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

FUNDAMENTOS BÁSICOS SOBRE CONCEPTOS Y HERRAMIENTAS UTILIZADAS EN EL ÁREA *DEVOPS*

Ricardo Antonio Cutz Hernández
Asesorado por el Ing. Jhonatan Wilfredo Pú Morales

Guatemala, febrero de 2022

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

FUNDAMENTOS BÁSICOS SOBRE CONCEPTOS Y HERRAMIENTAS UTILIZADAS EN EL ÁREA *DEVOPS*

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

RICARDO ANTONIO CUTZ HERNÁNDEZ
ASESORADO POR EL ING. JHONATAN WILFREDO PÚ MORALES

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, FEBRERO DE 2022

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Cordova Estrada
VOCAL I	Ing. Jose Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Kevin Vladimir Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANA	Inga. Aurelia Anabela Cordova Estrada
EXAMINADORA	Inga. Devora Emperatris Mesa
EXAMINADOR	Ing. Marlon Francisco Orellana López
EXAMINADOR	Ing. Netfalí de Jesús Calderón Méndez
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:

FUNDAMENTOS BÁSICOS SOBRE CONCEPTOS Y HERRAMIENTAS UTILIZADAS EN EL ÁREA *DEVOPS*

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha julio 2021.

Ricardo Antonio Cutz Hernández

Guatemala, 06 de noviembre de 2021

Ingeniero
Carlos Alfredo Azurdia
Coordinador de Privados y Trabajos de Tesis
Escuela de Ingeniería en Ciencias y Sistemas
Facultad de Ingeniería - USAC

Respetable Ingeniero Azurdia:

Por este medio hago de su conocimiento que en mi rol de asesor del trabajo de investigación realizado por el estudiante **RICARDO ANTONIO CUTZ HERNÁNDEZ** con carné **201503476 y CUI 2996 93406 010** titulado **"FUNDAMENTOS BÁSICOS SOBRE CONCEPTOS Y HERRAMIENTAS UTILIZADAS EN EL ÁREA DEVOPS",** lo he revisado y luego de corroborar que el mismo se encuentra con un avance superior al 75 por ciento y que cumple con los objetivos propuestos en el respectivo protocolo, procedo a la aprobación respectiva.

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

Atentamente,

Ing. Jhonatan Wilfredo Pú Morales
Colegiado No. 15628

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

Guatemala 9 de noviembre de 2021

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 **RICARDO ANTONIO CUTZ HERNÁNDEZ** con carné **201503476 y** CUI **2996 93406 0101** titulado **"FUNDAMENTOS BÁSICOS SOBRE CONCEPTOS Y HERRAMIENTAS UTILIZADAS EN EL ÁREA DEVOPS"** 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

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

LNG.DIRECTOR.022.EICCSS.2022

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, el visto bueno del Coordinador de área y la aprobación del área de lingüística del trabajo de graduación titulado: **FUNDAMENTOS BÁSICOS SOBRE CONCEPTOS Y HERRAMIENTAS UTILIZADAS EN EL ÁREA DEVOPS**, presentado por: **Ricardo Antonio Cutz Hernández**, procedo con el Aval del mismo, ya que cumple con los requisitos normados por la Facultad de Ingeniería.

"ID Y ENSEÑAD A TODOS"

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

Guatemala, febrero de 2022

Facultad de Ingeniería

Decanato
24189101-
24189102
secretariadecanato@ingenieria.usac.edu.gt

LNG.DECANATO.OI.061.2022

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: **FUNDAMENTOS BÁSICOS SOBRE CONCEPTOS Y HERRAMIENTAS UTILIZADAS EN EL ÁREA DEVOPS,** presentado por: **Ricardo Antonio Cutz Hernández,** 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. Aurelia Anabela Cordova Estrada
Decana

Guatemala, febrero de 2022

AACE/gaoc

ACTO QUE DEDICO A:

Dios	Por ser el primer lugar en mi vida y darme la oportunidad de completar cada una de mis metas, incluyendo la culminación de mi carrera profesional. Toda la gloria y la honra siempre sea a él.
Mis padres	Regina Hernández de Cutz y Crecencio Cutz Mutz. Por su apoyo en cada etapa de mi vida, su paciencia y cariño.
Mi hermana	Scarleth Yissel Cutz Hernández. Por ser alguien especial en mi vida, siempre dándome el ánimo y motivación durante cada proceso de mi carrera.

AGRADECIMIENTOS A:

Dios	Por ser siempre mi fortaleza en cada paso de mi vida.
Mis padres	Regina Hernández de Cutz y Crecencio Cutz Mutz. Por todo el apoyo que me brindaron durante el proceso. Aprecio mucho su cariño y amor incondicional.
Universidad de San Carlos de Guatemala	Por ser parte fundamental de mi formación profesional.
Facultad de Ingeniería	Por cada una de las enseñanzas que aportó a mi desarollo profesional y personal.
Ing. Jhonatan Pú	Por tomarse el tiempo de compartir sus conocimientos y asesorarme durante el proceso de culminación de mi carrera.
Mis amigos de la Facultad	Jhosef Cáceres, Christopher López y Douglas Aguilar. Por los momentos que compartimos durante el proceso de la carrera. Aprendí muchas cosas de cada uno de ellos.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
AMI	Nombre que se le da a las imágenes de servidores virtuales que provee AWS.
AWS	Servicio de nube pública de Amazon.
JSON	Extensión de archivos que contienen una notación de objetos.
YAML	Extensión de archivos de configuración.

GLOSARIO

Ambiente	Hace referencia a un conjunto de *software* y *hardware* que tiene características específicas que permiten soportar una aplicación.
Aplicación	Programa de computadora diseñado para poder llevar a cabo tareas específicas.
Cloud Computing	Término que hace referencia a el uso de recursos computacionales virtualizados.
Cluster	Se refiere a un conjunto de recursos computacionales que se utilizan para poder alcanzar un objetivo común.
Continues Delivery	En el ámbito de desarrollo de *software* este término es asociado a la capacidad de poder generar artefactos que están listos para ser desplegados en ambientes de producción.
Continues Deployment	En el ámbito de desarrollo de *software* este término es asociado a la capacidad de poder desplegar constantemente en ambientes de producción.
Continues Integration	En el ámbito de desarrollo de *software* este término hace referencia a la capacidad de integrar pequeños

	cambios de código, realizando pruebas automatizadas y validaciones.
Contenedor	Es una unidad estándar de *software* empaquetado junto a sus dependencias.
DevOps	El término hace referencia a las siglas en inglés que combinan los términos *development* y *operations* el cuál es utilizado para describir una cultura que busca optimizar los procesos en el desarrollo y mantenimiento de infraestructura y aplicaciones.
Docker	Es una herramienta utilizada para la creación y manejo de contenedores.
Playbook	Este término es utilizado haciendo referencia a los archivos de entrada que utiliza *Ansible.* Es una archivo que contiene la definición de tareas que se deben de automatizar.
Recipes	Este término es utilizado haciendo referencia a los archivos de entrada que utiliza la herramienta *Chef.* Estos archivos contienen la definición de tareas automatizadas.

Recipes En el ámbito de desarrollo de *software* y automatización de pruebas, este término se refiere a cada una de las fases por las que el código debe de pasar para llegar a generar un artefacto que se despliegue en ambientes de producción.

RESUMEN

En la actualidad, muchas de las empresas se ven en la necesidad de utilizar plataformas digitales y desplegar aplicaciones en donde sus clientes y usuarios puedan interactuar. Desde empresas pequeñas hasta las más grandes se ven con el reto de desarrollar y administrar la infraestructura necesaria para soportar las necesidades del negocio. Para muchos, utilizar la nube, se ha convertido en una opción viable ya que permite ajustar su presupuesto de acuerdo con las necesidades debido a la facilidad de pagar por recursos que se necesitan y escalar en caso de que sea necesario.

En el mercado laboral el tener conocimientos que involucren nuevas tecnologías y automatización de procesos se ha vuelto algo sumamente valorado. El estudiante de ingeniería en sistemas tiene la oportunidad en el campo laboral de participar de la implementación de nuevas soluciones que involucren el uso de recursos de la nube. Para lograr este cometido es necesario tener el conocimiento de las herramientas actuales que se utilizan como parte de las implementaciones de soluciones.

Además de tener conocimientos técnicos es necesario conocer los conceptos y buenas prácticas que se utilizan en el campo laboral, términos como "*DevOps*" y la "nube" deben ser conocidos al interesado. Hacer accesible este tipo de información a nuevas generaciones permitirá despertar el interés en esta área laboral la cual está disponible no solo a nivel nacional sino en otras partes del mundo.

OBJETIVOS

General

Proporcionar un conjunto de herramientas y conceptos relacionados al área de *DevOps* e implementaciones en la nube para todo estudiante de ingeniería en ciencias y sistemas interesado que desea iniciar su recorrido en esta rama del campo laboral.

Específicos

1. Facilitar el acceso a ejemplos de diferentes herramientas utilizadas en automatizaciones de hoy en día.

2. Persuadir a nuevas generaciones a involucrase en temas relacionados a la nube.

3. Crear un repositorio de ejemplos prácticos con explicaciones claras sobre el uso básico de herramientas que son utilizadas en la actualidad.

4. Compartir el conocimiento de nuevas tecnologías y tendencias que hoy en día se utilizan en el campo laboral.

5. Brindar conocimiento a los estudiantes de ingeniería en sistemas sobre una rama orientada a la construcción de infraestructura en la nube.

INTRODUCCIÓN

En el ámbito laboral del estudiante egresado de ingeniería en ciencias y sistemas existen diferentes ramas en las que un profesional puede especializarse. Una de las áreas que está creciendo en popularidad durante los últimos años ha sido el uso de la nube, conocimiento en arquitectura de aplicaciones orientadas a microservicios y el conocimiento de implementaciones de infraestructura en proveedores de servicios como Amazon y Google.

En el mercado laboral, el conocimiento del uso de metodologías ágiles, implementación de automatizaciones y buenas prácticas es bastante valorado, no solo a nivel nacional sino internacional. El conocer la cultura "*DevOps*" y sus aplicaciones va de la mano con los conocimientos de *Cloud Computing*, a lo anterior podemos sumar lo importante que es el dominar el idioma inglés como una herramienta fundamental para abrirse campo en diferentes ámbitos.

En Guatemala el tema de la nube es bastante reciente y no muchas empresas cuentan con el personal capacitado para lograr implementaciones de aplicaciones en la nube por lo que se convierte en un ámbito bastante atractivo para poder explorarse. Por ello existe la necesidad de fundamentar bases sobre las tendencias relacionados a este tema, proporcionar una fuente de conocimientos prácticos y teóricos de buenas prácticas que son valoradas en el ámbito laboral y por su puesto instar al interesado a especializarse con certificaciones a nivel internacional que incrementen su valor en el mercado laboral, no solo en Guatemala sino a nivel internacional.

1. ¿QUÉ ES *DEVOPS*?

El término *DevOps* viene de la unión de dos términos muy conocidos en el idioma inglés. Estos dos términos son: desarrollo y operaciones (*development* y *operations* respectivamente). Ambos términos son conocidos en el ámbito de la carrera, siendo el equipo de desarrollo el encargado de identificar requerimientos, diseñar, construir y agregar valor a una aplicación por medio de nuevas funcionalidades, mientras que por el otro lado el equipo de operaciones es encargado de dar estabilidad, manejar la seguridad y cumplir con los estándares necesarios para la infraestructura utilizada en la aplicación. Según el papel que cada equipo desempeña se puede inferir lo siguiente: el equipo de desarrollo busca rapidez en la entrega de funcionalidades mientras que el equipo de operaciones busca estabilidad en la plataforma, si ambos equipos buscan cumplir sus objetivos sin la colaboración, puede llegar a ocasionar dificultades que puede generar pérdidas a una empresa que presta servicios por medio de un *software* al que da mantenimiento.

