Universidad de Pamplona

Facultad de Ingenierías y Arquitectura

Programa de Ingeniería de Sistemas

Tema:

CREACIÓN DE UN PROCEDIMIENTO DE DESPLIEGUE AUTOMÁTICO DE APLICACIONES SOBRE LA INFRAESTRUCTURA TECNOLÓGICA DEL ÁREA DE DESARROLLO DE LA EMPRESA BEGO.

Autor:

Luis Fernando Arcila Acosta

Pamplona, Norte De Santander

Junio 2021

Universidad De Pamplona

Facultad de Ingenierías y Arquitectura

Programa de Ingeniería de Sistemas

Trabajo de grado presentado para optar al título de Ingeniero de Sistemas.

Tema:

CREACIÓN DE UN PROCEDIMIENTO DE DESPLIEGUE AUTOMÁTICO DE APLICACIONES SOBRE LA INFRAESTRUCTURA TECNOLÓGICA DEL ÁREA DE DESARROLLO DE LA EMPRESA BEGO.

Autor:

Luis Fernando Arcila Acosta

Director:

Luz Marina Santos Jaimes

PhD en Ciencias de la computación y matemáticas computacionales

Pamplona, Norte de Santander.

Junio 2021.

DEDICATORIA

Dedicado a mi familia, quienes con su apoyo han hecho de mi sueño de ser ingeniero, una realidad.

Resumen

El despliegue de aplicaciones es una de las tareas más comunes durante el ciclo de vida de estas, esto significa llevar dicha aplicación y los cambios realizados en ella a lo largo del tiempo, desde el entorno de desarrollo hasta el entorno de producción, pasando habitualmente por otros entornos como: integración, pruebas, preparación o pre-producción. El despliegue de aplicaciones generalmente es un proceso muy repetitivo el cual consiste en recuperar el código del sistema de control de versiones, copiarlo en las máquinas de destino y realizar ciertas tareas como: la instalación o actualización de dependencias mediante algunas herramientas o la ejecución de migraciones de la base de datos. En un proyecto pequeño por lo general no se requiere de despliegues repetitivos, por ende, estas tareas pueden ser realizadas de forma manual o semi-manual, por ejemplo, con el uso de algunos scripts que automaticen parte de las mismas, pero no descartando la intervención humana.

En este documento se busca formular un procedimiento bien estructurado que permita a la empresa BeGo automatizar la tarea del despliegue de aplicaciones, la cual se viene haciendo de forma manual. En esta empresa los despliegues suelen ser habituales y complejos, por esta razón es necesario automatizar todo el proceso, de manera que se reduzca el tiempo para hacer llegar las nuevas funcionalidades o correcciones a los usuarios finales y así poder minimizar los errores humanos.

Palabras claves: despliegue, integración, DevOps, pipeline, pruebas

Abstract

The deployment of applications is one of the most common tasks during their life cycle, this means taking the application and the changes made to it over time, from the development environment to the production environment, usually passing through other environments such as: integration, testing, preparation or pre-production. The deployment of applications is generally a very repetitive process which consists of retrieving the code from the version control system, copying it to the target machines and performing certain tasks such as: installing or updating dependencies using some tools or executing migrations from the database. In a small project, repetitive deployments are not generally required, therefore, these tasks can be carried out manually or semi-manually, for example, with the use of some scripts that automate part of them, but not ruling out the human intervention.

This document seeks to formulate a well-structured procedure that allows the BeGo company to automate the application deployment task, which has been done manually. In this company, deployments are usually common and complex, for this reason it is necessary to automate the entire process, so that the time to get the new functionalities or corrections to the end users is reduced and thus be able to minimize human errors.

Keywords: Deploy, integration, DevOps, pipeline, tests

Tabla de contenido

CAPITULO 6

Tablas

Tabla de figuras

CAPÍTULO 1

1 Descripción del proyecto

1.1 Planteamiento del problema

BeGo es una empresa que ofrece servicios de desarrollo de software, al día de hoy se han creado una serie de procedimientos que estandarizan la forma en que se deben llevar a cabo las tareas de integración continua y de pruebas dentro de la empresa, estas han establecido un orden organizacional internamente y han permitido llevar un mejor control. Una de las buenas prácticas del desarrollo de software lo constituyen la Integración Continua, la cual permite la reducción de riesgos y tareas repetitivas; otra práctica lo constituye la elaboración de pruebas, que permite controlar la ejecución de un producto software de manera automática, comparando los resultados obtenidos con los resultados esperados, y la generación de software listo para desplegar e incrementar la calidad, lo que permite el monitoreo de sistemas de control de versiones.

