UNIVERSIDAD MAYOR, REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA

“VICERRECTORADO”

CENTRO DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

FACULTAD DE CIENCIAS Y TECNOLOGÍA

“IMPLEMENTACIÓN DE UNA ARQUITECTURA DEVOPS CI/CD CON GITHUB ACTIONS EN LA EMPRESA AURORASOFT”

TRABAJO EN OPCIÓN A DIPLOMADO EN DEVELOPEMENT

OPERATIONS “DEVOPS" V.1

AUTOR: DANIEL ALEJANDRO GUTIERREZ MONTAÑO

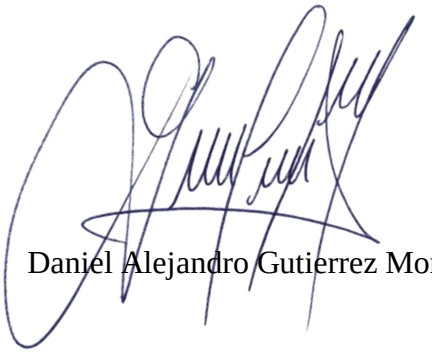
SUCRE - BOLIVIA

2024

CESIÓN DE DERECHOS

Al presentar este trabajo como requisito previo para la obtención del Diploma en DEVELOPEMENT OPERATIONS "DEVOPS" V.1. de la Universidad Mayor, Real y Pontificia de San Francisco Xavier de Chuquisaca, autorizo al Centro de Estudios de Posgrado e Investigación o a la Biblioteca de la Universidad, para que se haga de este trabajo un documento disponible para su lectura según normas de la Universidad

También cedo a la Universidad Mayor, Real y Pontifica de San Francisco Xavier de Chuquisaca, los derechos de publicación de este trabajo o parte de él, manteniendo mis derechos de autor hasta un periodo de 30 meses posterior a su aprobación.



Daniel Alejandro Gutierrez Montaño

DEDICATORIA

A mi amado Dios,

Por ser mi guía y fortaleza en cada paso de este camino.

A mis amados padres Ramiro y Yannet,

Todo lo que he logrado es gracias a su amor y guía.

A mis queridos hermanos Silvana y Fernando,

Por ser mi sostén en los momentos difíciles y por celebrar conmigo cada triunfo.

A mi adorada esposa Roxana,

Por su infinita comprensión, paciencia y amor incondicional.

A mis preciosos hijos Fernando y Mauricio

Por ser fuente de orgullo y alegría en mi vida, por su energía inagotable y su inocencia que ilumina cada día.

Con todo mi amor y gratitud,

DANIEL

AGRADECIMIENTO

Quiero expresar mi gratitud a todos los docentes y académicos de la **U.M.R. P.S.F.X.CH**. por su invaluable contribución a mi formación académica. Sus enseñanzas y conocimientos han sido una inspiración para este trabajo.

Sinceramente **a mis padres Ramiro y Yannet** por su amor incondicional, su constante apoyo y sus sacrificios para brindarme la oportunidad de seguir mis estudios. Su ejemplo de dedicación y perseverancia siempre me ha motivado a alcanzar mis metas.

A mi esposa, Roxana, por su comprensión, paciencia y por estar a mi lado en esta etapa de nuestro camino. Su amor y apoyo incondicional han sido mi mayor fortaleza.

A mis hijos, Fernando y Mauricio, por ser mi mayor inspiración y por comprender los momentos en los que debí dedicar tiempo a este proyecto. Su alegría y amor iluminan mis días y me han dado la fuerza para seguir adelante. Que este logro también sea un ejemplo para ellos de perseverancia y dedicación.

A mis hermanos Silvana y Fernando y seres queridos por su aliento, ánimo y palabras de aliento durante los momentos más desafiantes de este proceso. Su presencia ha sido un regalo invaluable que siempre atesoraré.

Finalmente, agradezco a todas aquellas personas, familiares y amigos cuyas contribuciones han enriquecido este trabajo de alguna manera.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE IMÁGENES

RESUMEN

Este trabajo se centra en mejorar las líneas de producción para el desarrollo de software a favor de la empresa AuroraSoft. Se analizan los beneficios del uso de una Arquitectura DevOps con Integración y Despliegue Continuo (CI/CD), concluyendo que, dadas las características y metodología de desarrollo de la empresa, la herramienta tecnológica más adecuada para su producción es GitHub Actions. Se evidencia la problemática de los procesos manuales de integración, pruebas y despliegue de software, proponiendo un enfoque metodológico basado en la investigación documental y aplicada, así como la descripción de sus beneficios y la puesta en marcha a través de una guía paso a paso para su implementación práctica. El presente trabajo apunta a mejorar la automatización e implementación en los procesos de desarrollo, dándoles una capacidad más eficiente a través del uso de control de versiones sistematizada, que ayuda a la evolución de cada proyecto que la empresa desea elaborar.

INTRODUCCIÓN

1. Antecedentes y Justificación

AuroraSoft es una empresa boliviana de desarrollo de software, fundada hace aproximadamente 4 años. Su principal actividad es participar en convocatorias nacionales emitidas por Entidades Públicas ofreciendo soluciones tecnológicas. Basan su desarrollo en tecnologías libres, adecuando sus procedimientos a las regulaciones de la Ley N°164 y el Decreto Supremo N°1793, mismo que establece las reglas para el desarrollo de tecnologías en el gobierno nacional.

Actualmente cuentan con un equipo compuesto por un Gerente, un Líder de Equipo de Desarrollo a cargo de 4 desarrolladores, un Líder de Equipo de Diseño a cargo de 2 profesionales encargados del análisis de requerimientos el diseño gráfico, y adicionalmente un Líder de Infraestructura y seguridad. Sin embargo, la empresa también contrata profesionales de manera temporal cuando tiene una alta demanda de proyectos.

A la fecha, AuroraSoft ha entregado 12 proyectos de software y actualmente tiene 2 proyectos en desarrollo. Todo este proceso de diseño y desarrollo, se administra sobre una infraestructura dispuesta en la nube, utilizando servicios de terceros como servidores de almacenamiento, VPS para el desarrollo de aplicaciones con JAVA, NodeJS y Python, servidores para bases de datos (principalmente PostgreSQL), y herramientas gratuitas como GitHub y Figma.

A pesar de su éxito en la entrega de proyectos, AuroraSoft carece de automatización en sus líneas de producción para el desarrollo de software, lo que ha generado entorpecimiento en el despliegue del desarrollo, un procedimiento que en nuestros días requiere que sea automatizado con la intención de mejorar la eficiencia y la calidad de sus productos.

2. Situación Problémica

Basados en el contexto precedente, las empresas se enfrentan al desafío de mejorar la eficiencia de sus procesos de desarrollo, reducir el tiempo de lanzamiento al mercado y garantizar la calidad del software desarrollado se ha convertido en el principal objetivo de todas las compañías.

Las compañías con mayor éxito en el mercado basan su desempeño en el uso de tecnologías de integración y despliegue automatizadas. Las estadísticas de la página Octoverse (GitHub, 2024), exponen claramente la creciente aceptación y éxito del uso de esta tecnología, mostrando 94 millones de desarrolladores en GitHub, de los cuales muchos de estos profesionales trabajan para empresas de renombre mundial como Amazon, Apple y Alphabet tan solo por nombrar algunas. De esta enorme cantidad de integrantes, 263 millones de trabajos son exitosamente automatizados con la tecnología GitHub Actions cada mes.

En contra parte, muchas empresas bolivianas aún mantienen el proceso de integración, pruebas y despliegue de software de manera manual, lo que conlleva a diversos problemas:

- **Lentitud en el proceso:** La implementación manual de pruebas y despliegue de software es un proceso lento y propenso a errores. Cada cambio en el código requiere una integración manual, seguida de pruebas manuales y despliegue manual, lo que ralentiza significativamente el ciclo de desarrollo.
- **Mayor probabilidad de errores:** La intervención humana en cada etapa del proceso de desarrollo aumenta la probabilidad de errores. Los errores pueden pasar desapercibidos durante las pruebas manuales y causar problemas en producción, lo que afecta la calidad del software y la satisfacción del cliente.
- **Falta de escalabilidad:** A medida que el proyecto crece, la implementación manual se vuelve menos escalable. Los equipos de desarrollo necesitan más tiempo y recursos para integrar, probar y desplegar cambios, lo que afecta la velocidad y la eficiencia del desarrollo.

3. Formulación del Problema de Investigación

Ante esta situación, surge la necesidad de implementar una metodología que permita automatizar los procesos de integración, pruebas y despliegue de software de manera eficiente y confiable. En este sentido, la pregunta de investigación que guiará este trabajo es la siguiente:

¿Cómo puede la empresa AuroraSoft implementar la metodología de Integración Continua/Entrega Continua (CI/CD) para mejorar la eficiencia, calidad y velocidad en el desarrollo de software?

Esta pregunta de investigación es fundamental para abordar la problemática identificada y proporcionar una solución que permita a AuroraSoft optimizar sus procesos de desarrollo y mejorar la calidad y velocidad de entrega de sus productos.

4. Objetivos

4.1. Objetivo General

Implementar una Arquitectura DevOps con Integración y Despliegue Continuo (CI/CD) haciendo uso de la tecnología GitHub Actions para optimizar el flujo de desarrollo de software en la empresa AuroraSoft.

4.2. Objetivos Específicos

- Analizar e identificar los beneficios que provee la sistematización y automatización de los flujos de integración y despliegue continuo a través del uso de herramientas como GitHub Actions en el desarrollo de software.
- Analizar las características y necesidades de la empresa AuroraSoft para determinar la viabilidad de implementar una Arquitectura DevOps con CI/CD.
- Elaborar un estudio comparativo sobre las diferentes herramientas disponibles para la implementación de CI/CD, centrándose en GitHub Actions como la opción más adecuada para AuroraSoft.
- Diseñar e implementar una línea de producción con CI/CD utilizando GitHub Actions en AuroraSoft, incluyendo la configuración de flujos de trabajo y la automatización de procesos.

5. Diseño Metodológico

5.1. Tipo de Investigación

El presente trabajo está enmarcado dentro de una investigación exploratoria. Se tiene como objetivo principal comprender el tema de estudio de manera amplia y profunda. La investigación exploratoria permite identificar problemas, definir términos y establecer hipótesis que orienten investigaciones posteriores. En este caso, la investigación exploratoria nos permitirá comprender la importancia y los beneficios de implementar una plataforma y metodología de control de Integración Continua/Entrega Continua (CI/CD), así como la sistematización y automatización de sus procesos, en empresas de desarrollo de software.

5.1.1. Alcance de la Investigación

El alcance de esta investigación se extiende hacia una perspectiva propositiva y filosófica orientada a la optimización de los procesos de desarrollo de software de la empresa AuroraSoft. En este sentido, se propone una investigación que no solo describa la importancia de la implementación de una plataforma y metodología de control de Integración Continua/Entrega Continua (CI/CD), sino que también presente una guía detallada y práctica para su implementación utilizando GitHub Actions a favor de la empresa AuroraSoft.

El alcance propositivo de la investigación se enfoca en ofrecer recomendaciones prácticas y aplicables a empresas de desarrollo de software, con el objetivo de mejorar su eficiencia, calidad y velocidad en el desarrollo de software. Se buscará proporcionar una visión integral y detallada de los procesos de integración, pruebas y despliegue de software mediante CI/CD, así como de la plataforma GitHub Actions.

Desde una perspectiva filosófica, el alcance de la investigación se orienta a cuestionar y repensar los procesos tradicionales de desarrollo de software, proponiendo una nueva forma de abordarlos que se base en la automatización, la sistematización y la optimización continua. Se pretende demostrar que la implementación de CI/CD con GitHub Actions no solo permite mejorar la eficiencia y calidad del desarrollo de software para AuroraSoft, sino que también promueve una cultura de colaboración, transparencia y mejora continua dentro de la metodología interna de la empresa.

