Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

PROPUESTA PARA ACTUALIZACIÓN CURRICULAR EN EL ÁREA DE SOFTWARE INCORPORANDO TALLERES DE DOCKER Y KUBERNETES

Elmer Orlando Real Ixcayau
Asesorado por el Ing. Jhonatan Wilfredo Pú Morales

Guatemala, marzo de 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

PROPUESTA PARA ACTUALIZACIÓN CURRICULAR EN EL ÁREA DE SOFTWARE INCORPORANDO TALLERES DE DOCKER Y KUBERNETES

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

ELMER ORLANDO REAL IXCAYAU
ASESORADO POR EL ING. JHONATAN WILFREDO PÚ MORALES

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, MARZO DE 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Christian Moisés de la Cruz Leal
VOCAL V	Br. Kevin Armando Cruz Lorente
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANA	Inga. Aurelia Anabela Cordova Estrada
EXAMINADOR	Ing. César Augusto Fernádnez Cácreres
EXAMINADOR	Ing. Herman Igor Véliz Linares
EXAMINADOR	Ing. César Rolando Batz Saquimux
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

PROPUESTA PARA ACTUALIZACIÓN CURRICULAR EN EL ÁREA DE SOFTWARE INCORPORANDO TALLERES DE DOCKER Y KUBERNETES

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha febrero de 2020.

Elmer Orlando Real Ixcayau

Guatemala, 25 de junio de 2020

Ingeniero
Carlos Alfredo Azurdia
Coordinador de Privados y Revisor de Trabajos de Graduación
Escuela de Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Ingeniero Azurdia:

Tengo el agrado de dirigirme a usted para informarle que he revisado el trabajo de graduación **"PROPUESTA PARA ACTUALIZACION CURRICULAR EN EL ÁREA DE SOFTWARE INCORPORANDO TALLERES DE DOCKER Y KUBERNETES."**, realizado por el estudiante universitario **ELMER ORLANDO REAL IXCAYAU** con carné **201503936**, quien contó con la asesoría del suscrito. Considero que el trabajo realizado por el estudiante, cumple con los objetivos bajo los cuales fue planteado y cumple satisfactoriamente cada una de las actividades planificadas, por lo que procedo a aprobarlo.

Agradeciendo la atención dada a la presente, me suscribo

Atentamente,

Ing. Jhonatan Wilfredo Pú Morales
Asesor
Colegiado 15,628

Jhonatan Wilfredo Pú Morales
Ingeniero en Ciencias y Sistemas
Colegiado 15,628

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 17 de agosto de 2020

Ingeniero
Carlos Gustavo Alonzo
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Alonzo:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **ELMER ORLANDO REAL IXCAYAU** con carné **201503936 y** CUI **3002 43030 0101** titulado **"PROPUESTA PARA ACTUALIZACIÓN CURRICULAR EN EL ÁREA DE SOFTWARE INCORPORANDO TALLERES DE DOCKER Y KUBERNETES"** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo aprobado.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"PROPUESTA PARA ACTUALIZACIÓN CURRICULAR EN EL ÁREA DE SOFTWARE INCORPORANDO TALLERES DE DOCKER Y KUBERNETES",*** *realizado por el estudiante, ELMER ORLANDO REAL IXCAYAU aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

Msc. Carlos Gustavo Alonzo
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 08 de marzo de 2021

Decanato
Facultad de Ingeniería
24189101- 24189102

DTG. 086.2021.

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **PROPUESTA PARA ACTUALIZACIÓN CURRICULAR EN EL ÁREA DE SOFTWARE INCORPORANDO TALLERES DE DOCKER Y KUBERNETES,** presentado por el estudiante universitario: **Elmer Orlando Real Ixcayau,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANA
FACULTAD DE INGENIERÍA

Inga. Anabela Cordova Estrada
Decana

Guatemala, marzo de 2021.

AACE/asga

ACTO QUE DEDICO A:

Dios	Por acompañarme durante todo el camino, por ser mi fortaleza en los momentos difíciles, por la vida y darme la oportunidad de alcanzar esta meta profesional.
Mis padres	Carmen Ixcayau y Guillermo Real. Por su amor inmensurable, paciencia, esfuerzo y los valores que me han inculcado.
Mis tíos	Alexander, Ángel, Eduardo, Mary, Matilde. Por todo su amor, comprensión y paciencia. Son parte fundamental en mi formación como persona y como profesional. Sin su apoyo, enseñanzas y correcciones nada de esto sería posible.
Luis Mata	Por su apoyo y confianza, no he conocido otra persona que haga el bien sin mirar a quién.
Mis hermanos	Christian, Diana, Melvin Real. Por su apoyo y motivación.
Mi abuela	Genoveva Ixcayau (q. e. p. d.). Por su gran amor y paciencia. Fue, es y siempre será mi motor para seguir esforzándome.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por darme la oportunidad de realizar este sueño. Es un honor poder decir que es mi *Alma Máter* donde me he forjado en educación superior.
Familia Mata Ixcayau	Por abrirme las puertas de su hogar, su amor y apoyo incondicional.
Edgar Calderón	Por ser una importante influencia en mi carrera, y por su apoyo y amistad.
Ing. Jhonatan Pú	Por su tiempo y ayuda en la asesoría, revisión y corrección de este trabajo.
Mis amigos de la Facultad	Por su amistad y apoyo.
Facultad de Ingeniería	Por brindarme los conocimientos que me permitieron desarrollarme como profesional.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
,	Coma
“	Comillas dobles
/	Diagonal
:	Dos puntos
&	Et
-	Guion
)	Paréntesis derecho
(	Paréntesis izquierdo
%	Porcentaje
.	Punto
;	Punto y coma

GLOSARIO

Algoritmo Conjunto de instrucciones utilizadas para llevar a cabo una tarea finita que produce un resultado predecible y replicable.

Alta disponibilidad Capacidad de funcionamiento continuo de un sistema debido a su diseño interno y de infraestructura.

API Sigla del inglés Application Programming Interface, se refiere a un puerto de información que permite servir la información de una aplicación a algún *frontend* o servicio que solicite información.

CD En el ámbito DevOps esta sigla es utilizada para representar los términos ingleses *Continuous Delivery* y *Continuous Deployment*. *Continuous Delivery* se refiere al proceso continuo de entrega de nuevos artefactos que hayan pasado las pruebas tanto de funcionamiento como de compatibilidad. *Continuous Deployment* se refiere al proceso continuo del despliegue de cada artefacto nuevo al ambiente de producción.

CI Sigla de *Continuous Integration,* que hace referencia a la constante integración de nuevos cambios al código existente para verificar la compatibilidad entre

	sí y poder tomar acciones correctivas de una manera más oportuna.
Cloud Computing	Paradigma de computación que ofrece servicios computacionales de red, almacenamiento y cómputo bajo demanda.
Contenedor	Empaquetamiento de software en unidades estándares que se ejecutan haciendo uso de virtualización a nivel de sistema operativo.
CPU	Unidad central de procesamiento.
Docker	Herramienta para la definición, creación y manejo de contenedores.
DevOps	Sigla producto de la unión de las palabras en inglés *developer* y *operations*. Se refiere a la combinación de tareas de desarrollo de un sistema y tareas operacionales con el objetivo de integrar, entregar y desplegar nuevas versiones de un sistema en ciclos cortos apoyándose fuertemente en el uso de herramientas de automatización.
Elasticidad	Capacidad de un sistema para escalar con el objetivo de satisfacer un pico de carga de trabajo y una vez la carga de trabajo disminuya los recursos para reducir costos.

Escalabilidad	Habilidad de crecimiento automatizado de la infraestructura de un sistema ante picos de demanda. Se puede incrementar la capacidad de la infraestructura actual (escalamiento vertical) o la cantidad de recursos computacionales en funcionamiento para distribuir entre ellas la carga de trabajo (escalamiento horizontal).
Infraestructura	Recursos computacionales, de red y almacenamiento requeridos para el correcto funcionamiento de un sistema informático.
Kubernetes	Herramienta para la orquestación de contenedores.
MAN	Red de área metropolitana.
Microservicio	Estilo de arquitectura de software que se basa en el desacoplamiento de lógica del negocio en unidades funcionales e independientes (idealmente, aunque puede existir la comunicación entre sí), por ello se les denomina individualmente como microservicio.
NIST	National Institute of Standards and Technology.
Nube	Hace referencia a algún modelo de despliegue paradigma de computación Cloud Computing (pública, privada o híbrida).

Sprint	Periodo de tiempo en que se miden los ciclos de desarrollo en la metodología ágil Scrum.
WAN	Red de área amplia.

RESUMEN

Cloud Computing es la unión de muchos conceptos que se han desarrollado a lo largo de la historia de la computación. Permite configurar y consumir recursos computacionales a través de Internet con un modelo de pago por consumo como si fuese un servicio, lo cual permite a sus usuarios invertir más en su negocio y olvidarse del mantenimiento e inversión que aprovisionarse de una infraestructura representa.

DevOps y Cloud Computing son dos conceptos totalmente diferentes, pero no excluyentes, en realidad ambos se complementan. DevOps se encarga de automatizar los procesos para mejorarlos y generar valor en menor tiempo al aplicar su lema de CI/CD, mientras que Cloud Computing se refiere a tecnología y recursos computacionales como servicios. Al emplearlos en conjunto se pueden eliminar los errores humanos, agilizar el proceso de desarrollo y establecer repetibilidad[1].

“A medida que las empresas adopten la nube como la plataforma de infraestructura principal, impulsará el auge del empleo de contenedores para manejar la diversidad y variedad de dependencias entre las aplicaciones empresariales de las empresas.”[2] Este aumento de contenedores requerirá del uso de herramientas con altos niveles de abstracción que permitan definir y manejar los contenedores de una manera más sencilla. El aumento de

[1] VYAS, Jaymin. *DevOps and Cloud: A symbiotic relationship.* https://devops.com/devops-and-cloud-a-symbiotic-relationship.

[2] RAYAPATI, Viyaj. *The 2017 Clouds Trends – From DevOps to NoOps. https://articles.microservices.com/the-2017-cloud-trends-from-devops-to-noops-1d12fa85d433.*

empresas migrando a Cloud Computing y adoptando la cultura DevOps abre nuevo mercado que en la actualidad no se ha podido satisfacer.

OBJETIVOS

General

Evidenciar la necesidad de la integración de talleres de herramientas DevOps a algunos cursos del área de software del pensum de ingeniería en ciencias y sistemas de la Universidad de San Carlos de Guatemala, para aumentar la competitividad de los estudiantes, así como elaborar talleres sobre tecnologías de empaquetamiento de software como Docker y la orquestación de estos utilizando Kubernetes.

Específicos

1. Facilitar talleres de Docker a los estudiantes de Análisis y Diseño 2.

2. Facilitar talleres de Kubernetes a los estudiantes de Software Avanzado.

3. Exponer la tendencia actual y futura de la industria respecto al fenómeno Cloud Computing y la orquestación de contenedores.

4. Exponer la utilidad de los contenedores y su orquestación como herramientas DevOps y sus beneficios en Cloud Computing.

5. Proponer la integración de otros talleres sobre herramientas DevOps en los cursos de Análisis 2 y Software Avanzado.

INTRODUCCIÓN

Cloud Computing es un paradigma de computación que ofrece recursos computacionales bajo demanda y en un modelo de pago sobre uso, lo cual permite la optimización de costos para las empresas y les permite acceder a grandes recursos computacionales sin necesidad de realizar alguna inversión inicial.

Parte de la innovación de este paradigma de computación es el modelo de facturación por consumo como si fuesen un servicio, tal como el agua potable o la electricidad, lo cual permite a sus usuarios invertir más en lo que mejor hacen, es decir, su negocio, y olvidarse tanto del mantenimiento como de la inversión que aprovisionarse de una infraestructura representa.

Además, Cloud les permite a las empresas una rápida salida de sus productos al mercado y el acceso a altas capacidades de cómputo en cuestión de segundos. Lo que ha desembocado en una alta migración de aplicaciones a Cloud. Sin embargo, las aplicaciones que migran a Cloud deben sufrir una refactorización para dejar de ser aplicaciones tradicionales y convertirse en aplicaciones nativas de la nube en orden de explotar los beneficios de Cloud Computing.

Las aplicaciones nativas en la nube tienen como base fundamental las arquitecturas orientadas a microservicios, los contenedores y las APIs. Debido en parte a que las arquitecturas orientadas a microservicios permiten agilizar el proceso de desarrollo al dividir de manera granular aplicaciones complejas y aislar entre sí a cada uno de los componentes del sistema, además que son

susceptibles a ser contenerizadas y a la vez automatizadas. De manera que sus microservicios se pueden desarrollar y mantener sin afectar a otros.

DevOps es una cultura de automatización que se apoya principalmente en herramientas para combinar los procesos de desarrollo y operacionales, en orden de lograr una integración, entrega y despliegue de las nuevas versiones de un sistema de manera continua. Se integra perfectamente en el desarrollo de aplicaciones nativas de la nube porque explota el dinamismo ofrecido por Cloud Computing.

Aunque los contenedores proveen entornos aislados del mundo exterior en arquitecturas complejas con un alto número de contenedores, producto de la cantidad de microservicios o la replicación exigida por estos, es necesario el uso de herramientas para la orquestación de contenedores, tal como lo es Kubernetes, para definir, manejar, controlar y comunicar todos los contenedores de una manera más sencilla. Además, Kubernetes implementa mecanismos para el balanceo de cargas de trabajo para obtener un rendimiento óptimo, la replicación de contenedores para la escalabilidad, y monitoreo de salud de microservicios para asegurar alta disponibilidad de manera autónoma sin la necesidad de interacción humana.

Alrededor del mundo existe una falta de personas capacitadas en el paradigma Cloud y en las distintas herramientas DevOps empleadas en el proceso de automatización, ya que aprovechan al máximo el dinamismo de Cloud. Cloud continúa creciendo, por lo tanto, es una necesidad latente.

Ante esta necesidad de personas capacitadas se propone la integración de talleres de diferentes herramientas DevOps que guarden relación con los temas impartidos en los cursos de Análisis y Diseño 2 y Software Avanzado.

1. BASES

Cloud Computing es considerada como la quinta generación de la computación[3]. El término es relativamente nuevo, sin embargo, el concepto como tal no lo es, sino que es el producto de la unión de múltiples conceptos que han ido evolucionando junto a la computación misma a través del tiempo.

1.1. Evolución

Contar es por naturaleza una necesidad para los seres humanos, es algo que realizan en varias actividades diarias relacionadas a locación, distribución, posesión, posición, dimensión, inventario, almacenamiento, entre otras. Sin embargo, errar es inherente al comportamiento humano, por lo que desde siempre se ha buscado desarrollar máquinas y dispositivos que sustituyan al ser humano en las actividades para minimizar los errores y facilitar la realización de estas.

Los inicios de la computación se remontan hasta antes de la creación de la electricidad misma con la creación del ábaco, para poder contar grandes cantidades manualmente. El ábaco permitió que el conteo de grandes cantidades no fuera más un inconveniente para las personas, sin embargo, conforme la civilización evolucionó sus necesidades también lo hicieron.

El desarrollo de la matemática originó un nuevo desafío: realizar cálculos numéricos extensos y complejos; realizarlos manualmente era complicado, sin mencionar que también impreciso, además representaba un alto consumo de

[3] KILARI, Nagaraju. *Cloud Computing - An Overview & Evolution.* p. 149

tiempo. Estas razones motivaron al ser humano a diseñar máquinas capaces de realizar cálculos extensos y complejos de manera automática con la mínima intervención humana para obtener resultados de una manera más rápida, eficiente y precisa.

Con la invención de la máquina de cálculo se logró cumplir el desafío de resolver cálculos extensos y complejos en un tiempo relativamente corto y dar resultados precisos, pero luego de haber experimentado las ventajas del empleo de máquinas en tareas intelectuales, el ser humano se aventuró en un nuevo desafío: manejar grandes cantidades de información de múltiple índole de manera automatizada empleando máquinas.

El brillante científico británico Alan Turing teorizó sobre una supermáquina capaz de resolver muchas tareas empleando múltiples máquinas de Turing. Turing planteó que cada máquina se debía encargar de interpretar un algoritmo escrito con un conjunto de instrucciones estandarizadas para llevar a cabo una tarea específica de manera que las múltiples máquinas juntas conformaran una sola supermáquina multipropósito. Esta teoría es en la que se basa la computadora y fue posible materializarla gracias a inventos como el álgebra de Boole, desarrollado en 1864 por George Boole, y los tubos electrónicos desarrollados en 1900 por Thomas Alva Edison.

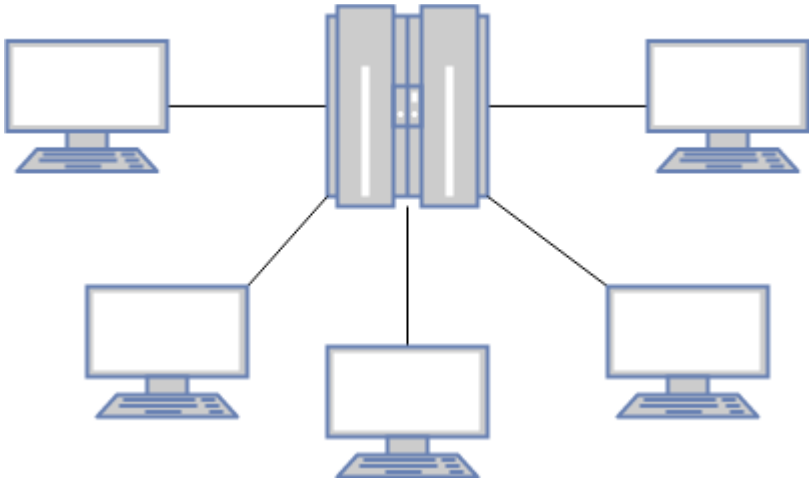
La creación de la computadora marcó el inicio de la computación, ciencia encargada del desarrollo de tecnologías para el procesamiento, almacenamiento y análisis de grandes cantidades de información empleando computadoras. La computación como la conocemos en la actualidad no es fruto de unos cuantos años de trabajo, muy lejos de ello, ha sido un proceso de décadas completas de arduo trabajo del cual la historia ha sido testigo y ha quedado registrado en la misma como las generaciones de la computación.

1.1.1. Primera generación: Mainframe

En la primera generación de la computación se diseñaron las primeras computadoras conocidas como Mainframes, o coloquialmente como grandes planchas. Debido a su gran tamaño (podían abarcar un sótano o incluso un edificio completo), alto precio y gran peso.

Su uso se limitó a compañías que necesitaban procesar una cantidad considerable de datos (producto de censos, estadísticas de consumidores, recursos de planeación, transacciones bancarias, entre otros), debido a su alto costo las empresas no podían adquirir varios Mainframes, por lo que se utilizaban como computadoras centrales, es decir, tenían múltiples terminales conectadas de manera que múltiples usuarios se turnaban el uso del poder de procesamiento del Mainframe. Esto marcó el inicio de la computación centralizada y el procesamiento de transacciones en tiempo real (ver figura1).

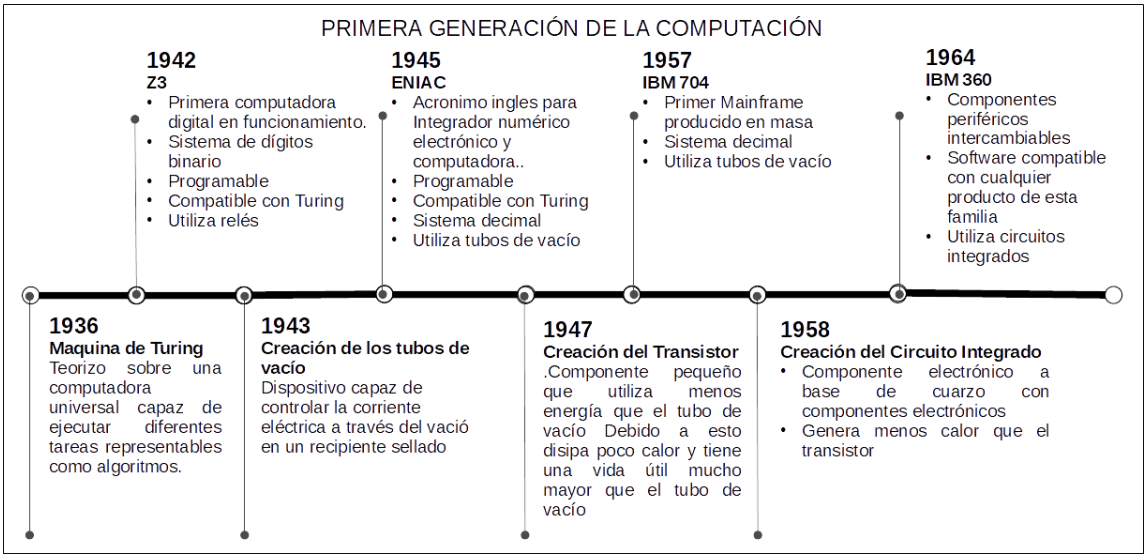
Figura 1. **Múltiples terminales conectadas a un Mainframe**



Fuente: elaboración propia.

La centralización fue un gran avance en esta primera generación de la computadora. Incluso desde esta generación se establecieron algunos conceptos bases de Cloud Computing, tal como los recursos computacionales como servicio, algunas empresas con alto poder adquisitivo que se equipaban con mainframes y ofrecían tiempo compartido por determinada cuota a otras empresas que no podían costearlas. Existen muchos Mainframes de primera generación, sin embargo, los más importantes son:

Figura 2. **Primera generación de la computación**



Fuente: elaboración propia.

Los Mainframes resolvieron parcialmente el problema del manejo de grandes cantidades de información, pero presentaron problemas en cuanto a funcionamiento y rendimiento debido a la tecnológica limitada e ineficiente de ese entonces (relés y tubos de vacío), sin embargo, la invención de un interruptor eléctrico hecho de materiales sólidos que no necesitaba vacío para transportar la electricidad, mejor conocido como transistor, en 1947 marcó un

antes y después en la historia de la computadora, debido a su mejor rendimiento, capacidad, eficiencia, confiabilidad y reducido tamaño se posicionó como el perfecto candidato para sustituir el uso de relés y tubos de vacío en la construcción de computadoras para mejorar su funcionamiento.