La problemática que tiene en estos momentos BeGo, es la falta de procedimientos automatizados para realizar las actividades de despliegue de aplicaciones, lo cual consume tiempo y en la mayoría de casos crea cuellos de botella por posibles errores humanos. La falta de un procedimiento de despliegue automatizado genera la siguiente pregunta **¿Será posible eliminar errores humanos, disminuir el consumo de tiempo y aumentar la productividad en el despliegue de aplicaciones web sobre la infraestructura tecnológica del área de desarrollo de la empresa BeGo?**

1.2 Justificación

Las empresas de desarrollo de software buscan la manera de entregar sus productos o mejoras de los mismos de una forma ágil y fiable a sus clientes, en la mayoría de las empresas en las que aún se ejecutan despliegues manuales los tiempos que se toman para realizar estas tareas son muy desfasados con respecto a lo planificado. Además, el despliegue de aplicaciones en máquinas virtuales tradicionales, consume demasiados recursos solo para que levanten ciertos servicios que el aplicativo necesita para su ejecución de manera satisfactoria, a esto se le suma los errores humanos que pueden haber de por medio. Es por ello que se necesita de manera inmediata el diseño de un procedimiento de despliegue automático de aplicaciones que incluya los mecanismos y herramientas necesarias para la realización del despliegue sin la intervención humana, que le permitirá a la empresa BeGo llevar un mejor control.

El poder eliminar cualquier tarea manual de aprovisionamiento de infraestructura o de despliegue de aplicaciones, evita que más individuos accedan a los entornos del sistema y de esta manera se reduce la posibilidad de equivocación en una intervención. De esta forma las empresas se ven netamente beneficiadas ya que la automatización del despliegue proporciona estándares de control, planeación y protocolos de funcionamiento, libera al equipo de desarrollo de realizar tareas repetitivas que consumen bastante tiempo, reducción en los tiempos de despliegue de aplicativos Web en el departamento de producción y la eliminación de la intervención humana con los errores que esta implica.

El número de clientes o empresas que demandan aplicativos Web aumenta significativamente cada año, la automatización de algunos procesos como la integración continua y el despliegue continuo mejoran la productividad gracias a la optimización progresiva del trabajo, que permite

iniciar continuamente nuevos proyectos, abarcando de esta manera una mayor cantidad de clientes en el mercado.

Con el modelo de integración y despliegue continuo se acorta significativamente el tiempo que deben esperar los usuarios para adquirir las nuevas versiones de los aplicativos Web, ya que se reduce el periodo desde que se definen los requisitos de un procedimiento, y de sus posteriores versiones, hasta el despliegue del producto.

De esta manera el usuario final dispondrá de una versión funcional en un menor tiempo, con la que podrá ir evaluando el resultado.

1.3 Delimitaciones

1.3.1 Objetivo general

- Construir un procedimiento de despliegue automático de aplicaciones sobre la infraestructura tecnológica del área de desarrollo de la empresa BeGo.

1.3.2 Objetivos específicos

- Determinar el estado del arte del proceso de despliegue en las empresas dedicadas al desarrollo de Software.
- Analizar las herramientas y técnicas para implementar el proceso de despliegue en la empresa BeGo.
- Validar el procedimiento de despliegue automático con una nueva versión del aplicativo web, el cual esta soportado en programación híbrida mediante el uso de la librería React Native.

1.4 Metodología

Para la realización de este proyecto el método conveniente a utilizar se basó en un enfoque de investigación aplicada, ya que esta ayudó a la generación de conocimiento con aplicación directa a los problemas del sector productivo en la infraestructura de desarrollo de las instalaciones de BeGo, se exploró en el conocimiento ya existente de este tema y se le dio uso con el fin de realizar un análisis muy puntual, de los recursos con los que se contaba para obtener los resultados esperados durante el desarrollo de los objetivos planteados en el proyecto.

Este proyecto se desarrolló teniendo en cuenta las siguientes etapas:

La primera se centró en investigar para dominar teóricamente el tema, de este modo se profundizo a nivel regional, nacional e internacional en lo referente a antecedentes de la estandarización de despliegues para aplicativos web, la puesta en marcha de despliegues automáticos en general y toda la parte técnica correspondiente al despliegue continuo. Seguido a esto se realizó un análisis comparativo entre las diferentes técnicas y herramientas de despliegue de aplicaciones web para determinar cuáles eran las herramientas y técnicas que mejor se adecuaban a la empresa, cabe aclarar que en esta etapa fue donde se tomó la decisión de la herramienta que se utilizaría para llevar a cabo esta tarea y así se compartió la decisión con el equipo de desarrollo de BeGo, realizando así una retroalimentación en términos de conveniencia para el departamento de desarrollo y para dar continuidad al método que se venía aplicando en las pruebas e integración continua. Finalmente, la última etapa consistió en la recolección de información organizada para generar un procedimiento de despliegue automático de aplicaciones bien estructurado, que permitiera así lanzar o actualizar versiones de los proyectos que se estén trabajando en la empresa.