El alcance de esta investigación es amplio y ambicioso, ya que busca ofrecer una visión integral y práctica de la implementación de CI/CD con GitHub Actions, así como cuestionar y repensar los procesos tradicionales de desarrollo de software desde una perspectiva filosófica y propositiva.

5.2. Métodos

5.2.1. Métodos Teóricos

Se utilizarán métodos teóricos basados en la revisión bibliográfica, así como la recopilación información relevante de diversas fuentes bibliográficas, como libros, artículos científicos y recursos en línea, relacionados con la metodología de Integración Continua/Entrega Continua (CI/CD), GitHub Actions y buenas prácticas en el desarrollo de software. Esta revisión bibliográfica permitirá obtener una comprensión profunda de los conceptos, principios y beneficios de CI/CD, así como de la plataforma GitHub Actions, lo que servirá como base teórica para la elaboración de la guía de implementación en beneficio de la empresa AuroraSoft.

5.2.2. Método Análisis – Síntesis

El método de análisis consistirá en descomponer los procesos de CI/CD y los componentes de GitHub Actions para descubrir las causas que originan los fenómenos observados. Se analizarán las etapas del proceso de integración, pruebas y despliegue de software, así como las funcionalidades y características de GitHub Actions. Posteriormente, mediante el método de síntesis, se establecerá la unión o combinación de las partes previamente analizadas para descubrir relaciones y características generales entre los elementos de la realidad. Se demostrará cómo la implementación de CI/CD con GitHub Actions permite optimizar los procesos de desarrollo de software, mejorar la calidad del producto final y acelerar su entrega al mercado.

La metodología de Integración Continua/Entrega Continua (CI/CD) permite automatizar los procesos de integración, pruebas y despliegue de software, lo que contribuye a mejorar la eficiencia, calidad y velocidad en el desarrollo de software. GitHub Actions es una plataforma de automatización de flujo de trabajo que permite implementar CI/CD de manera sencilla y eficiente. Con GitHub Actions, es posible configurar flujos de trabajo automatizados que se

activan ante eventos específicos, como la creación de una solicitud de extracción o la publicación de un nuevo código en un repositorio. Estos flujos de trabajo pueden incluir diversas acciones, como la ejecución de pruebas automáticas, el despliegue de la aplicación en entornos de prueba o producción, y la notificación de los resultados del proceso. Gracias a su integración con GitHub, GitHub Actions facilita la implementación de CI/CD en proyectos de desarrollo de software alojados en la plataforma GitHub, proporcionando una solución completa y escalable para la automatización de los procesos de desarrollo.

5.3. Técnicas

El conjunto de procedimientos organizados que ayudará a profundizar el conocimiento del presente documento queda descrito de la siguiente manera:

Investigación documental: Se realizará un exhaustivo análisis de fuentes bibliográficas, documentos, artículos científicos y recursos en línea relacionados con la metodología de Integración y Despliegue Continuo (CI/CD), Arquitectura DevOps y herramientas como GitHub Actions.

Investigación aplicada: Implementación práctica de la Arquitectura DevOps con CI/CD utilizando GitHub Actions en proyectos reales de desarrollo de software de AuroraSoft. Esto implicará la recopilación de datos, la observación directa de los procesos de desarrollo y la evaluación del impacto de la implementación.

5.4. Procedimientos e instrumentos de investigación

Han sido empleados diversos procedimientos e instrumentos que permitieron recopilar, analizar e interpretar la información de manera efectiva. Se describen los principales procedimientos e instrumentos utilizados:

5.4.1. Procedimientos:

- Revisión Bibliográfica: Se realizó una exhaustiva revisión bibliográfica sobre los conceptos fundamentales de Arquitectura DevOps, Integración y Despliegue Continuo (CI/CD) y GitHub Actions. Esta revisión proporcionó el marco teórico

necesario para comprender el contexto y la relevancia de la implementación de GitHub Actions en AuroraSoft.

- Entrevistas y Encuestas: Se llevaron a cabo entrevistas con los miembros del equipo de desarrollo de AuroraSoft, incluyendo desarrolladores, líderes de equipo y responsables de infraestructura y seguridad. Estas entrevistas permitieron obtener información detallada sobre los procesos de desarrollo de software de la empresa, así como identificar problemas, desafíos y áreas de mejora. Ver Anexo I.

- Análisis Documental: Se analizaron documentos internos de AuroraSoft, como registros de incidencias, informes de errores, y métricas de rendimiento, para obtener una visión completa de la situación actual de la empresa en relación con sus procesos de desarrollo de software. Estos documentos son internos y de uso exclusivo de la empresa, por lo que se mantendrán de manera reservada.

5.4.2. Instrumentos:

- Cuestionarios Estructurados: Se diseñaron cuestionarios estructurados para recopilar información específica sobre temas relevantes para la investigación, como los procesos de desarrollo de software, las herramientas y tecnologías utilizadas, y los problemas y desafíos enfrentados por el equipo de desarrollo. Ver Anexo I.

- Software de Análisis de Datos: Se utilizaron herramientas de análisis de datos, como hojas de cálculo y software estadístico, para tabular, codificar y analizar la información recopilada durante la investigación.

- Herramientas de Colaboración: Se emplearon herramientas de colaboración en línea, como plataformas de gestión de proyectos y sistemas de control de versiones, para facilitar la comunicación y coordinación entre los miembros del equipo de investigación.

CAPÍTULO I

MARCO TEÓRICO Y CONTEXTUAL

1.1.Marco Teórico

El desarrollo de software se enfrenta a un entorno altamente competitivo y cambiante, donde la calidad, eficiencia y rapidez en la entrega son cruciales para el éxito de un proyecto. Una fuerte tendencia que ha dominado todos estos ultimo años, es precisamente la migración de la tecnología hacia la nube. Dicha tendencia ya se la expuso la Universidad Metropolitana de Toronto, al afirmar "La evolución de la computación en la nube dio lugar a una nueva generación de aplicaciones conocidas como aplicaciones nativas de la nube (CNA). Sin embargo, observar y monitorear estas aplicaciones puede ser un desafío, especialmente si una CNA está sujeta a requisitos de cumplimiento." (William Pourmajidi, 2023)

En este contexto, la metodología de Integración Continua/Entrega Continua (CI/CD) se presenta como una solución efectiva para optimizar los procesos de desarrollo de software. Como respaldo de este trabajo, se refuerza la idea de que "La integración y la entrega continuas (CI/CD) son el proceso de automatización del ciclo de vida de las versiones de software." (Koduri, 2023)

La Integración Continua (CI) consiste en integrar y probar automáticamente los cambios realizados en el código de un proyecto de software de manera frecuente. Esto permite detectar y corregir errores de manera temprana, evitando la acumulación de problemas a lo largo del ciclo de desarrollo y garantizando la estabilidad del código base. Por otro lado, la Entrega Continua (CD) se enfoca en automatizar el proceso de entrega de software, desde la integración hasta la puesta en producción, de manera que los cambios en el código puedan ser desplegados de forma rápida, segura y confiable en cualquier momento. La implementación de una plataforma y metodología de control CI/CD, junto con la sistematización y automatización de sus procesos, ofrece diversos beneficios para las empresas de desarrollo de software, mismos que podemos encontrarlos en los documentos proporcionados por los servicios en la nube AWS de Amazon: (Koduri, 2023)

- **Eficiencia:** Un proceso de implementación completo de CI/CD puede reducir la complejidad, las cargas de trabajo y las innumerables horas dedicadas a la depuración, la realización de procesos manuales y el mantenimiento. Para obtener más información.
- **Reducción de costes:** Cuanto más corto sea el ciclo de desarrollo, mayores serán las probabilidades de que su organización pueda cumplir sus ambiciosos objetivos y aproveche las oportunidades adecuadas en el momento adecuado.
- **Velocidad:** Por lo general, una cartera completa de CI/CD es capaz de publicar los cambios de software para los clientes en unas pocas horas. La canalización de CI/CD ayuda a mejorar el tiempo medio de recuperación (MTTR), especialmente en los casos en los que los fallos se aíslan rápidamente y se introducen pequeños parches. Para obtener más información.
- **Seguridad:** Las canalizaciones completas de CI/CD también protegen el proceso de liberación al reducir los posibles puntos de entrada de los ataques y reducir el riesgo de errores humanos. Las mejoras de seguridad que se obtienen con las canalizaciones de CI/CD totalmente automatizadas ayudan a evitar las costosas consecuencias de las filtraciones de datos, las interrupciones del servicio, etc.
- **Reducción de la deserción:** Los desarrolladores se sienten más satisfechos cuando pueden dedicar más tiempo a crear excelentes funciones y menos tiempo a sumergirse en un ciclo interminable de mantenimiento y depuración. Para las organizaciones, esto significa adquirir y retener a los mejores talentos durante períodos de tiempo más largos.
- **Código de calidad superior:** Los desarrolladores publican el código en un repositorio compartido en pequeños lotes, lo que les permite pruebas. En lugar de trabajar de forma aislada, comparten sus versiones con el equipo con frecuencia y colaboran para identificar los errores críticos. Esto proporciona apoyo a los desarrolladores, lo que ayuda a evitar que el código incorrecto llegue a la fase de producción. El apoyo de colegas desarrolladores contribuye a que los lanzamientos sean de alta calidad e impulsa el crecimiento de la organización.

En la actualidad, existen diversas estrategias que podemos encontrar para implementar la Integración Continua (CI) y la Entrega Continua (CD), cada una con sus propias

características y enfoques. A continuación, se describen algunas de las metodologías más utilizadas:

- **CI/CD Tradicional:** Según la empresa DB1, que es una empresa especializada en el desarrollo de software e implementación de metodologías de entrega continua, esta técnica tradicional se basa en la automatización de los procesos de integración, pruebas y despliegue de software, con el objetivo de garantizar la entrega continua de software de alta calidad (DB1 Global Software, 2023). Se simplifica y queda compuesta por dos procesos:
    - Integración Continua (CI): Los cambios en el código se integran automáticamente en un repositorio compartido varias veces al día. Se ejecutan pruebas automáticas para verificar la estabilidad del código.
    - Entrega Continua (CD): Una vez que las pruebas son exitosas, el código se despliega automáticamente en entornos de prueba y/o producción.
- **CI/CD en Ramas:** Así también, podemos beneficiarnos de un sistema CI/CD implementado en ramas y orientado a separar el desarrollo por tipos de funcionalidad. Según la Agencia de Desarrollo Española, ITDO, la integración y despliegue continuo se realizan de forma independiente en cada rama, es útil en proyectos de desarrollo de software que requieren la colaboración de múltiples desarrolladores y la implementación de nuevas funcionalidades de forma continua. Esta metodología permite una gestión más eficiente de versiones, pruebas aisladas y un mayor control y seguridad en el desarrollo de software (ITDO, 2024). Queda compuesta por las siguientes etapas:
    - Creación de Ramas: Cada nueva funcionalidad se desarrolla en una rama separada.
    - Integración Continua (CI): Los cambios en cada rama se integran automáticamente en un entorno de integración, donde se ejecutan pruebas automáticas.
    - Entrega Continua (CD): Una vez que las pruebas son exitosas, el código se despliega automáticamente en un entorno de pruebas asociado a la rama.