Los conflictos y dificultades que existen constantemente entre ambos equipos abrieron la puerta al concepto de *DevOps*, el cual no hace referencia a un grupo de herramientas o metodología de desarrollo sino a una cultura de colaboración, en donde ambos equipos pueden compartir sus metas para cumplir un objetivo común. Una definición acertada dice lo siguiente "*DevOps* es una forma de pensar y una forma de trabajo. Es un marco de trabajo para compartir historias y desarrollar empatía, permitiendo que las personas y equipos pongan en práctica sus habilidades en formas efectivas y duraderas"[1].

[1] DAVIS, Jennifer; DANIELS, Ryan. *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale.* p. 3.

1.1. Cultura *DevOps*

Tomando en cuenta que *DevOps* es una cultura, esta debe de existir, no formando dos equipos distintos (desarrolladores y operaciones) sino siendo uno mismo. Un solo equipo cuyo fin es compartir las mismas metas, las cuales deben de priorizar la estabilidad, innovación de una aplicación e incluir automatizaciones de los procesos. Parte de esta cultura incluye características que se deben de tomar en cuenta, las cuales permiten que la colaboración en equipo sea efectiva y se busque alcanzar los objetivos trazados. Cabe mencionar que puede haber muchas más características y que no existe un estándar que lo regule. A continuación, menciona las características, que, en mi experiencia, han sido muy útiles al momento de trabajar en un ambiente *DevOps*.

Uno importante a mencionar es tener metas en común. Es crucial tener claro cuáles deben ser las metas del equipo y enfocar los esfuerzos en alcanzar estas metas en común. Parte de estas metas deben ser: incrementar la estabilidad del sistema, mejorar la calidad y los tiempos de despliegue de una aplicación, mejorar la experiencia del usuario final con el sistema o aplicación, etc. Las metas que se comprometan como equipo deben de ser repetibles, claras para el equipo y cada uno debe de estar comprometido en cumplirlas.

Compartir las responsabilidades como equipo. Parte de la cultura *DevOps* es poder compartir el conocimiento y las responsabilidades, con la confianza de que cualquier miembro del equipo puede tener acceso a las herramientas necesarias para dar soporte a los ambientes en donde la aplicación está desplegada. Cada miembro del equipo debe tener el hábito y conocimientos necesarios para realizar tareas y operaciones necesarias.

Implementar automatizaciones cuando sea posible. La automatización de tareas y rutinas permite agilizar los procesos del día a día y permite que el equipo pueda enfocar los esfuerzos en otras actividades que puedan agregar valor. La automatización reduce el riesgo del error humano.

Compartir el conocimiento obtenido con todo el equipo. Es necesario que el dominio de las tecnologías y conocimientos no esté disponible para una sola persona, debe de estar distribuido y documentado apropiadamente. Esta práctica lleva a realizar revisiones continuas, demostraciones de funcionalidades y documentación de los procesos y tecnologías utilizadas.

Implementar formas de medir y monitorear. El tener la habilidad de tener la visibilidad sobre el rendimiento de las aplicaciones y la infraestructura permite que se puedan tomar decisiones de forma crítica e inteligente además que permite ser proactivos para evitar riesgos en caso de fallos.

1.2. Conceptos *DevOps*

Hemos mencionado características que deben existir en una cultura *DevOps*, así mismo existen conceptos importantes que se deben de conocer. Es necesario familiarizarse con los conceptos relacionados con el tema para entender conceptos y herramientas que se utilizan.

1.2.1. Metodologías de trabajo

El proceso de desarrollo de *software* es importante ya que consiste en diferentes fases que deben de cumplirse. Cada una de estas fases comprenden pasos importantes, iniciando con la toma de requerimientos de una aplicación, iniciar el desarrollo y verificar el código de acuerdo con las especificaciones y por

último, realizar el despliegue de la aplicación a usuarios finales en un ambiente de producción.

Existen metodologías de trabajo que se utilizan para el manejo de proyectos en el desarrollo de *software*, las ágiles y tradicionales. En las metodologías tradicionales existe una muy utilizada denominada metodología de cascada. La metodología de cascada es una herramienta utilizada para definir etapas durante el proceso:

- Requerimientos
- Diseño
- Implementación
- Verificación
- Mantenimiento

La metodología de cascada es utilizada cuando la toma de requerimientos es rigurosa para evitar que existan cambios, ya que esta metodología no permite que exista flexibilidad. Esta metodología de desarrollo puede incurrir en costos elevados en caso de que se necesite realizar cambios, en caso de que se necesite la flexibilidad de cambios si hace uso de las metodologías ágiles.

Las metodologías ágiles se caracterizan por ser flexibles y dar espacio a responder en caso de que existan cambios y mejoras. En una cultura *DevOps*, utilizar una metodología ágil es necesaria, ya que muchos de los principios que se utilizan en una metodología ágil aplican cuando se trabaja bajo una cultura

DevOps. Una de las metodologías ágiles utilizadas es Scrum, esta metodología fue diseñada con el objetivo de maximizar la habilidad de un equipo para responder de forma rápida a los cambios de acuerdo con los requerimientos del proyecto.

La metodología de Scrum tiene ciclos cortos de trabajo definidos que permiten definir las metas, revisar el progreso y tener retroalimentación de lo realizado, estos ciclos de trabajo se les denomina *sprints*. También incluye la práctica de realizar retroalimentaciones diarias que mantienen al equipo enfocado en prioridades, estas reuniones las lleva a cabo el equipo con el objetivo de responder preguntas como las siguientes:

- ¿Qué realicé el día de ayer que apoyé el objetivo del equipo?
- ¿Qué tengo planificado realizar el día de hoy para cumplir la meta del equipo?
- ¿Tengo algún tipo de obstáculo que me permita alcanzar el objetivo del equipo?

Cada una de estas reuniones de equipo permiten tener una idea clara de lo que se desea realizar y cuáles son las prioridades, permite el tener retroalimentación inmediata durante el proceso que ayuda a la toma de decisiones. Esta metodología ágil de trabajo conlleva otro tipo de conceptos y buenas prácticas, dependiendo del contexto y el equipo que implementa la metodología de trabajo pueden existir variaciones.

Tabla I. **Características de metodologías de trabajo**

Metodologías Ágiles	Metodología Tradicional
• Flexibilidad al momento de manejar cambios. • Los proyectos se dividen en pequeños periodos de tiempo. • Cada iteración del proyecto se tiene un entregable. • Enfocado a satisfacer las necesidades del cliente. • Retroalimentación constante durante el proceso.	• Funciona bastante bien con proyectos pequeños donde los requerimientos son claros y entendibles. • Está enfocado en ser un proceso de desarrollo lineal. • La aprobación y prueba del proyecto es parte del proceso, es lo último que se realiza. • Existen estados definidos que muestran el progreso del proyecto • Entregables del producto son visibles en estados finales del proyecto.

Fuente: elaboración propia, empleando WPS Office

1.2.2. Desarrollo, construcción y despliegue de aplicaciones

Además de definir las metodologías de trabajo, es importante tener claro conceptos relacionados al desarrollo. Hemos definido que *DevOps* no son herramientas sino una cultura de trabajo entonces es necesario tener claro los conceptos que describen cómo se da el proceso desde el inicio del desarrollo una aplicación hasta desplegarla para el uso de los usuarios finales, esto con el

objetivo de entender cómo encajan la cultura de *DevOps* y el uso de las herramientas que permiten alcanzar los objetivo trazados.

Uno de los conceptos importantes hoy en día, relacionado al tema de desarrollo de aplicaciones, son los sistemas de control de versiones. Los sistemas de control de versiones permiten registrar los cambios a los archivos fuente, colaborar con el equipo de trabajo y manejar las versiones del código.

En el desarrollo de una aplicación existe el concepto llamado despliegue. Este concepto hace referencia a la planificación, mantenimiento y entrega del *software*. En rasgos generales este concepto toma en consideración el manejo de los cambios que deben de agregarse a la aplicación en cada una de las entregas.

Ya que hemos mencionado conceptos relacionados a la construcción del código, existe otro concepto que se denomina integración continua. La integración continua se le denomina al proceso de integrar el código escrito en uno solo. La práctica de integrar cada uno de los cambios constantemente elimina la dificultad de que cambios realizados al código introduzcan fallas en un ambiente de producción. La idea de integrar cada uno de los cambios constantemente permite realizar pruebas al código y detectar posibles problemas antes de que sea demasiado tarde.

Otros dos términos importantes relacionados al manejo de código son despliegue continuo y entrega continua (en inglés denominados como *continuous deployment* y *continuous delivery*). El término denominado como “entrega continua” depende de realizar la integración del código constantemente, esto con el objetivo de generar artefactos que estén listos para ser desplegados en ambientes de producción, ahora bien, el término “despliegue continuo” es el

proceso de entregar versiones del código en producción, el despliegue se hace definiendo pruebas y validaciones que permitan minimizar el riesgo.

Figura 1. **CICD**

Fuente: Red Hat. *What is a CI/CD pipeline?*. https://www.redhat.com/en/topics/*DevOps*/what-cicd-pipeline. Consulta: Julio 2021.

1.2.3. Conceptos relacionados a la Infraestructura

Las aplicaciones deben ser soportadas por la infraestructura, esta puede ser manejada por alguna empresa utilizando centros de cómputo o si se utiliza infraestructura en la nube. En un inicio los términos relacionados en infraestructura eran parte solamente del equipo de operaciones, pero es importante que aún las personas involucradas con el desarrollo de aplicaciones entiendan de la infraestructura utilizada.

1.2.3.1. Manejo de configuración

Parte importante del manejo de la infraestructura es mantener la consistencia, funcionalidad y los atributos de la infraestructura. Actualmente el manejo de la configuración de la infraestructura se utilizan herramientas que

permiten automatizar el proceso y llevar el control de los cambios utilizando herramientas de control de versiones.

1.2.3.2. *Cloud Computing*

Dentro del ámbito de *DevOps* no podemos dejar de mencionar el concepto de "*Cloud Computing*", este hace referencia a recursos computacionales que pueden ser accesados por medio del Internet. Permite el acceso a recursos de almacenamiento, procesamiento y otros tipos de servicios que permiten aprovisionar infraestructura necesaria para soportar los requerimientos de una aplicación. Existen diferentes implementaciones de computación en la nube:

- Nubes públicas: puede ser accedida como un servicio, ejemplo de este tipo de nubes son: Amazon Web Services, Google Cloud, Digital Ocean, entre otros.

- Nubes privadas: este tipo de nubes no proporciona servicios a terceros y se encuentran dentro de instalaciones privadas.

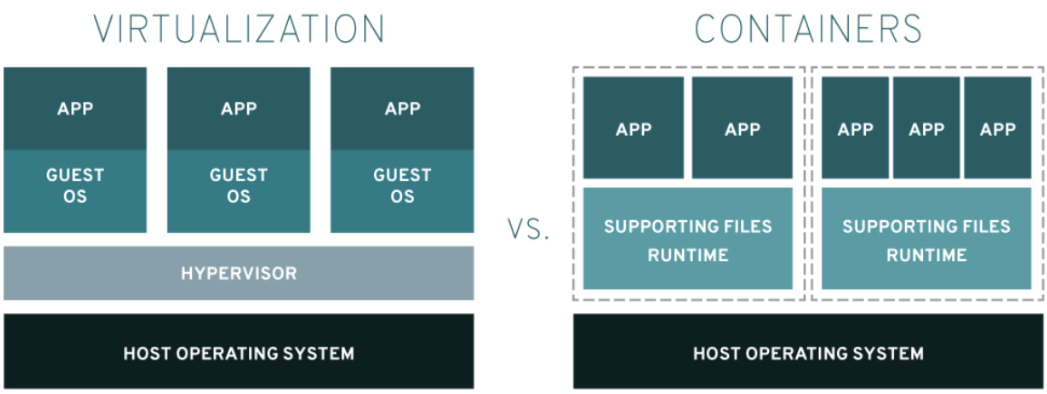
- Nubes híbridas: es una combinación de los tipos anteriormente mencionados, este tipo de arquitecturas permite a una empresa tener instalaciones privadas con una conexión con servicios prestados por nubes públicas. Esto puede tener varios propósitos como por ejemplo, tener un sistema de respaldo en caso de un desastre.

