ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

PROYECTO DE TITULACIÓN
PREVIO A LA OBTENCIÓN DEL TÍTULO DE
INGENIERO DE SISTEMAS E INFORMÁTICA

TEMA: PROTOTIPO DE PLATAFORMA COMO SERVICIO (PaaS) BASADO EN TECNOLOGÍAS ABIERTAS

AUTOR: ANDRÉS SEBASTIÁN RODRÍGUEZ FLORES

DIRECTOR: ING. FREDDY DUEÑAS
CODIRECTOR: ING. PABLO PARRA

SANGOLQUÍ
2015

CERTIFICADO DIRECTOR Y CODIRECTOR

En nuestra calidad de Director y Codirector certificamos que el trabajo de grado: "PROTOTIPO DE PLATAFORMA COMO SERVICIO (PaaS) BASADO EN TECNOLOGÍAS ABIERTAS", elaborado por el Sr. Andrés Sebastián Rodríguez Flores, ha sido dirigido y revisado durante su ejecución a través de reuniones periódicas para dar cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación, por lo tanto autorizamos la presentación del mismo.

Sangolquí, Agosto del 2015

Ing. Freddy Dueñas
Director

Ing. Pablo Parra
Codirector

AUTORÍA DE RESPONSABILIDAD

Yo, Andrés Sebastián Rodríguez Flores, declaro que el trabajo de grado: "PROTOTIPO DE PLATAFORMA COMO SERVICIO (PaaS) BASADO EN TECNOLOGÍAS ABIERTAS", ha sido desarrollado citando las fuentes correspondientes y respetando las disposiciones legales que protegen los derechos de autor vigentes.

Por tal razón, las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo, son de mi exclusiva responsabilidad como autor.

Sangolquí, Agosto del 2015

Andrés Sebastián Rodríguez Flores

AUTORIZACIÓN

Yo, Andrés Sebastián Rodríguez Flores, autorizo a la Universidad de las Fuerzas Armadas ESPE para la publicación en la Biblioteca Virtual de la institución, del trabajo de grado: "PROTOTIPO DE PLATAFORMA COMO SERVICIO (PaaS) BASADO EN TECNOLOGÍAS ABIERTAS", cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y autoría.

Sangolquí, Agosto del 2015

Andrés Sebastián Rodríguez Flores

DEDICATORIA

El presente proyecto está dedicado primeramente y en la misma medida a mi esposa, a mis padres y hermanos.

A mi esposa, que ha sido la persona que más me ha incentivado y apoyado para sacar adelante este proyecto de titulación, a pesar de los contratiempos encontrados en el camino. Su fuerza e interés han sido claves para llegar a este punto.

A mi familia, mis padres y hermanos, quienes siempre han sabido acompañarme con sus consejos, apoyo y cariño. Este paso había sido aplazado pero ahora que lo he dado espero llenarlos de orgullo.

De igual manera está dedicado a mis amigos, quienes de una u otra forma han estado siempre pendientes de la realización de este proyecto y con quienes de seguro también festejaremos este logro.

AGRADECIMIENTO

Agradezco a mi esposa quien ha sido pieza clave del desarrollo de este proyecto, a mis padres por todos los esfuerzos realizados para darme una educación de calidad, a mis hermanos por su cariño y apoyo.

Agradezco de igual manera al director y codirector de este proyecto, Ing. Freddy Dueñas e Ing. Pablo Parra, por su guía y consejos para poder desarrollar este trabajo de la mejor manera.

Agradezco también a la Universidad de las Fuerzas Armadas ESPE y a la empresa auspiciante CONSULTORES ADHOC S.A. por entregarme las facilidades para el desarrollo de este proyecto.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En el presente proyecto se revisan los conceptos asociados con la Computación en la Nube y las herramientas que habilitan su utilización. La Computación en la Nube es un modelo de negocio que permite a proveedores externos y a departamentos internos de Tecnología de la Información entregar servicios de infraestructura, plataforma y software listos para consumir bajo demanda por un costo específico y medible, evitando que los clientes finales corran con los riesgos asociados a la operación de sus sistemas de información. Al ser un modelo en constante evolución es recomendable tomar en cuenta iniciativas de estandarización y arquitecturas de referencia que faciliten el conocimiento y la comprensión de los elementos que deben ser incluidos dentro de las soluciones. Partiendo de estos estándares se han creado un gran número de proyectos, basados tanto en tecnologías abiertas como privativas, que facilitan la adopción de este modelo de gestión tecnológica dentro de las organizaciones. El análisis de las opciones disponibles en el mercado permite conocer y seleccionar qué herramientas pueden ser utilizadas de acuerdo a requerimientos específicos y comprobar si es posible la implantación de una nube privada que tenga opciones de integración con recursos públicos que faciliten la administración y permitan hacer uso de las mejores prácticas y tendencias del mercado de una forma ágil y poco riesgosa. En la práctica, la integración de soluciones de Computación en la Nube permite validar las características descritas dentro de un prototipo que permite visualizar el comportamiento de los componentes y probar su funcionalidad.

PALABRAS CLAVE

- **COMPUTACIÓN EN LA NUBE**
- **INFRAESTRUCTURA COMO SERVICIO**
- **PLATAFORMA COMO SERVICIO**
- **NUBE PRIVADA**
- **MODELO DE GESTIÓN DE TECNOLOGÍA**

ABSTRACT

In this project the concepts associated with Cloud Computing and tools that enable its use are reviewed. Cloud Computing is a business model that allows external suppliers and internal IT departments to deliver infrastructure, platform and software services ready to consume on demand for a specific and measurable cost, so the customers avoids bearing the risks associated with the operation of their information systems. As an evolving model, it is advisable to take into account standardization initiatives and reference architectures that facilitate the generation of knowledge and understanding of the elements that must be included in cloud solutions. Based on these standards a large number of projects have been created, both with open and privative technologies, which facilitate the adoption of this technology management model within organizations. The analysis of the technologies and tools available in the market permit the selection of the tools that can be used according to specific requirements and check whether the implementation of a private cloud that has integration options with public resources is possible, to allow administration and use of best practices and market trends in an agile and less risky way. In practice, the integration of Cloud Computing solutions in a prototype allow the validation of features required and the visualization of the components behavior and test its functionality.

KEYWORDS

- **CLOUD COMPUTING**
- **INFRASTRUCTURE AS A SERVICE**
- **PLATFORM AS A SERVICE**
- **PRIVATE CLOUD**
- **TECHNOLOGY MANAGEMENT MODEL**

CAPÍTULO 1. INTRODUCCIÓN

1.1. INTRODUCCIÓN

Según el Instituto Nacional de Estándares y Tecnología de los Estados Unidos de América (US National Institute of Standards and Technology, NIST) la Computación en la Nube es un modelo que permite el acceso conveniente y bajo demanda a un grupo de recursos de cómputo compartidos y configurados a medida, por ejemplo: redes, servidores, almacenamiento, aplicaciones y servicios, que pueden ser aprovisionados y entregados con un esfuerzo mínimo de gestión e interacción con el proveedor.

Este tipo de soluciones han ganado un gran apoyo en la industria debido a su flexibilidad y las ventajas de negocio que ofrecen a las empresas, habilitando la creación y existencia de muchas empresas y emprendimientos que apoyan sus operaciones en soluciones de Computación en la Nube.

Con la reducción de la carga administrativa y operativa de la infraestructura tecnológica, las empresas pueden enfocarse en los procesos y actividades que son parte del núcleo de su negocio sin dejar de aprovechar y beneficiarse de los últimos adelantos de las tecnologías de la información.

Uno de los modelos de Computación en la Nube es conocido como Plataforma como Servicio (Platform as a Service, PaaS). Este modelo provee software como herramientas de programación y componentes especializados para el desarrollo y creación de aplicaciones. Está soportado por el modelo conocido como Infraestructura como Servicio (Infrastructure as a Service, IaaS) que provee acceso a recursos de infraestructura bajo demanda.

El modelo PaaS entrega, entonces, el acceso a herramientas y lenguajes que permiten la creación de nuevas aplicaciones rápidamente y a bajo costo. Estas nuevas aplicaciones pueden ser entregadas públicamente o en modelos de nube privada para consumo interno de una sola empresa.

Sin embargo uno de los riesgos asociados al modelo PaaS es la existencia de tecnologías restrictivas que podrían provocar que un cliente solo consuma los recursos de un proveedor, quedando posiblemente atado al mismo, lo cual puede ser contraproducente. Para evitar este riesgo existen alternativas libres que están emergiendo y se siguen desarrollando para llenar este importante nicho de negocio entregando recursos estandarizados que apoyen a su vez el uso de otras tecnologías libres que pueden ser habilitadas.

El presente proyecto persigue el diseño e implementación de un prototipo de Plataforma como Servicio, que haciendo uso de tecnologías abiertas permita la entrega de este tipo de servicio en la nube. En el mercado existen tecnologías nuevas y abiertas que siguen ganando seguidores y apoyo debido a las ventajas y estándares que proponen.

1.2. PLANTEAMIENTO DEL PROBLEMA

El modelo PaaS de Computación en la Nube tiene sus ventajas y riesgos asociados. El uso de tecnologías abiertas para habilitar y posibilitar la entrega de este tipo de servicios es importante ya que evita el que los usuarios queden atados y obligados a consumir los servicios de un solo proveedor. Las tecnologías abiertas se apoyan en estándares que buscan evitar este tipo de restricciones permitiendo que cada usuario elija en qué momento migrará hacia otro proveedor de servicios sin que esto implique un proceso traumático, complejo y, en muchos casos, hasta imposible.

Dentro del desarrollo de aplicaciones existe otro punto a tomar en cuenta. Hace 15 años las aplicaciones eran desplegadas y desarrolladas usando un servidor físico que contenía un paquete de servicios y aplicativos bien definido. Hoy por hoy los desarrolladores construyen sus aplicaciones usando una selección de los mejores servicios disponibles y deben desplegar las mismas en ambientes de hardware heterogéneos que incluyen servidores virtualizados públicos y privados. Esto implica dificultades en las interacciones entre los componentes de aplicaciones que pueden llevar a provocar inconsistencias entre las dependencias de los paquetes y servicios utilizados para su ejecución.

Existe un gran número de combinaciones entre las aplicaciones, los servicios y el hardware que se puede utilizar. Esto crea la necesidad de considerar estos factores cada vez que una aplicación es escrita y modificada, situación que es compleja tanto para los desarrolladores que crean las aplicaciones con ciertos requisitos de plataforma e infraestructura así como para los operadores que intentan entregar un ambiente operativo escalable, seguro y que cumpla con los requisitos específicos de cada aplicación.

Todos estos antecedentes configuran el problema a resolver mediante la creación de un prototipo de Plataforma como Servicio (PaaS) ya que es necesario encontrar opciones que permitan mitigar los inconvenientes expuestos al brindar este tipo de servicio , facilitando la entrega de ambientes heterogéneos, portables y escalables que cumplan con los requerimientos de diferentes tipos de clientes. Las plataformas entregadas bajo demanda deben solucionar problemas de dependencias entre paquetes requeridos por cada aplicación de forma que los clientes obtengan las características de hardware necesarias y el software base e intermedio adecuados para crear y ejecutar aplicaciones. La automatización de esta entrega debe permitir a los usuarios mejorar sus tiempos de desarrollo y conseguir aplicaciones de mejor calidad.

1.3. JUSTIFICACIÓN E IMPORTANCIA

CONSULTORESADHOC S.A. es la empresa que auspiciará este proyecto. Tiene 5 años en el mercado y se especializa en la entrega de servicios de instalación, configuración, capacitación y soporte técnico de herramientas de software para la automatización de la gestión de infraestructura de TI.

La empresa auspiciante busca desarrollar conocimiento que permita fortalecer la propuesta de valor hacia sus clientes aplicando y haciendo uso de soluciones abiertas para incursionar en la entrega de servicios de Computación en la Nube.

Durante estos años la empresa ha podido encontrar una necesidad creciente en sus clientes de apegarse a la tendencia del mercado y disfrutar de los beneficios del uso de soluciones basadas en Computación en la Nube (Cloud Computing).

La empresa requiere crear este prototipo haciendo uso de soluciones abiertas que están ganando popularidad y apoyándose en estándares de la industria para proveer en el mediano plazo soluciones de Cloud Computing que permitan diversificar su portafolio de servicios y apoyar las operaciones de sus clientes con una infraestructura flexible y asequible, ya sea entregando el servicio o apoyando iniciativas de nube privada que estén siendo llevadas como proyectos por parte de sus clientes.

Este tipo de tecnologías tienen un punto a favor al entregar soluciones abiertas a problemas que se han generado con la demanda de la computación en la nube. Tienen apoyo de empresas y corporaciones interesadas en la

creación de soluciones estándar y una comunidad de usuarios entusiasta que permite que el desarrollo de las mismas avance a pasos agigantados.

Sin embargo, al tratarse de soluciones que se encuentran en desarrollo aún no son recomendadas para ambientes de producción, para lo cual se requiere la creación de un prototipos y pilotos que permitan evaluar y comprobar la efectividad de estas soluciones antes de comprometer procesamiento crítico apoyado en las mismas.

Al estar en este punto del desarrollo de las tecnologías se plantea el diseño e implementación de un prototipo que permita experimentar y validar las características ofrecidas, evaluando los distintos factores asociados a una futura entrega de servicios de Plataforma como Servicio.

1.4. OBJETIVOS

1.4.1. OBJETIVO GENERAL

Desarrollar e implantar un prototipo de Plataforma como Servicio (PaaS) para CONSULTORESADHOC S.A., haciendo uso de tecnologías abiertas que permitan experimentar con sus características y funcionalidades para evaluar la entrega de servicios en la nube.

1.4.2. OBJETIVOS ESPECÍFICOS

- Investigar la propuesta de funcionalidad y características de soluciones abiertas para Infraestructura como Servicio y Plataforma como Servicio, permitiendo la comprensión de los componentes necesarios para habilitar estas soluciones.

- Validar las características de soluciones de Infraestructura como Servicio y Plataforma como Servicio existentes en el mercado haciendo uso de la metodología incluida en el estándar ISO 25000, para seleccionar las herramientas más adecuadas para la creación del prototipo.

- Desarrollar e implantar un prototipo de Plataforma como Servicio, integrando las funcionalidades de las tecnologías seleccionadas.

- Implantar 3 imágenes que puedan ser consumidas a través de la infraestructura del prototipo de Plataforma como Servicio.

- Realizar pruebas funcionales del prototipo que permitirán analizar posteriormente la inclusión e inversión necesaria para entregar servicios de Computación en la Nube, específicamente PaaS.

1.5. ALCANCE

El alcance del presente proyecto se resume en los siguientes puntos:

- Desarrollo de arquitectura de prototipo

- Instalación y configuración del prototipo

- Evaluación y entrega de resultados

CAPÍTULO 2. ESTADO DEL ARTE

2.1. CONCEPTOS DE COMPUTACIÓN EN LA NUBE

En los últimos años, tecnologías como la virtualización y la computación en la nube han provocado cambios significativos en la forma como las áreas de Tecnología de la Información (TI) están organizadas en el entorno empresarial. El progreso de estas tecnologías ha permitido compartir recursos computacionales y habilitar el acceso a los mismos a múltiples usuarios. A continuación se incluyen conceptos básicos relacionados con la computación en la nube.

2.2. DEFINICIONES GENERALES

2.2.1. DEFINICIÓN DE COMPUTACIÓN EN LA NUBE

Según la definición oficial del Instituto Nacional de Estándares y Tecnología de Estados Unidos (NIST, por sus siglas en inglés), la computación en la nube es un modelo que habilita un acceso de red localizado, conveniente y bajo demanda a un grupo de recursos computacionales configurables compartidos (por ejemplo: redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser entregados rápidamente y con un mínimo esfuerzo administrativo o interacción del proveedor de servicios.

Al hablar de computación en la nube, un proveedor de servicios entrega acceso al software, se preocupa del alojamiento y almacenamiento y también ayuda a los desarrolladores a crear aplicaciones de forma ágil. La ubicación física del hardware es decisión exclusiva del proveedor. Puede estar en un

centro de datos dedicado, en centros de datos múltiples, o compartido entre centros de datos. Cuando no se conoce la ubicación exacta de la infraestructura, hablamos de computación en la nube.

La computación en la nube hace uso de tecnologías existentes, como la virtualización. Las tecnologías que están involucradas son conocidas, y esto combinado con la aparición de proveedores de servicios en la nube y el Internet ha creado nuevas posibilidades para las áreas de Tecnología de la Información y las unidades de negocio. Es por esto que se puede decir que más que una nueva tecnología, la computación en la nube es un nuevo modelo de negocio.

El modelo de computación en la nube identifica cinco características esenciales: autoservicio bajo demanda, acceso de red ampliado, recursos compartidos, elasticidad y medición del servicio.

La computación en la nube cuenta con tres modelos de servicio: software, plataforma e infraestructura. Adicionalmente cuenta con modelos de despliegue: nube privada, nube comunitaria, nube pública y nube híbrida. (National Institute of Standards and Technology, 2011)

2.2.2. CARACTERÍSTICAS DE LA COMPUTACIÓN EN LA NUBE

A continuación se detallan las características esenciales de la computación en la nube, según el NIST. (National Institute of Standards and Technology, 2011)

- **Autoservicio bajo demanda:** permite al consumidor obtener recursos computacionales según sean requeridos, sin requerir interacción humana con los proveedores de servicio.

- **Acceso de red ampliado:** los recursos están disponibles a través de redes de computacionales y se puede acceder a través de mecanismos estándar que promueven el uso de clientes variados como laptops, tablets, teléfonos, etc.

- **Recursos compartidos:** los recursos están agrupados para servir a múltiples consumidores, con recursos físicos y virtuales asignados dinámicamente y reasignados de acuerdo a las necesidades de los consumidores. Los recursos incluyen almacenamiento, procesamiento, memoria y ancho de banda de red.

- **Elasticidad:** los recursos pueden ser aprovisionados y desplegados, en algunos casos automáticamente, y escalar de acuerdo a lo requerido. Para el consumidor estos recursos aparentan ser ilimitados, pudiendo acceder a ellos en cualquier monto y en cualquier momento.

- **Medición del servicio:** los proveedores pueden medir de forma automatizada el uso de los recursos de acuerdo a las mediciones apropiadas para cada tipo. El uso de estos recursos puede ser monitoreado, controlado y reportado, entregado transparencia tanto para el proveedor como para el consumidor sobre los servicios utilizados.

2.3. PERSPECTIVAS DE NEGOCIO

La computación en la nube es un nuevo modelo de negocio de TI que puede entregar muchas ventajas. Sin embargo, también existen algunos puntos que pueden impactar de forma negativa y que hay que tener en cuenta al momento de adoptar este modelo. (Hausman, Cook, & Sampaio, 2013)

Desde una perspectiva netamente de negocio, la computación en la nube puede ser vista como un tipo de tercerización (outsourcing). La computación en la nube permite contratar servicios de TI de proveedores externos y también marcar claramente los servicios que el departamento de TI interno está entregando a las demás áreas de la organización.

Las siguientes son algunas de las razones por las cuales resulta interesante, desde el punto de vista de negocio, hacer uso de la computación en la nube.

- **Disminución de costos asociados a los servidores:** Permite contar con mayor claridad al momento de calcular los costos de los elementos de infraestructura requeridos para operar un servicio.
- **Disponibilidad de servicios que no forman parte del núcleo del negocio:** Permite entregar servicios de soporte que van a ser gestionados por personal especializado sin que esto implique un costo en personal y capacitación adicional para el cliente.
- **La entrega de infraestructura consume tiempo y puede retrasar la comercialización de productos importantes:** La flexibilidad y rapidez de entrega de recursos en la nube permite aprovechar las oportunidades de negocio.
- **Innovación:** Un proveedor de servicios en la nube puede entregar de forma más rápida tecnologías innovadoras que pueden resultar costosas al ser adoptadas internamente. Esto beneficia la experiencia del usuario.
- **Reducción de la inversión en mantenimiento:** implica el mantenimiento de centros de datos y tecnología asociada, incluyendo los costos de capacitación y suministros de mantenimiento relacionados.

Según publicaciones de The Open Group (The Open Group, 2011), estos son algunos de los inconvenientes que están asociados a la computación en la nube.

- **Cumplimiento de normas de seguridad:** es necesario que los proveedores y todos los componentes relacionados con un servicio en la nube, estén alineados en el cumplimiento de normas de seguridad. Las normas de seguridad relativas a la computación en la nube siguen en desarrollo.
- **Madurez:** a pesar de contar con mucho apoyo de la industria y haber dado pasos significativos en los últimos años, las soluciones de computación en la nube continúan en desarrollo lo que permite a algunos analistas cuestionar su madurez.
- **Falta de experiencia:** este factor incluye tanto a los proveedores de soluciones como a los clientes que se enfrentan a un nuevo modelo de negocio y necesitan seguir definiendo sobre la marcha los puntos y factores que son más beneficiosos.
- **Acceso a la información:** uno de los principales puntos a considerar ya que al contar con la información en servicios de nube, el acceso a la misma se ve condicionado al nivel de conectividad y disponibilidad que entreguen los proveedores. Existen también dudas en cuanto a la privacidad de la información.
- **Integración:** este punto incluye los factores relacionados a la facilidad para integrar las soluciones de nube con soluciones internas y tradicionales con las que se necesita interactuar. Así mismo contempla los problemas que se pueden presentar al tratar de pasar de un proveedor a otro.

Todos estos factores siguen siendo tomados en cuenta como parte del desarrollo de las distintas soluciones y alternativas existentes en el mercado de soluciones en la nube.

2.4. MODELOS DE SERVICIO EN LA NUBE

La computación en la nube es un concepto que incluye varios modelos de servicio. Aunque poco a poco empiezan a generarse nuevos modelos de servicio sin embargo, conceptualmente, se los puede dividir en los siguientes.

- Infraestructura como servicio (IaaS)
- Plataforma como servicio (PaaS)
- Software como servicio (SaaS)

En la Tabla 1 se compara el control que se tiene sobre los distintos componentes en una arquitectura tradicional y los modelos de computación en la nube.

Tabla 1. Control de componentes de Arquitectura tradicional y modelos de computación en la nube

Arquitectura tradicional	Nada como servicio	IaaS	PaaS	SaaS
Aplicación	Control	No control	No control	No control
Sistema Operativo / Middleware	Control	Control	No control	No control
Hardware	Control	Control	Control	No control

Fuente: (Hausman, Cook, & Sampaio, 2013)

Por otro lado, en la Tabla 2 se presenta otra forma de ver las diferencias entre una arquitectura de TI tradicional y los modelos de computación en la nube, esta vez visto desde la perspectiva de un quién es el responsable de la administración de cada uno de los componentes analizados, si el consumidor o el proveedor.

Tabla 2. Responsable de la administración de los componentes en TI tradicional y modelos de computación en la nube

Capa de tecnología	TI Tradicional	IaaS	PaaS	SaaS
Aplicación	Consumidor	Consumidor	Consumidor	Proveedor
Sistema operativo / Middleware	Consumidor	Proveedor	Consumidor	Proveedor
Hardware	Consumidor	Proveedor	Proveedor	Proveedor

Fuente: (Hausman, Cook, & Sampaio, 2013)

Según Hausman, Kirk y Sampaio, autores del libro Cloud Essentials (Hausman, Cook, & Sampaio, 2013), los modelos de servicio debido a su interrelación se pueden representar en una gráfica piramidal que apoya la comprensión del ámbito de cada uno y su dependencia en los otros servicios.

La Infraestructura como Servicio (IaaS) es la categoría fundamental de servicio en la nube, sirviendo como base para los demás servicios. De igual forma cada servicio depende de las capacidades entregadas por el servicio inferior en la pirámide. La Figura 1 muestra estas dependencias.

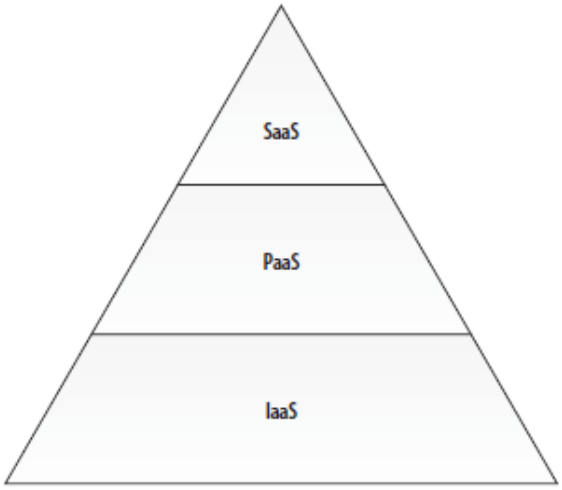

Figura 1. Relación de los tipos de servicio de computación en la nube.
Fuente: (Hausman, Cook, & Sampaio, 2013)

Creando estas capas de servicio, los proveedores empiezan con el nivel más fundamental de IaaS que incluye elementos familiares que se manejan dentro de una configuración tradicional, como redes, almacenamiento, y otros elementos de arquitectura que son de interés de los administradores de sistemas.

Los desarrolladores de aplicaciones consumirán los servicios que se entregan a nivel de PaaS, que está soportado en una infraestructura alojada en un proveedor.

Finalmente los usuarios consumirán aplicaciones que son entregadas a nivel de SaaS, incluyendo componentes de servicios de plataforma e infraestructura que están fuera de la visibilidad del consumidor.

Adicionalmente, se puede visualizar que la mayor cantidad de usuarios van a estar relacionados con el consumo de aplicaciones, conforme se desciende en los niveles de la pirámide se va a requerir que los usuarios de

cada nivel tengan conocimientos especializados para que interactúen con los diferentes componentes, disminuyendo la cantidad de usuarios que se relacionan con cada nivel. La figura 2 muestra esta interrelación.

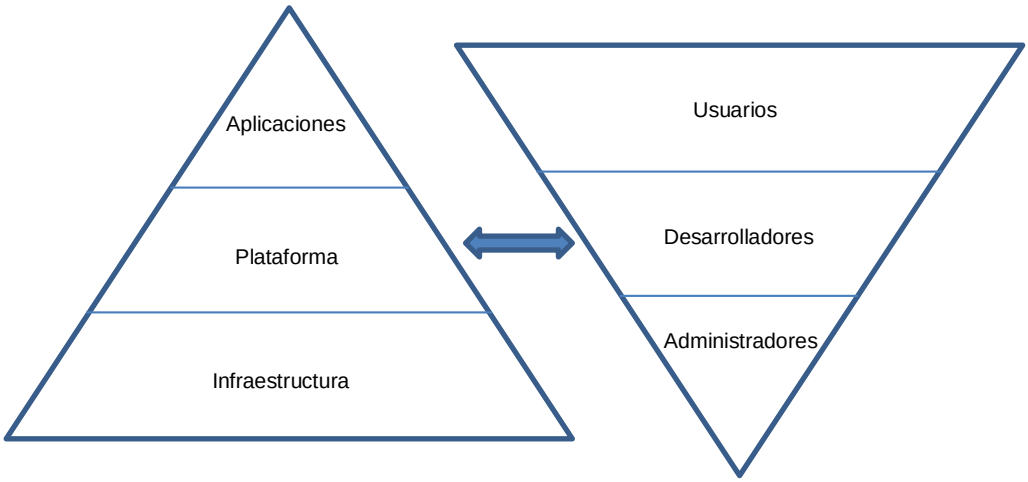

Figura 2. Modelos de servicio de nube alineados con sus principales consumidores

2.4.1. SOFTWARE COMO SERVICIO (SaaS)

El Software como Servicio es el primer ejemplo de computación en la nube que muchos usuarios pueden experimentar, esto puede darse inclusive sin que el usuario tenga conciencia que está interactuando con una nube. Las aplicaciones de software alojadas en un proveedor, disponibles a través de un navegador web, son entregadas al usuario, a quien solo le interesa hacer uso de la misma y no conocer los detalles de cómo opera esta aplicación tras bastidores. (National Institute of Standards and Technology, 2011)

Al igual que los productos de software adicionales, las alternativas SaaS están listas para ser utilizadas y entregan algunas capacidades de personalización y configuración al usuario, pero estas no pueden ser modificadas en su funcionamiento interno.

Las aplicaciones SaaS entregan varias ventajas sobre el software tradicional instalado localmente, estas ventajas han creado la necesidad de contar con esquemas de nube dentro de los ambientes empresariales existentes.

El software tradicional requiere un gasto de capital para comprar el software y también un gasto operativo para instalar, actualizar y mantener el software. La gestión de aplicaciones de software tradicional sigue un proceso predecible que en general se puede resumir en los siguientes puntos:

- Identificar un software que cumpla con un grupo de requerimientos.
- Realizar la adquisición del software identificado.
- Instalar el software en los equipos destinados (podría requerir la adquisición de hardware)
- Cumplir ciclos de mantenimiento en los cuales se apliquen parches y mejoras al software.
- Realizar la adquisición de actualizaciones del software.
- Instalar la actualización del software en los equipos destinados (podría requerir la adquisición de hardware)
- Cumplir los ciclos de mantenimiento necesarios hasta la próxima actualización del software.

Cuando a un usuario de software tradicional se le asigna un computador nuevo o es reubicado a otro espacio en la organización, sus

aplicaciones deben ser instaladas en el nuevo equipo asignado desde el cual va a acceder. Cambios en el grupo de trabajo o reorganización de los cargos empresariales podrían también requerir adquisiciones de software adicionales para cumplir con el licenciamiento de software que sea requerido.

La gestión de aplicaciones SaaS, en cambio, requiere menos pasos:

- Identificar proveedores de servicios en la nube cuyo software cumple con los requerimientos de la organización.

- Obtener licenciamiento para acceder al servicio de software identificado.

Una vez que se contrata el servicio de software a un proveedor, todo el mantenimiento, incluyendo parches y actualizaciones, es manejado por el proveedor. La reubicación de los usuarios y los cambios en los equipos de hardware no afectan a la disponibilidad de las aplicaciones SaaS mientras se cuenta con la aplicación cliente que normalmente puede ser un navegador web.

Cambios dentro de la estructura organizacional requieren solamente costos operativos para licenciar accesos adicionales al servicio en la nube, licencias que pueden ser incluso retenidas por el usuarios mientras va cambiando de roles en la organización.

Las aplicaciones SaaS cuentan con disponibilidad extendida, pudiendo soportar procesos de negocio adicionales como recuperación ante desastres y continuidad de negocio, habilitación de trabajo remoto y colaboración entre áreas de la organización o con asociados externos a la organización.

Dentro de las ventajas asociadas a SaaS se pueden mencionar:

- Agilidad en los procesos de negocio al asegurar que los trabajadores que están fuera de la oficina tengan acceso a aplicaciones de negocio claves mientras visitan nuevos territorios u operan a través de clientes móviles.

- Los empleados tienen la capacidad de operar durante desastres naturales y los recursos de información de la empresa pueden ser ubicados en el almacenamiento del proveedor de nube en las áreas afectadas.

- Las organizaciones pueden tomar ventaja de la facilidad para compartir recursos entre empleados que trabajen en diferentes zonas horarias. (Hausman, Cook, & Sampaio, 2013)

La Figura 3 muestra algunos ejemplos de Software como Servicio.

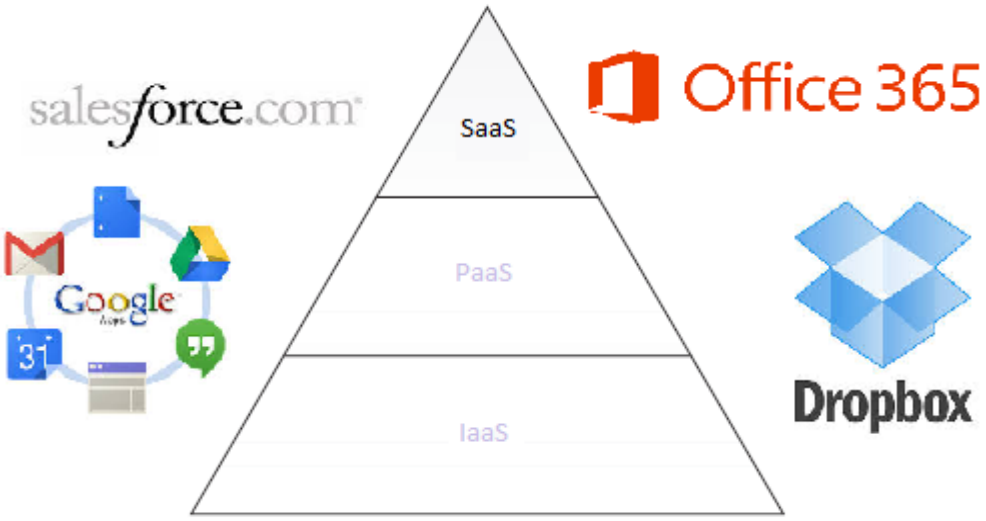

Figura 3. Ejemplos de Software como Servicio (SaaS)

2.4.2. PLATAFORMA COMO SERVICIO (PaaS)

Al hablar de Plataforma como Servicio se hace referencia a la expansión de las capacidades organizativas para personalizar el desarrollo de aplicaciones en la nube proveyendo acceso a herramientas de desarrollo de aplicaciones y ambientes de desarrollo en la nube.

Las opciones dentro de una Plataforma como Servicio están atadas a las tecnologías de un proveedor, así como a los lenguajes de programación y otras características propias de cada proveedor. Esto es similar a la forma como muchos ambientes de desarrollo de aplicaciones están asociados a un grupo de herramientas estándar que los desarrolladores usarán para crear software desplegable en redes empresariales tradicionales.

Los proveedores de PaaS pueden asociar sus plataformas de servicio en la nube a herramientas de desarrollo existentes para simplificar la adopción de nuevas alternativas de nube por parte de desarrolladores tradicionales.

Ya que se pueden usar las mismas tecnologías para ambientes tradicionales y de tipo PaaS, el proceso de migrar soluciones de software existentes de ambientes tradicionales a alojamiento en la nube se ve simplificado.

Se pueden percibir beneficios incluso al hacer uso de servicios de nube privada ubicados dentro del mismo centro de datos de un ambiente tradicional, las versiones en la nube de aplicaciones existentes se pueden expandir hacia múltiples servidores y recursos permitiendo al mismo software mayor capacidad de crecimiento y flexibilidad.

Debido a que los proveedores de PaaS entregan la infraestructura para sus servicios de desarrollo de aplicaciones en la nube, ellos pueden elegir qué lenguajes estarán disponibles para desarrollar aplicaciones en su plataforma. Esto produce que se creen temores sobre una posible dependencia de proveedor con un ambiente relativamente nuevo de servicio en el que los agentes pueden cambiar o incluso cerrar sus operaciones mientras las opciones de alojamiento evolucionan con el mercado.

Algunos proveedores PaaS han creado sus propios lenguajes de desarrollo de aplicaciones, aunque tratan de crear lenguajes análogos a los lenguajes existentes para disminuir la curva de aprendizaje para contar con programadores que desarrollen aplicaciones sobre sus plataformas. Proveedores más robustos permiten programar usando lenguajes estandarizados de código abierto para facilitar la adopción y migración de aplicaciones organizacionales existentes.

