PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

SEDE ESMERALDAS

CARRERA

INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

TESIS DE GRADO

HERRAMIENTAS DE PRUEBA DE SOFTWARE BASADOS EN LA NUBE (CLOUD TESTING)

LÍNEA DE INVESTIGACIÓN

PROGRAMACIÓN Y DESARROLLO DE SOFTWARE

AUTOR

ESTUPIÑÁN BENAVIDES ELIAS MAXIMILIANO

ASESOR

Mgt. XAVIER QUIÑONEZ KU

ESMERALDAS, 2022.

TRIBUNAL DE GRADUACIÓN

Trabajo de tesis aprobado luego de haber dado cumplimiento a los requisitos exigidos por el Reglamento de Grado de la PUCESE, previo a la obtención del título de Ingeniería en Sistemas y Computación.

Presidente del Tribunal de Graduación **f.**_____

Mgt. Xavier Quiñonez Ku **f.**_____
Asesor

Mgt. Susana Patiño **f.**_____
Lector #1

Mgt. Jaime Sayago **f.**_____
Lector #2

Mgt. Susana Patiño **f.**_____
Coord.ª de carrera

AUTORÍA DE TESIS

Yo, ELÍAS MAXIMILIANO ESTUPIÑÁN BENAVIDES, portador de la cédula de ciudadanía 080375439-9 declaro que el desarrollo de la presente investigación es totalmente de carácter original, auténtico y personal.

En tal virtud, manifiesto que el contenido, de las conclusiones, los efectos legales y académicos que se desprenden del trabajo propuesto de Investigación y luego de la redacción de este documento son y serán de mi sola y exclusiva responsabilidad legal y académica.

Elias Maximiliano Estupiñán Benavides

C.I: 0803754399

DEDICATORIA

A Dios por ser mi fuente de inspiración, por darme calma, salud y la fuerza para llevar a cabo mis metas y objetivos. Quiero darle gracias por su gran amor.

A mi familia, mi madre y hermano que siempre estuvieron apoyándome en buenos y malos momentos. De igual forma quiero agradecer especialmente a mis abuelos Sabina y Miguel por inculcarme buenos valores desde pequeño, sin ellos no sería nada en este mundo y ojalá pueda llegar a ser la mitad de excelentes personas como lo son ellos.

Elias Maximiliano Estupiñán Benavides

AGRADECIMIENTO

Agradezco a Dios por darme el don de la paciencia, perseverancia y dedicación que me permitió lograr cada reto presentado en esta etapa de mi vida

A mi familia, por ser pilar fundamental dentro de mi vida y estudios, por acompañarme en este arduo proceso siendo mi apoyo y refugio.

Agradezco a los docentes de la Escuela de Sistemas y Computación quienes con dedicación y profesionalismo supieron guiarme durante mi carrera universitaria.

A mis amigos, en especial a David Hurtado, Joe Magallán y Roberto Carvajal, por acompañarme desde los primeros días en la universidad y apoyarme en todo lo que fuera necesario.

Finalmente, quiero agradecerme a mí por todo el esfuerzo realizado alrededor de estos años de carrera, por no renunciar nunca.

Elias Maximiliano Estupiñán Benavides

Índice de contenidos

Índice anexos

Índice de figuras

Índice de tablas

Índice de gráficos

RESUMEN

El surgimiento de la computación en la nube y la gran demanda que ha obtenido el mismo, ha provocado que empresas que se dedican al desarrollo de software opten por aplicar este nuevo paradigma en sus instalaciones, por ende, el uso de herramientas para prueba de software convencional ha cambiado totalmente, al hecho de convertir este método en un proceso que se realiza únicamente en la nube.

A raíz de este cambio, a lo largo de estos años han surgido un sin número de herramientas orientadas a la ejecución de pruebas de software, por lo cual, esta investigación tiene el propósito de evaluar tres de las más conocidas, que son J Meter, Locust, Taurus en distintos proveedores de servicio en la nube, a fin de conocer e identificar qué herramienta se adapta mejor, ya sea en Google Cloud Platform, Amazon Web Services o Microsoft Azure.

Para poder evaluar de forma correcta dichas herramientas se utilizó una metodología experimental, debido a que es necesario probar, ejecutar scripts que creen una infraestructura con recursos, ejecutar pruebas de carga y estrés, y finalmente eliminar el recurso. Además, también se utilizó una ficha de evaluación al software para medir tres características basadas en la normativa internacional ISO/IEC 25010 referente a la calidad del producto software, las cuales son usabilidad, eficiencia de desempeño y compatibilidad entre la herramienta y el proveedor.

Los resultados obtenidos demuestran que, si existe una diferencia cuando se emplea una herramienta distinta en uno u otro proveedor, estas diferencias se deben a la sintaxis, cantidad de recursos y configuración que emplea cada proveedor. También se reveló que los tiempos de respuesta de las ejecuciones de las pruebas de software cambian dependiendo del proveedor lo cual es un factor importante a tomar en cuenta. De esta manera, queda demostrado que la herramienta qué más destaca es Locust siendo integrada en Google Cloud Platform, obteniendo un alto nivel de usabilidad, compatibilidad y eficiencia de desempeño.

ABSTRACT

The emergence of cloud computing and the great demand that it has obtained, has caused companies that are dedicated to software development to choose to apply this new paradigm in their facilities, therefore, the use of software testing tools Conventional has totally changed, by turning this method into a process that is carried out only in the cloud.

As a result of this change, over the years a number of tools oriented to the execution of software tests have emerged, therefore, this research has the purpose of evaluating three of the best known, which are J Meter, Locust, Taurus in different cloud service providers, in order to know and identify which tool is best suited, either on Google Cloud Platform, Amazon Web Services or Microsoft Azure.

In order to correctly evaluate these tools, an experimental methodology was used, since it is necessary to test, execute scripts that create an infrastructure with resources, execute load and stress tests, and finally eliminate the resource. In addition, a software evaluation form was also used to measure three characteristics based on the international standard ISO/IEC 25010 regarding the quality of the software product, which are usability, performance efficiency and compatibility between the tool and the supplier.

The results obtained show that, if there is a difference when a different tool is used in one or another provider, these differences are due to the syntax, amount of resources and configuration used by each provider. It was also revealed that response times for software test executions change depending on the vendor, which is an important factor to take into account. In this way, it is shown that the tool that stands out the most is Locust being integrated into Google Cloud Platform, obtaining a high level of usability, compatibility and performance efficiency.

INTRODUCCIÓN

Presentación del problema

La computación en la nube es considerada como una tecnología que se caracteriza por ofrecer a los usuarios una gran variedad de servicios por medio de Internet. Dentro de los servicios que brinda la nube se encuentran las pruebas de software, la cual consiste en determinar el rendimiento, escalabilidad y confiabilidad de una aplicación web, y mediante simulaciones minimiza el costo de pruebas basadas en el mundo real. Generalmente, el uso de estas plataformas es de paga, es decir que, se debe pagar en función a lo que se pretende realizar. Un claro ejemplo de esto es el almacenamiento ilimitado, el cual ayuda a reducir el tiempo de prueba al momento de testear aplicaciones web [1].

A medida que el tiempo avanza, la demanda que obtiene la computación en la nube a través de los servicios que brinda es alta, ganando así terreno en distintas comunidades de desarrolladores a nivel global. Este hecho, que ha marcado un antes y un después en el desarrollo de aplicaciones ha ocasionado que distintas compañías, como Microsoft, Google, Amazon, entre otras, creen sus propias plataformas para beneficio y uso de varias comunidades, entre ellas la de desarrolladores, teniendo como consecuencia el incremento excesivo de las mismas.

Por otra parte, este incremento de entornos en la nube provoca incertidumbre en el desarrollador al momento de usar una herramienta para evaluar la aplicación o servicio web, por lo cual es importante conocer las características funcionales y de calidad que poseen las mismas, mediante métodos de prueba y evaluación específico [2].

Planteamiento del problema

Hoy en día, el término computación en la nube (Cloud Computing) tiene gran repercusión en el ámbito informático, por lo cual se ha convertido en algo muy popular. La computación en la nube no solo ha cambiado la forma en que trabajan las organizaciones de TI, sino que también ha cambiado en gran medida la manera en que se realizan pruebas y se mantiene el software. La esencia de éste radica en compartir recursos, evitando la necesidad de tener servidores locales o dispositivos personales, teniendo como objetivo primordial reducir el tiempo total de las pruebas de software. Debido a su infraestructura, las nubes proporcionan grandes cantidades de almacenamiento, por lo tanto, permiten

mayor flexibilidad y producen un tiempo de respuesta mucho más rápido al probar una determinada aplicación [3].

Generalmente las pruebas de software son un proceso muy importante dentro de empresas que se dedican al desarrollo de softwares y aplicaciones. Aunque consumen mucho tiempo, dinero y recursos, estas tienen como objetivo facilitar y reducir los errores humanos al automatizar las prácticas de prueba utilizando herramientas de automatización de software [4]. A partir de esto, surge la necesidad de conocer cuánto alcance y qué beneficios ofrecen estas herramientas al momento de ejecutar pruebas de software dentro de un entorno en la nube.

Otra problemática que existe hoy en día es la gran variedad de plataformas basadas en la nube. Actualmente son muchas empresas las que se dedican a ofrecer servicios de pruebas de software. Entonces, esto crea incertidumbre dentro de las empresas dedicadas al desarrollo de aplicaciones e incluso a desarrolladores independientes, porque se desconoce qué herramienta o qué plataforma va a facilitar el proceso de pruebas de software.

Justificación

La presente investigación se basó en la gran demanda que existe hoy en día en cuanto a prestación de servicios en computación en la nube, debido a que gran cantidad de empresas han optado por crear plataformas donde ofrecen este servicio.

De ahí surgió la necesidad de evaluar las plataformas basadas en la nube más importantes en la actualidad, enfocando esta investigación especialmente en el ámbito de las herramientas que ofrece el servicio de pruebas de software, para el evalúo o testeo de aplicaciones web. Además, también se comprendió el alcance que posee estas herramientas y qué beneficios otorgan la misma, y finalmente a través de este proceso se indica cuál brinda un mejor servicio.

El proceso de esta investigación es de gran importancia debido a que las empresas que se dedican al desarrollo de aplicaciones, llevar a cabo pruebas de software son una fase importante dentro del ciclo de vida del desarrollo de software. Sin embargo, técnicamente se puede considerar que estas pruebas abarcan un proceso de verificación y validación a fin de garantizar que la aplicación cumpla con los requisitos previstos [5].

OBJETIVOS

General

Ejecutar una prueba de software mediante el uso de herramientas bajo distintos entornos de computación en la nube para la obtención de un análisis de usabilidad, eficiencia de desempeño y compatibilidad de dichas herramientas.

Específicos

a) Definir qué herramienta se usará dentro de los entornos de computación en la nube sujetos a estudio (Microsoft Azure, Amazon Web Services y Google Cloud Platform).
b) Especificar el tipo de pruebas de software que se realizará en los proveedores de servicio en la nube.
c) Integrar las herramientas de prueba de software dentro de los entornos de computación en la nube
d) Comparar la calidad de servicio de las plataformas de computación en la nube a través de la herramienta seleccionada.

Capítulo 1

1 MARCO TEÓRICO

1.1. Bases conceptuales

Este proyecto investigativo se llevó a cabo teniendo en cuenta ciertos conceptos relacionados con la línea de investigación establecida en este documento, por lo cual se recopila, sistematiza y se expone conceptos fundamentales para una mejor comprensión de las temáticas sirviendo en el desarrollo del estudio.

1.1.1. Computación en la nube

Actualmente la computación en la nube se puede ver como un nuevo paradigma de la computación [6], el cual promueve o facilita recursos dinámicamente escalables y frecuentemente virtualizados, los cuales están provistos como servicios en internet. Básicamente, el objetivo específico de esta tecnología se trata de una implementación que pretende transformar el modelo tradicional de la computación y la informática y trasladarla a internet [7].

Por otra parte, el uso de la computación en la nube ha permitido a gran escala, que un sin número de empresas, sin distinción de la rama vocacional que pertenezcan, tengan presencia en la web o simplemente puedan adquirir servicios informáticos a un precio razonable, sin necesidad de inversión en equipos físicos y lógicos [8]. Estos servicios son proporcionados por proveedores de la nube, como lo es Microsoft, Amazon, Google, IBM, entre otros.

1.1.1.1 Principales características

La computación en la nube cuenta con varias características esenciales que la diferencian de la computación tradicional, entre ellas está [9]:

- **Centrado en el usuario:** La informática en la nube básicamente gira en torno al usuario, esto se debe a que este es quien se encarga de dar a conocer sus requerimientos y/o necesidades, lo cual la nube lo recepta como solicitudes y brinda su servicio.
- **Centrado en tareas:** En la nube, la informática depende mayormente de las solicitudes o requerimientos que el usuario necesite. Por lo tanto, esta característica hace que la nube centre gran cantidad de sus tareas en cumplir o satisfacer las necesidades de los usuarios.

- **Potente:** Independientemente de los requerimientos y solicitudes, la computación en la nube es poderosa debido a sus ventajas sobre los paradigmas informáticos existentes.
- **Accesible:** Como su nombre lo indica, la nube y los servicios que la misma ofrece están disponibles en Internet y son de fácil acceso.
- **Inteligente:** A diferencia de la informática convencional, la computación en la nube brinda a los usuarios cualquier cantidad de servicios, y de esta misma manera se debe pagar por lo que consume, es decir que, el usuario paga por uso de servicios.
- **Programable:** La computación en la nube se adapta a los requerimientos del usuario, por lo cual, las aplicaciones y/o servicios en la nube se crean a la medida de las necesidades del usuario.

1.1.1.2 Tipos de nube

En la literatura, la computación en la nube puede implementarse según los siguientes modelos de nube: pública, privada e híbrida según los requerimientos de la organización, empresa, o desarrollador independiente de modo que se pueda aplicar de manera viable a la infraestructura, servicio o aplicación.

Nube pública

El público en general o un gran grupo de la industria pueden acceder a los servicios en la nube para su uso, pagando según el método de uso. A los usuarios se les asignan los recursos en la nube bajo demanda. Los recursos se proporcionan de forma dinámica a través de Internet.

Las pequeñas y medianas empresas (PYME) se benefician en gran medida del uso de nubes. Las ventajas de las nubes públicas son la independencia de la ubicación, la rentabilidad, la fiabilidad, la flexibilidad, el coste del estilo de servicios públicos y la alta escalabilidad. Las desventajas son la baja seguridad y menos personalizable [6].

Este tipo de nube tiene como ventaja la capacidad de procesamiento y almacenamiento sin instalación de máquinas localmente, lo cual evita requerir una inversión inicial o gasto de mantenimiento, sino que se paga solamente por su uso. Sin embargo, posee inconvenientes en cuanto al acceso o disposición de toda la información a terceras empresas, y la dependencia de los servicios en línea. Esto sumando que puede resultar difícil integrar servicios con otros sistemas propietarios [10].

Nube privada

Este tipo de nube informática opera únicamente para organizaciones. Puede ser gestionado por la propia organización o por un tercero. La nube privada puede existir en las instalaciones o fuera de ellas, por lo que suelen ser diseñadas para proporcionar un control óptimo de la información gestionada, de su seguridad y de la calidad de servicio que ofrecen [10].

Las ventajas de las nubes privadas son una mayor seguridad y más privacidad, más control, costos y eficiencia energética. Cabe señalar que la característica que más destaca a diferencia de las nubes públicas, es la localización de los datos dentro de la propia empresa, lo que conlleva una mayor seguridad de estos [10]. Por otra parte, las desventajas o principales inconvenientes que presenta este modelo son la escalabilidad limitada debido a los recursos limitados, los precios inflexibles y la nube privada está limitada a un área en particular [6].

Nube híbrida

En cuanto a la infraestructura de este tipo de nube, se puede decir que es una composición de dos o más nubes (privada, comunitaria o pública), lo que la convierte como su nombre lo indica, en una nube "híbrida" o mixta. En este modelo de nube, de cada entidad que lo conforma se mantiene como única, pero están vinculados entre sí por tecnologías estandarizadas o patentadas [6].

Las nubes híbridas se centran en combinar aplicaciones propias de la empresa con las consumidas a través de la nube pública, entendiéndose también como la incorporación de servicios de computación en la nube a las aplicaciones privadas de la organización. Las ventajas de las nubes híbridas son la escalabilidad, la flexibilidad, la rentabilidad, mientras que las desventajas son los problemas de red y las normas de seguridad en cuanto a protección de datos [10].

1.1.1.3 Modelos de servicio en la nube

Una de las características más relevantes dentro del ámbito de la computación en la nube son los modelos de servicio que posee. En la nube todo se proporciona como servicio al incluir distintos modelos, como software como servicio (SaaS), plataforma como servicio (PaaS) e infraestructura como servicio (IaaS) [11].

A continuación, en la Tabla 1 se presenta la perspectiva que posee cada modelo en cuanto a su objetivo.

Tabla 1 Síntesis de modelos de Servicios en la Nube [12]

Software como Servicio (SaaS)	Se basa en ofrecer aplicaciones y productos bajo demanda
Infraestructura como Servicio (IaaS)	Facilita a la comunidad desarrolladora la construcción e implementación de aplicaciones
Plataforma como Servicio (PaaS)	De donde se dispone de recursos necesarios, como el almacenamiento, componentes de redes, herramientas, entre otros bajo demanda.

Software como Servicio (SaaS)

El término software como servicio, infiere básicamente al software residente, es decir, el que está instalado en la nube, aunque actualmente no todos los sistemas SaaS son sistemas instalados en la nube, la mayoría sí. Este modelo de servicio se caracteriza por poseer, entregar y administrar software a uno o más proveedores de manera remota [13].

De esta manera se puede definir a SaaS como un modelo de software basado en la Web el cual provee distintos tipos de software a través del navegador web, en donde cada una de las aplicaciones son accesibles desde diferentes dispositivos hacia el usuario final, por medio de una interfaz, tal cual navegador [14].

Las recomendaciones al momento de elegir este modelo de servicio son la protección de datos donde se deberá validar qué estrategias de protección de datos ofrece el proveedor, la protección del dispositivo y aplicación del cliente, el uso de cifrado fuerte con algoritmos robustos y el borrado seguro de datos, esto debido a que un proveedor de SaaS es el responsable de toda la seguridad del perímetro, el registro, monitoreo, auditoría y seguridad de aplicaciones [15].

Algunos ejemplos de este modelo son: Google Docs, Gmail, Microsoft Office 365.