En 1958 se creó el primer circuito integrado, el cual consistía en la integración de un conjunto de transistores sobre un cuerpo semiconductor en un espacio reducido. Esto revolucionó la construcción de las computadoras, ya que cientos de tubos de vacío podían ser fácilmente sustituidos por un circuito integrado, reduciendo así considerablemente el tamaño de los mainframes.

1.1.2. Segunda generación: Computadora personal

Esta generación de la computación se caracterizó por las constantes mejoras a la computadora tanto en términos de hardware como de software. Al principio experimentó la reducción física de sus componentes gracias al empleo de transistores.

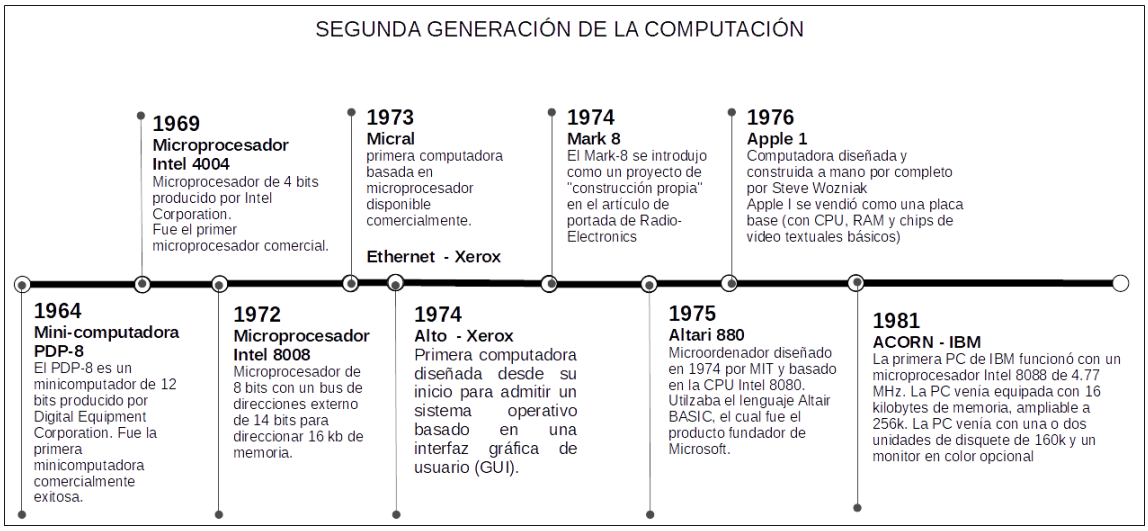
Además, uno de los principales logros alcanzados fue el desarrollo del componente encargado de llevar a cabo las instrucciones indicadas utilizando operaciones aritméticas básicas, lógicas y utilizando los dispositivos de entrada/salida conocida como unidad central de procesamiento (CPU) a base de transistores, no obstante, el invento que impulso de manera drástica el desarrollo de la computadora fue el desarrollo del circuito integrado más avanzado de ese entonces: el microprocesador. El microprocesador es la base de las computadoras personales modernas (PC), ya que permite integrar uno o más CPU en un componente de tamaño muy pequeño para aprovisionar de una gran capacidad de cómputo a una computadora sin aumentar su tamaño físico.

Las primeras PC bajaron de precio en comparación de los Mainframes, sin embargo, en un inicio resultaban muy costosas para ser adquiridas por el público general, pero las mejoras obtenidas en cuanto a funcionamiento y rendimiento las convirtió en una buena inversión para las empresas en orden de permitir a cada empleado tener a su disposición poder de procesamiento y almacenamiento independiente para aumentar la productividad eliminando los cuellos de botella que se experimentaban con el tiempo compartido de los Mainframes debido a la centralización de los recursos de procesamiento y almacenamiento.

Así mismo, en esta generación se desarrollaron algoritmos y técnicas para optimizar el uso de las nuevas capacidades computacionales alcanzadas y se dieron cambios radicales en cuanto la forma de interactuar con la computadora. Se introdujeron distintos dispositivos de entrada y salida de datos para lograr una comunicación plena usuario-computadora, así como se delegó el manejo de los componentes físicos a los sistemas operativos y se mejoró la experiencia de los usuarios implementando interfaces gráficas.

Con el pasar del tiempo se diseñaron una cantidad innumerable de computadoras, pero solamente algunas de ellas contribuyeron directamente al desarrollo de la computadora moderna como la conocemos hoy en día. Los eventos más importantes suscitados en esta generación fueron:

Figura 3. **Segunda generación de la computación**



Fuente: elaboración propia.

1.1.3. Tercera generación: Red

Las computadoras personales fueron diseñadas para proveerle un conjunto de recursos de cómputo y almacenamiento individual a cada usuario, sin embargo, en el ámbito empresarial surgió la necesidad de poder comunicar una computadora con otra para poder compartir información. Para resolverlo se diseñó una solución de hardware/software que permitiera el envío de información entre computadoras, esta tecnología se llamó ethernet y fue desarrollada por un grupo de investigación de Xerox en el año 1973. A la conexión de dos o más computadoras se le denominó una red de computadoras.

Las redes de computadoras son clasificadas según el área geográfica que son capaces de abarcar, en sus inicios solamente se podían conectar computadoras en una misma área geográfica reducida (una oficina o un hogar),

a este tipo de red se le conoció como red de área local (LAN). Además, se establecieron dos modelos para el uso de redes computacionales:

- Punto a punto
    - Este modelo conecta dos computadoras directamente para poder compartir entre sí recursos lógicos, sin necesidad de utilizar dispositivos de almacenamiento externo.
    - Comúnmente utilizado en hogares.
- Cliente-Servidor
    - Este modelo permite conectar dos o más computadoras directamente para la centralización de recursos lógicos, es decir, una sola máquina almacenará los recursos lógicos (servidor) y las demás maquinas conectadas a la red (clientes) podrán acceder a esos recursos comunicándose directamente con esa máquina.
    - Comúnmente utilizado en entornos laborales.
    - Los servidores se clasifican según su funcionalidad, los más comunes son:
        - Servidores web: alojan sitios web.
        - Servidores de aplicación: servidores dedicados para descentralizar la ejecución de tareas y mejorar el performance de un sistema.
        - Servidores de base de datos: para centralizar la información y asegurar consistencia en los datos de una organización.
    - Un problema asociado a este modelo es que cualquier sobrecarga de solicitudes podría dejar fuera de servicio al servidor, lo cual no permitiría a los clientes acceder a los recursos lógicos. Para resolverlo se requiere de aumentar la capacidad del servidor escalando verticalmente para evitar la sobrecarga.

Con la creciente adopción del uso de computadoras portátiles en diversas áreas y la poca inversión que requieren hoy en día, cada vez son más las empresas y personas particulares que adquieren una computadora, de manera que su uso se ha extendido a cada rincón del mundo, logrando así una globalización tecnológica y trayendo consigo nuevos retos y desafíos.

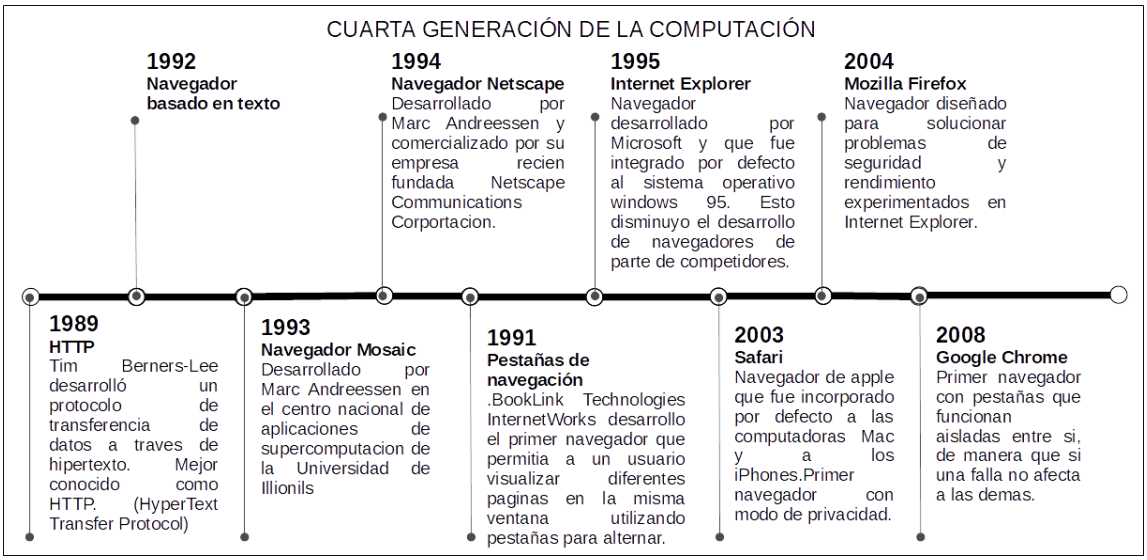
Esta globalización convirtió a la computadora en un dispositivo de comunicación, lo cual incitó a la creación de redes de computadoras de mayor alcance, tal como redes diseñadas para abarcar un área metropolitana (MAN) y redes de área extensa (WAN) que podría abarcar países o incluso continentes completos. Más adelante, en 1969 se consiguió uno de los hitos más grandes de la humanidad: la creación de una red de redes de alcance mundial, la cual tomó el nombre de Internet.

1.1.4. Cuarta generación: Web

Poco a poco la computadora pasó de ser una máquina capaz solamente de procesar información a ser un medio de comunicación, y con la llegada del Internet la comunicación entre computadoras a gran escala fue posible. Sin embargo, fue necesario e imperativo el desarrollo de un sistema estandarizado para el intercambio de información entre computadoras, dando origen a la web.

La web se define como un sistema de intercambio de información basado en el modelo cliente-servidor utilizando hipertexto e hipermedia. En este sistema los servidores son máquinas que almacenan archivos o información y los devuelven en formato hipertexto (HTML) de manera que puedan ser interpretados por programas instalados en los dispositivos clientes conocidos como navegadores. La evolución de la web a través del tiempo se puede resumir en los siguientes eventos:

Figura 4. **Cuarta generación de la computación**



Fuente: elaboración propia.

En esta generación de la computación se desarrolló el concepto de *hosting*, que consistía en que una empresa ofrecía espacio dentro de su centro de datos para que un individuo u organización a través de la web pudiera desplegar su sitio web o almacenar archivos sin necesidad de preocuparse de toda la infraestructura y configuraciones que esto implica, su característica principal era que el usuario aceptaba planes preestablecidos de servicio, comúnmente poco flexibles.

Con el transcurrir del tiempo la web se estableció como el sistema de intercambio de información por defecto de Internet gracias a su efectividad y gran potencial. En los últimos años se han desarrollado nuevas tecnologías (*frameworks*) que han permitido desarrollar programas completos e interactivos como: redes sociales, inventarios, procesadores de texto u hojas de cálculo, que pueden ser utilizados a través de la web.

A este concepto de programas a través de la web también se le conoce con el nombre de software como servicio (SaaS por sus siglas en inglés) y es la forma como funciona desde entonces la mayor parte de la web. SaaS elimina la necesidad de cumplir con requisitos específicos de hardware o software complementario para utilizar un programa, debido a que el usuario interactúa directamente con la información mediante páginas web, que pueden ser interpretadas por cualquier navegador web desde cualquier dispositivo conectado a Internet, delegando el cómputo de la información directamente al servidor donde se está ejecutando la aplicación.

Internet se ha convertido en el medio principal de comunicación en la era actual, su uso ha cambiado por completo modelos existentes de negocios e incluso introducido algunos nuevos, por ejemplo: ya no se visitan bibliotecas porque ahora toda la información puede ser obtenida fácilmente a través de una búsqueda en Google; ya no se rentan películas específicas en centros de entretenimiento porque ahora se tiene un catálogo variado en plataformas como Netflix; ya no es necesario hacer largas colas en el banco porque ahora se tiene la banca en línea; y así como estos casos hay muchos más que basan su funcionamiento completamente en Internet.

Hoy en día las aplicaciones web están al alcance de cualquier usuario con una conexión a Internet desde cualquier dispositivo móvil, por lo cual tanto la infraestructura donde esta se ejecuta como la arquitectura del sistema deben ser capaces de responder a términos de escalabilidad, alta disponibilidad, seguridad y demás, para lograr competir con las demás alternativas disponibles en el mercado.

1.2. Justificación

El pensum de Ingeniería en Ciencias de la Computación y Sistemas de Información de la Universidad de San Carlos de Guatemala cuenta con una rama de desarrollo de software. En esta rama los primeros siete semestres de la carrera se enfocan en desarrollar las habilidades de programación de los estudiantes para que puedan ser capaces de desarrollar sistemas, sin embargo, para desarrollar sistemas no hace falta ser ingeniero, porque cualquier persona con los conceptos de programación y la suficiente experiencia en cualquiera de los distintos lenguajes de programación puede ejecutar esta tarea.

La diferencia principal entre un ingeniero en sistemas y un programador, sin afán de restarle valor al trabajo de este último, consiste en que un ingeniero está capacitado para diseñar sistemas computacionales aplicando a su criterio los distintos patrones de diseño y de arquitectura para cumplir con los requerimientos y además asegurar un sistema óptimo y escalable.

Razón por la cual, del octavo semestre en adelante, los cursos del área de desarrollo de software del pensum de la carrera dejan por un lado los conceptos de programación y se enfocan en preparar al estudiante en el análisis de requerimientos tanto funcionales como no funcionales de un sistema para que, según su criterio, seleccione la metodología de trabajo que se utilizará en el proyecto, la arquitectura del sistema y la infraestructura necesaria para que el sistema funcione de una manera óptima y escalable, además de cómo aplicar conceptos de gobernabilidad en IT.

En el octavo semestre en el área de desarrollo de software se encuentra el curso de Análisis y Diseño de Sistemas 1, el cual se centra específicamente en el ciclo de vida de software e introduce conceptos de metodologías ágiles que

se utilizan actualmente en el mundo laboral para manejar un proyecto en equipo y que este logre culminar con éxito.

En el noveno semestre en el curso de Análisis y Diseño de Sistemas 2 el estudiante aprende a diseñar sistemas computacionales utilizando los distintos patrones de diseño y de arquitectura tanto de software como de hardware, para satisfacer los requerimientos funcionales y no funcionales de un sistema.

En el décimo semestre en el curso de Software Avanzado el estudiante interactúa con conceptos de manejo IT, gobernanza IT y SOA. En el laboratorio de este curso (a veces debido a que los programas son dependientes del ingeniero que imparte el curso) se introduce al estudiante a conceptos de DevOps.

Sin embargo, conocer solamente los conceptos representa menos de la mitad de lo que se necesita para que un egresado de sistemas de esta casa de estudios trabaje como un ingeniero DevOps, debido a que la cultura DevOps se apoya fuertemente en el uso de herramientas para la automatización en búsqueda de combinar procesos operacionales con procesos de desarrollo y generar más valor en menor tiempo. Por lo que se considera imperativo que los estudiantes desarrollen las habilidades necesarias para configurar y mantener infraestructuras basadas en la nube haciendo uso de estas herramientas.

El problema reside en que estas herramientas no son parte del programa del curso ni del laboratorio, por lo que impartir algún taller sobre estas herramientas queda completamente a discreción del auxiliar o del catedrático del curso. Aunque la escuela de sistemas introduzca cada vez más conceptos de Cloud Computing en los distintos cursos del pensum, sin el entrenamiento en

las diferentes herramientas que aprovechan al máximo el paradigma de computación Cloud, estos conocimientos serán subutilizados.

Para el informe *Upskilling 2020: Enterprise DevOps Skills Report* realizado por The DevOps Institute, se encuestó a 1 260 individuos alrededor del mundo. Entre sus principales claves exponen.

- Los puestos para ingenieros DevOps han casi duplicado su popularidad.
- El conocimiento de la cadena de herramientas de CI/CD es una habilidad indispensable.
- Encontrar y atraer a personas con habilidades DevOps continúa siendo un desafío en 2020[4].

El 58 % de los encuestados indicaron que encontrar personas capacitadas es un gran desafío, mientras que el 48 % indicó que la retención de personas capacitadas DevOps también lo es. Esto se debe a la alta demanda de personas con habilidades Cloud y DevOps entre empresas.

Ante esta demanda las empresas se ven obligadas a entrenar personal para posteriormente contratarlos. El desarrollar habilidades en las distintas herramientas DevOps desde la universidad promete a los egresados de la carrera de sistemas obtener mejores oportunidades en el ambiente laboral e impulsar su carrera profesional.

La falta de cursos y talleres de parte de la escuela que aborde el uso de estas herramientas y su uso en las aplicaciones Cloud nativas obliga al estudiante a buscar por sus medios la forma de aprender y practicar estas

[4] OEHRLICH, Eveline; GROLL, Jayne; GARBANI, Jean-Pierre. *Upskilling 2020: Enterprise DevOps Skills Report.* p. 10

herramientas, que son imprescindibles en la cultura DevOps. En el mejor de los casos aprenderlo por cuenta propia a través de Internet podría costar tiempo y en el peor de los casos abandonar la idea al sentirse agobiado debido a la amplitud de los temas y no tener algún tipo de mentor o tutor que le apoye en dudas o enseñándole buenas prácticas.

A través de la presente tesis se espera evidenciar la demanda que existe actualmente de ingenieros con habilidades en herramientas DevOps, la relación simbiótica que existe entre DevOps y Cloud Computing, además elaborar talleres sobre la tecnología de contenedores utilizando Docker y su orquestación con Kubernetes con el objetivo de que puedan ser utilizados como material complementario en los cursos de Análisis y Diseño de Sistemas 2 y Software Avanzado, para contribuir con el desarrollo de las destrezas de las futuras generaciones de ingenieros egresados de la Escuela de Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala.

2. CLOUD COMPUTING

2.1. Descripción

Cloud Computing es un paradigma de computación que abarca tanto la configuración como acceso a recursos computacionales a través de Internet. Aunque el término Cloud Computing es relativamente nuevo, en realidad el concepto no es más que la unión de conceptos que han existido desde hace mucho tiempo en el mundo IT.[5]

Algunos de estos conceptos que conforman a lo que se conoce como Cloud Computing son:

- Computación paralela: es un tipo de computación en la cual existen muchos procesos en ejecución de manera simultánea.
- Computación distribuida: es un modelo en el cual los componentes de un sistema informático son compartidos entre múltiples computadoras para mejorar la eficiencia y rendimiento.
- Computación Grid: se denomina computación Grid a la colección de recursos computacionales en múltiples locaciones que funcionan en conjunto para alcanzar un objetivo.
- Computación como utilidad: se origina cuando se ofrecen recursos computacionales a usuarios finales como servicios medidos en el modelo pago bajo consumo.

[5] BOHM, Markus; KRCMAR, Helmut; LEIMEISTER, Stefanie; RIEDL, Christoph. *Cloud Computing and Computing Evolution.* p. 10

- Virtualización: abstrae detalles de bajo nivel del hardware y otros recursos físicos para poner simulaciones de los recursos físicos al servicio de aplicaciones de alto nivel.

Cloud Computing hace lo mismo que ya se hacía desde hace mucho tiempo, pero de otra manera. La verdadera innovación de Cloud Computing reside en la forma en que este provee servicios computacionales a sus consumidores. Este paradigma de computación coincide con la visión de Jhon MacCarthy expuesta en su discurso en MIT en 1961, donde sugirió que la computación debería ser vendida como un servicio tal como la electricidad y la comida. Su idea era grandiosa pero muy adelantada a su época y tecnología en ese entonces.

La ventaja principal de este paradigma de computación es que el uso de los recursos se factura bajo un modelo de pago por consumo como si fuesen un servicio, tal como el agua potable o la electricidad, lo cual permite a sus usuarios invertir más en lo que mejor hacen, es decir, su negocio y olvidarse tanto del mantenimiento como la inversión que aprovisionarse de una infraestructura representa. Además, en Cloud Computing los usuarios pueden configurar y utilizar solamente los recursos que necesitan. Sin restricciones ni límites mínimos o máximos, de manera que sus costos son totalmente dependientes del uso de los recursos que implementan o utilizan en determinado periodo de tiempo.

Cloud Computing permite a las organizaciones o individuos compartir varios servicios de manera transparente y rentable a través de Internet.[6] La rentabilidad de este paradigma de computación se debe a que se apoya fuertemente en el uso de virtualización de recursos computacionales, es decir,

[6] VERMA, Trilochan; VERMA, Anjali. *Cloud computing: evolution and challenges.* p. 10 198

el proveedor de servicios posee una infraestructura física robusta, pero en orden de proveerle la flexibilidad a sus usuarios para que puedan configurar y utilizar la cantidad exacta de recursos que necesita se crea una virtualización del recurso físico con las características exactas que se necesitan permitiendo que múltiples usuarios compartan el mismo recurso físico de una manera controlada y segura.

La virtualización permite hacer de Cloud Computing un negocio rentable ya que tanto la inversión inicial como el mantenimiento de la infraestructura es costeada por cientos, miles e incluso millones de usuarios alrededor del mundo.

Es fácil confundir Cloud Computing con los tradicionales centros de datos, pero la gran diferencia que existe entre estos radica en que Cloud Computing utiliza software para virtualizar los recursos físicos para brindar a los usuarios la flexibilidad de utilizar solo lo que necesitan. Además, que estos servicios son configurados y consumidos directamente a través de Internet en lugar de hacerlo manualmente.