Esta etapa se desarrolló a través de la implementación y validación del despliegue automatizado, realizado en una serie de pasos, primeramente, se configuro cada uno de los parámetros necesarios para la conexión entre el servidor y GitHub, después de esto se realizó la configuración del archivo YML para GitHub actions para realizar las pruebas antes de entrar a tocar los repositorios de BeGo se creó un índex de prueba en el repositorio de GitHub el cual ya tenía toda la configuración, después de esto se agregaron unos cambios en el índex para verificar que se disparara el action configurado previamente, una vez la acción se tornó color verde nos dirigimos al navegador a probar con el dominio que se había configurado en el servidor esto con el objetivo de evidenciar los cambios en tiempo real, finalizando esta etapa se construyó una tabla que mostraba el aporte que se le dio a la empresa comparando el antes y el después de la automatización del proceso de despliegue continuo en las instalaciones de la empresa.

Se aclara que para la ejecución de este proyecto y la creación del procedimiento se recopilo información secundaria como: tesis de grados, artículos, libros e internet. A continuación, se describe el proceso empleado para la construcción del procedimiento:

Primeramente, se llevaron a cabo algunos cambios en la máquina local, esto con el objetivo de facilitar la conexión entre el servidor contratado por la empresa. Del mismo modo se realizaron una serie de configuraciones en los repositorios de GitHub de la empresa, dentro de estos se creó una nueva rama de trabajo para la realización de las debidas pruebas realizando algunos cambios en la nueva rama de github., a la vez se configuraron algunos protocolos de conexión mediante el uso del protocolo FTP en el servidor, Una vez que se terminó con la etapa pruebas y todo pareció perfecto, se buscó integrar estos nuevos cambios con la rama maestra, esta es la rama de producción, técnicamente se sabe que todo lo que se envíe a la rama maestra estará listo para funcionar.

Finalmente, se llegó a desplegar todo el proyecto probado e integrado previamente, se desplego la rama maestra de forma remota y se implementó automáticamente en el servidor que utiliza la empresa, en este caso cPanel, con una característica especial y esta permitirá que los usuarios aprecien los cambios en vivo.

Finalmente, en el anexo a este documento se puede apreciar el cumplimiento de objetivos por parte del pasante en la empresa BeGo.

CAPÍTULO 2

2 Marco teórico y estado del arte

2.1 Marco conceptual

2.1.1 Integración continua, entrega continua y despliegue continuo

El proceso de desarrollo de software se conforma por una serie de etapas **(ilustración 1)** en las que participan diferentes departamentos que actúan en entornos totalmente distintos. Todo esto con el objetivo de optimizar el ciclo de vida del software y así conseguir un producto de calidad, reduciendo de esta manera los posibles errores, en este momento las metodologías como DevOps (Development-Operations) apuestan por la automatización en las etapas de desarrollo para garantizar una reducción de tiempo y costo en cada una de las empresas dedicadas al desarrollo de software, para así mismo construir una mejor comunicación entre los desarrolladores y profesionales operadores de las tecnología de la información. (DEVOPS, 2020)

Ilustración 1. Procesos previos a la automatización del despliegue. *(DEVOPS, 2020)*

2.1.1.1 Integración Continua.

La integración continúa CI/CD (continuous integration), creada por Martin Fowler, surge con el objetivo de facilitar este trabajo en equipos de desarrollo y automatización de tareas de

integración. La integración continua se basa en la construcción automática de proyectos con alta frecuencia, promoviendo la detección de errores en un momento temprano para dar prioridad a corregir dichos errores. (Vargas & Marcos, 2019)

Ilustración 2. Beneficios de la Integración Continua "CI"
(Chávez, 2018)

La integración continua a demás es una práctica en la que los desarrolladores combinan el código en un repositorio común el cual se puede trabajar en Git o cualquier otra herramienta que facilite el control de versiones y trabajo simultáneo, facilitando de esta manera la realización de pruebas para detectar y resolver los posibles errores que se puedan encontrar. Cabe resaltar que con la CI se impide que se desarrollen distintas divisiones de una aplicación que luego puedan tener conflictos entre sí, lo que significa un control estricto que garantiza el orden. (DEVOPS, 2020)

2.1.1.2 Entrega Continua

La entrega continua CI/CD (continuous delivery) tiene estrecha relación con lo que es la integración continua, esta consiste en crear una estrategia para la automatización del proceso de entrega del software, permitiendo de esta manera que pueda ser implementado en producción de forma confiable y sencilla en general se puede entender que la CD es la entrega de actualizaciones de software a los usuarios de forma sólida y continua. (DEVOPS, 2020)

2.1.1.3 Despliegue continuo

Este proceso suele confundirse un poco con la entrega continua y esto es debido a que las siglas de ambos procesos es la misma, (en ambas se usa CD).