Infraestructura como Servicio (IaaS)

Este modelo de servicio proporciona al consumidor de nube, capacidades de procesamiento, almacenamiento, redes y otros recursos de computación fundamentales y requeridos donde el consumidor es capaz de desplegar y ejecutar software de manera arbitraria [13]. Además, puede incluir también la entrega de sistemas operativos SO y tecnología de virtualización para la gestión de recursos [14].

Un dato a tener en cuenta es que en este modelo de servicio el proveedor es responsable de la seguridad fundamental, mientras que el usuario de la nube es responsable de todo lo que construye en la infraestructura, es decir que en este modelo se le otorga mucha más responsabilidad al cliente [15].

Los aspectos que se deben tener en cuenta al momento de elegir este modelo de servicio es la recepción de múltiples clientes, corroborar que cuando el proveedor ofrezca múltiples recursos como máquinas virtuales, estos tengan mecanismos de protección contra ataques, además que se debe validar que exista una buena protección de datos, al igual que el borrado seguro de los mismos, también tener acceso administrativo, y poseer migración de máquinas virtuales [15].

Entre los ejemplos de este modelo de servicio encontramos: Google Cloud Compute Engine, Microsoft Azure, Amazon Web Services (AWS).

Plataforma como Servicio (PaaS)

Este modelo de servicio es un nivel intermedio entre la gestión interna y externa de computación en la nube, por lo que este modelo toma la infraestructura de aplicaciones, sistemas operativos y detalles en las configuraciones para que los desarrolladores puedan aprovisionar, desarrollar, crear, probar e implementar aplicaciones sin la necesidad de asistencia de TI (Tecnología de la información)[13].

La plataforma como servicio ofrece un entorno muy rentable, esto porque facilita a los desarrolladores de aplicaciones y pequeñas empresas a expandirse a través de aplicaciones web sin el coste y complejidad que supondría la compra de servidores, configuraciones y la puesta en funcionamiento [14].

Las recomendaciones al momento de elegir este modelo de servicio son evaluar si las interfaces de la aplicación son de uso genérico que ofrezcan portabilidad e interoperabilidad, tener en cuenta que se acepten lenguajes y herramientas comerciales, el acceso y la protección de datos, la seguridad y la prueba de componentes [15].

Ejemplos de Plataforma como Servicio son: App Engine de Google, Elastic Beanstalk de AWS Y Azure de Microsoft.

1.1.1.4 Proveedores de servicio en la nube

Los proveedores de nube utilizan economías de escala para ofrecer atractivos servicios de nube a precios asequibles. Esto libera a los innovadores de tecnología a implementar

en sus productos características claves que los diferencian entre otros. Sin embargo, al igual que con cualquier plataforma tecnológica, se debe tener cuidado de usarla con prudencia.

La selección de un proveedor de nube sobre los demás se reducirá a los deseos y necesidades de cada cliente individual y las cargas de trabajo que están ejecutando. A menudo ocurre que las organizaciones utilizarán múltiples proveedores dentro de diferentes partes de sus operaciones, o para diferentes casos de uso, lo que se denomina enfoque de múltiples nubes.

Recientemente la computación en la nube se ha convertido en una industria en crecimiento [16]. Actualmente existen tres proveedores que lideran el mercado: Microsoft Azure, Amazon Web Services (AWS) y Google Cloud Platform (GCP). Sin embargo, existe una serie de factores clave que separan o diferencian los enfoques de estas tres empresas, lo cual ayuda a los usuarios finales a considerar cuál es el adecuado para ellos.

Amazon Web Services

En el caso de AWS, esta empresa sigue creciendo exponencialmente debido a su amplio conjunto de herramientas, haciendo a este proveedor incomparable. AWS ofrece un extenso conjunto de servicios de infraestructura, como opciones de almacenamiento, redes y bases de datos que se suministran como una utilidad bajo demanda, es decir que proporcionan estos servicios con precios de pago por uso [17]. Sin embargo, la estructura que posee puede ser confusa y la interoperabilidad con el centro de datos no es la principal prioridad de AWS, debido a su enfoque singular en la nube pública en lugar de la híbrida o privada.

Microsoft Azure

Por otro lado, se tiene a Microsoft Azure el cual posee una infraestructura de nube excepcional, que le permite entregar rápidamente servicios nuevos e innovadores a sus usuarios, expandiéndose más allá de lo que se puede hacer con hardware y software estándar [18]. Por otra parte, Azure permite desarrollar aplicaciones en casi cualquier idioma, permitiendo integrar aplicaciones públicas desde la nube, y al poseer un centro de datos distribuido permite que sus usuarios tengan control de acceso a sus datos y aplicaciones [19].

Google Cloud Platform

Por último, se encuentra Google Cloud, que ingresó al mercado de la nube un poco más tarde que AWS y Azure pero que su experiencia técnica es profunda y su conjunto de servicios modulares basados en la nube que proporcionan componentes básicos le permite al usuario desarrollar todo, desde sitios web simples hasta aplicaciones sofisticadas basadas en web de varios niveles. Una de las características fundamentales de GCP son los contenedores de nivel superior que posee, donde a través de estos puede consolidar todos los recursos relacionados con la computación en la nube, permitiendo trabajar en varios proyectos al mismo tiempo mientras se asegura de que los recursos estén bajo dominios de control por separado, es decir que cada proyecto que se realice dentro de la plataforma se identificará bajo una tupla [20].

Tabla 2 Comparación entre AWS, GCP y AZURE [21]

Entornos	**Amazon EC2**	**GCP**	**Microsoft Azure**
General			
Año de inicio	2006	2011	2010
Regiones disponibles	16	21	52
Modelos de servicio	SaaS, PaaS & IaaS con importante contribución en IaaS	SaaS, PaaS & IaaS con importante contribución en PaaS	SaaS, PaaS & IaaS con importante contribución en PaaS
IDE	Soporte SDK para Eclipse	Soporte directo en Cloud9 IDE	Soporte SDK para Eclipse & Visual Studio
Base de datos y virtualización			
Base de datos	1. MySql 2. PostgreSQL 3. MariaDB 4. MongoDB 5. Redis 6. Memcached	1. Cloud SQL 2. Cloud Spanner 3. Cloud Bigtable 4. Cloud Firestore 5. Firebase Realtime Database 6. Cloud Memorystore	1. Azure SQL
Tipos de máquinas virtuales	1. Propósito general 2. Computación optimizada 3. Memoria optimizada 4. Optimizar almacenamiento 5. Computación acelerada	1. Máquinas estándar 2. Máquinas de alta memoria 3. Máquinas de alta CPU 4. Máquinas de mega memoria	1. Propósito general 2. Computación optimizada 3. Memoria optimizada 4. Optimizar el almacenamiento 5. Computación de alto rendimiento
Tecnología de virtualización	XEN	KVM	Hyper-V

Precios			
Tipos de precios	Facturación por segundo a pedido	Pago por uso, facturación a pedido por segundo	Precios de pago por uso
Especificaciones			
Server OS	Linux, Windows	Linux, Windows	Linux, Windows
OS preconfigurado	1. Amazon Linux 2. Cent OS 3. Debian 4. Oracle Linux 5. Red Hat Linux 6. Ubuntu 7. Windows Server	1. Cent OS 2. Debian 3. Ubuntu 4. Red Hat Linux 5. Windows Server	1. Cent OS 2. FreeBSD 3. OpenSUSE Linux 4. Oracle Linux 5. Ubuntu 6. Windows server
Runtimes	1. .NET 2. JAVA 3. PHP 4. Python 5. Ruby	1. Python 2. JAVA 3. Node 4. PHP 5. Ruby 6. GO	1. .NET 2. JAVA 3. Node 4. PHP 5. Python 6. Ruby
Frameworks Machine Learning Compatibles	1. Apache 2. MXNet (con Gluon API) 3. TensorFlow 4. Caffe framework	1. TensorFlow 2. DistBelief 3. Muchas API's integradas para respaldar el desarrollo	1. PyTorch 2. TensorFlow 3. Scikit-learn 4. MXNet 5. Chainer 6. Keras
Soporte y Administración			
Soporte disponible	24/7, foros, recursos de autoayuda y documentación	Las respuestas tardan mínimo 15 minutos (con precios de primer nivel) y más de 1 día, comunicaciones telefónicas, foros, capacitaciones y documentación.	24/7, foros, chats en vivo, comunicaciones telefónicas, documentación.

A partir de la comparativa relacionada con los proveedores de servicio en la nube, la Tabla 2 de manera sintetizada refleja en aspectos generales que Azure es la plataforma más extendida de las tres, luego está GPC que proporciona un soporte IDE directo, y AWS como el más antiguo de los tres con un conocimiento firme de modelo de servicio IaaS. Por otra parte, en cuanto a base de datos y virtualización, la nube de Google ofrece mayor cantidad de opciones en cuanto a base de datos, AWS ofrece mayor cantidad de servicios en cuanto a virtualización, y Azure tiene menos alternativas. Luego está el precio donde cada proveedor posee un esquema que se basa en el pago por uso.

En cuanto a especificaciones, Azure proporciona mayor cantidad de compatibilidad con el marco de aprendizaje automático, Google cuenta con mayor cantidad de tiempos de ejecución y Amazon tiene la mayor cantidad de sistemas operativos preconfigurados. Por último, en cuanto a soporte las tres plataformas disponen de buenos recursos para uso de la comunidad.

1.1.2 Pruebas de software

1.1.2.1 Fundamentos

Las pruebas basadas en la nube son una fase importante del ciclo de vida del desarrollo de software. Puede considerarse como un proceso de verificación y validación de la aplicación que se está probando, para garantizar que se construya según el diseño y las especificaciones, y que la aplicación cumpla con los requisitos previstos [22].

Generalmente se usan para evaluar las aplicaciones web en cuanto a escalabilidad, rendimiento, seguridad y confiabilidad. Como su nombre lo indica, estos tipos de pruebas se realizan en un entorno de computación en la nube, que alberga la infraestructura necesaria para realizar las pruebas. Los diversos tipos de procesos de prueba en la nube permiten probar software y hardware sin las limitaciones habituales como lo es el presupuesto limitado, problemas geográficos, múltiples casos de prueba, costos elevados por prueba, entre otros.

La prueba de software es el proceso que se puede realizar simultáneamente durante el desarrollo o después de la finalización del desarrollo. Por lo tanto, tienen como objetivo principal encontrar la corrección, integridad y calidad del software según los requisitos del usuario [23].

Las pruebas como tal son el paso más importante dentro del ciclo de vida del desarrollo de software. Las pruebas se pueden realizar de manera tradicional dentro de la organización o fuera de la misma, es decir, pruebas en la nube. La computación en la nube cambia el escenario de las pruebas tradicionales de manera que aprovecha los recursos proporcionados por la computación en la nube, siendo esta su característica o diferencia principal con relación a las pruebas tradicionales. Mientras que la similitud entre estas pruebas se basa en asegurar la calidad de las funciones y el rendimiento del software.

En la Tabla 3 se muestra una breve comparación entre una prueba tradicional y una basada en la nube.

Tabla 3 Comparación prueba tradicional vs nube [12]

Prueba en la nube	**Prueba tradicional**
El costo de prueba es menor debido a que el pago se basa en el uso de los recursos que están disponibles en la nube	El costo es alto debido a que el hardware, software que se usa junto con el costo de la licencia están incluidos
La simulación de tráfico es virtual y en línea	La simulación de tráfico de datos es en línea
La simulación de acceso de usuarios virtuales o en línea	La simulación de acceso de usuarios en línea
El costo de las pruebas depende del uso de recursos, es decir que se paga por uso	El costo dependerá del hardware y software que se use, incluyendo el costo de licencia
Las pruebas bajo demanda serán hechas bajo los recursos disponibles en la nube	Se deberá preinstalar todos los recursos para comenzar la prueba
Es económico en el caso de que se necesite más herramientas durante la realización de la prueba	No es económico debido a que se deberá instalar primero las herramientas adicionales que se necesiten incluyendo su licencia
Consume menos tiempo debido a que la propia nube proporciona herramientas e infraestructura de prueba	El proceso es más tardío debido a que se debe instalar las herramientas de prueba y la infraestructura se debe configurar en el sistema

Generalmente, para llevar a cabo pruebas de software en la nube se debe seguir una serie de pasos que son necesarios realizarlos de modo secuencial y ordenado. Teniendo en cuenta esto, como primer paso se deberá desarrollar el escenario del usuario, luego de esto diseñar los casos de prueba, seleccionar el proveedor de servicio en la nube que proporcionará las herramientas para las pruebas en la nube.

Luego, los siguientes pasos se basarán en crear tráfico web para después iniciar el proceso de prueba de software mediante la ayuda de los casos que se crearon inicialmente. Una vez finalizada la prueba, se verifica si se cumplen o no los objetivos de la prueba en particular. Y finalmente en el último paso, se procede a entregar el resultado a la organización que solicitó la prueba [24].

En la Figura 1 se muestra de manera resumida los pasos para llevar a cabo una prueba de software en la nube.

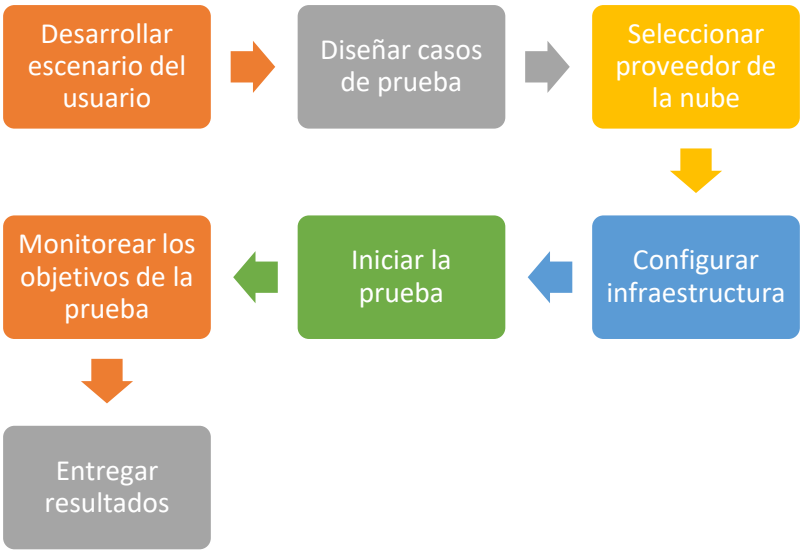

Figura 1 Pasos de prueba de software

1.1.2.2 Tipos de prueba en la nube

Actualmente, las pruebas de software varían dependiendo de la funcionalidad del servicio web o aplicación que se requiere analizar. Existen diferentes tipos de prueba de software en la nube, donde a través del uso de infraestructuras y entornos se simulan escenarios de tráfico de usuarios. De esta manera se puede medir la funcionalidad, rendimiento, la carga, seguridad del software, entre otras, sin necesidad de invertir parte del presupuesto en equipo.

Para trabajar en pruebas de software el desarrollador se debe centrar específicamente en el software como servicio. Este es uno de los modelos más importantes en la computación en la nube debido a que su funcionalidad se basa en ofrecer aplicaciones que se ejecutan mediante una infraestructura en la nube, estando únicamente enfocadas al usuario final. Es decir que, SaaS se centra en ofrecer una aplicación de software a través de internet, incluyendo todas sus funcionalidades de manera que esté disponible para todos los clientes que la soliciten [25].

Con el progreso de la computación en la nube las plataformas SaaS se han destacado en ofrecer diversos servicios en la nube. Por ello, para poder implementar alguna forma de prueba SaaS se necesita conocer detalles de este método de prueba en específico, debido a que después de completar una aplicación SaaS, esta debe ser probada para garantizar la

calidad del software a través de actividades de validación que se reflejan como pruebas de rendimiento, seguridad, carga, latencia, entre otros, como se muestra en la Figura 2.

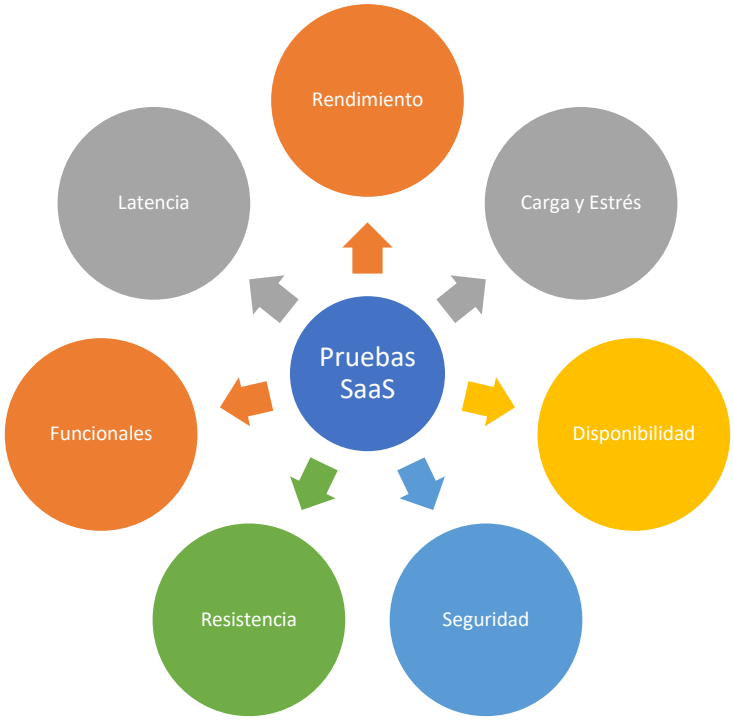

Figura 2 Pruebas SaaS

Teniendo en cuenta esto, se puede clasificar las pruebas de software SaaS en varios tipos. En la Tabla 4 se muestra una síntesis de cada prueba:

Tabla 4 Pruebas de software en la nube [12]

Pruebas de disponibilidad	Tienen como objetivo garantizar la disponibilidad del software 24/7.
Pruebas de seguridad	Buscan garantizar la integridad de datos y permitir acceso seguro a la nube.
Pruebas de interoperabilidad	Se centran en garantizar el funcionamiento correcto de las aplicaciones bajo diferentes entornos.
Pruebas de rendimiento	Integran pruebas de carga y stress, donde miden el tiempo de respuesta e incrementan la demanda de servicios y peticiones a fin de establecer un punto límite de la nube.
Pruebas de latencia	Se basan en analizar el proceso de envío y recepción de peticiones a fin de determinar fallos en la nube.

