    cidr_blocks = [
      "10.0.0.0/8",
      "172.16.0.0/12",
      "192.168.0.0/16",
    ]
  }
}
```

**Figura 38** Código para crear el grupo de seguridad para el clúster EKS

Proporciona todos los recursos (grupos de AutoScaling, entre otros.) necesarios para configurar un clúster de EKS mediante el módulo AWS EKS.

```
module "eks" {
  source          = "terraform-aws-modules/eks/aws"
  cluster_name    = local.cluster_name
  cluster_version = "1.20"
  subnets         = module.vpc.private_subnets

  tags = {
    Environment = "training"
    GithubRepo  = "terraform-aws-eks"
    GithubOrg   = "terraform-aws-modules"
  }

  vpc_id = module.vpc.vpc_id

  workers_group_defaults = {
    root_volume_type = "gp2"
  }

  worker_groups = [
    {
      name                          = "worker-group-1"
      instance_type                 = "t2.small"
      additional_userdata           = "echo foo bar"
      asg_desired_capacity          = 2
      additional_security_group_ids = [aws_security_group.worker_group_mgmt_one.id]
    },
    {
      name                          = "worker-group-2"
      instance_type                 = "t2.medium"
      additional_userdata           = "echo foo bar"
      additional_security_group_ids = [aws_security_group.worker_group_mgmt_two.id]
      asg_desired_capacity          = 1
    },
  ]
}

data "aws_eks_cluster" "cluster" {
  name = module.eks.cluster_id
}

data "aws_eks_cluster_auth" "cluster" {
  name = module.eks.cluster_id
}
```

**Figura 39** Código con parámetros necesario para el clúster EKS de AWS

Se obtiene como resultado del código Terraform para la construcción del Kubernetes en AWS Cloud.

```
azureuser@jenkins-vm:~$ kubectl get all --all-namespaces
NAMESPACE     NAME                                      READY   STATUS
default       pod/odoo-deployment-75dfd8cdd8-k6bj4      1/1     Running
kube-system   pod/aws-node-52d6f                        1/1     Running
kube-system   pod/aws-node-b4wdm                        1/1     Running
kube-system   pod/aws-node-cjq2b                        1/1     Running
kube-system   pod/coredns-5c778788f4-gpmvd              1/1     Running
kube-system   pod/coredns-5c778788f4-qvqrk              1/1     Running
kube-system   pod/kube-proxy-67zn6                      1/1     Running
kube-system   pod/kube-proxy-h7z4m                      1/1     Running
kube-system   pod/kube-proxy-w8hxb                      1/1     Running

NAMESPACE     NAME                  TYPE           CLUSTER-IP       EXTERNAL-IP                      PORT(S)
default       service/kubernetes    ClusterIP      172.20.0.1       <none>                           443/TCP
default       service/serviceodoo   LoadBalancer   172.20.255.141   aceba90d4cd894fe184a20b40376986  80:32401/TCP
kube-system   service/kube-dns      ClusterIP      172.20.0.10      <none>                           53/UDP,53/TCP

NAMESPACE     NAME                        DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE
kube-system   daemonset.apps/aws-node     3         3         3       3            3
kube-system   daemonset.apps/kube-proxy   3         3         3       3            3

NAMESPACE     NAME                              READY   UP-TO-DATE   AVAILABLE
default       deployment.apps/odoo-deployment   1/1     1            1
kube-system   deployment.apps/coredns           2/2     2            2

NAMESPACE     NAME                                         DESIRED   CURRENT   READY
default       replicaset.apps/odoo-deployment-75dfd8cdd8   1         1         1
kube-system   replicaset.apps/coredns-5c778788f4           2         2         2
```

**Figura 40** Servicios implementados en el EKS

### f) Código para proveer una BD en AWS

El subsiguiente código permitirá crear el servicio de base de datos relacional (RDS) y al PostgreSQL se nombra como “education”, trabaja en el puerto 5432.

```
resource "aws_security_group" "rds" {
  name   = "education_rds"
  vpc_id = module.vpc.vpc_id

  ingress {
    from_port   = 5432
    to_port     = 5432
    protocol    = "tcp"
    cidr_blocks = ["x.x.x.x/x"]
  }

  egress {
    from_port   = 5432
    to_port     = 5432
    protocol    = "tcp"
    cidr_blocks = ["x.x.x.x/x"]
  }

  tags = {
    Name = "education_rds"
  }
}
resource "aws_db_parameter_group" "education" {
  name   = "education"
  family = "postgres13"

  parameter {
    name  = "log_connections"
    value = "1"
  }
}
resource "aws_db_subnet_group" "education" {
  name       = "education2"
  subnet_ids = module.vpc.private_subnets

  tags = {
    Name = "Education"
  }
}
resource "aws_db_instance" "education" {
  identifier             = "education"
  instance_class         = "db.t3.micro"
  allocated_storage      = 5
  engine                 = "postgres"
  engine_version         = "13.1"
  username               = var.aws_pg_administrator_login
  password               = var.aws_pg_administrator_password
  db_subnet_group_name   = aws_db_subnet_group.education.name
  vpc_security_group_ids = [aws_security_group.rds.id]
  parameter_group_name   = aws_db_parameter_group.education.name
  publicly_accessible    = false
  skip_final_snapshot    = true
}
```

**Figura 41** Código para implementar la BD en AWS

Se obtiene el resultado del código Terraform para la construcción de una BD PostgreSQL en AWS cloud.

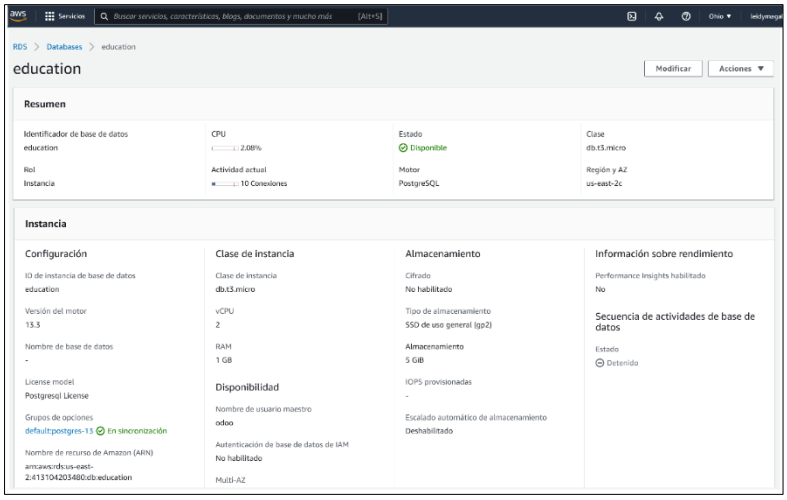


**Figura 42** Propiedades de la BD en EKS

**Fase 4: Despliegue continuo de la Aplicación Odoo mediante Jenkins**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: odoo-deployment
spec:
  selector:
    matchLabels:
      app: odoo
  replicas: 1
  template:
    metadata:
      labels:
        app: odoo
    spec:
      containers:
        - name: odoo-server
          image: odoo
          ports:
          - name: oddo-http-port
            containerPort: 8069
          env:
            #- name: HOST
            #  value: 127.0.0.1
            - name: "HOST"
              valueFrom:
                secretKeyRef:
                  name: odoo-secrets
                  key: postgres_host
            - name: "PORT"
              value: "5432"
            - name: "USER"
              valueFrom:
                secretKeyRef:
                  name: odoo-secrets
                  key: postgres_user
            - name: "PASSWORD"
              valueFrom:
                secretKeyRef:
                  name: odoo-secrets
                  key: postgres_password
apiVersion: v1
kind: Service
metadata:
  name: serviceodoo
spec:
  selector:
    app: odoo
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8069
  type: LoadBalancer
```