1.2.4. Maquinas virtuales y contenedores

Parte de los conceptos relacionados a *Cloud Computing* son las máquinas virtuales. Las máquinas virtuales son utilizadas para emular un equipo físico. La

infraestructura utilizada para proveer máquinas virtuales utiliza una capa ligera llamada hipervisor que permite distribuir los recursos necesarios para las máquinas virtuales, obteniendo así separación una de la otra aun cuando en realidad comparten el equipo físico. Por otra parte, los contenedores son otra forma de manejar la virtualización, pero más ligera que una máquina virtual. Los contenedores no hacen uso de un hipervisor por lo tanto pueden ser provistos de forma rápida.

Figura 2. **Máquinas virtuales y contenedores**



Fuente: Red Hat. *Diferencia entre los contenedores y las máquinas virtuales*. https://www.redhat.com/es/topics/containers/containers-vs-vms. Consulta: Julio 2021

1.3. Buenas prácticas

Parte de las bases de la implementación de *DevOps* de forma efectiva es la implementación de prácticas que permitan familiarizar a todo el equipo con la forma de trabajo. En general no hay una estandarización de prácticas que se deben de implementar sin embargo es importante mencionar algunas importantes.

Practicar la colaboración como equipo de trabajo. Es importante saber que cada uno de los individuos involucrados tiene los mismos objetivos y desean alcanzar las mismas metas. Parte de las buenas prácticas para fomentar la colaboración en equipo es poder comunicarse apropiadamente, dar participación a cada individuo, incluso el no estar de acuerdo en algún tema en específico es válido y esto permite la aceptación de nuevas ideas. Una de las buenas prácticas que *DevOps* implementa, como parte de fomentar la colaboración y comunicación, son las reuniones diarias y reportes de avances.

Las metodologías como Scrum permiten que el equipo siempre esté constantemente evaluando el progreso y medir si es posible alcanzar la meta prevista. Otras prácticas adoptadas para fomentar la colaboración son: documentar el trabajo realizado, revisar el progreso de cada miembro del equipo y por supuesto el compartir el conocimiento adquirido.

Es importante que como equipo se creen redes de trabajo. Parte esencial de un equipo es que cada uno se identifique como miembro y parte importante en alcanzar las metas trazadas. En un equipo donde existen conflictos constantes es muy difícil que puedan prosperar en lo que desean realizar. Como seres humanos tenemos la necesidad de socializar y relacionarnos con otros individuos, el poder tener una relación sana y de respeto con los miembros del equipo permite que la afinidad se fortalezca. Es importante y una práctica importante, el dar a conocer al equipo los valores que compartimos, esto ayuda a fortalecer el vínculo de trabajo en lo que se desea realizar.

Crecer y mejorar como equipo. Es importante entender que existen ciclos y en algún punto se necesita crecer y evolucionar. Parte de las buenas prácticas de *DevOps* es poder adaptarse a los cambios según sea necesario, es decir, ser

flexible. Poder ser flexible se refiere a la habilidad de entender la dirección que se desea seguir y poder adaptarse para lograr el objetivo, esto implica que como individuo necesito estar en constante capacitación y crecer en cuanto a mis conocimientos.

Implementar herramientas para agilizar procesos. Una creencia común es que *DevOps* se les llama a las herramientas, y esto es completamente falso. Las herramientas son utilizadas como parte de implementar la cultura *DevOps*. Hemos hablado de que se busca ser ágil y adaptarse a las situaciones en las que nos encontramos y parte de ello es agilizar los procesos que realizamos. Hoy en día existen muchas herramientas utilizadas en el ámbito del uso de la nube, desarrollo de *software* y manejo de infraestructura que hacen que los procesos sean automatizados, repetibles y que sean menos propensos a errores. Un pilar importante en una cultura *DevOps* es automatizar y por ello surge el uso de herramientas para realizarlo.

2. *DEVOPS* EN LA NUBE

Hoy en día conocer sobre la nube es indispensable. La nube se ha vuelto un tema popular en el ámbito de la informática gracias a los beneficios que trae utilizarla. Negocios pequeños o emprendimientos optan por utilizar la nube como base para la implementación de sus plataformas, ya que invertir en la infraestructura necesaria por medios propios puede incurrir en costos muy elevados, es en esta situación cuando los recursos en la nube se vuelven una opción viable que permite disminuir los costos de operación. En el mercado actual existen diferentes nubes públicas que las empresas utilizan, a continuación, se mencionan algunas de ellas.

2.1. Amazon Web Services

Amazon Web Services es una de las nubes más grandes que existe y utilizado por diferentes empresas ya sean grandes o pequeñas. La plataforma de AWS ofrece una gran variedad de servicios, cada uno de ellos escalables y altamente disponibles. Este proveedor cuenta con una infraestructura global, es decir que tienen instalaciones alrededor del mundo y dispersos en diferentes zonas lo cual permite alta disponibilidad. Este proveedor de nube cuenta con regiones especiales designadas por el gobierno de Estados Unidos.

AWS cuenta con un catálogo de servicios bastante extenso y variado, entre ellos cuenta con servicios de redes virtuales, almacenamiento, máquinas virtuales, servicios de migración entre muchos otros. En el ámbito de *DevOps*, este proveedor de nube ofrece servicios que permiten implementar automatizaciones y agilizar el aprovisionamiento de infraestructura. Existen

diferentes academias que permiten la capacitación relacionada a los servicios que *AWS* ofrece y certificaciones que avalan los conocimientos, las cuales son bastante valoradas en el ámbito laboral.

Figura 3. ***AWS* logo**

Fuente: Amazon. *Amazon Web Services*.
https://es.m.wikipedia.org/wiki/Archivo:AmazonWebservices_Logo.svg. Consulta: Julio 2021

2.2. Google Cloud

Google Cloud *Platform* es otro de los proveedores de nube más conocidos y utilizados hoy en día. Todos en algún momento hemos interactuado con los servicios que Google ofrece, ya sea el correo, documento o buscador. Los servicios que Google ofrece opera utilizando los recursos y servicios que ofrece este proveedor. Google Cloud *Platform* ofrece una gran variedad de servicios, cada uno de estos está separado en dominios: computación, almacenamiento y redes. Uno de los mayores fuertes de este proveedor es el servicio que ofrece Kubernetes, ya que sus orígenes se remontan a la solución que Google creó para el manejo y orquestación de contenedores.

Al igual que otros proveedores de servicio, las prácticas de *DevOps* pueden ser implementadas ya que existen herramientas y servicios que el proveedor ofrece que ayudan a la agilización y automatización de procesos, como por ejemplo herramientas como *Terraform* pueden ser utilizadas para crear recursos y manejarlos.

Figura 4. **GPC logo**

Fuente: UW Madison Information Technology. Google Cloud Platform Service Comes To Campus. https://it.wisc.edu/news/google-cloud-platform-service-comes-to-campus/. Consulta: Julio 2021

2.3. Otras nubes

Google Cloud y Amazon Web Services no son los únicos proveedores de nubes públicas que existen. Para las empresas es importante tomar en cuenta el proveedor de acuerdo con la conveniencia y los requerimientos de su infraestructura, esto incluyendo que el mantenimiento de infraestructura, que soporta los servicios que ofrece, queda en responsabilidad del proveedor de

servicio de la nube. Algunos factores que se deben de considerar al elegir un proveedor de nube pueden los siguientes:

- Seguridad que ofrece el proveedor de servicio.
- Arquitectura con la que cuenta el proveedor de servicio.
- La cantidad de servicios que ofrece y su disponibilidad.
- Soporte ofrecido a los usuarios.
- El costo de los servicios.
- Certificaciones y aceptaciones en el mercado.

Entre otros proveedores de servicios que podemos conocer se encuentran los siguientes:

- Microsoft Azure
- Digital Ocean
- Linode
- Alibaba Cloud
- IBM Cloud

3. HERRAMIENTAS PARA EL MANEJO DE INFRAESTRUCTURA

Parte importante de la cultura *DevOps* es automatizar todos los procesos posibles, esto involucra el aprovisionamiento y configuración de la infraestructura. En un modelo tradicional, el manejo de infraestructura o configuración de la misma se llevaba a cabo de forma manual, como aprovisionar una máquina virtual o configurar un servidor web, esta forma de trabajo puede ser aceptable si en caso los recursos que se desean manejar son pocos, pero manejar recursos de una aplicación que tiene ciertos requerimientos y una extensa infraestructura puede ser poco práctico y da paso a cometer errores y crear inconsistencia en la configuración. Con el objetivo de poder agilizar el proceso se crearon las herramientas que permiten manejar la infraestructura como código, lo cual hace que estos pasos sean repetibles, se trabajen en versiones e incluso crear pruebas que permitan determinar la funcionalidad. La automatización de la infraestructura como código debe de tomar en cuenta lo siguiente:

- Manejar los cambios en la infraestructura o configuración: la configuración debe de poder de comprobar el estado de las configuraciones en relación con el código y realizar los cambios necesarios para evitar inconsistencias

- Versionar el código de configuración: esto permite que se existan versiones de las configuraciones, en caso de que se necesite.

- Minimizar la complejidad: debe permitir el manejo y creación de ambientes que puedan ser repetibles y manejables.

3.1. Herramientas para el manejo de la configuración

Las herramientas de manejo de configuración permiten implementar un proceso para asegurar que un sistema se encuentre en un estado deseado y consistente. La importancia de estas herramientas radica en la importancia de mantener el estado y la consistencia de los sistemas, ya que en caso de que existan cambios, por muy pequeños que sean, pueden introducir problemas en ambientes de producción. El utilizar herramientas que permitan manejar el estado de los recursos abre la puerta a la posibilidad de contar con diferentes ambientes secundarios en donde se pueden realizar pruebas de la aplicación antes de desplegar en un ambiente de producción. Una de las ventajas más importante es la posibilidad de llevar el control de la versión y las configuraciones aplicadas, esto permite llevar la documentación de cada cambio incluso, si fuese necesario, revertir a una versión anterior.

3.1.1. Ventajas

El manejo de esta herramienta otorga las siguientes ventajas:

- Aplicar configuraciones consistentes
- Eficientes al momento de necesitar escalar la infraestructura
- Documentación de configuraciones realizadas, todo se encuentra en el código
- Tratar la configuración como código permite llevar historial en herramientas de control de versiones.

3.1.2. Ejemplos de aplicación

A continuación se presentan algunos ejemplos de herramientas utilizadas para manejar la configuración de recursos.