Pruebas de escalabilidad	Buscan determinar la capacidad de crecimiento que posee la aplicación y la infraestructura de la nube.
Pruebas de resistencia	Miden las incidencias que pueden presentarse en la nube en un determinado tiempo.
Pruebas Funcionales	Se enfocan en el cumplimiento de los requerimientos del cliente.

1.1.2.3 Técnicas de prueba

Existen diferentes técnicas de pruebas funcionales para poder llevar a cabo pruebas de software a una aplicación, entre ellas se encuentran las Prueba Caja Blanca y Caja Negra [26]. Una de las técnicas principales lleva por nombre prueba de caja blanca; en este tipo de técnica, las pruebas se desarrollan teniendo el conocimiento del código, es decir, los detalles de implementación interna y la estructura del código. Por lo tanto, es altamente productiva para detectar errores y resolver los problemas causados por esos errores.

En esta estrategia, la persona que lleve a cabo la prueba debe tener un conocimiento absoluto de lo que los componentes del programa se conectan y cooperan entre sí para encontrar errores. Para usar esta metodología de prueba, se requiere conocimiento del código, este método rara vez se usa en la práctica. La prueba de caja blanca también se conoce como prueba de caja transparente, análisis de caja blanca o análisis de caja transparente [17].

Por otro lado, existe también la prueba de caja negra; este tipo de técnica se enfoca en detalles internos, funcionamiento o estructura, de la aplicación. En este tipo de pruebas, el evaluador no tendrá ningún conocimiento sobre el código desarrollado o la forma en que está diseñado. De esta manera se tendrá el conocimiento sobre las especificaciones requeridas y los resultados esperados por el producto. Esta técnica tiene como principal objetivo probar cómo se comporta el sistema para diferentes entradas. Prueba solo para el requisito principal del producto. Verificará las salidas para varios tipos de entradas y verificará si el sistema falla en alguna entrada típica [27].

1.1.3 Herramientas de prueba de software

Todo proyecto por muy pequeño que sea puede llegar a tener una cantidad de casos de prueba muy elevado, por lo tanto, se necesita de una administración, planificación y ejecución, así como de herramientas que permitan realizar pruebas automáticas. Las

herramientas de prueba ayudan a que el proyecto que se está llevando a cabo se maneje de manera más eficiente y a su vez la calidad de este sea excelente. El evaluador es el encargado de estudiar, plantear y seleccionar el tipo de herramienta que se va a usar durante el proceso de prueba.

Las herramientas de gestión de pruebas se utilizan para almacenar información. Esa información puede ser similar a cómo se van a realizar las pruebas, las actividades del plan de pruebas y el estado del informe de las actividades de garantía de calidad. Las herramientas tienen formas distintivas de lidiar con las pruebas y, en consecuencia, tienen un conjunto diverso de características. Por lo general, se utilizan para mantenerse al día y organizar pruebas manuales, ejecutar o acumular información de ejecución de pruebas mecanizadas, lidiar con diferentes situaciones e ingresar datos sobre imperfecciones descubiertas [27].

Existen varias herramientas para la gestión de pruebas, en la Tabla 5 que se muestra a continuación se realiza una breve síntesis en cuanto al tipo de prueba y la metodología que abarcan.

Tabla 5 Herramientas para pruebas en la Nube [11]

Blazemeter	Permite realizar enormes pruebas de cargas al ejecutar en la nube, simulando casos de usuarios para aplicaciones, sitios y servicios web.
LoadStorm	Efectúa pruebas de rendimiento y de carga, donde a través de la infraestructura que emplea la nube se puede expandir la ponencia según sea necesario para probar la aplicación.
SOASTA	Permite al usuario construir, ejecutar y analizar pruebas de rendimiento y carga a cualquier escala de tráfico.
NeoLoad	Permite efectuar pruebas de carga completa y colaborativa permitiendo una validación continua para eventos o acciones de comportamiento por parte del usuario.
Locust	Realiza pruebas de rendimiento escalable, programable y fácil de usar permitiendo definir el comportamiento de sus usuarios.
Taurus	Es una herramienta de automatización que permite realizar pruebas de código abierto, ejecutado a través de línea de comandos
JMeter	Es un proyecto de Apache que se lo usa como herramienta de prueba de carga para analizar y medir el rendimiento de una variedad de servicios.

Blazemeter

Esta herramienta básicamente se utiliza para medir las pruebas de carga y el rendimiento de aplicativos webs, aplicaciones móviles, interfaz de aplicaciones, entre otros. Además la herramienta Blazemeter proporciona informes en tiempo real al momento de realizar dichas pruebas [28].

Al ser una extensión de Chrome, Blazemeter puede grabar, navegar, cargar y ejecutar pruebas, para ello se crean scripts de prueba y escenarios de carga adecuados, de manera que, se crea el script, se elige el número de motores de carga y se ejecuta la prueba. Cabe recalcar que la herramienta posee un número ilimitado de motores de carga los cuales están preconfigurados y disponibles para disposición del usuario [29].

Blazemeter al estar diseñada para uso profesional, está equipada con una plataforma de autoservicio, bajo demanda y capacidades avanzadas de scripting, lo que le permite ejecutar múltiples pruebas de carga tanto desde la nube pública como desde el interior del firewall corporativo, de manera que permite al usuario y/o cliente localizar y solucionar de manera ágil los cuellos de botella en el rendimiento de sus aplicativos web [30]. En la Figura 3 se muestra la interfaz de usuario que posee esta herramienta.

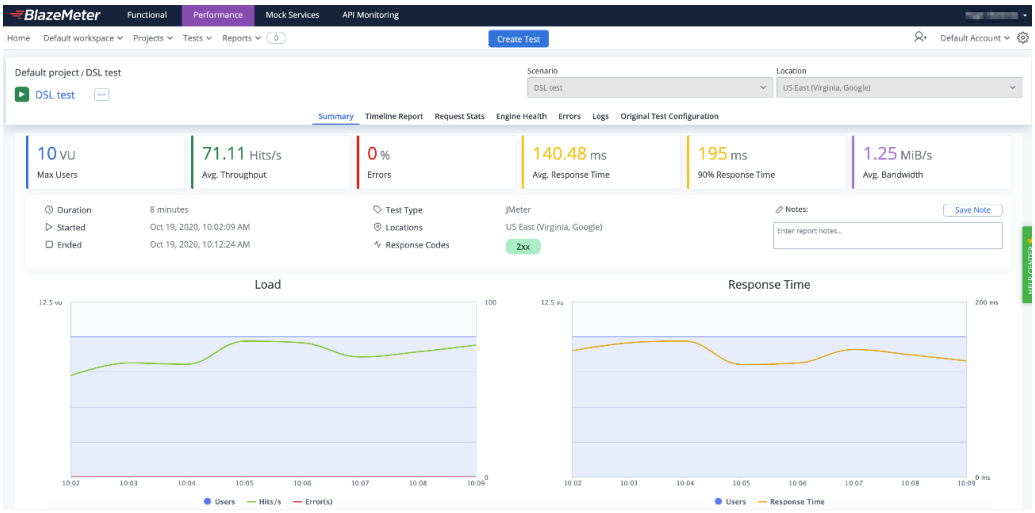

Figura 3 Interfaz de usuario en Blazemeter

LoadStorm

Esta herramienta es de naturaleza simple y de menor costo, se utiliza para realizar pruebas de carga de aplicaciones móviles y basadas en web [28]. LoadStorm proporciona interfaces web y móviles para aplicar varios escenarios de prueba de carga, de manera que se pueda personalizar de una manera sencilla y rentable los casos de prueba [31]. En la Figura 4 se muestra la interfaz de usuario que posee esta herramienta.

Figura 4 Interfaz de usuario en LoadStorm

SOASTA CloudTest

Es una herramienta diseñada para realizar pruebas de rendimiento en producción para aplicaciones web, permitiéndole al usuario simular miles de servicios de infraestructura de nube pública virtual [28]. Actualmente las aplicaciones web suelen seguir prácticas ágiles con compilaciones frecuentes y altas tasas de cambio, por lo cual esta herramienta hace hincapié de manera significativa en el entorno de producción en términos de escala, configuración, perfiles de usuario y entornos de red, logrando un mayor grado de precisión y confianza [30].

Además, esta herramienta se basa en un navegador y también es el principal proveedor de servicios de prueba basados en la nube, donde los nodos de trabajo se pueden distribuir en nubes públicas y privadas para cooperar en una prueba de carga grande [1]. En la Figura 5 se muestra la interfaz de usuario que posee esta herramienta.

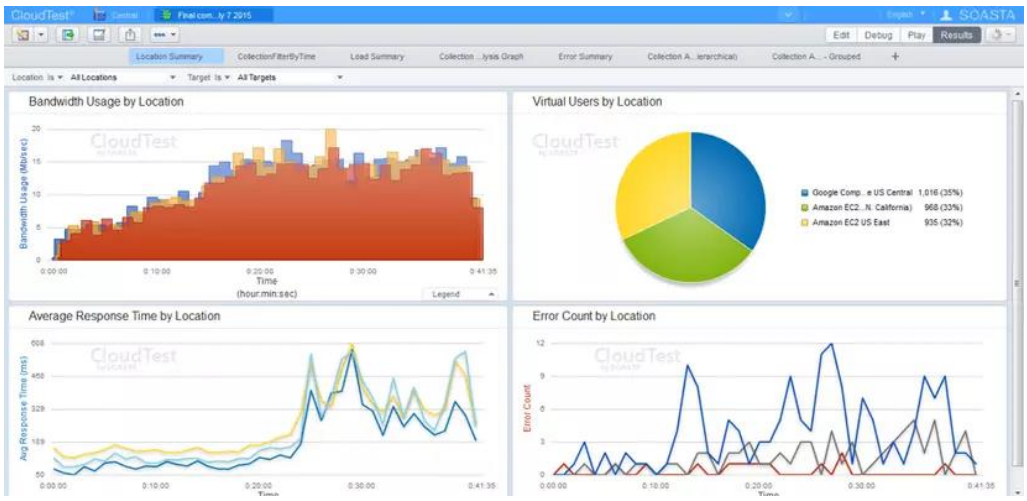

Figura 5 Interfaz de usuario en SOASTA CloudTest

NeoLoad

Siendo desarrollada por Neotys, una empresa francesa y escrita en java, la herramienta NeoLoad se utiliza para medir de manera precisa el rendimiento de aplicaciones web proporcionando seguimiento de los tiempos de respuesta de los usuarios y estadísticas de infraestructura [32].

Además, posee una interfaz gráfica de usuario que permite construir scripts con tan solo arrastrar y soltar, lo que hace que esta herramienta sea de fácil manejo tanto para el desarrollador como para cualquier otro tipo de usuario [33].

Esta herramienta consta de dos componentes, el controlador y generador. El controlador permite al usuario crear, registrar escenarios, ejecutar pruebas y analizar resultados por medio de la interfaz gráfica, mientras que el generador se utiliza básicamente para ejecutar escenarios mediante el envío de solicitudes [34]. En la Figura 6 se muestra la interfaz de usuario que posee esta herramienta.

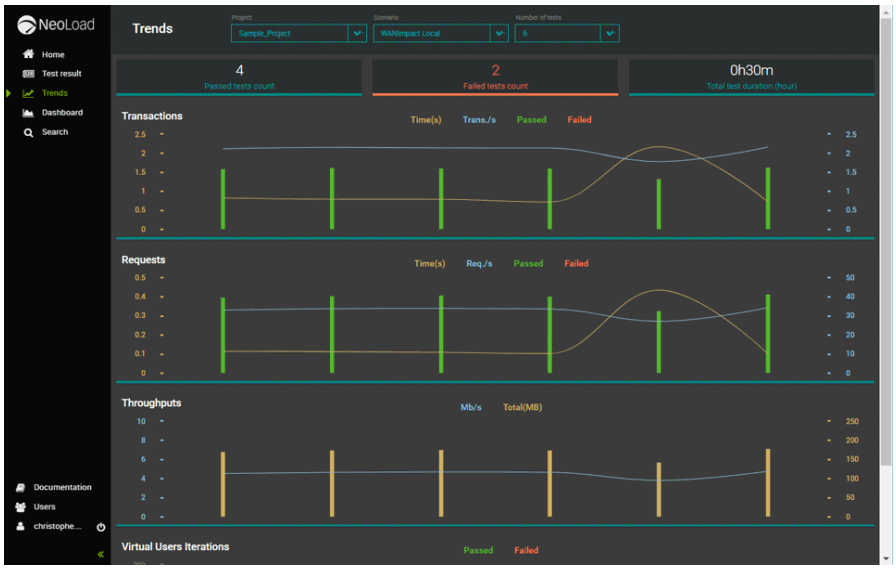

Figura 6 Interfaz de usuario en NeoLoad

Locust

Es una herramienta basada en Python utilizada para evaluar el comportamiento de aplicaciones web bajo múltiples usuarios recurrentes, de manera que se carga un recurso web para calcular el número de usuarios simultáneos que se puedan manejar, generando tráfico. Locust como herramienta es de alto rendimiento para ejecutar pruebas de carga, es de código abierto y gratuito. Además, evalúa sitios web independientes o en vivo y ejecuta pruebas para servidores web, específicamente para servidores web HTTP [35].

Locust es una herramienta basada en eventos. Sin embargo, a diferencia de la mayoría de las aplicaciones basadas en eventos, utiliza la biblioteca gevent4 para proporcionar procesos livianos en lugar del uso de devolución de llamada, lo que permite el soporte de miles de langostas en una sola máquina. Cada usuario de prueba de Locust tiene su propio proceso que permite codificar escenarios de prueba completos sin usar devoluciones de llamada [36].

Entre las características que más destacan en Locust se encuentra:

- ✓ Permite escribir escenarios de prueba usando Python
- ✓ Soporte de cientos de miles de usuarios
- ✓ Interfaz de usuario basada en web

En la Figura 7 se muestra la interfaz de usuario que posee esta herramienta.

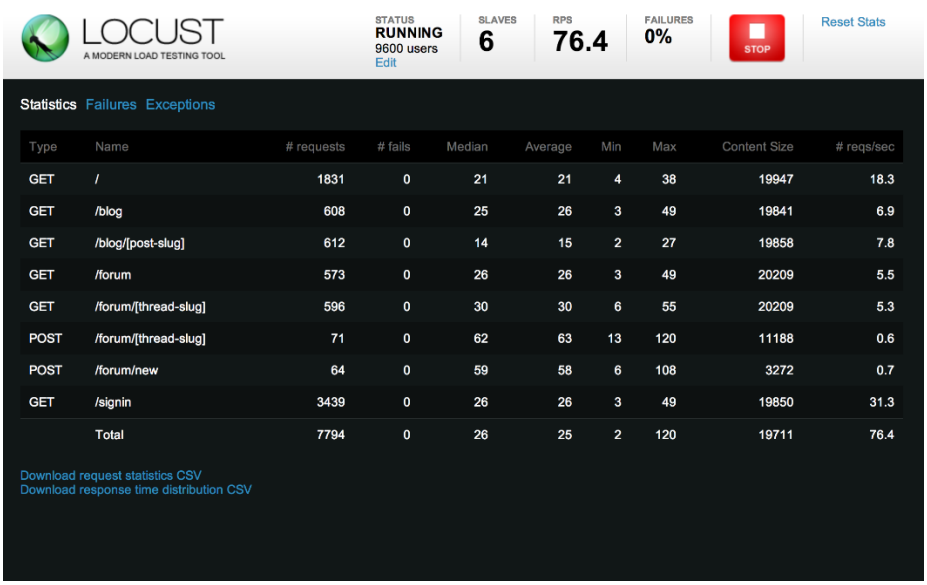

Figura 7 Interfaz de usuario en Locust

Taurus

Es una herramienta de código abierto la cual permite ejecutar pruebas performance con scripts simples, en texto plano en formato yml a través de una capa de abstracción sobre J Meter donde permite separar el script del criterio de aceptación o del escenario de pruebas [37].

Entre las características que más destacan en esta herramienta está:

- ✓ Fácil configuración, actualización y control de versiones

- ✓ Posee DSL (Lenguaje Específico de Dominio) unificado para la definición de escenarios de prueba de carga.
- ✓ Permite definir criterios flexibles para aprobar, reprobar, y si los resultados superan el valor de umbral, puede automáticamente marcar las pruebas como fallidas.
- ✓ Informes en tiempo real.
- ✓ Formato propio de resultados.

En la Figura 8 se muestra la interfaz de usuario que posee esta herramienta.

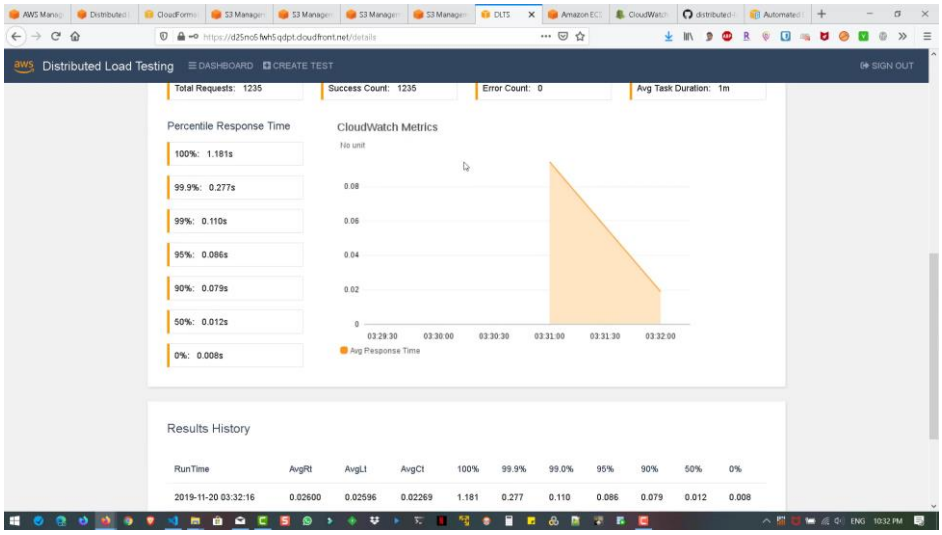

Figura 8 Interfaz de usuario en Taurus

J Meter

Apache J Meter es otra solución de prueba de carga y estrés de código abierto. Esta herramienta está cargado con el IDE completo y rico en funciones para realizar pruebas, depurar y realizar grabaciones de planes de prueba, de modo que se pueda realizar el análisis correcto de aplicaciones web [35].

El objetivo principal de esta herramienta es generar una carga extensa en un servidor o cualquier otro objeto compatible, esto con el fin de evaluar el rendimiento y detectar cuellos de botella en diversas circunstancias que se presenten durante la ejecución de la prueba [36].