**Figura 43** Código YML para el despliegue continuo de la aplicación Odoo

## 8. Código para establecer una conexión entre el servidor Jenkins y el Cluster Kubernetes

En la estructura de canalización se presenta las etapas para establecer una conexión segura en base a la información otorgada por el Service Principal de cada proveedor de nube como es Azure, Google Cloud y AWS.

master odooAzure / devops / Jenkinsfile

```
pipeline {
    agent any
    environment {
        username_sp = credentials('username_sp')
        password_sp = credentials('password_sp')
        tenant = credentials('tenant')
        suscripcion = credentials('suscripcion')
    }
    stages {
        stage('Conexion azure') {
            steps {
                sh 'az login --service-principal --username $username_sp --tenant $tenant  --password $password_sp'
            }
        }
        stage('Configuracion de la suscripcion') {
            steps {
                sh 'az account set -s $suscripcion'
            }
        }
        stage('Conexion al AKS') {
            steps {
                sh 'az aks get-credentials -n azkseuappd01 -g azrgeuappd01'
            }
        }
        stage('Ejecucion del despliegue odoo') {
            steps {
                sh 'kubectl apply -f odoo.yml'
            }
        }
    }
}
```

**Figura 44** Pipeline para establecer la conexión con AKS Cluster

master odooGoogle / devops / Jenkinsfile

```
pipeline {
    agent any
    stages {
        stage('Set Auth service account') {
            steps {
                sh '/var/lib/jenkins/google-cloud-sdk/bin/gcloud auth activate-service-account 593492995065-compute@developer.gserviceaccount.com
                    --key-file=/var/lib/jenkins/secretsGoogle.json --project=weighty-utility-332922'
            }
        }
        stage('Conexion GKS') {
            steps {
                sh '/var/lib/jenkins/google-cloud-sdk/bin/gcloud container clusters get-credentials gke-test-1 --region us-central1'
            }
        }
        stage('Ejecucion del despliegue odoo en Google cloud') {
            steps {
                sh 'kubectl apply -f odoo.yml'
            }
        }
    }
```

**Figura 45** Pipeline para establecer la conexión con GKE Cluster

master odooAWS / devops / Jenkinsfile

```
pipeline {
    agent any
    stages {
        stage('Conexion AWS') {
            steps {
                sh 'aws eks --region us-east-2 update-kubeconfig --name education-eks-odoo'
            }
        }
        stage('Ejecucion del despliegue odoo en AWS') {
            steps {
                sh 'kubectl apply -f odoo.yml'
            }
        }
    }
}
```

**Figura 46** Pipeline para establecer la conexión con EKS Cluster

**Anexo 2.**

Pruebas de carga y métricas

El rendimiento en cada proveedor varía porque el plan gratuito que proporciona a cada cloud es desproporcional tanto en CPU, RAM, almacenamiento, cantidad de nodos o Pods, entre otros. A continuación, se realiza pruebas de carga con JMeter y se observa el rendimiento que ejerce la BD y en el Cluster Kubernetes.

**Azure Cloud**

**Pruebas de carga con JMeter**

Con JMeter se realizó un test con 100 usuarios.

Summary Report

Name: Summary Report

Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| /web/dataset/call_... | 100 | 42248 | 20441 | 67699 | 11042.17 | 11.00% | 16.9/min | 0.45 | 0.16 | 1639.8 |
| /web/dataset/call_... | 100 | 40225 | 20405 | 60073 | 10660.11 | 11.00% | 16.1/min | 0.42 | 0.13 | 1593.1 |
| /web/image/res.u... | 100 | 42765 | 21035 | 77908 | 11981.73 | 9.00% | 16.3/min | 1.65 | 0.08 | 6234.6 |
| /sales/sale_quotat... | 100 | 42892 | 21051 | 70971 | 9253.89 | 3.00% | 15.9/min | 0.39 | 0.14 | 1513.6 |
| /sale/static/src/im... | 100 | 4606 | 256 | 21077 | 6245.45 | 9.00% | 18.1/min | 9.79 | 0.10 | 33211.8 |
| /web/dataset/call_... | 100 | 38908 | 21023 | 70634 | 11898.54 | 20.00% | 15.6/min | 0.45 | 0.16 | 1757.4 |
| /web/dataset/call_... | 100 | 43136 | 17818 | 70994 | 11166.71 | 9.00% | 15.4/min | 0.40 | 0.17 | 1614.7 |
| /web/dataset/call_... | 100 | 44081 | 21019 | 89181 | 12968.50 | 13.00% | 14.4/min | 0.39 | 0.16 | 1643.3 |
| /web/dataset/call_... | 100 | 45339 | 21025 | 74628 | 12633.37 | 16.00% | 14.7/min | 0.37 | 0.29 | 1542.3 |
| /web/dataset/call_... | 100 | 45618 | 21025 | 95218 | 14002.39 | 14.00% | 14.9/min | 0.40 | 0.14 | 1632.9 |
| /web/dataset/call_... | 100 | 45117 | 18336 | 70962 | 13411.69 | 16.00% | 15.4/min | 0.39 | 0.16 | 1565.6 |
| /web/dataset/call_... | 100 | 45446 | 21018 | 65076 | 12047.60 | 17.00% | 16.0/min | 0.39 | 0.27 | 1485.3 |
| /web/dataset/call_... | 100 | 44577 | 21024 | 78348 | 12350.48 | 17.00% | 14.5/min | 0.36 | 0.15 | 1508.6 |
| /web/dataset/call_... | 100 | 42092 | 13840 | 71435 | 12814.46 | 14.00% | 14.6/min | 0.38 | 0.14 | 1609.6 |
| /web/dataset/call_... | 100 | 43489 | 21020 | 72543 | 12242.18 | 15.00% | 14.7/min | 0.39 | 0.13 | 1645.9 |
| /web/dataset/call_... | 100 | 42235 | 19776 | 82152 | 10908.24 | 8.00% | 14.6/min | 0.38 | 0.14 | 1578.4 |
| /web/dataset/call_... | 100 | 42527 | 20050 | 70210 | 11028.23 | 10.00% | 16.0/min | 0.41 | 0.16 | 1581.1 |
| /web/static/img/pl... | 100 | 2307 | 170 | 21044 | 4269.39 | 4.00% | 16.8/min | 1.71 | 0.09 | 6240.4 |
| /partner_autocom... | 100 | 2665 | 335 | 21085 | 5098.58 | 5.00% | 16.0/min | 11.81 | 0.08 | 45475.7 |
| /web/dataset/call_... | 100 | 41452 | 14273 | 70103 | 11107.02 | 7.00% | 15.0/min | 0.38 | 0.12 | 1565.5 |
| /mail/get_suggest... | 100 | 41426 | 14435 | 66864 | 10728.57 | 7.00% | 16.2/min | 0.41 | 0.12 | 1565.5 |
| /web/dataset/call_... | 100 | 38913 | 18626 | 77156 | 12837.00 | 10.00% | 17.1/min | 0.45 | 0.13 | 1627.7 |
| /web/dataset/call_... | 100 | 35064 | 13529 | 65932 | 12303.44 | 19.00% | 17.6/min | 0.49 | 0.15 | 1721.1 |
| /mail/read_followe... | 100 | 32425 | 11223 | 65224 | 10654.36 | 13.00% | 17.2/min | 0.47 | 0.11 | 1666.6 |
| /web/dataset/call_... | 100 | 28104 | 13991 | 52027 | 8259.24 | 35.00% | 18.6/min | 0.59 | 0.13 | 1952.0 |
| /web/static/img/fo... | 100 | 9842 | 172 | 21071 | 8801.41 | 32.00% | 20.5/min | 0.39 | 0.09 | 1156.2 |
| /mail/get_suggest... | 100 | 25669 | 7904 | 50254 | 7172.10 | 26.00% | 19.4/min | 0.58 | 0.11 | 1835.2 |
| /mail/read_followe... | 100 | 25873 | 15121 | 41143 | 6107.93 | 25.00% | 20.3/min | 0.60 | 0.12 | 1822.2 |
| /mail/thread/mess... | 100 | 25943 | 4447 | 50098 | 8161.86 | 24.00% | 22.0/min | 0.65 | 0.13 | 1809.3 |
| /web/dataset/call_... | 100 | 24371 | 2717 | 50122 | 8896.97 | 32.00% | 24.6/min | 0.77 | 0.15 | 1913.0 |
| /web/dataset/call_... | 100 | 22429 | 935 | 50215 | 10278.54 | 21.00% | 26.2/min | 0.75 | 0.18 | 1770.4 |
| /web/dataset/call_... | 100 | 19715 | 588 | 49424 | 11328.86 | 21.00% | 28.6/min | 0.82 | 0.22 | 1770.4 |
| /mail/static/src/im... | 100 | 6743 | 171 | 21070 | 7804.43 | 19.00% | 34.0/min | 1.42 | 0.15 | 2573.1 |
| TOTAL | 7600 | 34253 | 58 | 152788 | 24058.63 | 14.83% | 2.9/sec | 262.70 | 1.26 | 94219.3 |