Las organizaciones normalmente elegirán a un proveedor PaaS que se ajuste a los lenguajes de programación que usan actualmente en sus ambientes de desarrollo internos, para facilitar la adopción de tecnologías de nube, pudiendo de esta forma usar sus herramientas de programación para conectarse a destino en la nube para desplegar las aplicaciones de la misma forma que si fuera un despliegue local. Los proveedores líderes de PaaS están tratando de asegurar sus ofertas de servicios entregando soporte a lenguajes abiertos como Java y Python para eliminar la posibilidad de dependencia de un proveedor.

La dependencia de un proveedor (vendor lock-in) se refiere a la condición en la cual una organización se encuentra apoyándose en una tecnología propietaria que restringe las posibilidades de migrar en un futuro a alguna solución alternativa. Las organizaciones que buscan agilidad deben

tomar en cuenta en su planificación este tema. (Hausman, Cook, & Sampaio, 2013)

La Figura 4 muestra algunos ejemplos de soluciones de Plataforma como Servicio.

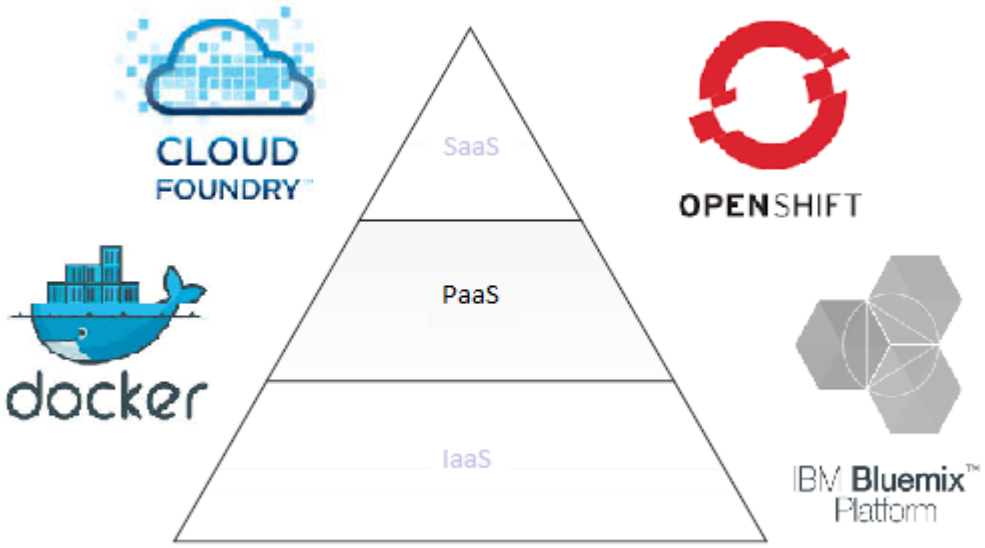

Figura 4. Ejemplos de Plataforma como Servicio (PaaS)

2.4.3. INFRAESTRUCTURA COMO SERVICIO (IaaS)

La Infraestructura como Servicio permite a un cliente control casi completo sobre las aplicaciones, lenguajes, y recursos fundamentales que soportan servicios organizacionales como bases de datos, almacenamiento y red.

En ocasiones también se conoce a esta tecnología como Hardware como Servicio (Hardware as a Service, HaaS) para reflejar su función al proveer recursos de hardware bajo demanda a sus clientes, quienes luego pueden entregar estos recursos para cumplir con las necesidades particulares

de la organización en la forma de máquinas virtuales para servicios de bases de datos, almacenamiento de archivos, servicios de autenticación y cualquier otra función requerida.

Esta capacidad significa que una organización puede contratar a un proveedor IaaS para eliminar de forma efectiva los requerimientos de recursos de servidor de un centro de datos mientras mantienen la habilidad de entregar y consumir recursos a través de control y selección local.

Se entregan controles para gestionar recursos de red, almacenamiento y otros que límites preestablecidos entregados a cada cuenta. Esta es una herramienta de nivel administrativo a través de la cual se puede entregar recursos para desarrolladores que luego construirán aplicaciones para usuarios finales, permitiendo controlar la pirámide de servicios de nube desde un solo punto. (Hausman, Cook, & Sampaio, 2013)

La Figura 5 muestra algunos ejemplos de solución de gestión de Infraestructura como Servicio.

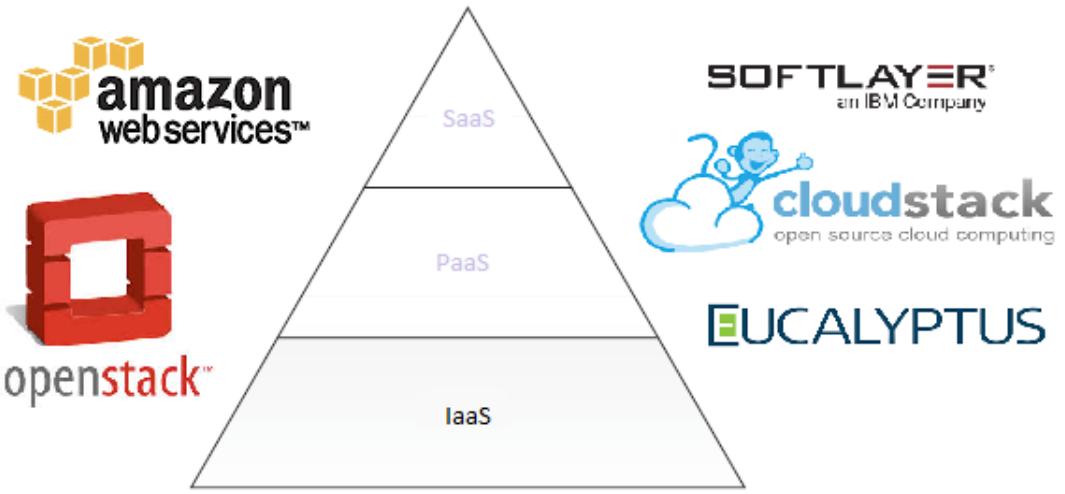

Figura 5. Ejemplos de Infraestructura como Servicio (IaaS)

2.5. MODELOS DE DESPLIEGUE DE COMPUTACIÓN EN LA NUBE

Según el Instituto Nacional de Estándares y Tecnología (NIST, por sus siglas en inglés) del Departamento de Comercio de los Estados Unidos de América, los modelos de despliegue de computación en la nube basan su clasificación en su audiencia y la ubicación física de los dispositivos tecnológicos que facilitan el servicio. (National Institute of Standards and Technology, 2011)

Basándose en estos parámetros, definen cuatro modelos comunes de despliegue de computación en la nube que son:

- Nube privada
- Nube comunitaria
- Nube pública
- Nube híbrida

A continuación se define cada uno de estos modelos.

2.5.1. NUBE PRIVADA

Un tipo de nube que es aprovisiona para ser utilizada por un usuario o grupo de usuarios dentro de una organización. Una nube privada es gestionada y operada por una organización que es su propietaria.

Las nubes privadas residen dentro de una red privada que es gestionada por la misma organización a la que pertenece.

El esquema de nube privada es usualmente el punto de entrada para la adopción de computación en la nube para las organizaciones, entregando

flexibilidad y monitoreo del consumo de recursos para cada uno de los sistemas ubicados en los centros de datos de la organización.

Este modelo de despliegue es usado con frecuencia cuando existen regulaciones y requerimientos legales que requieren un alto grado de control, seguimiento del acceso a los recursos y gobernabilidad de la infraestructura. La Figura 6 muestra un esquema de Nube privada.

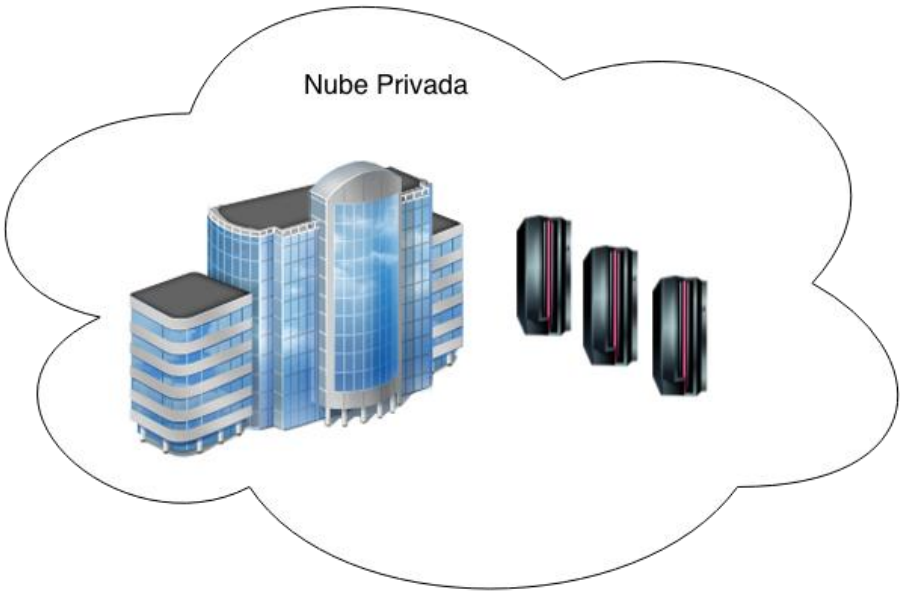

Figura 6. Nube Privada

2.5.2. NUBE COMUNITARIA

Una nube comunitaria es entregada para uso de un grupo de organizaciones relacionadas con propósitos compartidos, como por ejemplo instituciones educativas o gubernamentales que eligen usan una nube común de servicios que no están disponibles para el público en general. Las nubes comunitarias pueden encontrarse dentro de la infraestructura de una organización, que cuenta con una nube privada de servicios los cuales pueden ser consumidos remotamente por las organizaciones de la comunidad.

Nubes públicas particionadas también son ejemplos de nubes comunitarias, con servicios de nube públicos aislados del público en general restringiendo el acceso a esquemas de red específicos y otros controles. Las nubes comunitarias pueden ser usadas para obtener reutilización mejorada y compartimiento de recursos informáticos, brindando beneficios a cada uno de los miembros de la comunidad al facilitar el consumo de servicios que de otra forma tendrían que implantar, gestionar y mantener localmente. La Figura 7 muestra un esquema de Nube Comunitaria.

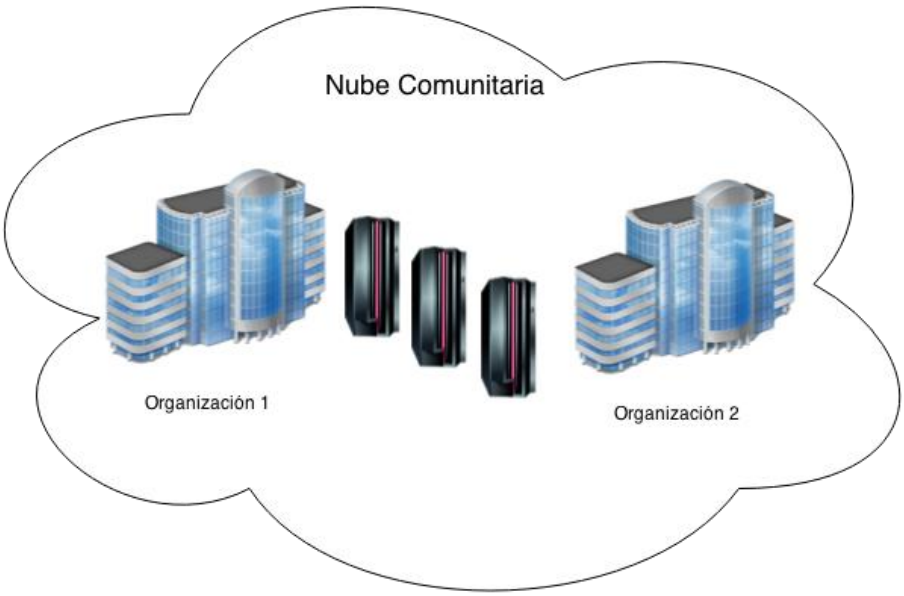

Figura 7. Nube Comunitaria

2.5.3. NUBE PÚBLICA

Una nube pública es entregada para el uso del público en general. Las nubes públicas representan el diseño más marcado de infraestructura virtual de nube, eliminando, parcial o completamente, los recursos informáticos de un centro de datos.

Las nubes públicas residen en recursos de centros de datos pertenecientes a proveedores que son accedidos por usuarios ubicados en cualquier parte del mundo a través conectividad pública a Internet.

La redirección transparente de servicios de nube pública hacia centros de datos en ubicaciones variadas crea preocupación para las organizaciones con regulaciones y legislación que exige el seguimiento y protección de los datos y la gobernanza de los mismos.

La Figura 8 muestra un esquema de Nube Pública.

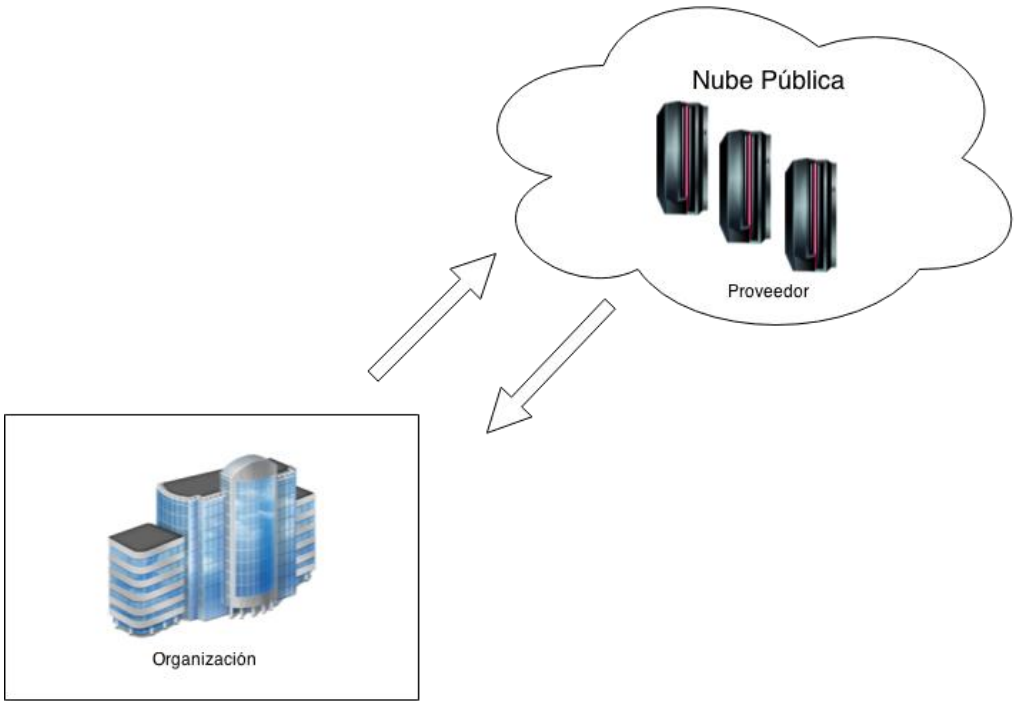

Figura 8. Nube Pública

2.5.4. NUBE HÍBRIDA

Una nube híbrida es entregada usando componentes de nubes públicas, comunitarias y/o privadas. La nube híbrida entrega acceso a dos o

más infraestructuras interconectadas por tecnologías estandarizadas o servicios de nube propietarios.

Las nubes híbridas, en resumen, son la mezcla de tipos de nube, permiten crecer bajo demanda expandiendo los límites de la nube privada. Los recursos de la nube privada local son usados para soportar una aplicación hasta que un pico en la demanda excede los límites de los recursos locales, en este punto la aplicación empieza a consumir recursos de nube pública que han sido configurados y contratados previamente para usarse en estos casos.

La Figura 9 muestra un esquema de Nube Híbrida.

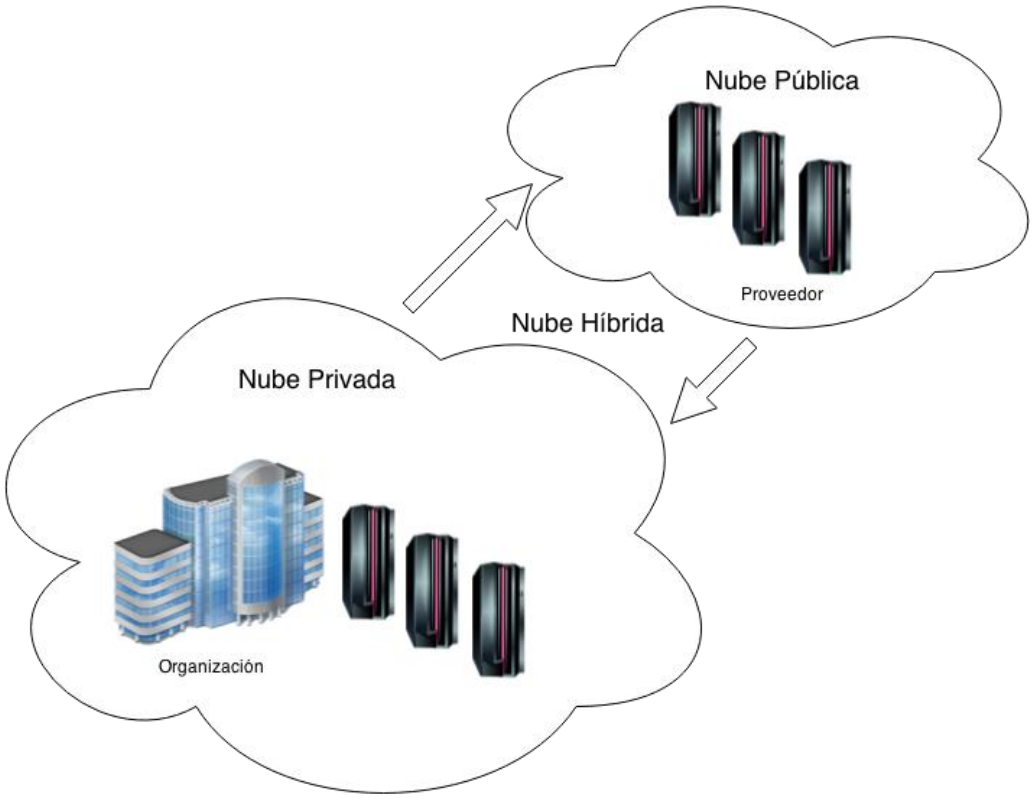

Figura 9. Nube Híbrida

CAPÍTULO 3. DESARROLLO

3.1. VALIDACIÓN DE SOLUCIONES A UTILIZAR

Para cumplir con los objetivos del proyecto es necesario contar con dos tecnologías que apoyen a las características requeridas.

Se requiere, entonces, contar con una solución de Infraestructura como Servicio y una solución de Plataforma como Servicio. Es importante que las soluciones puedan integrarse, siendo necesario analizar cuáles son los métodos y componentes que permiten la interacción entre las tecnologías para proveer Plataformas consumibles bajo demanda.

3.2. METODOLOGÍA DE EVALUACIÓN

Como referencia para definir la metodología de evaluación de las herramientas de software se ha elegido el grupo de normas ISO/IEC 25000.

Estas normas brindan una guía para el uso del nuevo estándar de Requisitos y Evaluación de Calidad de Sistemas y Software (System and Software Quality Requirements and Evaluation, SQuaRE, por sus siglas en inglés). SQuaRE establece criterios para la especificación de requerimientos de calidad de software, sus métricas y su evaluación. (ISO, 2014)

SQuaRE es el resultado del esfuerzo de las entidades de normalización que encontraron varios puntos en los que los estándares previos ISO/IEC 9126 (Calidad del Producto de Software) e ISO/IEC 14598 (Evaluación del producto de software) presentaban dificultades, por ejemplo:

- Ambos estándares tienen bases funcionales y normativas similares. Esto producía redundancia en algunos puntos y vacíos en otros.

- Ambos estándares forman un grupo complementario de normas.

- Al ser gestionados de forma independiente se crearon inconsistencias entre ambos estándares.

Partiendo de esto se crea SQuaRE, para crear una serie de estándares unificados y organizados de forma lógica. Cuenta con dos procesos principales:

- Especificación de requerimientos de calidad de sistemas y software.

- Evaluación de calidad de sistemas y software.

El propósito de SQuaRE es apoyar a quienes desarrollan y adquieren productos de software con la especificación y evaluación de la calidad de estos productos. Se establecen criterios de los requerimientos que deberían contemplarse como parte de una evaluación y se entregan recomendaciones del proceso que se debe seguir para ejecutar la evaluación. Incluye recomendaciones para alinear definiciones de calidad del cliente con atributos del proceso de desarrollo. Adicionalmente entrega características recomendadas y atributos de calidad del producto de software que pueden ser usadas por los interesados. (ISO, 2014)

Dentro de los elementos a evaluar recomendados por la norma se encuentran:

- **Funcionalidad:** propiedad de un sistema o software que define su capacidad para proporcionar funciones explícitas e implícitas cuando es usado bajo un marco de condiciones definidas.

    - **Idoneidad:** determina la capacidad del sistema o el software para entregar funciones apropiadas para usuarios especificados.

    - **Precisión:** determina la capacidad del sistema o el software para entregar exactamente las funciones esperadas por usuarios especificados.

    - **Interoperabilidad:** determina la capacidad del sistema o software para interactuar con uno o más sistemas o productos de software.

    - **Seguridad:** determina la capacidad del sistema o software para proteger la información que procesa de manera que solo las personas o sistemas autorizados puedan leerlos o modificarlos.

    - **Cumplimiento de la funcionalidad:** determina la capacidad del sistema o software para adherirse a normas, estándares y reglamentos relacionados con la funcionalidad.

- **Portabilidad:** propiedad de un sistema o software que representa la cantidad de esfuerzo requerido para transferirlo a diferentes entornos.

    - **Cumplimiento de la portabilidad:** determina la capacidad del sistema o software para adherirse a estándares o convenciones relacionadas con la portabilidad.

- **Facilidad de intercambio:** determina la capacidad del sistema o software para ser usado en lugar de otro producto, para el mismo propósito, en ese entorno.

- **Coexistencia:** determina la capacidad del sistema o software para coexistir con otro sistema o software independiente en un mismo ambiente compartiendo recursos.

- **Facilidad de instalación:** determina la capacidad del sistema o software para ser instalado en un entorno específico.

- **Adaptabilidad:** determina la capacidad del sistema o software para adaptarse a diferentes entornos entregando niveles adecuados de prestaciones.

- **Fiabilidad:** propiedad de un sistema o software que determina su capacidad para mantener un nivel esperado de prestaciones al ser usado bajo condiciones definidas.

  - **Madurez:** determina la capacidad del sistema o software para verificar posibles fallas y evitarlas.

  - **Tolerancia a fallos:** determina la capacidad del sistema o software para mantener sus prestaciones a pesar de producirse fallos.

  - **Capacidad de recuperación:** determina la capacidad del sistema o software para restablecer sus prestaciones y recuperar datos que se hayan visto comprometidos a causa de fallos.

  - **Cumplimiento de la fiabilidad:** determina la capacidad del sistema o software para cumplir normas, estándares y reglamentos relacionados con la fiabilidad.

- **Usabilidad:** propiedad de un sistema o software que determina su capacidad para ser entendido, usado, y amigable con el usuario cuando es usado bajo condiciones específicas.

  - **Inteligibilidad:** determina la capacidad del sistema o software para que un usuario lo entienda y pueda usarlo para tareas y casos de uso particulares.

  - **Facilidad de aprendizaje:** determina la capacidad del sistema o software para facilitar el aprendizaje del usuario.

  - **Facilidad de operación:** determina la capacidad del sistema o software para ser usado fácilmente por el usuario que requiere operar y controlar el producto.

  - **Atractivo:** determina la capacidad del sistema o software para ser atractivo al usuario.

  - **Cumplimiento de la usabilidad:** determina la capacidad del sistema o software para cumplir normas, estándares y reglamentos relacionados con la usabilidad.

- **Eficiencia:** propiedad de un sistema o software que mide su capacidad para proporcionar prestaciones apropiadas, relativas al uso adecuado de recursos usados bajo condiciones específicas.

- **Cumplimiento de la eficiencia:** determina la capacidad del sistema o software para adherirse a normas, estándares y reglamentos relacionados con la eficiencia.

- **Utilización de recursos:** determina la capacidad del sistema o software para usar y administrar recursos al llevar a cabo su función bajo condiciones determinadas.

- **Comportamiento en el tiempo:** determina la capacidad del sistema o software para cumplir con tiempos de respuesta y tiempos de proceso bajo condiciones determinadas.

- **Mantenibilidad:** propiedad de un sistema o software que representa la cantidad de esfuerzo requerida para conservar su funcionamiento normal o para restituirlo una vez se ha presentado un evento de falla.

  - **Cumplimiento de la mantenibilidad:** determinar la capacidad del sistema o software para cumplir a normas, estándares y reglamentos relacionados con la mantenibilidad.

  - **Facilidad de prueba:** determina la capacidad del sistema o software para permitir la evaluación de sus características.

  - **Estabilidad:** determina la capacidad del sistema o software para comportarse de forma esperada a pesar de las condiciones presentadas.

  - **Facilidad de modificación:** determina la capacidad del sistema o software para permitir que una modificación específica sea implementada.

- **Facilidad de análisis:** determina la capacidad del sistema o software para permitir la identificación de fallos y sus causas, o posibles puntos de mejora.

Estas características se deben ver traducidas a características relativas a soluciones de Infraestructura como Servicio y Plataforma como Servicio.

Para esto es necesario contar con una referencia de arquitectura de computación en la nube que permita determinar las características que se evaluarán en las opciones de solución.

3.3. ARQUITECTURAS DE REFERENCIA DE COMPUTACIÓN EN LA NUBE

Se toman como referencia dos referencias de arquitectura de computación en la nube que van a permitir contrastar las características que se espera que los distintos tipos de servicio y modelos de despliegue de computación en la nube posean.

Estas arquitecturas de referencia son:

- Arquitectura de referencia de computación en la nube de NIST
- Arquitectura de referencia de computación en la nube de IBM

3.3.1. ARQUITECTURA DE REFERENCIA DE COMPUTACIÓN EN LA NUBE DE NIST

La arquitectura de referencia de computación en la nube de NIST se encuentra detallada en el documento NIST Cloud Computing Reference

Architecture creado por el Departamento de Comercio de Estados Unidos. (National Institute of Standards and Technology, 2011)

Los elementos de arquitectura determinados por el NIST son:

- Despliegue del servicio
- Orquestación del servicio
- Gestión del servicio en la nube
- Seguridad
- Privacidad

A continuación se amplían los conceptos relacionados a cada uno de los elementos de arquitectura mencionados.

3.3.1.1. DESPLIEGUE DEL SERVICIO

Como se ha descrito en la sección de definiciones, los servicios en la nube pueden ser operados en varios modelos de despliegue que dependen de la ubicación y el tipo de servicio requerido. En otras palabras estos modelos están basados en qué tan exclusivos son los recursos tecnológicos que se ofertan para el consumidor.

Los modelos de despliegue como ya se ha mencionado son:

- Nube pública
- Nube privada
- Nube comunitaria
- Nube híbrida

3.3.1.2. ORQUESTACIÓN DEL SERVICIO

La orquestación del servicio se refiere a la composición de componentes de sistema que un proveedor de nube debe coordinar y gestionar para entregar recursos de cómputo y proveer servicio a sus clientes.

Una capa de servicio contiene las interfaces definidas por el proveedor para que sus clientes accedan a los servicios computacionales. Las interfaces dan acceso a cada uno de los modelos de servicio en la nube, SaaS, PaaS e IaaS.

Una capa de control y abstracción de recursos contiene los componentes que un proveedor usa para entregar y gestionar el acceso a los recursos de cómputo físicos que son abstraídos mediante software. Ejemplos de recursos abstraídos son: máquinas virtuales, hipervisores, almacenamiento virtualizado entre otros. En esta capa existen los medios para controlar todos los recursos físicos de acuerdo a su consumo.

Una capa de recursos físicos incluye los recursos de cómputo a nivel físico. En esta capa se encuentran recursos de hardware, red, almacenamiento, entre otros. Además esta capa incluye el manejo de instalaciones y edificios, componentes de aire acondicionado, energía, etc.

3.3.1.3. GESTIÓN DEL SERVICIO EN LA NUBE

Incluye todas las funciones relacionadas con el servicio que son necesarias para gestionar y operar los servicios requeridos y ofertados a los clientes. Esta sección incluye requerimientos de Soporte para el negocio,

configuración y aprovisionamiento; y portabilidad e interoperabilidad. A continuación se describe cada uno de estos grupos de requerimientos.

Soporte para el negocio

Incluye servicios relacionados con el negocio y los componentes usados para operar servicios que son usados por el cliente final. Dentro de este apartado existen los siguientes requerimientos:

- **Gestión de clientes:** gestión de cuentas, gestión de perfiles de usuario, gestión de relación con el cliente, resolución de problemas y requerimientos, etc.
- **Gestión de contratos:** gestión de contratos de servicio.
- **Gestión de inventario:** configuración y gestión de catálogos de servicio.
- **Contabilidad y Facturación:** gestión de información de facturación, envío de facturas, procesamiento de pagos, seguimiento de facturas, etc.
- **Auditoría y reportes:** monitoreo de operaciones del usuario, generación de reportes, etc.
- **Gestión de costos:** evaluación de servicios y determinación de precios, manejo de promociones y precios de acuerdo al perfil del usuario, etc.

Configuración y aprovisionamiento

- **Aprovisionamiento ágil:** entrega automatizada de sistemas en la nube, basada en las capacidades solicitadas.
- **Gestión de cambios:** ajustes de configuración y asignación de recursos para reparación, actualización e inclusión de nuevos nodos en la nube.

- **Monitoreo y reportes:** descubrimiento y monitoreo de recursos virtuales, monitoreo de operaciones y eventos; y generación de reportes de rendimiento.
- **Medición:** proveer capacidades de medición al nivel de abstracción apropiado para el tipo de servicio.
- **Gestión de niveles de servicio:** definición de contratos de nivel de servicio, monitoreo de cumplimiento de niveles de servicio de acuerdo a políticas definidas.

Portabilidad e interoperabilidad

Los requerimientos relacionados la facilidad para intercambiar información y recursos entre distintos proveedores de nube.

- **Portabilidad de datos:** la habilidad de los clientes para copiar objetos de información desde y hacia distintas nubes o de realizar transferencias de información entre plataformas.
- **Interoperabilidad de servicios:** la habilidad de los clientes para usar su información y servicios a través de múltiples proveedores de nube con una interfaz unificada de gestión.
- **Portabilidad de sistemas:** permite la migración de máquinas virtuales de un proveedor de nube a otro, así como la migración de aplicaciones, servicios y contenidos entre proveedores.

3.3.1.4. SEGURIDAD

La seguridad es un aspecto que debe ser reconocido como crítico en la arquitectura y está relacionado con todas las capas y componentes del modelo de referencia, pasando por la seguridad física hasta la seguridad de la aplicación. La seguridad no está relacionada solamente con los

proveedores de servicios sino también con los clientes que consumen dichos servicios, siendo la seguridad un campo de acción compartido.

Los sistemas basados en nube necesitan cubrir requerimientos de seguridad como: autenticación, autorización, disponibilidad, confidencialidad, gestión de identidad, integridad, auditoría, monitoreo de seguridad, respuesta ante incidentes, y gestión de políticas de seguridad.

3.3.1.5. PRIVACIDAD

Los proveedores de servicios en la nube deben proteger la información personal y de identificación de los clientes asegurando su debido uso, recopilación, procesamiento y comunicación.

Aunque la computación en la nube provee una solución flexible para compartir recursos, software e información, también crea retos adicionales en cuanto a la privacidad de los consumidores.

3.3.2. ARQUITECTURA DE REFERENCIA DE COMPUTACIÓN EN LA NUBE DE IBM

La arquitectura de referencia de computación en la nube de IBM se encuentra detallada en el documento IBM SmartCloud: Building a Cloud Enabled Data Center creado por la Corporación IBM. (IBM Corporation, 2013)

Los elementos de arquitectura determinados por IBM se resumen en una serie de requerimientos funcionales, llamados casos de uso y micropatrones, y requerimientos no funcionales.

Los micropatrones proveen una visión resumida de las capacidades esperadas de un servicio de nube, y están compuestos por casos de uso específicos. Adicionalmente existen interrelaciones entre micropatrones que habilitan la funcionalidad del servicio.

Los micropatrones y casos de uso son:

- **Virtualización:** incluye los casos de uso básicos para compartir recursos de hardware. Sus casos de uso son:
  - Virtualización de servidores
  - Virtualización básica de almacenamiento
  - Virtualización básica de red
  - Gestión de hipervisores
- **Monitoreo y gestión de eventos y capacidad:** incluye los casos de uso que permiten medir la salud de los componentes y servicios de nube entregados. Sus casos de uso son:
  - Gestión de eventos
  - Monitoreo
  - Gestión de capacidad
- **Capacidad de recuperación de datos:** incluye los casos de uso relacionados con el respaldo y recuperación de información en caso de producirse una falla o interrupción del servicio. Su caso de uso puntal es:
  - Respaldo y restauración
- **Gestión de imágenes:** incluye los casos de uso para el registro y manejo de imágenes de componentes de la infraestructura de nube. Su caso de uso de puntual es:
  - Gestión de imágenes

- **Automatización de autoservicio:** contiene los casos de uso requeridos para la entrega automatizada de recursos y servicios por medio del autoservicio. Sus casos de uso son:
    - Gestión de roles y autorización
    - Provisión de máquinas virtuales
    - Construcción de imágenes de máquina virtual
    - Gestión de nube
- **Contabilidad y medición:** incluye los casos de uso para la medición y seguimiento de consumo, así como los relacionados con la facturación por uso de recursos. Sus casos de uso son:
    - Medición de uso
    - Contabilidad
    - Cobro y facturación
- **Gestión de virtualización de almacenamiento y red:** contiene los casos de uso relacionados con funciones avanzados de gestión de red y almacenamiento. Sus casos de uso son:
    - Gestión de virtualización de almacenamiento
    - Gestión de virtualización de red
- **Automatización de autoservicio de red:** incluye los casos de uso relacionados con la automatización de la entrega de recursos de red. Sus casos de uso son:
    - Configuración y provisión de red
    - Gestión de nube
    - Gestión de roles y autorización
- **Automatización de autoservicio de almacenamiento:** incluye los casos de uso relacionados con la automatización de la entrega de recursos de almacenamiento. Sus casos de uso son:
    - Configuración y provisión de almacenamiento
    - Gestión de nube
    - Gestión de roles y autorización

- **Automatización de autoservicio de almacenamiento:** incluye los casos de uso relacionados con la automatización de la entrega de recursos de servicios complejos que cuentan con más de una máquina virtual y elementos de almacenamiento y red. Revisa también lo relacionado al rendimiento de los servicios. Sus casos de uso son:
    - Gestión de roles y autorización
    - Orquestación de servicios
    - Aprobación y negación de solicitudes
- **Integración con nube híbrida:** Incluye casos de uso para el despliegue de carga de trabajo en la nube pública, gestión del servicio y gobernanza en entornos híbridos. Sus caso de uso es:
    - Integración de nubes híbridas
- **Gestión del servicio de TI:** incluye los casos de uso que implementan procesos de ITIL para la gestión del servicio. Sus casos de uso son:
    - Gestión de problemas e incidentes
    - Gestión de activos de TI
    - Gestión de licencias
    - Gestión cambios y configuración
    - Mesa de servicios
    - Gestión de despliegues
- **Seguridad:** contiene los casos de uso para asegurar los distintos componentes y niveles de la infraestructura de nube y sus servicios. Sus casos de uso son:
    - Gestión de información y eventos de seguridad
    - Gestión de Accesos e identidad
    - Gestión de amenazas y vulnerabilidad

- Gestión de parches
- Gestión de cumplimiento de la seguridad

Los requerimientos no funcionales contemplados por la Arquitectura de Referencia de IBM se presentan en la Tabla 3 junto con algunos ejemplos de requerimientos y medición.