Entre las características que más destacan en esta herramienta está:

- ✓ Realiza pruebas de carga y rendimiento usando protocolos HTTP (S), SOAP, REST, JDBC, FTP, etc.
- ✓ Se puede usar en diferentes entornos

- ✓ Es implementado en Java
- ✓ Habilita la simultaneidad mediante el uso de subprocesos y permite separar grupos de subprocesos
- ✓ El diseño cuidadoso de la GUI permite una creación y depuración más rápida del plan de prueba

En la Figura 9 se muestra la interfaz de usuario que posee esta herramienta.

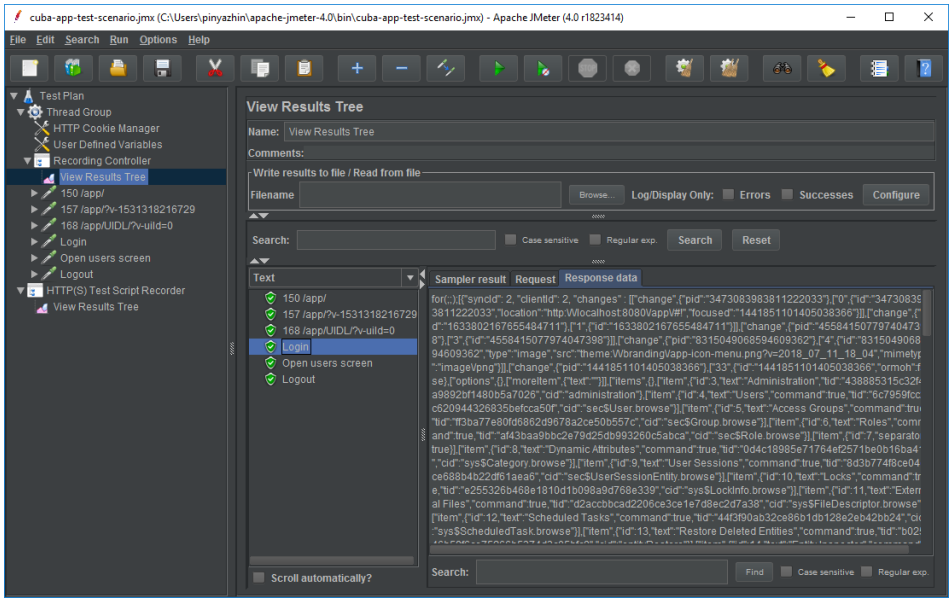

Figura 9 Interfaz de usuario en J Meter

1.1.4 Normativas para evaluar calidad de software

1.1.4.1 Calidad de software

Para evaluar la calidad de un software existen modelos que sirven específicamente para llevar a cabo este debido proceso, por ende, el objetivo de estas normas radica en calificar con los más altos estándares el funcionamiento adecuado de un determinado software. La definición más acertada a este concepto se puede encontrar en la página oficial de la norma ISO 25000, la cual se define como:

La calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. Son precisamente estos requisitos (funcionalidad, rendimiento, seguridad, mantenibilidad, etc.) los que se encuentran representados en el modelo de calidad, el cual categoriza la calidad del producto en características y subcaracterísticas [38], lo cual tiene como resultado obtener un producto de calidad.

1.1.4.2 ISO 25010

Dentro del marco de trabajo que forma la norma ISO 25000 existe una división que se encarga exclusivamente de evaluar el software, llamada ISO 25010. Los modelos de calidad propuestos por esta normativa internacional pueden ser utilizados específicamente para identificar características relevantes asociadas a la calidad de producto software, de manera que pueden ser empleadas para establecer criterios de satisfacción según el caso de estudio lo requiera. En la Figura 10 se visualizan las características y sub características de esta normativa.

Figura 10 Calidad del producto software [38]

Cada característica destaca una de otra, con el objetivo de evaluar un software de la mejor manera, y estas son [38]:

- **Adecuación funcional:** Se basa en la capacidad que posee el producto software para proporcionar funciones que satisfagan al usuario, de manera que, se cerciora que el producto ejecute sus funciones de manera correcta.
- **Eficiencia de desempeño:** Esta característica representa el desempeño relativo a la cantidad de recursos utilizados bajo determinadas condiciones, lo que permite cuantificar el rendimiento del software a fin de obtener información objetiva al respecto.
- **Compatibilidad:** Se refiere a la capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software.
- **Usabilidad**: Se basa en la capacidad del software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones, permitiendo comprobar qué tan interactivo es el software al momento de ser utilizado por un usuario.
- **Fiabilidad:** Es la capacidad de un sistema o componente para desempeñar las funciones especificadas, cuando se usa bajo unas condiciones y periodo de tiempo

determinados, lo que implica evaluar el software sometiéndolo a rigurosos procesos, a fin de corroborar si es benéfica o perjudicial.

- **Seguridad:** Se refiere a la capacidad que posee el sistema para brindar protección a la información almacenada por el usuario.
- **Mantenibilidad:** Se basa en evaluar la capacidad en la que el sistema pueda ser modificado de manera eficiente en base a las necesidades evolutivas, correctivas o perfectivas.
- **Portabilidad:** Se refiere a la capacidad del software o componente de ser transferido de manera correcta de un entorno hardware, software, operacional o de utilización a otro.

1.2. Antecedentes

La investigación del presente estudio se llevó a cabo mediante una exploración de artículos científicos relacionados directamente con el objeto de estudio ("Herramientas de prueba de software basado en la nube"). Para realizar dicha exploración, se tomó como punto de referencia un determinado intervalo de tiempo equivalentes a un lustro, (2015 - 2020). A parte, se usó una específica ecuación avanzada de búsqueda, la cual se presenta a continuación: (cloud) AND (testing). Esta ecuación se aplicó en distintas bases documentales como Scopus, IEEE Xplore y ACM, recuperando así un total 42 estudios científicos, de los cuales cinco de ellos se seleccionaron para la elaboración de los antecedentes.

La primera investigación titula "A methodology for automated penetration testing of cloud applications", presenta una manera de automatizar técnicas de pruebas de penetración, por medio de una metodología que se puede integrar de manera sencilla en un proceso de desarrollo de integración continua, ayudando así a evaluar la seguridad de aplicaciones.

La metodología presentada tiene como objetivo obtener una evaluación de grano grueso de las vulnerabilidades expuestas de una aplicación en la nube mediante una actividad de prueba de penetración automatizada, ejecutada en un entorno de hardware / software virtualizado que reproduce la arquitectura y el comportamiento de la operación real medio ambiente. Todo el sistema que se probará (aplicación y entorno operativo, incluido el sistema operativo, bibliotecas, marcos) se denominará en lo sucesivo Sistema en prueba (SuT) [39].

Esta investigación se relaciona con el tema de estudio debido a que presenta una manera de realizar técnicas de prueba de penetración en la nube, desde la configuración hasta su ejecución.

La segunda investigación presentada lleva por título "A Statistics-Based Performance Testing Methodology for Cloud Applications", se centra en la problemática de las fluctuaciones de rendimiento que experimentan las aplicaciones una vez que son migradas a la nube, teniendo en cuenta los costos de uso de la nube y la naturaleza de caja negra, obtener resultados de rendimiento precisos es extremadamente difícil. A partir de esto, se presenta una nueva metodología de prueba de rendimiento en la nube llamada PT4Cloud, el cual emplea enfoques estadísticos no paramétricos de la teoría de verosimilitud acompañado del método Bootstrap.

PT4Cloud proporciona condiciones de parada confiables para obtener distribuciones de rendimiento altamente precisas con bandas de confianza. Estos enfoques estadísticos también permiten a los usuarios especificar objetivos de precisión intuitivos y cambiar fácilmente entre precisión y costo de prueba. Esta metodología está evaluada con 33 configuraciones de referencia en Amazon Web Services (AWS) y nubes Chameleon [40].

Partiendo de esta segunda investigación, se denota similitud con el presente objeto de estudio, debido a que la metodología que se presenta se implementa en la plataforma de servicios web de Amazon, siendo esta una de las tres plataformas seleccionadas para realizar la presente investigación.

En la tercera investigación estudiada que lleva por título "A DSL-Based Approach for Elasticity Testing of Cloud Systems", se plantea reducir el esfuerzo del evaluador (tester) y los recursos usados en la configuración de la prueba, por medio de un lenguaje específico de dominio (DSL) al momento de ejecutar pruebas de elasticidad dentro de la nube.

Dado que la elasticidad no es algo común, para escribir y ejecutar este tipo de pruebas los evaluadores deben configurar parámetros específicos en la infraestructura de computación en la nube, especificar una secuencia de variaciones de recursos e impulsar sistemas basados en la nube (CBS). A través de esta secuencia el DSL extrae la especificación de casos de prueba de las bibliotecas de diferentes proveedores de nube (Amazon EC2, Google CE y Open Stack), haciendo que sea portátil [41].

La presente investigación tiene relación con el tema de estudio que se lleva a cabo, debido a que proporciona información relevante en cuanto a la ejecución de pruebas de

elasticidad dentro de sistemas basados en la nube. A parte, se hace uso de dos plataformas esenciales dentro de la investigación que se está realizando.

El cuarto artículo científico investigado lleva por título "Dynamic Random Testing Strategy for Test Case Optimization in Cloud Environment". Esta investigación presenta una estrategia de pruebas dinámicas aleatorias (DRT), misma que surge a raíz del problema que representa manejar múltiples tareas de pruebas de software simultáneamente. DRT aplica un mecanismo de retroalimentación a fin de guiar la selección de casos de prueba y esta puede mejorar combinándola en un entorno de nube, priorizando tanto casos de prueba como asignación de recursos.

La plataforma en la nube consta de un servidor de programación y varias máquinas virtuales. Los clientes envían el software bajo prueba (SUT) y el conjunto de pruebas a la plataforma en la nube. El SUT se implementa en cada VM y el conjunto de pruebas se envía al servidor de programación. El servidor de programación selecciona los casos de prueba y los asigna a las máquinas virtuales de acuerdo con la estrategia DRT basada en la nube. La estrategia DRT basada en la nube consta principalmente de tres sub algoritmos: algoritmo de selección de casos de prueba, algoritmo de selección de VM y algoritmo de ajuste del perfil de prueba [42].

La presente investigación analizada comparte mucha relación en cuanto al tema que se está investigando, en especial referente al funcionamiento y estructura de la plataforma en la nube. De igual manera, se hace referencia a las distintas pruebas de software que se realizan durante el proceso de ejecución.

La quinta investigación lleva por título "Cloud Computing and Inter-Clouds - Types, Topologies and Research Issues.", mismo que se centra en enfatizar la gran acogida que actualmente la computación en la nube posee y definir en cierta manera características, tipos, servicios y modelos que este ofrece. Por otra parte, describe la arquitectura Inter-Cloud, la necesidad de Inter-Clouds, diferentes topologías en Inter-Cloud y algunos de los desafíos de investigación en este tipo de arquitectura [43].

La quinta investigación estudiada es de gran utilidad con relación al tema de investigación propuesto. Según lo analizado, existe una gran variedad de conceptos acerca de lo que representa hoy en día la computación en la nube y todo lo que dentro de ella abarca. Cabe recalcar que, a través de este documento se obtuvo una mejor visión o perspectiva con respecto a tipos, modelos, arquitecturas y topologías dentro de la nube, dando así un mejor enfoque a la investigación.

1.3. Fundamentación legal

Generalmente, en una investigación es imprescindible analizar aspectos legales relacionados con el tema central que se está llevando a cabo. Desafortunadamente, el gobierno de Ecuador no consta con ninguna ley que regule servicios de computación en la nube, por lo tanto, solo se tomarán en cuenta únicamente normativas que son allegadas al ámbito informático, como son las siguientes: Ley de Comercio Electrónico, Firmas Electrónicas y Mensajes de Datos. Además, también considerará la Ley de Propiedad Intelectual, el cual vela por los derechos de autor y la Ley Orgánica de Educación Superior.

En la Ley de Propiedad Intelectual en el Libro I, en su Título I.- De los derechos de autor y derechos conexos, Capítulo I.- Del derecho de autor, Sección I Preceptos generales, el Art. 4 menciona que "Se reconocen y garantizan los derechos de los autores y los derechos de los demás titulares sobre sus obras"[44]. En consecuencia, de lo ya mencionado, el estudio investigativo cumplirá estrictamente con todos los parámetros de autoría, respetando los lineamientos a fines del derecho de autor.

Por otra parte, de acuerdo con la Ley Orgánica de Educación Superior en el Título I de Ámbito, objeto, fines y principios del sistema de educación superior, Capítulo 2.- Fines de la educación superior, en el Art. 8.- Fines de la Educación Superior, describe en uno de sus apartados que "Fomentar y ejecutar programas de investigación de carácter científico, tecnológico y pedagógico que coadyuven al mejoramiento y protección del ambiente"[45]. Todo esto a fin de ayudar de manera simultánea, a través de sistemas netamente pedagógicos, fomentando así la investigación tecnológica y colaborar en cierta manera con los ciudadanos.

Por último, conforme a lo que estipula la Ley de Comercio Electrónico, Firmas Electrónicas, y Mensajes de Datos menciona en su sexta disposición que "Se reconoce el derecho de las partes para optar libremente por el uso de tecnología y por el sometimiento a la jurisdicción que acuerden mediante convenio, acuerdo o contrato privado, salvo que la prestación de los servicios electrónicos o uso de estos servicios se realice de forma directa al consumidor"[46]. De esta manera, se da libre acceso a empresas, organizaciones y personas naturales el uso de varias tecnologías, entre ellas se encuentran las plataformas de servicios en la nube.

Capítulo 2

2 MATERIALES Y MÉTODOS

2.1 Delimitación de la investigación

El trabajo investigativo basado en servicios de computación en la nube está centrado específicamente en el uso y manejo de tres proveedores de servicios (Microsoft Azure, AWS y Google Cloud Platform), donde se seleccionó servidores que están localizados en la región de Estados Unidos, teniendo en cuenta que se busca disponibilidad y compatibilidad entre estos tres proveedores.

En cierta parte, esta selección de servidores por región en las plataformas no representa una delimitación espacial**,** debido a que la investigación no se basa específicamente en esa zona, por lo que el manejo de la computación en la nube se presenta como un paradigma global. Por lo tanto, esta investigación no se puede realizar determinando un continente, país o ciudad en específico.

En cuanto a delimitación temporal se refiere, se consideró relevante la recolección de información de carácter científico, que se encuentra dentro de un período de los últimos cinco años (2015 – 2020). Por otra parte, se tiene previsto que para finales del último semestre del presente año 2021 se dé por finalizada esta investigación.

2.2 Tipos de investigación

Dentro del presente tema de investigación se tomó en cuenta ciertas variables de estudio, mismas que son afines a distintas herramientas que fueron usadas durante la ejecución de pruebas de carga o rendimiento mediante un proveedor de servicio en la nube. Por lo tanto, en función de estas variables se estableció que esta investigación es de tipo cualitativa y cuantitativa, lo que la convierte en una investigación de tipo híbrida.

En cuanto a cualitativa se refiere, se puede decir que en esta investigación se analizaron distintas herramientas y proveedores de servicios en la nube, basándose en datos, información y experiencias por parte de autores que en su tiempo pusieron en práctica dichos componentes en distintas situaciones. Además, la investigación es cuantitativa debido a que se experimentó con dichas herramientas, y mediante esto se evaluó parámetros como el nivel de adaptación en la nube, además del tiempo de ejecución y respuesta al momento de realizar pruebas de software.

2.3 Métodos de la investigación

Existe una gran variedad de métodos para llevar a cabo una investigación. Uno de estos métodos es el experimental, el cual se basa en identificar causas y evaluar efectos. Por lo tanto, en esta investigación se aplicó este método, debido a que se implementó una herramienta diferente en cada proveedor de servicio en la nube, con el objetivo de sacar provecho a las herramientas alternativas que ofrece cada proveedor, de modo que se pueda calificar aspectos como la usabilidad, funcionalidad, tiempo de ejecución y respuesta entre otros.

2.4 Población y muestra

Dentro de esta investigación se consideró como población los proveedores de servicios en la nube, y a través de esta se estableció como muestra a los tres proveedores más usados actualmente, que son: Microsoft Azure, Amazon Web Services y Google Cloud Platform.

2.5 Variables e indicadores

Para poder llevar a cabo la investigación se necesitó identificar ciertas variables afines al tema propuesto, las mismas que están en función de las herramientas y proveedores que se usaron para la ejecución de la investigación. Estas variables están basadas en la normativa ISO 25010.

En la Tabla 6 se muestra un total de cuatro variables con su respectivo tipo, y junto con ello los indicadores que lo conforman los cuales instituyen un total de diez.

Tabla 6 Variables e indicadores sujetos a estudio

Concepto	Variables	Indicadores	Tipo de variable
Herramientas de prueba de software basados en la nube	Integración	Compatibilidad Interoperabilidad Alcance	Cualitativa
	Herramientas	Funcionalidad Eficiencia	Cualitativa
	Rendimiento	Tiempo de despliegue Tiempo de ejecución Tiempo de respuesta	Cuantitativa
	Usabilidad	Satisfacción Eficacia Curva de aprendizaje	Cualitativa

Las variables junto con sus indicadores que están listadas en la Tabla 6 son descritas a continuación:

Integración. – Hace referencia al proceso de aplicar herramientas mediante el uso de los distintos proveedores de servicio en la nube.

Indicadores

- Compatibilidad: Representa la facilidad y/o dificultad que presenta la herramienta al integrarse con el proveedor.
- Interoperabilidad: Es la capacidad de comunicación entre los distintos factores que involucran a la herramienta junto con el proveedor.
- Alcance: La capacidad que tiene el proveedor al automatizar procesos mediante el uso de la herramienta.

Herramienta. – Representa la herramienta que se va a emplear durante la ejecución de las pruebas de software.

Indicadores

- Funcionalidad: Es la capacidad que posee la herramienta para funcionar o desenvolverse correctamente en un determinado entorno.
- Eficiencia: Se refiere a la capacidad de la herramienta en función de la ejecución adecuada al cumplir una tarea.

Rendimiento. – Se refiere al desempeño que posee la herramienta durante su uso, es decir, si la herramienta es rápida, lenta, o se encuentra en un punto intermedio.

Indicadores

- Tiempo de despliegue: Representa el tiempo en que la herramienta se conecta con el proveedor.
- Tiempo de ejecución: Representa el tiempo que emplea la herramienta en función a la ejecución de la tarea que se le asignó.
- Tiempo de respuesta: Representa el tiempo en el que la herramienta visualiza el resultado de la tarea que le fue asignada.