3.1.2.1. Ansible

Ansible es un herramienta orientada a la automatización de procesos. Es ampliamente utilizada por profesionales en el ámbito de la tecnología de la información y *DevOps*. Las diferentes aplicaciones de esta herramienta van desde la automatización de tareas, desplegar una aplicación, realizar el manejo de la configuración hasta poder aprovisionar recursos en la nube. La ventaja de Ansible como herramienta es que no se requiere conocimientos en programación

Esta herramienta tiene un flujo de trabajo bastante sencillo, utiliza pequeños módulos que son ejecutados nodos remotos, cada uno de estos módulos realizan pasos para configurar cada uno de los nodos. Una de las ventajas de esta herramienta es que no necesita tener agentes instalados en los nodos a los que se quiere aplicar una configuración en lugar de ello utiliza el protocolo de “ssh” para ejecutar cada una de las tareas. La herramienta utiliza el lenguaje conocido como YAML para definir cada una de las tareas que buscan llevar un nodo a un estado deseado.

Figura 5. **Ansible**

Fuente: ANSIBLE. *Logo*. https://commons.wikimedia.org/wiki/File:Ansible_logo.svg. Consulta: julio 2021.

3.1.2.1.1. Chef

Chef es una herramienta que permite manejar la configuración de los recursos y automatizar pasos. Esta herramienta hace uso del lenguaje de programación Ruby y es ampliamente utilizada para el manejo de servidores y despliegue de aplicaciones, sean estos en servidores físicos o en la nube. Es bastante utilizada en el ámbito de *DevOps* para buscar que los procesos de configuración de servidores sean consistentes con el estado deseado, la herramienta no asume el estado del nodo, sino que siempre comprueba que la configuración que se está aplicando sea completada.

El nombre de la herramienta puede asociarse a cada uno de los conceptos que la componen ya que Chef utiliza un concepto de crear recetas o *Recipes* que describen la configuración deseada y estas recetas se pueden agrupar para poder conformar un libro de cocina o *cookbook*, es como tener una serie de pasos que se desean ejecutar para poder llegar al estado deseado. La herramienta

requiere que cada uno de los nodos en donde se ejecute la configuración tenga el cliente de Chef instalado.

Figura 6. **Chef logo**

Fuente: Chef. io. *Chef (Software)*. https://en.wikipedia.org/wiki/Chef_(*software*). Consulta: Julio de 2021.

3.2. Herramientas para aprovisionar infraestructura

Parte de las buenas prácticas en el ámbito *DevOps* es poder implementar automatización en los procesos, esto con el fin de evitar errores humanos y agilizar las tareas. Las herramientas de infraestructura como código que permiten definir recursos son utilizadas para crear consistencia en la estructura de un sistema, incluso poder definir y documentar todas las piezas que forman parte del mismo. En la industria, el uso de este tipo de herramientas se ha convertido en algo esencial. Empresas como Google, Amazon, Netflix y muchas otras, que

manejan un flujo de clientes bastante alto, requieren escalar y manejar su infraestructura de forma automatizada, repetible y consistente. Es importante aclarar que las herramientas de infraestructura como código no solo aplican a recursos en la nube, sino que pueden ser aplicados a diferentes tipos de ambientes.

3.2.1. Ventajas

El uso de este tipo de herramientas trae consigo ciertas ventajas que podemos observar:

- Consistencia en la configuración de recursos.
- Crear ambientes que puedan ser re-producibles.
- Automatización en el proceso de aprovisionamiento de infraestructura
- Control de las versiones y cambios de la infraestructura
- Opción para crear planes de recuperación en caso de desastres

3.2.2. Retos

Es cierto que los beneficios de usar este tipo de herramientas son varios pero es importante tomar en cuenta que esto implica retos que son necesarios mencionar.

- Manejar cambios en las configuraciones: existen ocasiones en que por alguna razón se realizan cambios manuales a la configuración de un recurso, lo que introduce inconsistencia en lo existe definido en el código.

- Duplicación de errores: en caso de que parte de la definición de la infraestructura incluya errores, este puede verse duplicado en los ambientes en donde la configuración se aplicó, por lo que realizar pruebas es necesario para mitigar este tipo de errores.

- Conocimientos y destrezas: introducir este tipo de herramientas incluye una curva de aprendizaje, y esto puede tomar tiempo para los equipos.

3.2.3. *Cloudformation*

Es una herramienta de infraestructura como código utilizada exclusivamente para definir y aprovisionar recursos en la nube de *AWS*. En el ámbito de implementación de proyectos de automatización en la nube de *AWS*, *Cloudformation* es una de las herramientas más ampliamente utilizadas.

Esta herramienta de *AWS* permite definir los recursos necesarios utilizando un lenguaje llamado YAML, el cuál es utilizado comúnmente para definir configuraciones. Existe una amplia documentación de cada uno de los recursos y servicios que *AWS* ofrece. Algunos de los conceptos que se manejan al utilizar la herramienta de *Cloudformation* son los siguientes:

- Las plantillas (*templates*): se refiere al archivo que define la configuración de cada uno de los recursos que se desean de crear.

- Las pilas de recursos (*stacks*): este concepto se refiere el tener un conjunto de recursos definidos en uno o más plantillas, estos se pueden crear, actualizar e incluso eliminar.

- Conjunto de cambios (*change sets*): se refiere a la planificación de cambios que se desean realizar al actualizar una pila de recursos, el servicio de *Cloudformation* en *AWS* permite visualizar las acciones que se realizará al momento de querer actualizar los recursos, esto con el objetivo de realizar una revisión de las acciones a ejecutar.

Figura 7. **Servicio de *Cloudformation* en *AWS***

Fuente: Archo Tech. *Automatización de la nube y DevOps*. https://www.arkho.tech/partners/*AWS*/*AWS*-cloudformation. Consulta: agosto 2021.

3.2.4. *Terraform*

Es una herramienta de código abierto que permite administrar y aprovisionar infraestructura utilizando un lenguaje declarativo, esta herramienta es desarrollada por una compañía llamada Hashicorp. *Terraform* es utilizada

mayormente como herramienta para aprovisionar recursos permitiendo la automatización del proceso y puede ser utilizada en conjunto a otras herramientas para cubrir las necesidades que se tienen para la infraestructura y configuración de una aplicación.

La herramienta de *Terraform* utiliza su propio lenguaje para definir cada uno de los recursos que se desean crear y existen diferentes proveedores que pueden utilizarse para provisionar recursos. Algunos de los proveedores que tiene Terraform son:

- Oracle Cloud
- *AWS*
- Google Cloud
- Kubernetes
- Azure

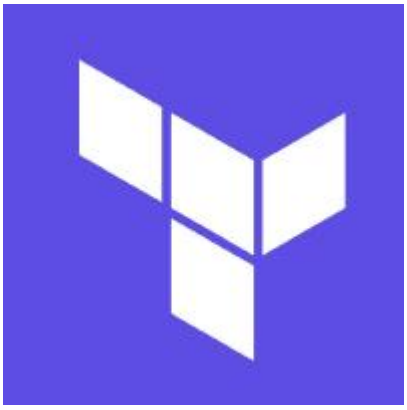
Los conceptos que se manejan en *Terraform* son:

- Archivos de configuración: los archivos de configuración contienen la sintaxis de *Terraform* para definir los recursos.
- Archivo de estado: este archivo de estado permite a *Terraform* conocer cuál es el estado de la configuración de la infraestructura. En caso de que la configuración cambie, *Terraform* se encarga de aplicar las configuraciones necesarias para llegar al estado deseado.

- Proveedores: representa una integración con proveedores de nube u otras tecnologías que permiten la interacción de *Terraform* y el proveedor para manejar los recursos.

- Espacios de trabajo: los espacios de trabajo son divisiones lógicas que la herramienta utiliza para dividir los recursos, un ejemplo común de su uso es cuando se necesita replicar diferentes ambientes (producción, desarrollo)

Figura 8. **Hashicorp *Terraform***



Fuente: Github. *Azure - Terraform*. https://github.com/Azure-*Terraform*. Consulta: agosto 2021

4. HERRAMIENTAS DE CI/CD Y USO DE SISTEMAS DE CONTROL DE VERSIONES

Parte de las buenas prácticas en el ámbito de *DevOps* es utilizar herramientas de control de versiones. Este tipo de herramientas permite llevar un historial de todos los cambios que se realizan a un conjunto de archivos, permitiendo tener diferentes versiones disponibles. Durante la carrera nos vemos con la necesidad de aprender a utilizar este tipo de herramientas para llevar el control del código de los proyectos en los que trabajamos, la mayoría del tiempo omitimos el uso de buenas prácticas. Una de las herramientas más utilizadas y populares de hoy en día es llamada *git*.

4.1. Sistemas de control de versiones: *git*

Git ha sido una de las herramientas más utilizadas para llevar el control de versiones de diferentes proyectos, existen diferentes proveedores que ofrecen el almacenamiento de repositorios de *git*. El sistema de control de versiones, denominado *git*, tuvo sus orígenes a partir de la necesidad de una herramienta que pudiera utilizarse para llevar el control de las versiones que se creaban del kernel de Linux y desde entonces ha evolucionado hasta lo que conocemos hoy en día.

4.1.1. *Git* y su relación con *DevOps*

Anteriormente hemos establecido que en una cultura *DevOps*, buscar ser eficientes, ágiles y asegurar la calidad de los entregables es muy importante y es aquí donde *git* nos muestra ser una herramienta que se acopla bastante bien para

la implementación de buenas prácticas y la colaboración entre equipos de trabajo. En el día a día podemos trabajar en diferentes proyectos para la automatización de diferentes tareas y podemos utilizar diferentes herramientas según la necesidad, estas herramientas utilizan archivos de configuración para poder realizar cada una de las actividades que deseamos, entonces aquí es donde vemos la utilidad de usar una herramienta de control de versiones para poder llevar un historial de las configuraciones aplicadas en cierto punto en el tiempo. Otra importante razón por la cual *git* es muy utilizado es que permite la colaboración en paralelo, esto quiere decir que podemos crear diferentes ramas para agilizar la colaboración en equipo con el propósito de poder llegar a un objetivo común en el entregable final.

Es importante mencionar que existen proveedores en línea que ofrecen el servicio de crear repositorios, esto con el fin de poder tener centralizado todo el código que un proyecto maneja, *git* permite sincronizar los cambios locales con el repositorio remoto, facilitando la colaboración entre individuos, este tipo de servicios que se ofrecen han dado a espacio a la colaboración de diferentes proyectos de código abierto que pueden ser utilizados. Algunos de estos proveedores que ofrecen el servicio de repositorios son: *bitbucket*, *github*, *Gitlab*, *codecommit* entre otros.

4.1.2. Herramientas de CI/CD

Un paso importante, al momento de la creación de aplicaciones, es automatizar todos los procesos que envuelve la construcción, instalación y despliegue de la aplicación. El conjunto de herramientas de control de versiones se integra con herramientas que permitan automatizar tareas para agilizar el despliegue e integración de cambios en el desarrollo de aplicaciones.

4.1.2.1. *Gitlab* CI

Gitlab es una de las plataformas utilizadas por equipos de desarrolladores para poder llevar el versionamiento de su código e implementar diferentes procesos de automatización para el desarrollo y manejo de proyectos. Actualmente podemos encontrar la plataforma de *Gitlab* que nos permite crear y manejar repositorios de código y otras funcionalidades, pero existe la posibilidad de implementar un servidor de *Gitlab* sobre una infraestructura propia. Existen dos versiones disponibles de *Gitlab*: la primera es la versión para empresas y la segunda es la versión para la comunidad, el código de *Gitlab* está disponible bajo la licencia MIT y es una plataforma de código abierto.

Algunas ventajas por mencionar de *Gitlab*:

- La administración y configuración es sencilla de utilizar
- Integración de prácticas para el ciclo *DevOps*
- Automatizar tareas para el despliegue de aplicaciones
- Manejo de repositorios de *Git*
- Puede realizarse una implementación con recursos propios
- Pueden utilizarse agentes externos que puedan ejecutar tareas
- Manejo de permisos de los diferentes colaboradores
- Permite poder importar y exportar proyectos de otras plataformas

Figura 9. ***Gitlab* CICD**

Fuente: Gitlab. *Gitlab CI Template*. https://*Gitlab*.web-toolbox.direct.canal-overseas.com/*Gitlab*-ci-template. Consulta: agosto 2021.