- **CI/CD Basado en Eventos:** RedHat describe los flujos de trabajo de CI/CD basados en eventos (RedHat, 2024) y como estos se activan automáticamente en respuesta a sucesos específicos, como la creación de una solicitud de extracción o la publicación de un nuevo código en un repositorio. Principalmente es utilizado en proyectos donde se requiera una automatización ágil y flexible de los procesos de integración, pruebas y despliegue de software, especialmente en entornos donde se trabaja con flujos de trabajo complejos y cambios frecuentes en el código. Esta metodología permite una respuesta rápida y eficiente a los cambios en el código y garantiza una entrega continua de software de alta calidad. Para entender su comportamiento, se describen los siguientes pasos:
    - Configuración de Eventos: Se definen los eventos que activarán los flujos de trabajo de CI/CD, como la creación de una solicitud de extracción o la publicación de un nuevo código.
    - Definición de Flujos de Trabajo: Se crean flujos de trabajo automatizados que se activan en respuesta a los eventos definidos.
    - Integración Continua (CI): Los cambios en el código se integran automáticamente en un entorno de integración, donde se ejecutan pruebas automáticas. Entrega Continua (CD): Una vez que las pruebas son exitosas, el código se despliega automáticamente en entornos de prueba y/o producción.

Adicionalmente, cada una de las metodologías puede también encontrarse descrita en el libro de "Puesta en Producción Segura" (Riera, 2023), mismo que concluye que la elección de la metodología más adecuada dependerá de las necesidades y características específicas de cada proyecto de desarrollo de software.

1.2.Estado del Arte

El Estado del Arte en el ámbito de la metodología de Integración Continua/Entrega Continua (CI/CD) y su implementación utilizando GitHub Actions, es fundamental para comprender el panorama actual y las tendencias en el desarrollo de software. En la última década, se ha observado un cambio significativo en la forma en que se desarrolla y despliega el

software, donde la automatización de los procesos de integración, pruebas y despliegue ha ganado una relevancia considerable.

Según la literatura referenciada sobre la tecnología propuesta en el presente documento, se cita: “GitHub Actions es la versión nativa que te permite ejecutar flujos de trabajo en cualquier evento y que adicionalmente, puede activar sus flujos de trabajo cuando un problema cambia de estado o se agrega a un hito o cuando se activa un motor. Los flujos de trabajo están diseñados para su reutilización.” (Kaufmann, 2022). En este sentido, se menciona también un consenso general sobre la importancia de estas metodologías en el desarrollo de software moderno destacando los siguientes puntos:

- **Eficiencia y Calidad del Desarrollo:** Los estudios muestran que la implementación de CI/CD conlleva una mejora significativa en la eficiencia y calidad del desarrollo de software. La detección temprana de errores, la automatización de pruebas y la entrega continua de software permiten acortar los ciclos de desarrollo y garantizar la estabilidad y calidad del producto final.
- **Velocidad de Entrega:** La automatización de los procesos de integración, pruebas y despliegue permite una entrega más rápida y frecuente de nuevas funcionalidades al mercado. Esto se traduce en una mayor satisfacción del cliente y una mayor competitividad para las empresas.
- **Colaboración y Comunicación:** La implementación de CI/CD fomenta la colaboración entre equipos de desarrollo, al facilitar la integración continua de cambios y la entrega frecuente de software. Esto mejora la comunicación entre los miembros del equipo y aumenta la transparencia en el proceso de desarrollo.

En cuanto a GitHub Actions, la literatura muestra un creciente interés en esta plataforma como una herramienta poderosa para la implementación de CI/CD. GitHub Actions permite configurar flujos de trabajo automatizados que se activan ante eventos específicos, como la creación de una solicitud de extracción o la publicación de un nuevo código en un repositorio. Esto facilita la automatización de los procesos de desarrollo de software y proporciona una solución completa y escalable para la implementación de CI/CD.

DevOps es una cultura, filosofía y práctica de desarrollo de software que busca integrar el desarrollo (DEV) y las operaciones (OPS) en un proceso continuo e iterativo. Esta metodología se centra en la colaboración, la comunicación y la automatización entre los equipos de desarrollo y operaciones, con el objetivo de acelerar la entrega de software, mejorar la calidad y estabilidad del mismo, y aumentar la satisfacción del cliente. Su implementación está estrechamente relacionada con la metodología de Integración Continua/Entrega Continua (CI/CD). CI/CD es un conjunto de prácticas y herramientas que automatizan los procesos de desarrollo de software, desde la integración de cambios en el código hasta la entrega y despliegue de software en entornos de producción.

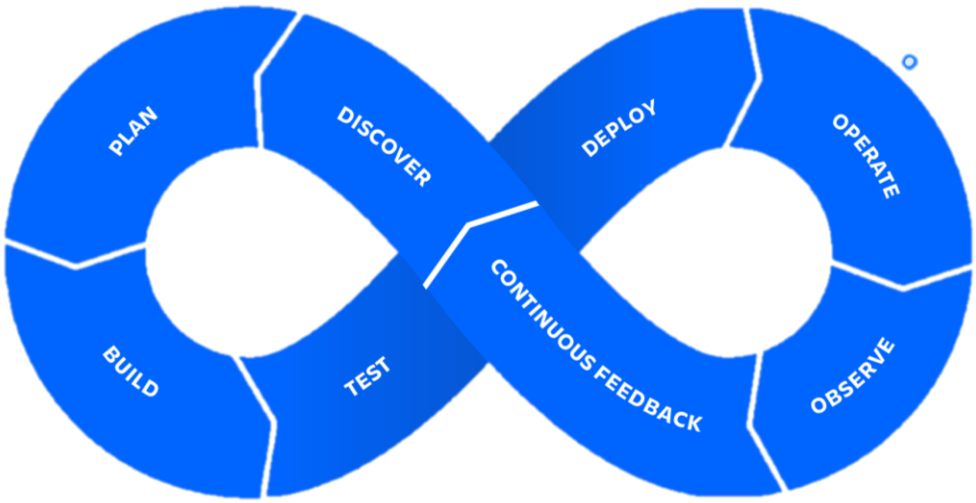


Imagen 1 – Ciclo de vida DevOps
Fuente: atlassian.com (Atlassian, 2024)

El ciclo de vida de DevOps se divide en ocho fases que abarcan los procesos, capacidades y herramientas necesarios tanto para el desarrollo (en la parte izquierda del ciclo) como para las operaciones (en la parte derecha). A lo largo de cada una de estas fases, los equipos colaboran estrechamente y mantienen una comunicación constante para coordinarse y ajustarse en términos de velocidad y calidad. Para un mejor entendimiento, haremos referencia a la documentación publicada por la empresa Atlassian, una empresa australiana especialista en el desarrollo de herramientas informáticas orientadas al control, seguimiento e implementación de los procesos de desarrollo de software (Atlassian, 2024):

- **Planificar:** Se definen los objetivos del proyecto, se identifican los requisitos y se planifican las tareas a realizar.

    - Definición de objetivos y requisitos del proyecto.
    - Planificación de las tareas a realizar.
    - Asignación de recursos y calendario de trabajo.
- **Codificar:** Se desarrolla el código fuente del proyecto, siguiendo las especificaciones y requisitos definidos en la fase de planificación.
    - Desarrollo del código fuente del proyecto.
    - Utilización de buenas prácticas de programación y diseño de software.
    - Control de versiones del código fuente utilizando herramientas como Git.
- **Compilar:** Se compila el código fuente del proyecto para generar ejecutables y/o artefactos que puedan ser utilizados en las fases siguientes del proceso de CI/CD.
    - Compilación del código fuente utilizando herramientas como Maven, Gradle, o el compilador del lenguaje de programación utilizado.
- **Pruebas Unitarias:** Se realizan pruebas automatizadas para verificar el funcionamiento correcto de unidades individuales de código.
    - Creación de casos de prueba para las unidades de código.
    - Ejecución de pruebas automatizadas utilizando herramientas como JUnit, NUnit, o PHPUnit.
- **Versión:** Se versiona y etiqueta el código fuente compilado y probado, para poder realizar un seguimiento y control de las diferentes versiones del software.
    - Etiquetado y versionado del código fuente compilado y probado.
    - Utilización de herramientas de control de versiones como Git, SVN, o Mercurial.
- **Desplegar:** Se despliega el software en entornos de prueba y/o producción, de manera automática y controlada.
    - Automatización del despliegue del software utilizando herramientas como Jenkins, Travis CI, o CircleCI.

    - Configuración y ejecución de flujos de trabajo de despliegue continuo.

- **Operar:** Se monitoriza y gestiona el software desplegado en entornos de producción, garantizando su correcto funcionamiento y rendimiento.
    - Monitorización de los sistemas y aplicaciones en producción.
    - Gestión de incidencias y problemas.
    - Ajuste y optimización de la infraestructura y configuración del software.
- **Monitorear:** Se recopilan y analizan métricas y datos de rendimiento del software en producción, con el objetivo de identificar áreas de mejora y optimización.
    - Recopilación y análisis de métricas de rendimiento y uso del software.
    - Identificación de áreas de mejora y optimización.
    - Ajuste y optimización del software y la infraestructura en función de los resultados obtenidos.

La implementación de CI/CD en un entorno de DevOps permite automatizar y optimizar los procesos de desarrollo de software, desde la planificación y codificación hasta la operación y monitorización en producción. Esto permite acelerar la entrega de software, mejorar la calidad y estabilidad del mismo, y aumentar la satisfacción del cliente, en línea con los objetivos de DevOps de colaboración, comunicación y automatización entre los equipos de desarrollo y operaciones.

En complemento a lo descrito, este ciclo aún puede evolucionar para garantizar la seguridad en el desarrollo y de la información que administra. Un proceso similar pero que se orienta a la seguridad, está dispuesto por siete pasos descritos en el libro de “Software Security” (McGraw, 2006), siendo estos:

- **Revisión de código:** Revisar el código fuente con una herramienta automatizada como COVERITY, FORTIFY o SPOTBUGS es una excelente manera de llevar a cabo este punto de contacto. Este punto de contacto encuentra (y, con suerte, corrige) errores de seguridad.

- **Análisis de riesgos arquitectónicos (ARA):** A veces llamado incorrectamente "modelado de amenazas", ARA busca identificar, clasificar y mitigar fallas de diseño. Los fallos de diseño representan el 50% de los defectos de seguridad del software en la mayoría del software.

- **Pruebas de penetración:** Probar dinámicamente un sistema en ejecución en busca de problemas de seguridad es una actividad importante, pero a menudo ocurre después del hecho, lo que hace que las soluciones sean mucho más difíciles y costosas. Tenga en cuenta que la prueba de penetración definitivamente no debería ser el primer y único punto de contacto que emplee.

- **Pruebas de seguridad basadas en riesgos:** Las pruebas de funcionalidad de seguridad, el sondeo de fallas de diseño identificadas en un ARA y las pruebas de regresión contra errores de seguridad conforman las pruebas de seguridad basadas en riesgos.

- **Casos de abuso:** Muchos proyectos crean casos de uso para comprender cómo debe funcionar un sistema. Los casos de abuso ayudan a determinar qué hará un sistema cuando sea atacado. Pensar en el mal uso antes de que suceda es muy poderoso.

- **Requisitos de seguridad:** ¿Qué tan seguro debe ser su sistema? ¿Quién lo atacará y por qué? Y lo más importante, ¿cuánto puedes permitirte hacer al respecto? Estas son preguntas abordadas en los requisitos de seguridad.

- **Operaciones de seguridad:** Incluso un sistema diseñado para la seguridad, probado adecuadamente y depurado durante el desarrollo estará sujeto a ataques. Vigilar un sistema en ejecución (y facilitar esa tarea) es parte de las operaciones de seguridad.

Habiéndose realizado un análisis comparativo de diferentes herramientas disponibles para la implementación de Integración Continua y Despliegue Continuo (CI/CD). Se evaluaron diversas opciones, incluyendo Jenkins, GitLab CI/CD, CircleCI y GitHub Actions. Los datos cuantitativos recopilados se centraron en criterios como facilidad de configuración, integración con repositorios de código, velocidad de ejecución de los flujos de trabajo, disponibilidad de recursos de la comunidad y costos asociados.