Usabilidad. – Se centra en que la herramienta posea una sintaxis fácil de entender y/o aprender, de manera que los usuarios puedan adaptarse.

Indicadores

- Satisfacción: Determina si el usuario siente que la usar la herramienta es fácil, y satisface sus necesidades.

- Eficacia: Determina si la herramienta es rentable usarla, o si consume muchos recursos.
- Curva de aprendizaje: Refleja el manejo de la herramienta por parte del usuario en base a la facilidad de adaptación y aprendizaje.

2.6 Técnicas e instrumentos de recolección de datos

Dentro del ámbito de la investigación existen múltiples técnicas para la recolección de datos. En esta investigación se opta como técnica el experimento, a fin de utilizarla como medio para obtener datos relevantes en cuanto al uso de herramientas de pruebas de software basadas en la nube, midiendo así aspectos como la eficiencia en cuanto a su desempeño, su funcionalidad, usabilidad y la compatibilidad que posee con los proveedores sujetos a estudio.

Por lo tanto, a través de esta técnica se aplica como instrumento el diseño del experimento, obteniendo resultados en función a las variables e indicadores que se aplicaron a la herramienta, a través de los distintos proveedores de servicio en la nube (AWS, Azure, GCP) a medida que se efectuaron las pruebas de software. De esta manera, también se pudo identificar el comportamiento de la herramienta según el proveedor que se usó para su ejecución. La evaluación se realizó mediante una ficha de evaluación que se detalla en el Anexo 1, esta ficha está basada en algunos parámetros que forman parte de la ISO 25010 para determinar la calidad de un determinado software, entre los cuales se consideró a la usabilidad, eficiencia de desempeño y compatibilidad como parámetros claves dentro de la integración de la herramienta en un entorno de computación en la nube.

A partir de este anexo en la Tabla 7 se presenta la escala para medir el grado de importancia con la que se calificó la usabilidad de la herramienta. Las puntuaciones que se utilizaron para realizar la correcta ponderación se establecieron en relación a los parámetros que se seleccionaron, siendo así que, 3 es la calificación alta, 2 la media, y 1 la baja. Para definir el grado de importancia se tomó el puntaje más alto que en este caso es 3 con el número de ítems a evaluar los cuales son 4.

Tabla 7 Intervalo de ponderación de la usabilidad

Intervalo	Significado	Grado de importancia
9-12	Grado de usabilidad alta	Alta
4-8	Grado de usabilidad media	Media
1-3	Grado de usabilidad baja	Baja

Por otra parte, en la Tabla 8 se presentan los parámetros tomados en cuenta al evaluar la herramienta de manera simple y detallada, donde se hizo hincapié en características esenciales relacionadas con el uso, aprendizaje y diseño de la misma como tal, es decir que, se tomó en cuenta la facilidad de adaptación o aprendizaje, la estética, simplicidad, y por último como más importante el nivel de documentación que posee la herramienta para poder hacer uso de ella.

Tabla 8 Parámetros a evaluar en la usabilidad de la herramienta

Marque con una X	Alta	Media	Baja
Capacidad de aprendizaje en base al uso de la herramienta			
Nivel de simplicidad de la herramienta			
Nivel de estética que presenta la herramienta			
Nivel de documentación			

Otro aspecto tomado en cuenta en la investigación al momento de evaluar la herramienta es la eficiencia de desempeño que posee la misma. Este factor es de suma importancia debido a que influyen parámetros como la creación y destrucción de recursos, el acondicionamiento de la herramienta en su entorno y la ejecución de la prueba de carga y estrés dentro del proveedor de servicio en la nube. Cabe recalcar que en este parámetro no se consideró necesario aplicar una tabla de intervalos de ponderación debido a que se tomó como referencia que, mientras menor sea el tiempo de ejecución de tareas, mejor eficiencia de desempeño posee la herramienta. Por otra parte, es necesario aclarar que en algunos casos el tiempo de creación de los recursos es diferente al tiempo completo de la creación, y hay diferencias de entre 1 y 2 segundos o más. En la Tabla 9 se muestra cada parámetro tomado en cuenta para la evaluación.

Tabla 9 Parámetros para evaluar la eficiencia de desempeño de la herramienta

Procesos	**Tiempo**	**Tiempo**
Creación de los recursos		
Acondicionamiento del entorno		
Ejecución de prueba de carga y estrés		
Destrucción del recurso		

Como último aspecto a evaluar dentro de la investigación se consideró la compatibilidad de la herramienta en relación al proveedor al cual está integrado. En la Tabla 10 se presenta la escala para medir el grado de importancia con la que se calificó esta característica, para ello se creó una escala similar a la de la ponderación relacionada con la usabilidad. Las puntuaciones usadas se establecieron en relación a los parámetros establecidos, siendo así que, 3 es el puntaje más alto, luego le sigue 2 que representa a la media, y por último 1 la puntuación más baja. Para definir el grado de importancia se tomó el puntaje más alto que en este caso es 3 con el número de ítems a evaluar los cuales son 2.

Tabla 10 Intervalo de ponderación de compatibilidad

Intervalo	**Significado**	**Grado de importancia**
5-6	Grado de usabilidad alta	Alta
3-4	Grado de usabilidad media	Media
1-2	Grado de usabilidad baja	Baja

Para evaluar la compatibilidad, en la Tabla 11 se contempló como parámetro de evaluación la coexistencia e interoperabilidad entre la herramienta integrada con el proveedor de la nube, de manera que se pudo conocer el nivel de adaptación de la herramienta y a su vez la manera de intercambiar datos para usarlos al momento de cumplir con las tareas.

Tabla 11 Parámetros a evaluar en la compatibilidad de la herramienta

Marque con una X	**Alta**	**Media**	**Baja**
Nivel de coexistencia con el proveedor			
Nivel de interoperabilidad con el proveedor			

2.7 Técnicas de procesamiento y análisis de datos

Para esta investigación se tomó como referencia algunas técnicas para el procesamiento y análisis de datos, entre ellas el análisis de escenario y la visualización de datos son técnicas eficaces para llevar a cabo la presente investigación, tal como se muestra en la Tabla 12.

Tabla 12 Técnicas de procesamiento y análisis de datos

Análisis de patentes y literatura	**No**
Análisis de escenario	Si
Análisis de correlación	No
Análisis basado en estadística descriptiva	No
Simulación de Monte Carlos	No
Experimentos A/B	No
Predicción matemática	No
Visualización de datos	Si

Se escogió como técnica el análisis de escenario, que se basa en la estadística descriptiva, debido a que permite analizar distintos eventos que surgen dentro de un entorno determinado, de manera que facilita analizar detenidamente el comportamiento que adapta la herramienta al efectuar la prueba de software por medio del proveedor de servicio en la nube seleccionado.

Por otra parte, también se escogió la visualización de datos como técnica, debido a que a través de este se pudo obtener gráficas, las mismas que ayudaron a una mejor interpretación de los datos, por ejemplo, los resultados de las ejecuciones de prueba de software mediante el uso de la herramienta seleccionada.

2.8 Normas éticas

La presente investigación se mantiene bajo normativas éticas, las mismas que están ligadas a diversos lineamientos y reglamentos de grados de la PUCESE, en los cuales se fundamenta la autoría, integridad y profesionalismo en base al tema propuesto, de manera que, toda información recopilada durante el transcurso del presente plan de estudio será de carácter confidencial, es decir que, la persona que tendrá acceso a los datos en cuestión será el autor. Por otra parte, se respeta el trabajo de cada uno de los autores que se han nombrado durante el transcurso del presente plan de estudio, incluyendo ideas y conceptos que se usan en esta investigación.

Capítulo 3

3 RESULTADOS

Los resultados obtenidos en esta investigación se presentan de manera secuencial en base a cada objetivo específico planteado, donde se establece qué herramientas y tipo de prueba se usó para llevar a cabo la ejecución de la propuesta, afín de realizar la comparativa de calidad de servicio de las plataformas de computación en la nube a través de las herramientas seleccionadas.

En este capítulo se explican las salidas obtenidas al final de cada ejecución de las pruebas de software a realizar, por medio de las herramientas basadas en la nube que son presentadas en forma de imagen, de modo que se puede visualizar de manera efectiva su implementación y posterior a esto son explicados. Cabe recalcar que en esta investigación se usó recursos básicos en cada proveedor dentro de los estándares necesarios para correr cada herramienta, por ende, cada prueba que se ejecutó está en igualdad de condiciones una de otra.

3.1 Selección de la herramienta de prueba de software basada en la nube

Para la elección de herramientas de prueba de software se consideró usar Locust, Taurus y J Meter, esto a fin de integrarlos dentro de los proveedores GCP, AWS y Azure respectivamente. Cabe mencionar que, se consideró aplicar dichas herramientas dentro de la investigación porque aparte de poder realizar pruebas de carga, estas herramientas tienen también la capacidad de ejecutar pruebas de estrés.

Estas herramientas se seleccionaron en base a un listado de quince herramientas de prueba de carga para 2021 proporcionado por Test Guild [47], de la cual se escogió las tres primeras. En la Figura 11 se puede observar dicho listado.

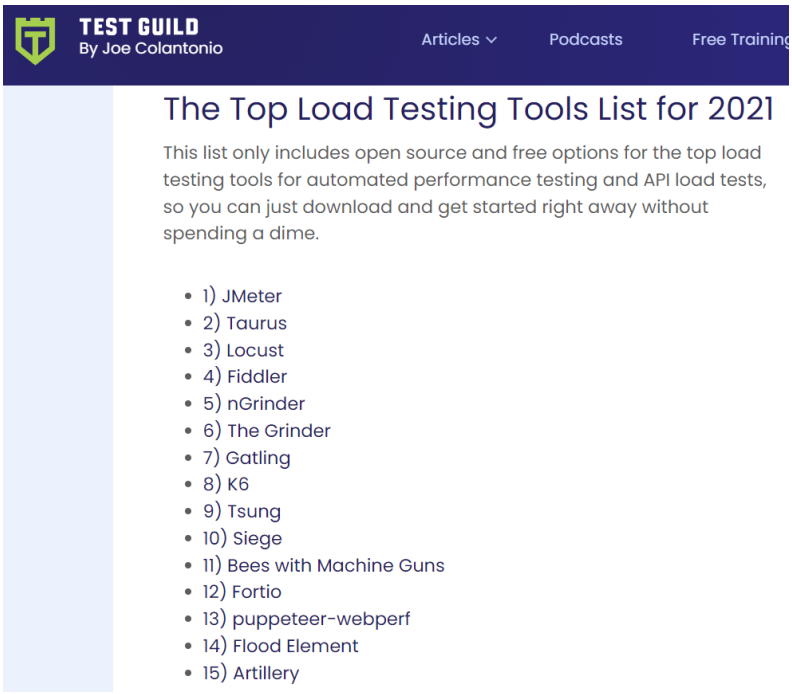

Figura 11 Lista de las principales herramientas de prueba de carga para 2021

A partir de esto, otro aspecto tomado en cuenta al seleccionar y usar estas herramientas dentro de cada entorno de computación en la nube se debe específicamente al nivel de información que se encontró en cada una de estas, teniendo en cuenta que lo que se priorizó en principio es la existencia de documentación relacionada al manejo e integración de herramientas dentro de un ambiente o entorno de servicio en la nube.

En el caso de Google Cloud Platform, este proveedor en su documentación facilita una solución adecuada donde determina el uso de la herramienta Locust como instancia principal para realizar pruebas de carga a través del uso de Kubernetes con el fin de implementar un marco de trabajo que use contenedores múltiples para crear tráfico, permitiendo definir el comportamiento o flujo de usuarios que se desee para cada instancia creada a partir de la herramienta. Además, también brinda la capacidad de monitorear el proceso de enjambre, que en términos generales y/o específicos se refiere a tener una mejor visualización de su proceso a través de generadores de carga de usuario en tiempo real.

Por otro lado, está el caso de Amazon Web Services, donde presenta como solución el uso de la herramienta Taurus para pruebas de carga permitiendo la construcción, ejecución y visualización de pruebas sin necesidad de escribir códigos extensos. Además, con el uso de CloudFormation, este proveedor permite de manera práctica y sencilla la creación de colecciones de recursos como AWS CodeBuild, Fargate, S3, lo que facilita

la integración de la herramienta a través del uso de imagen de contenedores para ejecutar tareas en el clúster que proporciona AWS Fargate.

En cuanto a Microsoft Azure, esta plataforma en principio poseía como servicio una herramienta capaz de realizar pruebas de carga y era con la herramienta que se contaba trabajar, pero desafortunadamente no recibió muy buena acogida por parte de su comunidad, por lo que fue removida como parte de sus servicios este año 2021. En vista de esta remoción la plataforma de servicio en la nube presentó varias alternativas, entre ellas Apache JMeter que fue la herramienta que se seleccionó para trabajar. JMeter se presenta en la cúspide de alternativas en la documentación que ofrece Azure donde en conjunto con Terraform permite el aprovisionamiento y despliegue de una arquitectura capaz de ejecutar pruebas de carga y a través de esta permite la observación y visualización de los resultados de las pruebas que JMeter ejecute.

Teniendo en cuenta eso, cabe recalcar que proveedores como GCP, AWS y Azure actualmente no cuentan con herramientas relacionadas a la ejecución de pruebas de software como tal, sino más bien se encuentran asociados a compañías externas las cuales proporcionan todo tipo de herramientas, incluyendo las que están relacionadas al ámbito de pruebas de software, lo que permite una mejor relación y expansión en cuanto a alternativas para los desarrolladores que se dedican a la actividad de testeo de software. Un claro ejemplo de esto es en el caso de Microsoft Azure, que hace dos años atrás poseía una herramienta capaz de realizar pruebas de carga, pero en vista de no poseer buena acogida o adopción en su comunidad la herramienta terminó siendo dada de baja y optó por asociarse a compañías que proveen este tipo de herramientas.

Otro aspecto tomado en cuenta es la elección de herramientas que posean licencia gratuita y sean de código abierto, de manera que se pueda aprovechar gran potencial de la misma a través de la integración o adaptación de la herramienta dentro de entornos de computación en la nube. Un factor clave en este aspecto es que el usuario no paga por la licencia de uso de dicha herramienta. Por lo tanto, en esta investigación solo se tuvo en cuenta al proveedor de servicio en la nube que cobró únicamente por los servicios internos que prestó, mientras que la herramienta al ser de una compañía externa y siendo de licencia gratuita no generó gastos.

3.2 Escogencia del tipo de prueba de software a aplicar

Para la selección de prueba de software se tomó como referencia a las pruebas de rendimiento. De modo que, a partir de ello se consideró aplicar pruebas de carga y estrés para el proceso práctico de esta investigación. Todo esto utilizando como base un listado de tipos de pruebas de software [48]; considerando que este tipo de pruebas son un medio de control de calidad, de manera que se realiza en aplicaciones o servicios web con el objetivo de asegurarse que todo funcione correctamente, y a su vez poder saber en qué circunstancias podrían fallar dichos sistemas.

Cabe recalcar que a través de la ejecución de pruebas de carga se puede garantizar la calidad en cada iteración del software mediante la manipulación de cierta cantidad de usuarios simultáneos, con el propósito de conocer los límites del mismo. Por otra parte, las pruebas de estrés básicamente evalúan el sistema sometiéndolo a una carga creciente hasta que el sistema o infraestructura colapsa, lo que permite identificar cuellos de botella y conocer la capacidad máxima permitida por el sistema. Teniendo en cuenta estas cualidades, aplicar este tipo de pruebas en la investigación fue de gran importancia debido a que proporcionó parámetros necesarios para conocer la capacidad de cada herramienta en cuenta a su usabilidad, compatibilidad y eficiencia de desempeño.

Otro aspecto tomado en cuenta al momento de escoger estos tipos de pruebas se debe a la documentación existente dentro de cada proveedor de servicio en la nube (Azure, GCP, AWS), esto permitió una mejor experiencia al momento de la implementación de la herramienta basada en la nube, por ende, cada plataforma indicó paso a paso el proceso de integración y ejecución de un ambiente de prueba.

3.3 Integración de la herramienta en entorno de computación en la nube

Locust en Google Cloud Platform

Para implementar Locust dentro de la plataforma de Google se ejecutó de manera secuencial una serie de procesos, de modo que se empezó definiendo variables de entorno para controlar la configuración de la implementación de la herramienta, se creó un clúster de Google Kubernetes Engine (GKE), y por último se manipuló la cantidad de usuarios para observar variaciones de rendimiento. Cabe recalcar que los componentes usados dentro de la plataforma de Google fueron: Google Kubernetes Engine para organizar y administrar los contenedores dentro del marco de trabajo de pruebas, App Engine para

desplegar la aplicación y Cloud Build para hacer uso de la imagen del contenedor Locust Docker.

En la Figura 12 se muestra la carga de trabajo en la que las solicitudes van del cliente a la aplicación. En este caso, Locust distribuye solicitudes a las rutas de destino */login* y */metrics* y estas las modela como un conjunto de tareas.

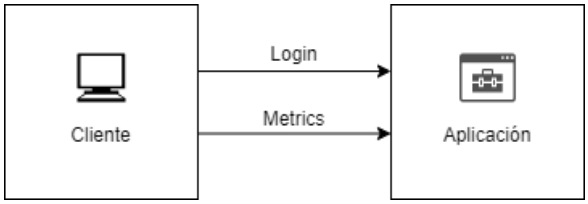

Figura 12 Diseño de la carga de trabajo en GCP

Para la implementación de las tareas para realizar las pruebas se añadió dentro del clúster de kubernetes un máster y también un grupo de workers, con el objetivo de crear una cierta cantidad de tráfico para las pruebas. En la Figura 13 se muestra la relación entre el máster y workers de Locust.