Para la implementación de tareas automatizadas, *Gitlab* permite crear *pipelines* que se encargan de ejecutar tareas automatizadas. Cada *pipeline* comprende los siguientes elementos:

- *Jobs* o trabajos: estos se encargan de definir qué acciones se deben de realizar

- *Recipes* o fases: cada una de las fases definen un conjunto de acciones que tienen que ejecutar.

4.1.2.2. *Github Actions*

Github es una de las plataformas más conocidas y utilizada para el manejo de proyectos, incluyendo muchos proyectos famosos así como el *kernel* de linux. Esta plataforma permite almacenar repositorios de *git* con el fin de tener una herramienta que permita la colaboración entre equipos. Esta herramienta no está disponible para la implementación en un servidor propio, es utilizada por medio de subscripción y existen limitaciones para uso en caso los proyectos que se desean almacenar sean privados.

Además del almacenamiento de repositorios, *Github* ofrece más funcionalidades que lo hacen una herramienta ideal para la colaboración y automatización de tareas. Una de estas funcionalidades, la cual es muy reciente, se llama *Github Actions*. Esta funcionalidad permite que se puedan ejecutar tareas automatizadas según la acción o el evento que se esté realizando en el repositorio. Las acciones automatizadas se pueden definir siguiendo los pasos a continuación:

- Cada repositorio debe de tener una carpeta que define diferentes “workflows” o flujos de trabajo. Cada flujo de trabajo contiene pasos que se deben de realizar para completar un determinado objetivo.

- Un flujo de trabajo puede ejecutarse de acuerdo con diferentes eventos. Los eventos son acciones que se realizan en el repositorio, como por ejemplo: cuando alguien agrega código a un repositorio puede ejecutarse una acción que genere algún tipo de artefacto o pruebas.

- Cada uno de los flujos de trabajo contienen diferentes *jobs* o trabajos que describen una serie de pasos que ejecutan acciones. Estos trabajos son las definiciones de cada una de las acciones que se deben de realizar al momento de que un evento en específico sea invocado.

Como se ha descrito anteriormente, este tipo funcionalidad permite la implementación de una de las prácticas *DevOps* muy importantes: automatizar.

Figura 10. ***Git*hub *Actions***

Fuente: NetSparker. *Integrating Netsparker Enterprise with GitHub Actions*. https://www.netsparker.com/support/integrating-netsparker-enterprise-*git*hub-actions/. Consulta: agosto 2021.

4.1.2.3. *Jenkins Server*

Hemos visto herramientas utilizadas para poder automatizar tareas que permiten desplegar y ejecutar acciones para desplegar aplicaciones. Una de las más importantes es el servidor *Jenkins*. Este servidor de automatización es uno

de los más utilizados por empresas ya que es muy versátil y puede adaptarse a las necesidades que se desean. Una de las características que hacen esta herramienta muy importante, es el hecho que es de código abierto y permite implementar estrategias *DevOps* para construir y desplegar aplicaciones de forma continua.

Este servidor puede implementarse en diferentes ambientes, según sea la necesidad. Una de las implementaciones más comunes es el uso de recursos en la nube para crear el servidor y poder escalar según sea las necesidades. Algunas de las funcionalidades que hacen a Jenkins un servidor muy utilizado son las siguientes:

- Instalable en diferentes sistemas operativos.
- Actualizaciones constantes y fáciles de realizar.
- Las funcionalidades pueden ser agregadas utilizando extensiones al servidor.
- Totalmente configurable según las necesidades.
- Utiliza el concepto de arquitectura llamada maestro y esclavo, lo que permite distribuir el trabajo en diferentes nodos.
- Integración con notificaciones.
- Construido en Java.

- La implementación de la configuración puede realizarse en el lenguaje de programación Groovy.

- Es gratuito, lo único costo que se debe de completar es el de la infraestructura y recursos utilizados para soportar las funcionalidades del servidor.

Este servidor de automatización permite configurar una gran variedad de acciones e implementar buenas prácticas del ciclo *DevOps*. Para la creación de las tareas automatizadas, Jenkins ofrece diferentes tipos de proyectos que pueden crearse, cada uno de estos proyectos configurables permiten utilizar diferentes tipos de acciones tales como ejecuciones de comandos, utilización de extensiones incluso el manejo de variables de ambientes.

Jenkins permite configurar cada una de las acciones de la siguiente forma:

- *Jobs* o trabajos: pueden contener un conjunto de pasos que conforman una tarea que se desea realizar.

- *Plugins* o extensiones: son herramientas que permiten extender la funcionalidad e incluso permiten agregar nuevas características al servidor. Existen más de cuatrocientas extensiones que pueden agregarse.

- Nodos: parte de la versatilidad de este servidor es la capacidad de poder agregar nodos en configuración de esclavos. Parte de las buenas prácticas es poder ejecutar las acciones en nodos esclavos y que el servidor maestro solo tenga la responsabilidad de manejar la configuración central.

Figura 11. **Jenkins Server**

Fuente: Lacework. *Up and Running with Lacework and Jenkins*. https://www.lacework.com/blog/running-with-jenkins/. Consulta: agosto 2021.

5. IMPLEMENTANDO APLICACIONES EN CONTENEDORES

En los últimos años se ha popularizado el uso de contenedores para el despliegue de aplicaciones, una de estas herramientas es *Docker*. Las personas encargadas del desarrollo de aplicaciones realizan pruebas y cambios en un ambiente local, donde la funcionalidad de la aplicación está condicionada a configuraciones y dependencias lo que complica el despliegue de la aplicación en ambientes de producción. El uso de contenedores permite encapsular una aplicación y sus dependencias, permite tener un ambiente aislado y puede utilizarse para el desarrollo en ambientes locales.

5.1. Contenedores y máquinas virtuales

Una tecnología similar a los contenedores son las máquinas virtuales, ambas tienen la capacidad de poder encapsular aplicaciones, pero tienen diferencias importantes que pueden definir el caso de uso entre una y otra.

5.1.1. Máquinas virtuales

Las máquinas virtuales son administradas por una herramienta que se le denomina hipervisor. El hipervisor permite crear múltiples máquinas virtuales en un sistema anfitrión, también permite la administración del *hardware* en las diferentes máquinas virtuales y por supuesto cada uno de los sistemas se encuentra en total aislamiento uno del otro. Proveedores de nube ofrecen crear máquinas virtuales con diferentes configuraciones de acuerdo con necesidades, es una de las elecciones preferidas por empresas para poder desplegar aplicaciones utilizando servidores virtuales.

Algunas de las ventajas que ofrece el uso de máquinas virtuales son:

- Ofrecen aislamiento de otras máquinas virtuales.
- El aislamiento del sistema operativo del anfitrión y el sistema operativo virtual es completo
- La virtualización ofrece la ventaja de poder recuperar ambientes completos en caso de desastre
- Es versátil cuando se desea realizar migraciones.

5.1.2. Contenedores con *Docker*

Los últimos años se ha popularizado el uso de contenedores para el desarrollo de aplicaciones, una de estas herramientas es *Docker*. Este es un proyecto de código abierto utilizado para la automatización y manejo de contenedores. Cuando se hace mención del término “contenedor” se hace referencia a una forma de virtualización mucho más ligera que permite crear un ambiente aislado donde se pueden empaquetar las librerías, dependencias y otro tipo de herramientas necesarias para una aplicación. En el caso de contenedores de *Docker*, se utiliza un motor para poder construir, guardar y correr contenedores, el cual corre a nivel del *Kernel* del sistema operativo anfitrión. El uso de los contenedores ha permitido la flexibilidad en el despliegue y empaquetado de aplicaciones, es muy utilizado como parte de las buenas prácticas *DevOps*.

Algunos conceptos que debemos de tomar conocer al hablar de contenedores son los siguientes:

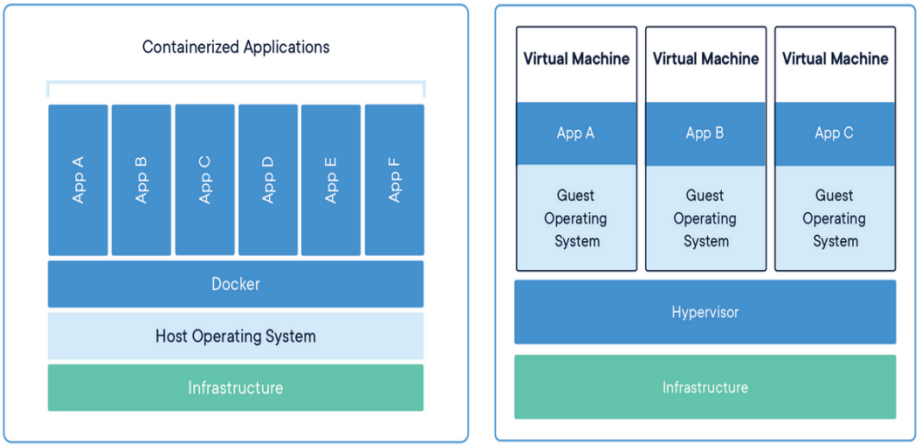
- Imagen: este término se refiere a una representación del estado de un contenedor. A diferencia de una máquina virtual, un contenedor es la representación mucho más liviana de un sistema operativo.

- Registro: para el almacenamiento y control de imágenes de contenedores existe algo llamado el registro en donde se pueden subir imágenes de contenedores que pueden ser distribuidas para su uso. Funcionan como un tipo de repositorio de imágenes para contenedores.

- *Dockerfile*: este es un término que le pertenece a los contenedores de *Docker*, este hace referencia a la representación de la configuración de una imagen. Contiene comandos necesarios para crear un contenedor y es representado por un archivo de texto que se le proporciona al motor de *Docker* para crear la imagen que se desea.

- Volúmenes: una de las funcionalidades de los contenedores de *Docker* es tener la capacidad de montar directorios locales y exponerlos al contenedor, esto con la finalidad de tener datos persistentes. Es importante aclarar que un contenedor es efímero por lo que toda información existirá siempre y cuando el contenedor esté corriendo, al momento de ser eliminado todo en el contenedor será borrado, por ello el uso de volúmenes es bastante provechoso para lograr persistir datos importantes aun cuando el contenedor sea destruido.

Figura 12. **Contenedores y máquinas virtuales**



Fuente: Docker.com. *Comparing Containers and Virtual Machines?* https://www.Docker.com/resources/what-container. Consulta septiembre de 2021.

5.1.3. Herramientas de orquestación con *Docker*

En un ambiente de producción se ha visto la necesidad de utilizar herramientas que puedan facilitar la administración de estos y esto incluyendo: escalar de acuerdo a la necesidad, realizar despliegues de nuevas versiones, poder agregar o eliminar contenedores, entre otros. Existen herramientas que permiten realizar tales tareas, a estas se les llama “orquestadores de contenedores”. El propósito de este tipo de herramientas es manejar el ciclo de vida de los contenedores utilizando archivos de configuración que describen el estado que se desea alcanzar. A continuación, se mencionan algunas de ellas

- *Kubernetes*: es una herramienta de código abierto bastante popular para la orquestación de contenedores, permite desplegar y manejar aplicaciones que están conformadas por múltiples servicios, ayuda a

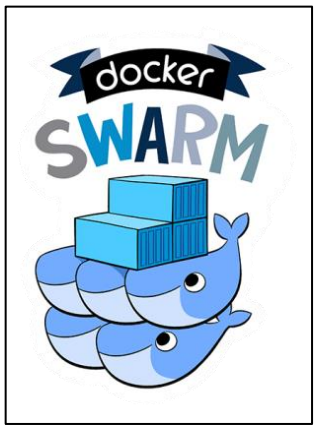
administrar la aplicación contenerizada de diferentes tipos, física, virtual y ambientes de nube.