El despliegue continuo o CD (continuous deployment) tiene una gran relación con la entrega continua. La diferencia es que con el despliegue continuo se da un paso más allá, este se da automatizando todo lo que es el proceso de entrega de software al usuario final y eliminando lo que es la acción manual o intervención humana necesaria en la entrega continua. (DEVOPS, 2020)

El proceso de despliegue en particular viene a ser una serie de pasos que deben ejecutarse en orden y de forma correcta. Esto quiere decir que si algunos de los pasos no se realizan de forma correcta lo más seguro es que el despliegue no se lleve a cabo. Por tal razón es fundamental que algunos diseños previos como el de la automatización de pruebas se haya realizado de forma correcta, ya que, al no producirse ninguna entrada o acción manual, este dependerá en gran medida de cómo sean esos diseños.

En la **ilustración 3** se evidencia la infraestructura necesaria para la realización de estos despliegues continuos, esta se compone de hardware y herramientas o frameworks que agilizarán el ciclo de vida de desarrollo de software. (Everac99, 2011)

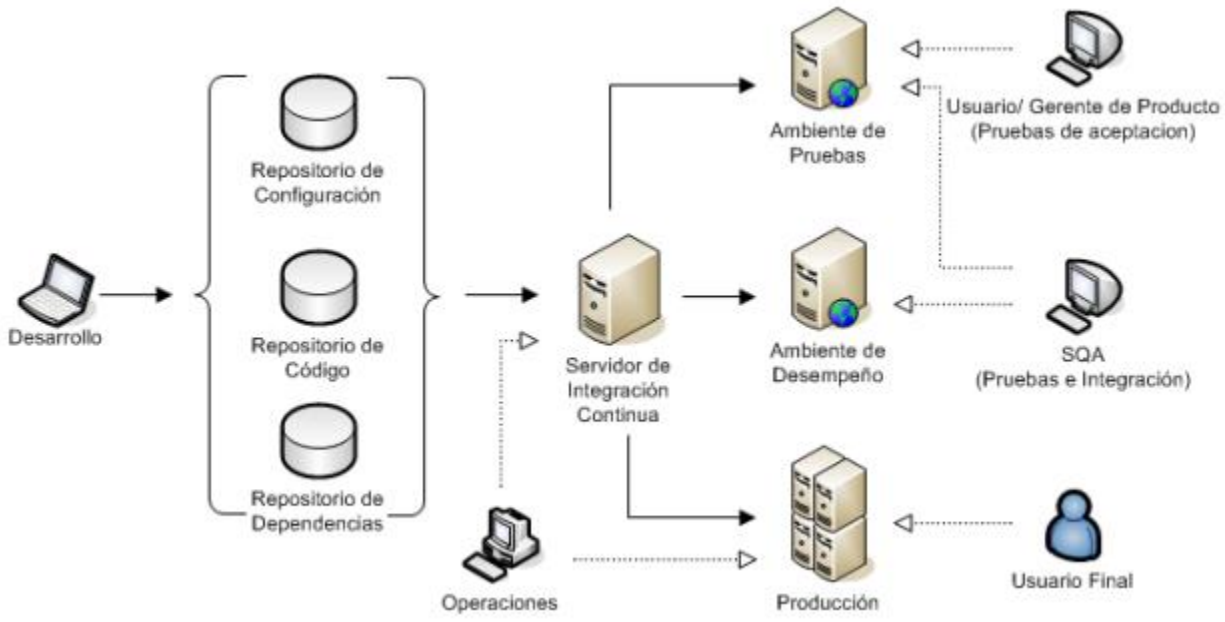


Ilustración 3. Infraestructura necesaria para realizar despliegues continuos (Everac99, 2011)

2.1.2 Ventajas al automatizar las etapas de Integración continua, entrega continua y despliegue continuo

Tener estas etapas automatizadas conllevan a las siguientes ventajas:

- Acelerar la entrega de cambios a producción.
- Mejorar la calidad del producto en ambientes productivos.
- Mejora los tiempos de recuperación de un ambiente productivo que presente fallas o donde se encuentren errores.
- Dejar registros y logs constantes que permitan el monitoreo y mediciones de cada una de las etapas para una mejora continua.