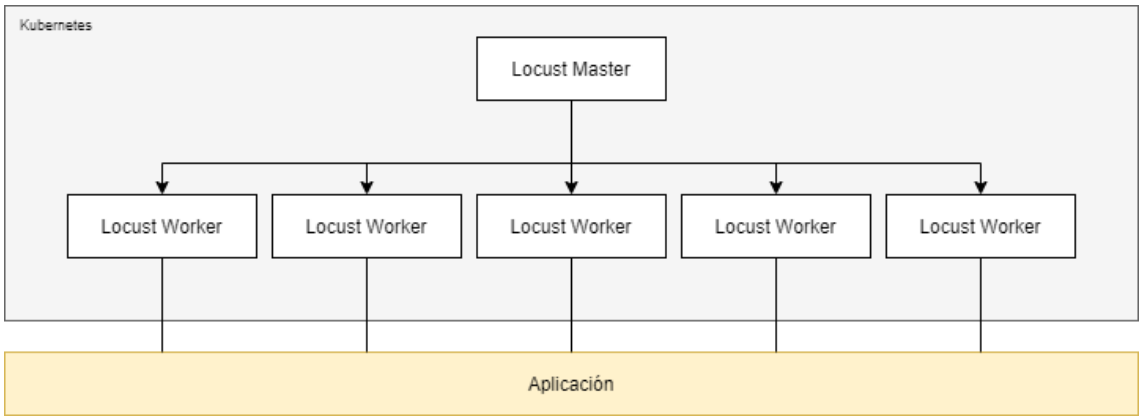

Figura 13 Esquema de relación entre máster y workers de Locust

Luego de haber configurado todo el ambiente dentro de la nube se ejecutó la herramienta Locust para realizar las tareas de prueba de carga y las de estrés en el sistema bajo prueba, para ello se obtuvo la dirección IP externa del sistema de la consola, se abrió el navegador y en la barra de direcciones se colocó lo siguiente: *http://[IP_EXTERNA]:8089*. En la Figura 14 se muestra la interfaz de Locust. Cabe recalcar que el mismo proceso aplica para llevar a cabo las dos pruebas de software antes mencionadas.

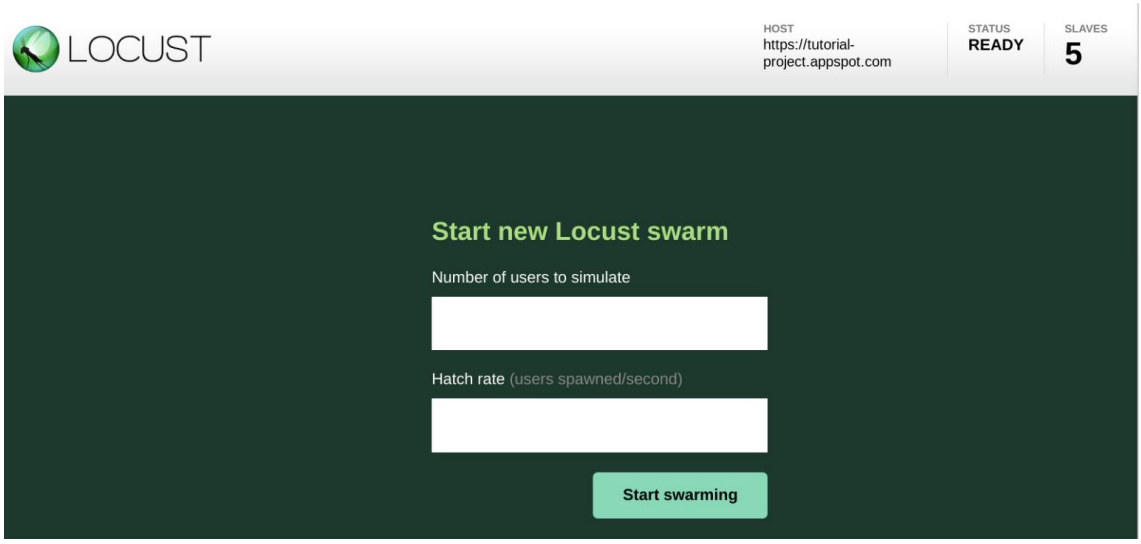

Figura 14 Interfaz de Locust en GCP

Por último, luego de ser ejecutada la herramienta se especificó el número total de usuarios a simular y se estableció la velocidad de surgimiento con la que los usuarios deben generarse por segundo. De esta manera, una vez que las solicitudes comienzan a agruparse, las estadísticas automáticamente se generan mediante las métricas de simulación, un claro ejemplo es la cantidad de solicitudes y solicitudes por segundo como se muestra en la Figura 15, mientras que en la Figura 16 se muestra cómo aumenta el número de peticiones para forzar la infraestructura que se está probando.

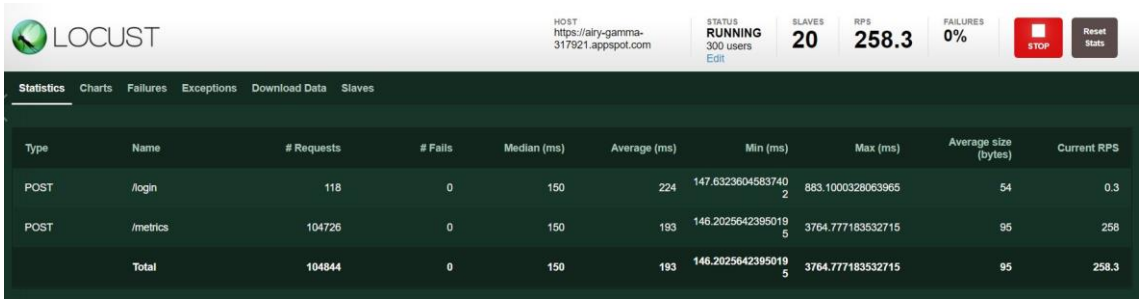

Figura 15 Estadísticas de simulación Locust

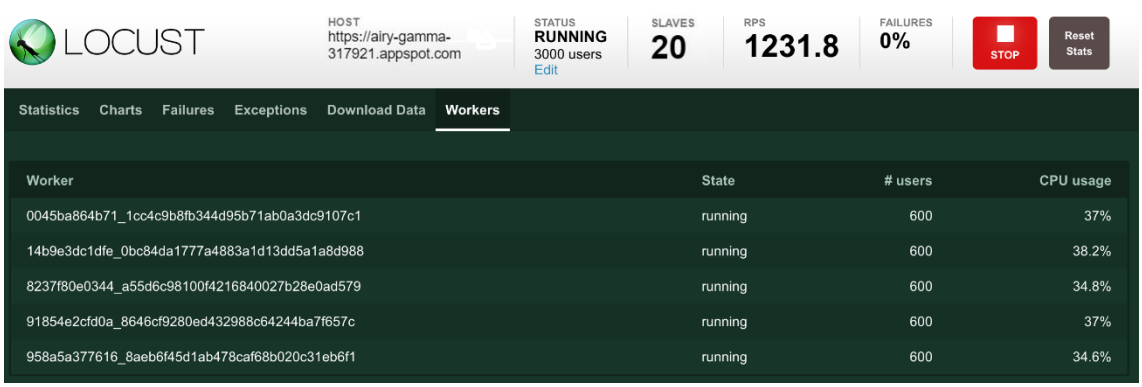

Figura 16 Aumento de rps para prueba de estrés en Locust

Taurus en Amazon Web Services

La implementación de Taurus en la nube de Amazon resultó un poco más compleja que la ejecución de la herramienta anterior, esto debido a que tomó tiempo relacionarse y entender previamente la estructura, el funcionamiento y componentes de la plataforma; esto al principio provocó confusión, por ende, la ejecución de las pruebas de software tomó un poco más del tiempo estimado.

El proceso de implementación de esta herramienta se efectuó bajo Amazon ECS en contenedores de AWS Fargate, donde se integró Taurus a través de contenedores Docker y a su vez fueron implementados en clústeres de Fargate, que se ejecutaron en diferentes regiones de AWS con el fin de simular solicitudes provenientes de diferentes ubicaciones geográficas del servicio de la nube. Cabe recalcar que los principales servicios usados dentro de la plataforma de Amazon fueron: AWS Fargate, AWS Lambda, Amazon DynamoDB y AWS Step Functions.

La carga de trabajo que se aplicó es similar a la de GCP, donde las solicitudes van desde el cliente hacia la aplicación. En este caso en AWS se manejó una arquitectura un poco más compleja debido a que usó varios recursos y/o servicios por medio del uso de AWS CloudFormation para la colección, aprovisionamiento y administración de recursos dentro de la plataforma.

En la Figura 17 se visualiza la arquitectura y a breves rasgos el proceso de implementación de prueba de software en AWS, el cual muestra la implementación de la herramienta Taurus junto con la solución que proporcionó la documentación de la plataforma con los parámetros predeterminados que sirvieron para la creación del entorno en la plataforma.

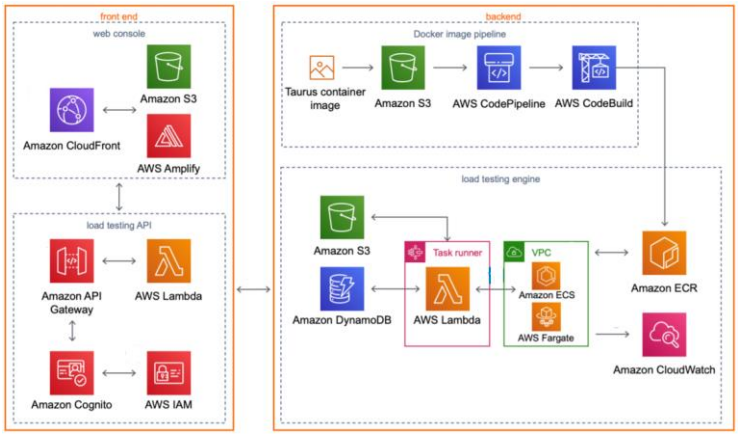

Figura 17 Arquitectura para las pruebas de software en AWS [49]

La solución que proporcionó AWS para realizar dichas pruebas en su plataforma cuenta con dos componentes, un Frontend que aloja una API para la prueba de carga y estrés, junto con una consola web para interactuar con los datos de prueba de forma segura, y consta también de un Backend para la canalización de contenedores y de la imagen Docker de Taurus y la simulación de usuarios que generan un número selecto de solicitudes por segundo por medio de un motor para la ejecución de dichas pruebas.

En la Figura 18 se muestra la interfaz de la herramienta Taurus una vez ejecutada luego de haber configurado todo el ambiente dentro de la nube, mientras que en la Figura 19 se visualizan los parámetros o datos que se ingresaron para realizar la prueba. Cabe recalcar que el mismo proceso se realiza para la ejecución de las dos pruebas llevadas a cabo.

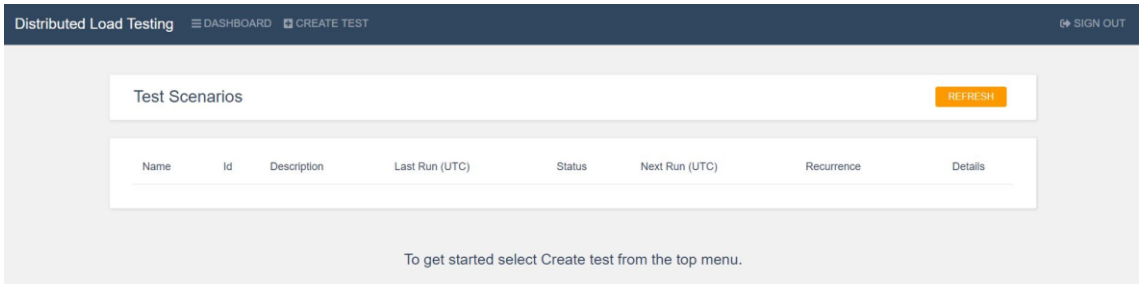

Figura 18 Interfaz de Taurus en AWS

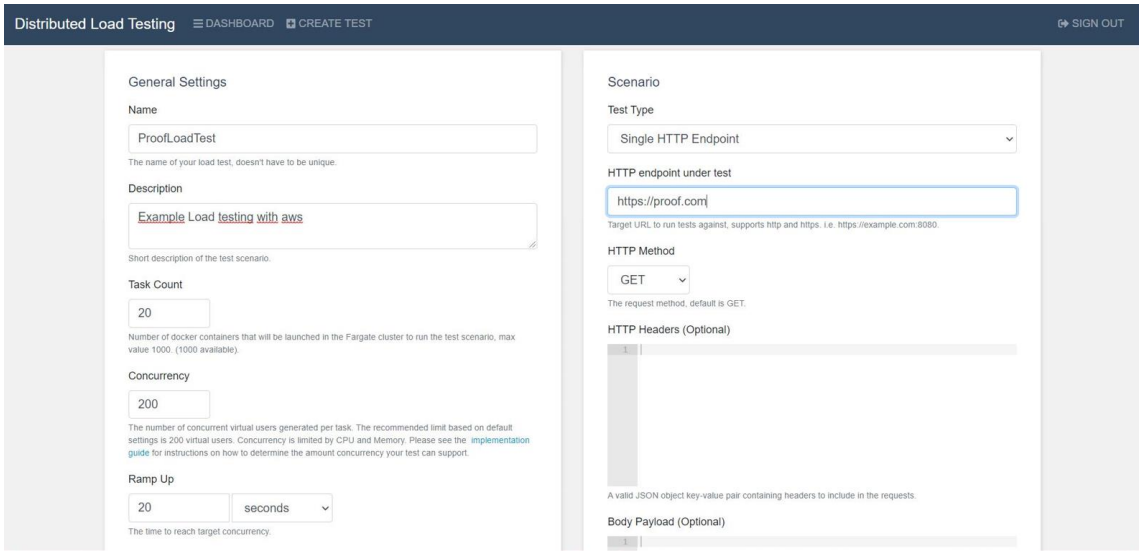

Figura 19 Parametrización de datos para las pruebas en AWS

Luego de ingresar los datos requeridos por la herramienta, esta procede a realizar dichas pruebas, de manera que las solicitudes comienzan a agruparse, y a medida que se procesan estas solicitudes las estadísticas automáticamente se generan mediante las métricas de simulación, un claro ejemplo es el tiempo promedio de respuesta de solicitudes, el tiempo

medio de conexión y la cantidad solicitudes por segundo que pueden variar según como se requiera, tal como se muestra en la Figura 20.

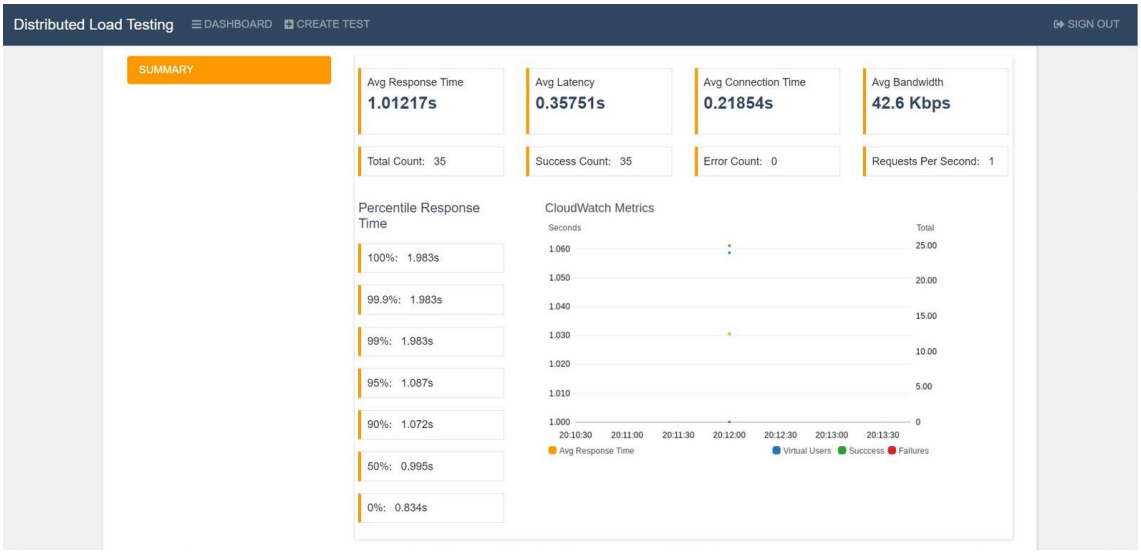

Figura 20 Resumen de prueba de carga en AWS

J Meter en Microsoft Azure

El querer aplicar pruebas de carga y estrés a través de una herramienta en Azure fue complicado en cuestiones de búsqueda, esto debido que Microsoft actualmente no cuenta con una amplia documentación en relación a ciertos servicios que ofrece, por ejemplo, la aplicación de herramientas basadas en la nube para pruebas de software. Esta falta de documentación se le atribuye a Microsoft Azure que recientemente cambió su política para permitir el nivel gratuito de agentes alojados para proyectos públicos y privados de organizaciones DevOps recién creadas, incluyendo también la eliminación definitiva del servicio de pruebas de carga basado en la nube que se alojaba en Azure DevOps.

Este fin de vida útil del servicio de pruebas disminuyó bastantes medios o alternativas al momento de querer buscar una solución, por ejemplo, no se podía ejecutar tareas de pruebas de carga mediante Azure Pipelines, CI /DC Pipelines e inclusive por medio de Azure Portal que ofrecía una opción de prueba de carga en App Services y Application Insights.

Una de las pocas posibles soluciones que se encontró en la documentación fue trabajar con herramientas de prueba de carga externas a Azure Marketplace, entre estas estaba Apache J Meter, que como se conoce sirve para analizar y medir el rendimiento de una variedad de servicios, con énfasis en aplicaciones web.

De modo que para integrar J Meter dentro de Azure se implementó en la nube una canalización para prueba de carga y una para prueba de estrés, para ello se ejecutó de manera secuencial una serie de tareas, de modo que se creó el recurso a petición, se implementó la infraestructura, se ejecutó las pruebas, se visualizó los resultados, y finalmente se procedió a la destrucción de la infraestructura a petición. La implementación en esta plataforma usó Apache J Meter y Terraform, de manera que, J Meter funcionó dentro de la arquitectura como herramienta de prueba y Terraform sirvió para aprovisionar y destruir de manera dinámica la infraestructura requerida en Azure. En la Figura 21 se visualiza la arquitectura con la que se efectuó esta solución.

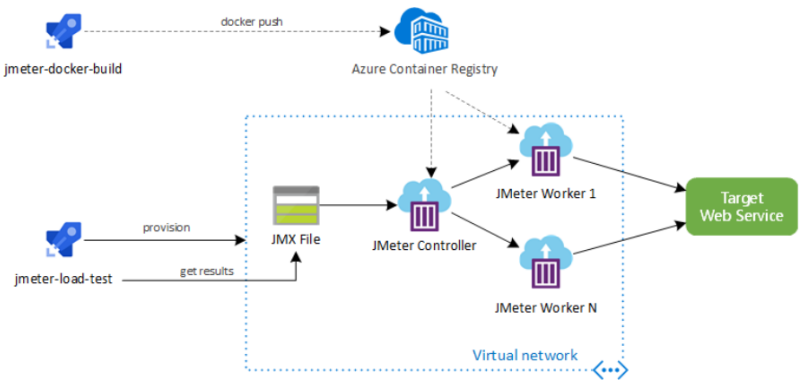

Figura 21 Arquitectura para las pruebas de software en Azure

Esta estructura proporcionada por Microsoft Azure está diseñada para implementar tanto la prueba de carga como la prueba de estrés en dos canalizaciones de Azure Pipelines. De manera que, una canalización (J Meter Worker 1) crea un contenedor Docker de J Meter e incorpora la imagen de Azure Container Registry (ACR) de modo que permita compilar, almacenar y administrar contenedores. La otra canalización (J Meter Worker N) se encarga de validar dichas pruebas por medio de un archivo de extensión jmx, que aprovisiona dinámicamente su infraestructura, ejecuta las pruebas, visualiza los resultados y finalmente destruye la infraestructura.