**Figura 47** Resultados en JMeter para Azure con un porcentaje de error de 14,83%

**Métricas de la CPU y memoria de la Base de Datos**

Las métricas son analizadas en el periodo de tiempo que se ejecutó la prueba de carga con JMeter, en ese tiempo específico se realiza un análisis de los datos obtenidos y se visualiza el valor Máximo (Max) de todos los valores de las muestras recopiladas en ese tiempo determinado. Este análisis es para los tres proveedores.

En la figura a continuación se visualiza el trabajo que realiza la BD cuando se envía una prueba de carga en JMeter con 100 usuario.

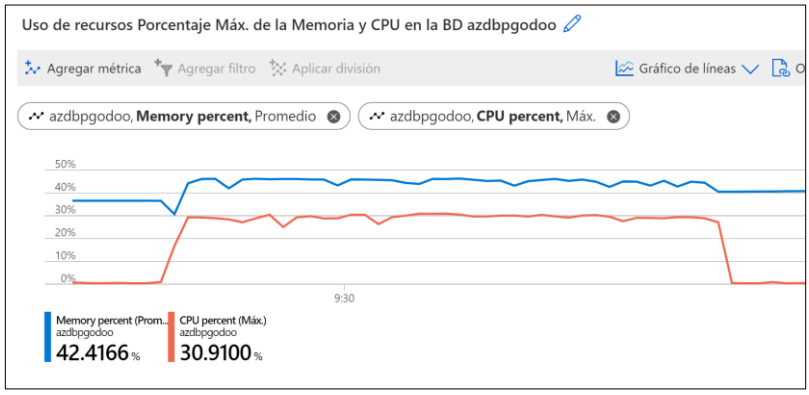


**Figura 48** Porcentaje Máximo del uso del CPU y Memoria al ejecutar JMeter

En la figura 41 se detalla el almacenamiento utilizado en la BD PostgreSQL. El almacenamiento para una BD básico es de 5Gb.

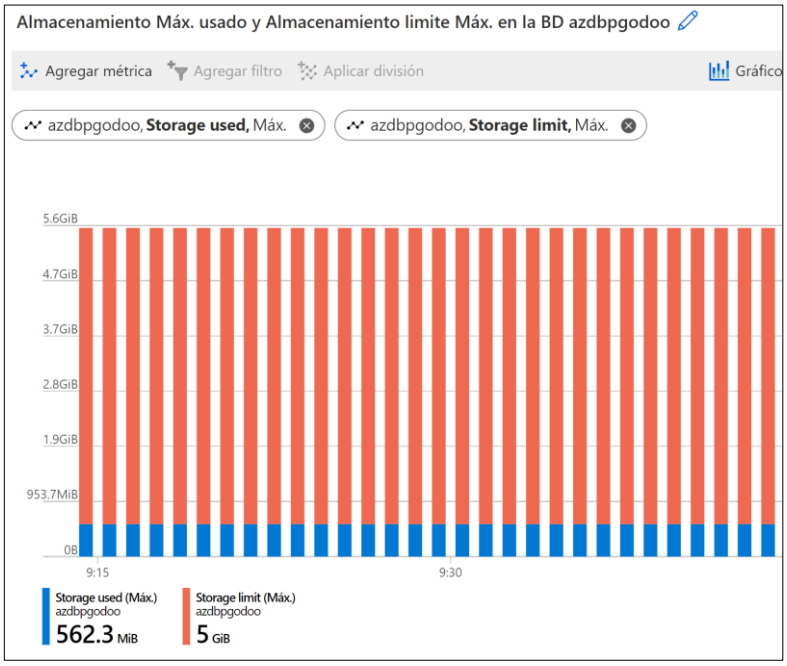


**Figura 49** Almacenamiento utilizado en la BD en Azure

En la figura 42 se visualiza la sumatoria de los bytes recibidos en todas las interfaces de red de la BD.

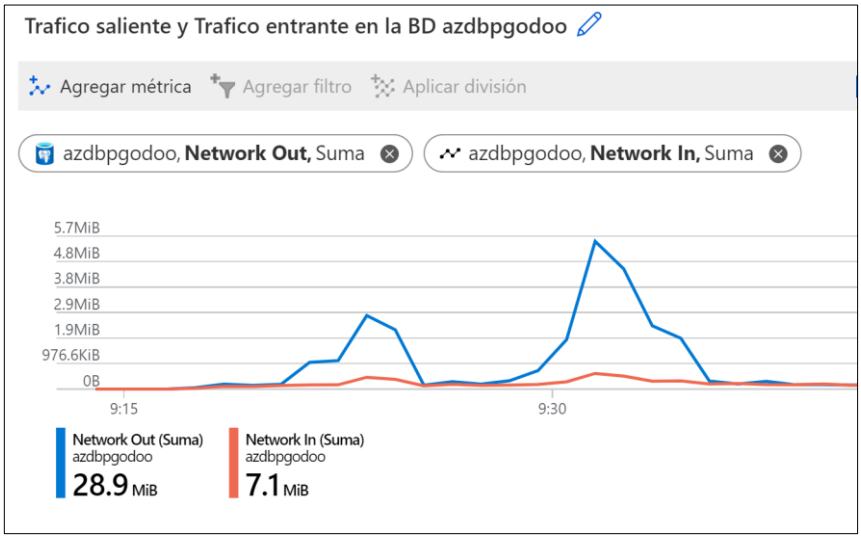


**Figura 50** Sumatoria de Bytes recibidos en las interfaces de red de la BD Azure

### Métricas del Nodo "azkseuappd01"

En el servicio Kubernetes de Azure se visualiza los bytes recibos de las interfaces de red, su valor máximo en la entrada de Red es de 1.7MB y salida de Res tiene un valor máximo de 3.2MB.