- Disminuir el desperdicio de talento, donde el equipo de desarrollo y personal de operaciones no estarán realizando tareas repetitivas y podrán estar asignados a tareas que generen más valor a la compañía. (Oquendo, 2021)

2.1.3 Pruebas de software

Las pruebas de software no son más que un conjunto de procesos con los que se pretende probar un sistema en diferentes momentos, esto con el objetivo de comprobar su correcto funcionamiento. Este tipo de pruebas abarca cualquier estado del desarrollo del sistema, desde su creación hasta su puesta en producción. Lo interesante de las pruebas es que se puedan iniciar de manera automática, para identificar en cualquier momento si se tiene una aplicación estable o si algún cambio ha afectado a otras partes sin darse cuenta. (Turrado, 2020)

2.1.4 Interés por el despliegue de aplicaciones

La automatización es especialmente interesante en aquellos proyectos donde se despliegan aplicaciones (por ejemplo, un software SaaS, como un CRM online, con gran cantidad de puestos de trabajo que hacen uso del mismo). Este proceso de despliegue puede ser un factor crítico para lograr los objetivos de un negocio, por lo que el despliegue automático y efectivo de los diversos componentes de las aplicaciones actuales es fundamental para evitar errores y aumentar los tiempos de actividad. (ILIMIT, 2020)

El despliegue de aplicaciones web o de apps para móviles son dos de los ejemplos en donde el despliegue de aplicaciones automatizado es muy interesante.

2.1.5 Los patrones del despliegue

Existen patrones para el desarrollo de software que son bien conocidos, pero también existen patrones para el despliegue de aplicaciones. A continuación, se analizan ocho patrones de despliegue.(Seta, 2009)

- Gestión de todos los archivos de configuración en un Repositorio centralizado, lo que permite utilizar un Despliegues por Scripts para crear software que funcione.
- Despliegues por Scripts, lo que automatiza todas las acciones de despliegue de manera que no haya intervención humana cuando se ejecuta un despliegue.
- Comando único, lo que reduce la complejidad del despliegue y permite realizar una ejecución sin humanos durante el proceso de despliegue.
- Configuración con tokens, lo que brinda una forma repetible de inyectar información variable dentro de los archivos de configuración.
- Configuración externalizada, que simplifica el ingreso de información variable para los ambientes destino.
- Plantilla de verificación, que ayuda a asegurar que todas las propiedades de los ambientes destino sean iguales.
- Ejecución sin humanos, que brinda una forma segura de acceder a múltiples equipos en un proceso automatizado.
- Despliegue unificado, que promueve un único script de despliegue que puede ejecutarse en distintos ambientes de destino.

2.1.6 Tecnologías open source básicas de integración y despliegue continuo

Ante un escenario donde el objetivo de las empresas es dar valor a los usuarios y clientes de forma frecuente, automatizada y libre de errores, llegando incluso a varios despliegues en entornos de producción en un mismo día, se busca aprovechar las tecnologías open source con el fin de permitir la colaboración activa y mejora continua entre desarrolladores de cada software.

Es por eso que las metodologías ágiles, la integración continua y DevOps, forman parte del mismo ambiente y su relación se hace cada vez más estrecha. (García, 2019)

A continuación, se dan a conocer algunas de estas tecnologías open source para realizar integración y despliegue continuo.

2.1.6.1 **Maven o Gradle**. Estas herramientas permiten realizar tareas como crear estructuras de directorios, hacer limpieza, compilar, generar documentación, lanzar pruebas y realizar despliegues, en la actualidad no basta con solo hacer entregas, si no que estas tienen que ser funcionales y libres de errores. Es por esta razón que se incluye esta herramienta en la lista de herramientas que aseguran la calidad del software desarrollado.

Ilustración 4. Maven y Gandle. *(García, 2019)*

2.1.6.2 **SonarQube.** Es la herramienta de gestión de la calidad del código fuente más extendida para desarrollos Java, la cual permite realizar un análisis estático del código, aunque también da soporte a otros lenguajes. Permite recopilar, analizar y visualizar métricas del código fuente.

Ilustración 5. SonarQube *(García, 2019)*

2.1.6.3 **Jenkins.** Es un servidor de integración continua, muy fácil de utilizar, y que también se encuentra entre los más aceptados. La base de Jenkins son las tareas y se utiliza habitualmente como centro del proceso de desarrollo, Jenkins es un servidor de automatización open source escrito en Java, el cual pone en funcionamiento y perfecta sincronización a todas las herramientas, asegurando que el código generado cumpla con los estándares de calidad y que, mediante las pruebas automatizadas, el código existente sigua funcionando correctamente.