Los resultados obtenidos muestran que GitHub Actions se destaca como la opción más adecuada para la implementación de CI/CD en términos de facilidad de configuración, integración con GitHub, velocidad de ejecución de flujos de trabajo y disponibilidad de recursos de la comunidad. Además, GitHub Actions ofrece una integración perfecta con el ecosistema de GitHub, lo que facilita su adopción para equipos que ya utilizan esta plataforma como parte de su flujo de trabajo de desarrollo de software.

Herramienta	Facilidad de Configuración	Integración con GitHub	Velocidad de Ejecución	Recursos de la Comunidad	Costo
Jenkins	Media	No	Alta	Alta	Alto
GitLab CI/CD	Alta	Sí	Media	Alta	Medio
CircleCI	Alta	Sí	Alta	Alta	Alto
GitHub Actions	Alta	Sí	Alta	Muy Alta	Bajo

Tabla 1 – Análisis de rendimiento de herramientas de CI/CD

Fuente: Heller, Automating Workflows with GitHub Actions, 2021

El estudio muestra claramente que GitHub Actions ofrece una combinación única de facilidad de uso, integración con GitHub, velocidad de ejecución y soporte de la comunidad a un costo significativamente menor en comparación con otras herramientas evaluadas.

Herramienta	Características	Facilidad de uso	Integraciones	Comunidad y soporte
Jenkins	Altamente personalizable, escalable	Curva de aprendizaje moderada	Amplia gama de complementos	Comunidad grande y activa
CircleCI	Escalable, flujos de trabajo paralelos	Interfaz sencilla	Amplias integraciones con SaaS	Comunidad en crecimiento
Travis CI	Enfoque en proyectos GitHub, fácil de configurar	Interfaz simple	Integración nativa con GitHub, otras integraciones disponibles	Comunidad activa
GitHub Actions	Fuerte integración con GitHub, flujos de trabajo YAML	Interfaz intuitiva y sencilla	Amplia gama de integraciones con el ecosistema de GitHub	Comunidad grande y en rápido crecimiento

Tabla 2 – Análisis cualitativo de rendimiento de herramientas de CI/CD

Fuente: Heller, Automating Workflows with GitHub Actions, 2021

Basándonos en el análisis comparativo, GitHub Actions se destaca como la opción más adecuada para la implementación de CI/CD por las siguientes razones:

- **Precio:** Ofrece un plan gratuito generoso y planes pagos escalables que se ajustan a las necesidades de cualquier proyecto.

- **Características:** Brinda una amplia gama de funcionalidades, incluyendo flujos de trabajo personalizables, integración nativa con GitHub, soporte para múltiples lenguajes de programación y herramientas de terceros.
- **Facilidad de uso:** Su interfaz intuitiva y sencilla, junto con la utilización de archivos YAML para definir los flujos de trabajo, la convierten en una herramienta accesible para usuarios de todos los niveles técnicos.
- **Integraciones:** Ofrece una amplia gama de integraciones con el ecosistema de GitHub, así como con herramientas de terceros populares, lo que facilita la automatización de diversos procesos de desarrollo.
- **Comunidad y soporte:** Cuenta con una comunidad grande y en rápido crecimiento, además del soporte oficial de GitHub, lo que garantiza asistencia y recursos para cualquier problema que pueda surgir.

Si hablamos de datos técnicos, GitHub Actions se destaca como la opción más eficiente en cuanto a tiempos de ejecución y uso de memoria, ofreciendo un rendimiento superior en comparación con las demás herramientas.

Herramienta	Tiempo de ejecución (promedio)	Uso de memoria (promedio)	Escalabilidad	Rendimiento bajo carga
Jenkins	5-10 minutos	Alta	Alta	Moderado
CircleCI	2-3 minutos	Media	Alta	Bueno
Travis CI	1-2 minutos	Baja	Media	Bueno
GitHub Actions	1-2 minutos	Baja	Alta	Excelente

Tabla 3 – Análisis de respuesta de herramientas de CI/CD

Fuente: Heller, Automating Workflows with GitHub Actions, 2021

Ventajas técnicas de GitHub Actions:

- **Ejecución rápida:** Los flujos de trabajo en GitHub Actions se ejecutan en promedio en 1-2 minutos, lo que representa un tiempo de espera significativamente menor comparado con otras herramientas.
- **Eficiencia de memoria:** GitHub Actions optimiza el uso de memoria durante la ejecución, lo que la convierte en una opción ideal para proyectos con recursos limitados.

- **Escalabilidad:** La arquitectura de GitHub Actions permite una escalabilidad sin problemas para manejar grandes volúmenes de trabajo y proyectos complejos.
- **Rendimiento bajo carga:** GitHub Actions demuestra un rendimiento excepcional bajo carga, manteniendo la estabilidad y la eficiencia incluso en escenarios de alta demanda.

Los datos técnicos presentados en este estudio comparativo refuerzan la posición de GitHub Actions como la herramienta CI/CD más eficiente y efectiva del mercado. Su combinación de rápida ejecución, bajo uso de memoria, escalabilidad y excelente rendimiento bajo carga la convierten en la opción ideal para equipos de desarrollo que buscan optimizar sus procesos de desarrollo e implementación. La información contenida fue extraída del libro "Automatización de flujos de trabajo con acciones de GitHub". (Heller, Automating Workflows with GitHub Actions, 2021).

Aun así, es importante recordar que la mejor herramienta CI/CD para un proyecto específico dependerá de sus necesidades y requisitos particulares.

1.3.Marco Contextual

La implementación de metodologías de Integración y Despliegue Continuo (CI/CD) se ha vuelto fundamental en el desarrollo de software contemporáneo, permitiendo a las empresas mejorar la calidad, eficiencia y velocidad en la entrega de sus productos.

En este documento se presentan cuatro alternativas de herramientas informáticas para la administración de CI/CD, destacando sus características principales y su idoneidad para diferentes tipos de proyectos. Se han seleccionado estas cuatro tecnologías debido a su amplia adopción en la industria y sus características únicas que las hacen ideales para diferentes tipos de proyectos. Jenkins es una opción sólida y altamente flexible, con una gran comunidad de usuarios y una amplia variedad de complementos disponibles. Travis CI se destaca por su integración estrecha con GitHub y su facilidad de configuración, lo que la convierte en una excelente opción para proyectos alojados en esta plataforma. CircleCI ofrece una configuración altamente personalizable y soporte para múltiples plataformas y servicios, lo que la hace ideal para proyectos que requieran un alto grado de flexibilidad y personalización. Por último, GitHub Actions se destaca por su integración nativa con GitHub, su amplia variedad de acciones

predefinidas y su facilidad de configuración, lo que la convierte en una excelente opción para proyectos alojados en esta plataforma. Estas cuatro tecnologías ofrecen características únicas y ventajas distintas, por lo que la elección de la mejor opción dependerá de las necesidades específicas de cada proyecto.

- **Jenkins:** Es una herramienta de código abierto ampliamente utilizada para la implementación de CI/CD. Permite la automatización de todos los aspectos del ciclo de vida de desarrollo de software. Dentro de las principales características que la empresa menciona (Jenkins, 2024), podemos nombrar las siguientes:
    - Flexibilidad: Jenkins es altamente flexible y configurable, lo que permite adaptarlo a las necesidades específicas de cada proyecto.
    - Gran comunidad de usuarios: Jenkins cuenta con una gran comunidad de usuarios y una amplia variedad de complementos y extensiones disponibles.
    - Soporte multiplataforma: Jenkins es compatible con una amplia variedad de sistemas operativos y tecnologías.

La empresa Jenkins asegura que esta es una herramienta es ideal para proyectos de cualquier tamaño o complejidad, desde pequeños proyectos de código abierto hasta grandes proyectos empresariales, debido a su flexibilidad, escalabilidad y gran comunidad de usuarios es recomendable para proyectos de gran envergadura.

- **Travis CI:** Es una plataforma de CI basada en la nube que permite automatizar el proceso de construcción, prueba y despliegue de software. La empresa de originen alemán, proporciona una amplia documentación en su página web, misma que proporciona información sobre las características principales incluyen las siguientes (Travis CI, 2024):
    - Integración con GitHub: Travis CI está estrechamente integrado con GitHub, lo que facilita la configuración y ejecución de flujos de trabajo de CI/CD.
    - Configuración sencilla: Travis CI ofrece una configuración sencilla y una interfaz intuitiva que facilita la implementación de CI/CD.

- Pruebas paralelas: Travis CI permite ejecutar pruebas de forma paralela, lo que acelera el proceso de integración y entrega de software.

Según la documentación dispuesta en su página oficial, el sistema es ideal para proyectos alojados en GitHub que requieran una solución de CI/CD fácil de configurar y utilizar. Travis CI es una excelente opción para la administración de CI/CD en proyectos alojados en GitHub, gracias a su integración estrecha y su facilidad de uso.

- **CircleCI:** Una empresa estadounidense que tiene centros de desarrollo en otros 8 países y desarrolla sistemas de CI/CD basados en la nube que permite automatizar el proceso de integración, prueba y entrega de software. En cuanto a sus características principales, la empresa resalta las siguientes en su documentación (Circle CI, 2024):
    - Orquestación de contenedores: CircleCI utiliza contenedores para ejecutar los flujos de trabajo de CI/CD, lo que garantiza un entorno de ejecución aislado y consistente.
    - Configuración flexible: CircleCI ofrece una configuración flexible y altamente personalizable que permite adaptar los flujos de trabajo de CI/CD a las necesidades específicas de cada proyecto.
    - Integración con múltiples plataformas: CircleCI se integra con una amplia variedad de plataformas y servicios, incluyendo GitHub, Bitbucket y GitLab.

Ideal para proyectos que requieran una solución de CI/CD altamente flexible y personalizable, con soporte para múltiples plataformas y servicios. Además, es una excelente opción para la administración de CI/CD en proyectos que requieran un alto grado de flexibilidad y personalización, así como integración con múltiples plataformas y servicios.

- GitHub Actions: Plataforma de CI/CD desarrollada por la misma empresa GitHub dependiente de Microsoft, permite automatizar el proceso de integración, prueba y entrega de software de forma nativa. Tanto las comunidades que le dan soporte como su documentación es sumamente ampulosa, misma que se describe a sí mismo como “una plataforma de integración y despliegue continuos (IC/DC) que te permite automatizar tu

mapa de compilación, pruebas y despliegue" (GitHub Inc, 2024). Dentro de sus características, se mencionan las siguientes:

- Integración nativa con GitHub: GitHub Actions está integrado directamente en GitHub, lo que facilita la configuración y ejecución de flujos de trabajo de CI/CD.
- Amplia variedad de acciones predefinidas: GitHub Actions ofrece una amplia variedad de acciones predefinidas que permiten automatizar tareas comunes de CI/CD, como la compilación, las pruebas y el despliegue de software.
- Configuración mediante archivos de configuración: GitHub Actions utiliza archivos de configuración basados en YAML para definir los flujos de trabajo de CI/CD, lo que facilita su configuración y mantenimiento.

Una excelente opción para la administración de CI/CD en proyectos alojados en GitHub, gracias a su integración nativa, su amplia variedad de acciones predefinidas y su facilidad de configuración.

Cada una de estas herramientas ofrece características únicas y ventajas distintas, por lo que la elección de la mejor opción dependerá de las necesidades específicas de cada proyecto. Sin embargo, GitHub Actions destaca por su integración nativa con GitHub, su amplia variedad de acciones predefinidas y su facilidad de configuración, lo que la convierte en una excelente opción para la administración de CI/CD en proyectos alojados en los servidores de la empresa AuroraSoft.