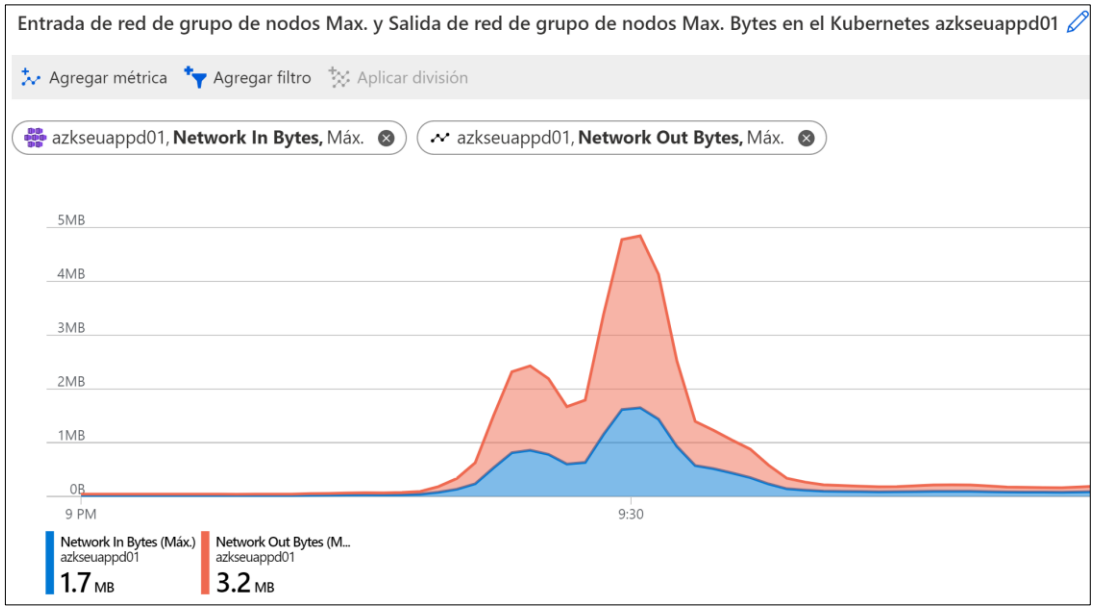


**Figura 51** Métricas de la Red en el Nodo (aks-default-26787434-vmss000000)

En el Cluster de Kubernetes, el máximo porcentaje de uso de la CPU es de 16%.

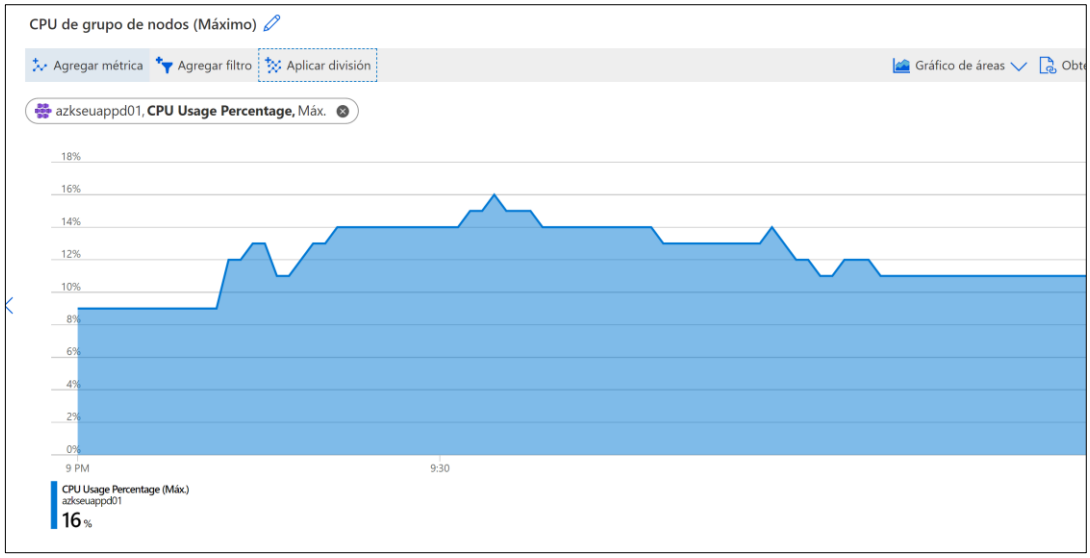


**Figura 52** Porcentaje Max. Del CPU en el Nodo

## Google Cloud

## Pruebas de carga con JMeter

a) Con JMeter se realizó un test con 100 usuarios.