Figura 13. ***Kubernetes***

Fuente: Kubernete. *Logo.* https://g.co/kgs/FfDvw8. Consulta: febrero de 2021.

- *Docker Swarm*: es la herramienta de orquestación de contenedores en clústeres que proporciona *Docker*, permite desplegar y orquestar contenedores en número grande de anfitriones que corren el motor de *Docker*.

Figura 14. ***Docker Swarm***



Fuente: Docker Swarm. *Logo*. https://*git*hub.com/*Docker*/classicswarm. Consulta: Septiembre 2021.

- *AWS* ECS: herramienta para el manejo de contenedores altamente escalable y rápida, es provista por Amazon para correr contenedores en clúster de máquinas virtuales (Amazon EC2) que tienen el motor de *Docker* preinstalado. ECS maneja la instalación, monitoreo y el control de las instancias del clúster por medio de la API de *AWS* o de la consola web. Además de poder utilizar instancias virtuales, *AWS* ofrece una solución *Serverless* en donde no es necesario manejar servidores sino solamente administrar los servicios y contenedores que se desean correr y *AWS* se encarga de la infraestructura para soportar los requerimientos necesarios.

Figura 15. ***AWS* ECS**

Fuente: AWS. *Implementando Clusters de ECS con Graviton2*. https://graviton2-workshop.workshop.*AWS*/es/amazoncontainers/amazonecs/ecs*Cluster*.html. Consulta: septiembre 2021.

6. HERRAMIENTAS DE MONITOREO DE INFRAESTRUCTURA

Parte de las buenas prácticas implementadas en el ámbito *DevOps* es el utilizar herramientas que permitan monitorear constantemente el estado de la infraestructura. Es necesario estar consciente de que pueden existir contratiempos, errores u otros inconvenientes que pueden afectar el estado de una aplicación, por lo que se hace necesario tomar en cuenta la implementación de herramientas que permitan visualizar el estado de cada componente de la infraestructura para identificar posibles problemas que pueden surgir. El monitoreo debe de ser implementado a nivel de infraestructura y a nivel de aplicación, esto con el objetivo de darnos una visión completa de todo el ambiente e identificar los posibles escenarios que pongan en riesgo la salud del sistema. Dependiendo del ambiente en que tengamos desplegada la infraestructura o aplicación podemos optar por herramientas que puedan cubrir la necesidad de monitorear constantemente los recursos, a continuación, se mencionan algunas de ellas.

6.1. Amazon Cloudwatch

Amazon Web Services tiene a su disposición un servicio que permite recolectar métricas de los servicios que ellos ofrecen. Esta herramienta está disponible para el uso en cada cuenta de *AWS* y permite visualizar métricas en tiempo real, permite la creación de gráficas y tableros que facilitan la observación de cada una de las medidas que se desean monitorear. Algunas de las funcionalidades que este servicio ofrece son las siguientes:

- Métricas en tiempo real.

- Configuración de alertas que pueden integrarse a otros servicios para enviar notificaciones.

- Creación de tableros y gráficas para la visualización de métricas.

- Recolección de *logs* de aplicaciones con tiempo de retención configurable.

- Los recursos que el servicio (*alarms*, gráficas, grupos de registros, entre otros) ofrece pueden crearse utilizando herramientas de infraestructura como código (*Terraform* o *Cloudformation* por ejemplo).

- Ofrece el uso de un agente que puede instalarse en servidores para la recolección de métricas específicas o personalizadas.

Figura 16. **Servicio de *AWS* Cloudwatch**

Fuente: AWS. *AWS Architecture Icons*. https://*AWS*.amazon.com/architecture/icons/. Consulta: julio 2021.

6.2. *Datadog*

Datadog es una de las herramientas de monitoreo más utilizadas, es un herramienta de pago que funciona como *Software as a Service*. Esta herramienta ofrece una gran variedad de integraciones, desde infraestructura en un *datacenter* local hasta infraestructura en diferentes proveedores de nube. Algunas de las funcionalidades que ofrece esta herramienta son las siguientes:

- Monitoreo de infraestructura.
- Recolección de métricas.
- Integración con diferentes tipos de proveedores de nubes y servicios.
- Crear métricas personalizadas.
- Recolección de “logs” de aplicaciones.
- Agente instalable en gran variedad de sistemas operativos para la recolección de métricas.
- Agente en contenedores de *Docker* para el monitoreo de *clústers* de *AWS* ECS o *Kubernetes*.
- Creación de monitores para la configuración de alertas y notificaciones que pueden ser enviadas a diferentes servicios como *Slack*, correo electrónico, entre otros.

- El uso de etiquetas para agrupar alertas, monitores y filtrar la información que se recolecta.
- Creación de tableros y gráficos que permiten el despliegue de la información en tiempo real.

- Se puede automatizar la creación de monitores, tableros y gráficas utilizando herramientas de infraestructura como código (*Terraform* permite crear recursos en *Datadog*).

Figura 17. **Servicio de *Datadog***

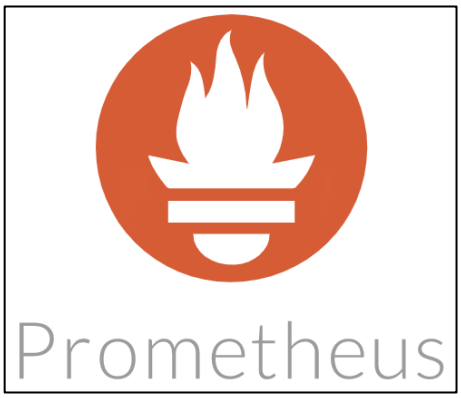
Fuente: Datadog. *Logos And Press Kits*. https://www.datadoghq.com/about/resources/. Consulta: Julio 2021.

6.3. Grafana y *Prometheus*

Estas dos son herramientas que trabajan en conjunto para monitorear métricas en la infraestructura. Definiendo cada una de las herramientas:

- *Prometheus* es una herramienta de código abierto que es utilizada para la recolección de métricas. Es considerada la herramienta por defecto para el monitoreo de *Cluster*s de *Kubernetes*. Esta herramienta permite exponer un puerto de tipo “http” para poder recolectar métricas y existen librerías que facilitan la integración de la aplicación con *Prometheus*. Los casos de uso más comunes para el uso de *Prometheus* se dan cuando se manejan aplicaciones que utilizan una arquitectura de microservicios.

Figura 18. **Logo de *Prometheus***



Fuente: Linux Foundation. *Prometheus Logo*. https://www.linuxfoundation.org/decentralized-innovation-built-with-trust/attachment/prometheus-2/. Consultado: julio 2021.

- Grafana es una herramienta para la visualización y análisis de información. Permite correr consultas sobre las métricas recolectadas, “logs” de

aplicaciones, configurar alertas y crear tableros con gráficos que permiten visualizar.

Figura 19. **Logo de Grafana**

Fuente: Grafana. Logo. https://en.wikipedia.org/wiki/Grafana#/media/File:Grafana_logo.svg. Consultado: julio 2021.

Ambas herramientas permiten integrarse nativamente por lo que hacen una solución gratuita y robusta para implementar el monitoreo de infraestructura. Puede utilizarse para monitorear recursos en servidores físicos o recursos en la nube.

7. BLOG DE HERRAMIENTAS *DEVOPS*

La tecnología va evolucionando con el tiempo y hoy en día existen nuevas herramientas que pueden utilizarse en la implementación de *DevOps* y otros ámbitos. Los estudiantes de la escuela de Ciencias y Sistemas, durante el transcurso de la carrera, son expuestos al aprendizaje de lenguajes de programación, bases de datos, compiladores y por supuesto temas relacionados a la nube y contenedores. Los cursos que actualmente abordan estos temas son los siguientes:

- Análisis y Diseño de Sistemas 2
- Software Avanzado

Los cursos anteriormente mencionados se encuentran en los últimos semestres de la carrera. Durante el transcurso de los cursos no es posible profundizar en temas que son muy importantes para el desarrollo profesional de un egresado.

7.1. Interés en el ámbito *DevOps*

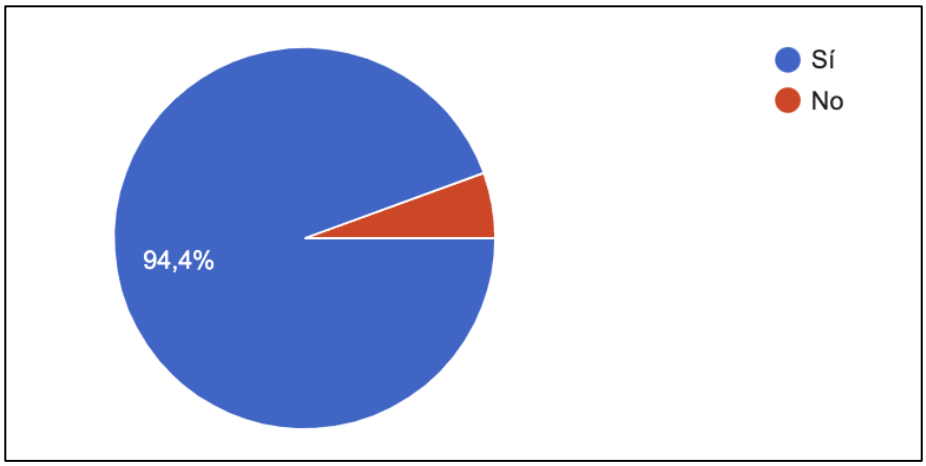
Parte del trabajo de investigación sobre el interés de los estudiantes en temas relacionados a *DevOps* y la nube se realizó una encuesta a estudiantes que cursan la carrera de Ingeniería en Ciencias y Sistemas que están cruzando desde el sexto a décimo semestre de la carrera, ya que en este rango se encuentran las personas interesadas en buscar empleo y hacer experiencia en

el ámbito laboral. El propósito de la encuesta es poder entender cuáles son los temas de interés del estudiante y evaluar si se tiene un conocimiento previo a temas de importantes en el ámbito laboral de hoy en día.

7.1.1. Evaluación del interés

La idea de poder elaborar una fuente de recursos, relacionados a herramientas *DevOps* y buenas prácticas, surgió de la evaluación del interés de estudiantes sobre temas que son importantes de conocer. Parte de esta evaluación fue presentar un conjunto de preguntas a estudiantes para poder conocer si existe un conocimiento previo y poder enfocar la segmentación de temas y herramientas que se pueden presentar en la iteración inicial.

Figura 20. **Porcentaje de interés sobre temas de la nube y *DevOps***



Fuente: elaboración propia, empleando Google Forms

Un 94 % de las respuestas fueron positivas a la pregunta que plantea el interés de aprender sobre temas relacionados a la nube y *DevOps*, afirmando

que el estudiante demuestra una inclinación a querer saber más sobre los temas relacionados a este ámbito.

Además de evaluar en general el interés por temas relacionados a la nube se realizó la evaluación para poder saber si los interesados tienen algún conocimiento previo en herramientas relacionados a ciertos aspectos importantes en el ámbito *DevOps*, estas herramientas son muy utilizadas actualmente para la implementación de automatizaciones y el manejo de infraestructura en la nube.