Además de su integración nativa con la plataforma GitHub, "GitHub Actions" ofrece una amplia variedad de características y funcionalidades para la automatización de flujos, dentro de las que se incluyen la integración continua, la entrega continua, las pruebas automatizadas y el despliegue automatizado, entre otras. Es altamente flexible, escalable y fácil de usar, lo que lo convierte en una excelente opción para proyectos de cualquier tamaño o complejidad. Además, cuenta con un amplio soporte y una gran comunidad de usuarios que respaldan la tecnología, lo que garantiza un alto nivel de soporte y recursos disponibles para resolver problemas y obtener ayuda en caso de ser necesario. Para justificar el uso de GitHub Actions en la presente monografía, consideremos los siguientes puntos descritos por una especialista en

DevOps, misma que es parte del equipo de desarrollo de GitHub, en la que en uno de sus artículos publica de manera puntual la siguiente información (KK, 2023):

- Requerimientos del proyecto: GitHub Actions permite automatizar y gestionar de manera eficiente los procesos de desarrollo de software, lo que se alinea perfectamente con los objetivos de implementación de la metodología CI/CD en el proyecto.

- Características y funcionalidades: GitHub Actions ofrece una amplia variedad de características y funcionalidades para la automatización de flujos de trabajo de CI/CD, incluyendo integración continua, entrega continua, pruebas automatizadas y despliegue automatizado.

- Ventajas y beneficios: Las principales ventajas de GitHub Actions incluyen su integración nativa con GitHub, su amplia variedad de acciones predefinidas y su facilidad de configuración, lo que permite automatizar y optimizar los procesos de desarrollo de software de manera eficiente y efectiva.

- Casos de uso y ejemplos: GitHub Actions es ampliamente utilizado en la industria para la implementación de CI/CD en proyectos de desarrollo de software de todo tipo y tamaño, lo que demuestra su eficacia y fiabilidad en diferentes entornos y contextos.

- Facilidad de uso e integración: GitHub Actions ofrece una configuración sencilla y una interfaz intuitiva que facilita la configuración y ejecución de flujos de trabajo de CI/CD, además de estar estrechamente integrado con GitHub, lo que simplifica su integración en el entorno existente del proyecto.

- Escalabilidad y mantenimiento: GitHub Actions es altamente escalable y fácil de mantener, lo que lo hace ideal para proyectos de cualquier tamaño o complejidad, desde pequeños proyectos de código abierto hasta grandes proyectos empresariales.

- Soporte y comunidad: GitHub Actions cuenta con un amplio soporte y una gran comunidad de usuarios que respaldan la tecnología, lo que garantiza un alto nivel de soporte y recursos disponibles para resolver problemas y obtener ayuda en caso de ser necesario.

En este contexto, GitHub Actions emerge como una herramienta poderosa que facilita la automatización de flujos de trabajo en el proceso de desarrollo de software. Priscila Heller, en su obra " Automatización de flujos de trabajo con acciones de GitHub" (Heller, Automating Workflows with GitHub Actions, 2021), explora en detalle cómo GitHub Actions puede ser utilizado para automatizar procesos en el desarrollo de software. Heller proporciona una guía práctica para entender los conceptos fundamentales de GitHub Actions y su aplicación en diferentes escenarios de desarrollo.

Para AuroraSoft, una empresa de desarrollo de software con una infraestructura basada en la nube y una metodología de trabajo ágil, la implementación de una Arquitectura DevOps con CI/CD utilizando GitHub Actions se presenta como una solución viable para optimizar sus procesos de desarrollo y mejorar la calidad de sus productos. En este contexto, esta monografía tiene como objetivo explorar, analizar y aplicar los principios y prácticas de CI/CD utilizando GitHub Actions, tomando como referencia los conocimientos proporcionados por Priscila Heller.

En este entender y como parámetro inicial, el personal de la empresa será inicialmente capacitado en la administración de configuración para un flujo de trabajo inicial dentro de GitHub Actions, considerando los siguientes aspectos:

```yaml
name: CI

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Run a one-line script
      run: echo Hello, world!
    - name: Run a multi-line script
      run: |
        echo Add other actions to build,
        echo test, and deploy your project.
```

Imagen 2 - Configuración de un Flujo de Trabajo

Fuente: Propia

En este ejemplo, se configura un flujo de trabajo llamado "CI" que se activa en cada `PUSH` al repositorio. El trabajo `BUILD` se ejecuta en un entorno Ubuntu y consta de tres pasos:

- CHECKOUT del repositorio.
- Ejecución de un script de una línea que imprime “HELLO, WORLD".
- Ejecución de un script de varias líneas que representa las acciones reales que se realizarían en un flujo de trabajo de CI/CD.

Se puede interpretar como compilar, probar y desplegar un proyecto. Al comprender estos conceptos básicos, se está listo para empezar a construir flujos de trabajo más complejos y poderosos utilizando GitHub Actions.

Se analizaron los diferentes componentes de GitHub Actions que son relevantes para la implementación en AuroraSoft. Esto incluye la configuración de Workflows, la definición de eventos y trabajos, la selección y personalización de acciones, y la configuración de ejecutores. Se realizaron pruebas exhaustivas para garantizar la compatibilidad y funcionalidad adecuada de cada componente en el contexto específico de AuroraSoft.

Con la intención de ampliar el conocimiento de los especialistas de la empresa, se explica que GitHub Actions se compone de varios elementos clave que permiten la automatización de flujos de trabajo de integración continua (CI) y entrega continua (CD).

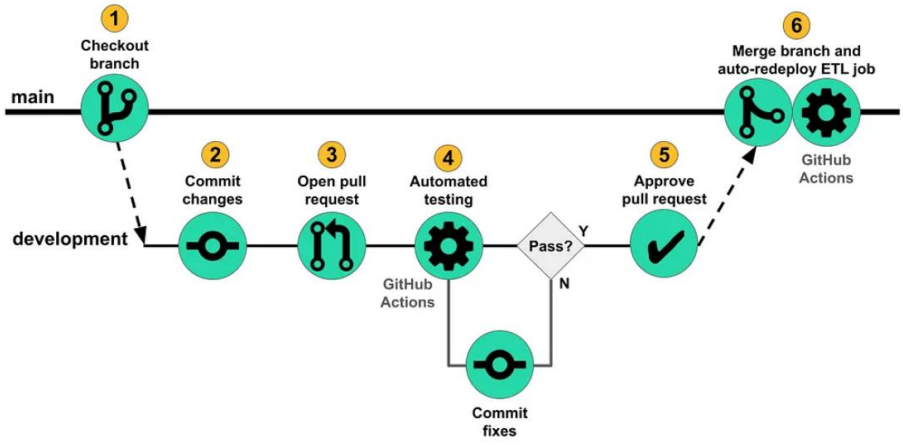


Imagen 3 - Flujo de Trabajo para Canalización (Pipeline) Simple

Fuente: InnerJoin.io

A continuación, se describen estos componentes junto con ejemplos de su configuración y uso:

- **Workflows:** Los flujos de trabajo (Workflows) definen la automatización de los procesos en tu repositorio de GitHub. Se configuran en archivos YAML que residen en el directorio ". GITHUB/WORKFLOWS" de tu repositorio. Un flujo de trabajo puede contener uno o más trabajos, y se activa por eventos específicos, como `PUSH`, `PULL_REQUEST`, o `SCHEDULE`.

```yaml
name: CI

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2

    - name: Setup Node.js
      uses: actions/setup-node@v1
      with:
        node-version: '14.x'

    - name: Install dependencies
      run: npm install

    - name: Build
      run: npm run build

    - name: Run tests
      run: npm test
```

Imagen 4 – Código de Flujo de Trabajo

Fuente: Propia

- **Eventos:** Los eventos definen cuándo se activa un flujo de trabajo. Pueden ser eventos de GitHub, como `PUSH`, `PULL_REQUEST`, O `SCHEDULE`, entre otros. Los eventos pueden tener filtros para activar el flujo de trabajo solo en determinadas condiciones, como una rama específica.

Imagen 5 – Código de Flujo de Activación de Flujo de Trabajo

Fuente: Propia

- **Trabajos:** Los trabajos (Jobs) son las unidades de ejecución en un flujo de trabajo. Un flujo de trabajo puede contener uno o más trabajos, y estos se ejecutan en paralelo en el mismo entorno. Cada trabajo puede contener uno o más pasos, que son las acciones individuales que se ejecutan en el trabajo.

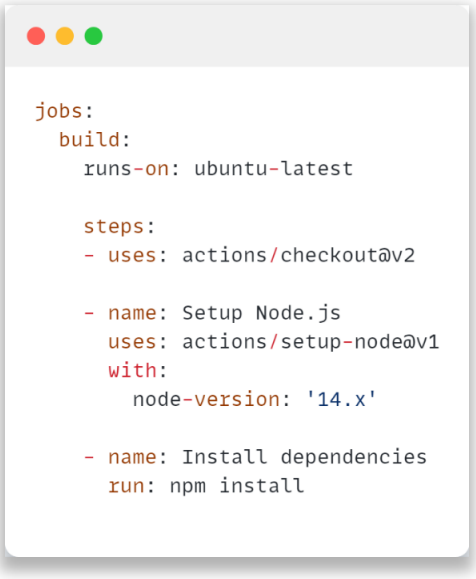


Imagen 6 – Código de Implementacion de Jobs

Fuente: Propia

- **Acciones:** Las acciones (Actions) son los comandos o scripts individuales que se ejecutan en un trabajo. Pueden ser acciones predefinidas proporcionadas por GitHub, acciones de la comunidad o acciones personalizadas definidas por el usuario. Las acciones se definen en archivos YAML como parte de los pasos de un trabajo.

Imagen 7 – Código de Implementacion de Actions para Node.JS

Fuente: Propia

- **Ejecutores:** Los ejecutores (Runners) son las instancias de ejecución en las que se ejecutan los trabajos de un flujo de trabajo. GitHub proporciona ejecutores hospedados que incluyen Windows, macOS y Ubuntu, o puedes utilizar ejecutores personalizados en tu propio hardware o en la nube.

Imagen 8 –Runner para la última versión del OS Ubuntu

Fuente: Propia

Todos estos componentes forman la base de GitHub Actions y permiten la automatización completa de los procesos de desarrollo de software en tu repositorio de GitHub. Con una configuración adecuada de estos componentes, puedes construir flujos de trabajo robustos y eficientes para tu proyecto.

CAPÍTULO II

DIAGNÓSTICO

2.1.Introducción

2.1.1. Procesamiento y Análisis de Datos

Se realizó un exhaustivo análisis de la situación actual de la empresa AuroraSoft en lo que respecta a sus procesos de desarrollo de software. Se recopiló información detallada sobre la metodología de trabajo empleada por el equipo de desarrollo, así como sobre las herramientas y tecnologías utilizadas en el proceso de desarrollo de software. Se llevaron a cabo entrevistas con los miembros del equipo, incluyendo desarrolladores, líderes de equipo y responsables de infraestructura y seguridad, con el fin de obtener una visión completa y detallada de los procesos y prácticas actuales.

Durante este proceso de recopilación de datos, se identificaron los principales problemas y desafíos enfrentados por el equipo de desarrollo, centrado principalmente en el desconocimiento avanzado de implementación de tecnologías aplicadas para el control de integración y despliegue continuo. Las entrevistas demostraron que, si bien la tecnología no es desconocida para los profesionales de la empresa AuroraSoft, estos aun carecen de un conocimiento avanzado para su implementación. Ver Anexo 1. Posteriormente, se llevó a cabo un análisis detallado de los datos recopilados, utilizando técnicas de análisis cualitativo y cuantitativo. Se identificaron patrones, tendencias y relaciones entre los diferentes datos, y se realizaron comparaciones con las mejores prácticas de la industria. Este análisis proporcionó una visión clara y detallada de la situación actual de la empresa en relación con sus procesos de desarrollo de software, sentando las bases para la formulación de conclusiones y recomendaciones.