Tabla 3. Requerimientos no funcionales

Categoría	Ejemplos de requerimiento
Fiabilidad, Disponibilidad, y Utilidad	- Alta disponibilidad y fiabilidad de gestión de nube e infraestructura gestionada.
	- Recuperación antes desastres de gestión de nube e infraestructura gestionada
Rendimiento	- Tiempo de respuesta de interfaz de usuario - Tiempo de provisión de máquinas virtuales. - Tiempo de provisión de servicios.
Escalabilidad	- Número de usuarios concurrentes - Número de máquinas virtuales y servicios provistos por minuto / hora - Soporte para centros de datos múltiples.
Usabilidad y facilidad de consumo	- Usabilidad de interfaz de usuario - Tiempo requerido para obtener valor (Time to value, TTV) - Retorno de la inversión (ROI) - Costo total de propiedad (TCO)

Continúa

Categoría	Ejemplos de requerimiento
Extensibilidad	- Soporte para conexión de nuevos hipervisores - Extensión de servicio con nuevas acciones - Personalización de interfaces de usuario - Múltiples propietarios
Seguridad	- Soporte para estándares de seguridad y mejores prácticas

Fuente: (IBM Corporation, 2013)

3.4. TABLAS COMPARATIVAS

Para realizar la comparación entre las soluciones de Infraestructura como Servicio y Plataforma como Servicio es necesario contar con una tabla que relacione los aspectos especificados por la metodología SQuaRe y los elementos de las arquitecturas de referencia de computación en la nube analizadas en el apartado anterior.

La Tabla 4 contiene los puntos a valorar con respecto a las soluciones de Infraestructura como Servicio.

Tabla 4. Tabla comparativa de Infraestructura como Servicio

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube
Funcionalidad	Idoneidad	Virtualización de servidores
		Virtualización básica de almacenamiento
		Virtualización básica de red
		Gestión de hipervisores
		Orquestación de servicios
	Precisión	Gestión de eventos

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube
		Monitoreo
		Gestión de capacidad
	Interoperabilidad	Interfaz de programación de aplicaciones (API)
		Gestión de imágenes
	Seguridad	Gestión de información y eventos de seguridad
		Gestión de Accesos e identidad
		Gestión de amenazas y vulnerabilidad
		Gestión de parches
		Soporte para estándares de seguridad y mejores prácticas
		Gestión de roles y autorización
	Cumplimiento de la funcionalidad	Nivel de cumplimiento de la funcionalidad
	General	Satisfacción general
	Subtotal Funcionalidad	
Portabilidad	Adaptabilidad	Arquitectura distribuida y asíncrona
		Uso de sistemas operativos
	Facilidad de intercambio	Portabilidad de datos
		Interoperabilidad de servicios
		Portabilidad de sistemas
	Coexistencia	Integración de nubes híbridas
		Uso de lenguajes de programación variados
		Uso de hipervisores
	Facilidad de instalación	Complejidad del proceso de instalación
		Manuales de instalación
		Compatibilidad con arquitecturas variadas
	Cumplimiento de la portabilidad	Nivel de cumplimiento de la portabilidad
	General	Satisfacción general
	Subtotal Portabilidad	
Fiabilidad	Madurez	Tiempo en el mercado
		Historial
		Actualizaciones
		Registro de modificaciones
		Gestión de problemas e incidentes

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube
	Tolerancia a fallos	Manejo de errores y excepciones
		Alta disponibilidad
		Recuperación ante fallos
		Respaldo de información y configuración
	Capacidad de recuperación	Recuperación de estado previo a errores
		Recuperación de datos
		Recuperación de configuración
	Cumplimiento de la fiabilidad	Nivel de cumplimiento de la fiabilidad
	General	Satisfacción general
	Subtotal Fiabilidad	
Usabilidad	Inteligibilidad	Usabilidad de interfaz de usuario
		Configuración y provisión de almacenamiento
		Gestión de nube
	Facilidad de aprendizaje	Documentación oficial
		Participación comunidad
		Cursos presenciales y/o online
	Facilidad de operación	Medición de uso
		Contabilidad
		Cobro y facturación
		Múltiples propietarios
	Atractivo	Personalización de interfaces de usuario
		Facilidad de navegación
		Interfaz visualmente agradable
	Cumplimiento de la usabilidad	Nivel de cumplimiento de la usabilidad
	General	Satisfacción general
	Subtotal Usabilidad	
Eficiencia	Comportamiento en el tiempo	Respuesta de interfaz de usuario
		Provisión de máquinas virtuales
		Provisión de servicios
	Utilización de recursos	Recursos usados para ejecución
		Gestión de recursos
	Cumplimiento de la eficiencia	Nivel de cumplimiento de la eficiencia
	General	Satisfacción general

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube
	Subtotal Eficiencia	
Mantenibilidad	Facilidad de análisis	Facilidad para detectar fallos
		Facilidad para detectar errores de uso
	Facilidad de prueba	Facilidad para ejecutar casos de prueba
	Estabilidad	Cantidad de modificaciones
		El sistema puede funcionar a pesar de cambios
	Facilidad de modificación	Soporte para conexión de nuevos hipervisores
		Extensión de servicio con nuevas acciones
		Acceso a código fuente
	Cumplimiento de la mantenibilidad	Nivel de cumplimiento de la mantenibilidad
	General	Satisfacción general
	Subtotal Mantenibilidad	
	Totales	

La Tabla 5 contiene los puntos a valorar con respecto a las soluciones de Plataforma como Servicio.

Tabla 5. Tabla comparativa de Plataforma como Servicio

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube
Funcionalidad	Idoneidad	Despliegue portable entre infraestructura
		Reutilización de componentes
		Compilación y construcción de aplicaciones
		Aplicaciones soportadas
	Precisión	Versionamiento de Plataforma
	Interoperabilidad	Interfaz de programación de aplicaciones (API)
		Compartir plataformas
		Gestión de imágenes

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube
	Seguridad	Gestión de información y eventos de seguridad
		Gestión de Accesos e identidad
		Gestión de amenazas y vulnerabilidad
		Gestión de parches
		Soporte para estándares de seguridad y mejores prácticas
		Gestión de roles y autorización
	Cumplimiento de la funcionalidad	Nivel de cumplimiento de la funcionalidad
	General	Satisfacción general
	Subtotal Funcionalidad	
Portabilidad	Adaptabilidad	Arquitectura distribuida y asíncrona
		Uso de sistemas operativos
	Facilidad de intercambio	Portabilidad de datos
		Interoperabilidad de servicios
		Portabilidad de sistemas
	Coexistencia	Integración de nubes híbridas
		Uso de lenguajes de programación variados
		Plataformas variadas
	Facilidad de instalación	Complejidad del proceso de instalación
		Manuales de instalación
		Compatibilidad con arquitecturas variadas
	Cumplimiento de la portabilidad	Nivel de cumplimiento de la portabilidad
	General	Satisfacción general
	Subtotal Portabilidad	
Fiabilidad	Madurez	Tiempo en el mercado
		Historial
		Actualizaciones
		Registro de modificaciones
		Gestión de problemas e incidentes
	Tolerancia a fallos	Manejo de errores y excepciones
		Alta disponibilidad
		Recuperación ante fallos

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube
		Respaldo de información y configuración
	Capacidad de recuperación	Recuperación de estado previo a errores
		Recuperación de datos
		Recuperación de configuración
	Cumplimiento de la fiabilidad	Nivel de cumplimiento de la fiabilidad
	General	Satisfacción general
	Subtotal Fiabilidad	
Usabilidad	Inteligibilidad	Usabilidad de interfaz de usuario
		Administración de plataforma
		Gestión de componentes
	Facilidad de aprendizaje	Documentación oficial
		Participación comunidad
		Cursos presenciales y/o online
	Facilidad de operación	Medición de uso
		Contabilidad
		Cobro y facturación
		Múltiples propietarios
	Atractivo	Personalización de interfaces de usuario
		Facilidad de navegación
		Interfaz visualmente agradable
	Cumplimiento de la usabilidad	Nivel de cumplimiento de la usabilidad
	General	Satisfacción general
	Subtotal Usabilidad	
Eficiencia	Comportamiento en el tiempo	Respuesta de interfaz de usuario
		Provisión de plataformas
		Provisión de servicios
	Utilización de recursos	Recursos usados para ejecución
		Gestión de recursos
	Cumplimiento de la eficiencia	Nivel de cumplimiento de la eficiencia
	General	Satisfacción general
	Subtotal Eficiencia	
Mantenibilidad	Facilidad de análisis	Facilidad para detectar fallos
		Facilidad para detectar errores de uso

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube
	Facilidad de prueba	Facilidad para ejecutar casos de prueba
	Estabilidad	Cantidad de modificaciones
		El sistema puede funcionar a pesar de cambios
	Facilidad de modificación	Soporte para nuevas plataformas
		Extensión de servicio con nuevas acciones
		Acceso a código fuente
	Cumplimiento de la mantenibilidad	Nivel de cumplimiento de la mantenibilidad
	General	Satisfacción general
	Subtotal Mantenibilidad	
	Totales	

Para validar las soluciones a utilizar se realizará un análisis basado en las tablas comparativas definidas donde se registrarán las características deseadas para cada una de las soluciones y el cumplimiento que cada una de las alternativas tiene. Los rangos de cumplimiento están determinados en la escala descrita en la Tabla 6.

Tabla 6. Escala de valores para comparación

Valor	Descripción
3	Excelente
2	Aceptable
1	Deficiente
0	No Cumple

- Criterio para el valor 3

Se asignará el valor de 3 si es máximo el cumplimiento del parámetro establecido para la evaluación de las soluciones.

- Criterio para el valor 2

Se asignará el valor de 2 si el cumplimiento del parámetro para la evaluación de las soluciones se realiza pero tiene pequeñas faltas, deficiencias o errores.

- Criterio para el valor 1

Se asignará el valor de 1 si el cumplimiento del parámetro para la evaluación de las soluciones, no cumple con las expectativas.

- Criterio para el valor 0

Se asignará el valor de 0 si el cumplimiento del parámetro para la evaluación de las soluciones, no cumple en absoluto.

3.5. PRESELECCIÓN DE HERRAMIENTAS

Como punto inicial de la comparativa de herramientas es necesario seleccionar a los candidatos que formarán parte del análisis.

En la presente sección se detallan los criterios que se han seguido para elegir a las herramientas y se realiza una breve descripción de cada una de las herramientas candidatas.

Para seleccionar las herramientas candidatas, se han tomado como referencia los resultados de la encuesta llevada a cabo por Linux.com y The New Stack (Williams, 2014). La encuesta se llevó a cabo en un periodo de 2 semanas en el mes de Julio de 2014. A los encuestados se les preguntó sobre los mejores proyectos de código abierto para la nube en diferentes categorías, dentro de las cuales se presentaron IaaS y PaaS.

En general, los encuestados fueron consultados sobre los que ellos consideran primero, segundo y tercero de los proyectos de código abierto en cada categoría. El resultado fue determinado por la siguiente fórmula.

Número de votos como primera opción x 3
+ Número de votos como segunda opción x 2
+ Número de votos como tercera opción
= Puntaje total

Los resultados obtenidos en lo referente a IaaS y PaaS se detallan a continuación.

3.5.1. INFRAESTRUCTURA COMO SERVICIO

La Figura 10 muestra el resultado de la encuesta para de los mejores proyectos de código abierto para Infraestructura como Servicio.

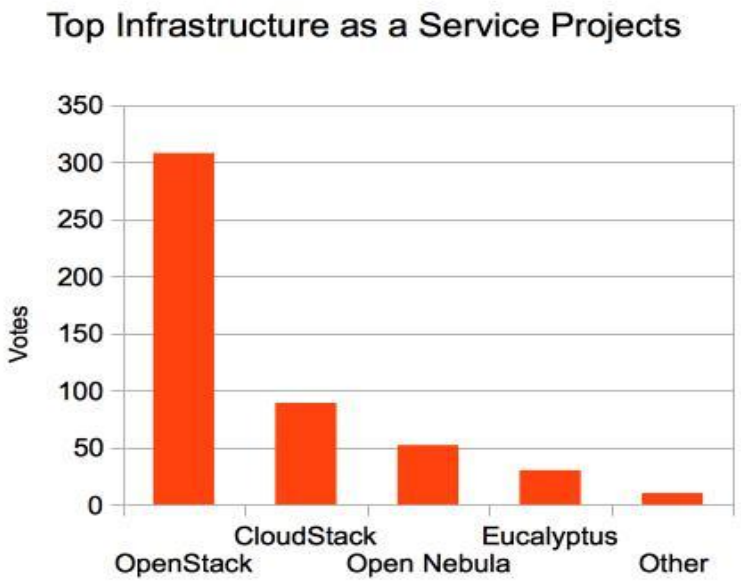

Figura 10. Proyectos de Infraestructura como Servicio más populares, según estudio de Linux.com y The New Stack.
Fuente: (Williams, 2014)

Adicionalmente se incluye como opción a VMware que es uno de los productos privativos más populares y que normalmente se toma como referencia en temas de virtualización.

La lista de herramientas para IaaS preseleccionadas es:

- OpenStack
- CloudStack
- Eucalyptus
- OpenNebula
- VMware

A continuación se presenta una breve descripción de cada una de las herramientas IaaS preseleccionadas.

3.5.1.1. OPENSTACK

El software de OpenStack controla grupos de recursos de cómputo, almacenamiento y redes en un centro de datos, gestionados por medio de su interfaz gráfica o a través de una API. OpenStack trabaja con tecnologías empresariales y de código abierto que le permiten trabajar con infraestructura heterogénea.

Muchas empresas se apoyan en OpenStack para habilitar sus negocios, en busca de reducir costos y adaptarse con rapidez a los cambios. OpenStack tiene un amplio “ecosistema”, y varias opciones asociadas de soporte comercial de productos y servicios potenciados por sus capacidades.

El software es construido por una amplia comunidad de desarrolladores, en colaboración con usuarios y es diseñado en eventos de forma colaborativa y abierta. (OpenStack, 2014)

3.5.1.2. CLOUDSTACK

Apache CloudStack es software de código abierto diseñado para desplegar y gestionar grandes redes de máquinas virtuales, como una plataforma de computación en la nube de Infraestructura como Servicio, altamente disponible y altamente escalable. CloudStack es usado por gran número de proveedores de servicio para ofrecer servicios de nube pública, y por compañías para ofrecer servicios de nube privada o como parte de una solución de nube híbrida.

Dentro de sus características ofrece orquestación de cómputo, red como servicio, gestión de usuarios y cuentas, APIs abiertas y nativas, contabilidad de recursos e interfaces gráficas de usuario. (Apache Software Foundation, 2014)

3.5.1.3. EUCALYPTUS

Eucalyptus es una arquitectura de software basada en Linux que implementa nubes híbridas y privadas escalables dentro de su infraestructura de TI existente. Eucalyptus permite usar sus propios recursos de hardware, almacenamiento y red, usando una interfaz de autoservicio bajo demanda.

Es posible desplegar una nube de Eucalyptus dentro del centro de datos local de cada organización para mantener los datos sensibles bajo control.

Eucalyptus fue diseñado para ser fácil de instalar y busca ser poco intrusivo. Eucalyptus fue recientemente adquirido por HP. (Eucalyptus, 2014)

3.5.1.4. OPENNEBULA

OpenNebula busca entregar simplicidad para la nube empresarial privada e híbrida. Combina tecnologías de virtualización existentes con características avanzadas para soporte a usuarios, elasticidad y provisión automática, siguiendo un acercamiento guiado por las necesidades de administradores de sistemas, personal de operaciones y desarrolladores.

OpenNebula permite la evolución hacia la nube usando infraestructura existente, protegiendo las inversiones previas y evitando la dependencia de proveedores.

OpenNebula busca ser la plataforma más simple de nube para las empresas trayendo simplicidad para la nube empresarial privada e híbrida. (OpenNebula, 2014)

3.5.1.5. VMWARE VCLOUD

VMware es una empresa que forma parte de la corporación EMC y es una de las marcas más reconocidas en el mercado de la virtualización. Con presencia fuerte en la industria VMware ofrece soluciones propietarias y también maneja proyectos gratuitos. En el campo del código abierto ha colaborado con proyectos importantes en la industria como Cloud Foundry.

La solución de nube privada de VMware vCloud es una iniciativa que permite a los clientes gestionar sus entornos privados y escalar de forma sencilla hacia nubes públicas que manejen tecnologías de VMware.

Si bien no se trata de una opción de código abierto es un punto de referencia importante para el proyecto tomando en cuenta su importancia dentro de la industria de soluciones de virtualización y nube. (VMware, Inc., 2014)

3.5.2. PLATAFORMA COMO SERVICIO

Dentro del mercado de soluciones de Plataforma como Servicio existen varias opciones dentro de las cuales destacan dos proyectos que han ganado mucha relevancia principalmente por el fuerte apoyo empresarial y corporativo con el que cuentan. Estos proyectos son Cloud Foundry y OpenShift. La Figura 11 evidencia esta condición.

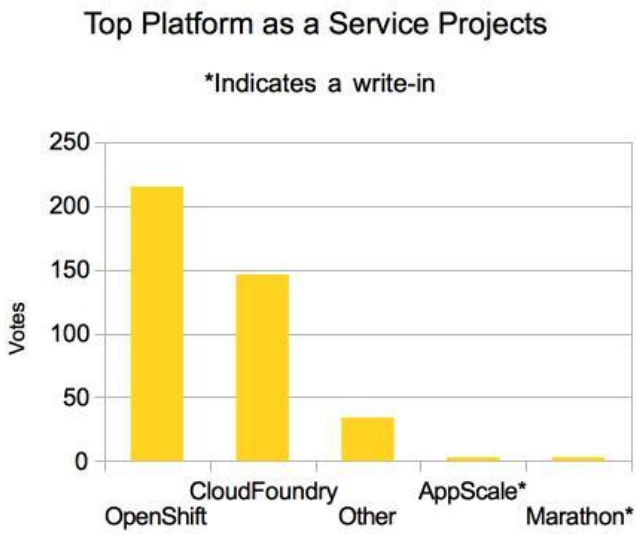

Figura 11. Proyectos de Plataforma como Servicio más populares, según estudio de Linux.com y The New Stack.
Fuente: (Williams, 2014)

En la Figura 11 se puede apreciar que son pocos los proyectos con relevancia suficiente como para competir en posicionamiento con los dos principales.

Sin embargo existen proyectos relacionados con las tecnologías que facilitan la entrega de Plataforma como Servicio que han ido ganando relevancia en el presente año por las facilidades y la visión que entregan.

Los hipervisores de código abierto y los proyectos relacionados con contenedores siguen ganando presencia en la industria lo cual se ve reflejado en la Figura 12 que forma parte del estudio de Linux.com y The New Stack citado a lo largo de este capítulo.

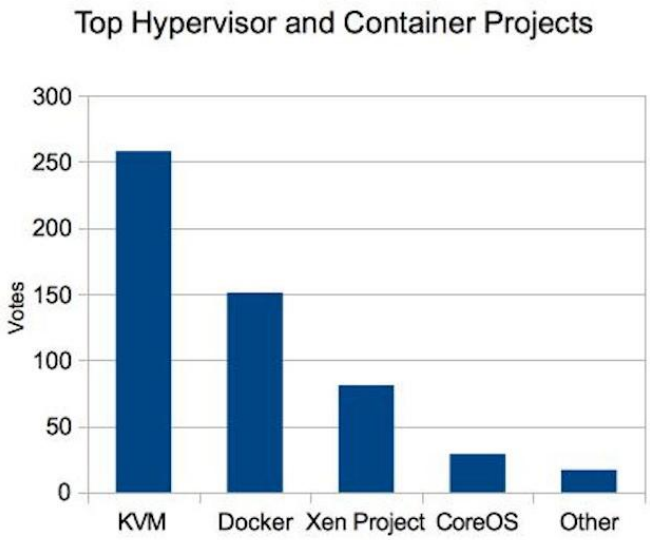

Figura 12. Proyectos de hipervisor y contenedores más populares según estudio de Linux.com y The New Stack.
Fuente: (Williams, 2014)

Se puede apreciar principalmente que las tecnologías asociadas a contenedores han ganado mucha aceptación por todas las ventajas teóricas que ofrecen aunque siguen en pleno desarrollo. En esta categoría la

tecnología más popular es Docker, que en la figura se ubica en segundo puesto superando a proyectos con mayor trayectoria y madurez.

Tomando en cuenta los resultados del estudio citado se han elegido para el análisis comparativo los siguientes proyectos.

- Docker
- Cloud Foundry
- OpenShift

A continuación se presenta una breve descripción de cada una de las herramientas PaaS preseleccionadas.

3.5.2.1. DOCKER

Docker es una plataforma para desarrolladores y administradores de sistemas para construir, desplegar y ejecutar aplicaciones distribuidas.

Compuesta por el motor de Docker, una herramienta de empaquetado y ejecución; y el concentrador de Docker, un servicio en la nube para compartir aplicaciones y automatizar flujos de trabajo; Docker permite que las aplicaciones sean construidas rápidamente desde componentes y elimina la fricción que existe entre los ambientes de desarrollo, control de calidad y producción. Como resultado, el área de TI puede desplegar rápidamente y ejecutar la misma aplicación sin cambios, en máquinas de escritorio, máquinas virtuales del centro de datos y cualquier nube.

Un contenedor de Docker comprende solamente la aplicación y sus dependencias. Se ejecuta como proceso aislado en un espacio de usuario dentro del sistema operativo anfitrión, compartiendo el kernel con los otros

contenedores. De esta forma, un contenedor disfruta de los beneficios de ubicación y aislamiento de recursos de las máquinas virtuales pero de una forma más eficiente y portable. (Docker, Inc., 2014)

3.5.2.2. CLOUD FOUNDRY

Cloud Foundry se hace llamar la Plataforma como Servicio abierta de la industria. Entrega una selección de nubes, marcos de trabajo y servicios de aplicación que pueden ser consumidos bajo demanda. Es un proyecto de código abierto que cuenta con una amplia comunidad de desarrolladores y patrocinio corporativo.

Cloud Foundry es parte de Pivotal Software, que a su vez es una iniciativa creada por VMware y EMC Corporation.

Cloud Foundry es apoyado por la Fundación Cloud Foundry que busca establecer y sostener al proyecto como la tecnología PaaS de código abierto estándar de la industria. (Pivotal Software Inc., 2014)

3.5.2.3. OPENSHIFT

OpenShift es la Plataforma como Servicio de Red Hat, que persigue facilitar las tareas de desarrollo, alojamiento y escalamiento de aplicaciones en ambientes de computación en la nube. Ofrece opciones de servicios públicos, privados y de código abierto.

El proyecto de código abierto recibe el nombre de OpenShift Origin y está disponible públicamente para ser utilizado. Este proyecto es la base de todas las ofertas de servicio de Red Hat asociadas a PaaS. Es un proyecto

que tiene fácil adopción para empresas que utilizan soluciones de Red Hat. (Red Hat, Inc., 2014)

3.6. COMPARATIVA DE CARACTERÍSTICAS

A continuación se detalla el análisis de las herramientas preseleccionadas contra las características deseadas para la solución de Infraestructura como Servicio y de Plataforma como Servicio.

3.6.1. COMPARACIÓN DE SOLUCIONES DE INFRAESTRUCTURA COMO SERVICIO

La Tabla 7 presenta la comparación de las características de las soluciones de Infraestructura como Servicio y los resultados que cada una ha obtenido por subcaracterística y requerimiento específico.

Tabla 7. Tabla comparativa de Infraestructura como Servicio

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube	CloudStack	Eucalyptus	OpenStack	OpenNebula	VMware	Total
Funcionalidad	Idoneidad	Virtualización de servidores	3	3	3	3	3	3
		Virtualización básica de almacenamiento	2	2	3	2	3	3
		Virtualización básica de red	3	3	3	2	3	3
		Gestión de hipervisores	3	3	3	2	1	3
		Orquestación de servicios	3	3	3	3	3	3
	Precisión	Gestión de eventos	2	2	3	2	3	3
		Monitoreo	2	2	3	2	3	3
		Gestión de capacidad	3	3	3	2	3	3
	Interoperabilidad	Interfaz de programación de aplicaciones (API)	3	3	3	3	2	3
		Gestión de imágenes	3	3	3	2	3	3

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube	CloudStack	Eucalyptus	OpenStack	OpenNebula	VMware	Total
	Seguridad	Gestión de información y eventos de seguridad	2	2	3	2	3	3
		Gestión de Accesos e identidad	3	3	3	2	3	3
		Gestión de amenazas y vulnerabilidad	3	3	2	3	3	3
		Gestión de parches	3	3	2	2	3	3
		Soporte para estándares de seguridad y mejores prácticas	3	3	3	3	3	3
		Gestión de roles y autorización	2	2	3	3	3	3
	Cumplimiento de la funcionalidad	Nivel de cumplimiento de la funcionalidad	3	3	3	2	3	3
	General	Satisfacción general	3	3	3	2	3	3
	Subtotal Funcionalidad		49	49	52	42	51	54
Portabilidad	Adaptabilidad	Arquitectura distribuida y asíncrona	2	3	3	2	3	3
		Uso de sistemas operativos	3	3	3	2	3	3
	Facilidad de intercambio	Portabilidad de datos	3	2	3	2	3	3
		Interoperabilidad de servicios	2	3	3	2	3	3
		Portabilidad de sistemas	2	3	3	2	3	3
	Coexistencia	Integración de nubes híbridas	2	3	3	2	2	3
		Uso de lenguajes de programación variados	3	3	3	2	2	3

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube	CloudStack	Eucalyptus	OpenStack	OpenNebula	VMware	Total
		Uso de hipervisores	3	3	3	2	1	3
	Facilidad de instalación	Complejidad del proceso de instalación	3	2	1	3	3	3
		Manuales de instalación	3	3	3	3	3	3
		Compatibilidad con arquitecturas variadas	3	3	3	2	3	3
	Cumplimiento de la portabilidad	Nivel de cumplimiento de la portabilidad	3	3	3	2	3	3
	General	Satisfacción general	3	3	3	2	3	3
	Subtotal Portabilidad		35	37	37	28	35	39
Fiabilidad	Madurez	Tiempo en el mercado	2	3	2	2	3	3
		Historial	3	2	3	2	3	3
		Actualizaciones	2	3	3	2	3	3
		Registro de modificaciones	3	3	3	2	3	3
		Gestión de problemas e incidentes	2	2	3	2	3	3
	Tolerancia a fallos	Manejo de errores y excepciones	3	3	3	2	3	3
		Alta disponibilidad	2	2	3	2	3	3
		Recuperación ante fallos	2	3	3	2	3	3
		Respaldo de información y configuración	3	2	3	2	3	3
	Capacidad de recuperación	Recuperación de estado previo a errores	2	3	3	2	3	3

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube	CloudStack	Eucalyptus	OpenStack	OpenNebula	VMware	Total
		Recuperación de datos	2	3	3	2	3	3
		Recuperación de configuración	3	3	3	2	3	3
	Cumplimiento de la fiabilidad	Nivel de cumplimiento de la fiabilidad	2	2	3	2	3	3
	General	Satisfacción general	2	2	3	2	3	3
	Subtotal Fiabilidad		33	36	41	28	42	42
Usabilidad	Inteligibilidad	Usabilidad de interfaz de usuario	3	3	2	2	3	3
		Configuración y provisión de almacenamiento	2	3	3	2	3	3
		Gestión de nube	2	2	3	2	3	3
	Facilidad de aprendizaje	Documentación oficial	2	2	3	2	3	3
		Participación comunidad	3	2	3	2	0	3
		Cursos presenciales y/o online	2	2	3	2	3	3
	Facilidad de operación	Medición de uso	3	3	3	2	3	3
		Contabilidad	2	2	3	2	3	3
		Cobro y facturación	2	2	3	2	3	3
		Múltiples propietarios	3	2	3	2	3	3
	Atractivo	Personalización de interfaces de usuario	2	2	3	2	3	3
		Facilidad de navegación	3	3	2	2	3	3
		Interfaz visualmente agradable	3	3	3	3	3	3

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube	CloudStack	Eucalyptus	OpenStack	OpenNebula	VMware	Total
	Cumplimiento de la usabilidad	Nivel de cumplimiento de la usabilidad	2	2	3	2	3	3
	General	Satisfacción general	3	3	3	2	3	3
	Subtotal Usabilidad		37	36	43	31	42	45
Eficiencia	Comportamiento en el tiempo	Respuesta de interfaz de usuario	3	3	3	3	3	3
		Provisión de máquinas virtuales	3	3	3	3	3	3
		Provisión de servicios	3	3	3	2	3	3
	Utilización de recursos	Recursos usados para ejecución	3	3	3	3	3	3
		Gestión de recursos	3	3	3	2	3	3
	Cumplimiento de la eficiencia	Nivel de cumplimiento de la eficiencia	3	3	3	3	3	3
	General	Satisfacción general	3	3	3	3	3	3
	Subtotal Eficiencia		21	21	21	19	21	21
Mantenibilidad	Facilidad de análisis	Facilidad para detectar fallos	3	3	3	2	3	3
		Facilidad para detectar errores de uso	3	3	3	2	3	3
	Facilidad de prueba	Facilidad para ejecutar casos de prueba	3	3	3	2	3	3
	Estabilidad	Cantidad de modificaciones	2	2	3	2	3	3
		El sistema puede funcionar a pesar de cambios	2	2	3	3	3	3

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube	CloudStack	Eucalyptus	OpenStack	OpenNebula	VMware	Total
	Facilidad de modificación	Soporte para conexión de nuevos hipervisores	2	2	3	3	3	3
		Extensión de servicio con nuevas acciones	2	2	3	2	3	3
		Acceso a código fuente	3	3	3	3	0	3
	Cumplimiento de la mantenibilidad	Nivel de cumplimiento de la mantenibilidad	3	3	3	3	2	3
	General	Satisfacción general	3	3	3	3	2	3
	Subtotal Mantenibilidad		26	26	30	25	25	30
	Totales		201	205	224	173	216	231

3.6.1.1. RESUMEN DE RESULTADOS DE INFRAESTRUCTURA COMO SERVICIO

La Tabla 8 muestra el resumen del puntaje obtenido por cada una de las soluciones de Infraestructura como servicio que han sido analizadas.

Tabla 8. Resumen de puntaje obtenido para Infraestructura como Servicio

Característica	CloudStack	Eucalyptus	OpenStack	OpenNebula	VMware	Total
Funcionalidad	49	49	52	42	51	54
Portabilidad	35	37	37	28	35	39
Fiabilidad	33	36	41	28	42	42
Usabilidad	37	36	43	31	42	45
Eficiencia	21	21	21	19	21	21
Mantenibilidad	26	26	30	25	25	30

La Figura 13 permite tener una visualización de los resultados obtenidos de la tabla comparativa de Infraestructura como Servicio en cada una de sus características.

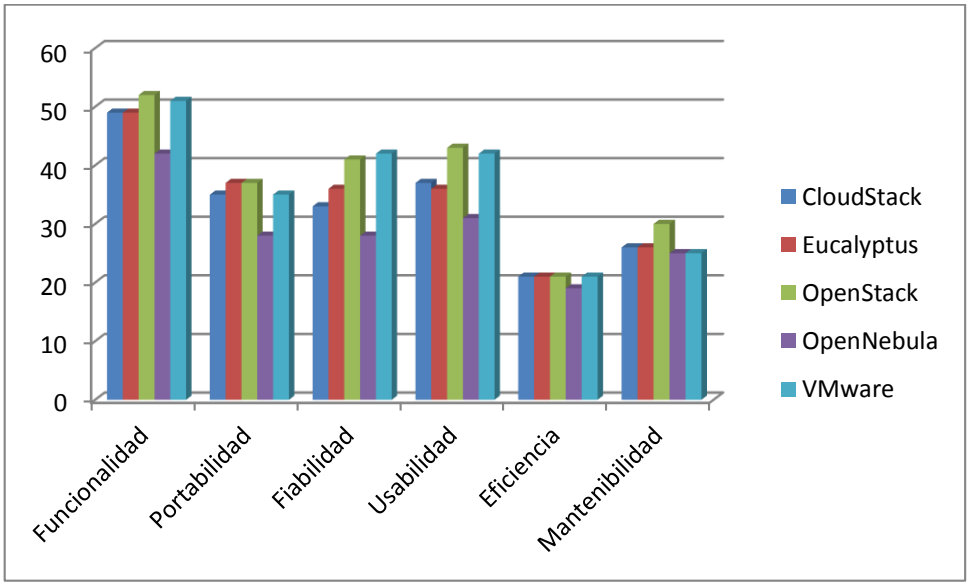

Figura 13. Resultados de Infraestructura como Servicio por característica

3.6.2. COMPARACIÓN DE SOLUCIONES DE PLATAFORMA COMO SERVICIO

La Tabla 9 presenta la comparación de las características de las soluciones de Plataforma como Servicio y los resultados que cada una ha obtenido por subcaracterística y requerimiento específico.