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/... | Sent KB/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| /web/dataset/call_kw/crm.tag/read-45 | 100 | 2379 | 203 | 21055 | 4131.45 | 38.00% | 43.6/min | 0.82 | 0.46 | 1150.8 |
| /web/dataset/call_kw/res.users/has_group-46 | 100 | 1837 | 214 | 21041 | 3566.86 | 40.00% | 42.8/min | 0.79 | 0.37 | 1130.1 |
| /web/image/res.users/2/avatar_128-47 | 100 | 2089 | 211 | 21045 | 4176.11 | 43.00% | 41.9/min | 2.76 | 0.23 | 4045.9 |
| /sales/sale_quotation_onboarding_panel-48 | 100 | 1489 | 208 | 21049 | 2659.34 | 35.00% | 41.7/min | 0.79 | 0.36 | 1159.1 |
| /sale/static/src/img/sale_quotation_onboarding_bg.jp... | 100 | 2157 | 282 | 21027 | 3464.06 | 1.00% | 40.8/min | 23.83 | 0.24 | 35885.8 |
| /web/dataset/call_kw/mail.followers/load_views-50 | 100 | 1796 | 203 | 21060 | 3617.65 | 42.00% | 35.9/min | 0.65 | 0.44 | 1110.0 |
| /web/dataset/call_kw/mail.activity/load_views-51 | 100 | 1637 | 209 | 21025 | 2911.54 | 40.00% | 35.9/min | 0.65 | 0.45 | 1107.3 |
| /web/dataset/call_kw/mail.message/load_views-52 | 100 | 1874 | 211 | 21046 | 3508.10 | 39.00% | 35.9/min | 0.65 | 0.45 | 1117.7 |
| /web/dataset/call_kw/sale.order/read-53 | 100 | 1783 | 217 | 21026 | 3168.31 | 33.00% | 35.9/min | 0.69 | 0.78 | 1179.8 |
| /web/dataset/call_kw/crm.tag/read-56 | 100 | 2195 | 240 | 15318 | 3175.05 | 34.00% | 35.3/min | 0.66 | 0.38 | 1146.1 |
| /web/dataset/call_kw/mail.message/read-58 | 100 | 2307 | 212 | 21061 | 4314.56 | 33.00% | 35.1/min | 0.70 | 0.38 | 1226.5 |
| /web/dataset/call_kw/mail.activity/read-57 | 100 | 2317 | 205 | 21043 | 4408.05 | 37.00% | 35.1/min | 0.69 | 0.38 | 1208.4 |
| /web/dataset/call_kw/sale.order.line/read-54 | 100 | 1972 | 213 | 21059 | 4010.48 | 28.00% | 35.2/min | 0.72 | 0.63 | 1254.9 |
| /web/dataset/call_kw/mail.followers/read-55 | 100 | 2026 | 215 | 21050 | 4039.36 | 28.00% | 34.7/min | 0.72 | 0.38 | 1278.2 |
| /web/dataset/call_kw/res.partner/name_get-61 | 100 | 2108 | 214 | 21046 | 3763.94 | 31.00% | 31.9/min | 0.64 | 0.32 | 1223.8 |
| /web/dataset/call_kw/res.partner/name_get-59 | 100 | 2159 | 206 | 21060 | 4178.62 | 29.00% | 31.5/min | 0.65 | 0.34 | 1267.9 |
| /web/dataset/call_kw/res.partner/name_get-60 | 100 | 2793 | 205 | 21070 | 5208.51 | 31.00% | 31.3/min | 0.65 | 0.31 | 1270.5 |
| /web/static/img/placeholder.png-63 | 100 | 1903 | 187 | 21054 | 4152.49 | 3.00% | 31.3/min | 3.20 | 0.17 | 6276.4 |
| /partner_autocomplete/static/lib/jsvat.js-62 | 100 | 2029 | 372 | 21042 | 2972.05 | 1.00% | 30.8/min | 23.74 | 0.16 | 47272.8 |
| /web/dataset/call_kw/ir.attachment/search_read-72 | 100 | 1773 | 206 | 21055 | 3614.90 | 29.00% | 30.8/min | 0.62 | 0.32 | 1244.5 |
| /web/static/img/form_sheetbg.png-73 | 100 | 2113 | 186 | 21050 | 3472.59 | 1.00% | 30.6/min | 0.20 | 0.19 | 409.1 |
| /mail/get_suggested_recipients-71 | 100 | 2113 | 207 | 21055 | 4237.07 | 34.00% | 30.7/min | 0.61 | 0.23 | 1216.1 |
| /mail/read_followers-66 | 100 | 2034 | 206 | 21057 | 4717.41 | 33.00% | 30.7/min | 0.64 | 0.23 | 1273.2 |
| /web/dataset/call_kw/sale.order/read-69 | 100 | 2426 | 208 | 21047 | 4363.73 | 20.00% | 29.6/min | 0.64 | 0.25 | 1337.7 |
| /web/dataset/call_kw/ir.attachment/search_read-67 | 100 | 2058 | 207 | 21032 | 3416.55 | 25.00% | 29.9/min | 0.61 | 0.31 | 1262.6 |
| /web/dataset/call_kw/sale.order/read-65 | 100 | 2151 | 204 | 21032 | 4298.11 | 22.00% | 29.5/min | 0.64 | 0.25 | 1340.3 |
| /mail/get_suggested_recipients-68 | 100 | 2291 | 202 | 21056 | 4362.57 | 26.00% | 29.5/min | 0.61 | 0.23 | 1275.6 |
| /mail/read_followers-70 | 100 | 2184 | 204 | 21045 | 3782.37 | 19.00% | 29.1/min | 0.63 | 0.22 | 1324.7 |
| /mail/thread/messages-64 | 100 | 2466 | 203 | 21041 | 4623.51 | 23.00% | 28.7/min | 0.63 | 0.22 | 1353.3 |
| /web/dataset/call_kw/mail.activity/activity_format-74 | 100 | 2426 | 205 | 21044 | 4039.60 | 31.00% | 28.7/min | 0.57 | 0.25 | 1223.8 |
| /web/dataset/call_kw/mail.activity/activity_format-75 | 100 | 2315 | 207 | 21047 | 4067.85 | 25.00% | 28.7/min | 0.59 | 0.25 | 1262.6 |
| /mail/static/src/img/odoobot.png-77 | 100 | 2273 | 192 | 21051 | 4011.49 | 2.00% | 28.5/min | 1.17 | 0.15 | 2526.5 |
| /web/dataset/call_kw/mail.message/mark_all_as_rea... | 100 | 2222 | 209 | 21045 | 3537.61 | 23.00% | 28.3/min | 0.59 | 0.27 | 1283.3 |
| TOTAL | 7300 | 3338 | 58 | 70027 | 6945.54 | 23.59% | 21.9/sec | 2197.56 | 10.62 | 102842.0 |

**Figura 53** Resultados en JMeter en Google con un porcentaje de error de 23,59%

## Métricas de la CPU y memoria de la Base de Datos

La CPU de la BD tuvo un porcentaje de 13.73% para procesar la información en el tiempo de ejecución del JMeter.

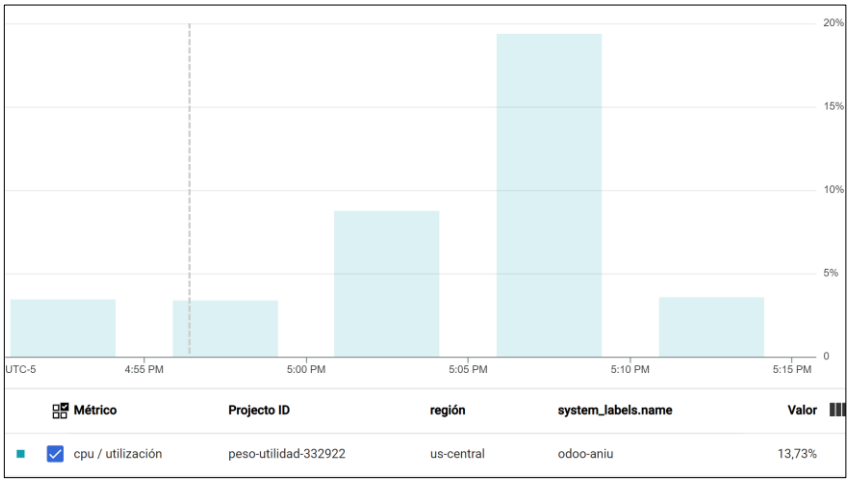


**Figura 54 Porcentaje** Máximo del uso del CPU al ejecutar JMeter en Google

En la BD de Google la memoria utilizada fue de 473.63 MiB, cuando se envió a ejecutar JMeter.

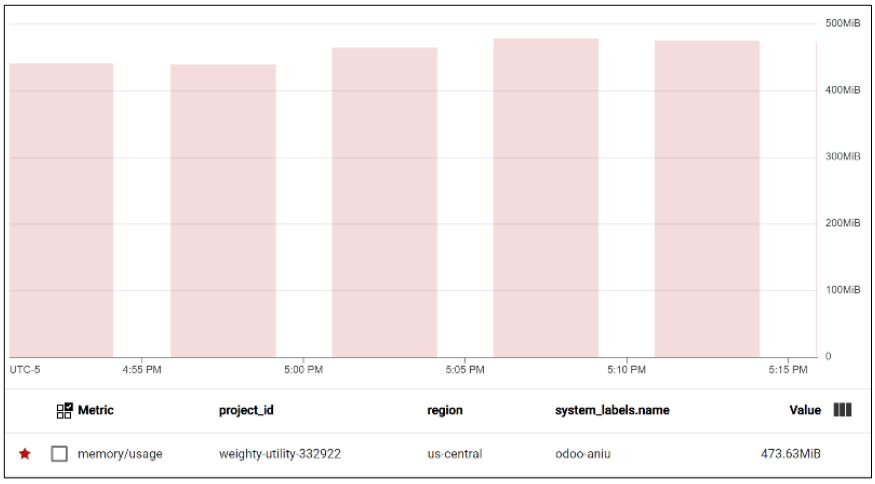


**Figura 55** Memoria usada de la BD al ejecutar JMeter en Google

La cantidad de almacenamiento para una cuenta en modo prueba es de 19.52Gib. En la siguiente figura se detalla el almacenamiento que se proporciona para la BD SQL.