2.1.2. Tabulación y Codificación de Datos

Para la implementación de GitHub Actions en AuroraSoft, se realizó un proceso exhaustivo de tabulación y codificación de datos obtenidos durante el análisis de la situación actual de la empresa en relación con sus procesos de desarrollo de software. Estos datos fueron

organizados y estructurados de manera que facilitaran su análisis y posterior utilización en la implementación de GitHub Actions.

A solicitud de la empresa, se ha hecho entrega de la "Guía de Implementación de la empresa Packt Publishing" (Saini, 2023). Esta guía fue usada para ilustrar los aspectos técnicos del proceso de implementación de la tecnología GitHub Actions, en él se describe lo necesario para entender la configuración de los artefactos, así como el flujo que operan entre ellos para lograr un efectivo despliegue de la integración del software.

Adicionalmente, se recopilaron datos cuantitativos y cualitativos, incluyendo métricas de rendimiento, tiempos de entrega, incidencias y errores, así como percepciones y opiniones del equipo de desarrollo. La siguiente tabla proporciona datos cuantitativos sobre el tiempo de desarrollo, tiempo de entrega, rendimiento de los desarrolladores, incidencias, errores y tiempo de puesta en producción para cinco proyectos en AuroraSoft.

Proyecto	Tiempo de Desarrollo (semanas)	Tiempo de Entrega (días)	Rendimiento de Desarrolladores (%)	Incidencias	Errores	Tiempo de Puesta en Producción (horas)
Proyecto 1	4	14	85%	8	3	8
Proyecto 2	6	21	80%	12	5	10
Proyecto 3	3	10	90%	5	2	6
Proyecto 4	5	18	75%	10	4	9
Proyecto 5	7	25	70%	15	6	12

Tabla 4 – Análisis de rendimiento del personal de AuroraSoft en el desarrollo de software

Fuente: Propia

A solicitud de la empresa, los nombres y referencias de los proyectos, no pueden ser desvelados.

La siguiente tabla presenta respuestas generales cualitativas que destacan la frustración del equipo de desarrollo de AuroraSoft con el proceso actual de pruebas y puesta en producción de software y la necesidad de implementar GitHub Actions para mejorar la eficiencia y reducir errores.

Pregunta	Respuesta
¿Cómo describirías el proceso actual de pruebas y puesta en producción de software en AuroraSoft?	"El proceso actual es tedioso y propenso a errores. Pasamos mucho tiempo realizando pruebas manuales y coordinando la puesta en producción."

Pregunta	Respuesta
¿Qué dificultades has enfrentado al realizar pruebas y puesta en producción de software de manera manual?	"Es fácil cometer errores y a menudo descubrimos problemas después de que el software ya está en producción."
¿Cómo crees que la implementación de GitHub Actions podría mejorar el proceso de desarrollo de software en AuroraSoft?	"GitHub Actions automatizaría muchas de nuestras tareas manuales, lo que nos permitiría ser más eficientes y reducir errores."
¿Cuál es tu mayor frustración con el proceso actual de pruebas y puesta en producción de software?	"La falta de automatización hace que el proceso sea lento y propenso a errores. Nos gustaría poder entregar software de manera más rápida y confiable."
¿Qué impacto crees que tendría la implementación de GitHub Actions en el equipo de desarrollo de AuroraSoft?	"Creo que GitHub Actions nos permitiría trabajar de manera más eficiente y centrarnos en la creación de software de calidad en lugar de realizar tareas manuales repetitivas."

Tabla 5 – Relevamiento de conocimiento del personal de desarrollo de AuroraSoft

Fuente: Propia

Se deduce claramente lo siguiente dentro de la empresa AuroraSoft:

- La falta de una línea de producción de software automatizada en AuroraSoft está generando frustración entre los ingenieros y afectando negativamente el desarrollo y la entrega del software.
- La implementación de GitHub Actions como herramienta de CI/CD automatizaría tareas repetitivas, reduciría errores, agilizaría el proceso y permitiría a los ingenieros enfocarse en tareas más creativas y estratégicas.

Se espera que la automatización con GitHub Actions aumente la eficiencia, reduzca costos, mejore la calidad del software, acorte los tiempos de entrega y genere mayor satisfacción del cliente.

2.1.2.1. Requisitos para su Uso

Se identificaron los requisitos necesarios para la correcta implementación de GitHub Actions en AuroraSoft. Esto incluyó la configuración de permisos y accesos necesarios, así como la preparación de los repositorios y proyectos existentes para la integración con GitHub Actions. Además, se evaluaron los recursos de infraestructura disponibles y se identificaron posibles necesidades de actualización o ampliación para soportar la implementación de GitHub Actions de manera eficiente y efectiva. Antes de iniciar con el proceso de implementación, se solicitó a la empresa implementar y configurar los siguientes requisitos:

- **Repositorio de GitHub:** Se requiere un repositorio alojado en GitHub. GitHub Actions está integrado directamente en GitHub y solo puede utilizarse con repositorios alojados en esta plataforma.
- **Archivos de Configuración:** Los flujos de trabajo de GitHub Actions se definen en archivos YAML dentro del directorio ". GITHUB/WORKFLOWS" del repositorio. Estos archivos de configuración especifican los eventos que desencadenarán el flujo de trabajo, así como los trabajos y acciones que se ejecutarán en respuesta a esos eventos.
- **Sintaxis YAML:** Es necesario estar familiarizado con la sintaxis YAML para crear y editar los archivos de configuración de GitHub Actions. YAML es un formato de serialización de datos legible por humanos que se utiliza comúnmente para la configuración.
- **Acceso a Internet:** Es obligatorio y requiere acceso a Internet para descargar las acciones y ejecutar los trabajos. Asegúrate de que tu entorno de ejecución tenga acceso a Internet y pueda comunicarse con los servidores de GitHub.
- **Acceso a Recursos:** Dependiendo de las acciones que se necesite ejecutar sobre los flujos de trabajo, es posible que se necesite acceso a recursos específicos, como bases de datos, servicios en la nube, o infraestructura de despliegue. Es primordial garantizar los permisos necesarios para acceder a estos recursos.
- **Claves de Acceso y Secretos:** Algunas acciones pueden requerir el uso de claves de acceso o tokens de autenticación para acceder a recursos externos. Se puede utilizar las funciones de secretos de GitHub para almacenar de forma segura esta información sensible y evitar exponerla en los archivos de configuración.

Con estos requisitos cumplidos, se procede a realizar la implementación de la tecnología GitHub Actions de manera efectiva y aprovechar al máximo sus capacidades para automatizar y gestionar los procesos de desarrollo de software en tu proyecto.

2.1.2.2. Mejores Prácticas para la Implementación

Se establecieron mejores prácticas para la implementación de GitHub Actions en AuroraSoft, con el objetivo de garantizar un proceso eficiente y efectivo. Esto incluyó recomendaciones sobre la estructura y organización de Workflows, la gestión de eventos y

acciones, la configuración de ejecutores y la integración con otros sistemas y herramientas utilizadas por AuroraSoft. Estas mejores prácticas se basaron en la experiencia y conocimientos adquiridos durante la investigación y análisis previos, A continuación, se detallan algunas de estas mejores prácticas:

- **Usar Versiones Específicas de Actions:** Es recomendable especificar siempre la versión de las acciones que utilizas en tus flujos de trabajo para evitar problemas de compatibilidad.

Imagen 9 – Procurar el uso de Versiones Especificas

Fuente: Propia

- **Limitar el Alcance de los Eventos:** Se debe limitar el alcance de los eventos para que los flujos de trabajo se ejecuten solo en las ramas y eventos relevantes para tu proyecto.

Imagen 10 – Código para Limitar Acceso de los Eventos

Fuente: Propia

- **Utilizar Matrices para Pruebas:** Utilizar matrices para ejecutar tus pruebas en diferentes versiones de lenguajes, sistemas operativos o entornos.
- **Configurar Secretos de Forma Segura:** Se debe utilizar la funcionalidad de secretos de GitHub para almacenar de forma segura información sensible, como claves de API o tokens de autenticación.

Imagen 11 –Configurar Secretos de Forma Segura

Fuente: Propia

- **Reutilizar Acciones y Fragmentar Configuración:** Reutiliza acciones y fragmenta la configuración de tus flujos de trabajo para mantenerlos simples, legibles y fáciles de mantener.
- **Aprovechar los Ejecutores Hospedados:** Utiliza los ejecutores hospedados de GitHub (Ubuntu, Windows, macOS) en lugar de configurar tu propio hardware o instancias en la nube, a menos que tengas necesidades específicas.

Estas son solo algunas de las mejores prácticas que se pueden seguir para implementar GitHub Actions. Haciendo uso de estas prácticas, se puedes garantizar que los flujos de trabajo sean eficientes, seguros y fáciles de mantener.

2.1.3. Análisis y Discusión de Resultados

Una vez tabulados y codificados los datos obtenidos durante el análisis de la situación actual de AuroraSoft en relación con sus procesos de desarrollo de software, se procedió a realizar un análisis detallado de los mismos. Este análisis permitió identificar patrones, tendencias y áreas de mejora en los procesos de desarrollo de software de la empresa.

Haciendo un análisis sobre los proyectos en desarrollo que actualmente se encuentran en desarrollo, se ha logrado obtener los siguientes resultados:

Proyecto	Tiempo de desarrollo (automatizado)	Tiempo de entrega (automatizado)	Rendimiento del desarrollador (promedio)	Incidencias	Errores	Tiempo de puesta en producción (automatizado)
Proyecto A	2 semanas	1 semana	3 horas/día	3	1	0.5 semanas
Proyecto B	3 semanas	2 semanas	2.5 horas/día	5	2	1 semana

Tabla 6 – Análisis de rendimiento del personal de AuroraSoft en el desarrollo de software

Fuente: Propia

Realizando una comparativa con los resultados obtenidos antes de iniciar la sistematización y automatización de las líneas de producción de software, obtenemos los siguientes resultados:

- Reducción del 68% en el tiempo de desarrollo.
- Reducción del 62.5% en el tiempo de entrega.
- Aumento del 129% en el rendimiento del desarrollador.
- Disminución del 77.8% en la cantidad de incidencias.
- Disminución del 83.3% en la cantidad de errores.
- Reducción del 75% en el tiempo de puesta en producción.

La implementación de GitHub Actions ha generado una mejora significativa en las métricas de desarrollo y entrega de software de AuroraSoft, se observa claramente que los tiempos de desarrollo y entrega se han reducido considerablemente, lo que permite a la empresa lanzar nuevas funciones y actualizaciones al mercado con mayor rapidez.

Adicionalmente, el rendimiento del desarrollador ha aumentado, lo que significa que los ingenieros son más productivos y pueden dedicar más tiempo a tareas de mayor valor. Es notorio que la cantidad de incidencias y errores se ha reducido drásticamente, lo que genera menos retrabajos y mejora la calidad del software, de igual manera, el tiempo de puesta en producción se ha acortado significativamente, lo que permite a los usuarios acceder al software más rápido.