Tabla II. **Interés por herramientas *DevOps***

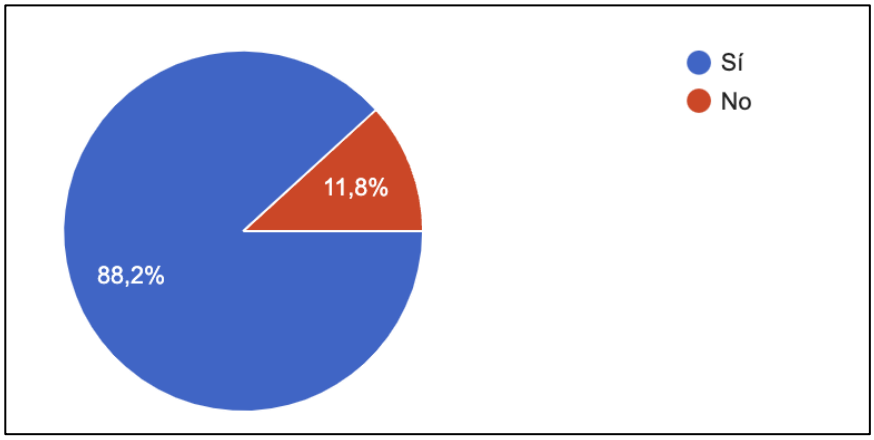
Tipo de Herramientas	**Porcentaje que conoce la herramienta**	**Porcentaje que no tiene conocimientos de la herramienta**
Herramientas para el manejo de la configuración: Ansible, Puppet, Chef	2,9 %	97,1 %
Herramientas de Integración continua y despliegue continuo: *Gitlab* CI, Jenkins	55 %	44 %
Herramientas para el manejo de infraestructura como código: *Terraform*, Cloudformation	5,9 %	94,1 %
Contenedores	41,2 %	58,8 %
Herramientas de monitoreo: Prometheus, Cloudwatch	20,6 %	79,4 %

Fuente: elaboración propia, empleando WPS Office

En la tabla anteriormente mencionada se observa que la mayor parte de los encuestados no han tenido un conocimiento previo a las herramientas

mencionas por lo que se le presentó la pregunta al encuestado si estaría interesado en conocer más sobre estos temas.

Figura 21. **Porcentaje de Interesados en las herramientas mencionadas**



Fuente: elaboración propia, empleando WPS Office

En la figura anteriormente mencionada se observa una respuesta positiva al interés de poder profundizar en las herramientas mencionadas, además se presentó una pregunta abierta para mencionar otros temas de interés, de los cuales destacaron los siguientes:

- Serverless
- Amazon Web Services
- Jenkins
- Diseño de arquitecturas de *software* en la nube

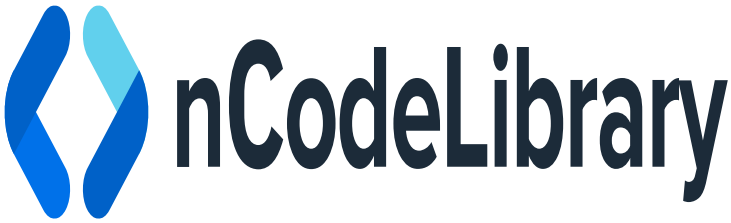
7.2. Propuesta de guía de herramientas *DevOps*

Hoy en día es bastante sencillo tener acceso a diferentes fuentes de información ya que existen diferentes medios para poder compartir el conocimiento y hacerlo llegar al público objetivo. La propuesta consiste en presentar una recopilación de ejemplos prácticos sobre diferentes tipos de herramientas utilizadas en el ámbito *DevOps*, esto con el fin de poder entregar a los interesados una fuente gratuita, en español y presentable para fomentar el interés en este tipo de temas.

7.2.1. Antecedente: nCodeLibrary

Uno de los antecedentes que presenta una propuesta similar, es conocida como nCodeLibrary la cuál es una recopilación de módulos de *Terraform* cuyo objetivo es presentar una librería robusta que puede ser utilizada para aprovisionar infraestructura en *AWS*. Cada uno de estos módulos está elaborado y probado siguiendo buenas prácticas a seguir en la creación y mantenimiento de recursos en la nube. Los módulos han sido elaborados por equipos de ingenieros de la empresa nClouds.

Figura 22. **nCodeLibrary**



Fuente: nCodeLibrary. *Logo*. https://ncodelibrary.com/. Consulta: octubre 2021.

La empresa encargada de mantener la librería de módulos de nCodeLibrary se llama nClouds. Esta empresa que ofrece servicios orientados a el mantenimiento de aplicaciones en la nube es asociada de Amazon Web Services y tiene una larga trayectoria en el ámbito de *DevOps* en la nube. La empresa cuenta con ingenieros especializados en el área por lo que la experiencia en la implementación de infraestructura como código respalda la librería de módulos presentados por “nCodeLibrary”.

Figura 23. **nClouds logo**

Fuente: nClouds. *Our Compay*. https://www.nclouds.com/about-us. Consulta: octubre 2021

7.2.2. Propuesta: *DevOps-tools*

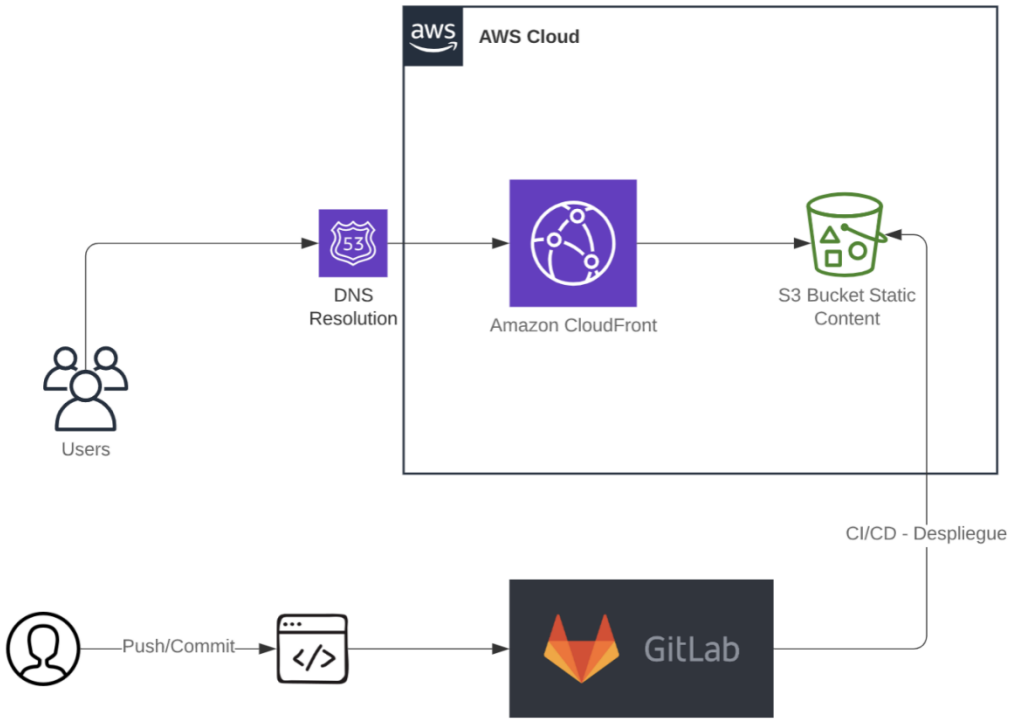
La propuesta pretende recopilar herramientas, que en el ámbito laboral me han ayudado, para implementar soluciones y diseñar proyectos,

automatizaciones y mejoras a diferentes aplicaciones que son desplegadas en la nube. Este repositorio de herramientas pretende seguir aportar a lo enseñado en la Universidad de San Carlos de Guatemala, que es id y enseñad a todos mediante la difusión del conocimiento utilizando herramientas gratuitas y de código abierto.

7.2.2.1. Infraestructura de la propuesta

La infraestructura de la propuesta se encuentra desplegada en la nube de Amazon Web Services, se ha hecho el uso de la herramienta de *Terraform* para mantener la infraestructura de forma consistente, se ha utilizado herramientas de control de versiones para poder controlar cada uno de los cambios que se agreguen.

Figura 24. **Diagrama de infraestructura**



Fuente: elaboración propia, empleando Microsoft Visio 2016.

La infraestructura está diseñada para el despliegue de un sitio estático que permita ser altamente disponible para las personas que visiten el sitio. Los componentes que forman parte de la solución son los siguientes:

- El código fuente del sitio está almacenado en *Gitlab* y se hace uso de la opción de integración y despliegue continuo que ofrece, esto con el fin de construir y desplegar el sitio en cada cambio en la rama principal.

- El servicio de S3 de Amazon Webs Services para almacenar el sitio generado de la operación de integración y despliegue de *Gitlab*.

- El servicio de Cloudfront de Amazon Web Services para poder servir el contenido del sitio de forma segura y altamente disponible.

7.2.2.2. Uso de Hugo para el sitio

El sitio se ha elaborado utilizando una herramienta muy popular en en ámbito de elaboración de sitios estáticos. Esta herramienta consiste en poder construir sitios a partir del uso de lenguaje llamado *Markdown*, el cual es utilizado para la elaboración de documentación de repositorios de código. La ventaja del uso de esta herramienta son las siguientes:

- Uso de lenguaje *Markdown* para elaborar el sitio
- Uso de plantillas para mostrar contenido.
- Cantidad variada de temas para darle estilo al sitio.
- Facilidad en la integración de herramientas de integración y despliegue continuo.
- Provee una herramienta de comandos para poder desarrollar de forma local.

Para la automatización del despliegue del sitio se ha utilizado la herramienta de *Gitlab* CI. Hugo provee una herramienta de comandos que permite generar el sitio que puede desplegarse en un servidor web. El contenido estático consta de archivos de estilo, imágenes y archivos con extensión html. Los pasos de automatización del sitio constan de los siguientes:

- Construcción y generación de archivos estáticos.

- Creación de artefactos para el despliegue.

- Subir los archivos a el servicio de S3 de Amazon Web Services

- Despliegue del sitio

Tabla III. **Código fuente del sitio**

Herramienta de control de versiones	*Gitlab*
Enlace al código fuente	https://Gitlab.com/DevOps_tools1/web-blog
Enlace a la documentación	https://Gitlab.com/DevOps_tools1/web-blog/-/blob/main/README.md

Fuente: elaboración propia, empleando WPS Office

7.2.2.3. Contenido del sitio

El sitio consta de tres partes muy importantes:

- El repositorio de ejemplos: donde se encuentra el código fuente de los ejemplos que van a ser presentados en cada uno de los apartados del sitio.

- El sitio: lugar centralizado donde cada uno de los temas será separado, presentando ejemplos y el contenido para poder aportar información al interesado.

- Demostraciones: además del contenido escrito se hace uso de demostraciones en formato video que permitan dar mayor amplitud de los temas con la finalidad de que sea de mayor interés para el usuario. El objetivo del uso de videos es que puedan ser usados como talleres en donde pueden explicarse a profundidad los ejemplos presentados.

El sitio está dividido en categorías que pretenden presentar las herramientas organizadas en dichas categorías, esto con el fin de organizar los ejemplos y caso de uso para cada tipo de herramienta:

- Infraestructura como código: todo tipo de herramientas que puedan describir recursos.

- Manejo de configuración: herramientas que permitan administrar y configurar recursos.

- Integración y despliegue continuo: herramientas que permitan automatizar tareas de despliegue e integración de código.

- Monitoreo de infraestructura: parte importante de buenas prácticas en el desarrollo de aplicaciones en la nube es el uso de monitoreo que permitan visualizar la salud del sistema.

- Contenedores: uso de contenedores, herramientas que permitan demostrar el uso de contenedores para el despliegue de aplicaciones en la nube.