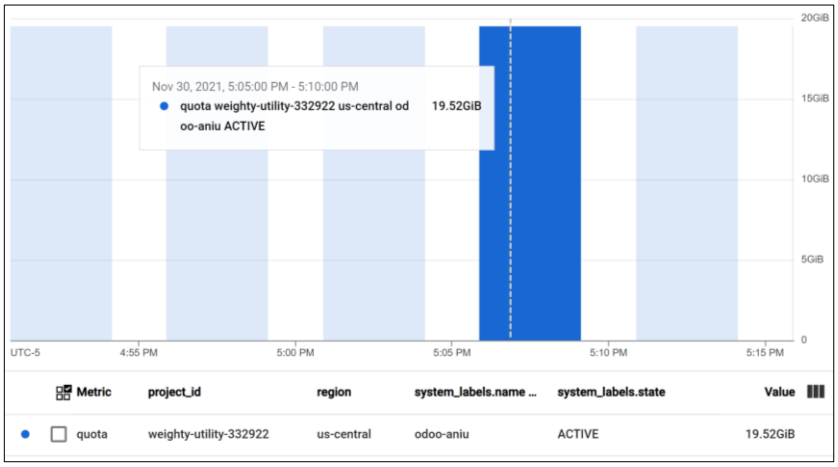


**Figura 56** Almacenamiento total de la BD en Google

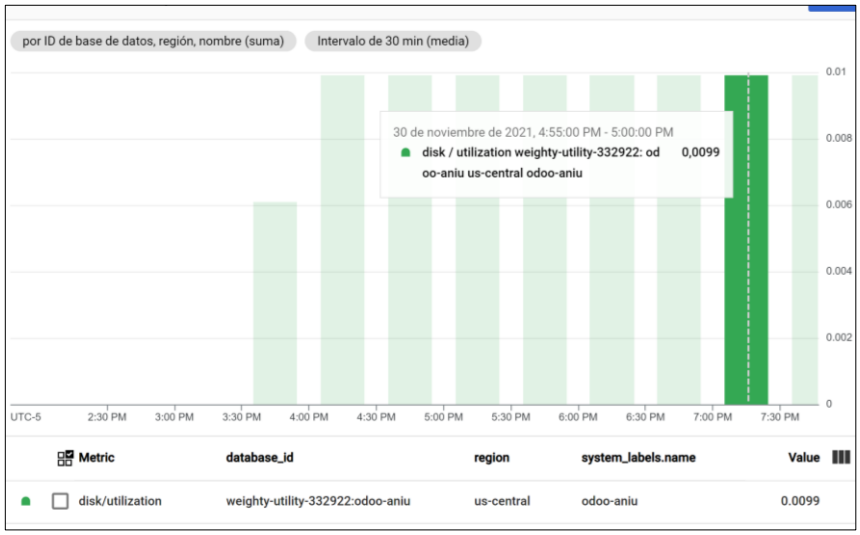


**Figura 57** Almacenamiento utilizado en la BD de Google

En las siguientes figuras se muestra los bytes recibos y enviados por la red, estos valores representan la cantidad de datos que entran y salen de las conexiones DB.


**Figura 58** El máximo Byte recibido a treves de la Red para la BD en Google

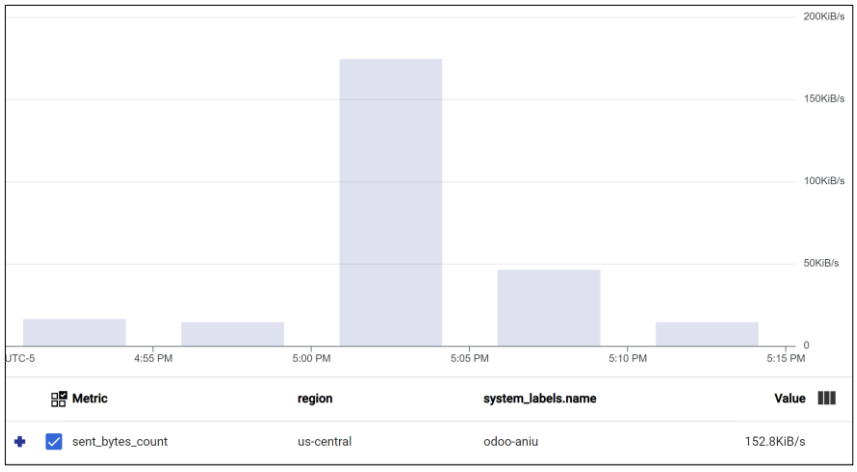


**Figura 59** El máximo Byte enviado a treves de la Red para la BD en Google

## Métricas de los Nodos en Google


**Figura 60** Kubernetes Nodos bytes recibido en Google


**Figura 61** Kubernetes Nodos bytes transmitidos en Google

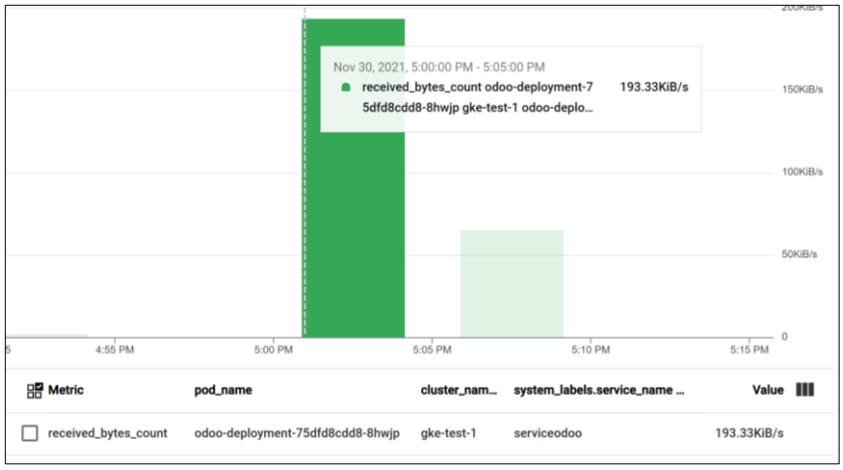


**Figura 62** Bytes recibido en el servicio del Odoo en Google

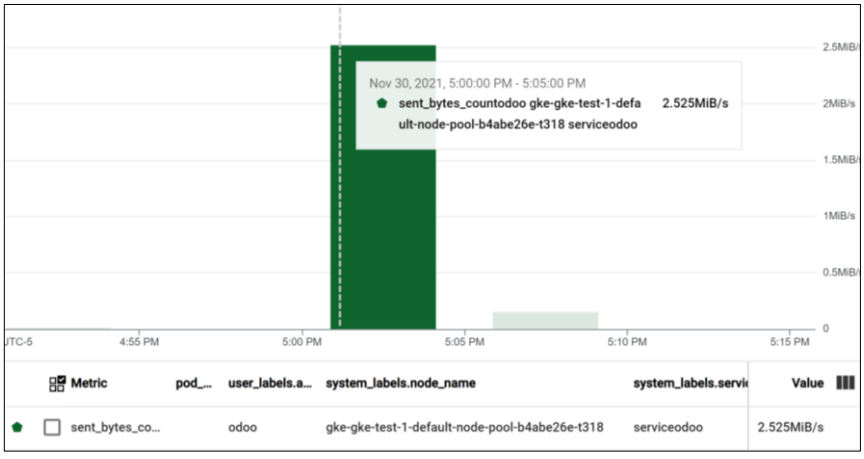


**Figura 63** Bytes transmitidos en el servicio del Odoo en Google

## AWS Cloud

### Pruebas de carga con JMeter

Con JMeter se realizó un test con 100 usuarios.