Según NIST para que una infraestructura sea considerada como Cloud debe cumplir con las siguientes cinco características esenciales:

- Proveer servicios bajo demanda
- Ofrecer un amplio acceso a la red
- Ofrecer un conjunto de recursos compartidos
- Tener rápida elasticidad
- Medir el uso de los servicios

Se recomienda el uso de Cloud Computing en los siguientes escenarios de negocio:

- Proyectos con presupuestos limitados
- Cuando los costos de manejo y operación IT son muy altos o complejos
- Procesos de negocio con demanda impredecible

2.2. Modelos de despliegue

Cloud Computing se puede clasificar según la propiedad de la infraestructura que utiliza para proporcionar los servicios y según el tipo de usuario que puede hacer uso de estos servicios. De acuerdo con el Instituto Nacional de Estándares y Tecnología de Estados Unidos (NIST, por sus siglas en inglés) existen cuatro modelos de despliegue de nubes: pública, privada, comunitaria e hibrida.[7]

2.2.1. Pública

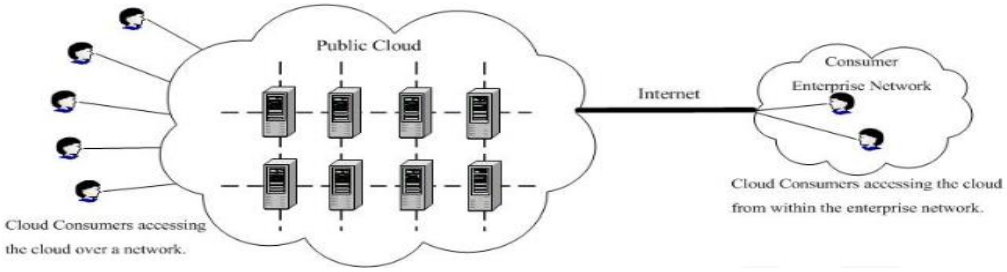
Este modelo de nube se despliega en la infraestructura de una empresa con un alto poder adquisitivo, mejor conocida como proveedor de servicio, y pone a disposición de cualquier empresa, organización o individuo el uso de su infraestructura para configurar o consumir recursos computacionales como servicio. La ventaja principal de utilizar los servicios de algún proveedor de servicios es que tienen centros de datos distribuidos alrededor del mundo que ayudan a minimizar la latencia en la comunicación y permite mejorar el rendimiento de una aplicación.

En este tipo de nube el proveedor de servicio asume tanto la inversión inicial requerida para adquirir la infraestructura como el costo de mantenimiento de esta, aliviando de cierta manera los presupuestos de sus clientes

[7] SHARMA, Kunal; THAKUR, Shusheel; KALIA, Arvind; THAKUR, Jawahar; KUMAR, Sunil. *Emerging cloud computing paradigm: vision, research challenges and development trends.* p. 2

brindándoles la oportunidad de invertir más en su negocio y lanzar de una manera más económica y rápida nuevas aplicaciones. Los servicios son contratados bajo acuerdos de términos y condiciones donde el proveedor expone su compromiso en cuanto al nivel de disponibilidad de los servicios contratados, privacidad y seguridad de los datos que almacenará en sus centros de datos.

Figura 5. **Nube pública**



Fuente: National Institute of Standards and Technology of United States. *Emerging cloud computing paradigm: vision, research challenges and development trends.* https://www.nist.gov/system/files/documents/2017/05/31/evaluation_of_cloud_computing_services_based_on_nist_800-145_20170427clean.pdf. Consulta: abril de 2020.

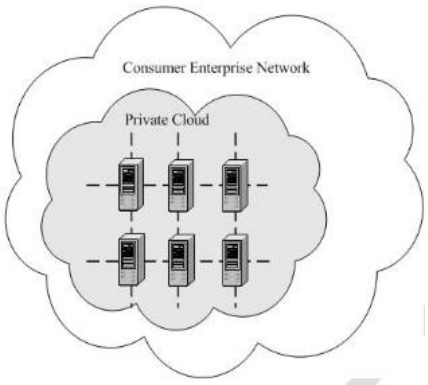
2.2.2. Privada

Las empresas que trabajan con información sensible son escépticas en cuanto a la seguridad de los datos que los proveedores de servicios Cloud ofrecen, por lo cual para aprovechar los beneficios de los servicios Cloud tradicionalmente se ven en la obligación de aprovisionarse de una infraestructura, por lo que deben cubrir tanto la inversión inicial como los gastos de mantenimiento de esta. Estos servicios Cloud serán solamente para uso

exclusivo de sus dependencias. La principal razón de implementar este modelo de nube es tener mayor control sobre los datos.

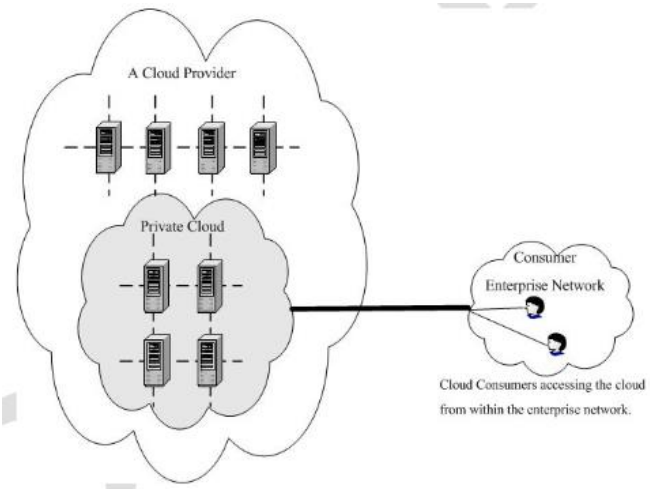
Cada vez el número de instituciones gubernamentales y empresas que requieren nubes privadas va en aumento, ante este mercado los proveedores de servicio han abierto la posibilidad de dedicarle parte de su infraestructura exclusivamente a estas organizaciones de manera que, aunque estos no tengan la infraestructura en casa puedan aprovechar los beneficios de la nube al llegar a un acuerdo donde el proveedor de servicios les pueda garantizar que le dedicarán parte de su infraestructura y no será utilizada por ninguna otra empresa, además de establecer de manera explícita los niveles de seguridad y privacidad bajo los cuales se deberán regir en todo momento los datos almacenados en sus centros de datos.

Figura 6. **Nube privada propia**



Fuente: National Institute of Standards and Technology of United States. *Emerging cloud computing paradigm: vision, research challenges and development trends.* https://www.nist.gov/system/files/documents/2017/05/31/evaluation_of_cloud_computing_services_based_on_nist_800-145_20170427clean.pdf. Consulta: abril de 2020.

Figura 7. **Nube privada subcontratada**

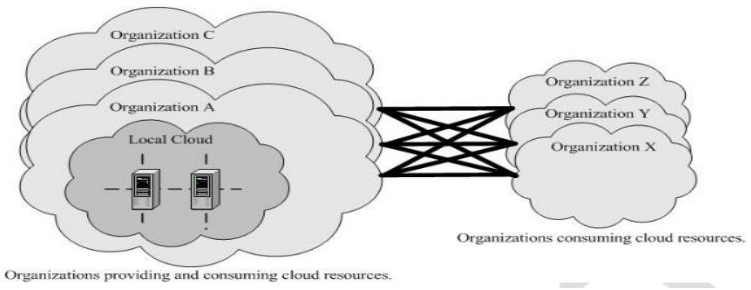


Fuente: National Institute of Standards and Technology of United States. *Emerging cloud computing paradigm: vision, research challenges and development trends.* https://www.nist.gov/system/files/documents/2017/05/31/evaluation_of_cloud_computing_services_based_on_nist_800-145_20170427clean.pdf. Consulta: abril de 2020.

2.2.3. Comunitaria

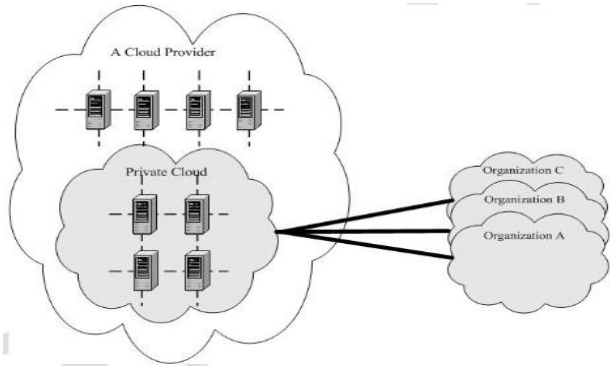
Este modelo se presenta cuando múltiples empresas con necesidades Cloud similares se ponen de acuerdo y asumen en conjunto la inversión inicial para la adquisición de recursos computacionales y el gasto periódico utilizado para el mantenimiento de este. Los servicios solamente pueden ser utilizados por dichas empresas. Además, también existe la posibilidad de subcontratar la infraestructura de un proveedor de servicios, este concepto es similar al de nube privada subcontratada, pero con la diferencia que esta parte de la infraestructura del proveedor de servicios la utilizarán múltiples empresas con fines y necesidades en común.

Figura 8. **Nube comunitaria propia**



Fuente: National Institute of Standards and Technology of United States. *Emerging cloud computing paradigm: vision, research challenges and development trends.* https://www.nist.gov/system/files/documents/2017/05/31/evaluation_of_cloud_computing_services_based_on_nist_800-145_20170427clean.pdf. Consulta: abril de 2020.

Figura 9. **Nube comunitaria subcontratada**

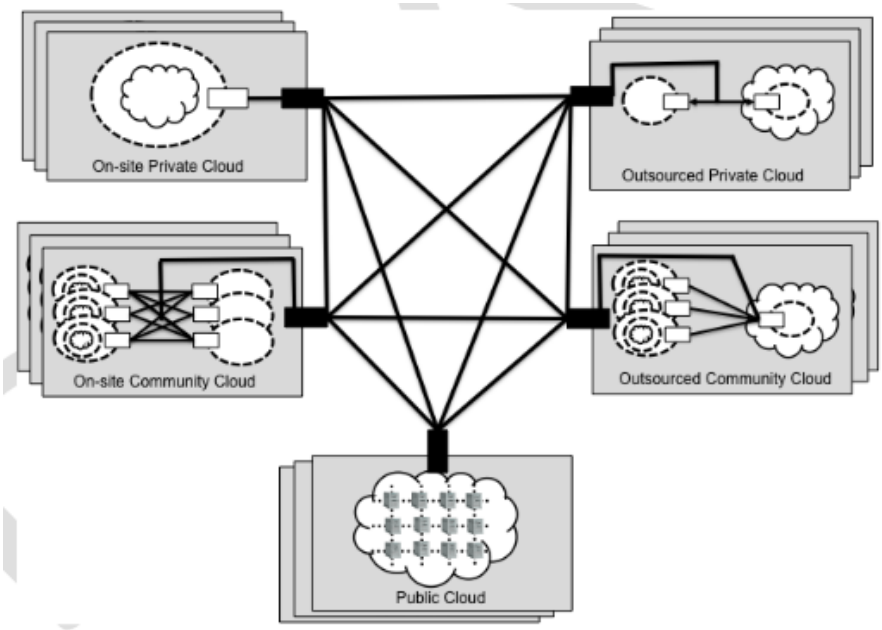


Fuente: National Institute of Standards and Technology of United States. *Emerging cloud computing paradigm: vision, research challenges and development trends.* https://www.nist.gov/system/files/documents/2017/05/31/evaluation_of_cloud_computing_services_based_on_nist_800-145_20170427clean.pdf. Consulta: abril de 2020.

2.2.4. Híbrida

Este tipo de nube se origina cuando se combinan dos o más tipos de nube (pública, privada o comunitaria), los cuales permanecen como entidades únicas, pero están unidas por tecnologías estandarizadas o patentadas que permite portabilidad de los datos y aplicaciones.

Figura 10. **Nube híbrida**



Fuente: National Institute of Standards and Technology of United States. *Emerging cloud computing paradigm: vision, research challenges and development trends.* https://www.nist.gov/system/files/documents/2017/05/31/evaluation_of_cloud_computing_services_based_on_nist_800-145_20170427clean.pdf._Consulta: abril de 2020.

2.3. Proveedores de servicio

Los proveedores de servicio en la nube son empresas con alto poder adquisitivo que se aprovisionan de recursos computacionales para arrendarlos a organizaciones gubernamentales, empresas o individuos particulares.

El proveedor de servicio es quien asume la inversión inicial necesaria para adquirir los recursos informáticos, en su defecto grandes cantidades de recursos informáticos, y los venden como servicios para facilitar que las organizaciones puedan escalar sus aplicaciones de manera automatizada.

Aunque los proveedores ofrecen una solución atractiva al problema de costos de una organización, así mismo también tienen sus puntos en contra. El principal punto en contra es la seguridad de los datos, este es el principal motivo por el cual muchas empresas se resisten a migrar a Cloud, porque no se tiene la completa certeza de dónde están almacenados los datos, lo cual genera desconfianza. Particularmente este uno de los principales problemas para la completa adopción de Cloud Computing en Guatemala, sumado al hecho de que son pocas personas que conocen y han tenido preparación en conceptos de Cloud Computing.

En la actualidad cada vez son más las empresas que ofrecen soluciones Cloud, sin embargo, este campo está liderado por tres empresas principales: Amazon Web Services (AWS), Google Cloud Platform (GCP) y Microsoft Azure.

2.3.1. Amazon Web Services

AWS es una plataforma que ofrece soluciones rápidas, flexibles, confiables y económicas. Se le considera como el proveedor líder de servicios de infraestructura como servicio.

Figura 11. **Amazon Web Services**

Fuente: *Logo de Amazon Web Services.* https://www.guru99.com/cloud-computing-service-provider.html. Consulta: abril de 2020.

2.3.2. Azure

Azure es la plataforma de servicios de la nube de Microsoft. Esta nube se encuentra en el puesto 2 y tiene como fuerte sus soluciones de software como servicio tal como Office 365 y Dynamics.

Figura 12. **Azure**

Fuente: *Logo de Microsoft Azure.* https://www.guru99.com/cloud-computing-service-provider.html. Consulta: abril de 2020.

2.3.3. Google Cloud Platform

Es un conjunto de soluciones y productos que incluyen Google Cloud Platform y G Suite. Ayuda a resolver todo tipo de desafíos de negocio fácilmente.

Figura 13. **Google Cloud Platform**

Fuente: Guru99. *Logo de Google Cloud Platform*. https://www.guru99.com/cloud-computing-service-provider.html. Consulta: abril de 2020.

2.4. Modelos de servicio

Cloud Computing es un paradigma de computación en el cual sus servicios se clasifican por capas o niveles, cada servicio de nivel superior le abstrae al usuario el manejo y control de todos los servicios de nivel inferior que necesita para funcionar correctamente. A estos niveles de abstracción se les conoce como modelos de servicio:

2.4.1. IaaS

Este modelo de infraestructura como servicio (del inglés, *Infrastructure as a Service*) es el de más bajo nivel porque el usuario se encarga de crear,

configurar y mantener tanto la aplicación como todos los servicios básicos de almacenamiento, red y cómputo necesarios para que la aplicación funcione y a la vez satisfaga las necesidades de una organización o individuo, tanto en términos de seguridad como en rendimiento. IaaS está diseñado específicamente para arquitectos de infraestructura porque para configurar cualquiera de los servicios de este modelo se requiere de un alto conocimiento técnico.

En este nivel el proveedor de servicios solamente se encarga de la infraestructura física donde los servicios Cloud se estarán virtualizando.

2.4.2. PaaS

En el modelo de plataforma como servicio (del inglés, *Platform as a Service*), la creación, manejo y mantenimiento de servicios de red, almacenamiento y cómputo se le delega directamente al proveedor de servicios, por lo que para hacer uso de los distintos servicios que ofrece no es necesario ser un ingeniero en sistemas o tener un amplio conocimiento de infraestructura computacional.

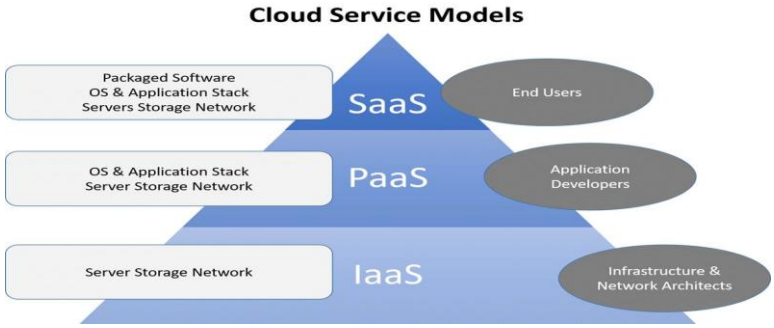
Este modelo está diseñado específicamente para desarrolladores porque le abstrae los servicios de infraestructura y permite que se enfoquen en lo que mejor hacen: desarrollar. Gracias a esto los desarrolladores se ahorran el tiempo que emplearían en configuración y mantenimiento de los servicios de cómputo, almacenamiento y redes. Además, les permite ahorrar dinero porque este modelo factura por cantidad de tráfico y tiempo de uso.

2.4.3. SaaS

El modelo de programas como servicio, del inglés *Software as a Service*, es el modelo de más alto nivel, debido a que le abstrae al usuario la infraestructura, plataforma y la aplicación misma. Está diseñado para los clientes finales. Los servicios (programas) de este modelo son accedidos y consumidos a través de un navegador eliminando la necesidad de instalar el programa o tener que cumplir con requerimientos de software o hardware para utilizar el programa. Algunos ejemplos de este modelo SaaS podrían ser:

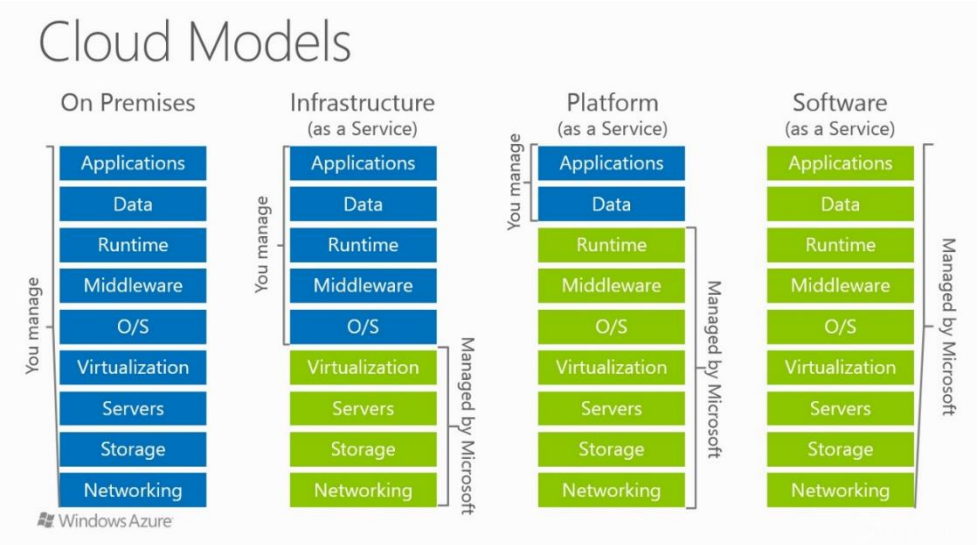
- Google Apps
- Gmail
- Youtube
- Facebook
- Twitter

Figura 14. **Principales modelos de servicio Cloud**



Fuente: Smartkidsproject.*Cloud Service Models.* https://smartkidsproject.appspot.com/ecte/unit?unit=24&lesson=85. Consulta: abril de 2020.

Figura 15. **Abstracciones de los modelos de servicio**



Fuente: Carlfreeman. *Cloud Models.* https://carlfreeman.net/2017/02/11/iaas-vs-paas-vs-caas-this-is-a-good-summary-of-the-terms-and-what-to-know-confused-by-these-types-of-pictures-your-tech-teams-keep-showing-you/. Consulta: abril de 2020.

2.5. Servicios ofrecidos por proveedores Cloud

Algunos de los tipos de servicios que ofrecen los distintos proveedores Cloud son:

- Almacenamiento
- Cómputo
- Redes
- *Big Data*
- *Machine Learning*
- IoT

Para configurar estos servicios los proveedores de servicio también facilitan a sus usuarios herramientas de:

- Gestión
- Seguridad
- Desarrollo

2.6. Beneficios de Cloud Computing

- Flexibilidad: el usuario puede escoger la capacidad de sus servicios a voluntad y decidir así mismo por cuánto tiempo los necesita funcionando para pagar solo lo que utilizó.
- Escalabilidad: al ser todo configurado a través de la nube se pueden establecer políticas para que la infraestructura crezca o disminuya según la necesidad del cliente.
- Disminución de costos: gracias a la escalabilidad y la flexibilidad ofrecida por Cloud, el usuario solamente utiliza los recursos según su carga de trabajo en el momento, por lo que le permite ahorrar en aquellos momentos donde la carga no es mucha.
- Sustentabilidad: no se requiere de una inversión inicial, por lo que los negocios pueden sustentar el uso de los servicios computacionales con parte de lo que generan.
- Fácil implementación: es mucho más fácil utilizar los servicios de la nube, basta con configurarlos desde la plataforma del proveedor.

2.7. Desventajas

Así como Cloud Computing ofrece ventajas, también tiene desventajas propias del paradigma. Algunas de estas son:

- Dependencia de proveedor (del inglés, *vendor lock-in*), esto sucede cuando un usuario utiliza los servicios ofrecidos por proveedor, pero a la vez se ata a este porque las configuraciones utilizadas para utilizar ese servicio son propias del proveedor o solamente tiene acoplamiento con otros servicios del mismo proveedor. Esto le quita la oportunidad al usuario de elegir algún servicio de otro proveedor que le pudiera funcionar de mejor manera o le sea más económico
- Entre más datos tenga el usuario en un proveedor de servicios, más dependiente se vuelve de este.
- Privacidad de los datos, el usuario no está completamente seguro en donde están almacenados y qué nivel de seguridad tienen estos.

2.8. Vulnerabilidades

De acuerdo con Kilari, algunas de las vulnerabilidades de Cloud Computing son:

- Vulnerabilidades en virtualización
- Vulnerabilidades en los protocolos de Internet
- Accesos no autorizados para interfaces de manejos
- Vulnerabilidades de inyección
- Vulnerabilidades en navegadores y APIs[8]

2.9. Impacto de Cloud Computing

A la era actual se le considera como la era de consumo, ya que todo puede ser consumido fácilmente a través de Internet. Por lo que las necesidades de infraestructura computacional de las empresas han

[8] KILARI, Nagaraju. *Cloud Computing - An Overview & Evolution.* p. 150

evolucionado. Estas deben estar listas para responder ante cualquier eventual crecimiento de usuarios que puede suceder de una manera impredecible, para siempre asegurarle el servicio a cualquier usuario que lo necesite y evitar una fuga de clientes a la competencia.

Cloud Computing permite a las organizaciones mantener la competitividad en el mercado al permitirles aprovisionarse de altos niveles de cómputo en cuestión de segundos, consecuentemente pueden poner en funcionamiento sus productos en el menor tiempo posible, ahorrándole costos de inversión y de mantenimiento de infraestructura, ya que solo se cancela según el tiempo utilizado y tráfico recibido.