Tabla 9. Tabla comparativa de Plataforma como Servicio

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube	OpenShift	Cloud Foundry	Docker	Total
Funcionalidad	Idoneidad	Despliegue portable entre infraestructura	2	2	3	3
		Reutilización de componentes	3	3	3	3
		Compilación y construcción de aplicaciones	3	3	3	3
		Aplicaciones soportadas	3	3	3	3
	Precisión	Versionamiento de Plataforma	3	3	3	3
	Interoperabilidad	Interfaz de programación de aplicaciones (API)	3	3	3	3
		Compartir plataformas	2	2	3	3
		Gestión de imágenes	3	3	3	3
	Seguridad	Gestión de información y eventos de seguridad	3	3	2	3
		Gestión de Accesos e identidad	2	2	3	3
		Gestión de amenazas y vulnerabilidad	3	3	2	3

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube	OpenShift	Cloud Foundry	Docker	Total
		Gestión de parches	2	2	3	3
		Soporte para estándares de seguridad y mejores prácticas	2	2	2	3
		Gestión de roles y autorización	2	2	2	3
	Cumplimiento de la funcionalidad	Nivel de cumplimiento de la funcionalidad	2	2	3	3
	General	Satisfacción general	3	3	3	3
	Subtotal Funcionalidad		41	41	44	48
Portabilidad	Adaptabilidad	Arquitectura distribuida y asíncrona	3	3	3	3
		Uso de sistemas operativos	2	2	2	3
	Facilidad de intercambio	Portabilidad de datos	3	3	3	3
		Interoperabilidad de servicios	3	3	3	3
		Portabilidad de sistemas	2	2	3	3
	Coexistencia	Integración de nubes híbridas	2	3	3	3
		Uso de lenguajes de programación variados	3	2	3	3
		Plataformas variadas	3	3	3	3
	Facilidad de instalación	Complejidad del proceso de instalación	2	2	3	3
		Manuales de instalación	3	3	3	3
		Compatibilidad con arquitecturas variadas	3	3	3	3

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube	OpenShift	Cloud Foundry	Docker	Total
	Cumplimiento de la portabilidad	Nivel de cumplimiento de la portabilidad	3	3	3	3
	General	Satisfacción general	3	3	3	3
	Subtotal Portabilidad		35	35	38	39
Fiabilidad	Madurez	Tiempo en el mercado	3	3	2	3
		Historial	3	3	3	3
		Actualizaciones	2	2	3	3
		Registro de modificaciones	3	3	3	3
		Gestión de problemas e incidentes	3	3	3	3
	Tolerancia a fallos	Manejo de errores y excepciones	3	3	3	3
		Alta disponibilidad	2	2	2	3
		Recuperación ante fallos	3	3	3	3
		Respaldo de información y configuración	3	3	3	3
	Capacidad de recuperación	Recuperación de estado previo a errores	3	3	3	3
		Recuperación de datos	3	3	3	3
		Recuperación de configuración	3	3	3	3
	Cumplimiento de la fiabilidad	Nivel de cumplimiento de la fiabilidad	3	3	3	3
	General	Satisfacción general	3	3	3	3
	Subtotal Fiabilidad		40	40	40	42

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube	OpenShift	Cloud Foundry	Docker	Total
Usabilidad	Inteligibilidad	Usabilidad de interfaz de usuario	3	3	2	3
		Administración de plataforma	3	3	3	3
		Gestión de componentes	2	2	3	3
	Facilidad de aprendizaje	Documentación oficial	3	3	3	3
		Participación comunidad	3	3	3	3
		Cursos presenciales y/o online	3	3	3	3
	Facilidad de operación	Medición de uso	2	2	3	3
		Contabilidad	3	2	2	3
		Cobro y facturación	3	3	3	3
		Múltiples propietarios	3	3	3	3
	Atractivo	Personalización de interfaces de usuario	1	1	1	3
		Facilidad de navegación	2	2	3	3
		Interfaz visualmente agradable	3	3	2	3
	Cumplimiento de la usabilidad	Nivel de cumplimiento de la usabilidad	2	2	2	3
	General	Satisfacción general	2	2	2	3
	Subtotal Usabilidad		38	37	38	45
Eficiencia	Comportamiento en el tiempo	Respuesta de interfaz de usuario	3	3	3	3
		Provisión de plataformas	2	2	3	3
		Provisión de servicios	3	3	3	3
	Utilización de recursos	Recursos usados para ejecución	2	2	3	3

Continúa

Característica	Subcaracterística	Requerimiento de marco de referencia de computación en la nube	OpenShift	Cloud Foundry	Docker	Total
		Gestión de recursos	2	2	3	3
	Cumplimiento de la eficiencia	Nivel de cumplimiento de la eficiencia	2	2	3	3
	General	Satisfacción general	2	2	3	3
	Subtotal Eficiencia		16	16	21	21
Mantenibilidad	Facilidad de análisis	Facilidad para detectar fallos	3	3	3	3
		Facilidad para detectar errores de uso	3	3	3	3
	Facilidad de prueba	Facilidad para ejecutar casos de prueba	3	3	3	3
	Estabilidad	Cantidad de modificaciones	2	2	3	3
		El sistema puede funcionar a pesar de cambios	2	2	3	3
	Facilidad de modificación	Soporte para nuevas plataformas	2	2	3	3
		Extensión de servicio con nuevas acciones	3	3	3	3
		Acceso a código fuente	3	3	3	3
	Cumplimiento de la mantenibilidad	Nivel de cumplimiento de la mantenibilidad	2	2	3	3
	General	Satisfacción general	3	3	3	3
	Subtotal Mantenibilidad		26	26	30	30
	Totales		196	195	211	225

3.6.2.1. RESUMEN DE RESULTADOS DE PLATAFORMA COMO SERVICIO

La Tabla 10 muestra el resumen del puntaje obtenido por cada una de las soluciones de Plataforma como Servicio que han sido analizadas.

Tabla 10. Resumen de puntaje obtenido para Plataforma como Servicio

Característica	OpenShift	Cloud Foundry	Docker	Total
Funcionalidad	41	41	44	48
Portabilidad	35	35	38	39
Fiabilidad	40	40	40	42
Usabilidad	38	37	38	45
Eficiencia	16	16	21	21
Mantenibilidad	26	26	30	30

La Figura 14 permite tener una visualización de los resultados obtenidos de la tabla comparativa de Plataforma como Servicio en cada una de sus características.

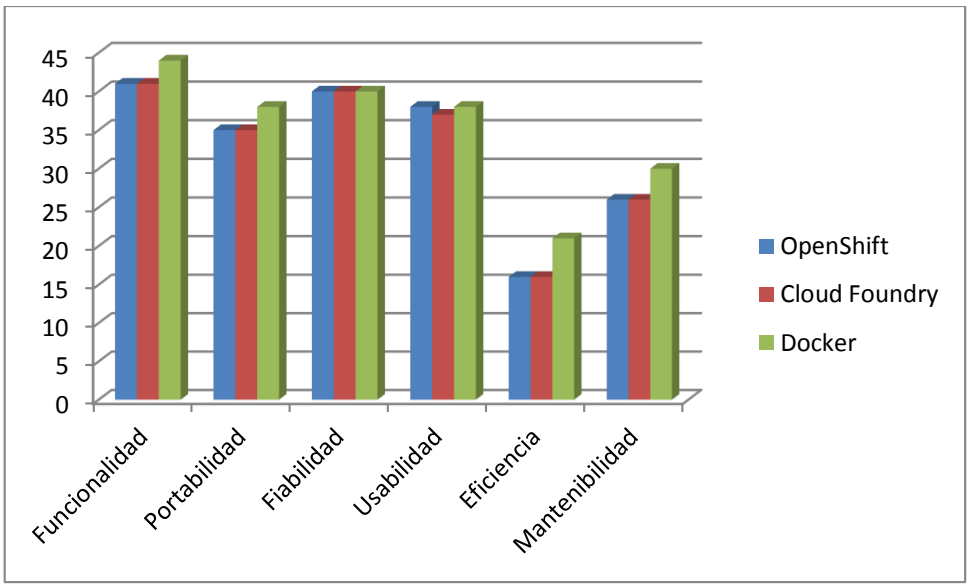

Figura 14. Resultados de Plataforma como Servicio por característica

3.7. SELECCIÓN DE SOLUCIONES

3.7.1. ANÁLISIS DE TABLA COMPARATIVA DE INFRAESTRUCTURA COMO SERVICIO

A continuación se analizan los resultados obtenidos por cada una de las herramientas incluidas en la tabla comparativa de Infraestructura como Servicio.

3.7.1.1. ANÁLISIS DE CLOUDSTACK

Cumple con la mayoría de características buscadas para la solución de Infraestructura como Servicio. Las funcionalidades de gestión de procesamiento se cumplen en general con la particularidad de contar con un proyecto paralelo para la gestión a través de línea de comandos.

En cuanto al almacenamiento cumple con las características generales pero no cuenta con características completas de replicación y auditoría de los dispositivos de almacenamiento.

Las características esperadas de gestión de recursos de red se cumplen.

La interfaz gráfica de usuario permite la gestión simplificada de los recursos y tiene capacidad de extensión y personalización, otorga capacidad de autoservicio y es posible controlar el acceso.

En las características de seguridad si bien cuenta con un control centralizado de identidades, políticas y métodos de autenticación, no llega al punto de gestionar bajo el esquema de control de acceso basado en roles.

En cuanto a la gestión de imágenes cuenta con interesantes características y soporte extendido para distintos hipervisores.

En el campo de orquestación de recursos cumple las funciones esperadas y cuenta con herramientas adicionales que potencian sus características.

Dentro de los aspectos adicionales es importante que su desarrollo sea hecho principalmente en lenguaje Java, ya que existe una gran cantidad de talento humano que tiene ese conocimiento. Los hipervisores soportados son los principales de código abierto e importantes hipervisores de código propietario.

Sus requerimientos de instalación confirman el soporte de hardware común para ser instalado, y el proceso de instalación tiende a ser simplificado. Se busca tener mayor adopción buscando su uso tanto para soluciones de nube privada como de nube pública. Para esta última tiene una capa de integración con la que se considera la principal nube pública, la nube de Amazon.

CloudStack es un proyecto de código abierto que es llevado por la Fundación Apache, conocida ampliamente por su aporte a la industria con proyectos de código abierto que trascienden. Adicionalmente cuenta con el apoyo de importantes corporaciones como Yahoo, Google, Microsoft, Facebook, Citrix e IBM. Al ser un proyecto de código abierto la comunidad de desarrolladores juega un papel importante en el crecimiento y la adopción de la herramienta.

La percepción generada por parte de CloudStack es positiva. Cuenta con el apoyo de la Fundación Apache y la comunidad continúa con su desarrollo. Se tiene un enfoque en simplificar el uso de la solución.

3.7.1.2. ANÁLISIS DE VMWARE

VMware es un referente en cuanto a software de virtualización. Partiendo de esta base la solución de Infraestructura como Servicio de VMware es muy completa. En general cumple con todas las características que se esperan de la solución de Infraestructura como Servicio. Esta solución ha sido un buen punto de referencia para conocer las características que pueden ser evaluadas como parte de la herramienta.

Sin embargo uno de los objetivos del proyecto es hacer uso de tecnologías de código abierto lo que restringe la selección de VMware como la herramienta de IaaS para el presente esfuerzo.

Es importante notar que la solución de VMware también es extendida para hacer uso no solo del hipervisor de este fabricante, y VMware filial de la corporación EMC cuenta con una estructura que se encarga de hacer que su tecnología siga avanzando e incluso se involucre con los proyectos de código abierto más llamativos e importantes para seguir acentuando su presencia en el mercado.

3.7.1.3. ANÁLISIS DE OPENSTACK

OpenStack en el campo de procesamiento cumple con las características requeridas. A destacar que en el campo de arquitectura distribuida se puede mencionar su división en varios proyectos que se especializan en cada uno de los campos requeridos.

En cuanto al almacenamiento cuenta con todas las características deseadas. Destaca principalmente en los campos de replicación y auditoría de dispositivos, adicionalmente es la única alternativa abierta que permite controlar el ciclo de vida de la información y la expiración de la información de forma automatizada. Su proyecto de almacenamiento basado en objeto es referente inclusive para las otras alternativas analizadas en este documento. En cuanto a los dispositivos especializados de almacenamiento incluye a todos los más populares entre los que se pueden nombrar Ceph, NetApp, Nexenta, SolidFire y Zadara.

En el campo de red, OpenStack ofrece características importantes como el uso de direcciones IP flotantes y la creación redes definidas por software. La extensión de sus características de red permite el acceso a dispositivos y tecnologías como Sistemas de detección de intrusos, balanceo de carga, cortafuegos y redes privadas virtuales. OpenStack cuenta con dos proyectos que ofrecen dos acercamientos hacia la gestión de red permitiendo que se adapte a distintos ambientes y necesidades.

En cuanto a la interfaz gráfica, se encuentra que es uno de los puntos donde más avances ha mostrado y llega nuevamente a cumplir las características requeridas. Sin embargo, la impresión general es que dentro de las opciones manejadas en este análisis no se trata de la interfaz más intuitiva, de todas formas su enfoque en la personalización la hacen una alternativa interesante.

En el campo de seguridad OpenStack impone la tendencia. Con un proyecto dedicado específicamente a este propósito incluye las funcionalidades de gestión centralizada de identidades de usuario y un completo control de acceso basado en roles. La autenticación permite también

la integración con directorios LDAP lo cual facilita la integración con el entorno con el que cuentan las empresas.

La gestión de imágenes es otro de los proyectos de OpenStack. Cuenta con las características requeridas, destacando que el respaldo de imágenes se realiza para todos los formatos de imagen soportados. Ofrece en general soporte a los formatos de imagen esperados, tanto para hipervisores abiertos como propietarios.

En el campo de orquestación, cuenta con las características esperadas. Adicionalmente existen herramientas creadas por sus auspiciantes que buscan apoyar estas características y la complementan con funcionalidades adicionales.

Los aspectos adicionales ofrecen una visión interesante del enfoque de esta alternativa y de la importancia que va ganando en el mercado. Su lenguaje de programación es principalmente Python, lo cual ofrece facilidad para los desarrolladores para integrarse y crear características nuevas y a medida. Su soporte de hipervisores incluye los más importantes y deseados para este proyecto. En cuanto a hardware es posible instalar OpenStack en equipos no especializados. Ofrece capacidad de alta disponibilidad para varios de sus proyectos tratando de eliminar la mayor cantidad de puntos de falla.

OpenStack se presenta como un sistema operativo para la nube. Paradójicamente su fortaleza, el contar con varios proyectos especializados que le dan muchas características específicas, es también uno de sus puntos más discutidos ya que eleva la complejidad de su instalación y uso. La instalación se considera de complejidad alta, se requiere entender cada uno de los componentes y cómo interactúa para entregar las características

descritas, existen esfuerzos de la comunidad buscan entregar procesos de instalación simplificados para apoyar la comprensión y acceso a más usuarios.

OpenStack es dirigido por la Fundación OpenStack, este organismo cuenta con importantes empresas y corporaciones que lo patrocinan y apoyan económicamente y técnicamente, y ofrecen sus propias soluciones basadas en OpenStack pero con componentes propietarios para los cuales entregan soporte y por los cuales se debe pagar un costo.

OpenStack es el proyecto que mayor revuelo expectativa está causando por sus objetivos altos y su alto nivel de personalización. El esfuerzo de desarrollo y los auspiciantes de este “sistema operativo para la nube” hacen que sea uno de los proyectos más destacados. Así mismo es un proyecto que al momento de la redacción de este documento no se considera por sí solo listo para entrar en ambientes de producción.

3.7.1.4. ANÁLISIS DE EUCALYPTUS

Se trata de uno de los proyectos referentes en el campo de IaaS de código abierto. Es un proyecto que ha tenido altibajos pero que hoy por hoy cuenta con un gran apoyo por parte de la comunidad y de sus patrocinadores. Concebido como un facilitador de nubes híbridas, Eucalyptus cuenta con APIs enfocadas en la integración con la nube pública de Amazon.

En los campos analizados Eucalyptus cuenta con la mayoría de las características buscadas. Uno de los puntos a destacar es la inclusión del componente euca2ools que contiene una serie de herramientas que buscan facilitar las tareas de administración e integración de Eucalyptus con la infraestructura de los clientes.

En el campo de procesamiento cumple con todas las características buscadas, resaltando que hace uso de euca2ools para lo referente a la interfaz de línea de comandos administrativos.

En lo referente a almacenamiento soporta los diferentes tipos solicitados pero en características avanzadas como la replicación y la auditoría de dispositivos físicos requiere de herramientas que complementen sus funcionalidades.

En cuanto a la gestión de red cumple con los puntos solicitados, sin embargo en lo relacionado a redes definidas por software apenas alcanza a cumplir, existe un compromiso de la comunidad de desarrolladores para aumentar la integración de Eucalyptus con este tipo de funcionalidad.

La interfaz gráfica es una de las más ricas y usables de las opciones analizadas, sus facilidades de administración incluyen la interacción con la nube pública, materializando de esta forma la capacidad de administración de nube híbrida por medio de una sola interfaz.

En cuanto a la seguridad, gestión de imágenes y orquestación de recursos cumple con las características buscadas.

En los aspectos adicionales se puede resaltar que su comunidad es variada y existen muchos desarrollos hechos en lenguajes de programación diversos, siendo los principales, Java, C, Python y Perl. Dentro de los hipervisores soportados se echa de menos a Hyper-V, aunque existe la intención de integrarlo en un futuro. Sus requerimientos de instalación son mínimos, y bajos en comparación a las otras herramientas. Su proceso de instalación es de complejidad media con enfoque en la simplificación. En lo

referente a alta disponibilidad, la parte más madura la componen sus elementos desarrollados en Java.

Sus contribuidores son la empresa Eucalyptus, fundada para dar soporte a la herramienta; Amazon, Dell, HP, Intel, entre otros. Lo que deja entrever que Eucalyptus es una opción fuerte que cuenta con mucho apoyo y que tiene en su compatibilidad con la nube pública uno de sus puntos más altos y fuertes.

3.7.1.5. ANÁLISIS DE OPENNEBULA

OpenNebula se enfoca en la simplicidad y en algunos casos esa simplicidad la lleva a no buscar algunas de las características que se están evaluando en el presente proyecto. En general es una opción robusta que permite acceder a un ambiente de nube privada de forma sencilla.

En cuanto a procesamiento cumple con todas las características básicas.

En cuanto a almacenamiento se apoya en proyectos externos para cumplir con algunas de las características, por ejemplo, para lograr el almacenamiento basado en objetos hace uso de Ceph. Sin embargo cabe destacar que no cumple con el punto de control de fallos en los dispositivos y la replicación depende mucho del sistema de archivos que se esté utilizando. Así mismo no cumple con la auditoría de los dispositivos, el respaldo de volúmenes de disco y el soporte de dispositivos de almacenamiento especializados.

En el campo de redes, cumple con los puntos genéricos, sin embargo no cumple con el uso de direcciones IP flotantes y requiere complementos externos para cumplir con las redes definidas por software.

La interfaz gráfica busca ser sencilla, usable e intuitiva. Así mismo permite personalizaciones que faciliten aún más su adopción.

En cuanto a seguridad cumple con la gestión de usuarios sin embargo no cuenta con la gestión de acceso basada en roles.

La gestión de imágenes y la orquestación de recursos es otro de los puntos que busca ser muy sencillo y usable y termina cumpliendo con las características buscadas.

En otros aspectos se puede encontrar que su desarrollo incluye una gran cantidad de lenguajes de programación dentro de los que se puede contar C++, C, Ruby, Java, Shell script, yacc y lex. Dentro de los hipervisores soportados se deja de lado a Hyper-V. No registra requerimientos genéricos para instalación sino que indica que siempre se debe contar con un dimensionamiento que facilite este análisis.

El proceso de instalación es de complejidad baja, siendo este uno de los objetivos de este proyecto, cuenta con compatibilidad para la nube pública. La simplicidad también se ve plasmada en el objetivo de entregar la posibilidad a las empresas de crear una nube privada a partir de la infraestructura existente. Sus componentes principales pueden ser manejados con alta disponibilidad.

Dentro de sus contribuidores se cuenta a la comunidad de OpenNebula, C12G Labs, IBM, Blackberry y CentOS.

La propuesta de enfoque en la simplicidad es uno de los puntos altos de OpenNebula.

3.7.2. ANÁLISIS DE TABLA COMPARATIVA DE PLATAFORMA COMO SERVICIO

A continuación se analizan los resultados obtenidos por cada una de las herramientas incluidas en la tabla comparativa de Plataforma como Servicio.

3.7.2.1. ANÁLISIS DE OPENSHIFT

La opción de Plataforma como Servicio de código abierto de Red Hat, se vuelve un candidato muy fuerte para empresas que tiene su infraestructura basada en Red Hat. Aunque la compañía propone que su tecnología permite minimizar que un cliente quede sujeto a un proveedor (Openshift, 2014), así mismo se mira que OpenShift es una solución que puede resultar más aparente para empresas que utilicen el sistema operativo de Red Hat.

Al margen de esto, OpenShift es una solución que ofrece todas las características esperadas para Plataforma como Servicio, entregando finalmente muchas opciones de integración con diferentes proyectos y productos. OpenShift puede ser integrado con OpenStack, lo cual es un punto importante dentro del análisis de este proyecto que tiene en OpenStack a su herramienta seleccionada para IaaS.

OpenShift ofrece también la posibilidad de hacer uso del servicio en la nube pública entregando facilidades para la creación de nubes híbridas.

OpenShift Origin es el proyecto de código abierto de Red Hat, que es base de sus ofertas de Plataforma como Servicio.

3.7.2.2. ANÁLISIS DE CLOUD FOUNDRY

Cloud Foundry cuenta con el apoyo de la fundación que ha captado a la mayoría de fabricantes y corporaciones enfocadas en la nube.

Es un punto interesante ya que la mayoría de empresas que están relacionadas con OpenStack están apoyando a Cloud Foundry entregándole un aire de solución preferida en la industria.

Las características que Cloud Foundry ofrece permiten interactuar con nubes privadas y públicas y automatizar los procedimientos operativos para poner aplicaciones en producción. Se busca hacer de Cloud Foundry el estándar en cuanto a soluciones de PaaS.

3.7.2.3. ANÁLISIS DE DOCKER

El proyecto que ha recibido la mayor cantidad de atención desde su aparición. Docker hace sencillo el uso de contenedores en Linux, creando facilidades para crear ambientes aislados que corren sobre el mismo kernel de Linux. Con estas características se hace factible la creación de ambientes a medida fácilmente reproducibles, volviéndose una opción de Plataforma como Servicio.

Existen muchos esfuerzos enfocados en la utilización de Docker dentro de los proyectos de código abierto que lideran las tendencias de computación en la nube. Docker es usado por OpenShift como parte de sus características y también puede ser utilizado en Cloud Foundry gracias a un proyecto

conocido como Decker. La flexibilidad y el momento con los que cuenta Docker en la actualidad lo hacen una opción fuerte para enfocar la investigación.

3.7.3. SOLUCIONES SELECCIONADAS

A continuación se describen las características de las soluciones seleccionadas para Infraestructura como Servicio y Plataforma como Servicio.

3.7.3.1. SOLUCIÓN SELECCIONADA PARA INFRAESTRUCTURA COMO SERVICIO

Luego de realizar la comparación de las soluciones preseleccionadas se Infraestructura como Servicio se han obtenido los resultados totales que se muestran en la Figura 15.

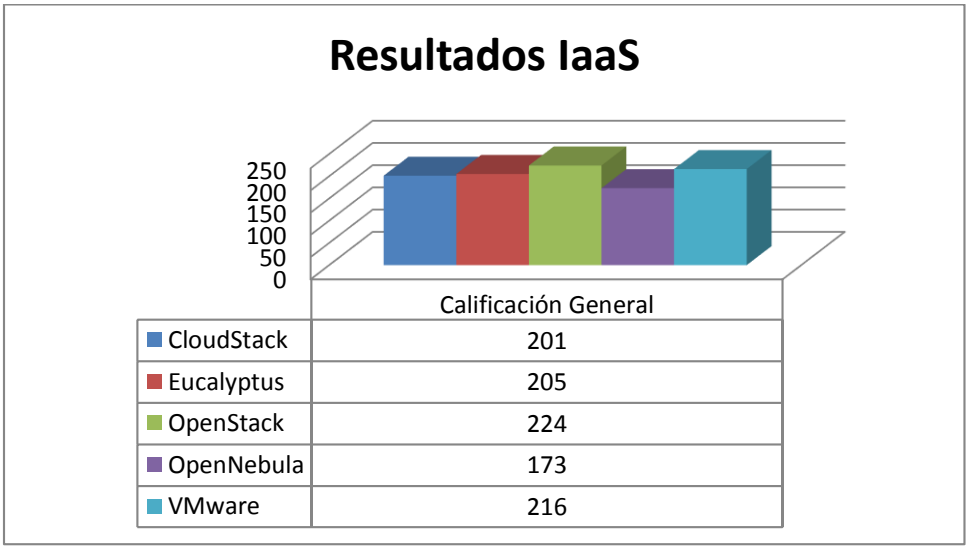

Figura 15. Resultados totales de Infraestructura como Servicio

Resultado: La solución que obtiene el resultado más alto y por lo tanto es la seleccionada para Infraestructura como Servicio es OpenStack.

La solución cumple con todas las características requeridas, tiene gran cantidad de información disponible y su uso se extiende de forma consistente. Los auspiciantes de esta solución son empresas a las que el auspiciante de este proyecto está asociado, como IBM y Red Hat.

La elección de OpenStack implica un riesgo que se debe manejar y que tiene que ver con la complejidad inmersa en la solución, sin embargo es parte de los objetivos de este proyecto el sortear la curva de aprendizaje necesaria para hacer uso de los componentes e interactuar con los distintos proyectos de OpenStack.

3.7.3.2. SOLUCIÓN SELECCIONADA PARA PLATAFORMA COMO SERVICIO

Luego de realizar la comparación de las soluciones preseleccionadas se Infraestructura como Servicio se han obtenido los resultados totales que se muestran en la Figura 16.

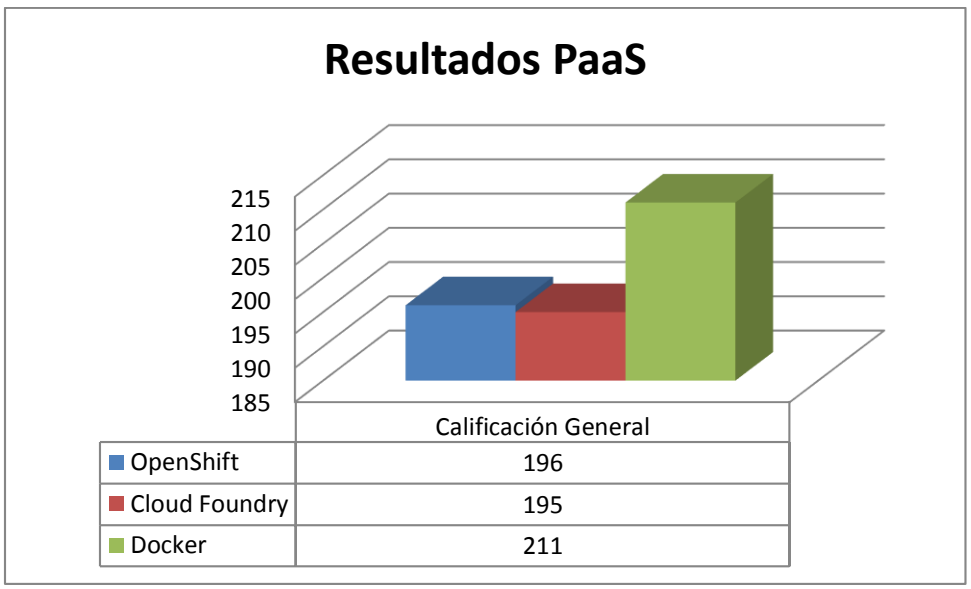

Figura 16. Resultados totales de Plataforma como Servicio

Resultado: La solución que obtiene el mejor resultado en la comparación y, por lo tanto, es la seleccionada para Plataforma como Servicio es Docker.

Docker es uno de los proyectos más populares en la comunidad de código abierto junto con OpenStack (Williams, 2014). Si bien es una solución difícil de encasillar sus funcionalidades resultan aplicables para los requerimientos buscados en el siguiente proyecto.

Aunque la principal competencia en la industria en lo que se refiere a soluciones de PaaS privada se da entre OpenShift y Cloud Foundry como tendencias principales, Docker es la solución que todos toman como denominador común apoyándose en sus capacidades para ampliar su funcionalidad nativa. De todas formas no se puede descartar el uso de componentes asociados principalmente a OpenShift y a Cloud Foundry para complementar las capacidades buscadas en este proyecto.

Es necesario validar las opciones de integración entre Docker y OpenStack siendo estas las soluciones elegidas. Hoy por hoy se cuenta con varias opciones de integración a distintos niveles, las cuales serán analizadas con miras a encontrar la integración que solvente las necesidades planteadas en este proyecto.

Los resultados obtenidos en este proceso de comparación coinciden con los resultados de la encuesta de Linux.com y The New Stack, que muestran como los proyectos de código abierto para computación en la nube más populares a OpenStack y a Docker (Williams, 2014). En la Figura 17 se aprecian los resultados de la encuesta.

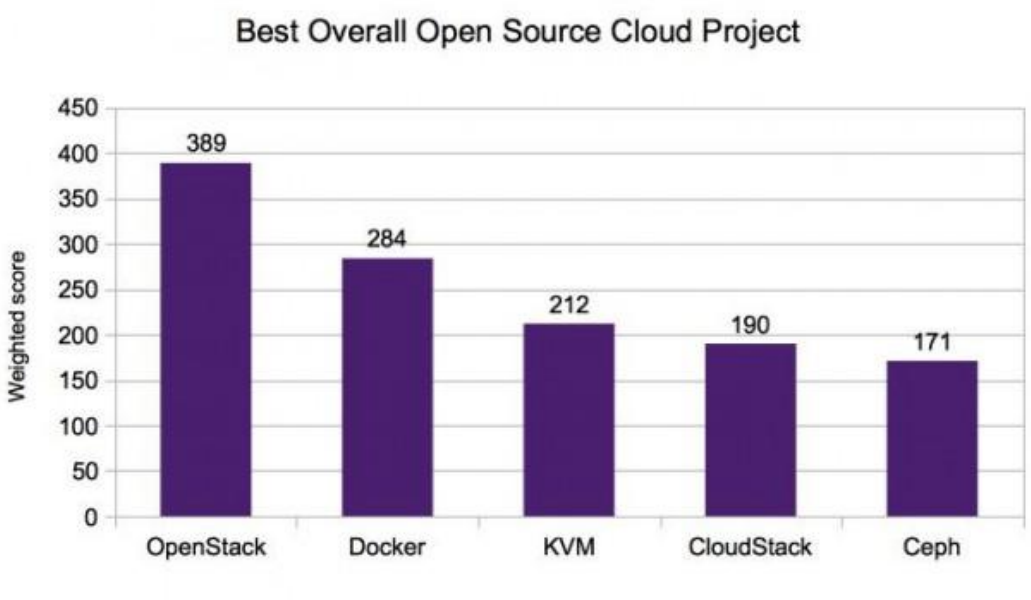

Figura 17. Proyectos de código abierto para computación en la nube más populares según Linux.com y The New Stack.
Fuente:(Williams, 2014)

3.8. SOLUCIÓN DE INFRAESTRUCTURA COMO SERVICIO

La solución de Infraestructura como Servicio seleccionada es OpenStack. A continuación se presenta una descripción de sus características y componentes principales.

3.8.1. VISIÓN GENERAL DE LA SOLUCIÓN DE INFRAESTRUCTURA COMO SERVICIO

OpenStack es un sistema operativo de nube que controla grupos de recursos de cómputo, almacenamiento y red de gran tamaño en un centro de datos, todo esto a través de un panel de instrumentos que entrega control de a los administradores mientras permite que los usuarios finales obtengan recursos a través de su interfaz web.

OpenStack es una colección de proyectos tecnológicos de código abierto patrocinados por un amplio grupo de líderes de la industria.

Entrega una plataforma operativa para orquestar nubes a escala masiva. Su tecnología es independiente del hipervisor e incluye software para gestionar máquinas virtuales en hardware común. Adicionalmente ofrece un almacén de objetos distribuido y una amplia gama de funcionalidades opcionales como un controlador de red, gestor de autenticación, un panel de gestión y almacenamiento en bloque.

Iniciado en el 2010 por Rackspace Cloud y Administración Nacional Espacial y Aeronáutica de los Estados Unidos (National Aeronautics and Space Administration, NASA) actualmente está bajo control de la Fundación OpenStack que está compuesta por más de 100 miembros, entre los que se encuentran varias de las organizaciones más grande de la industria entre la cuales se puede mencionar a IBM, AT&T, Canonical, HP, Nebula, Rackspace, Red Hat, y SUSE.

3.8.2. ARQUITECTURA LÓGICA DE LA SOLUCIÓN DE INFRAESTRUCTURA COMO SERVICIO

OpenStack está compuesto por una serie de proyectos que cumplen funciones específicas. La interrelación de estos proyectos permite entregar las funcionalidades de gestión de nube requeridas. La Figura 18 muestra la interacción de los proyectos de OpenStack.

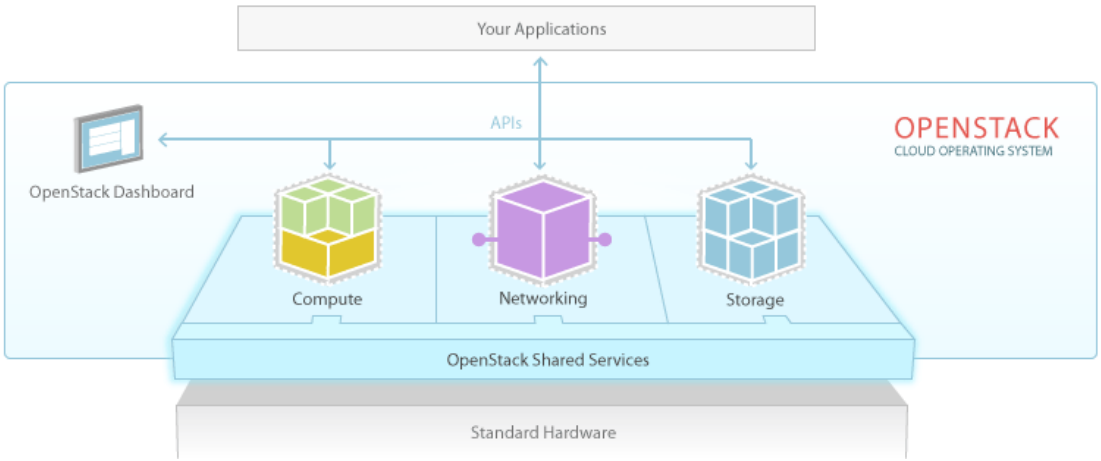

Figura 18. Arquitectura lógica de OpenStack
Fuente: (OpenStack, 2014)

La interacción de los proyectos de OpenStack permite gestionar los recursos de hardware como una Infraestructura de computación en la nube que puede ser consumida como servicio bajo demanda.

La capa física está compuesta por hardware común y estándar. Esto quiere decir que se puede administrar tanto hardware no especializado como hardware estándar especializado. Esto permite que OpenStack sea una solución que abarca todo tipo y tamaño de empresa, altamente escalable y con una gran capacidad de adaptación.

A continuación se cuenta con los proyectos básicos que permiten la gestión de recursos de cómputo, almacenamiento y red para proveer entornos virtualizados bajo demanda.

Estos proyectos básicos están también relacionados con un grupo de servicios compartidos que entregan funcionalidad complementaria que entrega mayor versatilidad y control en la nube. Entre estos servicios están: control de identidades y autenticación, gestión de imágenes, servicios de medición de consumo, servicios de orquestación y servicios de base de datos.