Ilustración 6. Jenkins, Herramienta popular para CI/CD *(García, 2019)*

2.1.6.4 **Git**. Se ha convertido en la herramienta preferida por los desarrolladores modernos. Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Se puede llamar a Git un sistema de control de versiones que puede rastrear cada cambio en un archivo. Su principal objetivo es gestionar todos estos cambios realizados en un proyecto durante un período de tiempo. Git almacena estos cambios y la información relacionada en una estructura de datos o repositorio.

Ilustración 7. Git herramienta *(git-scm.com)*

2.1.7 DevOps

DevOps es una combinación de las palabras "development" (desarrollo) y "operations" (operaciones), pero esta representa un conjunto de ideas y prácticas que van más allá de ambos conceptos, ya sea que estén juntos o separados. DevOps incluye sistemas de seguridad, maneras de trabajar en colaboración, análisis de datos, entre otras características. (DEVOPS, 2020)

Esta combinación describe los enfoques para agilizar los procesos con los que una idea puede pasar del desarrollo a la implementación, en un entorno de producción en que puede generar valor para el usuario. (RedHat)

Un aspecto muy importante aquí es que DevOps es prácticamente un modo de abordar la automatización y el diseño de las plataformas, esto mediante la prestación ágil de servicios que brinden una alta calidad. Los desarrolladores, que generalmente realizan codificaciones en un entorno de desarrollo estándar, trabajan en estrecha colaboración con los equipos de operaciones de TI para agilizar el diseño, las pruebas y el lanzamiento de los sistemas de software, sin comprometer la confiabilidad. (RedHat)

2.1.8 Pipeline de despliegue

Los pipeline de despliegue están compuestos por tres etapas principales y cada una de ellas tiene una serie de tareas repetitivas. Estas pueden ser automatizadas mediante scripts escritos en formatos YAML, un formato que permite describir etapas y tareas en forma de árbol que se ejecutan de forma secuencial. Actualmente existen diversas herramientas visuales de CI/CD (integración continua y entrega continua) en el mercado y se encuentran de todos los tipos. Las etapas del Pipeline de despliegue (ilustración 8) se han venido explicando previamente, por ende, se nombran a continuación y se dará a conocer las tareas que se llevan a cabo en cada una de estas. (Oquendo, 2021)

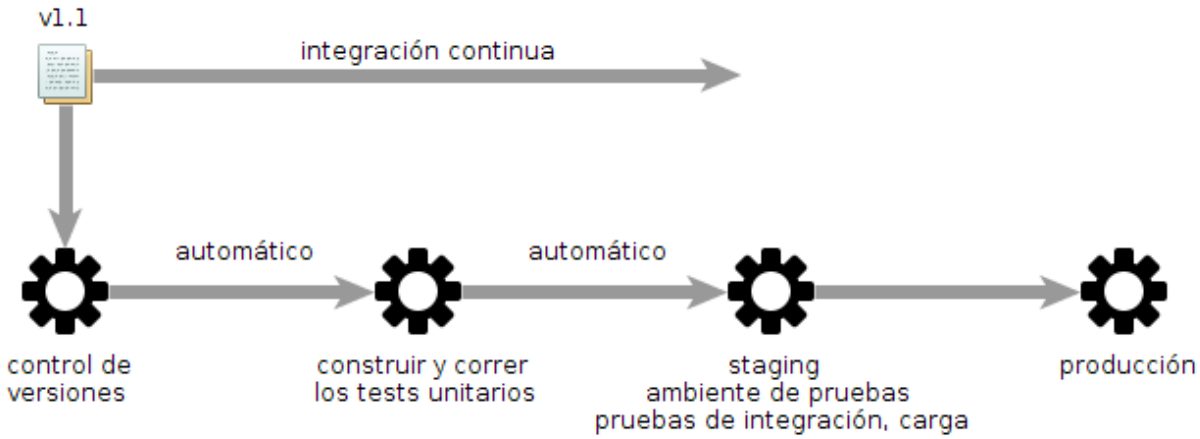


Ilustración 8. Etapas del Pipeline de despliegue *(Oquendo, 2021)*

Etapa 1 - Integración:

- Compilación del código.
- Ejecución de pruebas unitarias.
- Análisis de la calidad de código.

Si algo llegara a fallar en esta etapa, el pipeline se debe detener y se deben corregir de inmediato los errores en el código o en los scripts de integración para volver a ser ejecutado y así poder continuar con la siguiente etapa. (Oquendo, 2021)

Etapa 2 - Entrega a un ambiente de pruebas o pre-productivo

Una vez la primera etapa es exitosa, entonces se ejecutará la siguiente etapa la cual es la entrega a un ambiente de pruebas o pre-productivo, en la cual se realizan las siguientes tareas:

- Despliegue de los archivos generados por la compilación de la primera etapa.
- Automatización de pruebas de integración y aceptación.
- Pruebas no funcionales (usabilidad, seguridad, rendimiento).
- Pruebas manuales de usuario.