| Label | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| /web/image/res.... | 200 | 6310 | 200 | 42540 | 11002.57 | 9.50% | 2.9/min | 0.50 | 0.02 | 6127.4 |
| /sales/sale_quotat... | 200 | 9728 | 226 | 42123 | 12920.28 | 7.50% | 2.9/min | 0.07 | 0.03 | 1611.8 |
| /sale/static/src/im... | 200 | 9309 | 300 | 42222 | 11647.25 | 4.00% | 2.8/min | 1.61 | 0.02 | 34891.8 |
| /web/dataset/call_... | 200 | 8802 | 227 | 42230 | 11671.13 | 4.50% | 2.8/min | 0.07 | 0.04 | 1566.3 |
| /web/dataset/call_... | 200 | 10676 | 226 | 42199 | 12815.75 | 8.50% | 2.8/min | 0.07 | 0.04 | 1626.9 |
| /web/dataset/call_... | 200 | 9353 | 225 | 42191 | 11239.58 | 3.50% | 2.8/min | 0.07 | 0.04 | 1551.1 |
| /web/dataset/call_... | 200 | 8758 | 225 | 42447 | 11575.16 | 5.00% | 2.8/min | 0.07 | 0.07 | 1573.8 |
| /web/dataset/call_... | 200 | 9982 | 226 | 42258 | 12174.20 | 6.00% | 2.8/min | 0.07 | 0.04 | 1589.0 |
| /web/dataset/call_... | 200 | 9369 | 227 | 42214 | 12561.61 | 6.50% | 2.8/min | 0.07 | 0.04 | 1596.6 |
| /web/dataset/call_... | 200 | 10387 | 225 | 42116 | 13078.11 | 7.50% | 2.8/min | 0.07 | 0.04 | 1611.8 |
| /web/dataset/call_... | 200 | 11540 | 225 | 42115 | 13749.77 | 9.50% | 2.8/min | 0.07 | 0.04 | 1642.1 |
| /web/dataset/call_... | 200 | 9648 | 226 | 42095 | 11676.64 | 5.00% | 2.7/min | 0.07 | 0.06 | 1573.8 |
| /web/dataset/call_... | 200 | 10874 | 227 | 42117 | 13728.62 | 9.00% | 2.7/min | 0.07 | 0.03 | 1634.5 |
| /web/dataset/call_... | 200 | 12368 | 227 | 42363 | 14310.07 | 11.00% | 2.7/min | 0.07 | 0.03 | 1664.9 |
| /web/dataset/call_... | 200 | 11685 | 224 | 42310 | 13470.95 | 9.50% | 2.7/min | 0.07 | 0.03 | 1642.1 |
| /web/static/img/pl... | 200 | 12881 | 199 | 42109 | 14076.94 | 11.50% | 2.7/min | 0.26 | 0.02 | 5996.6 |
| /partner_autocom... | 200 | 14741 | 399 | 42114 | 15343.68 | 15.50% | 2.7/min | 1.76 | 0.02 | 40792.4 |
| /web/dataset/call_... | 200 | 13992 | 226 | 42116 | 14702.06 | 13.50% | 2.7/min | 0.07 | 0.03 | 1702.8 |
| /web/dataset/call_... | 200 | 14459 | 225 | 42131 | 15536.76 | 15.50% | 2.7/min | 0.07 | 0.03 | 1733.1 |
| /mail/read_followe... | 200 | 15515 | 226 | 42188 | 15757.86 | 18.50% | 2.7/min | 0.08 | 0.02 | 1778.6 |
| /mail/get_suggest... | 200 | 16531 | 226 | 42121 | 16443.67 | 22.50% | 2.7/min | 0.08 | 0.02 | 1839.3 |
| /web/dataset/call_... | 200 | 16038 | 227 | 42114 | 15349.80 | 17.50% | 2.7/min | 0.08 | 0.03 | 1763.5 |
| /web/dataset/call_... | 200 | 15168 | 224 | 42161 | 15612.59 | 18.00% | 2.7/min | 0.08 | 0.03 | 1771.1 |
| /web/static/img/fo... | 198 | 15858 | 198 | 42112 | 15860.00 | 18.69% | 2.6/min | 0.04 | 0.02 | 876.5 |
| /mail/thread/mess... | 197 | 16591 | 226 | 42223 | 14917.87 | 15.23% | 2.6/min | 0.07 | 0.02 | 1729.0 |
| /mail/get_suggest... | 195 | 16223 | 223 | 42199 | 15243.97 | 16.41% | 2.6/min | 0.07 | 0.02 | 1746.9 |
| /mail/read_followe... | 194 | 15050 | 226 | 42121 | 15150.69 | 17.01% | 2.6/min | 0.07 | 0.02 | 1756.0 |
| /web/dataset/call_... | 193 | 17042 | 223 | 42155 | 15837.43 | 20.73% | 2.6/min | 0.08 | 0.02 | 1812.4 |
| /mail/static/src/im... | 193 | 15170 | 199 | 42113 | 15134.27 | 15.03% | 2.6/min | 0.11 | 0.02 | 2595.2 |
| /web/dataset/call_... | 192 | 15188 | 227 | 42179 | 15557.13 | 16.67% | 2.6/min | 0.07 | 0.03 | 1750.8 |
| /web/dataset/call_... | 191 | 16049 | 226 | 42181 | 16288.40 | 21.99% | 2.5/min | 0.08 | 0.03 | 1831.6 |
| /canonical.html-96 | 191 | 224 | 117 | 2499 | 251.63 | 0.00% | 2.5/min | 0.01 | 0.01 | 302.0 |
| /success.txt-98 | 191 | 138 | 57 | 3503 | 322.94 | 0.00% | 2.5/min | 0.01 | 0.01 | 220.0 |
| /success.txt-97 | 191 | 106 | 58 | 1441 | 127.00 | 0.00% | 2.5/min | 0.01 | 0.01 | 220.0 |
| TOTAL | 18726 | 9226 | 57 | 214095 | 14895.06 | 8.08% | 3.9/sec | 299.80 | 2.05 | 78523.1 |

**Figura 64** Resultados en JMeter en AWS con un porcentaje de error de 8.08%

### Métricas de la CPU en la BD de AWS

Se analiza el porcentaje de utilización de CPU, con un valor máximo de 7.43%, cuando se envía una carga de trabajo en JMeter.