Esta infraestructura cuenta con una serie APIs que permiten que cada uno de los componentes sea administrado y controlado a través de un panel de instrumentos que, a su vez, es otro de los proyectos que forma parte de OpenStack. A través de este panel de instrumentos es posible gestionar todos los aspectos de la infraestructura, así como automatizar tareas y facilitar el acceso a los usuarios finales.

Finalmente, haciendo uso de las APIs ya mencionadas, se cuenta con aplicaciones que permiten interactuar con la infraestructura y también permiten el uso de los recursos para soportar las cargas de trabajo de las organizaciones.

3.8.3. SERVICIOS Y PROYECTOS

A continuación se presenta una descripción de cada uno de los proyectos que forman parte de OpenStack.

3.8.3.1. CÓMPUTO

El proyecto de Cómputo de OpenStack es conocido como OpenStack Compute o con el nombre Nova.

Nova permite el acceso bajo demanda a recursos de cómputo entregando y gestionando grandes redes de máquinas virtuales. Estos recursos de cómputo son accesibles vía APIs de forma que los desarrolladores pueden crear aplicaciones para la nube; y vía interfaces web para los administradores y usuarios finales.

La arquitectura de cómputo está diseñada para escalar horizontalmente permitiendo las ventajas económicas que las organizaciones esperan conseguir a través de este tipo de solución.

OpenStack está creado desde su base para proveer flexibilidad al momento de diseñar nubes, no requiere hardware o software propietario y puede integrar sistemas de legado y tecnologías de terceros.

OpenStack está diseñado para gestionar y automatizar grupos de recursos de cómputo y puede trabajar con variadas tecnologías de virtualización, así como en configuraciones de servidores físicos dedicados de alto rendimiento.

Los administradores a menudo usan uno de los variados hipervisores soportados para desplegar OpenStack Compute, por ejemplo: KVM y XenServer, así como tecnologías de contenedores para Linux como LXC o Docker que están plenamente soportados para escenarios en los que se quiera minimizar el consumo de recursos virtualizados y obtener mayor eficiencia y rendimiento.

Nova soporta varias arquitecturas de hardware como x86, ARM y otras alternativas.

Entre los casos de uso populares para Nova se puede mencionar:

- **Proveedores de Servicio:** Que ofrecen servicios de Infraestructura como Servicio.
- **Departamentos de TI:** que actúan como proveedores de servicio en la nube para otras unidades de negocio y equipos de proyectos.

- **Procesamiento de Big Data:** para soportar la carga de soluciones de minería y analítica de grandes cantidades de información no relacionada como Hadoop.
- **Cómputo en escala:** que permite aumentar y disminuir la capacidad para cubrir la demanda de recursos web y aplicaciones.
- **Computación de alto rendimiento:** ambientes que requieren procesamiento diverso para cargas de trabajo intensivas.

Un detalle de las características y beneficios entregados por OpenStack Compute (Nova) se incluyen en el Anexo 1.

3.8.3.2. ALMACENAMIENTO

La gestión de Almacenamiento de OpenStack ofrece distintas opciones que están contenidos dentro del concepto de OpenStack Storage. En general en esta área se ofrece soporte para almacenamiento a nivel de bloque y de objeto para ser usado por servidores y aplicaciones.

Existen dos proyectos principales de acuerdo al tipo de almacenamiento que manejan:

- Almacenamiento a nivel de bloque: Proyecto Cinder
- Almacenamiento a nivel de objeto: Proyecto Swift

Aparte de las tecnologías de almacenamiento tradicionales, las organizaciones tienen una variedad de necesidades de almacenamiento con diversos requerimientos de precio y rendimiento.

OpenStack soporta almacenamiento a nivel de bloque y a nivel de objeto, con muchas opciones de despliegue e integración dependiendo del caso de uso.

Un detalle de las características y beneficios de OpenStack Storage se puede visualizar en el Anexo 2.

3.8.3.3. RED

La gestión de Red de OpenStack ofrece distintas opciones que están contenidos dentro del concepto de OpenStack Networking, proyecto que también es conocido con el nombre clave Neutron. En general en este proyecto se encuentran todos los componentes relacionados con la gestión de red y direcciones IP.

OpenStack Networking puede ser usado por administradores y usuarios para incrementar el valor de los activos existentes, permitiendo soportar las nuevas necesidades de gestión de red. Asegura que la red no será un cuello de botella ni el factor limitante dentro de un despliegue de nube, brindando, inclusive, autoservicio para las configuraciones de red para usuarios finales.

Un detalle de las características y beneficios de OpenStack Networking se puede visualizar en el Anexo 3.

3.8.3.4. PANEL DE INSTRUMENTOS

La interfaz gráfica de administración está a cargo del proyecto OpenStack Dashboard, que también es conocido con el nombre clave Horizon. En general en este proyecto se encuentran todos los componentes

relacionados con la administración gráfica de nube. La Figura 19 muestra una de las vistas de esta interfaz.

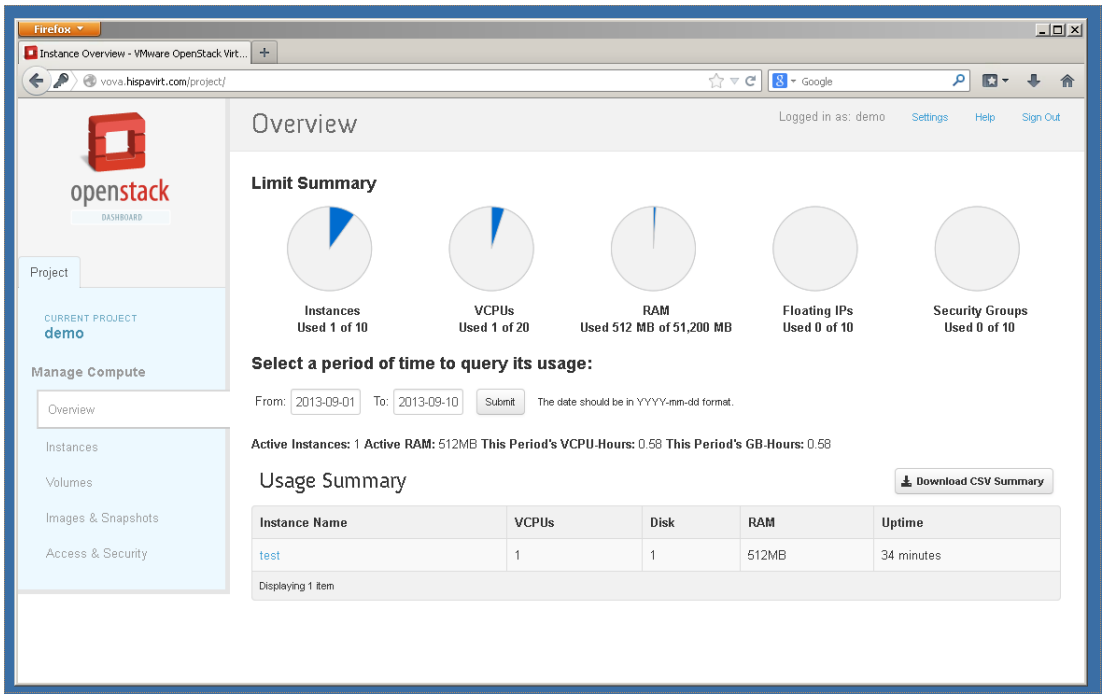

Figura 19. Captura del panel de instrumentos OpenStack Dashboard (Horizon).

OpenStack Dashboard entrega a los administradores y usuarios una interfaz gráfica para acceder, aprovisionar y automatizar el uso de recursos basados en nube. Su diseño extensible permite acoplar y exponer productos y servicios de terceros como facturación, monitoreo y herramientas adicionales de gestión. El panel de instrumentos también puede ser personalizado para adecuarse a la imagen de un proveedor de servicios que quiera hacer uso del mismo.

El panel de instrumentos es solo una forma de interactuar con los recursos de OpenStack. Entre las opciones se encuentran el uso de las APIs de OpenStack o de Amazon EC2.

Dentro de las características de Horizon se encuentran:

- El panel de instrumentos es una aplicación web extensible que permite a los administradores de nube controlar sus recursos de cómputo, almacenamiento y red.
- El panel de instrumentos provee una vista general del tamaño y estado de la nube. Se pueden crear usuarios y proyectos, asignar usuarios a proyectos y configurar los límites de recursos para esos proyectos.
- El panel de instrumentos entrega a los usuarios finales la capacidad de autoservicio de aprovisionamiento para gestionar sus propios recursos dentro de los límites configurados por los administradores.

3.8.3.5. SERVICIOS COMPARTIDOS

OpenStack tiene varios servicios compartidos que amplían las capacidades de los tres pilares principales de cómputo, almacenamiento y red, facilitando la implementación y operación de la nube.

Estos servicios integran los componentes de OpenStack y sistemas externos para entregar una experiencia unificada para los usuarios mientras interactúan con diferentes recursos de nube.

La Tabla 11 muestra los servicios compartidos y el nombre del proyecto que los implementa.

Tabla 11. Servicios compartidos de OpenStack

Proyecto	Servicio entregado
KeyStone	Servicio de identidad
Glance	Servicio de imágenes
Ceilometer	Servicio de telemetría
Heat	Servicio de Orquestación
Trove	Servicio de base de datos

Las funcionalidades de los servicios compartidos de OpenStack se describen en el Anexo 4.

3.9. SOLUCIÓN DE PLATAFORMA COMO SERVICIO

La solución de Plataforma como Servicio seleccionada es Docker. A continuación se presenta una descripción de sus características y componentes principales.

3.9.1. VISIÓN GENERAL DE SOLUCIÓN DE PLATAFORMA COMO SERVICIO

Docker es un motor de código abierto que permite automatizar el despliegue de una aplicación como un contenedor autosuficiente, ligero y portable que puede ser ejecutado sobre, virtualmente, cualquier plataforma.

Los contenedores pueden encapsular cualquier carga de trabajo, y pueden ser ejecutados de forma consistente en y entre cualquier tipo de servidor. El mismo contenedor que un desarrollador construye y prueba en un computador personal, puede ser ejecutado y escalar a entornos de producción, en máquinas virtuales, servidores físicos, clústeres de OpenStack, instancias de nube pública, o cualquier combinación de infraestructura.

Dentro de las aplicaciones de este tipo de tecnología se puede citar:

- Creación de ambientes privados ligeros de Plataforma como Servicio (PaaS).
- Automatización del empaquetado y despliegue de aplicaciones.
- Creación de Ambientes de pruebas e integración y despliegue continuos.
- Despliegue y escalamiento de aplicaciones, bases de datos y servicios en segundo plano.

3.9.2. CARACTERÍSTICAS PRINCIPALES

Docker es una plataforma para desarrollar, portar y ejecutar aplicaciones.

El propósito es desplegar aplicaciones más rápidamente separando las aplicaciones de la infraestructura y tratar a la infraestructura como si se tratase de una aplicación gestionada; apoyando así al ciclo de creación de código y puesta en producción de aplicaciones.

Para esto se combina una plataforma de virtualización de contenedores ligeros con flujos de trabajo y herramientas que ayudan a gestionar y desplegar aplicaciones.

Docker entrega la forma de ejecutar una aplicación aislada de forma segura en un contenedor. El aislamiento y la seguridad permiten ejecutar varios contenedores en un mismo host de forma simultánea. La naturaleza ligera de los contenedores implica que se puede explotar de mejor manera el hardware, al no tener la carga adicional de un hipervisor.

3.9.3. FUNCIONES BÁSICAS

La Tabla 12 resume las funciones básicas del proyecto Docker.

Tabla 12. Funciones básicas de Docker

Función	Descripción
Despliegue acelerado de aplicaciones	Apoya al ciclo de vida de desarrollo de aplicaciones permitiendo a los desarrolladores crear contenedores locales con todo los componentes requeridos e integrarlos en un flujo de despliegue. El mismo contenedor puede ser publicado en ambientes de pruebas y producción asegurando consistencia y despliegue.
Facilidad para desplegar y escalar	Los contenedores pueden correr localmente en el equipo del desarrollador, o en máquinas físicas o virtuales en un centro de datos o en la nube, permitiendo escalar fácilmente.
Mayor densidad y ejecución de mayores cargas de trabajo	Es útil para ambientes de alta densidad, como ambientes de nube y Plataforma como Servicio. Los administradores de ambientes pequeños y medianos pueden sacar mayor provecho de los recursos con los que cuentan.

3.9.4. COMPONENTES Y ARQUITECTURA

Docker usa una arquitectura cliente – servidor. El cliente habla con el proceso demonio Docker, que tiene la tarea de construir, ejecutar y distribuir contenedores. Tanto el cliente como el proceso Docker pueden correr en el mismo sistema, o se puede conectar un cliente a un proceso de control remoto. Los componentes se comunican vía sockets o a través de APIs.

- Proceso Docker

Un proceso que corre en segundo plano en una máquina anfitriona. El usuario final interactúa con este proceso a través de un cliente.

- Cliente Docker

Interfaz de usuario principal. Acepta comandos de usuario e intercambia instrucciones con el proceso Docker.

- Componentes internos

Internamente Docker usa tres componentes:

  - **Imágenes:** una imagen es una plantilla de solo lectura, que cuenta con aplicaciones y componentes instalados. Las imágenes son usadas para crear contenedores.

  - **Registros:** los registros gestionan imágenes formando repositorios públicos o privados para subir o descargar las mismas. El registro público de Docker se conoce como Docker Hub.

  - **Contenedores:** los contenedores son similares a un directorio. Un contenedor mantiene todos los componentes necesarios para que una aplicación se ejecute. Cada contenedor es creado a partir de una imagen. Los contenedores pueden ejecutarse, iniciar, detenerse, moverse y eliminarse. Cada contenedor es una plataforma de aplicación aislada y segura.

    La Figura 20 muestra la interacción de componentes de Docker.

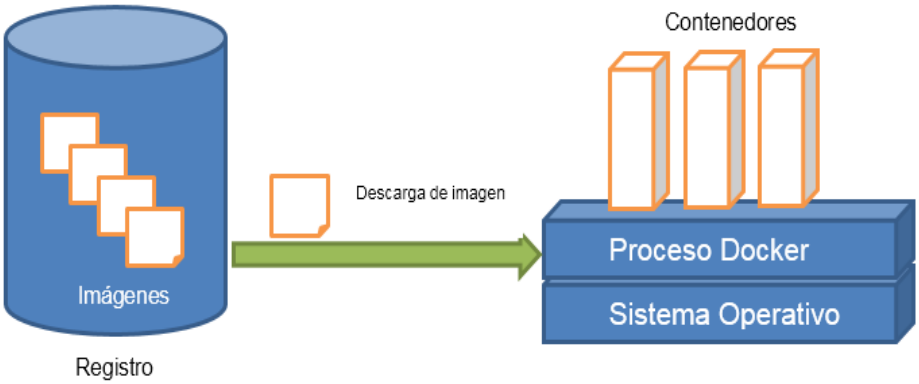

Figura 20. Interacción de componentes de Docker

La interacción de los componentes mencionados en esta sección se detalla en el Anexo 5.

3.9.5. TECNOLOGÍA ASOCIADA

Docker está desarrollado con el lenguaje de programación Go y hace uso de varias características del kernel de Linux para entregar sus funcionalidades.

La Tabla 13 resume las tecnologías asociadas con Docker.

Tabla 13. Tecnologías asociadas con Docker

Tecnología	Descripción
Espacios de nombre	Permite entregar un espacio de trabajo aislado llamado contenedor, cuando este se ejecuta se crea un grupo de espacios de nombre (namespaces) para el mismo. Cada aspecto de un contenedor se ejecuta en su propio espacio de nombre aislado y no tiene acceso por fuera de sí mismo.

Continúa

Tecnología	Descripción
Grupos de control	Permiten que las aplicaciones se ejecuten aisladamente usando los recursos que requieren. Esto asegura que los contenedores tengan múltiples propietarios. Los grupos de control (cgroups) permiten compartir recursos de hardware disponibles entre contenedores y, de ser requerido, configurar límites y restricciones.
Sistema de archivos de unión	Los sistemas de archivos de unión (UnionFS) operan creando capas, haciendo que los sistemas de archivos sean ligeros y rápidos. Estos son los bloques de construcción para contenedores.
Formato de contenedor	Los componentes se combinan en un envoltorio llamado formato de contenedor. El formato de contenedor predeterminado se llama libcontainer. Docker también soporta contenedores Linux tradicionales usando LXC y, a futuro, BSD Jails y Solaris Zones.

3.10. INTEGRACIÓN DE INFRAESTRUCTURA COMO SERVICIO Y PLATAFORMA COMO SERVICIO

Tanto OpenStack como Docker son tecnologías que pueden funcionar de forma independiente entregando sus prestaciones y características. Sin embargo al combinar estas tecnologías es posible obtener funcionalidades de computación en la nube muy versátiles.

3.10.1. VISIÓN GENERAL DE LA INTEGRACIÓN

La integración de OpenStack con variados proyectos de código abierto avanza a grandes pasos dado el interés y la atención con la que cuenta el proyecto, gracias a los resultados que sigue obteniendo tanto para ambientes de prueba como para entornos empresariales.

Dentro de estas integraciones existen esfuerzos que se están desarrollando para unir las capacidades de OpenStack con Docker, siendo ambos proyectos muy populares que experimentan un momento de crecimiento y adopción importante.

Concretamente para OpenStack, Docker, al ser una forma de manejar múltiples contenedores en una sola máquina, se convierte en un recurso muy poderoso al hacer posible que se gestionen varios anfitriones, lo que se traduce en la gestión de cientos de contenedores simultáneamente. Docker apunta a ampliar la compatibilidad con OpenStack.

Es necesario aclarar que los contenedores no apuntan a ser un reemplazo de las máquinas virtuales, sino más bien se busca que sean una tecnología complementaria que puede ser aplicada para casos de uso específicos.

Dentro de las capacidades específicas que aporta Docker junto a OpenStack se puede mencionar las siguientes:

- Docker toma ventaja de tecnologías de contenedor y de sistema de archivos que deberían ser gestionados por OpenStack usando máquinas virtuales gestionadas por un hipervisor.

- API a nivel de proceso, que recolecta entradas y salidas del proceso en ejecución en cada contenedor, para registro o interacción directa, permitiendo bloquear un contenedor hasta su salida, configurar su ambiente, y otras primitivas orientadas a procesos que no pueden ser manejadas por un hipervisor.

- Control de cambios avanzado a nivel de sistema de archivos que permite gestionar cada cambio hecho en el sistema de archivos a través de un grupo de capas que pueden ser capturadas, retrotraídas y gestionadas.

- Portabilidad de imágenes que permite aplicar el estado de cualquier contenedor Docker y compartirlo a través del registro central de imágenes. Las imágenes Docker son diseñadas para ser portables entre infraestructuras.

- Docker puede automatizar el ensamblaje de un contenedor desde el código fuente de una aplicación, dando a los desarrolladores una forma sencilla de desplegar sus aplicaciones a un clúster de OpenStack como parte de su flujo de desarrollo.

3.10.2. OPCIONES DE INTEGRACIÓN

Las principales opciones de integración de OpenStack y Docker son:

- Integración a través de Nova
- Integración a través de Heat

Ambos acercamientos son fundamentalmente diferentes, uno no reemplaza a otro; y cada uno sigue contando con apoyo y desarrollo por parte de la comunidad. Cada acercamiento tiene sus ventajas y desventajas. A continuación se describe cada una de las opciones.

3.10.2.1. INTEGRACIÓN A TRAVÉS DE NOVA

Nova soporta a Docker como un hipervisor permitiendo desplegar contenedores en vez de máquinas virtuales con la misma API.

El controlador de Docker para Nova tiene un cliente HTTP embebido que conversa con la API interna de Docker a través de un socket. Usa la API HTTP para controlar contenedores y obtener información sobre ellos, obtiene imágenes del servicio de imágenes de OpenStack, Glance, y los carga en el sistema de archivos de Docker. Las imágenes pueden ser cargadas en Glance exportándolas desde Docker. Versiones anteriores de este driver requerían ejecutar un registro privado de Docker que se comunicaría con Glance, pero esto ya no es requerido.

La Figura 21 detalla la interacción de los componentes de integración a través de Nova.

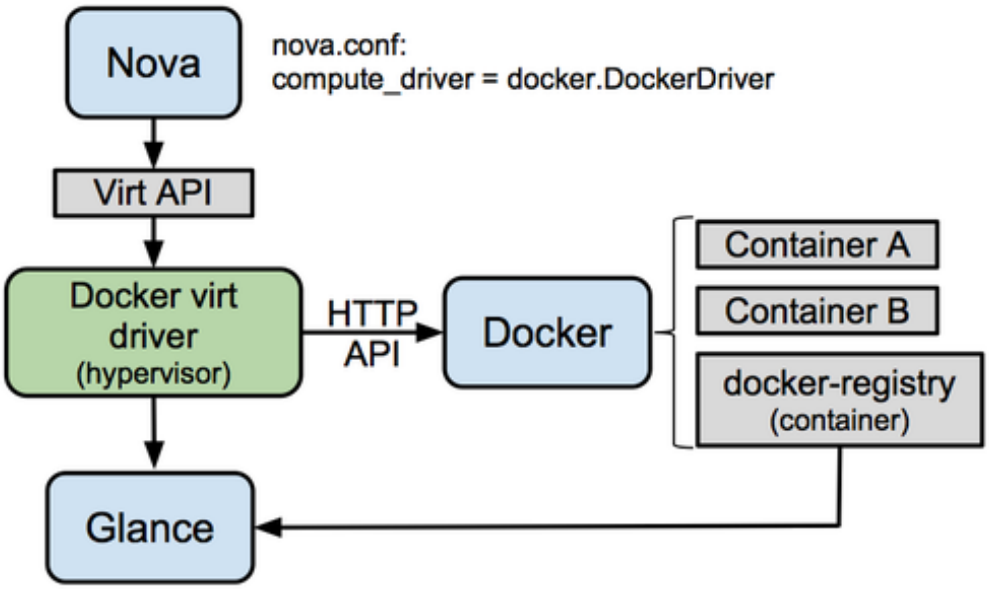

Figura 21. Controlador de Nova para Docker

Fuente: (Docker, Inc., 2014)

3.10.2.2. INTEGRACIÓN A TRAVÉS DE HEAT

Al usar Heat se permite orquestar los contenedores de Docker directamente en vez de tratar a Docker como un hipervisor de Nova.

Sin embargo, al usar Nova, las APIs estándar esperan ciertos comportamientos de máquina virtual que en muchos casos no tienen sentido en un contexto de contenedores de Docker. Así mismo, usar Docker como una máquina virtual de Nova también hace difícil exponer funcionalidades útiles y nativas de Docker, como enlazar contenedores.

El complemento de Heat permite a los usuarios desplegar y gestionar contenedores sobre despliegues tradicionales de OpenStack, haciéndolos compatibles con nubes OpenStack. El complemento para Heat ha sido aceptado dentro de OpenStack.

La Figura 22 detalla la interacción de los componentes de integración a través de Heat.

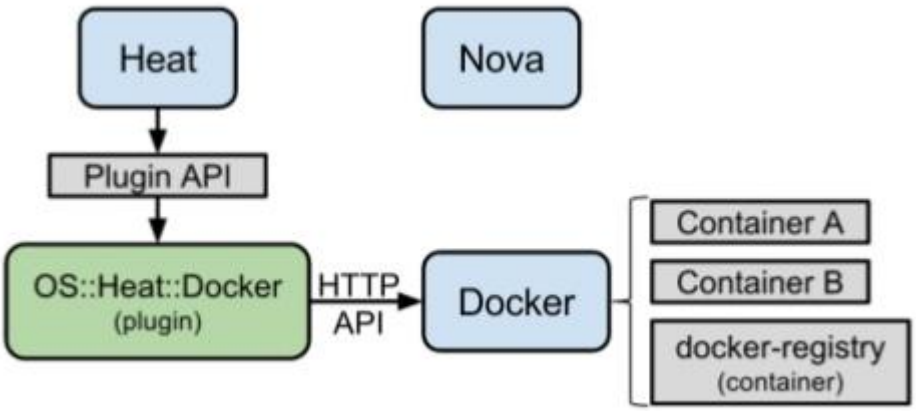

Figura 22. Complemento de Heat para Docker.

Fuente: (Docker, Inc., 2014)

CAPÍTULO 4. DISEÑO E IMPLEMENTACIÓN DE PROTOTIPO PARA PLATAFORMA COMO SERVICIO

4.1. DISEÑO DE ARQUITECTURA DEL PROTOTIPO

En la presente sección se describen los aspectos técnicos y de arquitectura que se usarán para crear el prototipo de Plataforma como Servicio.

4.1.1. DIAGRAMA DE PROTOTIPO

La Figura 23 incluye los componentes de las dos tecnologías, tanto OpenStack como Docker, y la forma cómo interactuarán en el prototipo.

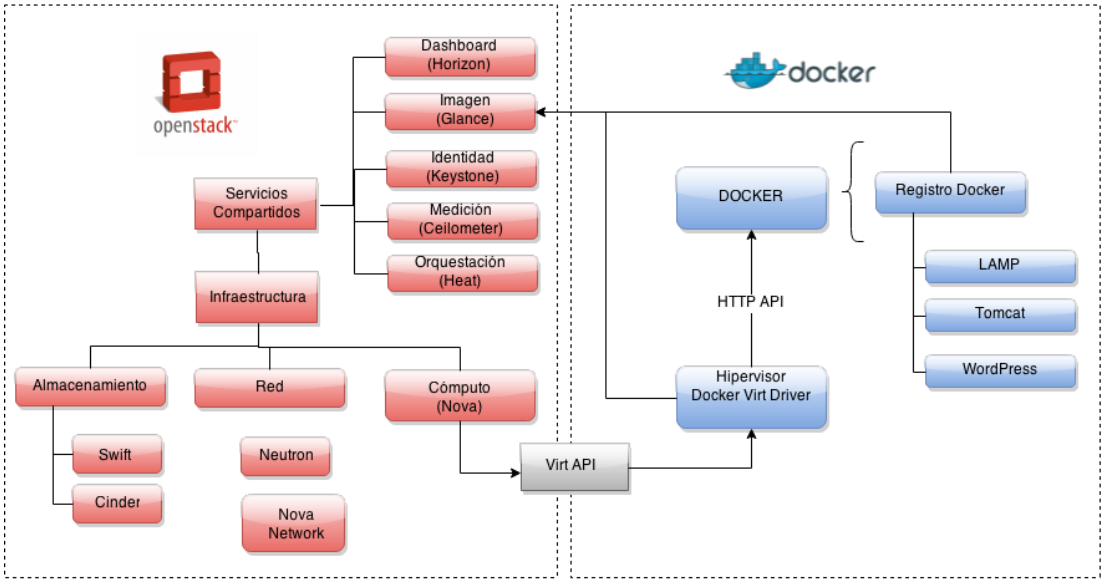

Figura 23. Diagrama de prototipo

Todos los componentes de OpenStack serán creados dentro de un equipo con sistema operativo de edición para servidor. Cada componente será un servicio de sistema operativo. Adicionalmente los servicios de Docker estarán integrados con el servicio de Cómputo como un hipervisor que apoyará la creación de máquinas virtuales a partir de imágenes Docker.

Del lado de Docker se cuenta con el hipervisor que conversa con el proceso Docker a través de APIs HTTP. El registro local de Docker contará con imágenes que luego serán montadas y consumidas apoyándose en los procesos de OpenStack para crear la infraestructura.

4.1.2. HARDWARE

Para el desarrollo del prototipo se cuenta con un computador con las características detalladas en la Tabla 14.

Tabla 14. Características de hardware para prototipo

Elemento	Valor
Marca	Toshiba
Modelo	Satellite L845
Procesador	Intel Core i5 3ra. Generación
Memoria RAM	10 GB
Disco	500 GB
Sistema Operativo	Ubuntu Server 14.04 LTS

Se decide hacer uso de una máquina física para la elaboración del prototipo para poder contar con todos los recursos del sistema dedicados para máquinas virtuales e imágenes. Esta decisión también facilita la gestión de red para el ambiente del prototipo.

4.2. INSTALACIÓN Y CONFIGURACIÓN DE COMPONENTES

La instalación de componentes incluye la instalación del sistema operativo y accesorios para facilitar el proceso de configuración. Adicionalmente incluye la configuración del ambiente y la descarga de los paquetes necesarios para ejecutar OpenStack y Docker dentro de la máquina utilizada.

Una vez instalado y configurado el sistema operativo, se procede con la instalación de los componentes de OpenStack y Docker. El detalle de la instalación del sistema operativo se incluye en el Anexo 6.

Se crea la cuenta de usuario que será la propietaria de los componentes del prototipo.

Nombre de usuario: stack
Contraseña: Pass4dh0c

Para la creación de la cuenta se ejecuta el comando

```
sudo adduser stack
```

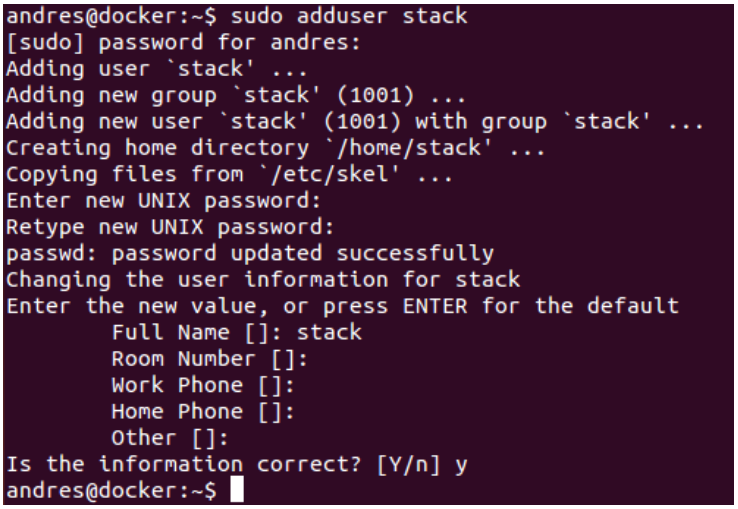

Figura 24. Creación de usuario propietario de componentes del prototipo

Se otorgan permisos de ejecución administrativos para el usuario stack por medio de la configuración del comando sudo. Para esto se ejecuta el comando.

```
sudo visudo
```

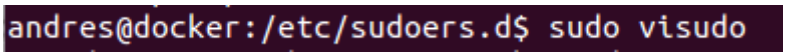

Figura 25. Modificar configuración de comando sudo

En el editor de texto por defecto se abre el archivo /etc/sudoers. Se incluye el siguiente registro:

```
stack ALL=(ALL) NOPASSWD: ALL
```

Esto permite que el usuario stack pueda obtener permisos de superusuario por medio del comando sudo y que no se le solicite contraseña. Se guardan los cambios realizados.

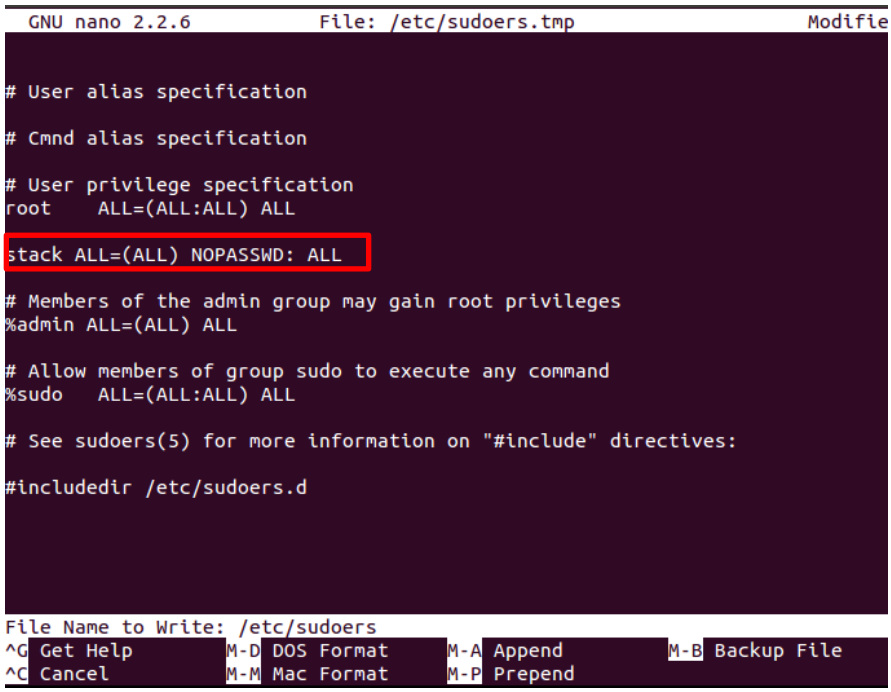

Figura 26. Registro de permisos para usuario stack

Se cierra la sesión con la cuenta creada durante la instalación y se ingresa con el usuario stack, para proceder con la configuración de los componentes del prototipo.