Cabe aclarar que esta etapa se puede repetir en cada uno de los entornos que el negocio crea necesario, tratando de que cada uno de los ambientes sea muy parecido al ambiente productivo y así garantizar la mejor calidad en el producto. (Oquendo, 2021)

Etapa 3 – Despliegue a Producción

En última instancia, si la etapa de integración y la etapa de entrega fueron satisfactorias, se procederá al despliegue en producción, en donde se realizan las siguientes tareas:

- Despliegue de los archivos generados por la compilación de la primera etapa.
- Validación de nuevas funcionalidades.
- Validación de funcionalidades anteriores.

2.1.9 Ventajas de automatizar sistemas

La automatización en cada uno de los entornos siempre genera ventajas y beneficios, en este caso en el desarrollo de software se proporciona una serie de beneficios entre los que podemos destacar los siguientes:

1. Reduce el tiempo necesario para completar una tarea. Con la automatización se consigue liberar al equipo de desarrollo de realizar tareas repetitivas que consumen mucho tiempo. La automatización recorta el tiempo de comercialización de una aplicación.

2. Una vez automatizado un proceso, se puede reutilizar el número de veces que sea necesario.

3. La automatización requiere comprender bien los procesos por lo que se consiguen procesos mejorados y mejor definidos que optimizan la ejecución y con ello el valor del servicio.

4. Ofrece mayor consistencia en el desarrollo, ya que analiza en profundidad los riesgos, reduciendo la incertidumbre y el margen de error por fallos en la ejecución. La reducción de este número de errores e incidencias hace que el desarrollo sea más confiable.

5. Ofrece una visibilidad y control de todos los flujos de trabajo.

6. Permite ser proactivos, anticipando y evitando interrupciones (por ejemplo, liberar espacio en un disco antes de que se agote).

7. Todas estas ventajas permiten el despliegue más rápido y con un servicio de mayor calidad lo que aumenta la satisfacción del cliente.

8. La automatización de procesos reduce los costes operativos. (ILIMIT, 2020)

2.1.10 Open Source:

"Diseñado de manera que sea accesible al público: todos pueden ver, modificar y distribuir el código de la forma que consideren conveniente. El software open source se desarrolla de manera descentralizada y colaborativa, así que depende de la revisión entre compañeros y la producción de la comunidad. Además, suele ser más económico, flexible y duradero que sus alternativas propietarias, ya que las encargadas de su desarrollo son las comunidades y no un solo autor o una sola empresa." (RedHat). Existe una diferencia entre código libre y software libre, suelen ser lo mismo en la mayor parte de los casos con excepciones. En algunos casos se basan principalmente en el acuerdo de licencias de uso y distribución.

2.2 Estado del arte

2.2.1 Internacional

Titulo	Autor	Resumen
Despliegue ágil en proyectos de gran escala.	(Orias & Marfil, 2018)	En este trabajo de grado realizado por un estudiante de la Universidad de la plata en Argentina se expone un análisis acerca de cómo es tratado por la Ingeniería de software el proceso de despliegue, este a través de diferentes modelos. Se describe la metodología ágil SCRUM y como se utilizan algunas ideas de la mismas para mejorar la organización y planificación en la etapa de despliegue.

		Implementación y descripción de la herramienta P.E.P. utilizada en un proyecto de gran escala para mejorar la gestión y ejecución del proceso de despliegue. Se afirma allí que el despliegue de una aplicación es una parte muy importante en el desarrollo de software, mediante el mismo, se pone en marcha lo construido en un determinado tiempo, ya sea un sistema nuevo de cero o una nueva versión del mismo. Dicho producto de software a desplegar, en organizaciones de gran tamaño, involucra el esfuerzo de las distintas áreas implicadas de la organización (diseño y desarrollo, análisis funcional, testing, capacitación, arquitectura, soporte, seguridad, coordinación, gestión de proyectos, etc.). En proyectos de gran tamaño, cuyo sistema a desplegar es una pieza fundamental para el funcionamiento de la organización, el proceso de despliegue cobra una relevancia notoria. Por tal motivo, es fundamental que el mismo se lleve a cabo de la manera más rápida y eficiente posible, permitiendo entregar nueva funcionalidad a los usuarios en el menor tiempo posible sin afectar el funcionamiento del sistema.
Arquitectura tecnológica para la entrega continua de software con despliegue en contenedores	(Sánchez, 2017)	La Universidad de la Cuenca en Ecuador a través de la Dirección de Tecnologías de la Información y Comunicación ofrece a la Comunidad Universitaria servicios relacionados con las TICS para dar soporte a las actividades administrativas, académicas y de investigación. La Dirección dispone de una unidad de desarrollo de software que se encuentra encargada de dar mantenimiento a los sistemas de información que se encuentran en producción y el desarrollo de nuevos proyectos enfocados a fortalecer la estrategia institucional. La carencia