Cloud Computing a la vez ha generado un cambio en el modelo de costos para recursos IT, ya que en lugar de la tradicional inversión inicial las empresas cada vez se acostumbran a modelos de pago bajo consumo y son considerados como gastos diarios. Es por esto por lo que Cloud Computing se ha establecido firmemente como el nuevo normal para empresas IT. Además, les permite a las empresas, con el empleo de las herramientas de automatización adecuadas, agilizar de manera inigualable sus procesos de desarrollo de software.

2.10. Panorama actual de Cloud Computing

De acuerdo con el artículo *4 trends impacting cloud adoption in 2020*, la falta de personas con habilidades en IaaS retrasará las migraciones de aplicaciones a la nube alrededor del mundo hasta el 2022.[9]

[9] RIMOL, Meghan. *4 trends impacting cloud adoption in 2020.* https://www.gartner.com/smarterwithgartner/4-trends-impacting-cloud-adoption-in-2020/.

Esta falta de personas capacitadas conduce a las empresas a una incorrecta migración a Cloud. Actualmente las estrategias de migración de las empresas tienden a levantar y entregar[10]. Los errores más comunes que se cometen son:

- Migrar por completo una aplicación de negocios utilizando el paradigma tradicional de IT. Como resultado, se despliegan en Cloud aplicaciones monolíticas y que son configuradas de manera estática en un par de centros de datos, sin aprovechar las características dinámicas de Cloud.
- Se desarrolla aplicaciones de negocio en el tradicional marco de trabajo IT.

Pero, aunque existan las personas con las destrezas Cloud requeridas, sin las habilidades en las distintas herramientas de automatización que aprovechan al máximo el dinamismo de Cloud, no se podrá sacar el máximo provecho a este paradigma de computación.

Cloud Computing aún es considerado por algunas universidades y empresas como una moda que en algún momento puede desaparecer, esta postura es entendible porque en el campo IT cada año se desarrollan nuevas tecnologías, paradigmas de computación, herramientas, entre otras, y después de cierto tiempo la fiebre por el uso de estas pasa y no vuelven a usarse nunca más. Sin embargo, este no es el caso de Cloud Computing ya que se prevé que hasta el 80 % de las empresas migren sus aplicaciones a la nube para 2025.[11]

[10] BOMMADEVARA, Nagendra; DEL MIGLIO, Andrea; JANSEN, Steve. *Cloud adoption to accelerate IT modernization.* https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/cloud-adoption-to-accelerate-it-modernization.
[11] CAREY, Scott. *Cloud Computing*. https://www.computerworld.com/article/3511418/cloud-computing-trends-for-2020.

2.11. ¿Qué se puede esperar en Cloud Computing?

- La efectividad de costos conducirá a una adopción de Cloud Computing.
    - La escalabilidad y flexibilidad de los servicios que ofrece Cloud Computing permitirá a las empresas optimizar sus costos. Por lo que esta característica conducirá a una completa adopción de Cloud Computing.
- Mercado liderado por algunos proveedores de servicio Cloud.
    - Gartner pronostica que 10 de los grandes proveedores Cloud públicos controlarán al menos la mitad del total del mercado Cloud público hasta al menos 2023.
- Nuevos servicios Cloud
    - Cloud Computing va madurando conforme pasa el tiempo, cada vez integrará más servicios, entre los que más se esperan son:
        - Inteligencia artificial
        - *Big Data*
        - IoT
- *Serverless*
    - Poco a poco se utilizará el nuevo tipo de computación *Serverless* que aprovecha al máximo las ventajas de los contenedores y les permite a los usuarios ejecutar código sin necesidad de aprovisionar y mantener servidores. Además, pagan según el tiempo que utilizaron la capacidad de cómputo.
    - Las soluciones actuales *Serverless* crean dependencia de proveedor de servicios, pero con la llegada de soluciones *Open Source* podrán implementarse en innumerables escenarios.
- *Edge Computing*
    - Ante la inminente proliferación de dispositivos IoT que requieren almacenamiento y análisis de grandes cantidades de datos en

tiempo real, los proveedores de servicio han apostado al despliegue de *micro data centers* que prometen minimizar la latencia en la comunicación, a esto se le conoce como *Edge Computing.*

- De acuerdo con Gartner, se espera que para 2023 los líderes de servicio Cloud tengan una presencia parecida a los ATM para estar cada vez más cerca de los usuarios y disminuir considerablemente la latencia en la comunicación[12].

- Multi Cloud
  - En una encuesta de Gartner dirigida a usuarios de Cloud pública, el 81 % respondió que ya están trabajando con dos o más proveedores.
  - Algunas razones de por qué muchas empresas adoptarán un modelo multi Cloud son:
    - Eliminar la dependencia de proveedor de servicios.
      - ✓ Se prevé que para 2024 se reduzca la dependencia de proveedores de servicio para dos tercios de todas las organizaciones a través del uso de multi Cloud.[13]
    - Optimizar el rendimiento de sus aplicaciones.
      - ✓ No todos los proveedores de servicio tienen la misma presencia en una misma zona geográfica, por lo que en orden de minimizar la latencia de comunicación para sus usuarios las empresas utilizarán servicios de múltiples proveedores de servicio Cloud.
      - ✓ Se puede dar el caso que un diferente proveedor de servicios ofrece un servicio que se acopla de mejor

[12] RIMOL, Meghan. *4 trends impacting cloud adoption in 2020.* https://www.gartner.com/smarterwithgartner/4-trends-impacting-cloud-adoption-in-2020/.
[13] Ibíd.

manera a las necesidades técnicas o de costo del usuario.

- Los contenedores y máquinas virtuales jugarán un papel clave para lograr adoptar un modelo multi Cloud.

- Inversión de empresas para capacitar personas.
  - La cada vez creciente migración de empresas a Cloud está creando un mercado donde los proveedores de servicio no pueden capacitar y certificar a las personas lo suficientemente rápido como para satisfacer la necesidad de profesionales calificados en la nube
  - Ante la falta de personal capacitado las empresas invertirán cada vez más en el entrenamiento de nuevos talentos.
  - Las empresas también optarán por contratar empresas conocidas como integradores de servicios, cuyo trabajo es ayudar a migrar a Cloud. Preferirán a aquellas que tengan un historial comprobado de migraciones exitosas dentro de la industria objetivo.
  - Citando el artículo *Cloud Computing predictions for 2020*: "A medida que aumenta la demanda de transformación en la nube, también lo hace la necesidad de nuevas habilidades dentro de una organización. El único problema es la brecha de habilidades en la nube con la que lidiar."[14]
  - Acorde con Carey, el enfoque multi Cloud requiere de especialistas, y a veces de herramientas específicas, para cada una de las plataformas Cloud. Lastimosamente alrededor del mundo estas habilidades están escasas y esto podría conducir a:
    - Esperar por muchos meses para contratar ingenieros con la suficiente experiencia Cloud.

[14] CAREY, Scott. *Cloud Computing predictions for 2020.* https://www.computerworld.com/article/3511418/cloud-computing-trends-for-2020.

- Contratar alguien mediocre.
- Invertir grandes cantidades de dinero para desarrollar talento en casa, lo cual podría llevar meses.
- Crear programas para capacitar a su personal en nuevas herramientas.
- Mejorar constantemente las condiciones de trabajo para evitar una fuga de talentos ante la escasez de personas capacitadas.

3. DESPLIEGUE DE APLICACIONES

El desarrollo de un sistema computacional sigue un ciclo de vida que consiste en varias etapas que se realizan secuencialmente tantas veces sean necesarias hasta que se logre satisfacer los requerimientos funcionales y no funcionales de un sistema. El ciclo de vida de un software consiste en las siguientes etapas:

- Plan: en esta etapa se elabora la planificación de cómo y qué funcionalidad se desarrollará durante la iteración.
- Desarrollo: se escribe el código de las nuevas funcionalidades lo más apegado posible a la planificación.
- Pruebas: se integran los cambios y se prueban todas las funcionalidades para asegurar una completa integración y calidad del software
- Despliegue/implementación: la etapa de despliegue/instalación en el ciclo de vida de software consiste en configurar y poner en marcha las nuevas versiones de una aplicación en la infraestructura que será utilizada directamente por los usuarios finales.
- Mantenimiento: se realizan los cambios de algún problema encontrado que no se haya observado en las etapas de prueba.

Una de las etapas cruciales, y más costosas, en el ciclo de vida de un proyecto es el despliegue/implementación de nuevas versiones del sistema que se está desarrollando porque es la única forma de interactuar con las mejoras o cambios introducidos en el sistema.

Figura 16. **Ciclo de desarrollo de software**

Fuente: Inflectra. *Ciclo de desarrollo de software.* https://www.inflectra.com/spirateam/highlights/understanding-alm-tools.aspx. Consulta: abril de 2020.

3.1. Ambientes de despliegue

El despliegue de una aplicación se lleva a cabo en múltiples etapas del ciclo de desarrollo de software (cada una de ellas cuenta con una infraestructura que también se conoce como ambiente) con el objetivo de integrar y verificar el comportamiento de las nuevas funcionalidades junto al resto del código.

3.1.1. Ambiente de desarrollo

Es utilizado en la etapa de desarrollo. Para no codificar a ciegas, es necesario que el equipo de desarrollo cuente con un entorno donde pueda verificar y validar si las nuevas funcionalidades programadas se comportan como se espera, este ambiente funciona como la primera barrera para encontrar errores de programación.

3.1.2. Ambiente de pruebas

Es utilizado en la etapa de pruebas. Una vez el programador considera que ha terminado por completo con la funcionalidad específica que se le solicitó se procede a realizar pruebas objetivas y exhaustivas del sistema completo, por lo cual se requiere un entorno completamente diferente al de desarrollo para no perjudicar ni interferir con el trabajo de los programadores en ese momento.

Sirve para asegurar la calidad del código escrito, verificando que todo funcione como se espera y que el nuevo código se integre con las demás funcionalidades de manera correcta.

3.1.3. Ambiente de producción

Es utilizado en la etapa de despliegue/instalación, es la etapa más importante y critica, porque es cuando se pone a disposición de los clientes la nueva versión de la aplicación. Es hasta en este punto donde los clientes pueden verificar si el sistema cumple con sus expectativas y dar retroalimentación.

3.2. Tipos de despliegue

Sin importar el ambiente donde será desplegada la nueva versión del sistema, existen múltiples maneras de desplegar una aplicación. Estas pueden ser:

3.2.1. Despliegue tradicional

Este despliegue se realiza directamente sobre una máquina física donde funcionará la aplicación, se puede realizar de manera manual o automatizada utilizando *scripts* o herramientas de configuraciones.

Uno de los problemas principales es que los recursos computacionales pueden ser subutilizados debido a que en ocasiones es imposible desplegar múltiples aplicaciones por conflictos entre versiones de una misma librería o incompatibilidad de sistema operativo. No existe solución alguna a estos conflictos, solamente desplegar la segunda aplicación en una máquina diferente.

3.2.2. Despliegue con máquinas virtuales

Este despliegue permite optimizar el uso de los recursos físicos, al permitir ejecutar múltiples aplicaciones en un mismo equipo. Al igual que la manera tradicional se puede configurar de forma manual o utilizando herramientas de configuración tal como Cheff, Puppet o Ansible.

Cada máquina virtual representa un entorno completamente aislado dentro de la computadora. Posee su propio sistema operativo, conjunto de binarios, librerías y aplicaciones. Internamente por cada máquina virtual se

virtualizan los recursos físicos de la computadora y un programa centinela llamado hipervisor se encarga de monitorear las interacciones que el usuario realice con estas virtualizaciones para identificarlas y replicarlas en los componentes físicos.

Este modelo de despliegue es uno de los más utilizados ya que permite aprovechar de mejor manera los recursos, sin embargo, presenta algunos inconvenientes:

- Alto consumo de tiempo para su configuración.
- El hipervisor consume recursos como memoria RAM.
- Cada máquina física reserva RAM y disco duro, aunque no lo utilice.
- Replicación de Kernel cuando se tienen múltiples máquinas virtuales con sistema operativo Linux.

Figura 17. **Despliegue tradicional**

Fuente: elaboración propia.

Figura 18. **Diagrama de despliegue en máquinas virtuales**

Fuente: elaboración propia.

3.2.3. Despliegue con contenedores

En este modelo de despliegue se utilizan unidades de despliegue portables y ligeras conocidas como contenedores. Estas unidades contienen tanto la aplicación como las herramientas y librerías necesarias para que esta funcione, lo que permite que funcione de la misma manera en cualquier ambiente donde sea desplegada.

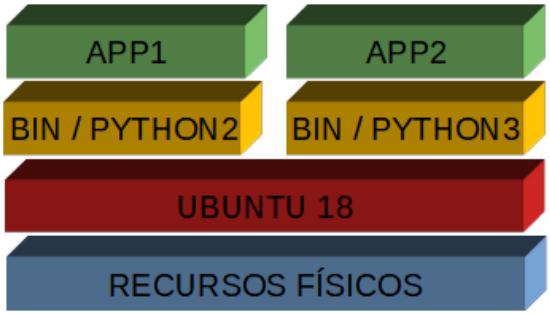
Algunas de las ventajas de utilizar contenedores para el despliegue en lugar del despliegue en máquinas virtuales son:

- Se elimina el uso de un hipervisor.
- No hay necesidad de virtualizar los componentes físicos de la computadora.

- Se optimiza el uso de los recursos computacionales al poder desplegar múltiples aplicaciones en un mismo equipo, VM o instancia.
- Estandarización del despliegue de comunicación, en cualquier lugar que se ejecute se obtendrá exactamente el mismo resultado.
- Automatización, puede ser integrado a herramientas de CI/CD.

Los contenedores se exponen de manera más extensa en el capítulo 5.

Figura 19. **Despliegue en contenedores**



Fuente: elaboración propia.

4. DEVOPS

4.1. Arquitecturas de software

En el mundo del desarrollo de software a través del tiempo se han detectado patrones comunes a la hora de diseñar sistemas informáticos, dando origen a las arquitecturas de software, las cuales permiten desarrollar de una manera más estandarizada los sistemas.

Aunque no necesariamente un sistema se tiene que regir estrictamente bajo una de estas arquitecturas, utilizarlas ofrece ventajas comprobadas. La arquitectura de un sistema puede variar según la complejidad, alcance y objetivo de la aplicación. Existe un gran número de arquitecturas de software, cada una de ellas ofrece un conjunto de ventajas a su manera.

Algunas arquitecturas de software comunes son:

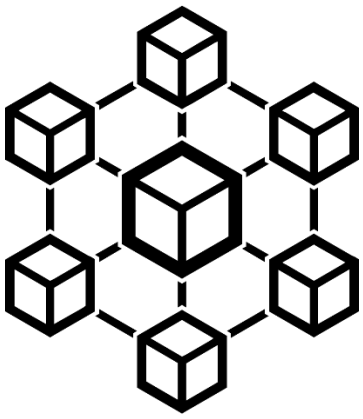
- Cliente-servidor
- Arquitectura de tres capas
- Modelo vista controlador
- Orientado a servicios (SOA)
- Microservicios (MSA)
- Dirigida por eventos
- Monolítico

De la selección de la arquitectura de software depende el buen funcionamiento de un sistema y puede condicionar la metodología que se

utilizará para desarrollarlo. Algunas arquitecturas de software introducen una alta dependencia entre componentes que no permiten un ágil desarrollo del software.

"La arquitectura de microservicios verá una rápida adopción a medida que los contenedores se conviertan en la forma estándar de implementar componentes en la nube."[15] Esto se debe a que la arquitectura de microservicios agiliza considerablemente el desarrollo de sistemas al descomponer un sistema complejo en componentes pequeños que pueden ser desarrollados y mantenidos de manera independiente, pero siendo capaces de intercambiar información con otros microservicios a través de APIs.

Figura 20. **Arquitectura orientada a microservicios**



Fuente: Think Tandem. *Arquitectura de microservicios.* https://thinktandem.io/blog/2016/09/10/let-s-get-small-introduction-to-microservices/. Consulta: abril de 2020.

[15] RAYAPATI, Viyaj. *The 2017 Clouds Trends – From DevOps to NoOps.* https://articles.microservices.com/the-2017-cloud-trends-from-devops-to-noops-1d12fa85d433.

El empleo de arquitecturas de microservicios ofrece ventajas como:

- Automatización
- Escalamiento dinámico
- Fácil adopción de nuevas tecnologías y procesos
- Agiliza el proceso de desarrollo
- Perfecto escalamiento en entornos Cloud

4.2. Metodologías de desarrollo de software

El tiempo de desarrollo de un software es parte fundamental, que puede implicar el éxito o fracaso de un proceso. Con el objetivo de lograr los objetivos se aplican metodologías de desarrollo de software que establecen la forma en que se desarrolla el software tanto desde la perspectiva administrativa como de programación. Existen dos principales tipos de metodologías:

4.2.1. Metodologías tradicionales

Las metodologías tradicionales fueron las primeras en aparecer en la historia del desarrollo de software, son completamente funcionales, pero presentan ciertos aspectos que ponen en serio peligro la culminación e incluso el éxito del sistema que se está desarrollando.

Al utilizar este tipo de metodologías se experimentan los siguientes inconvenientes:

- Cuellos de botella en el desarrollo.
    - Cuando un sistema tiene una arquitectura con alta dependencia entre componentes, distintos equipos de trabajo no podrán

desarrollar de manera simultánea. Por lo que deben esperar que una funcionalidad sea terminada para comenzar otra.

- Duración de las etapas.
  - Estas pueden abarcar semanas o meses, por lo que al intentar integrar los cambios de todos los equipos de trabajo y desarrolladores se corre un alto riesgo de que alguna funcionalidad no se integre correctamente con el resto del código, por lo que se debe emplear más tiempo en corregir errores, lo cual representa retrasar las próximas entregas.
- Tardía retroalimentación
  - En la etapa de implementación es cuando el cliente puede interactuar directamente con la versión funcional en la cual el equipo de desarrollo ha empleado semanas e inclusos meses completos desarrollando, pero si el avance no satisface por completo las necesidades de los clientes o no es lo que necesitaba (debido a la ambigüedad que ha existido y siempre existirá en el mundo de software), podría acabar en esfuerzo, recursos y tiempo desperdiciados.

Figura 21. **Cadena de comunicación en las metodologías tradicionales**

Fuente: Gurú 99. *Comunicación en metodologías tradicionales.* https://www.guru99.com/agile-vs-devops.html. Consulta: mayo de 2020.

4.2.2. Metodologías ágiles

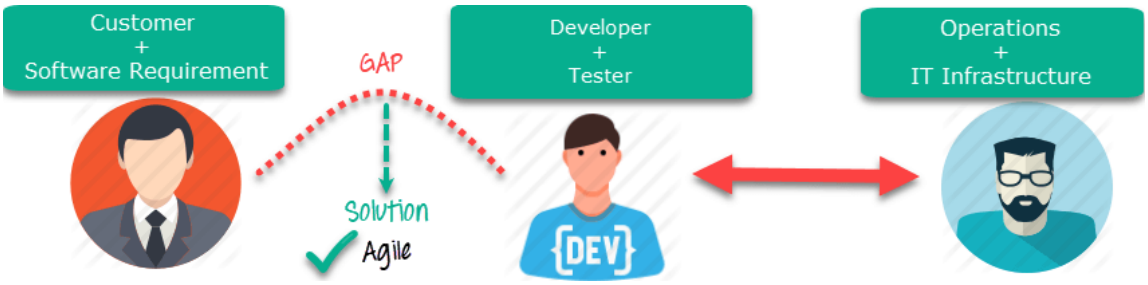
Las metodologías ágiles se enfocan en acortar los ciclos de desarrollo e impulsar la participación de las partes interesadas en el sistema para que en etapas tempranas se pueda obtener retroalimentación del sistema en desarrollo.

Esta metodología promete mejores resultados en el desarrollo de los sistemas de software porque mientras el cliente se integra más en el proceso de construcción del sistema se obtienen progresos más significativos. En metodologías ágiles:

- Los ciclos de desarrollo se miden en *sprints,* que son periodos menores a un mes.
- Se hace énfasis en:
  - Constantes cambios resultado de la constante retroalimentación.
  - Mejorar el desarrollo del software.
  - Aumentar la comunicación con las partes interesadas.

Aunque aplicando las metodologías ágiles se mejora la parte administrativa y de desarrollo del sistema, lo cual resulta acortando el tiempo de desarrollo, estos beneficios obtenidos no son aprovechados al máximo si la parte técnica para la implementación de las aplicaciones se rige bajo procesos manuales que crean cuellos de botella y hacen más lenta la entrega y despliegue de las aplicaciones. Por lo que en orden de agilizar por completo el proceso de desarrollo es imperativo utilizar herramientas de automatización que ayuden a disminuir considerablemente el tiempo de integración, entrega y despliegue de las aplicaciones.

Figura 22. **Cadena de comunicación en las metodologías ágiles**



Fuente: Gurú 99. *Comunicación en metodologías ágiles.* https://www.guru99.com/agile-vs-devops.html. Consulta: mayo de 2020.

4.3. Descripción

DevOps es una cultura de automatización que se apoya principalmente en herramientas para combinar los procesos de desarrollo y operacionales, en orden de lograr una integración, entrega y despliegue de las nuevas versiones de un sistema de manera continua.

No es una metodología de desarrollo de software, sino una cultura cuyo objetivo principal es integrar, entregar y desplegar nuevas versiones de una aplicación a los usuarios de manera periódica o en cuestión de horas a través de la automatización. La integración de DevOps con las metodologías ágiles aumenta las posibilidades de entregar proyectos de calidad y bajo los estándares y requerimientos de las partes interesadas en menor tiempo.

La aplicación de herramientas para la automatización, pilar de la cultura DevOps, permite reducir considerablemente el tiempo de implementación. Sin embargo, no aplican para todas las arquitecturas de sistemas computacionales. La arquitectura de software que mejor se acopla a la cultura de automatización

DevOps es la arquitectura de microservicios por su facilidad de desarrollo y mantenimiento independiente.

Figura 23. **Cadena de comunicación en DevOps**

Fuente: Gurú 99. *Comunicación en DevOps.* https://www.guru99.com/agile-vs-devops.html. Consulta: mayo de 2020.