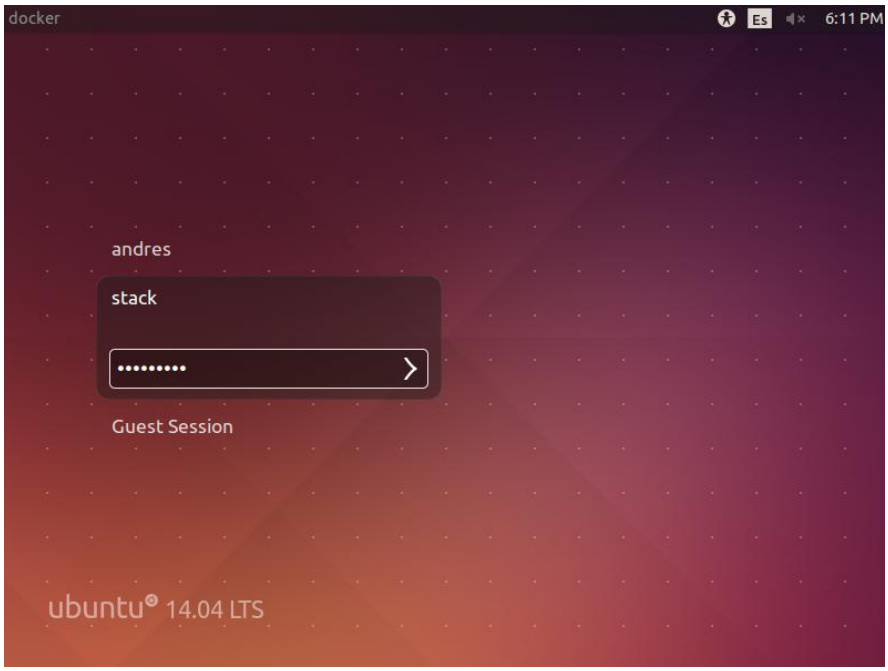

Figura 27. Inicio de sesión como usuario stack

Se asigna una dirección IP estática al equipo. Para esto se modifica el archivo /etc/interfaces registrando los valores.

```
auto eth0
iface eth0 inet static
     address 192.168.0.120
     netmask 255.255.255.0
     network 192.168.0.0
     broadcast 192.168.0.255
     gateway 192.168.0.1
     dns-nameservers 192.168.0.1
```

```
stack@docker:/etc/network$ cat interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 192.168.0.120
        netmask 255.255.255.0
        network 192.168.0.0
        broadcast 192.168.0.255
        gateway 192.168.0.1
        dns-nameservers 192.168.0.1
stack@docker:/etc/network$
```

Figura 28. Configuración de dirección IP estática

Una vez configurada la red se realiza la descarga e instalación de los paquetes de prototipo. El paso inicial es instalar los componentes del gestor de versiones Git. Se ejecuta el comando:

```
sudo apt-get install git -y
```

```
stack@docker:~$ sudo apt-get install git -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  firefox-locale-en
Use 'apt-get autoremove' to remove it.
The following extra packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-arch git-bzr git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 93 not upgraded.
Need to get 3,346 kB of archives.
After this operation, 21.6 MB of additional disk space will be used.
Get:1 http://ec.archive.ubuntu.com/ubuntu/ trusty/main liberror-perl all 0.17-1
.1 [21.1 kB]
Get:2 http://ec.archive.ubuntu.com/ubuntu/ trusty-updates/main git-man all 1:1.
9.1-1ubuntu0.1 [698 kB]
Get:3 http://ec.archive.ubuntu.com/ubuntu/ trusty-updates/main git amd64 1:1.9.
1-1ubuntu0.1 [2,627 kB]
Fetched 3,346 kB in 6s (501 kB/s)
Selecting previously unselected package liberror-perl.
(Reading database ... 99463 files and directories currently installed.)
Preparing to unpack .../liberror-perl_0.17-1.1_all.deb ...
Unpacking liberror-perl (0.17-1.1) ...
Selecting previously unselected package git-man.
```

Figura 29. Instalación de gestor de versiones de software Git

Se descargan los paquetes de OpenStack. Para el prototipo se utiliza el paquete conocido como devstack, que permite la instalación de todos los

procesos y componentes de OpenStack necesarios en una sola máquina. Se ejecuta el comando:

```
git clone https://git.openstack.org/openstack-dev/devstack
```

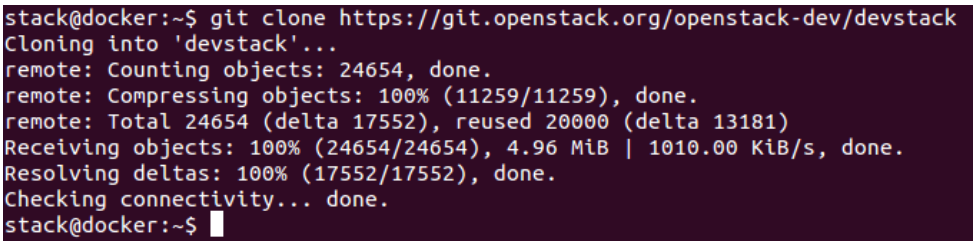

Figura 30. Descargar de componentes de OpenStack

Se descargan los componentes de Docker usando el comando:

```
sudo git clone https://git.openstack.org/stackforge/nova-docker /opt/stack/nova-docker
```

Figura 31. Descarga de componentes de Docker

Adicionalmente se descargan los componentes del proyecto OpenStack Compute (nova). De esta forma se tiene acceso a los paquetes necesarios para integración inicial. Se ejecuta el comando:

```
sudo git clone https://git.openstack.org/openstack/nova /opt/stack/nova
```

```
stack@docker:~$ sudo git clone https://git.openstack.org/openstack/nova /opt/st
ack/nova
Cloning into '/opt/stack/nova'...
remote: Counting objects: 319719, done.
remote: Compressing objects: 100% (136079/136079), done.
remote: Total 319719 (delta 249555), reused 240750 (delta 173088)
Receiving objects: 100% (319719/319719), 97.44 MiB | 419.00 KiB/s, done.
Resolving deltas: 100% (249555/249555), done.
Checking connectivity... done.
stack@docker:~$
```

Figura 32. Descarga de componentes de nova

Se modifica el archivo prepare_devstack.sh para la integración de OpenStack y Docker. El archivo se encuentra en la ruta:

`/opt/stack/nova-docker/contrib/devstack`

Se modifica el valor INSTALLDIR para que contenga la ruta en la que se encuentran los archivos de devstack. La ruta es:

`/home/stack`

```
  GNU nano 2.2.6          File: prepare_devstack.sh                  Modified

#!/bin/bash
set -xe

env

NOVADOCKERDIR=$(readlink -f $(dirname $0)/../..)
INSTALLDIR=${INSTALLDIR:-/home/stack}

cp $NOVADOCKERDIR/contrib/devstack/extras.d/70-docker.sh $INSTALLDIR/devstack/$
cp $NOVADOCKERDIR/contrib/devstack/lib/nova_plugins/hypervisor-docker $INSTALL$

cat - <<-EOF >> $INSTALLDIR/devstack/localrc
export VIRT_DRIVER=docker
export DEFAULT_IMAGE_NAME=cirros
export NON_STANDARD_REQS=1
export IMAGE_URLS=" "
EOF

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Figura 33. Afinamiento de script prepare_devstack.sh

Del lado de devstack se crea el archivo local.conf para determinar las variables de entorno que regirán el despliegue de los procesos y servicios de OpenStack. El posible copiar una plantilla del archivo local.conf desde la ruta:

```
/home/stack/devstack/samples
```

Una vez copiado el archivo se debe modificar para incluir los valores:

```
FLOATING_RANGE=192.168.0.224/27
FIXED_RANGE=10.11.12.0/24
FIXED_NETWORK_SIZE=256
FLAT_INTERFACE=p3p1
SERVICE_TOKEN=Pass4dh0c
ADMIN_PASSWORD=Pass4dh0c
MYSQL_PASSWORD=Pass4dh0c
RABBIT_PASSWORD=Pass4dh0c
SERVICE_PASSWORD=$ADMIN_PASSWORD
VIRT_DRIVER=docker
```

Estos valores configuran las claves de acceso, los rangos de direcciones IP y configuraciones de red y el uso de Docker como hipervisor para cargar instancias de OpenStack. A continuación se describen los valores utilizados

- FLOATING_RANGE: rango de direcciones IP definidas para ser usadas por los servicios de Neutron.

- FIXED_RANGE: rango de direcciones IP de la red interna creada para la comunicación entre instancias.

- FIXED_NETWORK_SIZE: la cantidad de direcciones IP creadas para la red interna de las instancias de OpenStack.

- FLAT_INTERFACE: la tarjeta de red física que será utilizada para tramitar las comunicaciones entre las instancias y la red física.

- SERVICE_TOKEN y SERVICE_PASSWORD: son los valores que usarán para autenticar los servicios de OpenStack con el servicio de gestión de identidad Keystone.

- ADMIN_PASSWORD: la contraseña de los usuarios creados por defecto. Los nombres de usuario que usarán esta contraseña son admin y demo.

- MYSQL_PASSWORD: la contraseña de administración de la base de datos de OpenStack.

- RABBIT_PASSWORD: la contraseña de administración del servicio de cola de mensajes de OpenStack.

- VIRT_DRIVER: el driver de virtualización que se utilizará para crear las instancias de OpenStack. En este caso el driver utilizado es Docker, lo que permite la integración de las herramientas seleccionadas para el prototipo.

Una vez configurados los archivos, según lo descrito se ejecuta el comando de configuración de servicios de devstack.

```
stack.sh
```

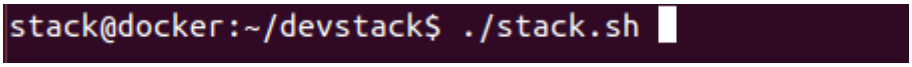

Figura 34. Instalación de componentes integrados

4.3. IMÁGENES A SER USADAS

Las imágenes a ser utilizadas son una muestra de la capacidad de Docker de crear ambientes portables, lo que facilita la entrega de Plataforma como Servicio.

A continuación se detallan las 3 imágenes que serán utilizadas dentro del prototipo, las cuales estarán incluidas dentro del registro de Docker y quedarán disponibles para el servicio de imágenes de OpenStack (glance).

4.3.1. IMAGEN LAMP

Imagen para desarrollo web en lenguaje PHP. Una plataforma de desarrollo muy utilizada para aplicaciones web. LAMP son las siglas de cada uno de los componentes de la plataforma.

- Sistema operativo Linux
- Servidor web Apache
- Base de datos MySQL
- Lenguaje de programación PHP

La Tabla 15 describe los pasos para obtener esta imagen.

Tabla 15. Pasos para obtener imagen LAMP

Tarea	Comando
Buscar imagen en el registro público de Docker	docker search lamp
Obtener imagen desde el registro público de Docker	docker pull tutum/lamp
Cargar imagen al registro local de Docker y almacenarla en el servicio de imágenes de OpenStack	docker save tutum/lamp | glance image-create --container-format=docker --disk-format=raw --name tutum/lamp

4.3.2. IMAGEN TOMCAT

Imagen para levantar un servidor de aplicaciones Apache Tomcat para ambiente de programación y ejecución de aplicaciones web Java Enterprise Edition (JEE). Apache Tomcat es uno de los servidores de aplicaciones de código abierto más populares.

El componente instalado dentro de esta imagen es solamente el servidor de aplicaciones. No cuenta con una base de datos instalada.

La Tabla 16 describe los pasos para obtener esta imagen.

Tabla 16. Pasos para obtener imagen Tomcat

Tarea	Comando
Buscar imagen en el registro público de Docker	docker search tomcat
Obtener imagen desde el registro público de Docker	docker pull tutum/tomcat
Cargar imagen al registro local de Docker y almacenarla en el servicio de imágenes de OpenStack	docker save tutum/tomcat | glance image-create --container-format=docker --disk-format=raw --name tutum/tomcat

4.3.3. IMAGEN WORDPRESS

Imagen para levantar una plataforma de gestión de contenido web y de publicación. WordPress es uno de los sistemas de gestión de contenidos más populares por la facilidad para crear sitios y portales web así como para gestionar blogs y publicación de contenido. Esta imagen permite levantar una instalación base de WordPress para que a partir de esta los desarrolladores puedan crear sitios web personalizados.

La Tabla 17 describe los pasos para obtener esta imagen.

Tabla 17. Pasos para obtener imagen WordPress

Tarea	Comando
Buscar imagen en el registro público de Docker	docker search wordpress
Obtener imagen desde el registro público de Docker	docker pull tutum/wordpress
Cargar imagen al registro local de Docker y almacenarla en el servicio de imágenes de OpenStack	docker save tutum/wordpress | glance image-create --container-format=docker --disk-format=raw --name tutum/wordpress

CAPÍTULO 5. PRUEBAS DEL PROTOTIPO

5.1. PRUEBAS FUNCIONALES

En el presente apartado se detallan los casos de prueba que serán aplicados al prototipo.

5.1.1. MODELO DE CASOS DE PRUEBA

Cada uno de los casos de prueba se define con el nombre del caso, el objetivo a conseguir, los pasos que deben ser realizados, los resultados esperados y, finalmente, se registran los resultados obtenidos.

En los que casos que así lo permitan se incluirán capturas de pantalla de los resultados obtenidos.

En la Tabla 18 se presenta la estructura de los casos de prueba a utilizar.

Tabla 18. Modelo de caso prueba

Nombre de caso de prueba	N° ##. <Descripción>
Objetivo del caso de prueba	<Objetivo del caso de prueba>
Procedimiento de prueba <Detalle de los pasos a realizarse como parte de la prueba>	
Resultados esperados <Descripción de los resultados que se esperan luego de la ejecución del procedimiento de prueba>	
Resultados obtenidos <Registro de los resultados obtenidos luego de la ejecución del procedimiento de prueba>	
Análisis de Resultados <Se analizan los resultados obtenidos y se determina si el caso de prueba ha sido Satisfactorio o Fallido>	

5.1.2. CASOS DE PRUEBA Y REGISTRO DE RESULTADOS

En las siguientes tablas se detallan los casos de prueba que se aplicaron al prototipo y los resultados obtenidos para cada uno.

Tabla 19. Resultado de Caso de Prueba N°01. Inicialización de ambiente

Nombre de caso de prueba	N° 01. Inicialización de ambiente
Objetivo del caso de prueba	Validar que el ambiente se encuentre operativo para poder ejecutar las operaciones relativas a las pruebas
Procedimiento de prueba - Ingresar al sistema operativo con las credenciales del usuario stack ▪ Usuario: stack ▪ Contraseña: Pass4dh0c - Acceder a la ruta de instalación de OpenStack ▪ cd devstack - Cargar variables de entorno ▪ . ./openrc - Ejecutar el comando de inicialización de ambiente ▪ ./rejoin-stack.sh - Ejecutar los siguientes comandos para validar que OpenStack se encuentra activo. ▪ glance image-list ▪ nova list - Ejecutar el siguiente comando para validar que OpenStack se encuentra activo. ▪ docker ps	
Resultados esperados 1. Luego de ejecutar el comando rejoinstack.sh se espera una serie de mensajes indicando operación de los procesos relacionados con el prototipo. 2. Luego de ejecutar el comando glance image-list se espera una lista que incluya las imágenes cargadas en el registro local. 3. Luego de ejecutar el comando nova list se espera una lista vacía de instancias, ya que ninguna instancia está siendo ejecutada. 4. Luego ejecutar el comando docker ps se espera una lista de procesos Docker vacía, ya que ningún proceso está siendo ejecutado.	
Resultados obtenidos 1. Ejecución del comando rejoinstack.sh.	

Continúa

Nombre de caso de prueba	N° 01. Inicialización de ambiente

```
stack@docker:~$ cd devstack/
stack@docker:~/devstack$ ls
accrc          exercise.sh       LICENSE          run_tests.sh     tempest.log
clean.sh       extras.d          local.conf       samples          tests
doc            files             localrc          setup.cfg        tools
driver_certs   functions         MAINTAINERS.rst  setup.py         tox.ini
eucarc         functions-common  openrc           stackrc          unstack.sh
exerciserc     HACKING.rst       README.md        stack-screenrc
exercises      lib               rejoin-stack.sh  stack.sh
stack@docker:~/devstack$ . ./openrc
stack@docker:~/devstack$ ./rejoin-stack.sh
```

Figura 35. Ejecución de comando rejoin-stack.sh

```
2015-03-03 21:58:13.499 DEBUG heat-api-cloudwatch [-] clients_keystone.endpoint_
type = None from (pid=4134) log_opt_values /usr/local/lib/python2.7/dist-package
s/oslo_config/cfg.py:2073
2015-03-03 21:58:13.499 DEBUG heat-api-cloudwatch [-] clients_keystone.insecure
     = None from (pid=4134) log_opt_values /usr/local/lib/python2.7/dist-package
s/oslo_config/cfg.py:2073
2015-03-03 21:58:13.500 DEBUG heat-api-cloudwatch [-] clients_keystone.key_file
     = None from (pid=4134) log_opt_values /usr/local/lib/python2.7/dist-package
s/oslo_config/cfg.py:2073
2015-03-03 21:58:13.500 DEBUG heat-api-cloudwatch [-] revision.heat_revision
     = unknown from (pid=4134) log_opt_values /usr/local/lib/python2.7/dist-pack
ages/oslo_config/cfg.py:2073
2015-03-03 21:58:13.500 DEBUG heat-api-cloudwatch [-] ****************************
************************************************** from (pid=4134) log_opt_v
alues /usr/local/lib/python2.7/dist-packages/oslo_config/cfg.py:2075
2015-03-03 21:58:13.500 INFO heat.api.cloudwatch [-] Starting Heat CloudWatch AP
I on 0.0.0.0:8003
2015-03-03 21:58:13.515 INFO eventlet.wsgi.server [-] Starting single process se
rver
2015-03-03 21:58:13.516 DEBUG eventlet.wsgi.server [-] (4134) wsgi starting up o
n http://0.0.0.0:8003/ from (pid=4134) write /opt/stack/heat/heat/common/wsgi.py
:183
```

Figura 36. Script rejoin-stack.sh ejecutado

2. Ejecución del comando glance image-list

```
stack@docker:~/devstack$ glance image-list
+--------------------------------------+-------------------------+-------------+------------------+-----------+--------+
| ID                                   | Name                    | Disk Format | Container Format | Size      | Status |
+--------------------------------------+-------------------------+-------------+------------------+-----------+--------+
| 13d2e0b1-9cef-4950-b587-8e4aad6526f4 | cirros                  | raw         | docker           | 4705280   | active |
| f59ae15f-2b0e-44b9-ba3b-666a7582591a | java                    | raw         | docker           | 601053184 | active |
| dfa62677-8db0-40cd-a1cd-f73289275910 | python                  | raw         | docker           | 778168320 | active |
| df50ab47-fdbf-4aef-be3c-58045f892c5b | ruby                    | raw         | docker           | 799169536 | active |
| 3648ae56-67ff-4f4a-883b-13744e4bd8c3 | tutum/lamp              | raw         | docker           | 450159616 | active |
| 84ae26a6-bc1b-4979-9dca-fdc7c950c997 | tutum/quickstart-python | raw         | docker           | 779131392 | active |
| 292491a0-262b-4024-9f7c-50a0385b0886 | tutum/tomcat            | raw         | docker           | 564718080 | active |
| 70ad9343-faf5-4538-bb16-64522ae67be5 | tutum/wordpress         | raw         | docker           | 517649920 | active |
| e7570d73-ebd4-4ddb-bc97-f540f83a0b37 | wordpress               | raw         | docker           | 495927808 | active |
+--------------------------------------+-------------------------+-------------+------------------+-----------+--------+
```

Figura 37. Lista de imágenes disponibles

Continúa

Nombre de caso de prueba	N° 01. Inicialización de ambiente

3. Ejecución del comando nova list

```
stack@docker:~/devstack$ nova list
+--------------------------------------+-------------+--------+------------+-------------+------------------+
| ID                                   | Name        | Status | Task State | Power State | Networks         |
+--------------------------------------+-------------+--------+------------+-------------+------------------+
```

Figura 38. Lista de instancias vacía.

4. Ejecución del comando docker ps

```
stack@docker:~/devstack$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

Figura 39. Lista de procesos Docker

Análisis de Resultados

El resultado del caso de prueba N°01 es Satisfactorio.

- Una vez configurado el ambiente es posible inicializar y detener los servicios del prototipo.
- La ejecución de comandos del prototipo permiten comprobar que los servicios se encuentran operativos y que pueden ser consumidos.

Tabla 20. Resultado de Caso de Prueba N°02. Acceso a interfaz gráfica de administración

Nombre de caso de prueba	N° 02. Acceso a interfaz gráfica de administración
Objetivo del caso de prueba	Acceder a la interfaz gráfica de administración de OpenStack (horizon) y navegar entre sus opciones.

Procedimiento de prueba

- Inicializar el ambiente del prototipo.
- Abrir el navegador web Mozilla Firefox
- Acceder a la URL http://192.168.0.120
- Ingresar con las credenciales de usuario administrador.
  - Usuario: admin
  - Contraseña: Pass4dh0c
- Acceder a la opción Proyecto > Cómputo > Visión General
- Acceder a la opción Proyecto > Cómputo > Instancias
- Acceder a la opción Proyecto > Cómputo > Imágenes

Resultados esperados

1. Acceso a la interfaz gráfica de administración.
2. Acceso a las opciones descritas donde se visualiza el estado de cada uno de los elementos mostrados.

Resultados obtenidos

1. Acceso a la interfaz gráfica de administración usando las credenciales solicitadas.

Continúa

Nombre de caso de prueba	N° 02. Acceso a interfaz gráfica de administración

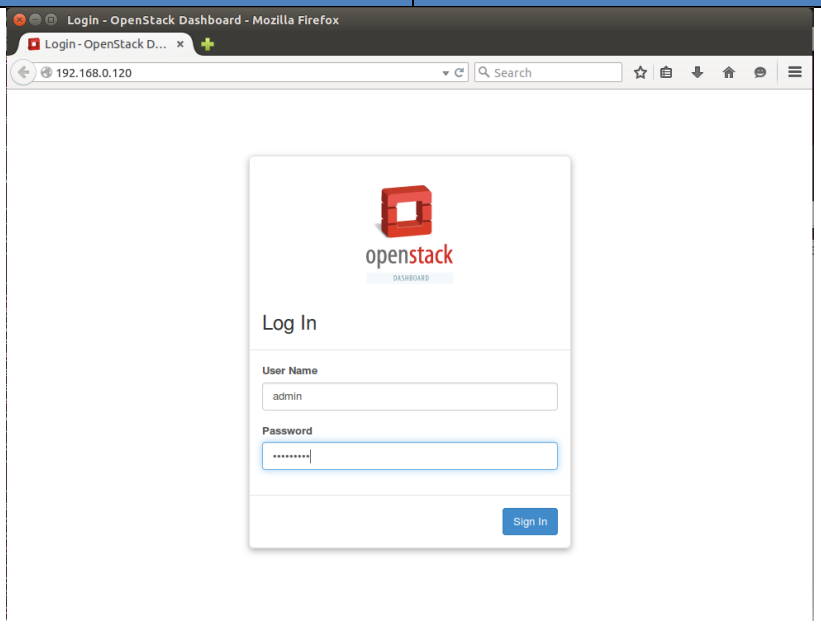

Figura 40. Pantalla de ingreso a administración gráfica

2. Acceso a las opciones descritas:

- Proyecto > Cómputo > Visión General

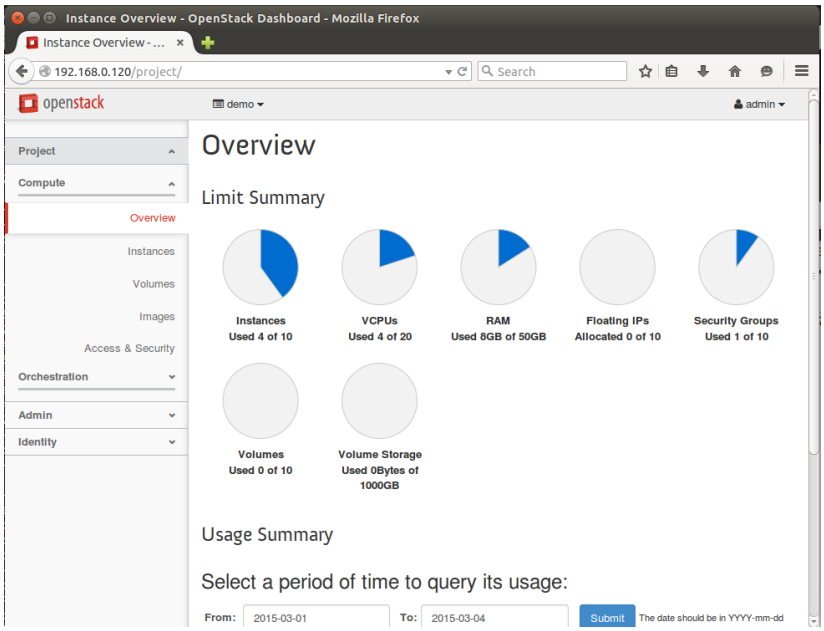

Figura 41. Visión general

Continúa

Nombre de caso de prueba	N° 02. Acceso a interfaz gráfica de administración

- Proyecto > Cómputo > Instancias

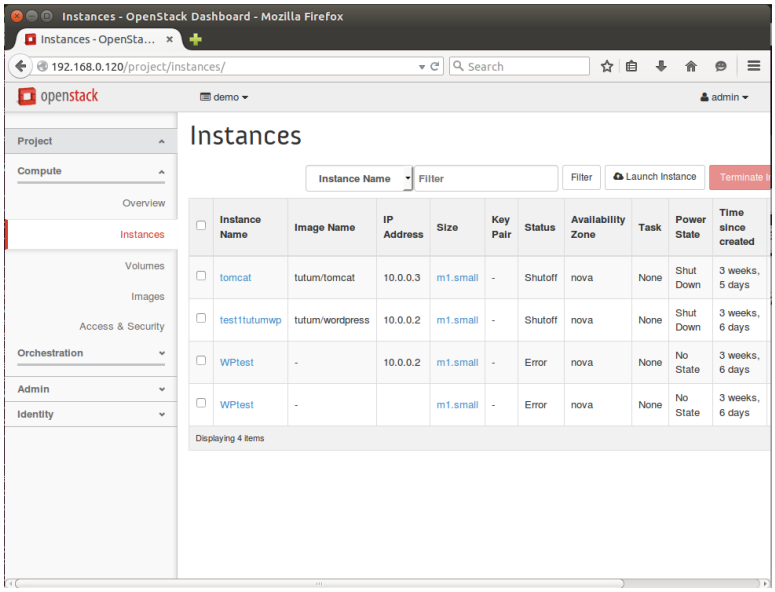

Figura 42. Instancias

- Proyecto > Cómputo > Imágenes

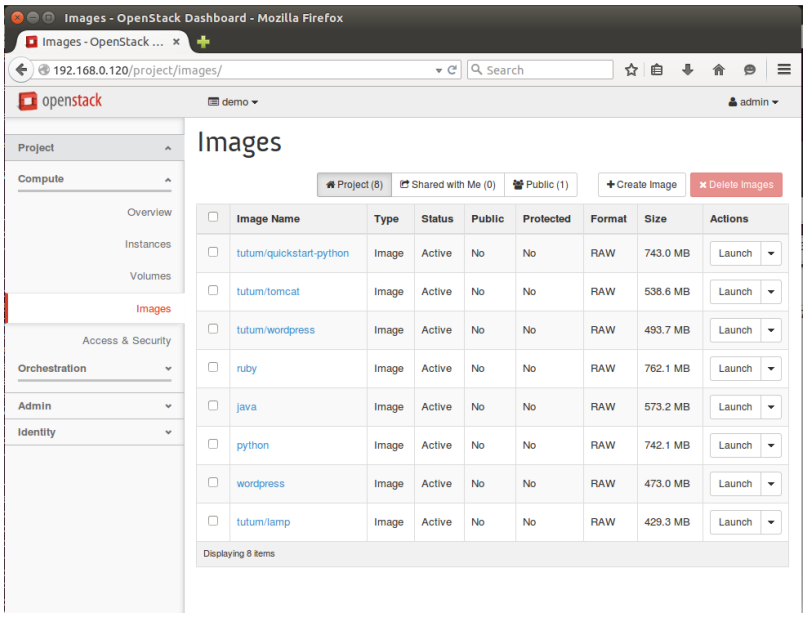

Figura 43. Imágenes

Análisis de Resultados

El resultado del caso de prueba N°02 es Satisfactorio.

- Se valida el estado de los servicios de interfaz gráfica de administración.
- Se verifica el estado de los componentes a través de la interfaz gráfica.

Tabla 21. Resultado Caso de Prueba N°03. Inicio de instancia con imagen LAMP

Nombre de caso de prueba	N° 03. Inicio de instancia con imagen LAMP
Objetivo del caso de prueba	Inicializar una instancia basada en la imagen LAMP.

Procedimiento de prueba

- Ingresar a la interfaz gráfica de administración.
- Acceder a la opción Proyecto > Cómputo > Instancias
- Presionar el botón Lanzar instancia
- Elegir las opciones de instancia siguientes:
  - Zona de disponibilidad: nova
  - Nombre de instancia: lamp01
  - Sabor: m1.small
  - Conteo de instancias: 1
  - Fuente de inicio de instancia: Inicio desde imagen
  - Nombre de imagen: tutum/lamp
- Presionar el botón Iniciar instancia.
- Probar acceso a instancia por medio de un navegador web. Apuntar hacia la dirección IP asignada.

Resultados esperados

1. Instancia LAMP creada.
2. Instancia LAMP iniciada.
3. Acceso a página “Hello World” cargada dentro de la imagen.

Resultados obtenidos

1. Instancia LAMP creada en prototipo con características especificadas.

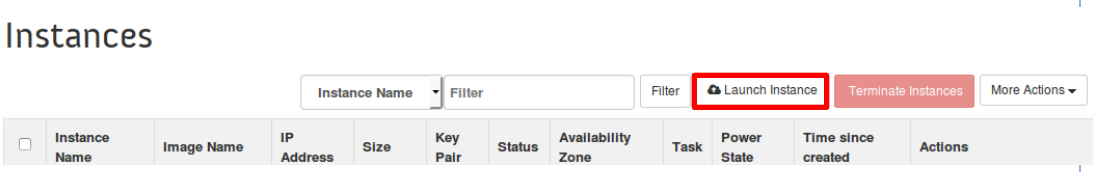

Figura 44. Botón Lanzar iniciar instancia

Continúa

Nombre de caso de prueba	N° 03. Inicio de instancia con imagen LAMP

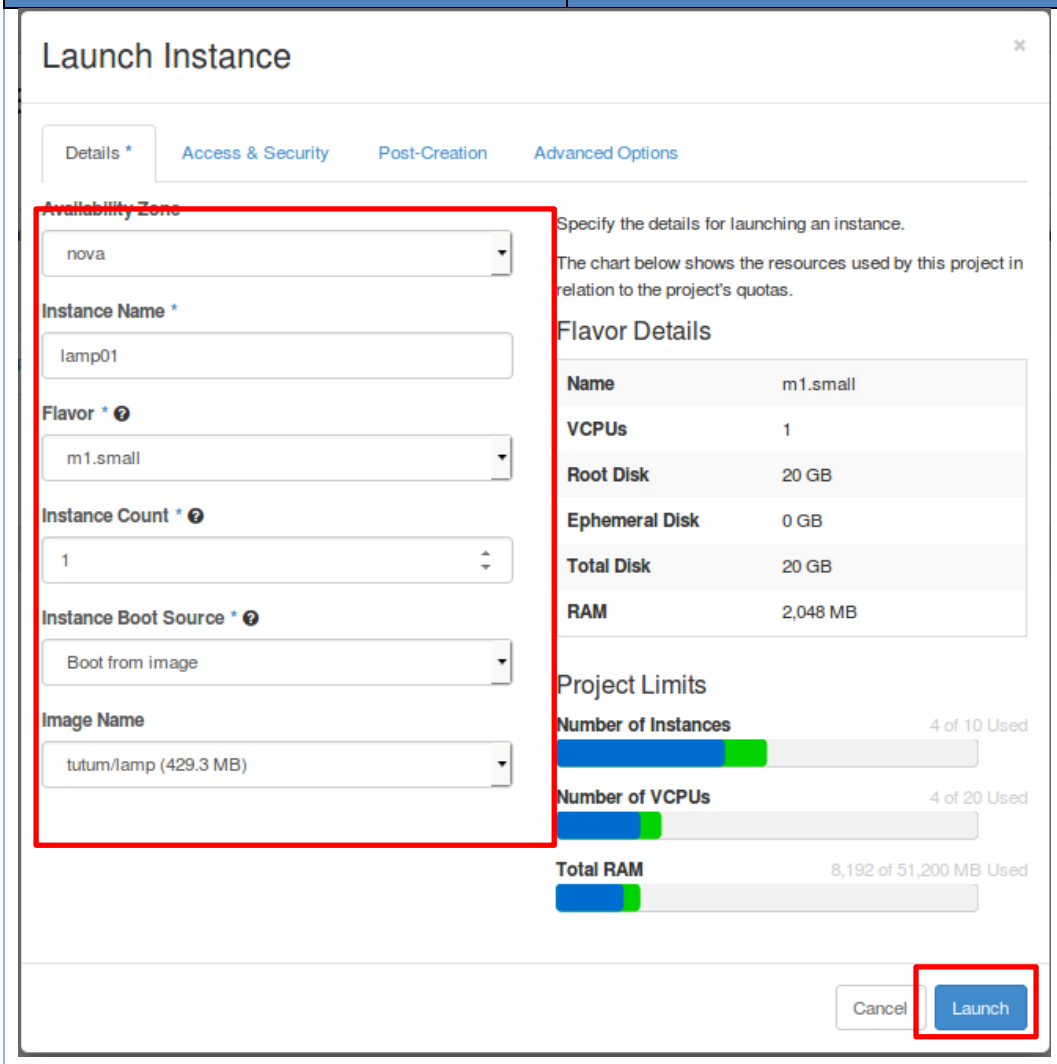

Figura 45. Detalles de instancia LAMP y botón Iniciar

2. Instancia LAMP iniciada en ambiente.

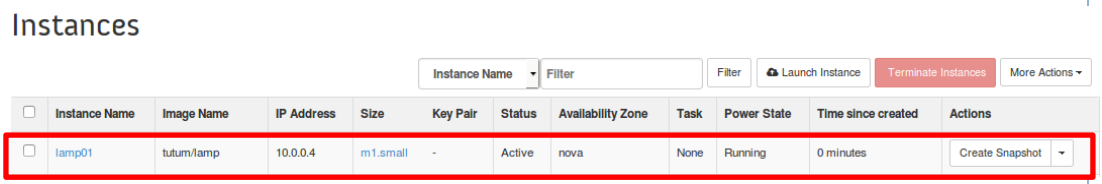

Figura 46. Instancia LAMP Iniciada

3. Se muestra la página “Hello World” al acceder a la dirección IP de la instancia.
   http://10.0.0.4

Continúa

Nombre de caso de prueba	N° 03. Inicio de instancia con imagen LAMP

Figura 47. Página "Hello World de instancia LAMP.

Análisis de Resultados

El resultado del caso de prueba N°03 es Satisfactorio.

- Se ha creado la instancia a partir de imagen LAMP con las características solicitadas.
- Se ha iniciado la instancia.
- Se puede acceder a la página "Hello World" a través de la dirección IP de la instancia.

Tabla 22. Resultado de Caso de Prueba N°04. Inicio de instancia con imagen Tomcat.

Nombre de caso de prueba	N° 04. Inicio de instancia con imagen Tomcat
Objetivo del caso de prueba	Inicializar una instancia basada en la imagen Tomcat.

Procedimiento de prueba

- Ingresar a la interfaz gráfica de administración.
- Acceder a la opción Proyecto > Cómputo > Instancias
- Presionar el botón Lanzar instancia
- Elegir las opciones de instancia siguientes:
  - Zona de disponibilidad: nova
  - Nombre de instancia: tomcat01
  - Sabor: m1.small
  - Conteo de instancias: 1

Continúa

Nombre de caso de prueba	N° 04. Inicio de instancia con imagen Tomcat

- Fuente de inicio de instancia: Inicio desde imagen
- Nombre de imagen: tutum/tomcat

- Presionar el botón Iniciar instancia
- Desde la línea de comandos del sistema operativo ejecutar el comando:
  `docker ps`
- Desde la línea de comandos del sistema operativo ejecutar el comando:
  `docker exec <id contenedor> cat tomcat/conf/tomcat-users.xml`
- Probar acceso a instancia por medio de un navegador web. Apuntar hacia la dirección IP asignada usando el puerto 8080.
- Acceder a la opción Server Status y proporcionar las credenciales del usuario admin obtenidas del archivo tomcat-users.xml.

Resultados esperados

1. Instancia Tomcat creada.
2. Instancia Tomcat iniciada.
3. Obtener las credenciales del usuario admin de Tomcat.
4. Acceso a página de inicio de Apache Tomcat en ejecución dentro de la imagen.
5. Acceso a la opción Server Status de Tomcat.