Tabla IV. **Enlaces a recursos**

Recurso	Enlace
Sitio de herramientas	https://DevOps-tools.ricardcutzh.com/
Repositorio de ejemplos	https://Gitlab.com/DevOps_tools1

Fuente: elaboración propia, empleando WPS Office

CONCLUSIONES

1. La implementación de ejemplos prácticos permite que el estudiante pueda familiarizarse con el uso de nuevas herramientas que son populares en el área de *DevOps*.

2. La presentación de herramientas permite al interesado familiarizarse con los conceptos teóricos y aplicación en casos de uso reales.

3. El uso de plataformas virtuales es esencial para poder compartir y difundir información, dando acceso a los interesados.

4. La popularidad del mercado laboral para desarrollarse en el área de la nube y *DevOps* es amplia y es una de las áreas que es necesario explorar.

5. El estudiante de ingeniería en Ciencias y Sistemas tiene la posibilidad de expandir su desarrollo profesional a otras áreas que existen en el mercado laboral.

6. El conocimiento de servicios en la nube abre las posibilidades al involucrado a desarrollarse no solo en Guatemala sino en ámbitos internacionales.

RECOMENDACIONES

1. Mejorar la difusión *deel blog web* presentado para que más personas interesadas en el área de *DevOps* puedan involucrarse a participar

2. Proveer talleres opcionales relacionados a este tema con la finalidad de que el estudiante interesado pueda involucrarse en esta rama de la ingeniería.

3. Plantear la posibilidad que proyectos requeridos por los estudiantes incluyan el uso de prácticas *DevOps* que permitan desarrollar los conocimientos básicos en este tema.

4. Incluir capacitaciones que permitan a estudiantes optar por certificaciones en temas de *DevOps* y la nube con el fin de abrir campo al estudiante en otros ámbitos de la carrera.

5. Incluir temas relacionados a diferentes proveedores de servicios de nube, además de AWS.

BIBLIOGRAFÍA

1. Amazon. *Amazon Web Services*. [en línea]. <https://es.m.wikipedia.org/wiki/Archivo:AmazonWebservices_Logo.svg>. [Consulta: Julio 2021].

2. ANSIBLE. *Logo*. [en línea]. <https://commons.wikimedia.org/wiki/File:Ansible_logo.svg>. [Consulta: julio 2021].

3. Archo Tech. *Automatización de la nube y DevOps*. [en línea]. <https://www.arkho.tech/partners/*AWS*/*AWS*-cloudformation>. [Consulta: agosto 2021].

4. AWS. *AWS Architecture Icons*. [en línea]. <https://*AWS*.amazon.com/architecture/icons/>. [Consulta: julio 2021].

5. AWS. *Implementando Clusters de ECS con Graviton2*. [en línea]. <https://graviton2-workshop.workshop.*AWS*/es/amazoncontainers/amazonecs/ecs*Cluster*.html>. [Consulta: septiembre 2021].

6. Chef. io. *Chef (Software)*. [en línea]. <https://en.wikipedia.org/wiki/Chef_(*software*). [Consulta: Julio de 2021].

7. Datadog. *Logos And Press Kits*. [en línea]. <https://www.datadoghq.com/about/resources/>. [Consulta: Julio 2021].

8. DAVIS, Jennifer; DANIELS, Ryan. *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale.* California: O'Reilly Media, 2016. 410 p.

9. Docker Swarm. *Logo*. [en línea]. <https://*git*hub.com/*Docker*/classicswarm>. [Consulta: septiembre 2021].

10. Docker. *Use containers to Build, Share and Run your applications. [en línea].* <https://www.*Docker*.com/resources/what-container>. [Consulta: 10 de septiembre de 2021].

11. Docker.com. *Comparing Containers and Virtual Machines?* [en línea]. <https://www.Docker.com/resources/what-container>. [Consulta septiembre de 2021].

12. FARCIC, Viktor. *The DevOps 2.0 Toolkit.* Reino Unido: CreateSpace Independent Publishing Platform, 2016. 414 p.

13. Github. *Azure - Terraform*. [en línea]. <https://github.com/Azure-*Terraform*>. [Consulta: agosto 2021].

14. Gitlab. *Gitlab CI Template*. [en línea]. <https://*Gitlab*.web-toolbox.direct.canal-overseas.com/*Gitlab*-ci-template>. [Consulta: agosto 2021].

15. GOASGUEN, Sébastien. *Docker Cookbook. California:* O'Reilly Media, 2016. 354 p.

16. Grafana. Logo. [en línea]. <https://en.wikipedia.org/wiki/Grafana#/media/File:Grafana_logo.svg>. [Consultado: julio 2021].

17. JOURDAN, Stephane; POMES, Pierre. *Infrastructure as Code (IAC) Cookbook. Reino Unido:* Packt Publishing, 2017. 440 p.

18. KANTSEV, Vaselin. *Implementing DevOps on AWS.* Reino Unido: Packt Publishing, 2017. 258 p.

19. Kubernete. *Logo.* [en línea]. <https://g.co/kgs/FfDvw8>. [Consulta: febrero de 2021].

20. Lacework. *Up and Running with Lacework and Jenkins*. [en línea]. <https://www.lacework.com/blog/running-with-jenkins/>. [Consulta: agosto 2021].

21. Linux Foundation. *Prometheus Logo*. [en línea]. <https://www.linuxfoundation.org/decentralized-innovation-built-with-trust/attachment/prometheus-2/>. [Consultado: julio 2021].

22. MAYER, Marina. *Supply and Demand Chain.* [en línea]. <https://www.sdcexec.com/professional-development/training/news/21391662/DevOps-institute-DevOps-engineers-most-indemand-job-title-in-2021-report-says>. [Consulta: 20 de agosto de 2021].

23. MORRIS, Kief. *Infrastructure as Code, managing servers in the cloud. USA:* O'Reilly Media, Inc., 2016. 362 p.

24. nClouds. *Our Compay*. [en línea]. <https://www.nclouds.com/about-us>. [Consulta: octubre 2021].

25. nCodeLibrary. *Logo*. [en línea]. <https://ncodelibrary.com/>. [Consulta: octubre 2021].

26. NetSparker. *Integrating Netsparker Enterprise with GitHub Actions*. [en línea]. <https://www.netsparker.com/support/integrating-netsparker-enterprise-*git*hub-actions/>. [Consulta: agosto 2021].

27. Red Hat. *Diferencia entre los contenedores y las máquinas virtuales*. [en línea]. <https://www.redhat.com/es/topics/containers/containers-vs-vms>. [Consulta: Julio 2021].

28. Red Hat. *What is a CI/CD pipeline?* [en línea]. <https://www.redhat.com/en/topics/DevOps/what-cicd-pipeline>. [Consulta: Julio 2021].

29. SCHENKER, Garbiel. *Containerize your Apps with Docker and Kubernetes.* Reino Unido: Packt, 2018. 325 p.

30. SONI, Mitesh. *Jenkins Essentials: Continuous Integration – setting up the stage for a DevOps culture.* Reino Unido: Packt Publishing, 2015. 186 p.

31. SWARTOUT, Paul. *Continuous Delivery and DevOps: A Quickstart Guide.* Reino Unido: Packt Publishing, 2012. 154 p.

32. UW Madison Information Technology. *Google Cloud Platform Service Comes To Campus*. [en línea]. <https://it.wisc.edu/news/google-cloud-platform-service-comes-to-campus/>. [Consulta: Julio 2021].

33. VOHRA, Deepak. *Pro Docker.* New York: APRESS, 2016. 271 p.

APÉNDICES

Apéndice 1. **Ejemplos de Chef**

Enlace de videos	https://youtube.com/playlist?list=PL8Xnw9lMxPCGblGw2b3W9EZUC45ARNJLR
Enlace de código	https://gitlab.com/devops_tools1/iac_chef
Enlace de blog	https://devops-tools.ricardcutzh.com/mgmt-tools/chef.html

Fuente: elaboración propia, empleando WPS Office

Apéndice 2. **Ejemplos de Ansible**

Enlace de videos	https://youtube.com/playlist?list=PL8Xnw9lMxPCEau6i-bSfBXtiwfF7ZsWcY
Enlace de código	https://gitlab.com/devops_tools1/iac_ansible
Enlace de blog	https://devops-tools.ricardcutzh.com/mgmt-tools/ansible.html

Fuente: elaboración propia, empleando WPS Office

Apéndice 3. **Ejemplos de Cloudformation**

Enlace de videos	https://youtube.com/playlist?list=PL8Xnw9lMxPCH8jC310Utc774FQDHmm-MT
Enlace de código	https://gitlab.com/devops_tools1/iac-cloudformation
Enlace de blog	https://devops-tools.ricardcutzh.com/iac-tools/cloudformation.html

Fuente: elaboración propia, empleando WPS Office

Apéndice 4. **Ejemplos de Terraform**

Enlace de videos	https://youtube.com/playlist?list=PL8Xnw9lMxPCFOH2iDpG6SQdqonfIB0ntu
Enlace de código	https://gitlab.com/devops_tools1/Iac-Terraform
Enlace de blog	https://devops-tools.ricardcutzh.com/iac-tools/terraform.html

Fuente: elaboración propia, empleando WPS Office

Apéndice 5. **Ejemplos de Gitlab CI**

Enlace de videos	https://youtube.com/playlist?list=PL8Xnw9lMxPCFV5gUKh0aRtEYVYYkNRZGD
Enlace de código	https://gitlab.com/devops_tools1/cicd_gitlab
Enlace de blog	https://devops-tools.ricardcutzh.com/ci-cd-tools/gitlab_ci.html

Fuente: elaboración propia, empleando WPS Office

Apéndice 6. **Ejemplos de Github Actions**

Enlace de videos	https://youtube.com/playlist?list=PL8Xnw9lMxPCGY_YChsjd8dEEbladBWGPl
Enlace de código	https://github.com/ricardocutzhernandez/cicd_github_actions
Enlace de blog	https://devops-tools.ricardcutzh.com/ci-cd-tools/github_actions.html

Fuente: elaboración propia, empleando WPS Office

Apéndice 7. **Ejemplos de Jenkins**

Enlace de videos	https://youtube.com/playlist?list=PL8Xnw9lMxPCExQKE3RV9qDNijzNIsA8HH
Enlace de código	https://gitlab.com/devops_tools1/cicd_jenkins
Enlace de blog	https://devops-tools.ricardcutzh.com/ci-cd-tools/jenkins.html

Fuente: elaboración propia, empleando WPS Office

Apéndice 8. **Ejemplos de Docker**

Enlace de videos	https://youtube.com/playlist?list=PL8Xnw9lMxPCGQnhQwntYQrRagyFAzrE52
Enlace de código	https://gitlab.com/devops_tools1/containers-docker
Enlace de blog	https://devops-tools.ricardcutzh.com/docker/docker-basic.html

Fuente: elaboración propia, empleando WPS Office

Apéndice 9. **Ejemplos de docker-compose**

Enlace de videos	https://youtube.com/playlist?list=PL8Xnw9lMxPCHGqZVnpxqAqOhzZDAZVYJL
Enlace de código	https://gitlab.com/devops_tools1/containers-docker-compose
Enlace de blog	https://devops-tools.ricardcutzh.com/docker/docker-compose.html

Fuente: elaboración propia, empleando WPS Office

Apéndice 10. **Ejemplos de Serverless Framework**

Enlace de videos	https://youtube.com/playlist?list=PL8Xnw9lMxPCGYpntz2JIZKVms1M5l7fYi
Enlace de código	https://gitlab.com/devops_tools1/serverless-sls-framework
Enlace de blog	https://devops-tools.ricardcutzh.com/serverless/serverless-framework.html

Fuente: elaboración propia, empleando WPS Office