Los objetivos de DevOps son:

- Mejorar la frecuencia de despliegue
- Lograr fácil salida al mercado
- Minimizar el porcentaje de fallas de nuevas versiones
- Disminuir el tiempo medio entre correcciones
- Mejorar el tiempo medio de la recuperación ante desastres

Para lograr estos objetivos DevOps se centra en CI/CD. Estos acrónimos significan:

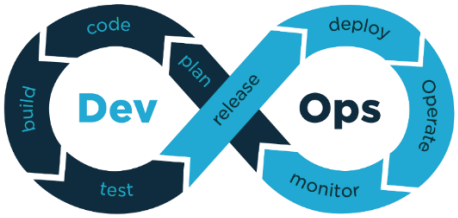
- CI.
  - Integración continua (del inglés *Continous Integration*).
    - Es el proceso de automatizar el acoplamiento de nuevo código al código existente para validar, utilizando diferentes

tipos de pruebas, que estos sean compatibles y funcionen perfectamente en conjunto.

- CD.
  - Entrega continua (del inglés *Continous Delivery*).
    - Es el proceso de preparar nuevos artefactos listos para ser desplegados en una infraestructura.
  - Despliegue continuo (del inglés *Continous Deployment*)
    - Es el proceso de desplegar los nuevos artefactos en la infraestructura de producción, para que los clientes puedan tener acceso a nuevas versiones de una aplicación de manera periódica.

CI/CD solamente se logra reemplazando los procesos manuales que requieren de interacción humana, por procesos automatizados con la ayuda de herramientas que ejecuten los mismos procesos, pero de manera autónoma.

Figura 24. **DevOps**



Fuente: Medium. *DevOps.* https://medium.com/tech-tajawal/devops-in-a-scaling-environment-9d5416ecb928. Consulta: mayo de 2020.

4.4. DevOps Tools y su relación simbiótica con Cloud Computing

"Tanto DevOps como Cloud Computing son vitales en el camino de cada organización hacia una transformación digital efectiva."[16] DevOps y Cloud Computing son dos conceptos totalmente diferentes, pero no excluyentes, en realidad ambos se complementan.

DevOps se encarga de automatizar los procesos para mejorarlos y generar valor en menor tiempo al aplicar su lema de CI/CD, mientras que Cloud Computing se refiere a tecnología y recursos computacionales como servicios. Al emplearlos en conjunto se pueden eliminar los errores humanos, agilizar el proceso de desarrollo y establecer repetibilidad.[17] La correcta aplicación de ambos conceptos da lugar a las aplicaciones nativas de la nube.

Al combinar DevOps con el paradigma Cloud se obtienen cinco ventajas específicas:

- Capacidad de sacar productos al mercado más rápido, gracias a la automatización de los procesos de desarrollo que acortan los ciclos de desarrollo.
- La automatización y la infraestructura como código reducen la complejidad de Cloud y facilitan el mantenimiento de los sistemas.
- Aumenta la fiabilidad al eliminar errores de inexactitud ya que todos los procesos son automatizados y repetibles, por lo que siempre se obtiene el mismo comportamiento.

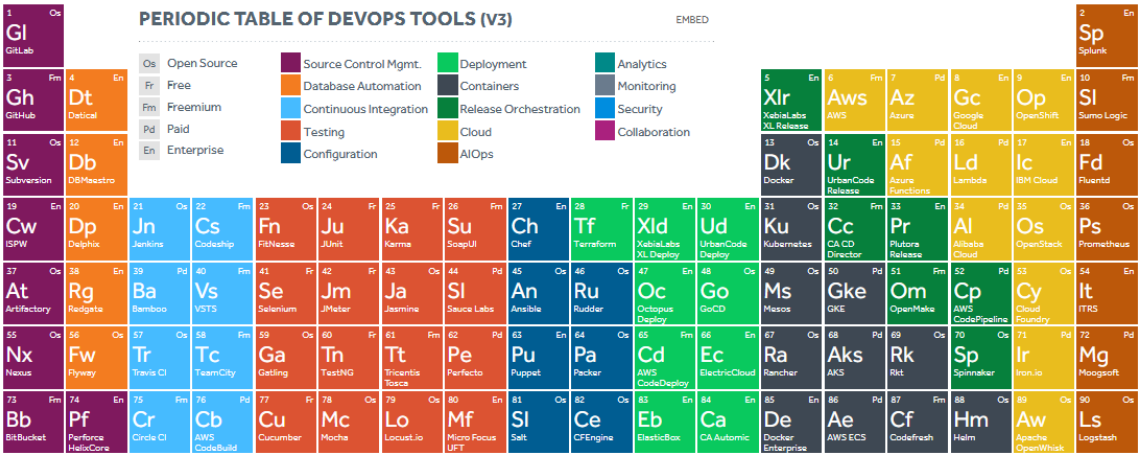
[16] Veritis. *DevOps and Cloud: A Symbiotic Relationship Aimed at Business Success.* https://devops.com/devops-and-cloud-a-symbiotic-relationship-aimed-at-business-success/.
[17] VYAS, Jaymin. *DevOps and Cloud: A symbiotic relationship*. https://devops.com/devops-and-cloud-a-symbiotic-relationship.

- Reducir el tiempo de inactividad por desastre mediante la automatización del monitoreo. Además, se aumenta la disponibilidad y la habilidad de tolerancia a fallos, por consecuencia esto aumenta la confiabilidad del negocio y la satisfacción de los clientes.
- Aumenta la escalabilidad, integrando herramientas DevOps para monitoreo, una aplicación en Cloud puede crecer o decrecer de una manera inteligente, lo cual permite experimentar una reducción en los costos de la infraestructura y aumentar su alcance global.

4.5. Herramientas DevOps

Son las siguientes:

Figura 25. **Tabla periódica de herramientas DevOps**



Fuente: Xebialabs. *Herramientas DevOps.* https://xebialabs.com/periodic-table-of-devops-tools/. Consulta: mayo de 2020.

Las herramientas DevOps se pueden catalogar según su uso y modalidad de pago (Open Source, gratuito, de paga o corporativo). Existen una gran cantidad de herramientas, algunas de ellas son:

Tabla I. **Herramientas DevOps**

Uso	Ejemplos de herramientas
Control de versiones	Gitlab Github BitBucket
Automatización de bases de datos	Datical Redgate Flyway
Integración continua	Jenkins Travis CI Circle CI
Pruebas	JUnit Selenium Cucumber
Configuración	Chef Ansible Pupet
Despliegue	Terraform ElasticBox GoCD
Contenedores	Docker Kubernetes Mesos Rancher Swarm
Orquestación de lanzamientos	Spinnaker XebiaLabs Urban Code Release
Analíticas	Kibana Datadog
Monitoreo	Nagios Zabbix Zenos
Seguridad	Sonarqube

Continuación de la tabla I.

Seguridad	Veracode BlackDuck
Colaboración	Jira Trello Slack
Plataformas Cloud	AWS Microsoft Azure Google Cloud Platform

Fuente: elaboración propia.

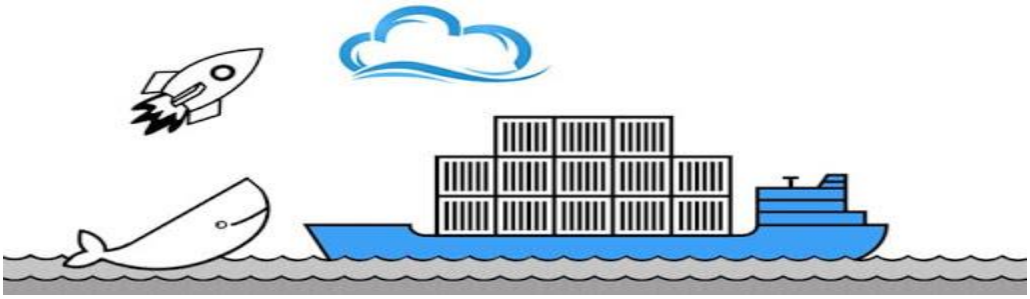
5. CONTENEDORES

5.1. ¿Qué son los contenedores?

Los contenedores son unidades de despliegue que funcionan utilizando virtualización a nivel de sistema operativo. Cada contenedor tiene su propio conjunto de procesos, sistemas de archivos, aplicaciones y librerías. Todos los contenedores residen en la misma máquina, pero en espacios diferentes y completamente aislados entre sí.

Los contenedores toman el nombre como analogía a los contenedores de carga utilizados para el transporte de productos en barcos, ya que ambos proveen una forma estandarizada de agrupar distintos tipos de contenidos, pero aislados entre sí.

Figura 26. **Contenedores**



Fuente: Virtual Tech Gurus. *Contenedores.* http://www.virtualtechgurus.com/2016/03/03/container-technology-future-cloud/. Consulta: mayo de 2020.

5.2. ¿Para qué se utilizan?

Los contenedores se utilizan para:

- Automatizar el despliegue de aplicaciones.
  - El despliegue de una aplicación utilizando la manera tradicional o máquinas virtuales es un proceso que requiere esfuerzo y consume tiempo, ambas opciones utilizan imágenes de sistemas operativos vacíos y esto atrasa el proceso de despliegue. Manualmente o utilizando software para el manejo de librerías y configuraciones (como *chef* o *puppet*) se debe preparar primero el entorno con todo lo que la aplicación necesita para funcionar correctamente y luego instalar la aplicación, mientras que los contendores ofrecen entornos previamente configurados listos para ejecutarse en cualquier lugar y en cualquier momento.

- Optimizar el uso de los recursos computacionales.
  - Los contenedores son entornos livianos. Un contenedor utiliza solamente los recursos que necesita. Por lo que se pueden desplegar muchos contenedores en un mismo equipo de cómputo (máquina física, máquina virtual o instancia de cómputo en la nube).
  - Al utilizar la virtualización a nivel de sistema operativo se elimina la necesidad de virtualizar todos los componentes físicos de la computadora tal como sucedía con las máquinas virtuales, por consecuente también se elimina el uso de hipervisores.
- Resolver conflictos de compatibilidad.
  - Cuando dos o más aplicaciones que utilizan diferentes versiones de una misma librería o trabajan con diferentes sistemas

operativos se desean desplegar en una misma máquina generan conflictos de compatibilidad que no se pueden resolver sin alterar alguna de las aplicaciones. Este conflicto se puede resolver usando contenedores para desplegar fácilmente las aplicaciones en contenedores separados, donde cada uno es un entorno con su propio sistema operativo, conjunto de binarios, librerías y aplicaciones.

- Estandarizar el despliegue de aplicaciones.
  - Los contenedores permiten estandarizar la manera en que las aplicaciones se ejecutan al empaquetar dentro del contenedor todas las configuraciones y aplicaciones necesarias para funcionar, por lo cual se pueden ejecutar en cualquier lugar, en cualquier momento, en cuestión de segundos y siempre se obtendrá el mismo resultado

5.3. ¿Cómo funcionan?

Los contenedores no son una nueva tecnología, sino que explotan las características ya existentes de aislamiento de espacios de nombre y grupos de control que ofrece el Kernel Linux. Estas características han estado disponibles desde hace mucho tiempo, sin embargo, no fue hasta hace unos cuantos años atrás que su adopción ha experimentado un crecimiento exponencial.

5.3.1. Aislamiento de espacios de nombre

Linux implementa el aislamiento de espacios de nombre para separar recursos y limitar el acceso de estos. Los contenedores implementan el aislamiento de espacios de nombre para diferentes propósitos.

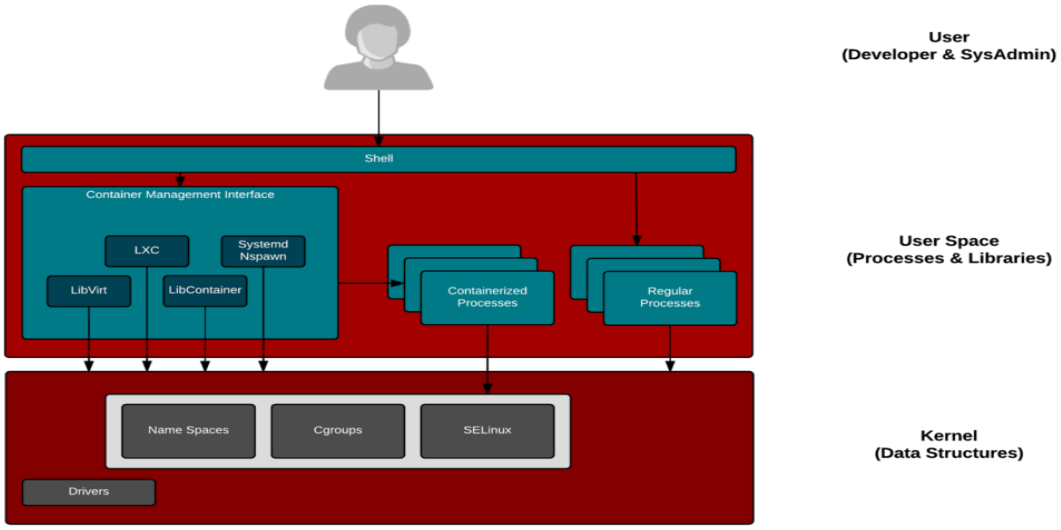
- Aislamiento de grupos de procesos.
    - Los procesos en Linux anteriormente se manejaban a través de un solo árbol de procesos con jerarquía padre e hijos. Cualquier proceso hijo con los permisos suficientes podía modificar, inspeccionar o matar a cualquier otro proceso. Esta jerarquía se logra a través de la referencia de PID entre procesos hijos y procesos padres, pero con la introducción de las características de aislamiento de grupos de nombre, aunque un proceso tenga la referencia de su proceso padre y tenga los permisos necesarios, si no reside en el mismo nombre de espacio no lo puede modificar, inspeccionar, ni eliminar ya que para este no existe.
- Aislamiento de interfaces de red.
    - Se pueden implementar espacios de nombre aislados para los distintos contenedores de manera que tengan su propia tabla de ruteos y configuraciones, requiere de la configuración de interfaces de red que abarquen múltiples espacios de nombre y luego configurar la tabla de procesos de la máquina anfitrión para que se comunique con cualquiera de estas interfaces de red. Hacerlo de forma manual es desafiante pero posible, aunque la mayoría de las organizaciones prefieren utilizar herramientas de contenedores que se encarguen de este trabajo técnico.
- Aislamiento de puntos de montaje.
    - Con el aislamiento de puntos de montaje cada uno de los contenedores tiene una perspectiva propia de la jerarquía del sistema de archivos.
- Aislamiento de procesos de intercomunicación
    - El nombre de espacio de los IPC (acrónimo del inglés *InterProcess communication*) es el responsable de aislar los procesos de intercomunicación para cada contenedor.

- Unix tiempo compartido
    - El aislamiento del UTS (acrónimo del inglés *Unix TimeSharing System*) permite que cada contenedor tenga un *hostname* y nombre de dominio independiente de otros contenedores y de la máquina anfitriona.

5.3.2. Control Groups

Los grupos de controles o Cgroups (del inglés *control groups*) permiten manejar, medir, limitar y aislar el acceso de recursos a un grupo de procesos. Con los Cgroups se puede limitar el CPU, memoria y uso del disco duro de cada contenedor de ser necesario.

Figura 27. **Vista general del funcionamiento de los contenedores**



Fuente: Openshift. *Funcionamiento de los contenedores.* https://www.openshift.com/blog/containers-are-linux. Consulta: mayo de 2020.

5.4. Principales herramientas de contenedores

El desarrollo de la tecnología no es nuevo, ni propio de alguna empresa privada, esta tecnología explota al máximo la capacidad del Kernel Linux de aislar recursos a través de espacios de nombre y limitarlos utilizando Cgroups. En el mercado existe un gran número de herramientas de contenedores, pero se diferencia entre sí según la arquitectura que implementa cada una para ofrecer mayor seguridad y mejor funcionamiento. Algunas de estas son:

Tabla II. **Herramientas de contenedores**

Herramienta	Mantenido por
Docker	Docker, inc.
rkt	CoreOs
LXC	Canonical Ltd
OpenVz	Virtuozzo
Containerd	Containerd.io

Fuente: elaboración propia.

5.5. Razón de selección Docker

Se presenta a continuación:

Figura 28. **Docker**

Fuente: Docker. *Logo de Docker.* https://www.docker.com/company/newsroom/media-resources. Consulta: mayo de 2020.

La utilización de Docker como piedra angular del conjunto de herramientas de administración de software ahorra tiempo y permite centrarse en actividades de alto valor, porque Docker minimiza el tiempo que se dedicará a realizar tareas cotidianas.[18]

Docker es una de las muchas tecnologías de contenedores que utilizan la virtualización a nivel de sistema operativo para empaquetar y desplegar software de una manera óptima y más sencilla. Sin embargo, por mucho Docker es la herramienta de contenedores líder en la industria (véase anexo 3), su posicionamiento es tan firme que incluso al hablar de contendores la mayoría de las personas lo asocian rápidamente a Docker. Su éxito se debe a:

- Es una plataforma Open Source
  - Es gratuito y puede ser utilizado por cualquiera, en cualquier entorno.
  - Tiene amplio soporte e incluso muchas empresas invierten para mejorar cada vez más el funcionamiento de los contenedores Docker.
- Posee un amplio registro público de imágenes de contenedores.
  - En el registro público DockerHub existe un gran número de aplicaciones, desde sistemas operativos para ser utilizados como cajas de arena (*sandbox*) e incluso hasta pilas complejas de aplicaciones.
  - Cualquier usuario tiene acceso a imágenes.
    - Certificadas por Docker Enterprise.
    - Publicados y mantenidos por entidades comerciales (Oracle, IBM, entre otros).
    - Creadas por cualquier usuario del registro.

[18] NICKLOFF, Jeff. *Docker in action.* p. 3

- Cualquier usuario puede subir y versionar imágenes de contenedores propias.

- Abstrae la implementación y manejo del aislamiento de espacios de nombre para:
  - PID: aislar los identificadores de procesos.
  - Red: configurar y manejar las direcciones IP, tablas de ruteos, entre otros.
  - Puntos de montaje: provee un punto de vista específico del almacenamiento.
  - IPC: manejar todos los procesos de intercomunicación.
  - UTS: le permite a cada contener tener su propio *hostname* y nombre de dominio.
- Abstrae la configuración y manejo de cgroups.
  - Por defecto Docker ajusta los cgroups según los procesos del contenedor para que sean livianos, pero de ser necesario pueden configurarse con cantidades de recursos específicas.
- Permite construir, transportar y ejecutar cualquier aplicación en cualquier lado.

5.5.1. Arquitectura Docker

Docker funciona con una arquitectura cliente-servidor. El servidor es un proceso conocido como demonio Docker que se ejecuta en un equipo de cómputo (maquina física, máquina virtual o instancia en cualquier proveedor de servicios) y el cliente a través de una interfaz de línea de comandos (CLI) puede comunicarse con el servidor a través del programa Docker para crear, editar o eliminar contenedores e imágenes de contenedores.

5.5.1.1. Imagen

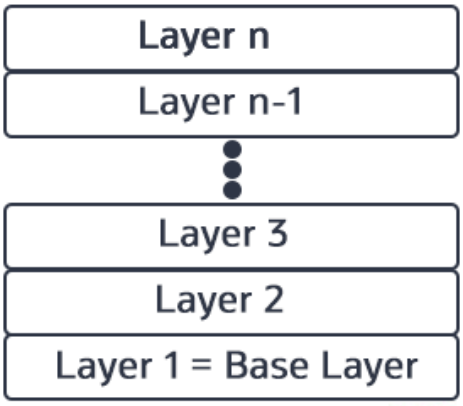
Las imágenes Docker son plantillas que se utilizan para empaquetar aplicaciones junto a todos los binarios y librerías que esta necesita para funcionar. Las imágenes de contenedores pueden ser desde sistemas operativos limpios para utilizar como cajas de arena hasta sofisticadas pilas de aplicaciones listas para ser ejecutadas.

Las imágenes Docker son archivos que se componen de múltiples capas para la optimización de recursos. Cada una de las capas representa una imagen independiente que cuenta con su propio conjunto de binarios y librerías, además posee una referencia a capas padres que tienen parte de las configuraciones que la imagen final necesita para que la aplicación pueda funcionar.

La unión de todas las capas da como resultado una imagen con un superconjunto de binarios y librerías producto de la unión de todas esas capas, esto permite ahorrar espacio de almacenamiento ya que, si varias imágenes tienen en común un conjunto de binarios y librerías, en lugar de duplicarlas y consumir el doble de espacio de almacenamiento se crea solamente una imagen padre de la cual muchas imágenes podrán hacer referencia. La capa más baja se conoce como capa base.

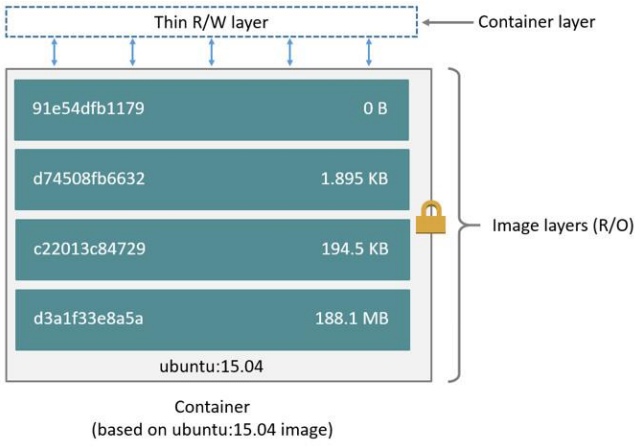
Cada una de las capas de una imagen solamente son de lectura y escritura. No se pueden realizar cambios sobre ninguna capa, lo cual permite que cualquier contenedor creado a partir de la misma imagen tenga siempre el mismo comportamiento. Además, al ser un archivo, se puede versionar e incrustar en pipelines de CI/CD a través de *scripts* de automatización.