Para la ejecución de la canalización de J Meter se debe hacerlo por medio de líneas de comandos, de manera que se elige el archivo J Meter que se desea ejecutar y el número de trabajadores que se requiera para dichas pruebas. En la Figura 22 se muestra el comando para ejecutar la canalización de la herramienta.

```
JMETER_JMX_FILE=sample.jmx
JMETER_WORKERS_COUNT=1
az pipelines run --name $PIPELINE_NAME_JMETER \ --variables TF_VAR_JMETER_JMX_FILE=$JMETER_JMX_FILE TF_VAR_JMETER_WORKERS_COUNT=$JMETER_WORKERS_COUNT
```

Figura 22 Comando para ejecutar canalización de JMeter en Azure

Una vez ejecutada la herramienta los resultados de la prueba se crean en un archivo JTL con formato CSV. Para ello, la solución proporciona un script de Python para convertir JTL a formato JUnit de manera que, durante la canalización se pueda tener una visualización de prueba de Azure DevOps. En la Figura 23 y Figura 24 se muestra su interfaz una vez ejecutada luego de haber configurado todo el ambiente dentro de la nube.

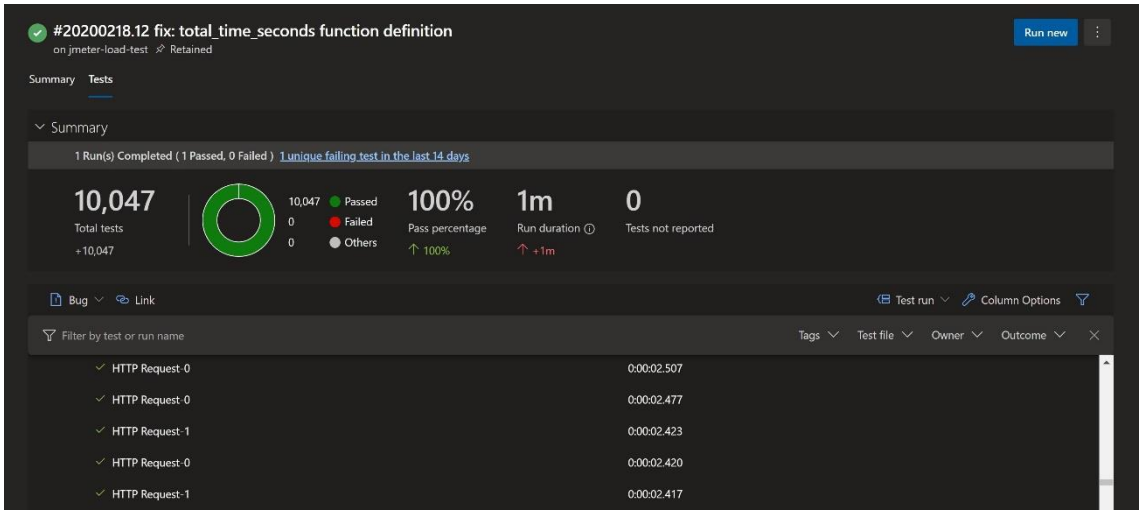

Figura 23 Interfaz y visualización del resultado de prueba con JMeter y Azure

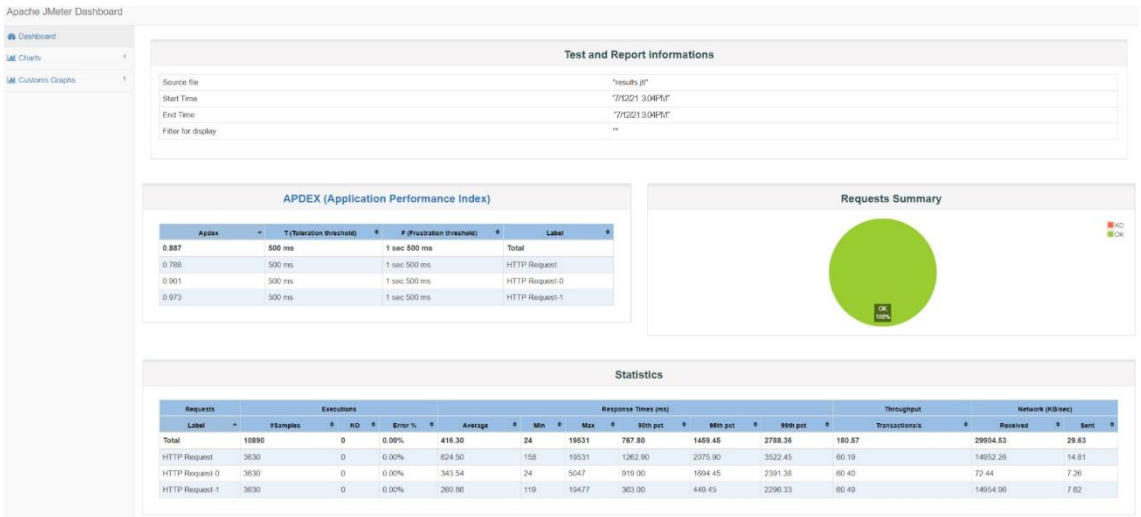

Figura 24 Panel de reportes de J Meter en Azure

3.4 Comparación de calidad entre herramientas

La calidad no es algo que se pueda agregar a un determinado software después de desarrollarlo o integrarlo a nuevos entornos, muchas de las veces se tiene la idea de que el software de calidad es aquel que brinda lo que se necesita con una adecuada velocidad de procesamiento, pero es mucho más que eso. Tiene que ver con la usabilidad, eficiencia de desempeño y compatibilidad, siendo estos estándares objetos de pruebas que determinen el grado de calidad de un software, o en este caso de la herramienta como tal.

Usabilidad

Para evaluar esta característica se contempló como parámetro la capacidad de aprendizaje con base en el uso de la herramienta evaluada, debido a que ésta tiene su propio funcionamiento y cada plataforma se maneja con una configuración distinta. Otro aspecto contemplado es el nivel de simplicidad y estética, donde se buscó conocer la complejidad y qué tan llamativa se presenta al usuario. Como último parámetro se evaluó el nivel de documentación existente, considerando que una herramienta que no posee buena documentación es inútil, dificultando el aprendizaje y entorpeciendo el nivel de adaptación de ésta.

Cada puntaje obtenido en base a las herramientas que fueron integradas dentro de los proveedores se visualiza en el Gráfico 1, donde J Meter alcanzó un puntaje de 6 siendo integrado en la nube de Microsoft Azure, lo que sitúa a esta herramienta en un grado medio de usabilidad, luego le sigue la herramienta Taurus la cual está integrada en AWS obteniendo un puntaje de 9 lo cual sitúa a esta herramienta en un grado alto de usabilidad junto con Locust siendo integrado en la nube de Google con un puntaje de 10.

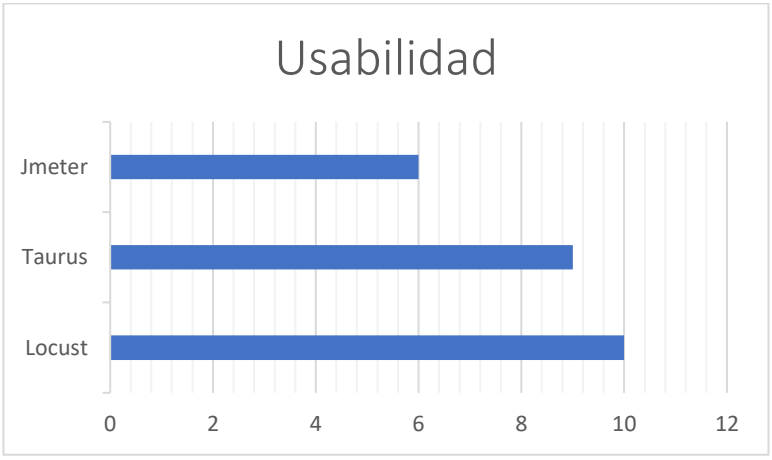

Gráfico 1 Puntajes obtenidos en cuanto a usabilidad de herramientas

Analizando este gráfico es notable que la herramienta Locust es mucho más sencilla de usar, esto se debe a que la sintaxis que se maneja dentro del proveedor GCP es mucho más simple, y las configuraciones requeridas dentro del mismo son relativamente sencillas de aplicar dentro del entorno. Además, un aspecto importante que destaca en la plataforma de Google es la documentación, este proveedor tiene a disposición muy buena información, bastante entendible para cualquier persona que esté incursionando en este nuevo paradigma de la computación.

En cuanto a nivel de estética y simplicidad cabe destacar que la interfaz de Taurus integrada en el proveedor de AWS está por encima, es mucho más amigable en comparación a las demás. Los colores que usa en su interfaz son muy dinámicos y son agradables a la vista, de igual forma, la manera de presentar las estadísticas es muy amigables y por ende permite una buena comprensión de la misma.

Eficiencia de Desempeño

Otro aspecto que se tomó en cuenta en la investigación fue evaluar la eficiencia de desempeño que posee cada herramienta de prueba de software. Esta característica es un aspecto importante, debido que a través del mismo se obtuvo el tiempo (en segundos) que tomó cada herramienta en ejecutar sus respectivas tareas, teniendo en cuenta que lo que se buscaba en sí de la herramienta al momento de usarla es que su proceso de ejecución sea veloz o ágil al momento de realizar alguna acción, teniendo en cuenta que a menor tiempo de ejecución de tareas mejor es la eficiencia de desempeño de la herramienta.

En el Gráfico 2, se visualiza los resultados obtenidos de las herramientas integradas en cada proveedor, en base a cada ítem que se tomó en cuenta al evaluar la herramienta integrada a la plataforma.

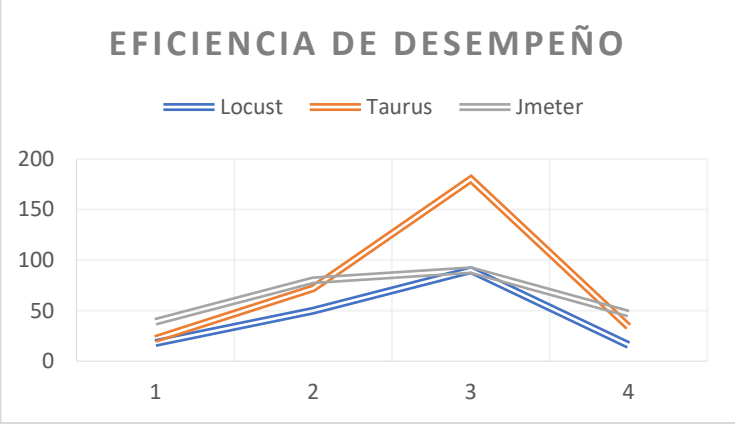

Gráfico 2 Resultados basados en el desempeño de herramientas

A partir de los resultados basados en el desempeño de cada herramienta, Locust integrada en GCP vuelve y se sitúa como la mejor al haber obtenido mejores tiempos, en relación a AWS y Azure en cuanto a creación de recursos, acondicionamiento del entorno, ejecución de prueba y por último la destrucción de recursos. Por otra parte, evidentemente se puede observar en cuanto a eficiencia de desempeño, que la herramienta que tuvo más complicaciones fue Taurus, específicamente al momento donde se ejecutó la prueba de software, y esto se debe mayormente porque la plataforma a la que fue integrada (AWS) hace uso de varios recursos para poder ambientar una correcta arquitectura de prueba de carga y estrés, en comparación con los proveedores Azure y GCP. Por otra parte, J Meter integrado en la plataforma de Microsoft Azure se mantiene en medio, lo que demuestra que su desempeño no es lo suficientemente bueno comparado con Locust, pero si es relativamente mejor que Taurus en AWS por lo que le tomó menos tiempo en ejecutar dichas pruebas.

Compatibilidad

Los puntajes obtenidos en base a las herramientas que fueron integradas dentro de los proveedores se visualizan en el Gráfico 3, donde Locust demuestra que posee una mejor coexistencia con el proveedor de GCP y a su vez es capaz de intercambiar y manejar datos de manera sencilla, lo que le otorga un nivel de interoperabilidad superior a los otros dos proveedores. De esta manera Locust alcanzó un puntaje de 6, mientras que Taurus obtuvo como puntaje 5, y finalmente J Meter consiguió como calificación un 4 en cuanto a compatibilidad entre herramienta y proveedor de servicio en la nube.

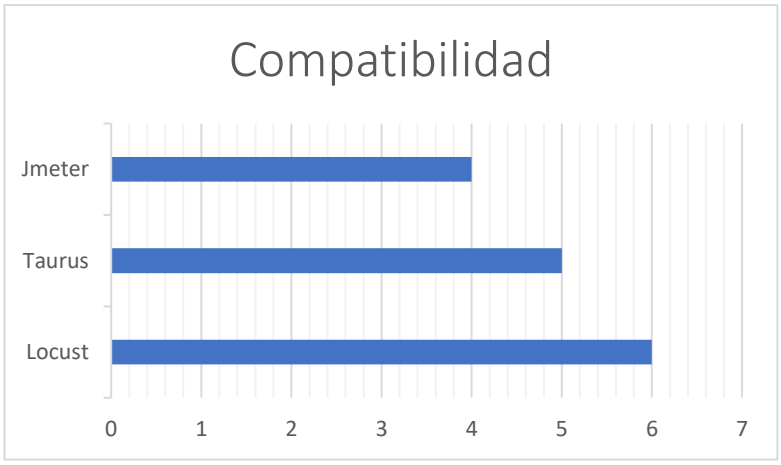

Gráfico 3 Puntajes obtenidos en cuanto a compatibilidad de herramientas

A partir de esta comparativa, la ejecución de dichas pruebas permitió conocer y calificar aspectos como la usabilidad, eficiencia de desempeño y compatibilidad que ofrecen estas herramientas en base al proveedor donde fueron integradas, de modo que se demostró y se dio por sentado que Locust en comparación a Taurus y J Meter ofrece al usuario un entorno sencillo y ágil de utilizar. Esto en principio se debe a la sintaxis que Google Cloud Platform maneja, siendo mucho más simple de gestionar, y a su vez permitir aplicar configuraciones requeridas dentro del entorno.

Capítulo 4

4 DISCUSIÓN

En la investigación de Saunders *et al.*[40] menciona de manera general que los bajos costos de propiedad y la elasticidad de los recursos han llevado a muchas organizaciones a cambiar sus aplicaciones a la nube, pero consecuente a esto presenta una problemática en cuando a las fluctuaciones de rendimiento que experimentan las aplicaciones una vez que son migradas y puestas a prueba, siendo esto una gran verdad debido a que si se ocupa una aplicación con un gran flujo de trabajo este de por si requerirá obtener mayor cantidad de recursos por parte del proveedor que se plantee usar para la migración. Es por esto que es fundamental que el desarrollador o evaluador tenga un conocimiento previo y preciso del rendimiento del software, aplicación o infraestructura que se requiera evaluar porque a partir de esto dependerán las configuraciones de máquina virtual dentro del proveedor de servicio en la nube y por ende el costo por uso de la misma.

No obstante, los autores de la investigación aplican una metodología llamada PT4Cloud, que se basa en el empleo de herramientas estadísticas no paramétricas, de teoría de probabilidad y arranque, la cual se aplicó en las nubes Chameleon y AWS, ayudando a proporcionar condiciones de paradas fiables para obtener resultados de rendimiento precisos. De modo que, teniendo en cuenta esta información, se puede entender dentro de la investigación el por qué resultó un poco tardía la respuesta de AWS al momento de ejecutar pruebas de carga y estrés; esto se debe principalmente a que dicho proveedor utiliza gran cantidad de recursos para poder crear una infraestructura y consecuente a esto ejecutar una prueba de software.

Por otra parte, es necesario tener en cuenta conceptos como IaaS, PaaS y SaaS los cuales son cada vez más usados en la actualidad, esto se debe a que van ligados a la disponibilidad de ofrecer gran variedad de servicios, y es en [43] donde enmarca el uso de proveedores de servicio en la nube de modo que proporcione al desarrollador un entorno donde los requisitos como potencia informática, almacenamiento de datos, memoria, software o herramientas estén a disposición del mismo. Los autores de esta investigación de manera general centran su estudio en los modelos de servicio, los diferentes tipos de nubes existentes y la necesidad de integrar Inter-Cloud dentro de empresas dedicadas al desarrollo en la nube. Con esto se puede apreciar al igual en esta investigación que es necesario conocer a cabalidad este tipo de conceptos, para que a

partir de aquello se pueda tener una idea de cómo funcionan los proveedores de servicio en la nube que se plantean utilizar, los requisitos que abarca el mismo, sus ventajas, desventajas, y más que todo definir si será factible el uso de dichos proveedores en base a las herramientas que se plantean integrar. De esta manera se puede apreciar la gran ventaja que genera el uso de plataformas basadas en la nube, y a su vez la aplicación de herramientas como Locust debido que como herramienta para pruebas de software obtuvo un mayor control dentro del proveedor CGP, a diferencia de Taurus y J Meter integrados en AWS y Azure respectivamente; por lo tanto, ambas investigaciones comparten ideas y resultados.

Es importante recordar que los evaluadores deben ser precisos al momento de ejecutar pruebas de software, y es por ello por lo que Amine Benelallam *et al.*[41] profundiza en el saber configurar parámetros necesarios dentro de los proveedores de servicio en la nube, donde se definen las secuencias de variaciones de recursos, casos de pruebas e infraestructuras; en este caso los autores de la investigación proponen un enfoque basado en un lenguaje especifico de dominio que ayuda a reducir el esfuerzo del evaluador al escribir y ejecutar pruebas de software, lo cual resulta interesante aplicar. De modo que se concuerda con las propuestas presentadas en [41] para realizar una correcta prueba de software.