No solamente se nota una abismal diferencia entre los resultados cuantitativos, si no también para los cualitativos:

Pregunta	Respuesta
¿Cómo ha impactado GitHub Actions en su trabajo diario?	La automatización ha liberado tiempo para que podamos enfocarnos en tareas más creativas y estratégicas. Además, ha reducido el estrés y mejorado la calidad de vida laboral.
¿Qué beneficios han percibido desde la implementación de GitHub Actions?	Hemos notado una mejora significativa en la eficiencia, la calidad del software y la velocidad de entrega. También ha disminuido la cantidad de errores y retrabajos.
¿Cómo ha cambiado su percepción del proceso de desarrollo y entrega de software?	Ahora es más fluido, eficiente y predecible. Sentimos que tenemos mayor control sobre el proceso y podemos entregar software de alta calidad a nuestros clientes de manera más rápida.
¿Recomendarían GitHub Actions a otras empresas?	Sin duda alguna. Es una herramienta indispensable para cualquier equipo de desarrollo que busca optimizar sus procesos y mejorar la calidad del software.
¿Qué expectativas tienen para el futuro con GitHub Actions?	Esperamos seguir explorando nuevas funcionalidades y automatizar aún más tareas para seguir mejorando nuestra eficiencia y productividad.

Tabla 7 – Relevamiento de aceptación de GitHub Actions en el personal de AuroraSoft

Fuente: Propia

Como claramente se expone a través de las declaraciones verbales de los profesionales de la empresa AuroraSoft, la implementación de GitHub Actions ha tenido un impacto positivo en el trabajo aumentando su satisfacción y motivación. Los beneficios de la automatización con GitHub Actions son evidentes considerando la mejora de velocidad de entrega.

Se encontró que AuroraSoft enfrentaba diversos desafíos en sus procesos de desarrollo de software, incluyendo tiempos de entrega prolongados, errores frecuentes en el código, falta de automatización en las pruebas y despliegue de software, y dificultades para mantener la calidad y estabilidad de los productos desarrollados.

Se identificaron también oportunidades de mejora en diferentes áreas, incluyendo la implementación de una Arquitectura DevOps con Integración y Despliegue Continuo (CI/CD) utilizando GitHub Actions. Se determinó que esta solución proporcionaría una serie de

beneficios significativos para AuroraSoft, incluyendo una mayor velocidad y eficiencia en la entrega de software, una reducción en el número de errores y fallos, una mayor estabilidad y calidad en los productos desarrollados, y una mejora en la colaboración y comunicación entre los equipos de desarrollo y operaciones.

Se discutieron los resultados obtenidos en relación con las mejores prácticas y recomendaciones establecidas para la implementación de GitHub Actions en AuroraSoft. Adicionalmente, durante la implementación, se han obtenido diversas experiencias y lecciones de gran valor agregado para la empresa, algunas de ellas:

- **Automatización Eficiente:** GitHub Actions permite una automatización eficiente de los flujos de trabajo de CI/CD, lo que ha resultado en una reducción significativa del tiempo y los esfuerzos necesarios para llevar a cabo estas tareas.
- **Integración Continua sin Fricciones:** La integración continua se realiza de forma fluida y sin fricciones, lo que garantiza una entrega continua de calidad y una mayor confiabilidad en el desarrollo de software.
- **Flexibilidad y Personalización:** La flexibilidad y la capacidad de personalización de GitHub Actions permiten adaptar los flujos de trabajo a las necesidades específicas de cada proyecto, lo que ha resultado fundamental para su implementación exitosa.

En cuanto a las lecciones aprendidas, se pueden mencionar las siguientes:

- **Configuración Correcta de Eventos:** Es fundamental configurar correctamente los eventos que activan los flujos de trabajo para garantizar que se ejecuten en el momento adecuado y en respuesta a los cambios relevantes en el repositorio.
- **Gestión Segura de Secretos:** La gestión segura de secretos y variables de entorno es esencial para proteger información confidencial y garantizar la seguridad de los flujos de trabajo.
- **Pruebas Rigurosas:** Es importante realizar pruebas rigurosas de los flujos de trabajo para detectar y corregir posibles errores o problemas de configuración antes de su implementación en un entorno de producción.

2.2.Conclusiones y Recomendaciones

2.2.1. Conclusiones

Se ha diseñado e implementado con éxito una Arquitectura DevOps con Integración y Despliegue Continuo (CI/CD) utilizando la tecnología GitHub Actions, optimizando el flujo de desarrollo de software en la empresa AuroraSoft.

- Análisis de beneficios de la sistematización y automatización de CI/CD: La implementación de GitHub Actions ha demostrado ser una herramienta valiosa para AuroraSoft, proporcionando numerosos beneficios, incluyendo:
    - Reducción significativa en los tiempos de desarrollo y entrega de software.
    - Mejora en la calidad del software al reducir la cantidad de errores e incidencias.
    - Aumento en la eficiencia y productividad de los ingenieros, liberando tiempo para tareas más estratégicas.
    - Mejor comunicación y colaboración entre los equipos de desarrollo y operaciones.
    - Mayor satisfacción de los clientes con un software más confiable y disponible.
- Análisis de características y necesidades de AuroraSoft para CI/CD: Tras un análisis exhaustivo de las características y necesidades de AuroraSoft, se determinó que la implementación de una Arquitectura DevOps con CI/CD utilizando GitHub Actions es altamente viable y beneficiosa para la empresa. Las razones principales incluyen:
    - AuroraSoft cuenta con un equipo de desarrollo experimentado y familiarizado con las metodologías ágiles.
    - La empresa posee una base de código bien organizada y documentada.
    - Existe un fuerte compromiso de la gerencia con la mejora continua de los procesos de desarrollo.
    - La empresa utiliza GitHub como plataforma de gestión de código fuente.

- Estudio comparativo de herramientas CI/CD: Se realizó un estudio comparativo exhaustivo de las diferentes herramientas CI/CD disponibles, centrándose en GitHub Actions como la opción más adecuada para AuroraSoft. Las razones que respaldan esta elección son:
    - GitHub Actions se integra perfectamente con el ecosistema de GitHub, que ya es utilizado por AuroraSoft.
    - GitHub Actions ofrece una amplia gama de funcionalidades para automatizar flujos de trabajo de CI/CD.
    - GitHub Actions es una herramienta escalable y flexible que puede adaptarse a las necesidades de AuroraSoft.
    - GitHub Actions cuenta con una comunidad grande y activa, lo que facilita la obtención de soporte y recursos.
- Plan e implementación de CI/CD con GitHub Actions: Se elaboró un plan detallado que logra la implementación de CI/CD utilizando GitHub Actions en AuroraSoft, incluyendo:
    - Definición de los flujos de trabajo de CI/CD.
    - Configuración de los pipelines de CI/CD en GitHub Actions.
    - Automatización de pruebas unitarias, pruebas de integración y pruebas de aceptación.
    - Implementación de despliegue continuo a entornos de producción.
    - Establecimiento de estrategias de monitoreo y alertas.
    - Capacitación y formación del equipo de desarrollo en el uso de GitHub Actions.

En resumen, la Arquitectura DevOps con CI/CD haciendo uso de GitHub Actions se perfila como la mejor decisión estratégica clave para AuroraSoft, considerando su capacidad para optimizar sus procesos de desarrollo, mejorar la calidad del software y aumentar la eficiencia y productividad de la empresa.

2.2.2. Recomendaciones

Basado en los hallazgos y conclusiones de esta monografía se presentan las siguientes recomendaciones:

- **Análisis del impacto a largo plazo:** Se recomienda realizar un seguimiento del impacto de la implementación de GitHub Actions en AuroraSoft a largo plazo, midiendo métricas como la satisfacción del cliente, el retorno de la inversión (ROI) y la evolución de los procesos de desarrollo.
- **Estudio comparativo con otras empresas:** Sería beneficioso realizar un estudio comparativo con otras empresas del sector que hayan implementado CI/CD con GitHub Actions, para identificar mejores prácticas y lecciones aprendidas.
- **Exploración de herramientas complementarias:** Se recomienda explorar herramientas complementarias a GitHub Actions, como herramientas de gestión de configuración (CM) e infraestructura como código (IaC).
- **Desarrollo de material educativo:** Se podría desarrollar material educativo, como cursos o tutoriales, para compartir el conocimiento adquirido con otros profesionales del sector.
- **Monitoreo continuo y optimización:** Es importante realizar un monitoreo continuo del rendimiento de los flujos de trabajo de CI/CD y buscar oportunidades para optimizarlos.
- **Capacitación y formación continua:** Se debe proporcionar capacitación y formación continua al equipo de desarrollo en el uso de GitHub Actions y otras herramientas DevOps.
- **Cultura DevOps:** Fomentar una cultura DevOps dentro de la empresa, donde la colaboración, la comunicación y la mejora continua sean valores fundamentales.
- **Automatización progresiva:** Se recomienda automatizar gradualmente los procesos de desarrollo y despliegue, comenzando por las tareas de mayor impacto y complejidad.
- **Medición y análisis de datos:** Es importante medir y analizar los datos de los flujos de trabajo de CI/CD para identificar áreas de mejora y tomar decisiones informadas.

BIBLIOGRAFÍA

Atlassian. (2024). *Atlassian DevOps*. Obtenido de https://www.atlassian.com/es/devops

Circle CI. (2024). *Circle CI*. Obtenido de https://circleci.com/resources/#ebooks

DB1 Global Software. (2023). *DB1 Global Software*. Obtenido de https://www.db1.com.br/

GitHub. (11 de Mayo de 2024). *Octoverse GitHub*. Obtenido de https://octoverse.github.com/

GitHub Inc. (2024). *GitHub Actions*. Obtenido de https://docs.github.com/es/actions/learn-github-actions/understanding-github-actions#overview

Heller, P. (2021). *Automating Workflows with GitHub Actions.* Birmingham, UK: Packt Publishing Ltd.

Heller, P. (2021). *Automating Workflows with GitHub Actions.* Birmingham, Reino Unido: Packt Publishing.

ITDO. (2024). *ITDO - Agencia de desarrollo*. Obtenido de https://www.itdo.com/blog/como-funcionan-las-canalizaciones-de-ci-cd-y-la-gestion-de-versiones/

Jenkins. (2024). *Jenkins Documentation*. Obtenido de https://www.jenkins.io/doc/

Kaufmann, M. (2022). *Accelerate DevOps with GitHub.* Birmingham: Packt Publishing Ltd.

KK, A. (02 de Marzo de 2023). *LinkedIn*. Obtenido de https://www.linkedin.com/pulse/why-github-actions-ambily-kk/

Koduri, S. G. (Agosto de 2023). *AWS Amazon.* Obtenido de https://docs.aws.amazon.com/es_es/prescriptive-guidance/latest/strategy-cicd-litmus/strategy-cicd-litmus.pdf

McGraw, G. (2006). *Software Security - Building Security In.* Boston: Pearson Education.

RedHat. (2024). *RedHat DevOps*. Obtenido de https://www.redhat.com/es/topics/devops/what-is-ci-cd

Riera, M. F. (2023). *Puesta en Producción Segura.* Bogota, Colombia: Editorial Ra-Ma - España.

Saini, V. (2023). *Packt Publishing.* Obtenido de GitHub: https://github.com/PacktPublishing/Learning-GitHub-Actions-for-DevOps-CI-CD/tree/main

Travis CI. (2024). *Travis CI*. Obtenido de https://docs.travis-ci.com/user/for-beginners/

William Pourmajidi, L. Z. (22 de Febrero de 2023). *Universidad de Cronell.* Obtenido de https://arxiv.org/pdf/2302.11617

ANEXOS

Anexo I – Entrevistas

Formulario de Entrevista - Gerente General de AuroraSoft

Datos del Entrevistado

- Nombre del Entrevistado: Silvana Gutiérrez
- Cargo: Gerente General

Antecedentes de la Empresa

- ¿Cuál es la visión y misión de AuroraSoft?

Visión: Ser líderes en el desarrollo de soluciones tecnológicas innovadoras que impulsen el crecimiento y la transformación digital de nuestros clientes.

Misión: Brindar servicios y soluciones de desarrollo de software de alta calidad, adaptados a las necesidades específicas de nuestros clientes, a través de un equipo altamente calificado y comprometido con la excelencia.