Resultados obtenidos

1. Instancia Tomcat creada de acuerdo a los valores solicitados.

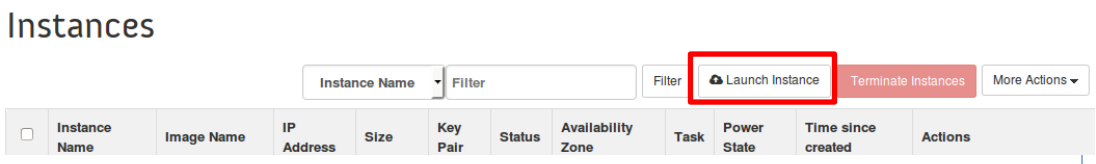

Figura 48. Botón Iniciar Instancia

Continúa

Nombre de caso de prueba	N° 04. Inicio de instancia con imagen Tomcat

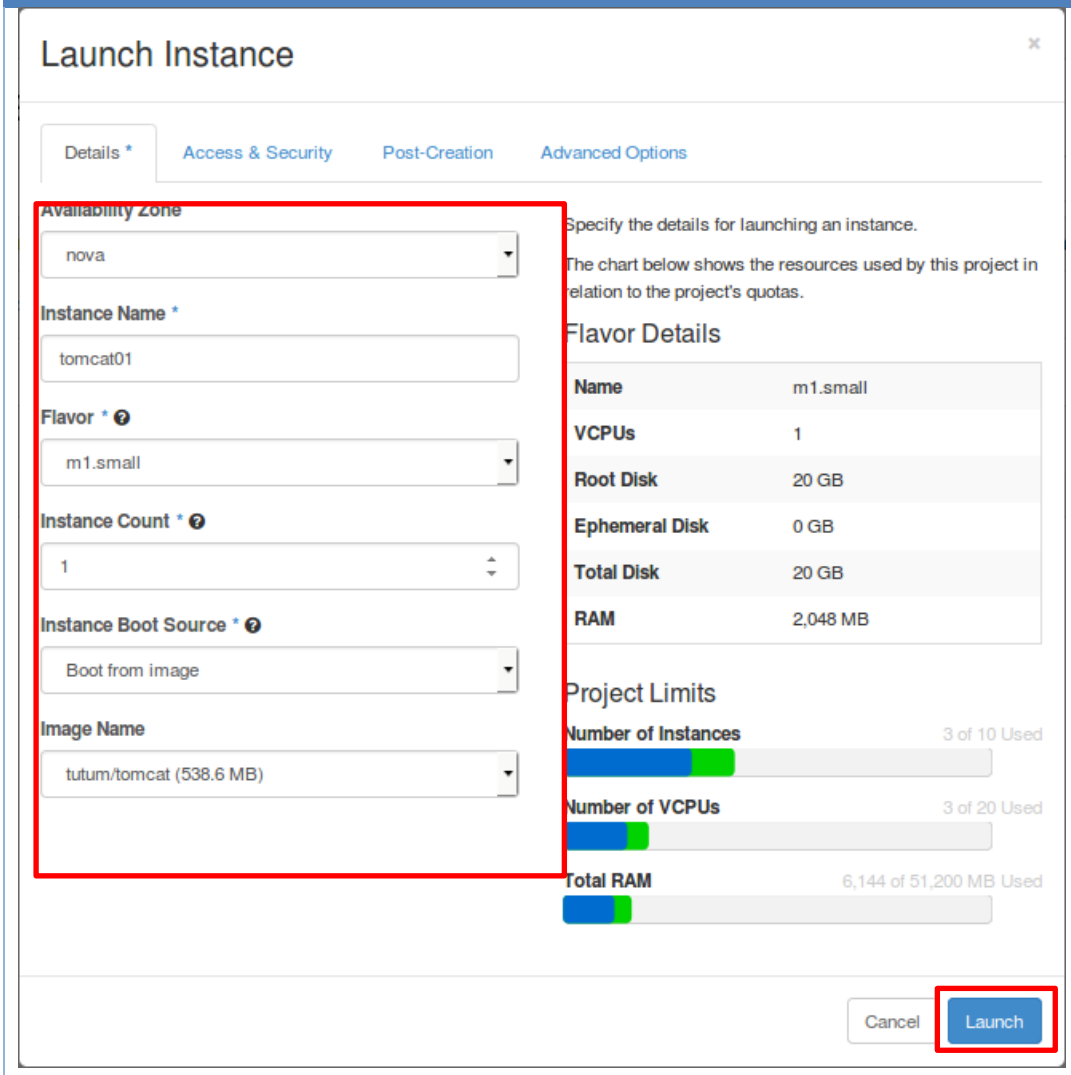

Figura 49. Detalles de instancia Tomcat y botón Iniciar

2. Instancia Tomcat iniciada.

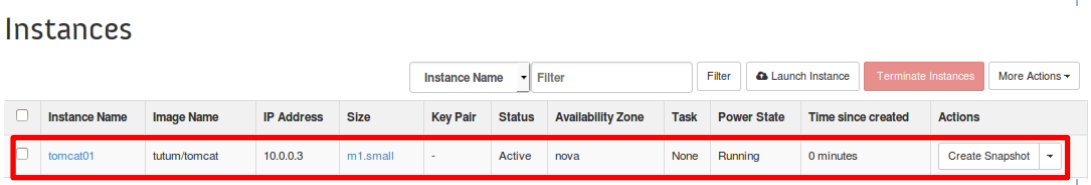

Figura 50. Instancia Tomcat iniciada

3. Credenciales de usuario admin de Tomcat obtenidas desde archivo de texto.

Continúa

Nombre de caso de prueba	N° 04. Inicio de instancia con imagen Tomcat

```
stack@docker:~/devstack$ docker ps
CONTAINER ID        IMAGE                COMMAND             CREATED
  STATUS              PORTS               NAMES
07475a8e59ea        tutum/tomcat:latest   "/run.sh"           22 seconds ago
  Up 21 seconds                           nova-034ce1a1-4809-40fb-8279-49ad7b464
aa4
```

Figura 51. Obtener ID de contenedor Docker

```
stack@docker:~/devstack$ docker exec 07475a8e59ea cat tomcat/conf/tomcat-users.x
ml
<?xml version='1.0' encoding='utf-8'?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at
```

Figura 52. Ejecutar comandos en contenedor Docker

```
<!--
  NOTE:  The sample user and role entries below are wrapped in a comment
  and thus are ignored when reading this file. Do not forget to remove
  <!.. ..> that surrounds them.
-->
<!--
  <role rolename="tomcat"/>
  <role rolename="role1"/>
  <user username="tomcat" password="tomcat" roles="tomcat"/>
  <user username="both" password="tomcat" roles="tomcat,role1"/>
  <user username="role1" password="tomcat" roles="role1"/>
-->

<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<role rolename="manager-jmx"/>
<role rolename="admin-gui"/>
<role rolename="admin-script"/>
<user username="admin" password="brOgv0YSD6nn" roles="manager-gui,manager-script
,manager-jmx,admin-gui, admin-script"/>
</tomcat-users>
```

Figura 53. Contenido de archivo de usuarios de Tomcat

4. Acceso a página de inicio de Apache Tomcat a través de la IP asignada.
   http://10.0.0.3:8080

Continúa

Nombre de caso de prueba	N° 04. Inicio de instancia con imagen Tomcat

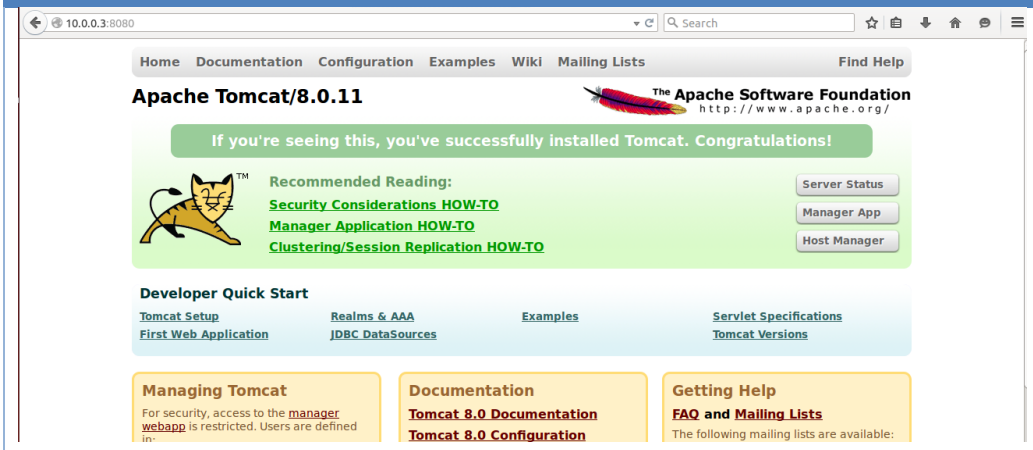

Figura 54. Página de inicio de Tomcat

5. Acceso a la opción Server Status de Tomcat usando las credenciales de usuario obtenidas.

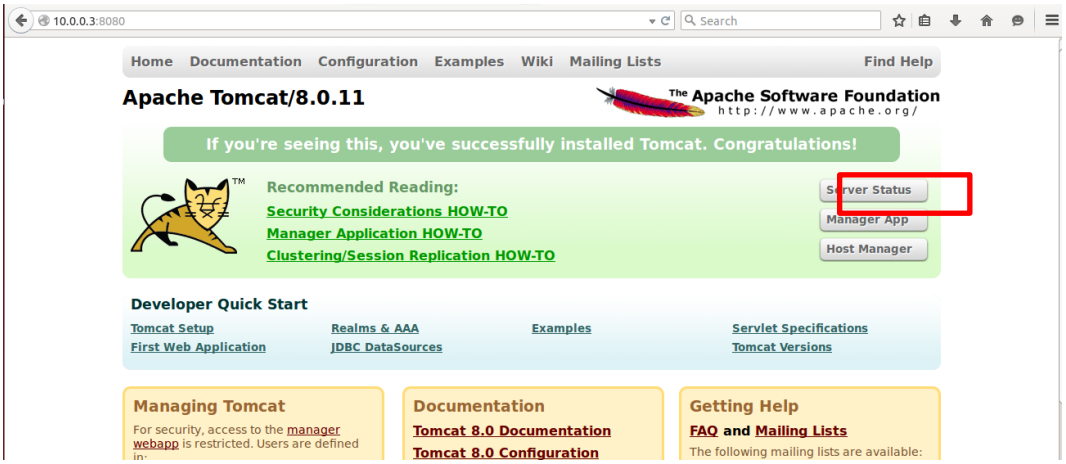

Figura 55. Opción Server Status

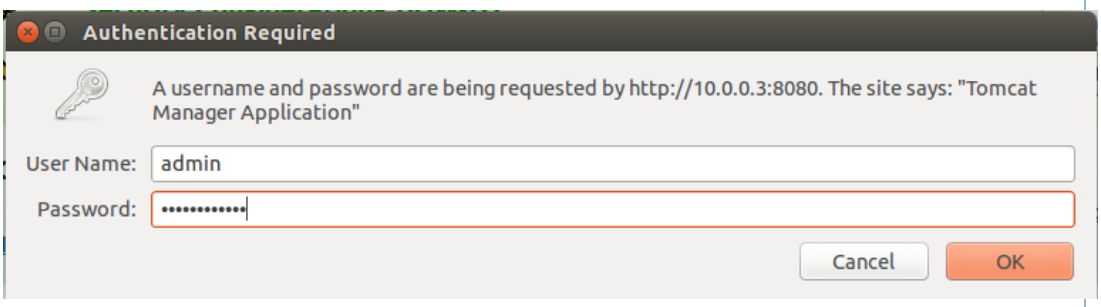

Figura 56. Autenticación con credenciales de Tomcat

Continúa

Nombre de caso de prueba	N° 04. Inicio de instancia con imagen Tomcat

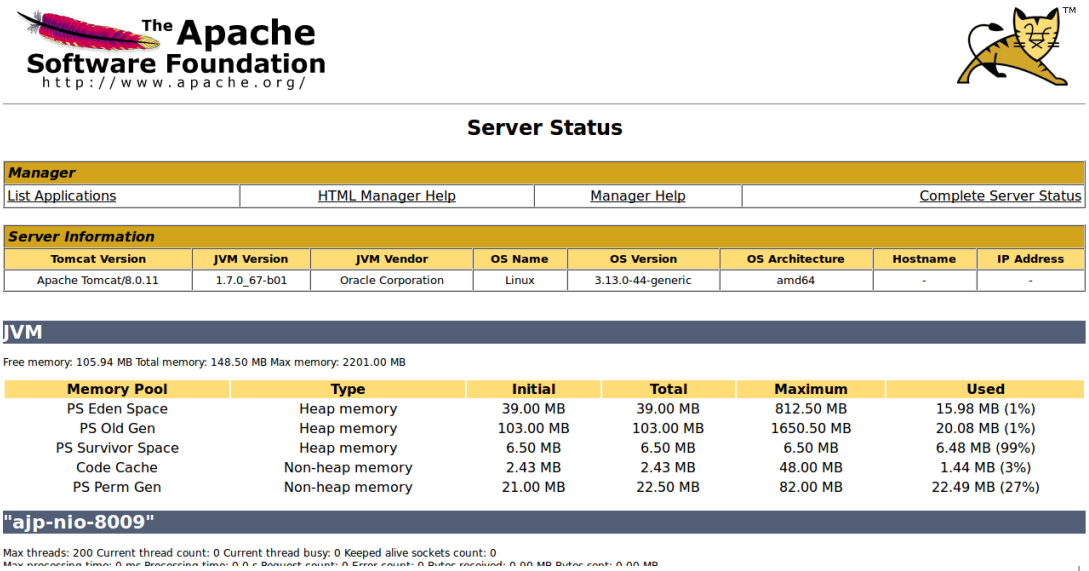

Figura 57. Estado de servidor Tomcat de la instancia

Análisis de Resultados

El resultado del caso de prueba N°04 es Satisfactorio.

- Se ha creado la instancia a partir de imagen Tomcat con las características solicitadas.
- Se ha iniciado la instancia.
- Se obtienen las credenciales de usuario admin interactuando con la instancia a través de línea de comandos.
- Se puede acceder a la página de inicio de Tomcat a través de la dirección IP y el puerto 8080.
- Se valida el estado del servidor Tomcat ingresando a la opción correspondiente con las credenciales de usuario obtenidas por línea de comandos.

Tabla 23. Resultado de Caso de Prueba N°05. Inicio de instancia con imagen WordPress.

Nombre de caso de prueba	N° 05. Inicio de instancia con imagen WordPress
Objetivo del caso de prueba	Inicializar una instancia basada en la imagen WordPress.
Procedimiento de prueba - Ingresar a la interfaz gráfica de administración. - Acceder a la opción Proyecto > Cómputo > Instancias	

Continúa

Nombre de caso de prueba	N° 05. Inicio de instancia con imagen WordPress

- Presionar el botón Lanzar instancia
- Elegir las opciones de instancia siguientes:
  - Zona de disponibilidad: nova
  - Nombre de instancia: wordpress01
  - Sabor: m1.small
  - Conteo de instancias: 1
  - Fuente de inicio de instancia: Inicio desde imagen
  - Nombre de imagen: tutum/wordpress
- Presionar el botón Iniciar instancia.
- Probar acceso a instancia por medio de un navegador web. Apuntar hacia la dirección IP.
- Ejecutar el proceso de instalación de WordPress.
- Acceder a la interfaz de administración de WordPress.

Resultados esperados

1. Instancia WordPress creada.
2. Instancia WordPress iniciada.
3. Acceso al proceso de instalación de WordPress.
4. Acceso a la interfaz de administración de WordPress.

Resultados obtenidos

1. Instancia WordPress creada.

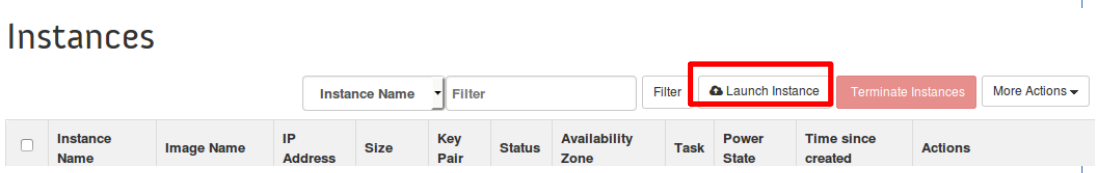

Figura 58. Botón Iniciar instancia

Continúa

Nombre de caso de prueba	N° 05. Inicio de instancia con imagen WordPress

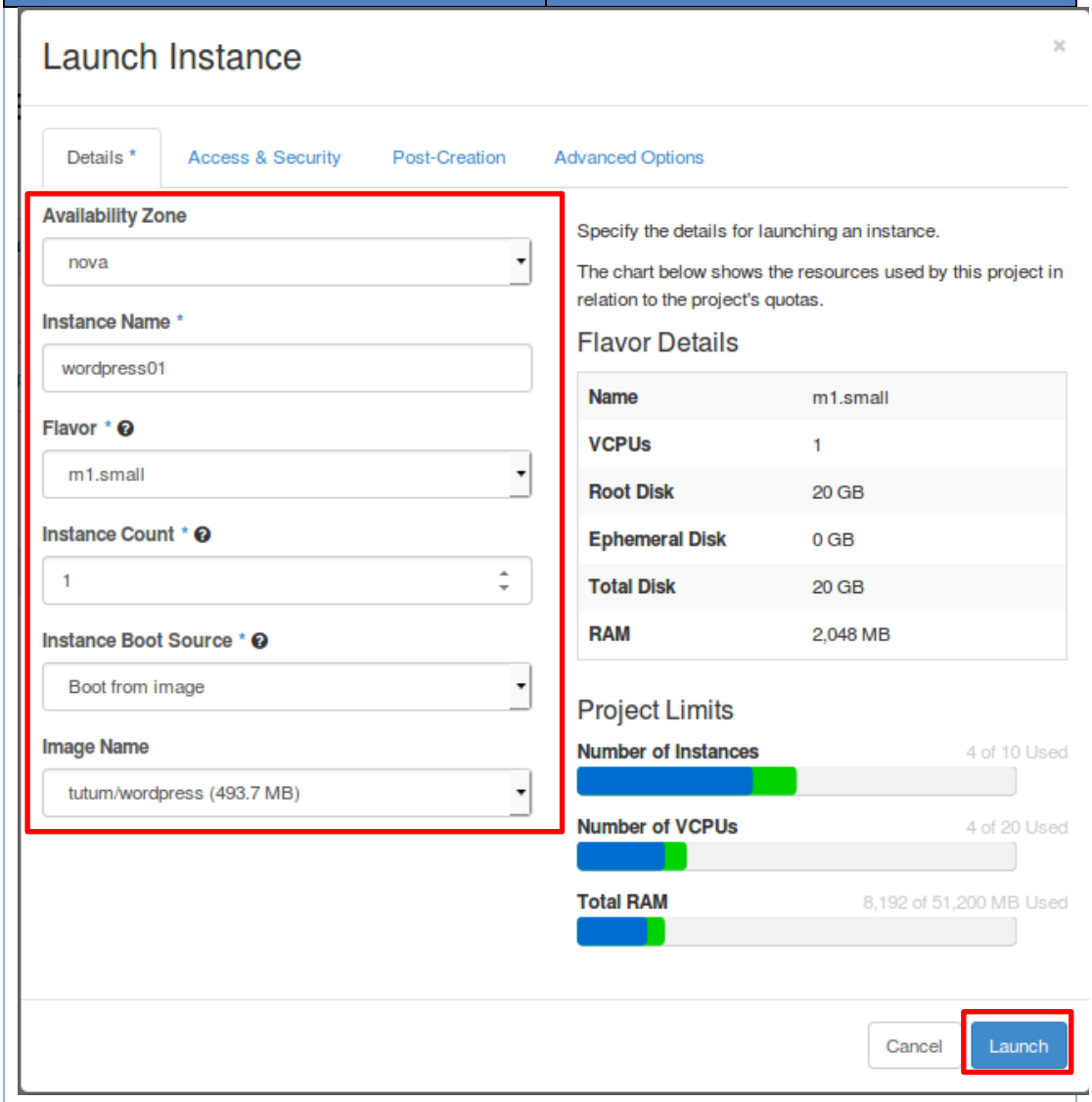

Figura 59. Detalles de instancia WordPress y botón Iniciar

2. Instancia WordPress iniciada.

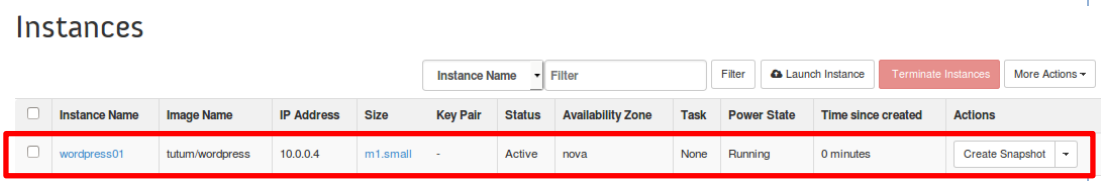

Figura 60. Instancia WordPress iniciada

3. Error general de instancia.

Continúa

Nombre de caso de prueba	N° 05. Inicio de instancia con imagen WordPress

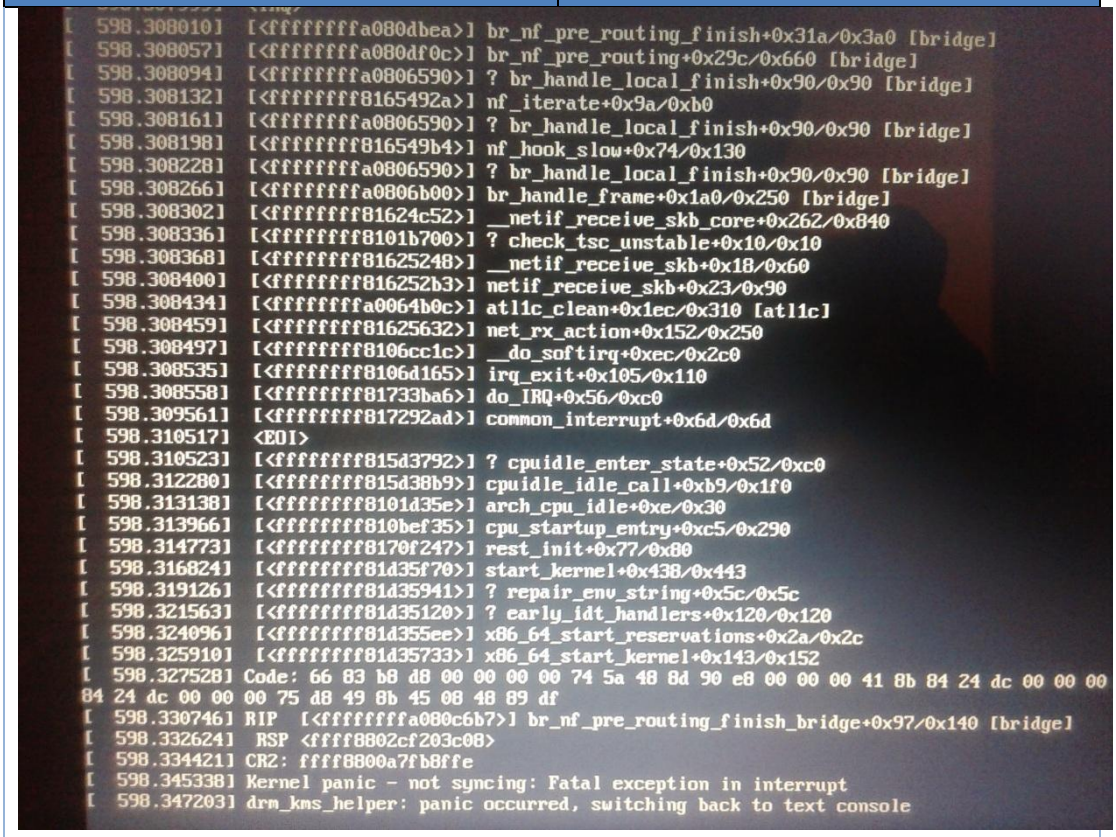

Figura 61. Error general de instancia y prototipo

Análisis de Resultados

El resultado del caso de prueba N°05 es Fallido.

- Se ha creado la instancia a partir de imagen WordPress con las características solicitadas.
- Se ha iniciado la instancia.
- Se presenta un error general del sistema. Del análisis realizado se detecta una sobrecarga a nivel de tarjeta de red que provoca un volcado general del sistema operativo. (Kernel Panic).

Tabla 24. Resultado de Caso de Prueba N°05. Inicio de instancia con imagen WordPress. (Segundo intento)

Nombre de caso de prueba	N° 05. Inicio de instancia con imagen WordPress (Segundo intento)
Objetivo del caso de prueba	Inicializar una instancia basada en la imagen WordPress.

Continúa

Nombre de caso de prueba	N° 05. Inicio de instancia con imagen WordPress (Segundo intento)

Procedimiento de prueba

- Ingresar a la interfaz gráfica de administración.
- Acceder a la opción Proyecto > Cómputo > Instancias
- Presionar el botón Lanzar instancia
- Elegir las opciones de instancia siguientes:
  - Zona de disponibilidad: nova
  - Nombre de instancia: wordpress01
  - Sabor: m1.medium
  - Conteo de instancias: 1
  - Fuente de inicio de instancia: Inicio desde imagen
  - Nombre de imagen: tutum/wordpress
- Presionar el botón Iniciar instancia.
- Probar acceso a instancia por medio de un navegador web. Apuntar hacia la dirección IP.
- Ejecutar el proceso de instalación de WordPress.
- Acceder a la interfaz de administración de WordPress.

Resultados esperados

1. Instancia WordPress creada.
2. Instancia WordPress iniciada.
3. Acceso al proceso de instalación de WordPress.
4. Acceso a la interfaz de administración de WordPress.

Resultados obtenidos

1. Instancia WordPress creada.

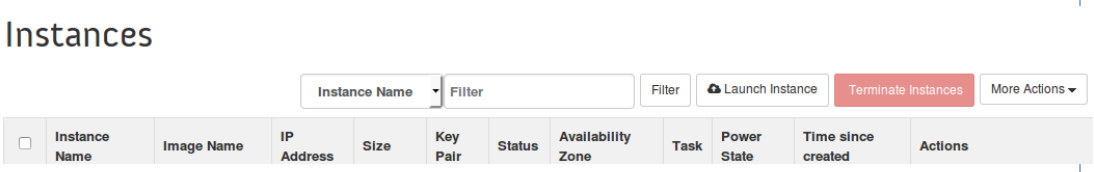

Figura 62. Botón Iniciar instancia

Continúa

Nombre de caso de prueba	N° 05. Inicio de instancia con imagen WordPress (Segundo intento)

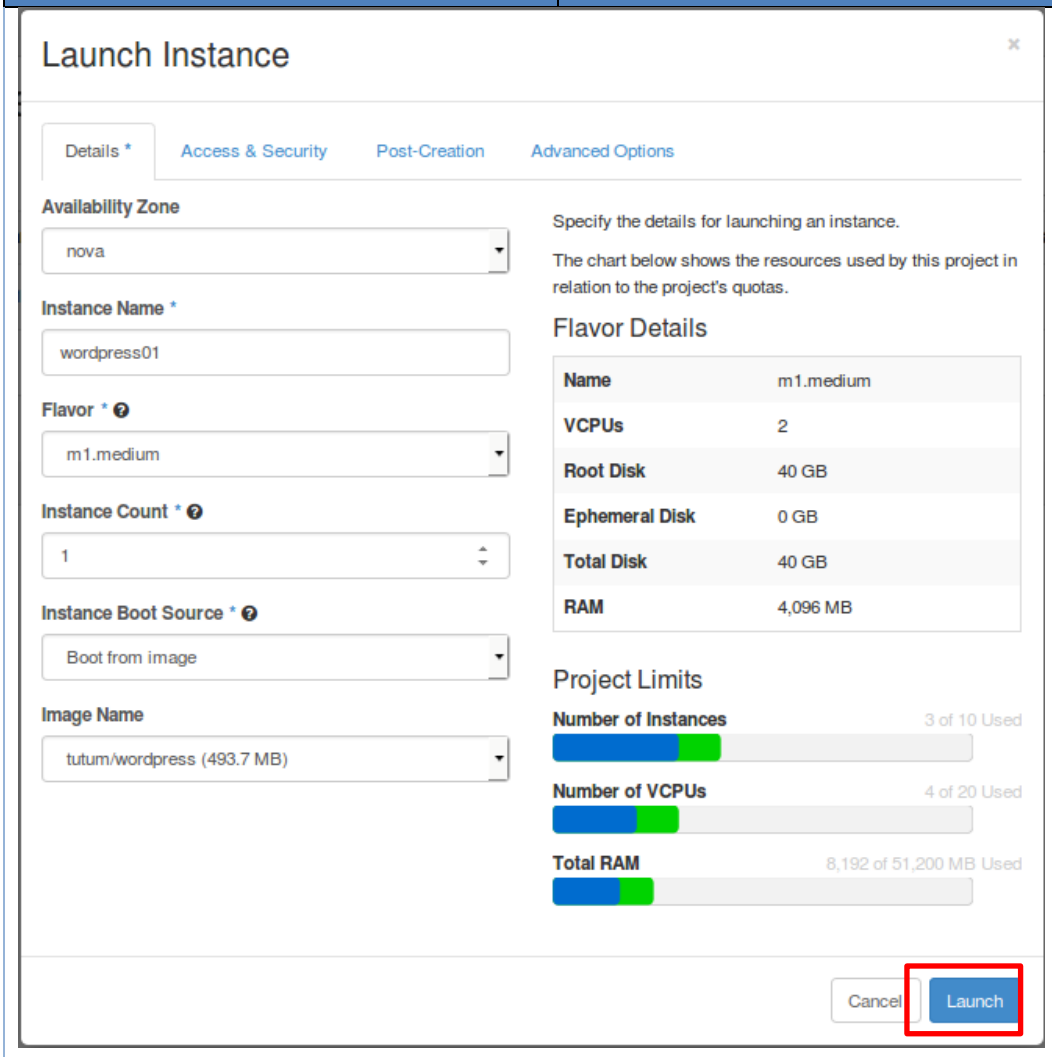

Figura 63. Detalles de instancia WordPress y botón Iniciar

2. Instancia WordPress iniciada.

Figura 64. Instancia WordPress iniciada.

3. Acceso al proceso de instalación de WordPress.

Continúa

Nombre de caso de prueba	N° 05. Inicio de instancia con imagen WordPress (Segundo intento)

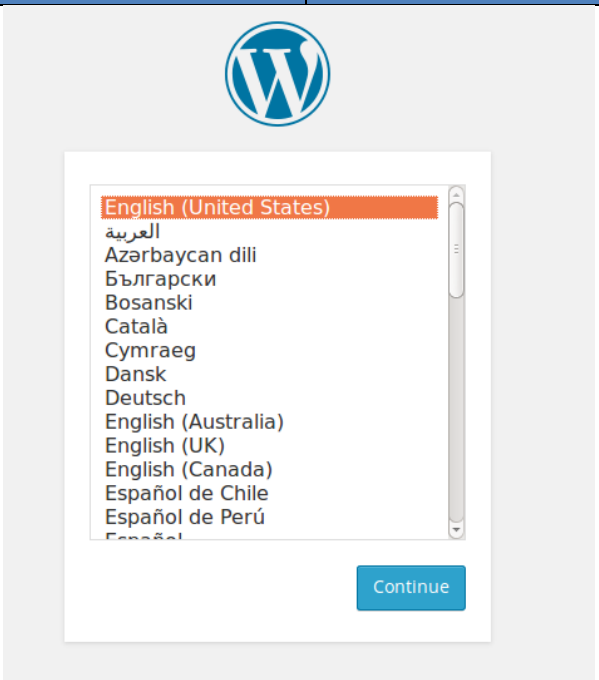

Figura 65. Selección de idioma para instalación

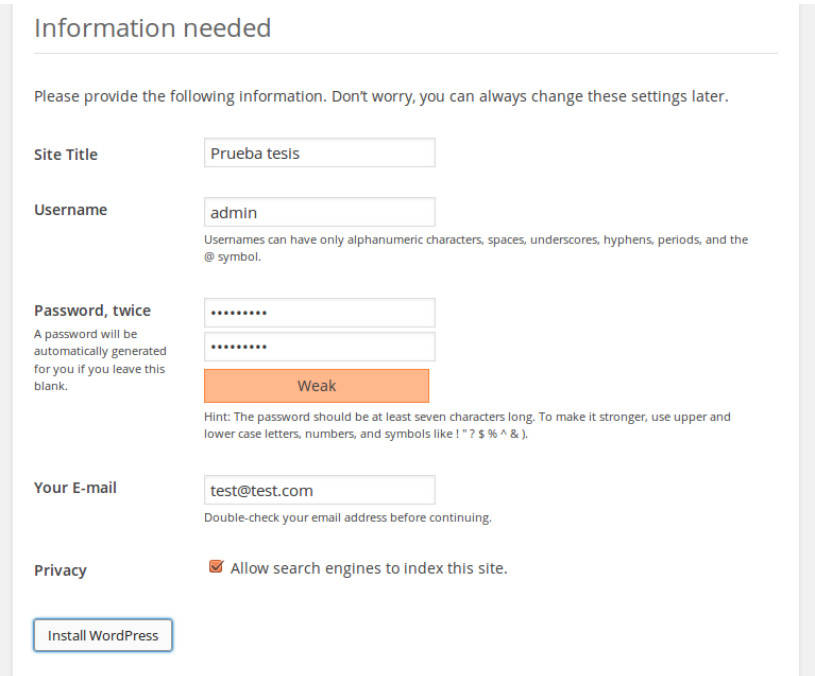

Figura 66. Parámetros de instalación y botón Instalar WordPress

Continúa

Nombre de caso de prueba	N° 05. Inicio de instancia con imagen WordPress (Segundo intento)

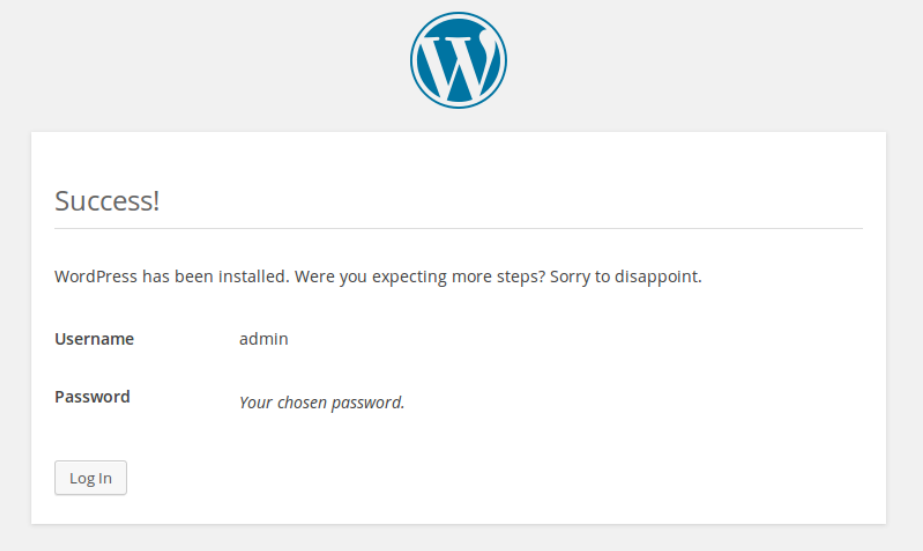

Figura 67. Resultado de instalación de WordPress

4. Acceso a la interfaz de administración de WordPress.

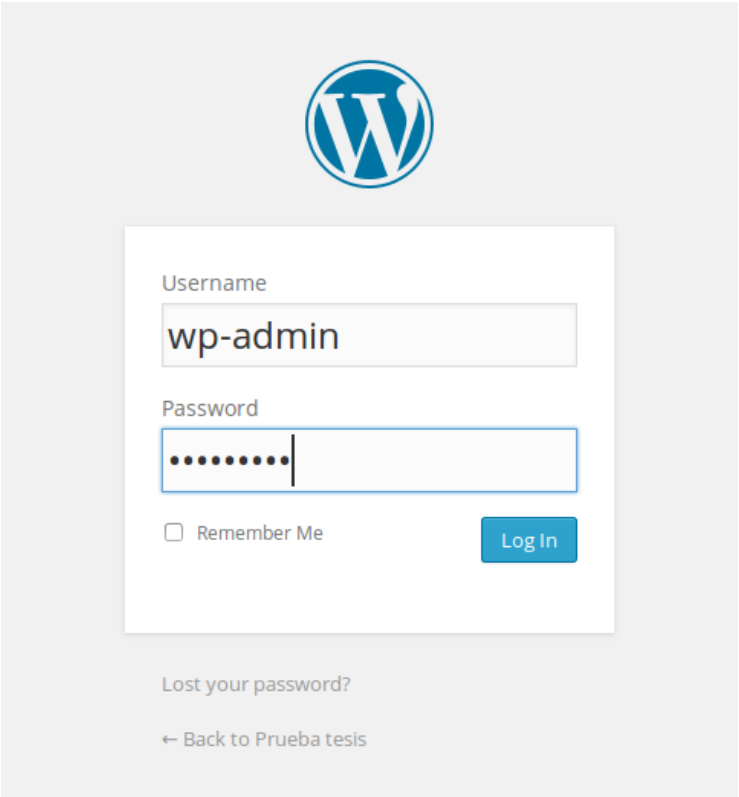

Figura 68. Pantalla de acceso a WordPress

Continúa

Nombre de caso de prueba	N° 05. Inicio de instancia con imagen WordPress (Segundo intento)

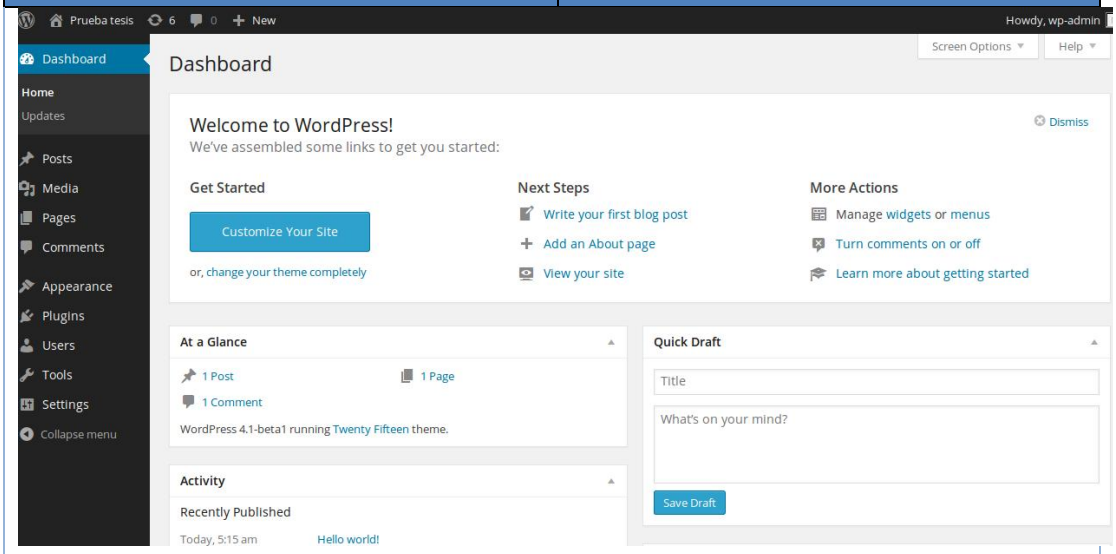

Figura 69. Interfaz de administración de WordPress

Análisis de Resultados

El resultado del segundo intento de caso de prueba N°05 es Satisfactorio.