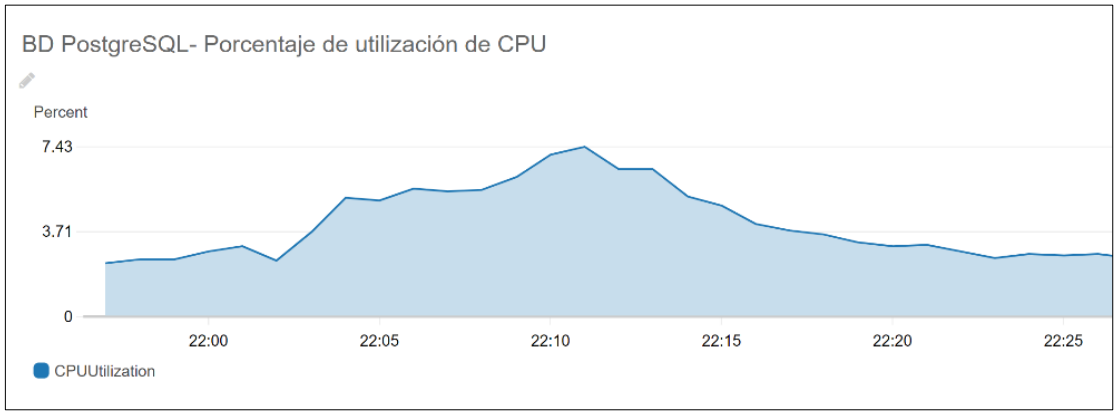


**Figura 65** Porcentaje Máximo del uso del CPU en AWS al ejecutar Jmeter

En la figura 63 se detalla el almacenamiento utilizado en la BD PostgreSQL. El

almacenamiento para una BD básico en AWS es de 5 GB. La métrica TransactionLogsDiskUsage es el espacio de disco utilizado por los registros de transacciones, da un valor de 2.35GB.

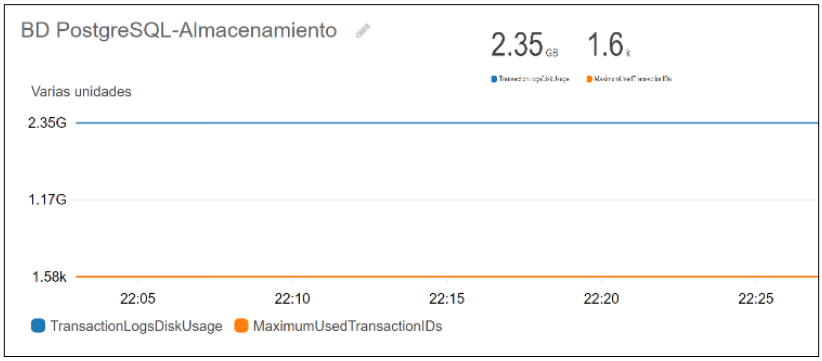


**Figura 66** Información del espacio de disco en la BD AWS

La métrica MaximumUsedTransactionIDs, es el número máximo de ID de transacción que se han utilizado con un valor de 1.60K.

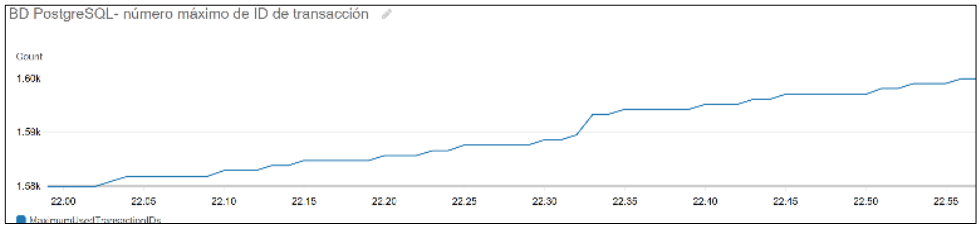


**Figura 67** número máximo de ID de transacción de la BD en AWS

A continuación, se detalla el tráfico de red de salida (transferencia) en la instancia de base de datos, incluidos el tráfico de base de datos del cliente y el tráfico de Amazon RDS utilizado en monitoreo y replicación con un valor máximo aproximado de 116Kbs y el tráfico de red de entrada (recepción) tiene un valor máximo aproximado de 35Kbs.

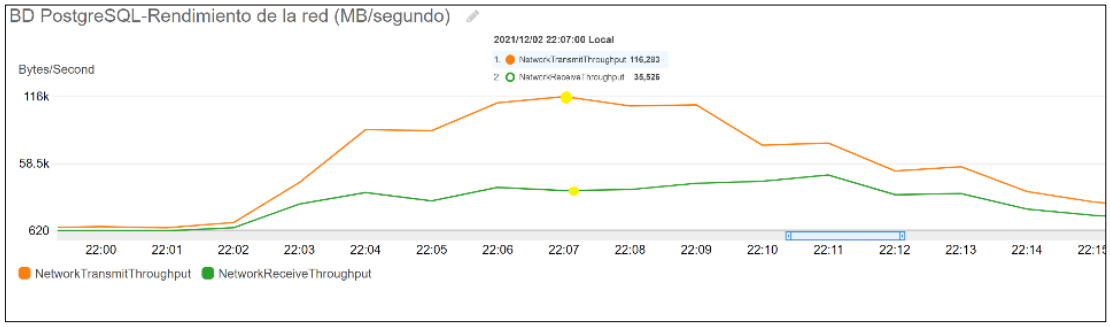


**Figura 68** Rendimiento de transmisión y recepción de red en la BD de AWS

## Métricas de la instancia Odoo de AWS

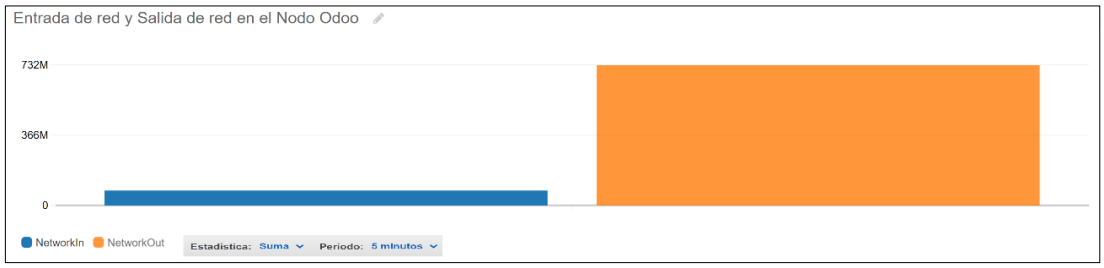


**Figura 69** Entrada y salida de la red en el nodo

## Métricas de la Red en el Nodo (aks-default-26787434-vmss000000)

En la figura 67 se muestra el porcentaje de unidades informáticas EC2 asignada en la instancia con la imagen Odoo. Esta métrica identifica la capacidad de procesamiento necesaria para ejecutar Odoo en la instancia donde fue instalada el servicio. El pico más alto fue aproximadamente 9.66%.

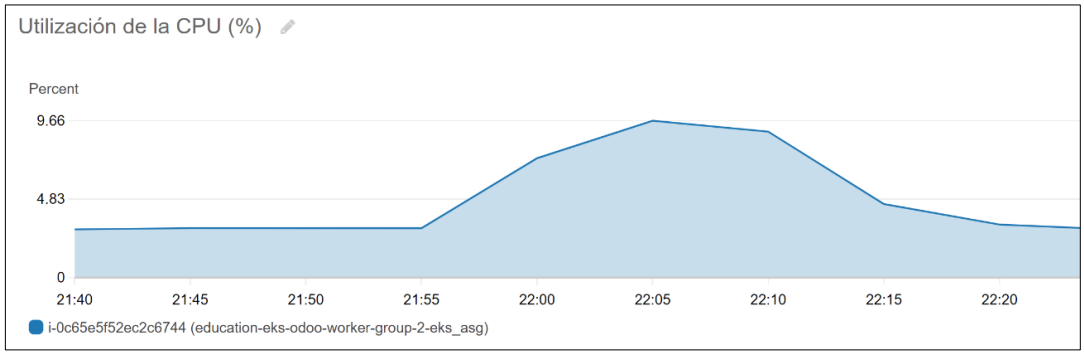


**Figura 70** Porcentaje Max. Del CPU en la instancia con la aplicación Odoo