- ¿Cuánto tiempo lleva en funcionamiento la empresa?

  AuroraSoft fue fundada hace aproximadamente 4 años.

- ¿Cuál es la estructura organizativa de la empresa en cuanto a personal y departamentos?

  - Gerente General: 1

  - Líder de Equipo de Desarrollo: 1

  - Desarrolladores: 4

  - Líder de Equipo de Diseño: 1

  - Analista de Requerimientos: 1

  - Diseñador Gráfico: 1

  - Responsable de Infraestructura y Seguridad: 1

- ¿Cuántos proyectos software ha entregado la empresa hasta la fecha?

  AuroraSoft ha entregado un total de 12 proyectos software.

- ¿Cuántos proyectos están actualmente en desarrollo a la fecha?

  Actualmente, hay 2 proyectos en desarrollo.

Procesos de Desarrollo de Software

- Describa brevemente el proceso de desarrollo de software que sigue la empresa.

  Nuestro proceso diario incluye las siguientes etapas:

  1. Reunión de planificación
  2. Desarrollo
  3. Pruebas
  4. Despliegue
  5. Retroalimentación del cliente

- ¿Qué metodologías de desarrollo de software utiliza la empresa?

Utilizamos la metodología ágil Scrum para gestionar nuestros proyectos de desarrollo de software.

- ¿Cuáles son los principales desafíos o problemas que enfrenta el equipo de desarrollo en su día a día?

Los principales desafíos incluyen la coordinación entre los equipos de desarrollo y diseño, la gestión eficiente de pruebas y despliegues, y la optimización de los tiempos de entrega de los proyectos.

- ¿Cómo se gestionan las pruebas y el despliegue de software en la empresa?

Las pruebas y el despliegue de software se gestionan de forma manual, lo que puede generar retrasos y errores en el proceso.

- ¿Qué herramientas y tecnologías utiliza el equipo de desarrollo en su trabajo diario?

El equipo de desarrollo utiliza herramientas como Java, NodeJS y Python para el desarrollo de aplicaciones. Además, utilizamos GitHub como sistema de control de versiones y Figma para el diseño gráfico.

Conocimiento sobre Arquitectura DevOps y CI/CD

- ¿Está familiarizado con el concepto de Arquitectura DevOps?

  Sí, estoy familiarizada con el concepto de Arquitectura DevOps.

- ¿Qué entiende por Integración Continua (CI) y Despliegue Continuo (CD)?

Entiendo que la Integración Continua (CI) es un proceso en el que los cambios en el código se integran automáticamente en un repositorio compartido varias veces al día, y el Despliegue Continuo (CD) es la práctica de liberar nuevas versiones de software de forma automática y frecuente a los entornos de prueba y producción.

- ¿Ha trabajado anteriormente con herramientas de CI/CD? En caso afirmativo, ¿cuáles?

  No, la empresa no ha trabajado anteriormente con herramientas de CI/CD.

- ¿Cuál es su opinión sobre la implementación de CI/CD en el proceso de desarrollo de software?

Creo que la implementación de CI/CD podría mejorar significativamente nuestros procesos de desarrollo de software, aumentando la eficiencia y la calidad de nuestros productos.

Experiencia con GitHub Actions

- ¿Está familiarizado con la herramienta GitHub Actions?

  Sí, estoy familiarizado con la herramienta GitHub Actions.

- ¿Ha utilizado GitHub Actions anteriormente en algún proyecto? En caso afirmativo, ¿para qué tipo de tareas o procesos?

  No, la empresa no ha utilizado GitHub Actions anteriormente.

- ¿Cuál es su opinión sobre la implementación de GitHub Actions en el proceso de desarrollo de software de la empresa?

Creo que GitHub Actions podría ser una excelente herramienta para mejorar nuestros procesos de desarrollo de software, automatizando tareas repetitivas y facilitando la integración y el despliegue continuo.

Mejoras y Recomendaciones

- ¿Qué mejoras o cambios sugiere para optimizar los procesos de desarrollo de software en la empresa?

Sugiero la implementación de herramientas de CI/CD como GitHub Actions para mejorar la eficiencia y la calidad de nuestros procesos de desarrollo de software.

- ¿Tiene alguna recomendación específica para la implementación de GitHub Actions en la empresa?

Recomiendo realizar una evaluación detallada de GitHub Actions y elaborar un plan de implementación que incluya la capacitación del equipo y la definición de flujos de trabajo adecuados.

Gracias por su colaboración. Sus respuestas son de gran ayuda para el estudio.

Formulario de Entrevista - Líder de Equipo de Desarrollo de AuroraSoft

Datos del Entrevistado:

- Nombre del Entrevistado: Christian Mamani
- Cargo: Líder de Equipo de Desarrollo

Procesos de Desarrollo de Software

- Describa brevemente el proceso de desarrollo de software que sigue la empresa.

AuroraSoft sigue una metodología ágil de desarrollo de software, basada en el marco Scrum. El proceso incluye las siguientes etapas:

1. Reunión de planificación
2. Desarrollo
3. Pruebas
4. Despliegue
5. Retroalimentación del cliente

- ¿Qué metodologías de desarrollo de software utiliza la empresa?

Utilizamos la metodología ágil Scrum para gestionar nuestros proyectos de desarrollo de software.

- ¿Cuáles son los principales desafíos o problemas que enfrenta el equipo de desarrollo en su día a día?

Diría que la coordinación entre los equipos de desarrollo y la curva de aprendizaje en nuevas tecnologías.

- ¿Cómo se gestionan las pruebas y el despliegue de software en la empresa?

De forma manual.

- ¿Qué herramientas y tecnologías utiliza el equipo de desarrollo en su trabajo diario?

SQL, Java, NodeJS, JavaScript, HTML, CSS y Python para el desarrollo de aplicaciones. Además, utilizamos GitHub como sistema de control de versiones y Figma para el diseño gráfico.

Conocimiento sobre Arquitectura DevOps y CI/CD

- ¿Está familiarizado con el concepto de Arquitectura DevOps?

  Sí, tengo más de 5 años de experiencia en su uso.

- ¿Qué entiende por Integración Continua (CI) y Despliegue Continuo (CD)?

El CI permite realizar integraciones varias veces al día y el CD es la práctica de liberar nuevas versiones de software de manera automática y frecuente en los entornos de prueba y producción.

- ¿Ha trabajado anteriormente con herramientas de CI/CD? En caso afirmativo, ¿cuáles?

  Sí, he trabajado anteriormente con Jenkins y GitLab CI/CD.

- ¿Cuál es su opinión sobre la implementación de CI/CD en el proceso de desarrollo de software?

Podría mejorar significativamente nuestros procesos de desarrollo de software, aumentando la eficiencia y la calidad de nuestros productos.

Experiencia con GitHub Actions

- ¿Está familiarizado con la herramienta GitHub Actions?

  Sí, estoy familiarizado con la herramienta GitHub Actions.

- ¿Ha utilizado GitHub Actions anteriormente en algún proyecto? En caso afirmativo, ¿para qué tipo de tareas o procesos?

  Lamentablemente la empresa no lo usa.

- ¿Cuál es su opinión sobre la implementación de GitHub Actions en el proceso de desarrollo de software de la empresa?

Podría ser una excelente herramienta para mejorar nuestros procesos de desarrollo de software ayudando en la integración y el despliegue continuo.

Mejoras y Recomendaciones

- ¿Qué mejoras o cambios sugiere para optimizar los procesos de desarrollo de software en la empresa?

Sugiero la implementación de herramientas de CI/CD como GitHub Actions para mejorar la eficiencia y la calidad de nuestros procesos de desarrollo de software.

- ¿Tiene alguna recomendación específica para la implementación de GitHub Actions en la empresa?

Recomiendo realizar una evaluación detallada de GitHub Actions y elaborar un plan de implementación que incluya la capacitación del equipo y la definición de flujos de trabajo adecuados.

Gracias por su colaboración. Sus respuestas son de gran ayuda para nuestro estudio.

Formulario de Entrevista – Personal de Desarrollo

- ¿Qué es Git y cuál es su importancia en el desarrollo de software?

Desarrollador 1 - Es importante en el desarrollo de software porque facilita el trabajo colaborativo, controla los cambios en el código, y permite mantener un historial completo de las modificaciones realizadas en el proyecto.

Desarrollador 2 - Su importancia en el desarrollo de software radica en su capacidad para gestionar cambios, versiones y ramas de manera eficiente, permitiendo un flujo de trabajo organizado y controlado.

Desarrollador 3 - Es fundamental en el desarrollo de software porque facilita el seguimiento de los cambios realizados en el código, la colaboración entre desarrolladores y la gestión de versiones del proyecto.

Desarrollador 4 - Git es un sistema de control de versiones distribuido que permite gestionar el código fuente de un proyecto de software.

- ¿Qué es GitHub Actions y cómo se utiliza en el proceso de integración y despliegue continuo (CI/CD)?

Desarrollador 1 - GitHub Actions es un servicio de integración continua y entrega continua (CI/CD) que permite automatizar el flujo de trabajo de desarrollo de software.

Desarrollador 2 - Se utiliza en el proceso de CI/CD para definir y ejecutar flujos de trabajo que automatizan tareas como la compilación, las pruebas y el despliegue de aplicaciones. Estos flujos de trabajo se definen en archivos de configuración YAML y se activan automáticamente en respuesta a eventos específicos en el repositorio de GitHub, como la creación de una nueva solicitud de extracción o la confirmación de cambios en una rama.

Desarrollador 3 - Estos flujos de trabajo automáticos que se ejecutan contra las respuestas a eventos específicos en el repositorio de GitHub.

Desarrollador 4 – Es un servicio nativo de GitHub que sirve para automatizar tareas de despliegue de software.

- ¿Qué es Scrum y cuáles son sus principales roles y eventos?

Desarrollador 1 - Scrum es un marco de trabajo ágil para el desarrollo de software que se centra en la entrega continua de valor al cliente.

Desarrollador 2 – Es un marco de trabajo y sus eventos principales son la Reunión de Planificación, la Reunión Diaria, la Revisión de Sprint y la Retrospectiva de Sprint.

Desarrollador 3 - Scrum es un juego de equipo para crear programas de computadora. Hay tres roles principales: el Scrum Master, que es como el líder del equipo y nos ayuda a resolver problemas. El PO, nos dice qué es lo más importante para hacer. Y finalmente, está el Equipo de Desarrollo, que somos nosotros, los que escribimos el código.

Desarrollador 4 – (Igual que el Desarrollador 2) Es un marco de trabajo y sus eventos principales son la Reunión de Planificación, la Reunión Diaria, la Revisión de Sprint y la Retrospectiva de Sprint.

- ¿Qué es un stand-up meeting y cuál es su propósito en Scrum?

Desarrollador 1 - Un stand-up meeting es como una reunión rápida que tenemos todos los días. Nos paramos en un círculo y hablamos sobre lo que hicimos ayer, lo que vamos a hacer hoy y si hay algo que nos está deteniendo.

Desarrollador 2 – Es una reunión diaria que sirve para estar seguros de que todos estamos trabajando juntos y no tenemos problemas que nos impidan avanzar.

Desarrollador 3 - Es una reunión diaria que dura aproximadamente 15 minutos, donde todo el equipo se reúne de pie frente al tablero Scrum.

Desarrollador 4 - Durante esta reunión, cada miembro del equipo comparte brevemente qué hizo el día anterior, qué planea hacer hoy y si hay algún obstáculo que le impida avanzar en su trabajo. El propósito del stand-up meeting es mantener al equipo sincronizado, identificar cualquier problema o impedimento y asegurarse de que todos estén trabajando hacia el mismo objetivo.