- Se ha creado la instancia a partir de imagen WordPress con las características solicitadas de una instancia de tipo mediano, con mayores recursos de hardware.
- Se ha iniciado la instancia de WordPress.
- Se realiza el proceso de instalación de WordPress.
- Se ingresa a la interfaz de administración de la interfaz de WordPress creada.

Tabla 25. Resultado de Caso de Prueba N°06. Inicio de múltiples instancias de imagen LAMP.

Nombre de caso de prueba	N° 06. Inicio de múltiples instancias de imagen LAMP
Objetivo del caso de prueba	Crear varias instancias de LAMP a partir de la imagen original

Procedimiento de prueba

- Ingresar a la interfaz gráfica de administración.
- Acceder a la opción Proyecto > Cómputo > Instancias
- Presionar el botón Lanzar instancia
- Elegir las opciones de instancia siguientes:
  - Zona de disponibilidad: nova
  - Nombre de instancia: lamp02
  - Sabor: m1.small

Continúa

Nombre de caso de prueba	N° 06. Inicio de múltiples instancias de imagen LAMP

- Conteo de instancias: 5
- Fuente de inicio de instancia: Inicio desde imagen
- Nombre de imagen: tutum/lamp

- Presionar el botón Iniciar instancias.
- Probar acceso a instancias por medio de un navegador web. Apuntar hacia las direcciones IP asignadas.

Resultados esperados

1. Instancias LAMP creadas.
2. Instancias LAMP iniciadas.
3. Acceso a páginas “Hello World” cargadas dentro de las instancias creadas.

Resultados obtenidos

1. Se solicita la creación de instancias LAMP.

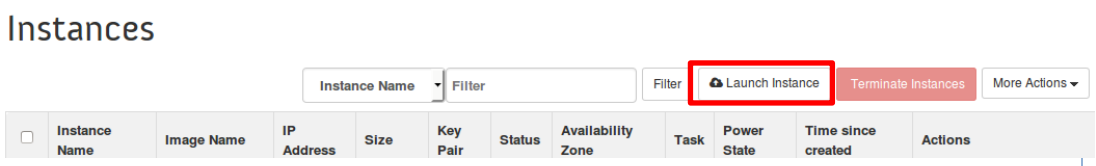

Figura 70. Botón Iniciar instancia

Continúa

Nombre de caso de prueba	N° 06. Inicio de múltiples instancias de imagen LAMP

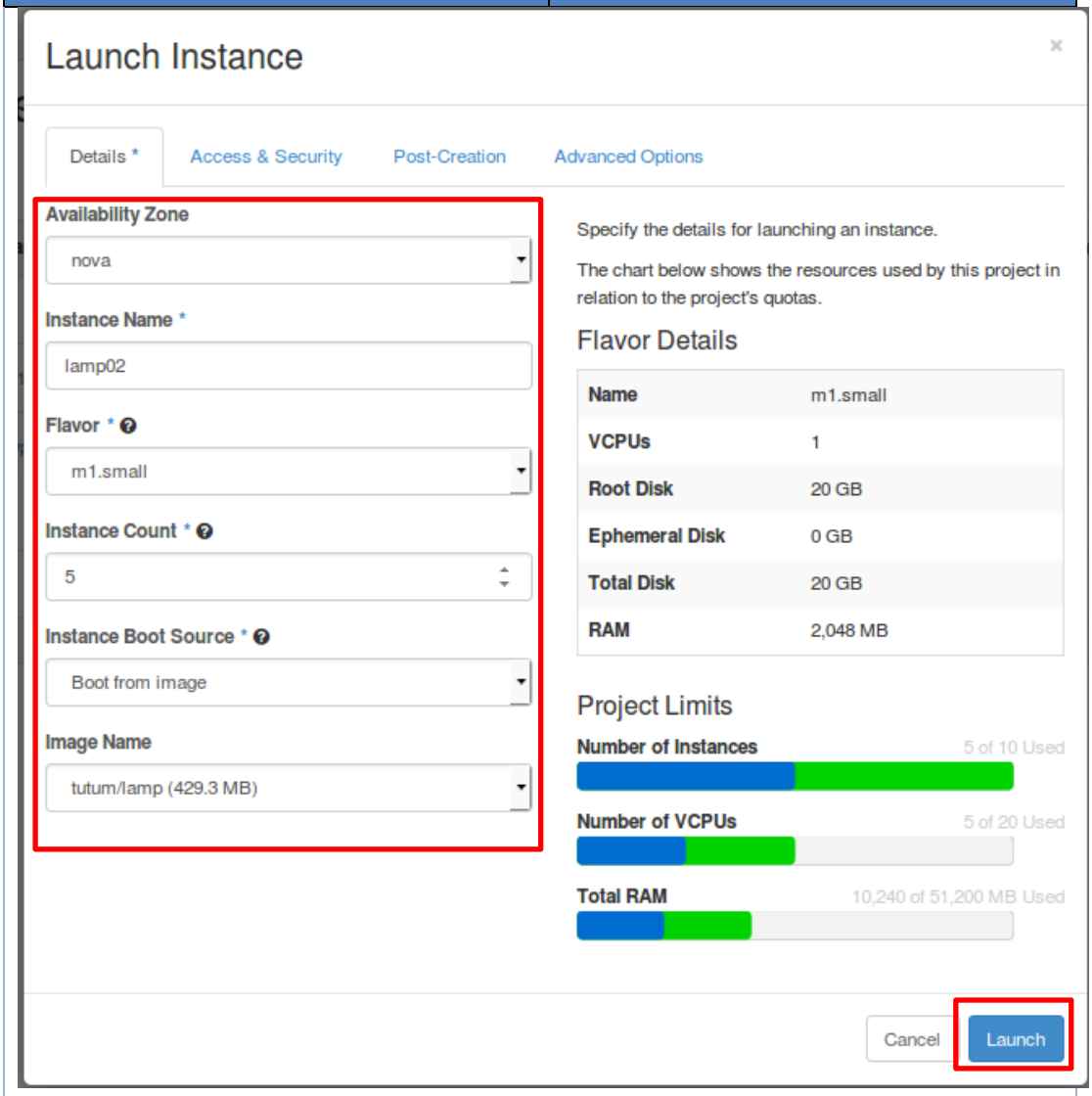

Figura 71. Detalles de instancias y botón Iniciar

2. Error de falta de recursos para la creación simultánea de las 5 instancias solicitadas.

Continúa

Nombre de caso de prueba	N° 06. Inicio de múltiples instancias de imagen LAMP

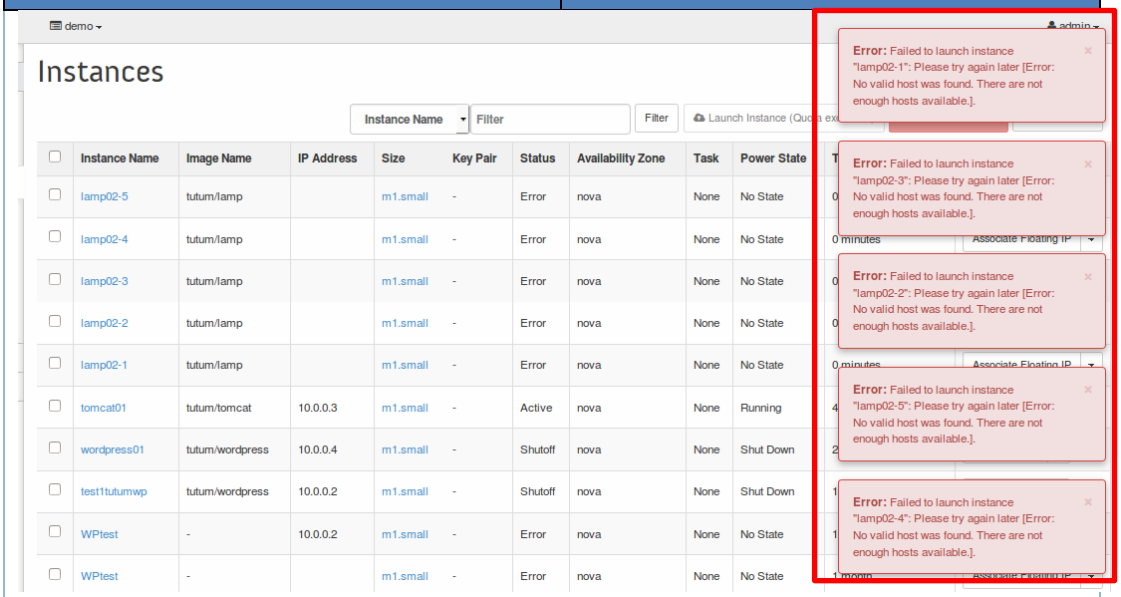

Figura 72. Error al crear instancias múltiples

Análisis de Resultados

El resultado del caso de prueba N°06 es Fallido.

- Se ha realizado la solicitud de creación de 5 instancias LAMP simultáneas.
- Se presentan mensajes de error que indican que se sobrepasa la capacidad de creación del prototipo.

Tabla 26. Resultado de Caso de Prueba N°06. Inicio de múltiples instancias de imagen LAMP (Segundo intento).

Nombre de caso de prueba	N° 06. Inicio de múltiples instancias de imagen LAMP (Segundo intento)
Objetivo del caso de prueba	Crear varias instancias de LAMP a partir de la imagen original

Procedimiento de prueba

- Ingresar a la interfaz gráfica de administración.
- Acceder a la opción Proyecto > Cómputo > Instancias
- Presionar el botón Lanzar instancia
- Elegir las opciones de instancia siguientes:
  - Zona de disponibilidad: nova
  - Nombre de instancia: lamp02
  - Sabor: m1.small
  - Conteo de instancias: 3

Continúa

Nombre de caso de prueba	N° 06. Inicio de múltiples instancias de imagen LAMP (Segundo intento)

- Fuente de inicio de instancia: Inicio desde imagen
- Nombre de imagen: tutum/lamp

- Presionar el botón Iniciar instancias.
- Probar acceso a instancias por medio de un navegador web. Apuntar hacia las direcciones IP asignadas.

Resultados esperados

1. Instancias LAMP creadas.
2. Instancias LAMP iniciadas.
3. Acceso a páginas “Hello World” cargadas dentro de las instancias creadas.

Resultados obtenidos

1. Instancias LAMP creadas.

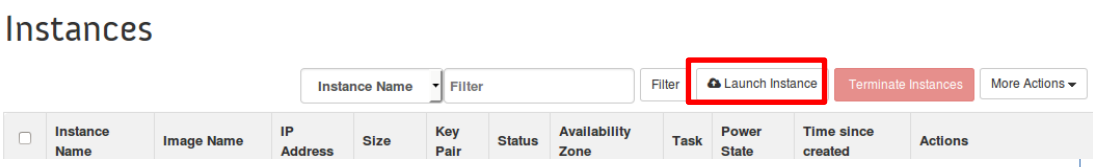

Figura 73. Botón Iniciar instancia

Continúa

Nombre de caso de prueba	N° 06. Inicio de múltiples instancias de imagen LAMP (Segundo intento)

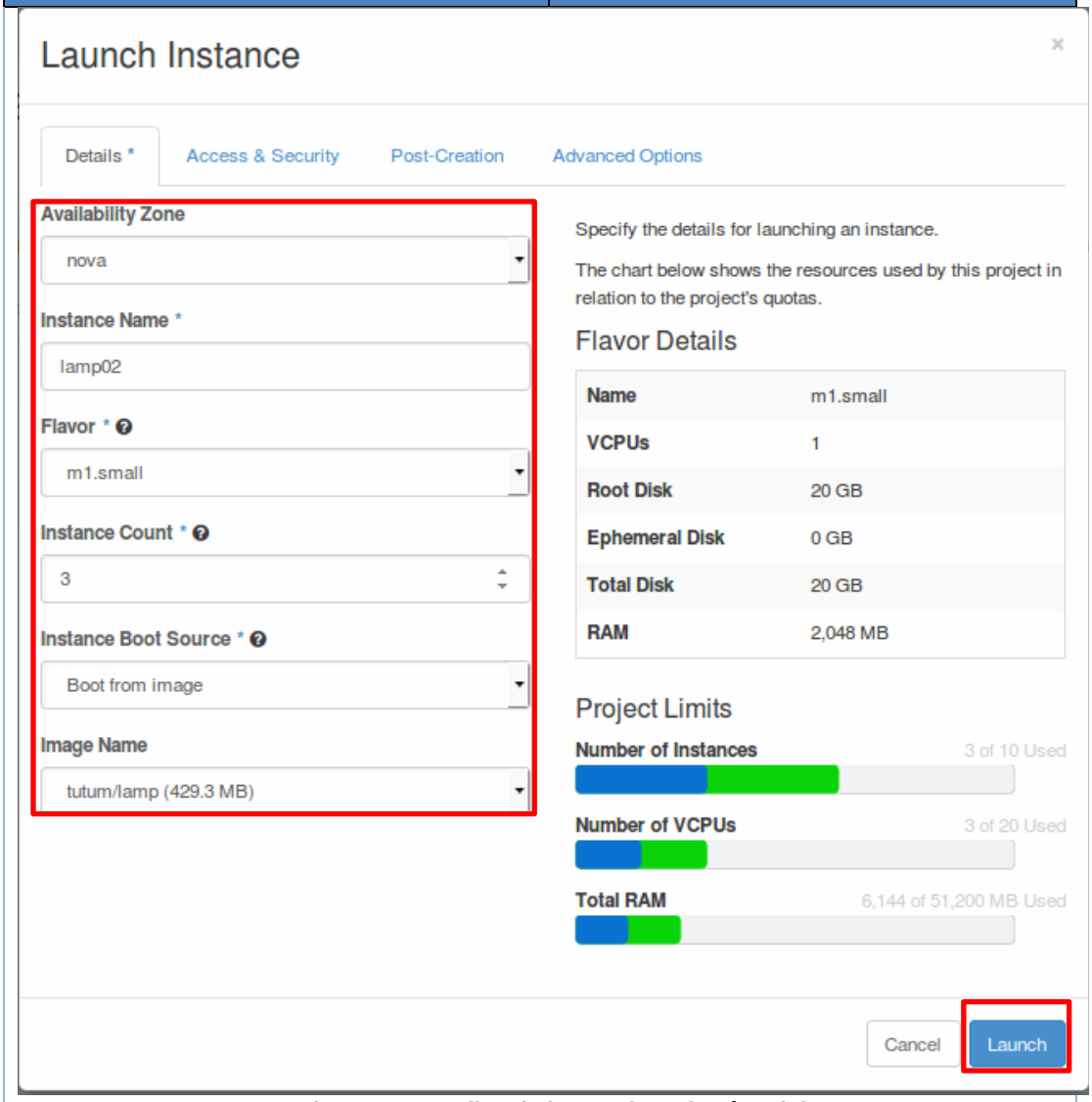

Figura 74. Detalles de instancias y botón Iniciar

2. Instancias LAMP iniciadas.

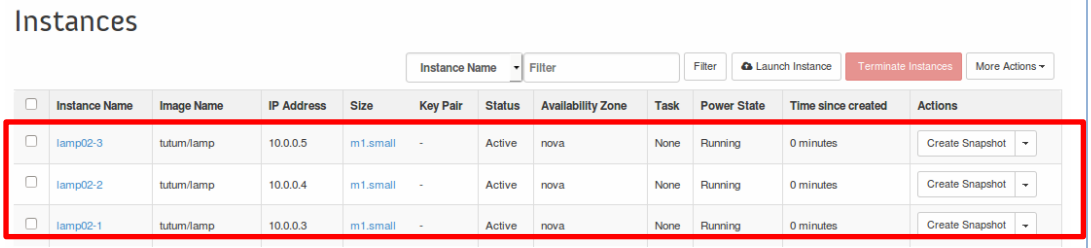

Figura 75. Instancias LAMP iniciadas simultáneamente

Continúa

Nombre de caso de prueba	N° 06. Inicio de múltiples instancias de imagen LAMP (Segundo intento)

```
stack@docker:~/devstack$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
fb3e369090cc        tutum/lamp:latest   "/run.sh"           2 minutes ago
Up 2 minutes                            nova-11bc6504-eaa0-433a-a58f-96449a035e4
0
b9fa93dede66        tutum/lamp:latest   "/run.sh"           2 minutes ago
Up 2 minutes                            nova-7ac7bde8-d731-496a-a840-7d46f10e155
1
8618abb9dced        tutum/lamp:latest   "/run.sh"           2 minutes ago
Up 2 minutes                            nova-6650e45a-3231-4992-9966-cc2e5e76b3d
d
```

Figura 76. Instancias iniciadas y sus contenedores Docker asociados

3. Acceso a páginas “Hello World” cargadas dentro de las instancias creadas.

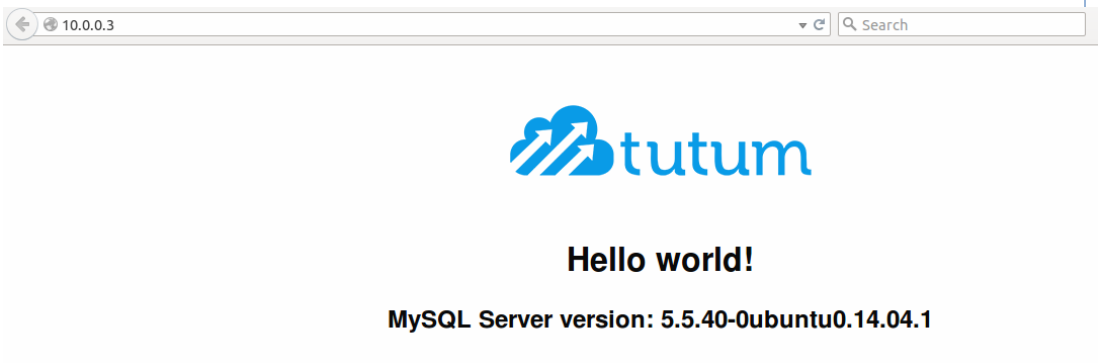

Figura 77. Página “Hello World” de instancia lamp02-1

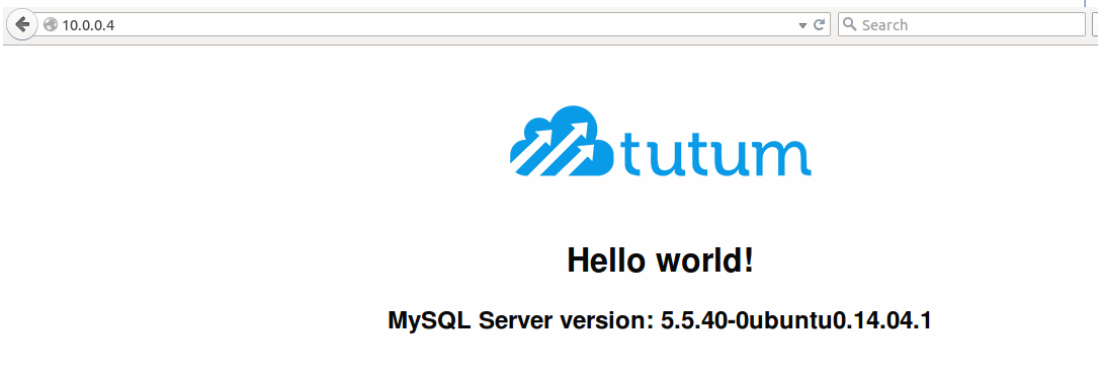

Figura 78. Página “Hello World” de instancia lamp02-2

Continúa

Nombre de caso de prueba	N° 06. Inicio de múltiples instancias de imagen LAMP (Segundo intento)

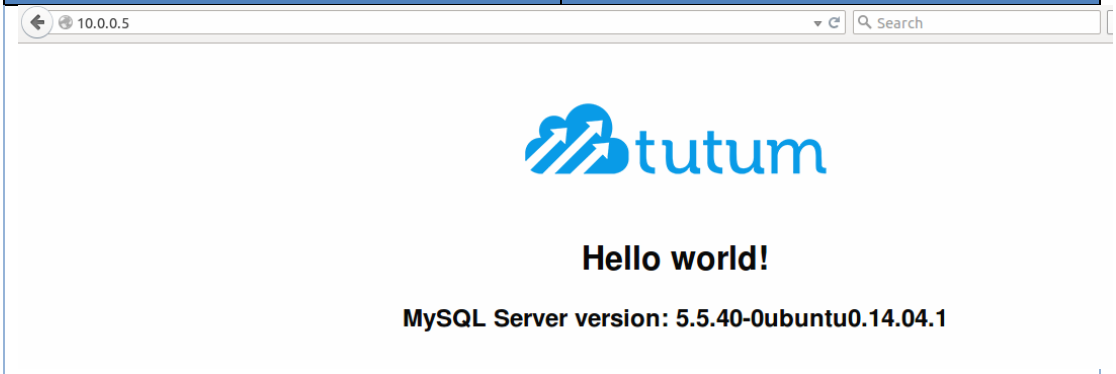

Figura 79. Página "Hello World" de instancia lamp02-3

Análisis de Resultados

El resultado del segundo intento de caso de prueba N°06 es Satisfactorio.

- Se han creado las instancias a partir de la imagen LAMP. En este segundo intento se solicita la creación de 3 instancias.
- La creación de 3 instancias simultáneas se produce sin contratiempos.
- Se puede acceder a la página "Hello World" de cada una de las instancias creadas.

Tabla 27. Resultado de Caso de Prueba N°07. Detener instancias creadas

Nombre de caso de prueba	N° 07. Detener instancias creadas
Objetivo del caso de prueba	Detener las instancias creadas y validar la liberación de los recursos utilizados por las mismas.

Procedimiento de prueba

- Ingresar a la interfaz gráfica de administración.
- Acceder a la opción Proyecto > Cómputo > Instancias
- Seleccionar las instancias que quieren ser eliminadas.
- Presionar el botón Terminar instancias.
- En el cuadro de confirmación, se presiona el botón Terminar instancias
- Acceder a la opción Proyecto > Cómputo > Visión General.
- Validar el estado de los recursos utilizados por el prototipo.

Resultados esperados

1. Instancias detenidas y eliminadas.
2. Recursos de prototipo liberados para ser reutilizados.

Resultados obtenidos

1. Instancias detenidas y eliminadas.

Continúa

Nombre de caso de prueba	N° 07. Detener instancias creadas

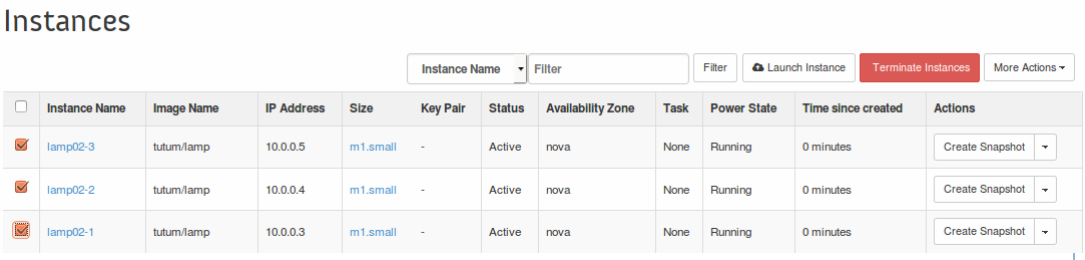

Figura 80. Instancias elegidas para eliminación y botón Terminar instancias

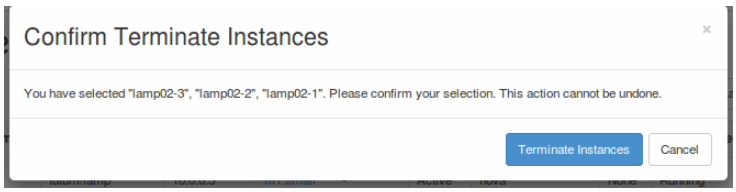

Figura 81. Confirmación de eliminación de instancias

2. Recursos de prototipo liberados para ser reutilizados.

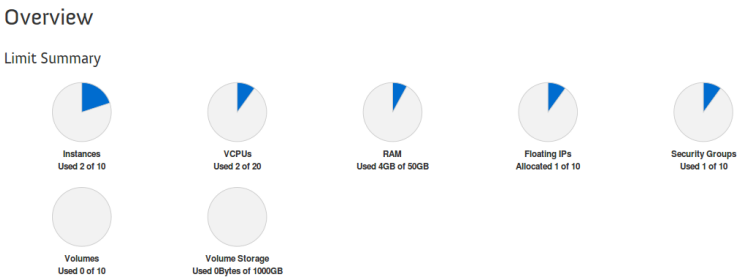

Figura 82. Estado de recursos luego de eliminación de instancias seleccionadas.

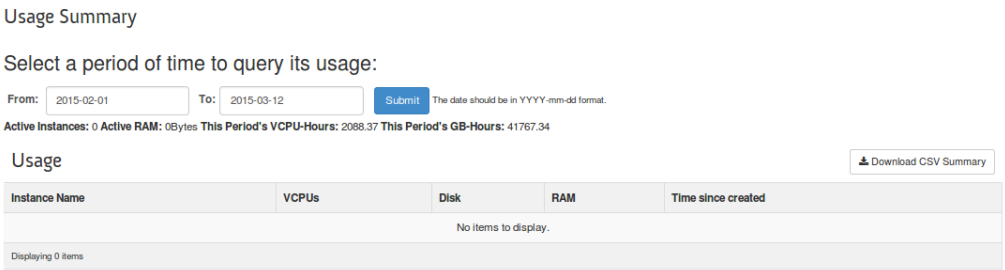

Figura 83. Resumen de uso de recursos

Análisis de Resultados

El resultado del caso de prueba N°07 es Satisfactorio.

- Las instancias seleccionadas como parte de la prueba son eliminadas bajo demanda.
- Los recursos que estaban siendo utilizados por las instancias han sido liberados y quedan listo para ser reasignados según se solicite.

CAPÍTULO 6. CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

- La creación del prototipo de Plataforma como Servicio basado en tecnologías abiertas ha permitido experimentar con los beneficios de estas tecnologías y detectar los retos que implica su adopción.

- Las soluciones de nube híbrida permiten a las empresas contar con flexibilidad y adaptabilidad en el manejo de sus cargas de trabajo; y al mismo tiempo cumplir con los requerimientos de gestión de datos críticos, cumplimiento de estándares y seguridad de la información.

- Las soluciones abiertas basadas en estándares y arquitecturas de referencia permiten a las organizaciones explotar los beneficios de la computación en la nube sin quedar atadas a proveedores o soluciones específicas.

- El uso de una metodología de selección basada en estándares permitió una elección técnica de las soluciones utilizadas en el proyecto, gracias a estos lineamientos es posible garantizar que los componentes de una solución se apegan a los requerimientos.

- El prototipo de Plataforma como Servicio ha podido ser implantado gracias a las capacidades de integración de las tecnologías seleccionadas y a las características propias de

OpenStack y Docker para gestión de recursos y consumo de imágenes de sistemas operativos y aplicativos bajo demanda.

- Las pruebas de prototipo han permitido detectar que las características de hardware utilizadas en el presente proyecto no soportan una alta carga de creación de instancias, por lo que en su estado actual, el prototipo no podría ser usado en ambientes de producción.

6.2. RECOMENDACIONES

- Para trabajos futuros se recomienda crear proyectos que analicen la configuración de almacenamiento y red, ya que estos campos ofrecen alternativas variadas que pueden ser investigadas y aplicadas.

- Se recomienda el uso de hardware con características superiores a las usadas en este proyecto, con miras a implantar este tipo de solución en ambiente de producción.

- Participar en las comunidades de OpenStack y Docker para apoyar a los proyectos y conocer más a fondo las funcionalidades que incluyen y que se encuentran dentro de la planificación.

- Para trabajos futuros se recomienda experimentar con la integración de OpenStack y Docker a través del proyecto Heat para validar sus funcionalidades.

- Se recomienda realizar prototipos similares haciendo uso de las alternativas de PaaS, Cloud Foundry y OpenShift, las cuales cuentan con respaldo de la industria apoyan al proyecto Docker, lo cual abre interesantes oportunidades dentro de este campo.

- Se debería implantar este tipo de soluciones en ambientes de desarrollo y pruebas de empresas para validar su utilización con aplicaciones reales y con cargas de trabajo de producción.

BIBLIOGRAFÍA

- Apache Software Foundation. (2014). *CloudStack*. Obtenido de http://cloudstack.apache.org/
- Docker, Inc. (2014). *Docker*. Obtenido de https://www.docker.com/
- Eucalyptus. (2014). *Eucalyptus*. Obtenido de https://www.eucalyptus.com/docs/eucalyptus/4.0.1/index.html#shared/index.html
- Hausman, K., Cook, S., & Sampaio, T. (2013). *Cloud Essentials.* Indianapolis, Indiana: John Wiley & Sons, Inc.
- IBM Corporation. (2013). IBM Cloud Computing Reference Architecture. En P. Iannucci, & M. Gupta, *IBM SmartCloud: Building a Cloud Enabled Data Center* (págs. 8-15). IBM Corporation.
- ISO. (2014). *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE.* Obtenido de ISO Online Browsing Platform: https://www.iso.org/obp/ui/#iso:std:iso-iec:25000:ed-2:v1:en
- National Institute of Standards and Technology. (2011). NIST Cloud Computing Reference Architecture. En J. T. Fang Liu. Gaithersburg: U.S. Department of Commerce.
- OpenNebula. (2014). *OpenNebula*. Obtenido de http://opennebula.org/about/project/
- Openshift. (2014). *OpenShift, Enterprise Features and Benefits*. Obtenido de https://www.openshift.com/products/enterprise/features-and-benefits
- OpenStack. (Mayo de 2014). *OpenStack*. Obtenido de http://www.openstack.org/
- Pivotal Software Inc. (2014). *Cloud Foundry Community*. Obtenido de http://cloudfoundry.org

- Red Hat, Inc. (2014). *OpenShift*. Obtenido de https://www.openshift.com
- The Apache Software Foundation. (2014). *Apache Stratos*. Obtenido de http://stratos.apache.org/
- The Open Group. (2011). Cloud Computing Explained. En S. Ghosh, & G. Hughes, *Cloud Computing Explained* (págs. 49-52). San Francisco, CA: The Open Group.
- Tsuru Community. (2014). *Tsuru*. Obtenido de http://www.tsuru.io/
- VMware, Inc. (2014). *VMware*. Obtenido de http://www.vmware.com/
- Williams, A. (18 de 08 de 2014). *The New Stack.* Obtenido de http://thenewstack.io/the-new-stack-and-linux-foundation-survey-openstack-and-docker-are-the-most-popular-open-source-projects/