Capítulo 5

5 CONCLUSIONES

Mediante esta investigación se comprendió el funcionamiento y rendimiento que ofrecen herramientas como J Meter, Locust y Taurus por medio de la integración de pruebas de carga de software mediante un determinado proveedor de servicio en la nube, que en este caso se usó AWS, GCP y Azure, lo que permitió llevar procesos relacionados con la creación y gestión de pruebas de manera eficiente.

En base al primer objetivo del presente proyecto se concluye que, a partir de la selección se pudo comprender la importancia que tiene el documentarse previamente antes de elegir y/o querer integrar una determinada herramienta para realizar una prueba de software dentro de cualquier flujo de trabajo. Cabe mencionar que en la mayoría de los casos la documentación fue lo suficientemente buena en cuanto a herramienta y proveedor de servicio en la nube, por lo que no presentó complicaciones, a excepción de la documentación que ofrece Azure que es considerablemente desactualizada, lo cual dificultó el aprendizaje.

De acuerdo al segundo objetivo, si bien es cierto la ejecución de pruebas de software es una de las actividades más importantes y fundamentales dentro del desarrollo de un proyecto, esto porque ayudan verificando la funcionalidad de cualquier sistema o aplicación. En este caso se aplicó de manera ejemplificada una prueba de carga y estrés lo cual permitió conocer aspectos como la usabilidad, eficiencia y compatibilidad que poseen las herramientas presentadas frente a un entorno basado en la nube, donde se demostró que Locust en comparación a Taurus y J Meter ofrece un entorno mucho más sencillo de usar y esto en principio también se debe a la sintaxis que se maneja en GCP, por lo que es mucho más simple, y las configuraciones requeridas del mismo son relativamente sencillas de aplicar dentro del entorno.

A partir del tercer objetivo, se concluyó que la integración de este tipo de herramientas dentro de la nube es de gran utilidad, debido a que ayudan o facilitan la ejecución de pruebas de software antes de llevar un sistema o aplicativo web a producción; además, esta práctica en el mundo empresarial permite que empresas que se dedican al desarrollo de software obtengan importantes ahorros de costes, externalizando su infraestructura, plataformas y servicios de proveedores por medio de lo que se conoce hoy en día como computación en la nube. Por ende, organizaciones de todos los tamaños están migrando

sus soluciones, lo que hace necesario este concepto para validar cómo sus aplicaciones funcionan en distintas condiciones de uso, incluyendo su usabilidad, eficiencia de desempeño y compatibilidad.

Por último, en cuanto lo abordado en el cuarto objetivo, comparar la calidad de servicio de las plataformas de computación en la nube a través de las herramientas seleccionadas, permitió obtener una mejor y/o mayor percepción del concepto calidad como tal. Características como la usabilidad, eficiencia de desempeño y compatibilidad son fundamentales al momento de determinar si el uso de una herramienta puede beneficiar o no, la práctica que se está llevando a cabo dentro de un proveedor. En este caso, cada herramienta que se aplicó dentro de cada entorno de servicio en la nube obtuvo distintas reacciones, por lo cual, se da por sentado que no toda herramienta puede ser beneficiosa para el desarrollador al momento de querer aplicarla dentro de un ecosistema de servicio en la nube.

6 RECOMENDACIONES

Una vez concluido el presente trabajo de investigación se recomienda ampliamente el uso de proveedores de servicio en la nube debido a su apogeo en el mundo empresarial; plataformas como Google Cloud Platform, Amazon Web Services o Microsoft Azure, son indispensables dentro de una organización debido al sin número de servicios que ofrecen y su compatibilidad con diversas herramientas que ayudan y facilitan la creación, integración y manejo de recursos dentro de la nube. Considerando que, si se pretende replicar el experimento de esta investigación, se priorice el uso de GCP, el cual fue el proveedor que obtuvo mejores resultados en cuanto a integración y ejecución de herramientas de pruebas de software.

Por otra parte, en cuanto a herramientas de prueba de software, se recomienda documentarse previamente antes de seleccionar una de estas, y posterior a esto inclinarse a la que más se ajuste a las necesidades y/o gustos del desarrollador. En este caso, cada herramienta seleccionada brindó una experiencia distinta en cuanto a creación de pruebas de rendimiento (carga y estrés), por lo cual, si se desea codificar se recomienda el uso de Locust o Taurus y si se desea crear pruebas por medio de una interfaz de usuario lo más recomendable es el uso de J Meter.

Cabe recalcar que las herramientas que se integraron en la nube para esta investigación son de código abierto, por ende, hay que tener en cuenta que el uso de este tipo de

herramientas tiene sus pros y sus contras, por lo cual se recomienda tener presente factores como la precisión, facilidad de uso, compatibilidad, facilidad de scripting, informes, monitoreos, costos, etc. antes de elegir uno. Por ejemplo, durante la investigación con J Meter se tuvo complicaciones en cuanto a generador de informes, porque esta herramienta no tiene ninguna característica integrada relacionado a aquello, por lo que se tuvo que añadir una solución externa basada en scripts para poder visualizar datos. Adicional a esto, también este tipo de herramientas al ser de código abierto tienen un soporte limitado, por lo cual, si se enfrenta a algún desafío en la implementación o el mantenimiento, requerirá mayor esfuerzo y cantidad de recursos.

Por último, también se recomienda el no quedarse únicamente con el uso de estas tres herramientas presentadas, sino más bien seguir adaptándose a las nuevas que vayan surgiendo con el tiempo, bien se sabe que la industria tecnológica avanza de manera rápida, por lo cual es necesario estar actualizado de las nuevas herramientas que se presenten. Se considera esta última recomendación como la más importante, debido a que como desarrollador es necesario aprender a adaptarse a nuevas herramientas y/o tecnologías, caso contrario no podrá destacar en la vida laboral como tal.

REFERENCIAS BIBLIOGRÁFICAS

[1] V. Mittal, L. Nautiyal, and M. Mittal, “Cloud testing-the future of contemporary software testing,” *Proc. - 2017 Int. Conf. Next Gener. Comput. Inf. Syst. ICNGCIS 2017*, pp. 142–146, 2018, doi: 10.1109/ICNGCIS.2017.11.

[2] C. Hung Kao, “Testing and evaluation methods for cloud environments: A review,” *ACM Int. Conf. Proceeding Ser.*, vol. Part F1296, pp. 56–60, 2017, doi: 10.1145/3108421.3108435.

[3] M. Kaur, “Testing in the cloud: New challenges,” *Proceeding - IEEE Int. Conf. Comput. Commun. Autom. ICCCA 2016*, pp. 742–746, 2017, doi: 10.1109/CCAA.2016.7813826.

[4] A. Qusef, L. Issa, E. Ayoubi, and S. Murad, “Challenges and opportunities in cloud testing,” *ACM Int. Conf. Proceeding Ser.*, 2019, doi: 10.1145/3368691.3368706.

[5] R. Subramanyan, “Testing as a service in the cloud,” *Proc. - 2013 IEEE 7th Int. Symp. Serv. Syst. Eng. SOSE 2013*, p. 441, 2013, doi: 10.1109/SOSE.2013.100.

[6] H. Geng, “Internet of things and data analytics handbook,” *Internet Things Data Anal. Handb.*, pp. 1–776, 2017, doi: 10.1002/9781119173601.

[7] N. C. Cárdenas, “Computacion En La Nube Cloud Computing,” pp. 46–51, 2014.

[8] J. F. Vecchio, F. J. Paternina, and C. H. Miranda, “La computación en la nube: un modelo para el desarrollo de las empresas,” *Prospect*, vol. 13, no. 2, 2015, [Online]. Available: http://www.scielo.org.co/pdf/prosp/v13n2/v13n2a10.pdf.

[9] A. Abhilasha, “A Survey on Cloud Computing and Its Benefits,” *Int. J. Comput. Technol.*, vol. 15, no. 2, pp. 6499–6503, 2015, doi: 10.24297/ijct.v15i2.568.

[10] José Manuel and Navarro Arévalo, “Cloud Computing: Fundamentos, diseño y arquitectura aplicados a un caso de estudio,” *Tesis*, pp. 1–78, 2017.

[11] H. Husni and A. A. Saifan, “Cloud Testing : Steps , Tools , Challenges,” no. April, 2017.

[12] F. Ramírez, “Una Perspectiva De Las Pruebas De Software En La Nube,” *Academia.Edu*, [Online]. Available: http://www.academia.edu/download/31340372/Testinginthecloud2.pdf.

[13] X. G. Ram, M. Andr, and J. Hern, “Seguridad en la nube, evolución indispensable en el siglo XXI,” vol. 16, pp. 110–127, 2019.

[14] U. Privada and R. Belloso, “La nueva era de los negocios: Computación en la nube,” vol. 2017, no. 2010, pp. 172–191, 2017.

[15] O. Montoya and J. Antonio, “Gestión de riesgo y seguridad en computación en la nube para pymes,” *Univ. Pilot. Colomb.*, pp. 1–11, 2018, [Online]. Available: http://polux.unipiloto.edu.co:8080/00004649.pdf.

[16] N. J. Mitchell and K. Zunnurhain, “Google cloud platform security,” *Proc. 4th ACM/IEEE Symp. Edge Comput. SEC 2019*, pp. 319–322, 2019, doi: 10.1145/3318216.3363371.

[17] J. Varia and S. Mathew, “Overview of Amazon Web Services (Survey Report),” no. January, pp. 1–30, 2014, [Online]. Available: http://media.amazonwebservices.com/AWS_Overview.pdf.

[18] A. J. Ruiz Caldas, “Migración de servidores a la nube de Microsoft Azure para mejorar la continuidad de los servicios de TI, de la Fiduciaria en el año 2018,” *Univ. San Ignacio Loyola*, 2019.

[19] G. Carutasu, M. A. Botezatu, C. Botezatu, and M. Pirnau, “Cloud computing and windows azure,” *Proc. 8th Int. Conf. Electron. Comput. Artif. Intell. ECAI 2016*, 2017, doi: 10.1109/ECAI.2016.7861168.

[20] S. P. T. Krishnan, J. L. U. Gonzalez, S. P. T. Krishnan, and J. L. U. Gonzalez, “Getting Started with Google Cloud Platform,” *Build. Your Next Big Thing with Google Cloud Platf.*, pp. 13–25, 2015, doi: 10.1007/978-1-4842-1004-8_2.

[21] P. Wankhede, M. Talati, and R. Chinchamalatpure, “Comparative Study of Cloud Platforms -Microsoft Azure, Google Cloud Platform and Amazon Ec2,” *J. Res. Eng. Appl. Sci.*, vol. 05, no. 02, pp. 60–64, 2020, doi: 10.46565/jreas.2020.v05i02.004.

[22] N. Dangwal, N. M. Dewan, and S. Sachdeva, “Testing the Cloud and Testing as a Service,” 2016.

[23] P. Arora and A. DIxit, “Comparison of Traditional Testing and Cloud Testing,” *Proc. - IEEE 2018 Int. Conf. Adv. Comput. Commun. Control Networking,*

ICACCCN 2018, pp. 420–423, 2018, doi: 10.1109/ICACCCN.2018.8748403.

[24] P. Harikrishna and A. Amuthan, "A survey of testing as a service in cloud computing," *2016 Int. Conf. Comput. Commun. Informatics, ICCCI 2016*, 2016, doi: 10.1109/ICCCI.2016.7479949.

[25] A. Mateo, M. Tutor, R. Sotomayor Fernández Director, and L. Miguel Sánchez García, "Análisis y diseño de la aplicación SaaS para gestión de proyectos SimpleDesk," 2012.

[26] E. Chinarro Morales, M. E. Ruiz Rivera, and E. Ruiz Lizama, "Desarrollo de un Modelo de Pruebas Funcionales de Software Basado en la Herramienta SELENIUM," *Ind. Data*, vol. 20, no. 1, p. 139, 2017, doi: 10.15381/idata.v20i1.13498.

[27] K. Sneha and G. M. Malle, "Research on software testing techniques and software automation testing tools," *2017 Int. Conf. Energy, Commun. Data Anal. Soft Comput.*, pp. 77–81, 2017.

[28] A. B. Bada and A. Abubakar, "Software Testing On the Cloud," pp. 34–40, 2021, doi: 10.9790/1813-1001013440.

[29] S. Suryadevara and S. Ali, "Preperformance Testing of A Website," pp. 33–52, 2020, doi: 10.5121/csit.2020.100703.

[30] S. Nachiyappan and S. Justus, "Cloud testing tools and its challenges: A comparative study," *Procedia Comput. Sci.*, vol. 50, pp. 482–489, 2015, doi: 10.1016/j.procs.2015.04.018.

[31] N. Kilinç, L. Sezer, and A. Mishra, "Cloud-based test tools: A brief comparative view," *Cybern. Inf. Technol.*, vol. 18, no. 4, pp. 3–14, 2018, doi: 10.2478/cait-2018-0044.

[32] N. Srivastava, U. Kumar, and P. Singh, "Software and Performance Testing Tools," *J. Informatics Electr. Electron. Eng.*, vol. 2, no. 1, pp. 1–12, 2021, doi: 10.54060/jieee/002.01.001.

[33] M. Hayek, P. Farhat, Y. Yamout, C. Ghorra, and R. A. Haraty, "Web 2.0 Testing Tools: A Compendium," *2019 Int. Conf. Innov. Intell. Informatics, Comput. Technol. 3ICT 2019*, pp. 1–6, 2019, doi: 10.1109/3ICT.2019.8910274.

[34] S. Bhatti and R. Kumari, “Comparative Study of Load Testing Tools,” pp. 2334–2338, 2015.

[35] S. Pradeep and Y. K. Sharma, “A Pragmatic Evaluation of Stress and Performance Testing Technologies for Web Based Applications,” *Proc. - 2019 Amity Int. Conf. Artif. Intell. AICAI 2019*, pp. 399–403, 2019, doi: 10.1109/AICAI.2019.8701327.

[36] A. Proskurin, “Adapting a Stress Testing Framework to a Multi- module Security-oriented Spring Application,” 2017.

[37] Aucla, “Автоматизация нагрузочного тестирования web-приложений с использованием инструмента Taurus,” *Αγαη*, vol. 8, no. 5, p. 55, 2019.

[38] “ISO 25010.” https://iso25000.com/index.php/normas-iso-25000/iso-25010?limit=3&limitstart=0 (accessed Jul. 23, 2021).

[39] V. Casola, A. De Benedictis, M. Rak, and U. Villano, “A methodology for automated penetration testing of cloud applications,” 2020.

[40] S. He, G. Manns, J. Saunders, W. Wang, L. Pollock, and M. Lou Soffa, “A statistics-based performance testing methodology for cloud applications,” *ESEC/FSE 2019 - Proc. 2019 27th ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, pp. 188–199, 2019, doi: 10.1145/3338906.3338912.

[41] M. Albonico, A. Benelallam, J. M. Mottu, and G. Sunyé, “A DSL-based approach for elasticity testing of cloud systems,” *DSM 2016 - Proc. Int. Work. Domain-Specific Model. co-located with SPLASH 2016*, pp. 8–14, 2016, doi: 10.1145/3023147.3023149.

[42] H. Pei, B. Yin, and M. Xie, “Dynamic Random Testing Strategy for Test Case Optimization in Cloud Environment,” *Proc. - 29th IEEE Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2018*, pp. 148–149, 2018, doi: 10.1109/ISSREW.2018.000-9.

[43] B. K. Rani, B. P. Rani, and A. V. Babu, “Cloud computing and inter-clouds-types, topologies and research issues,” *Procedia Comput. Sci.*, vol. 50, pp. 24–29, 2015, doi: 10.1016/j.procs.2015.04.006.

[44] “Registro Oficial No 320 Ley de Propiedad Intelectual.”

[45] “Ley Orgánica de Educación Superior, LOES.”

[46] “Ley de Comercio Electrónico, Firmas y Mensajes de datos.”

[47] J. Colantonio, “15 herramientas de prueba de carga superior para 2021 (guía de código abierto),” 2021. https://testguild.com/load-testing-tools/ (accessed Nov. 16, 2021).

[48] J. B. Jiménez, “Tipos de Pruebas de Software | OpenWebinars,” 2019. https://openwebinars.net/blog/tipos-de-pruebas-de-software/ (accessed Nov. 16, 2021).

[49] Amazon Web Services, “Distributed Load Testing on AWS,” *AWS Solut. Libr. AWS Solut. Implementations*, 2020, [Online]. Available: https://aws.amazon.com/solutions/implementations/distributed-load-testing-on-aws/.

ANEXOS

INSTRUMENTO PARA EVALUAR HERRAMIENTAS DE PRUEBA DE SOFTWARE BASADO EN LA ISO 25010

FICHA DE EVALUACIÓN DE HERRAMIENTA

Descripción general de la herramienta	
Nombre de la herramienta	Locust
Proveedor que se usa	Google Cloud Platform
Prueba de software a realizar	Prueba de Carga
Evaluador	Elías Estupiñán

Usabilidad: Para obtener los valores cuantitativos dentro de este aspecto se realiza una equivalencia en relación a la opción que se escoja, siendo así que, 3 es la calificación alta, 2 la media, y 1 la baja.

Marque con una X	Alta	Media	Baja
Facilidad de aprendizaje en base al uso de la herramienta			
Nivel de simplicidad de la herramienta			
Nivel de estética que presenta la herramienta			
Nivel de documentación			

Intervalo de ponderación de usabilidad

Intervalo	Significado	Grado de importancia
9-12	Grado de usabilidad alta	Alta
4-8	Grado de usabilidad media	Media
1-3	Grado de usabilidad baja	Baja

Desempeño: Se hará uso de recursos básicos y necesarios dentro de cada proveedor dentro de los estándares necesario para correr cada herramienta, por ende, cada prueba que se ejecute estará en igualdad de condiciones una de otra.

Procesos	Tiempos
Creación de los recursos	
Acondicionamiento del entorno	
Ejecución de la prueba de carga	
Destrucción del recurso	

Compatibilidad: Para obtener los valores cuantitativos dentro de este aspecto se realiza una equivalencia en relación a la opción que se escoja, siendo así que, 3 es la calificación alta, 2 la media, y 1 la baja.

Marque con una X	Alta	Media	Baja
Nivel de coexistencia con el proveedor			
Nivel de operabilidad con el proveedor			

Intervalo de ponderación de compatibilidad

Intervalo	Significado	Grado de importancia
5-6	Grado de usabilidad alta	Alta
3-4	Grado de usabilidad media	Media
1-2	Grado de usabilidad baja	Baja

Anexo 1 Ficha de evaluación