Figura 29. **Capas Docker**



Fuente: Codenuclear. *Capas Docker.* https://codenuclear.com/what-are-docker-images/. Consulta: junio de 2020

Figura 30. **Imagen Docker**



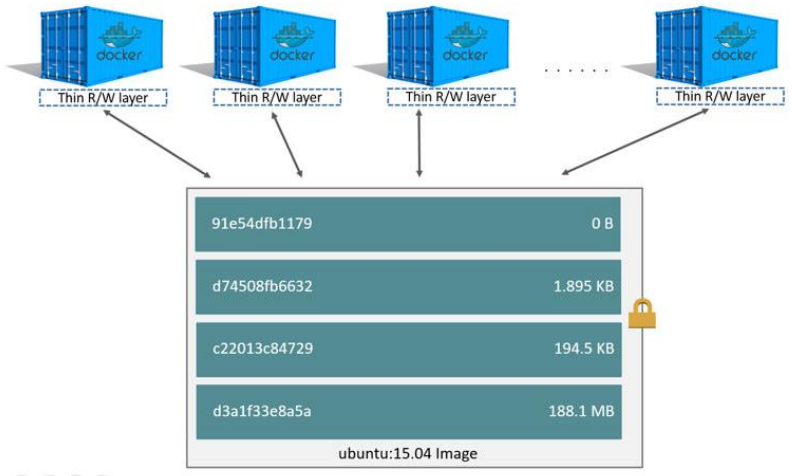
Fuente: Docker *Imagen Docker.* https://docs.docker.com/storage/storagedriver/. Consulta: junio de 2020.

5.5.1.2. Contenedor

Cuando se crea un contenedor a partir de una imagen, en realidad se crea una capa de lectura/escritura sobre las demás capas de lectura de la imagen. En esta capa se almacenan todos los cambios que se realicen en el contenedor, tal como nuevas aplicaciones y librerías o datos generados. Todas las capas padres sirven como referencia para establecer el comportamiento del contenedor.

De una imagen se pueden crear tantos contenedores como se necesiten. Cada contenedor creará su propia capa de lectura/escritura y tendrá asignado recursos en diferentes espacios de nombre, por lo que cada uno se considera como un entorno aislado.

Figura 31. **Contenedores Docker**



Fuente: Docker *Contenedores Docker.* https://docs.docker.com/storage/storagedriver/. Consulta: junio de 2020.

5.5.1.3. Registro

Plataforma que permite manejar un control de versiones de imágenes. Puede ser:

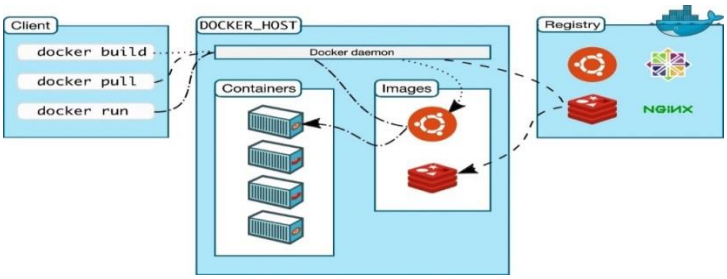
- Público: básicos y simples de usar, pueden funcionar perfectamente para individuos y equipos pequeños. Ejemplo: Docker Hub.
- Privado: útil para equipos de gran tamaño ya que permite el establecimiento de límites basados en roles para obtener mayor seguridad y gobernanza. Ejemplo: AWS Elastic Container Registry (ECR).

5.5.1.4. Demonio Docker

Este es el proceso que se ejecuta en segundo plano, funciona como servidor. El usuario lleva a cabo sus acciones utilizando comandos Docker y el demonio es quien se encarga de manejar y crear los contenedores e imágenes. Además, es el encargado de comunicarse con el registro de imágenes para descargar o subir nuevas versiones de imágenes.

Empleando contenedores Docker se puede alcanzar altos niveles de agilidad y eficiencia al separar el proceso de empaquetamiento de software del proceso de despliegue de estos. Se pueden establecer etapas dentro del pipeline de CI/CD para el empaquetamiento de software generando nuevas imágenes de contenedores que representan artefactos listos para implementar. En etapas posteriores se puedan desplegar dichos artefactos directamente sobre el motor de contenedores en una máquina o utilizando una herramienta de orquestación de contenedores como Kubernetes, Apache Mesos, Docker Swarm, entre otras.

Figura 32. **Arquitectura Docker**



Fuente: Arquitectoit. *Arquitectura Docker.* http://www.arquitectoit.com/docker/arquitectura-docker/. Consulta: mayo de 2020.

5.5.2. Ventajas

- Portabilidad.
  - Solamente necesita el motor de contenedores para funcionar, elimina la necesidad de utilizar herramientas para el manejo de librerías y configuraciones necesarias en despliegues tradicionales y en máquinas virtuales.
- Automatización.
  - Se pueden crear, manejar y eliminar los contenedores utilizando *scripts*, por lo que se pueden integrar sin ningún problema a pipelines de CI/CD.
- Modularidad.
  - Se puede emplear en arquitecturas de microservicios donde la lógica es dividida de manera granular, creando independencia entre contenedores, consecuentemente agiliza el proceso tanto de desarrollo como de despliegue.
- Capas y control de versiones de imágenes.
  - Las imágenes son archivos compuestos por múltiples capas.

    - Al componerse de capas se puede optimizar el uso de espacio de almacenamiento, ya que múltiples imágenes pueden hacer referencia a una misma imagen padre, por lo que se evita la duplicación al máximo.
    - Al ser un archivo, puede ser versionado a través de un registro de imágenes ya sea público o privado.
- *Rollback*
    - Al ser versionado se pueden regresar sin mucho esfuerzo a versiones anteriores del sistema.
- Despliegues rápidos.
    - Un contenedor ejecuta sus procesos en el sistema operativo anfitrión, por lo que se elimina la espera del inicio de un sistema operativo experimentado en las máquinas virtuales, por lo que permite despliegues en cuestión de minutos e incluso segundos.
- Eficiencia de recursos.
    - Por defecto los contenedores Docker solamente utilizan los recursos que necesitan.
    - Se elimina la duplicación de las capas.
- Independencia de plataforma.
    - Un contenedor se genera a partir de una imagen que es la que contiene todas las configuraciones, por lo que en cualquier lugar que se ejecute se obtendrá siempre los mismos resultados.
- Compartición efectiva de recursos.
    - Docker se encarga de implementar y manejar el aislamiento de espacios de nombre, por lo que no existe el riesgo de que los procesos de un contenedor afecten a otro ni a la máquina física.
- Entornos predecibles.
    - Se pueden crear imágenes con pilas complejas de aplicaciones preinstaladas, y al ejecutarlos siempre se tendrá el mismo

resultado ya que todos los cambios en un contenedor se realizan sobre la capa de escritura y lectura sin afectar a las capas de la imagen.

- Escalamiento suave e inteligente.
  - Utilizando contenedores una aplicación puede escalar horizontalmente para responder a picos de actividad. En lugar de aumentar la capacidad de los contenedores, se lanzan más contenedores y mediante la implementación de balanceadores de carga y sistemas de autodescubrimiento de servicios la carga de trabajo se divide entre todos los contenedores.
- Optimización de costos.
  - El escalamiento horizontal creciente y decreciente permite desplegar solamente los contenedores necesarios con base en cargas de trabajo o parámetros establecidos optimizando los costos operativos.
  - Es parte fundamental también en servicios Cloud tal como:
    - PaaS:
      - ✓ Este servicio Cloud se encarga de abstraer la configuración de las instancias requeridas para ejecutar una aplicación. Internamente el proveedor de servicios se encarga de poner en funcionamiento la aplicación en un contenedor y realiza las configuraciones necesarias para exponerla.
      - ✓ Empleando contenedores se pueden establecer periodos de inactividad para pagar directamente por la carga de trabajo recibida y tiempo útil.
    - *Serverless*
      - ✓ Con la llegada de los dispositivos IoT se necesitará almacenar y analizar grandes cantidades de datos en

tiempo real. Se pueden emplear contenedores para ejecutar funciones con propósito específico y una vez completadas se elimina. Por lo cual los usuarios pagarán por la cantidad de segundo o fracción de segundo que utilizó el poder de cómputo.

- ✓ Al usar contenedores se pueden evitar los cuellos de botella que se podrían experimentar cuando numerosos dispositivos requieran poder de cómputo para el almacenamiento o análisis de datos, ya que cada función utiliza específicamente un contenedor, por lo que se tendría un rendimiento óptimo.

5.5.3. Desventajas

Aunque los contenedores parecieran la solución definitiva para los problemas de despliegues y la optimización tanto de costos como de recursos computacionales, al igual que las demás tecnologías, posee ciertas desventajas que merecen ser consideradas antes de adoptarlos.

- La seguridad en contenedores es una piedra en el camino para la completa adopción de esta tecnología, debido a que al ejecutarse directamente sobre el Kernel de la computadora anfitrión existe la posibilidad de la existencia de alguna vulnerabilidad que pueda exponer el sistema donde se están ejecutando los contenedores.
- No todas las aplicaciones pueden ser desplegadas con arquitecturas orientadas a microservicios, por lo que no pueden emplear contenedores.
- La persistencia de datos es complicada. Los contenedores son volátiles, por lo que toda la información que reside en este se pierde al apagarlo o reiniciarlo.

- Fue diseñado para ejecutar servidores de aplicaciones que no necesitan interfaces gráficas.
- Optimiza el uso de recursos, pero no alcanza las velocidades de ejecutar las aplicaciones en bare-metal por la capa extra de administración del motor de contenedores.

5.6. Impacto en el mundo de la arquitectura de software e infraestructura computacional.

La migración a Cloud Computing es inminente, pero erróneamente se puede pensar que el migrar a Cloud consiste en pasar una aplicación tradicional de un centro de datos *on-premise* a la infraestructura de cualquier proveedor de servicios, es decir, seguir haciendo lo mismo, pero ahora en la nube. Sin embargo, las empresas tendrán que detenerse a rediseñar sus aplicaciones tradicionales para convertirlas en aplicaciones nativas de la nube, y así sacar el máximo provecho a Cloud Computing. Las arquitecturas de microservicios contenerizadas junto a las APIs y los equipos DevOps son la base para las aplicaciones nativas de la nube.

Los contenedores han cambiado la forma de realizar negocios y como los desarrolladores de software trabajan. En la actualidad los desarrolladores ya no invierten tiempo en las configuraciones necesarias para la correcta implementación de las nuevas versiones de un software en los distintos ambientes, sino que aprovechan las facilidades que ofrecen los contenedores.

Además, empleando contenedores se puede aprovechar la impresionante flexibilidad de escalamiento en la nube, esta escalabilidad se realiza en número y no en recursos. En lugar de crecer verticalmente y darle más recurso a un solo contenedor para satisfacer toda la carga de trabajo, se crece

horizontalmente para que múltiples contenedores se distribuyan el trabajo y juntas soporten la completa carga de trabajo. Una vez la carga de trabajo disminuye, algunos contenedores son eliminados porque terminaron de cumplir con su propósito y dejarlos funcionando no es óptimo en términos de costos.

Esta naturaleza descartable de los contenedores también permite que ante cualquier fallo o infección con algún programa maligno pueden ser eliminados y reemplazados por otros completamente nuevos disminuyendo el tiempo en que un microservicio está fuera de funcionamiento.

> Es increíble que a pesar de que la tecnología de contenedores ha existido desde hace mucho tiempo, no fue hasta hace unos cuantos años que su adopción ha crecido a tal manera que en la actualidad se le considera como la forma estándar de resolver uno de los aspectos más costosos en el desarrollo de software: el despliegue.[19]

El mundo desde 2016 se mueve a una completa adopción de contenedores[20], lo cual a su vez ha impulsado al empleo de la arquitectura de microservicios. Ha sido tanto el impacto de los contenedores que inclusive grandes proveedores de servicios han cambiado su manera de virtualizar sus servicios, sustituyendo las máquinas virtuales por contenedores para ejecutar aplicaciones nativas Cloud. Algunos de ellos son:

- Google
- IBM/Softlayer
- Joyent

[19] MIELL, Ian; HOBSON SAYERS, Aidan. *Docker in practice*. p. 3

[20] RAYAPATI, Viyaj. *The 2017 Clouds Trends – From DevOps to NoOps.* https://articles.microservices.com/the-2017-cloud-trends-from-devops-to-noops-1d12fa85d433.

“A medida que las empresas adopten la nube como la plataforma de infraestructura principal, impulsará el auge del empleo de contenedores para manejar la diversidad y variedad de dependencias entre las aplicaciones empresariales de las empresas.”[21] Este aumento de contenedores requiere de herramientas con altos niveles de abstracción que permitan definir y manejar los contenedores de una manera más sencilla. Estas herramientas se conocen como herramientas de orquestación de contenedores.

5.7. Panorama actual

Cualquiera que aprenda cómo los contenedores Linux proveen aislamiento para ejecutar programas y cómo usar Docker para controlar ese aislamiento puede lograr asombrosas hazañas de reutilización, eficiencia de recursos y simplificación de sistemas.[22]

La migración a Cloud Computing está introduciendo la necesidad de refactorizar aplicaciones tradicionales en aplicaciones nativas, por lo que el empleo de contenedores va en aumento, de la mano con la adopción de la cultura DevOps, esto está creando un mercado que hasta el momento no se ha podido satisfacer por completo por la falta de personas capacitadas. Es necesario que los estudiantes egresados de la Universidad de San Carlos de Guatemala desarrollen habilidades en las herramientas de contenedores, el cual es solamente un punto en el vasto horizonte de herramientas DevOps, sin embargo, uno de los más cruciales.

[21] RAYAPATI, Viyaj. *The 2017 Clouds Trends – From DevOps to NoOps.* https://articles.microservices.com/the-2017-cloud-trends-from-devops-to-noops-1d12fa85d433.
[22] NICKLOFF, Jeff. *Docker in action*. p. 7

5.8. Comportamiento esperado

Según Mishra, se aproxima el auge de los contenedores, se espera que los contenedores no sean considerados más como una herramienta, sino que se conviertan en la norma de cómo desplegar aplicaciones.

Gartner pronostica que para 2023 más del 70 % de las organizaciones globales estará corriendo más de dos aplicaciones contenerizadas en producción.

Por otro lado, IDC pronostica que 95 % de nuevos microservicios serán desplegados en contenedores para 2021. Algunas tecnologías emergentes que aumentarán el uso de contenedores en los próximos años son:

- Plataformas para funciones *serverless*
- Servicios *mesh* para la comunicación entre contenedores
- Analíticas
- *Machine Learning*

Además de las ventajas de los contenedores en el área de despliegue, estos han sido un aliado clave para el desarrollo ágil en la nube. Es por ello por lo que alrededor del mundo cientos de millones de dólares han sido invertidos en docenas de emprendimientos de tecnología de contenedores que están construyendo herramientas de orquestación, administración, redes y seguridad para el ecosistema de contenedores.

Los contenedores son una tecnología que aprovecha al máximo Cloud Computing, ante esta demanda todos los proveedores de servicio han diseñado servicios específicos que permiten al usuario desplegar contenedores sin

preocuparse directamente por temas de servidores. Delegando esa tarea al proveedor de servicio, el cliente solamente paga por el tráfico, almacenamiento y uso de recursos computacionales de los contenedores.

6. ORQUESTACIÓN DE CONTENEDORES

6.1. ¿Qué son las herramientas para la orquestación de contenedores?

Los contenedores son piezas clave para el movimiento DevOps, han cambiado de manera dramática la forma en que las organizaciones construyen, transportan y mantienen sus aplicaciones. Conforme aumenta la complejidad de un sistema con arquitectura de microservicios contenerizado aumenta también el número de contenedores que se requieren ejecutar simultáneamente, ya sea por la cantidad de microservicios que lo conforman o por necesidades de replicación.

> Las tecnologías de contenedores han revolucionado el desarrollo de software. Pero, a medida que su uso vaya más allá de un sandbox, compañías necesitan encontrar la forma de sacar ventaja del poder de los contendores sin ser retenidos por procesos manuales no escalables[23].

Manejar manualmente el ciclo de vida de un alto número de contenedores es completamente ineficiente. La escalabilidad y alta disponibilidad se convierte en una tarea desafiante en estos escenarios, razón por la cual se emplean herramientas de orquestación de contenedores que se encargan de manejar el ciclo de vida de los contenedores y además permiten integrarlas a los flujos de trabajo de CI/CD.

Empleando herramientas para la orquestación de contenedores se sustituyen todos los procesos manuales por *scripts,* para automatizar estos

[23] STROUD, Robert. *Containers myths vs facts*. https://digital.ai/catalyst-blog/containers-myths-vs-facts.

procesos, lo cual ahorra tiempo y por consecuente dinero. Por lo tanto, el empleo de herramientas para la orquestación de contenedores se ha convertido en parte esencial de las organizaciones que tienen aplicaciones con arquitectura de microservicios.

Las herramientas de orquestación de contenedores no manejan directamente los contenedores, sino que implementan potentes abstracciones que hacen posible el manejo y despliegue de un alto número de contenedores de una manera sencilla.

6.2. ¿Para qué se utilizan?

Las herramientas de orquestación de contenedores proveen abstracciones potentes que se encargan de:

- Aprovisionar y desplegar contenedores.
  - Utilizan unidades de despliegue en las cuales se ejecutan los contenedores. El usuario se encarga de definir los atributos de cada una de estas unidades de despliegue y es tarea del orquestador decidir en qué *host* del clúster ubicarlo.
- Redundancia.
  - Permite desplegar la cantidad necesaria de contenedores de un mismo tipo para tener un rendimiento óptimo.
- Alta disponibilidad de contenedores.
  - Emplea mecanismos para monitorear la salud de los contenedores y *hosts* para asegurar que la cantidad de contenedores requeridos se cumpla en todo momento.
- Escalamiento dinámico.

- Se pueden crecer horizontalmente y agregar contenedores para satisfacer una carga de trabajo y removerlos para optimizar costos.

- Balanceo de carga interno a través del autodescubrimiento de servicios.
  - Implementa mecanismos internos para identificar automáticamente los contenedores nuevos que se vayan agregando en todo el clúster para dividir y redirigir la carga de trabajo a cada uno de ellos y soportar en conjunto la carga completa de trabajo.
- Propagación de microservicios a través de un sistema distribuido.
  - Un servicio puede tener desplegadas réplicas en distintos nodos del clúster.

6.3. ¿Cómo funcionan?

Las herramientas de orquestación de contenedores, a pesar de que su tarea principal es ejecutar un alto número de contendores, monitorearlos y controlarlos, no trabajan directamente con contenedores, sino con abstracciones de más alto nivel a las cuales se les delega la monitorización y el completo control de los contenedores, el objetivo es evitar al máximo la interacción humana producto de automatizar los proceso.

El concepto de cualquiera de los orquestadores de contenedores es el mismo, proveer abstracciones de alto nivel para desplegar y manejar grandes cantidades de contenedores de manera simultánea de una forma más sencilla y confiable, sin embargo, no hay un estándar definido, por lo que cada herramienta tiene su propia arquitectura además de su propia manera de definir y manejar los contenedores. Al menos la mayoría de las herramientas para la

orquestación de contenedores utilizan archivos de configuración tales como YAML o JSON para la definición de sus abstracciones.

Pero en general implementan unidades minimizadas de despliegue de los contenedores para monitorearlos y mecanismos de autodescubrimiento de servicios para mantener el rastro de los distintos contenedores, ya que estos se van creando y eliminando de manera dinámica, para compartir equitativamente las cargas de trabajo entre microservicios iguales.

6.4. Principales herramientas para la orquestación de contenedores

Existen múltiples herramientas para la orquestación de contenedores, algunas son *Open Source* y otras propias de algunos proveedores de servicios, algunas de las principales herramientas para la orquestación de contenedores en el mercado son:

- *Open Source*
  - *Kubernetes*
  - *Docker Swarm*
  - *Apache Mesos*
- Modelo de servicio de proveedores de servicio
  - Amazon Elastic Container Service (Amazon ECS)
- Kubernetes clúster en proveedores de servicio
  - Google Kubernetes Engine (GKE)
  - Azure Kubernetes Service (AKS)
  - Amazon Elastik Kubernetes Service (Amazon EKS)
  - AWS EKS Fargate (Serverless)

6.5. Razón de estudio de Kubernetes

Inicialmente fue desarrollado por Google, pero desde 2015 se ha convertido en el proyecto emblema de la Fundación Cloud Native Computing (CNCF, acrónimo en inglés). Kubernetes es un *framework* de orquestación de orquestación de contenedores *open-source* y portable. Es respaldada por importantes empresas tales como: Google, Amazon Web Services (AWS), Microsoft, IBM, Intel, Cisco y RedHat.

Kubernetes permite manejar de una forma fácil y eficiente clúster compuestos de máquinas físicas, virtuales e instancias de la nube, para ejecutar contenedores Linux. Utilizando Kubernetes se pueden eliminar todos los procesos manuales necesarios para desplegar y escalar aplicaciones contenerizadas.

Un clúster de Kubernetes puede tener nodos de diferentes tipos de nubes, públicas, privadas o hibridas. Por esta razón es una plataforma ideal para las aplicaciones nativas de la nube que requiere un rápido escalamiento.

Los recursos de Kubernetes son definidos a través de archivos de configuración escritos en formato JSON o YAML. Por lo que las configuraciones son extremadamente portables, los archivos de configuración para el despliegue de aplicaciones contenerizadas permanecen intactos sin importar el proveedor de servicio en el que se esté desplegado el clúster de Kubernetes. Estos archivos de configuración se cargan y ejecutan en el clúster utilizando la interfaz de línea de comandos Kubectl.

Kubernetes es muy poderoso y automatiza manejo, despliegue y escalamiento de contenedores de una manera inteligente y organizada, permite

mover cargas de trabajo entre proveedores de servicio sin necesidad de rediseñar las aplicaciones o reinventar la infraestructura. Esto ayuda a estandarizar y evitar la dependencia de proveedor, lo cual será aprovechado en los próximos años con la adopción de estrategias multi-cloud (véase anexo1).

Figura 33. **Kubernetes logo**

Fuente: Kubernetes*. Logo de Kubernetes.* https://en.wikipedia.org/wiki/Kubernetes. Consulta: junio de 2020.

6.6. Componentes principales de Kubernetes

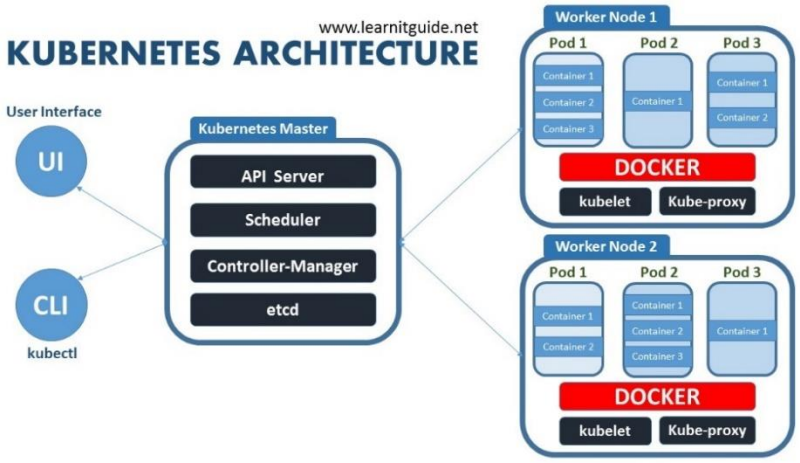
- Clúster: se compone por *control plane* y múltiples *worker nodes.*
- *Control plane:* este es el centro de control de Kubernetes, aquí se ejecutan los procesos que se necesitan para tomar control tanto de los nodos como de los objetos del clúster. Los procesos internos del *control plane* son:
    - *Scheduler.*
        - Se encarga de ubicar en el nodo más conveniente a los *pods.*
    - *Api-Server.*
        - Es el único punto de entrada a un clúster de Kubernetes, normalmente es accesible a través de una interfaz de línea

de comandos (CLI, del inglés: *command line interface*), el más común es kubectl.

- *Control manager.*
  - Contiene controladores para los distintos tipos de objetos que se pueden crear en el clúster.
- Etcd.
  - Es la base de datos donde se almacena la información de todos los objetos creados en el clúster.

- *Worker nodes:* son todos los nodos trabajadores donde se despliegan aplicaciones, los procesos internos de los *worker nodes* son:
  - *Container runtime.*
    - Kubernetes permite la ejecución de diferentes motores de contenedores, aunque por defecto utiliza Docker.
  - Kubelet.
    - Es el proceso que recibe las instrucciones del *control plane* para crear, modificar o eliminar algún *pod* que exista dentro del *worker node.*
  - *Kube proxy*
    - Sirve para establecer y manejar la red dentro del *worker node.*

Figura 34. **Arquitectura clúster de Kubernetes**



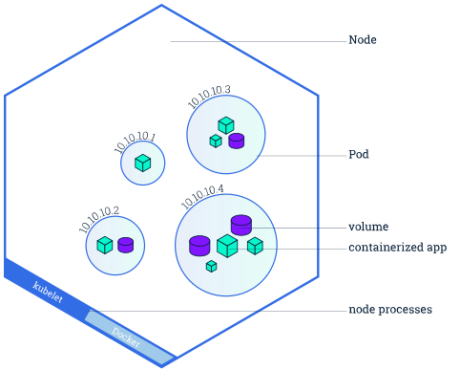
Fuente: Learnitguide. *Arquitectura de Kubernetes.* https://www.learnitguide.net/2018/08/what-is-kubernetes-learn-kubernetes.html. Consulta: mayo de 2020.

- *Pod:*
  - Es la unidad base de despliegue en Kubernetes, consiste en un grupo de uno o más contenedores.
  - Los contenedores se agrupan en *pods* para evitar la alteración de estos al desplegarlos en Kubernetes, ya que para el monitoreo se necesitan determinadas herramientas y configuraciones complementarias.
- *Service:*
  - Son la forma en que se pueden acceder a los microservicios desplegados en un clúster de Kubernetes.
  - Implementa mecanismos para el autodescubrimiento de servicios a través de la implementación de selectores que buscan *pods* con determinadas etiquetas para distribuir la carga de trabajo.

        - *ClusterIP:* expone un microservicio en determinado puerto. Solamente puede ser alcanzado por microservicios que residan en el mismo clúster.
        - *NodePort:* expone un microservicio en determinado puerto. Puede ser alcanzado desde el mundo exterior a través de la ip del nodo donde está desplegado junto al puerto asignado por Kubernetes. Su uso se recomienda solamente para ambientes de desarrollo.
        - *LoadBalancer:* crea un servicio de tipo balanceador de carga según el proveedor de servicios donde el cléster esté configurado. Genera un *endpoint* por el cual el mundo exterior puede alcanzar al microservicio.
- Deployment, ReplicaSet:
    - Manejar los *pods* individualmente nos regresa a la forma tradicional de desplegar contenedores utilizando un motor de contenedores, la escalabilidad y alta disponibilidad representan un gran desafío. Por lo que se implementan abstracciones que se encarguen de la replicación y *self-healing* de manera autónoma.
    - Los *ReplicaSet* son abstracciones encargadas de monitorear el estado de los *pods* y verificar que en todo momento el estado de estos coincida con el estado deseado declarado por el usuario. Estos objetos solamente pueden monitorear, en caso de una actualización a los *pod* deberán eliminarse y crearse de nuevo.
    - Los *deployments* son abstracciones más potentes que declaran y manejan *ReplicaSets,* pero además implementan estrategias de despliegue ante nuevas actualizaciones de un *pod.* Esto lo logran disminuyendo gradualmente el *ReplicaSet* existente y creando uno nuevo que va aumentando también gradualmente hasta lograr una

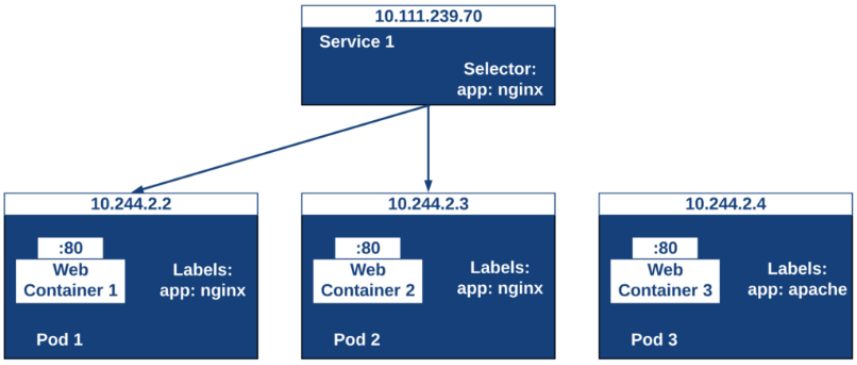
actualización completa, de manera que en ningún momento el microservicio está inactivo.

Figura 35. **Pod**



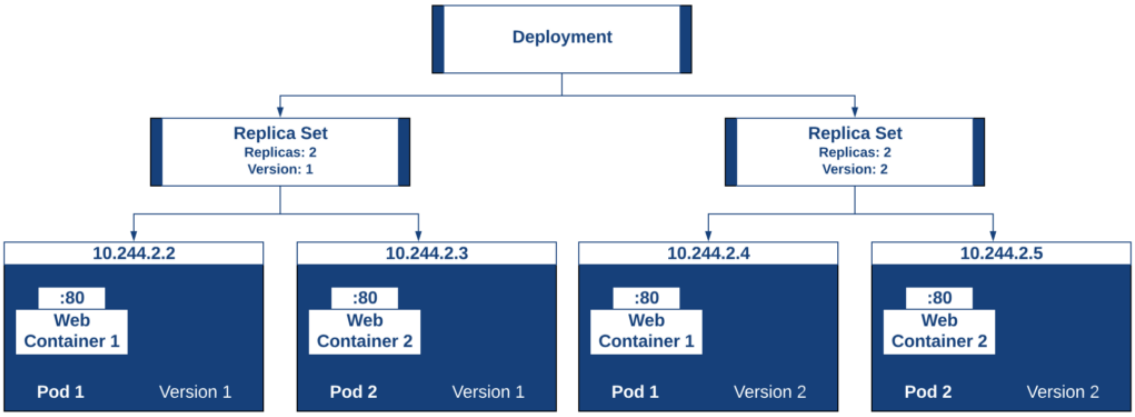
Fuente: Kubernetes. *Pods.* https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/. Consulta: junio de 2020.

Figura 36. **Service**



Fuente: Theithollow. *Service*. https://theithollow.com/2019/01/31/kubernetes-services-and-labels/. Consulta: junio de 2020.

Figura 37. **Deployment**



Fuente: Theithollow. *Deployment.* https://theithollow.com/2019/01/30/kubernetes-deployments/. Consulta: junio de 2020.

6.7. Impacto en el mundo de la arquitectura de software e infraestructura computacional

En su mayoría, las aplicaciones con arquitecturas de microservicios contenerizadas emplean un alto número de contenedores, razón por la cual los proveedores de servicios han introducido nuevos modelos de servicios para liberar al usuario del manejo y mantenimiento de servicios de nivel inferior requeridos para lograr la orquestación de estos.

6.7.1. CaaS

Los contenedores necesitan instancias de cómputo para ejecutarse, por lo que en lugar de que el usuario tenga que aprovisionar estas máquinas y configurarlas manualmente los proveedores de servicio han empezado a ofrecer soluciones donde se abstrae la configuración, creación y manejo de estas instancias que se necesitan para ejecutar las aplicaciones contenerizadas, por

lo que el usuario solamente parametriza cuál es el comportamiento esperado. Algunas de las soluciones CaaS existentes en el mercado son:

- Amazon - Elastic Container Service (ECS)
- Google Cloud - Google Container Engine (GCE)
- Azure - Azure Container Service (ACS)

Sin embargo, el usuario sigue pagando por el tiempo que están activas sus instancias, aunque no las aproveche al máximo. AWS se ha aventurado un poco más allá y ha desarrollado un servicio de tipo CaaS donde el usuario se olvida por completo de los servidores donde los contenedores se ejecutarán y solamente paga por el poder de procesamiento utilizado por sus contenedores, a este servicio AWS se le conoce como Fargate y se le considera como Serverless.

6.7.2. KaaS

Kubenetes ha tenido un éxito rotundo posicionándose actualmente como la herramienta estándar para la orquestación de contenedores. Ante esta demanda varios proveedores Cloud han comenzado a ofrecer como servicio el manejo de clústeres de Kubernetes.

Además del éxito de Kubernetes, existen otras razones que motivaron a los proveedores Cloud a ofrecer clústeres de Kubernetes como servicio:

- El proceso para crear manualmente un clúster de Kubernetes es complicado.
- Existen muchas formas en la que se puede instalar y configurar el clúster, no todas son óptimas.

- El proceso de configuración de los nodos es monótono y consume mucho tiempo.
- En un clúster manejado por el usuario, este debe implementar una solución para la replicación del *control plane*, ya que es el único punto de acceso al clúster, por lo que si se pierde, se pierde por completo el acceso y control de los recursos desplegados.

En el modelo de clústeres de Kubernetes como servicio (KaaS), los distintos proveedores Cloud se encargan de:

- La creación, mantenimiento y configuración de las instancias que conforman el clúster.
- Replicación del *control plane* en varias zonas disponibles y/o regiones.

Por lo que el usuario solamente se encarga de los archivos de configuración a utilizar para desplegar la aplicación en el clúster.

6.8. Panorama actual

- Kubernetes es el orquestador de contenedores con mayor auge actualmente en la industria gracias a las facilidades que ofrece para desplegar aplicaciones contenerizadas de una manera más sencilla.
- Kubernetes se convirtiendo en la herramienta de orquestación de contenedores estándar, particularmente por compañías que desean ejecutar sus aplicaciones en más de una plataforma Cloud.
- Kubernetes puede desplegar sistemas distribuidos, gracias a que un clúster puede componerse de múltiples nodos que a la vez pueden ser de múltiples nubes.

- La tendencia de adopción de Kubernetes se puede reflejar con el crecimiento del 2 300 % de asistencia a la conferencia dedicada a esta tecnología, KubeCon. En 2015 se recibieron aproximadamente 500 participantes y en 2019 recibieron alrededor de 12 000.
- En Linkedin la cantidad de puestos para personas con habilidades en Kubernetes alrededor del mundo asciende a 48 107 para este 2020.

6.9. Comportamiento esperado

- Con la adopción de Multi-Cloud como estrategia de muchas empresas, Kubernetes afianzará su dominio en la rama de herramientas de orquestación de contenedores.
- AWS está implementando Kubernetes como servicio, pero *serverlees*, es decir eliminando las instancias o nodos del clúster, por lo que los contenedores se ejecutarán en entornos compartidos, lo cual ahorra costos considerablemente. Se prevé que más proveedores sigan el mismo camino.

7. CASOS DE ESTUDIO

La constante evolución de la tecnología plantea un enorme desafío para que las empresas se mantengan al día en el aspecto tecnológico[24]. Para las empresas la inversión en IT les podría resultar contraproducente porque en orden de estar a la altura del mercado deben invertir tiempo y recursos en el área de IT para mantener la competitividad, pero a su vez les resta capacidad de inversión en las distintas áreas de su negocio. Conforme el tiempo avanza cada vez son menos las empresas que optan por infraestructuras *on-premise,* en su lugar contratan los servicios que necesitan a un proveedor de servicios.[25]

La transformación digital de los negocios está siendo impulsada principalmente por el fácil acceso a Internet y el uso de este como medio de comunicación. Hoy en día, todo, absolutamente todo puede ser consumido o accedido a través de Internet. Por lo que las empresas se ven obligadas a adaptarse y someterse a una transformación digital donde cada uno de sus procesos de negocio evolucionen con la ayuda de la tecnología, con el objetivo de aumentar la productividad, competitividad y no quedar fuera del mercado.

Los servicios Cloud representan una buena oportunidad para que las empresas logren esa transformación digital que tanto necesitan para crecer y ser competitivas, ya que pueden hacerse de alto poder de cómputo sin necesidad de invertir, ni dar ningún anticipo, sino que pagan según el uso de la

[24] CHUNG, Wang; YU, William; RASHID, Ammar y CHUANG, Huan-Ming. *Toward the trend of Cloud Computing.* p. 238
[25] GODDARD, William. *The evolution of Cloud Computing – Where's It going Next.* https://the-report.cloud/the-evolution-of-cloud-computing-wheres-it-going-next.

capacidad de cómputo, por lo que puede ser sostenible con parte de las utilidades generadas.

Sin embargo, la infraestructura representa solo la mitad del camino por recorrer. Aunque estos entornos ofrecen la posibilidad de escalar de manera exponencial, sin las personas con los conocimientos técnicos de infraestructura ni las habilidades en las herramientas que aprovechan al máximo este paradigma de computación, las cosas se seguirán haciendo de la manera tradicional, pero en la nube, por lo que las empresas no podrán experimentar al cien por ciento las ventajas de Cloud.

Esto motiva a las empresas a adoptar culturas de automatización, tal como lo es DevOps, evitando los procesos repetitivos que pueden llegar a poner el peligro el avance y cumplimiento de un proyecto. Las herramientas Devops agilizan el proceso de desarrollo y además aprovechan al máximo los beneficios Cloud, existe una inmensa cantidad de herramientas, se pueden clasificar según su uso.

La migración a Cloud producto de la transformación digital de las empresas es un hecho que está en progreso alrededor del mundo, por lo que las empresas en orden de lograr una migración exitosa requieren de personal capacitado con conocimiento técnico y habilidades no solo en los diferentes servicios de las diferentes nubes, sino que, además, tengan habilidades en las diferentes herramientas que aprovechan al máximo los beneficios de Cloud.

Con la facilidad del Internet como medio de comunicación, empresas alrededor del mundo buscan talentos con habilidades y conocimientos en Cloud y herramientas DevOps en diferentes países para poder suplir su creciente necesidad de personas capacitadas, por lo que es imperativo que los egresados

de una prestigiosa casa de estudios como lo es la Universidad de San Carlos de Guatemala brinden a sus egresados las competencias necesarias para enfrentar estos retos y mantener el nivel competitivo que le caracteriza.

Tecnologías de contenedores tal como Docker y Kubernetes se están convirtiendo en los estándares líderes para las construcciones de aplicaciones contenerizadas. Estas tecnologías ayudan a liberar organizaciones de la complejidad que limita la agilidad de desarrollo. Contenedores, infraestructuras de contenedores y tecnologías para el despliegue de contenedores han probado por sí mismos que son abstracciones muy potentes que pueden ser aplicados en un sin número de diferentes escenarios.

En lugar de lanzar a los estudiantes a la exploración por cuenta propia de estas herramientas, el objetivo de la presente tesis es elaborar material que cubra los conceptos necesarios para que un estudiante pueda desarrollar habilidades junto al apoyo del auxiliar de cátedra como mentor.

Implementando estos talleres se puede estandarizar la calidad de aprendizaje y asegurarse que todos los estudiantes tengan el mismo nivel de conocimiento que a veces no sucede cuando se trabajan proyectos grupales y este se termina dividiendo entre los integrantes del equipo, por lo que algunos estudiantes tendrán poca o nula interacción con estas herramientas.

Las propuestas van centradas a herramientas *open-source* con alta aceptación en el mercado actual. La razón deriva de evitar lo que se conoce como *vendor lock-in* que hace alusión a la dependencia que existe con un proveedor de servicios específicos al utilizar sus herramientas, ya que las configuraciones no son portables entre proveedores Cloud y ante la inminente adopción de *multi-cloud* las herramientas *open source* jugarán un papel clave.

7.1.1. Perspectiva de universidades líderes

Múltiples universidades alrededor del mundo ofrecen cursos en donde enseñan conceptos DevOps. Una de ellas es la universidad Johns Hopkins de Estado Unidos, a través del curso Devops Development Course (605.609).

Metas del curso:

- Entender y ser capaz de utilizar herramientas modernas con las capacidades DevOps y método para desarrollar y operar aplicaciones. Algunos de los métodos enseñados y ejemplo de herramientas modernas incluyen:

Tabla III. **Ejemplos herramientas DevOps**

Categoria	Herramientas
Source code control	Git, GitHub, Gitlab
Virtualizacion/Contenerizacion	Virtualbox, VMware EXSi, docker
Built automation / CM	Puppet, Chef, SaltStack, ansible
Distribución binria.	Creación de artefactos
Integracion continua	Jenkins, Circle CI
Prueba automatizadas	Junit

Fuente: elaboración propia.

7.2. Análisis y Diseño 2

Se describe en la siguiente página:

Tabla IV. **Programa de curso AYD2 primer semestre 2020**

Unidad	Contenido
Elementos de diseño	Patrones de diseño Principios de diseño
Arquitectura de software	Vista, puntos de vista y perspectivas Atributos de calidad Estilos de arquitectura Proceso de arquitectura Temas transversales Descripciones estructurales Descripciones de comportamiento
Optimización	Uso de perspectivas Pruebas cuantitativas de rendimiento y escalabilidad *Profiling*

Fuente: Dtt-ecys. *Programa de curso AYD2.* https://dtt-ecys.org/. Consulta: junio de 2020.

Tabla V. **Programa de laboratorio AYD2 primer semestre 2020**

Unidad	Contenido
Framework	¿Qué es un *framework*? Tipos de *framework* Beneficios del uso de *framework* Factores de elección de un *framework*
Elementos de diseño	Patrones de diseño Contexto del diseño Principios de diseño
Despliegue en la nube	Arquitecturas monolíticas vs Microservicios PaaS SaaS Docker
Patrones de arquitectura	Categoría Objetivos

Continuación de la tabla V.

Performance y *stress*	Pruebas de rendimiento Pruebas de carga Pruebas de *stress* Herramientas para las pruebas de rendimiento *Profiling* Afinamiento de aplicaciones Métricas y herramientas de análisis de calidad de código
DevOps	Beneficios de DevOps Ciclo de vida de las aplicaciones Plan/*Code/Build/Test Release/Deploy/Operate/Monitor* DevOps Tools

Fuente: Dtt-ecys. *Programa de laboratorio AYD2.* https://dtt-ecys.org/. Consulta: junio de 2020.

Acorde a los programas tanto de clase como de laboratorio se propone integrar talleres de las siguientes herramientas, las cuales también marcan el inicio al vasto conjunto de herramientas DevOps que existen en la actualidad. Se seleccionaron las que mayor aceptación en la industria tienen actualmente.

- Herramientas de control de versiones
  - Github
  - Gitlab
- Pruebas
  - Junit
  - Cucumber
- Cloud Computing
  - AWS

    - Google Cloud
    - Azure
- Herramientas de Integración continua
    - Gitlab CI
    - Circle CI
    - Jenkins
    - Travis CI
- Empaquetamiento de software
    - Docker
- Orquestación de contenedores en entornos limitados
    - Docker-Compose

7.3. Software Avanzado

Se describe a continuación:

Tabla VI. **Programa de clase Software Avanzado primer semestre 2020**

No.	Contenido
Unidad 1	*Service Oriented* Arquitectura
Unidad 2	Microservicios
Unidad 3	COBIT
Unidad 4	ITIL
Unidad 5	*Software Qualitiy*

Fuente: Dtt-ecys. *Programa de clase Software Avanzado.* https://dtt-ecys.org/.
Consulta: junio de 2020.

Tabla VII. **Programa de laboratorio Software Avanzado primer semestre 2020**

No	Contenido
Unidad 1	Introducción y conceptos básicos de SOA
Unidad 2	Coreografía y orquestación de servicios
Unidad 3	ESB: Concepto y topologías
Unidad 4	BPEL
Unidad 5	DevOps: Introducción, desarrollo y análisis de estándares de código
Unidad 6	DevOps: Control de calidad y liberación.
Unidad 7	Imágenes y contenedores Docker
Unidad 8	Selección y definición conjunta de estándares de comunicación del proyecto
Unidad 9	Contenedores: Construcción y uso de compose
Unidad 10	Infraestructura como código
Unidad 11	Entrega de proyecto

Fuente: Dtt-ecys. *Programa de laboratorio Software Avanzado.* https://dtt-ecys.org/. Consulta: junio de 2020.

Acorde a los programas tanto de clase como de laboratorio se propone integrar talleres de las siguientes herramientas, las cuales son parte del gran ecosistema de herramientas DevOps que existen en la actualidad. Se seleccionaron las que mayor aceptación e implementación en la industria tienen.

- Configuración de servidores
  - *Chef*
  - *Ansible*
  - *Puppet*
- Despliegue de infraestructura
  - *Terraform*
- Orquestación de contenedores

- Kubernetes
  - *Helm*

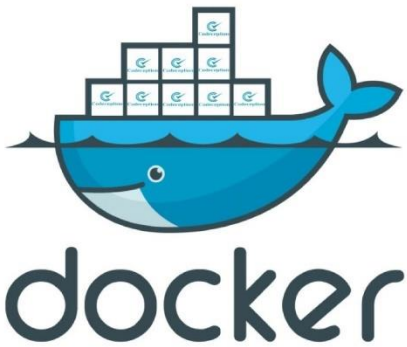
8. TALLERES PARA LABORATORIO

El material de ambos talleres puede obtenerse en el siguiente repositorio: https://github.com/ElmerReal/Tesis-201503936.

8.1. Taller Docker

El material elaborado para el taller Docker puede verse en el siguiente enlace: https://github.com/ElmerReal/Tesis-201503936/tree/master/Docker.

Figura 38. **Logo de Docker**



Fuente: Droptica. *Logo de Docker.* https://www.droptica.com/blog/codeception-how-start-automatic-tests/. Consulta: mayo de 2020.

Tabla VIII. **Información lección 1**

Titulo	Introducción
Enlace	https://youtu.be/yTcA1dr9FZs
Duración	7 minutos, 39 segundos
Contenido	Concepto de despliegue Despliegue tradicional Despliegue en máquinas virtuales Despliegue utilizando tecnología de contenedores.

Fuente: elaboración propia.

Tabla IX. **Información lección 2**

Titulo	Conceptos básicos
Enlace	https://youtu.be/c-6q4K792Ks
Duración	13 minutos, 50 segundos.
Contenido	¿Qué es Docker? Ventajas de Docker Imagen Contenedor Arquitectura Docker Instalación

Fuente: elaboración propia.

Tabla X. **Información lección 3**

Titulo	Interactuar con imágenes Docker
Enlace	https://youtu.be/z3cSg5gyzOc
Duración	8 minutos, 23 segundos.
Contenido	*docker search* *docker pull* *docker images* *docker rmi* *docker run*

Fuente: elaboración propia.

Tabla XI. **Información lección 4**

Titulo	Interactuar con contenedores
Enlace	https://youtu.be/6lnhR1nOS78
Duración	12 minutos, 59 segundos.
Contenido	*docker inspect* *docker logs* *docker ps* *docker stop* *docker start* *docker kill* *docker rm* *docker exec*

Fuente: elaboración propia.

Tabla XII. **Información lección 5**

Titulo	Configurar atributos de un contenedor utilizando *Docker run*
Enlace	https://youtu.be/5cdPrCFASjU
Duración	26 minutos, 28 segundos.
Contenido	*docker run --detach* *docker run --name* *docker run --env* *docker run --workdir* *docker run --expose* *docker run --publish* *docker run --volumen* *docker run --tmpfs* *docker run --links*

Fuente: elaboración propia.

Tabla XIII. **Información lección 6**

Titulo	Crear nuevas imágenes
Enlace	https://youtu.be/-RKMbB4-huI
Duración	24 minutos, 58 segundos.
Contenido	Capas de una imagen Imágenes base Comandos *docker commit* *docker tag* *docker login* *docker push* *Dockerfile*

Fuente: elaboración propia.

Tabla XIV. **Información lección 7**

Titulo	Enlazar múltiples contenedores
Enlace	https://youtu.be/_gUfz85GqaY
Duración	6 minutos, 16 segundos.
Contenido	*docker run --links*

Fuente: elaboración propia.

Tabla XV. **Información lección 8**

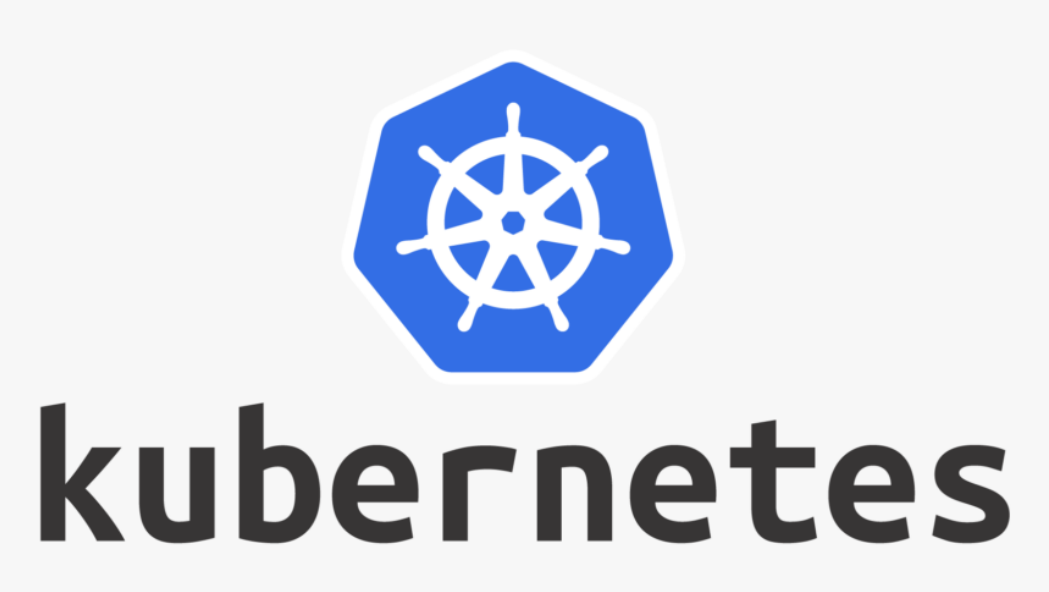
Titulo	Orquestar contenedores
Enlace	https://youtu.be/Aq9hXpdTns0
Duración	16 minutos, 17 segundos.
Contenido	*Docker compose*

Fuente: elaboración propia.

8.2. Taller de Kubernetes

El material elaborado para el taller de Kubernetes puede obtenerse en el siguiente enlace: https://github.com/ElmerReal/Tesis-201503936/tree/master/Kubernetes

Figura 39. **Logo de Kubernetes**



Fuente: Kubernetes. *Logo de Kubernetes.* https://www.pngitem. com/middle/ whJiTx_kubernetes-logo-transparent-hd-png-download/. Consulta: mayo de 2020.

Tabla XVI. **Información lección 1**

Titulo	Introducción
Enlace	https://youtu.be/bqK7RNkXOU8
Duración	26 minutos, 03 segundos
Contenido	Tipos de despliegue ¿Qué es Kubernetes? Arquitectura de un clúster k8. Casos de éxito Opinión: ¿Por qué es importante k8?

Fuente: elaboración propia.

Tabla XVII. **Información lección 2**

Titulo	*Play with* Kubernetes
Enlace	https://youtu.be/rTRAgbG9PhM
Duración	1 minutos, 15 segundos
Contenido	*Play with* Kubernetes

Fuente: elaboración propia.

Tabla XVIII. **Información lección 3**

Titulo	Preparación del entorno de trabajo
Enlace	https://youtu.be/M0vBOna5WbI
Duración	14 minutos, 52 segundos
Contenido	Diagrama de clúster ¿Qué es *kubectl*? ¿Qué es *minikube*? ¿Qué es *kubeadm*? Configuración clúster local con minikube.

Fuente: elaboración propia.

Tabla XIX. **Información lección 4**

Titulo	Conceptos básicos
Enlace	https://youtu.be/Tr3YCLe2LAI
Duración	16 minutos, 26 segundos
Contenido	*Pod* *Service* *Namespace* *Volume* *Labels* Selectores

Fuente: elaboración propia.

Tabla XX. **Información lección 5**

Titulo	Introducción a comandos imperativos
Enlace	https://youtu.be/1qv1dfOpDPA
Duración	11 minutos, 34 segundos
Contenido	Interacción con *kubectl* Interacción con *minikube*

Fuente: elaboración propia.

Tabla XXI. **Información lección 6**

Titulo	Creando *pods* y servicios utilizando comandos imperativos
Enlace	https://youtu.be/SjCiR-gzKQk
Duración	19 minutos, 44 segundos
Contenido	Ejemplo utilizando comandos imperativos.

Fuente: elaboración propia.

Tabla XXII. **Información lección 7**

Titulo	Trabajando con múltiples *namespaces* (comandos imperativos)
Enlace	https://youtu.be/WoCh_5FrtEE
Duración	5 minutos, 41 segundos
Contenido	Ejemplo creando objetos en diferentes *namespaces* con comandos imperativos.

Fuente: elaboración propia.

Tabla XXIII. **Información lección 8**

Titulo	Diferencia entre comandos imperativos y archivos de configuración declarativos
Enlace	https://youtu.be/x5jGb6wgSuA
Duración	2 minutos, 29 segundos
Contenido	Diferencias entre comandos imperativos y declarativos para crear objetos.

Fuente: elaboración propia.

Tabla XXIV. **Información lección 9**

Titulo	Trabajando con múltiples *namespaces* (archivos de configuración declarativos)
Enlace	https://youtu.be/0uEo7Wbz0K0
Duración	13 minutos, 10 segundos
Contenido	Ejemplo creando objetos en diferentes *namespaces* con archivos de declaración.

Fuente: elaboración propia.

Tabla XXV. **Información lección 10**

Titulo	*ReplicaSet*
Enlace	https://youtu.be/OwJAYXTyRxk
Duración	11 minutos, 49 segundos
Contenido	*ReplicaSet*

Fuente: elaboración propia.

Tabla XXVI. **Información lección 11**

Titulo	*Deployment*
Enlace	https://youtu.be/YJemfaecq1M
Duración	8 minutos, 33 segundos
Contenido	*Deployment*

Fuente: elaboración propia.

Tabla XXVII. **Información lección 12**

Titulo	*StatefulSet & DaemonSet*
Enlace	https://youtu.be/rIsQGRrZCos
Duración	1 minutos, 59 segundos
Contenido	*StatefulSet* *DaemonSet*

Fuente: elaboración propia.

Tabla XXVIII. **Información lección 13**

Titulo	Ejemplo *Deployment, Secrets & Configmaps*
Enlace	https://youtu.be/mBds3rHBNqQ
Duración	22 minutos, 36 segundos
Contenido	*Deployment* *Services* *Configmaps* *Secrets*

Fuente: elaboración propia.

Tabla XXIX. **Información lección 14**

Titulo	Crear clúster con *kubeadm*
Enlace	https://youtu.be/QAl3BwwvRyk
Duración	16 minutos, 02 segundos
Contenido	Crear instancias Configurar nodo *master* Configurar nodos *workers*

Fuente: elaboración propia.

CONCLUSIONES

1. Las arquitecturas de microservicios contenerizadas son parte fundamental de las aplicaciones nativas de la nube.

2. Para aprovechar todos los beneficios de Cloud Computing, las empresas deberán refactorizar sus aplicaciones para convertirlas en aplicaciones nativas de la nube.

3. Las herramientas de orquestación de contenedores son esenciales en el proceso de despliegue de las aplicaciones nativas Cloud.

4. Kubernetes aumentará su popularidad y uso en los próximos años debido a la adopción de Multi-Cloud.

5. DevOps y Cloud Computing son conceptos completamente diferentes, pero se complementan entre sí.

6. La falta de personas capacitadas en Cloud Computing y las herramientas DevOps están retrasando la completa migración de las empresas a la nube.

7. El personal con las habilidades en Cloud Computing y las distintas herramientas DevOps tendrán mejores oportunidades laborales y podrán optar a puestos de trabajo remoto en empresas extranjeras.

RECOMENDACIONES

1. Incluir más talleres sobre herramientas DevOps que pueden ser útiles para aprovechar las ventajas ofrecidas por el paradigma de computación Cloud Computing.

2. Considerar la inclusión de un curso optativo dedicado a las herramientas DevOps y entrenamiento en las diferentes nubes.

3. Complementar los talleres con proyectos que busquen poner en práctica los conocimientos adquiridos.

4. Certificar a los estudiantes en diferentes herramientas DevOps.

BIBLIOGRAFÍA

1. AHMADI, Mohammad; RAD, Babak; HARRISON, John. *An Introduction to Docker and Analysis of its Performance*. Kuala Lumpur, Malaysia. *International Journal of Computer Science and Network Security,* 2017, Vol. *17* (1), 228-235.

2. BOHM, Markus; KRCMAR, Helmut; LEIMEISTER, Stefanie; RIEDL, Christoph. *Cloud Computing and Computing Evolution.* Munchen, Germany: Technische Universität München (TUM), 2010. 28 p.

3. BOMMADEVARA, Nagendra; DEL MIGLIO, Andrea; JANSEN, Steve. *Funciones de negocio.* [en línea]. <https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/cloud-adoption-to-accelerate-it-modernization>. [Consulta: 6 de junio de 2020].

4. CAREY, Scott. *Computer World.* [en línea]. <https://www.computerworld.com/article/3511418/cloud-computing-trends-for-2020.html>. [Consulta: 6 de junio de 2020].

5. CHUNG, Wang; YU, William; RASHID, Ammar; CHUANG, Huan-Ming. *Toward the trend of Cloud Computing*. *Journal of Electronic Commerce Research,* 2011, Vol. *12* (1), 238-242.

6. Free Code Camp. *Sobre contenedores.* [en línea]. <https://www.freecodecamp.org/news/demystifying-containers-101-a-deep-dive-into-container-technology-for-beginners-d7b60d8511c1/>. [Consulta: 10 junio de 2020].

7. GODDARD, William. *The cloud report. 2 de enero de 2020*. [en línea]. <https://the-report.cloud/the-evolution-of-cloud-computing-wheres-it-going-next>. [Consulta: 20 de marzo de 2020].

8. KILARI, Nagaraju. *Cloud Computing - An Overview & Evolution.* Karnataka, India: IJSRCSEIT, 2018. 152 p.

9. MARSTON, Sean; LI, Zhi; BANDYOPADHYAY, Subhajyoti; ZHANG, Juheng; GHALSASI, Anand. Cloud computing. The business perspective. Florida, United States. *Decision Support Systems*, 2011, Vol. 51, 176-189

10. MIELL, Ian y HOBSON Sayers. *Docker in Practice.* New York, United States: Manning, 2016. 344 p.

11. MISHRA, Nitin. *Entrepeneur.* 3 de febrero de 2020. [en línea]. <https://www.entrepreneur.com/article/345826>. [Consulta: 20 de marzo de 2020].

12. NICKLOFF, Jeff. *Docker in action.* New York, United States: Manning Publications Co., 2016. 284 p.

13. OEHRLICH, Eveline; GROLL, Jayne; GARBANI, Jean-Pierre. *Upskilling 2020: Enterprise DevOps Skills Report.* Florida, United States: DevOps Institute, 2020. 64 p.

14. OpenStack. *Containers.* [en línea]. <https://www.openstack.org/containers/leveraging-containers-and-openstack/>. [Consulta: 10 de junio de 2020].

15. PADHY, Rabi; PATRA, Manas. *Evolution of Cloud Computing and Enabling Technologies. International Journal of Cloud Computing and Services Science.* 2012, Vol. *1* (1), 2089-3337.

16. RAYAPATI, Vijay. *Medium.* 7 de febrero de 2017. [en línea]. <https://articles.microservices.com/the-2017-cloud-trends-from-devops-to-noops-1d12fa85d433>. [Consulta: 10 de junio de 2020].

17. RIMOL, Meghan. *Gartner Site.* 22 de enero de 2020. [en línea]. <https://www.gartner.com/smarterwithgartner/4-trends-impacting-cloud-adoption-in-2020/>. [Consulta: 6 de junio de 2020].

18. RODRÍGUEZ, María; BUYYA, Rajkumar. *Containers Orchestration with Cost-Efficient Autoscaling in Cloud.* Melbourne, Australia: ArXiv, 2018. 22 p.

19. SHARMA, Kunal; THAKUR, Shusheel; KALIA, Arvind; THAKUR, Jawahar; KUMAR, Sunil. Emerging cloud computing paradigm: vision, research challenges and development trends. *International Journal of Research in Engineering and Technology,* 2014, Vol. *3* (1), 2319-1163.

20. STROUD, Robert. *XebiaLabs.* [en línea]. <https://xebialabs.com/resources/infographics/containers-myths-vs-facts/>. [Consulta: 10 de junio de 2020].

21. TOZZI, Christopher. *Channel Futures.* 26 de marzo de 2017. [en línea]. <https://www.channelfutures.com/open-source/docker-downsides-container-cons-to-consider-before-adopting-docker>. [Consulta: 10 de junio de 2020].

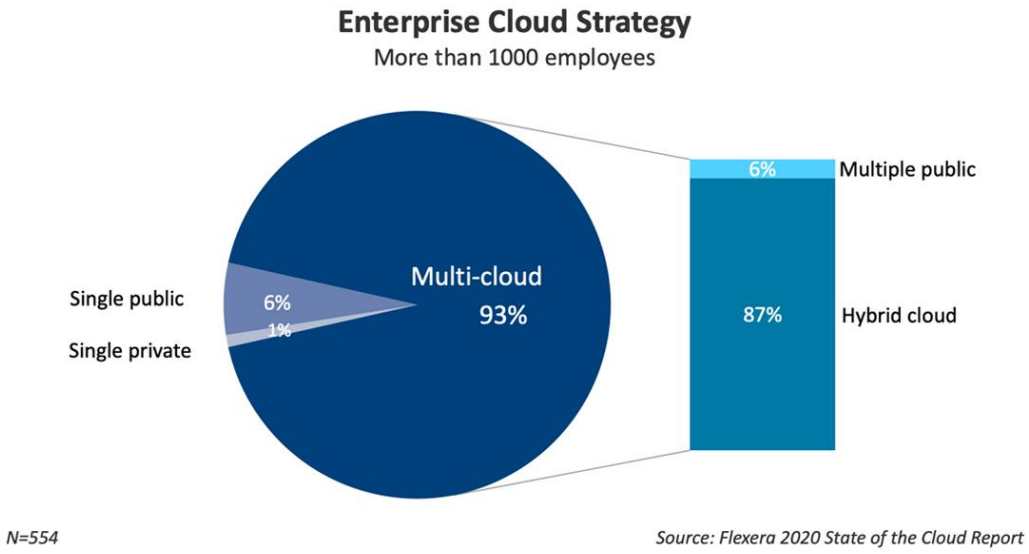
22. Veritis. *DevOps.* 5 de abril de 2018. [en línea]. <https://devops.com/devops-and-cloud-a-symbiotic-relationship-aimed-at-business-success/>. [Consulta: 10 de junio de 2020].

23. VERMA, Trilochan; VERMA, Anjali. *Cloud Computing: Evolution and Challenges.* Gharuan, Punjab, India: IJESC, 2017. 10 200 p.

24. VYAS, Jaymin. *DevOps.* 7 de noviembre de 2018. [en línea]. <https://devops.com/devops-and-cloud-a-symbiotic-relationship/>. [Consulta: 10 de junio de 2020].

ANEXOS

Anexo 1. **Estrategia empresarial en la nube**



Fuente: Flexera. *State of the Cloud report. Estrategia empresarial en la nube*. info.flexera.com/SLO-CM-REPORT-State-of-the-Cloud-2020. Consulta: junio de 2020.

Anexo 2. **Servicios de la nube con mayor crecimiento**

Top Growing Cloud Services
% of enterprise respondents

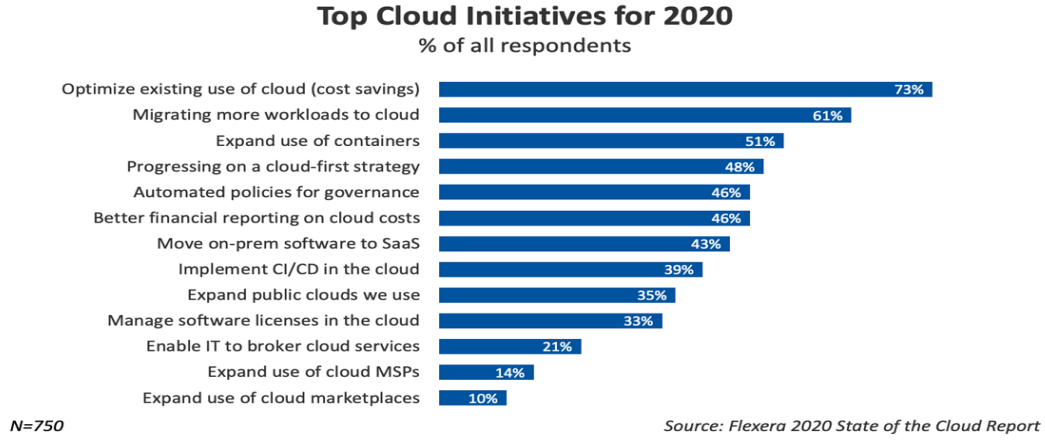
PLACE	SERVICE	2019	2020	GROWTH
1	IoT	29%	35%	21%
2	Container-as-a-service	48%	56%	17%
3	Machine learning/AI	35%	41%	17%
4	Data warehouse	50%	56%	12%
5	Serverless	43%	48%	12%

N=554

Source: Flexera 2020 State of the Cloud Report

Fuente: Flexera. *State of the Cloud report. Servicios en la nube con mayor crecimiento*. info.flexera.com/SLO-CM-REPORT-State-of-the-Cloud-2020. Consulta: junio de 2020.

Anexo 3. **Iniciativas en la nube**



Fuente: Flexera. *State of the Cloud report. Iniciativas en la nube*. info.flexera.com/SLO-CM-REPORT-State-of-the-Cloud-2020. Consulta: junio de 2020.

Anexo 4. **Uso de herramientas de contenedores en la industria**

Herramienta	Porcentaje de uso
Docker	57 %
Kubernetes	48 %
AWS ECS/EKS	44 %
Azure Container Service	28 %
Docker enterprise	27 %
RedHat openShift	24 %
Docker Swarm	21 %
Google Container Engine (GKE)	15 %
Pivotal Cloud Foundry	13 %
Mesosphere	10 %

Fuente: Tech Republic. *The 10 most container tools for business. Uso de herramientas de contenedores en la industria*. www.techrepublic.com/article/the-10-most-popular-container-tools-for-businesses/. Consulta: junio de 2020.

Anexo 5. **Uso de Kubernetes por organizaciones**

Organizaciones	Porcentaje de uso
Grandes	60 %
Pequeñas y medianas	32 %

Fuente: Tech Republic. *The 10 most container tools for business. Uso de herramientas de contenedores en la industria*. www.techrepublic.com/article/the-10-most-popular-container-tools-for-businesses/. Consulta: junio de 2020.