		de una arquitectura estándar para el desarrollo de software ha provocado que el despliegue de nuevas versiones de las aplicaciones sea una tarea extensa y complicada, introduciendo con facilidad errores de regresión en el software que se pone en producción dificultando el aseguramiento de calidad. Por tanto, se ha planteado la necesidad de implementar un pipeline de entrega continua de software que permita potenciar al equipo de desarrollo para mitigar los riesgos inherentes a la liberación de producto. Para el éxito en la implementación de un pipeline de entrega continua se requiere un alto nivel de automatización en el proceso de construcción y verificación del software que permita despliegues automatizados de manera confiable. Las herramientas que lo soportan se encuentran en función de los stacks de tecnologías y el ambiente de ejecución de las aplicaciones. Este proyecto de tesis define a partir de los fundamentos de la entrega continua de software y el análisis del trabajo realizado hasta el momento, una arquitectura tecnológica para la implementación de un proceso automatizado de entrega continua acorde a los lineamientos y necesidades de la unidad de desarrollo de software de la Universidad de Cuenca.
Diseño e implementación de un sistema de entrega continua para aplicaciones web sobre contenedores Docker.	(Morales, 2015).	El objetivo principal de este proyecto de grado es reducir el tiempo necesario para desplegar cualquier aplicación desarrollada en BiiT Sourcing Solutions en un entorno de producción. Todo ello con total garantía de que el software generado cumplirá con los estándares mínimos de calidad. Para lograrlo, hemos implementado un sistema de entrega continua, donde una vez que el equipo de desarrollo ha finalizado una nueva versión de la aplicación, se realizan automáticamente todas las pruebas necesarias asegurando el correcto

		funcionamiento del software. Además de esto, las soluciones propuestas se basan en la virtualización utilizando contenedores Docker para asegurar la portabilidad y la optimización de los recursos de la empresa.
DevOps in practice: A multiple case study of five companies	(ELSEVIER, 2019)	DevOps presupone la automatización y la colaboración multifuncional entre el desarrollo de software y las operaciones. La adopción e implementación de DevOps en las empresas no es trivial debido a los cambios necesarios en los aspectos técnicos, aspectos organizativos y culturales. Este estudio exploratorio presenta descripciones detalladas de cómo se implementa DevOps en la práctica. El contexto de la investigación empírica es el desarrollo de aplicaciones y servicios web en pequeñas y medianas empresas o compañías.
Procedure for the security of the deployment process of web applications	(Yeja, 2016)	El auge que ha adquirido la utilización de las aplicaciones web ha propiciado el surgimiento de riesgos de seguridad que atentan contra la disponibilidad, integridad y confidencialidad de la información. Por ello, se requiere la adopción de mecanismos para el enfrentamiento a los problemas de seguridad que se presentan diariamente en las aplicaciones web. Una de las alternativas que se recomiendan incluye la adopción de la seguridad a lo largo de todo el ciclo de desarrollo de software. En este sentido, la etapa de despliegue no está exenta de dificultades y desafíos que afrontar desde el punto de vista de seguridad, por lo que una configuración cuidadosa y la personalización del entorno pueden mejorar en gran medida el estado de seguridad de la aplicación. En este trabajo se realiza un estudio sobre la utilización de buenas prácticas establecidas y estándares en torno a la seguridad

		del despliegue de aplicaciones web, con un enfoque vinculado a la automatización y prácticas tradicionales del desarrollo de software. El procedimiento se ha aplicado en el Nodo Central de la Universidad de las Ciencias Informáticas y su utilización ha permitido la reducción del tiempo de realización de las operaciones y la cantidad de errores humanos cometidos desde la perspectiva de la seguridad informática. Los resultados obtenidos permiten establecer una referencia para la posible aplicación del procedimiento en otros entornos. Las herramientas utilizadas pueden ajustarse a las necesidades y políticas de la organización para su aplicación.
Desarrollo de software continuo e integrado usando DevOps	(IEEE, 2018)	Devops tiene que ver con la cultura, la automatización, la medición y el intercambio (CAMS). Está ganando popularidad debido a su enfoque continuo: integración continua (CI), implementación continua (CD) y entrega de software. CI requiere que el desarrollador confirme el código varias veces al día, seguido de compilación y prueba automáticas y comentarios inmediatos al desarrollador cada vez que se encuentre algún error. Si no se encuentra ningún error, el código comprometido se envía a producción. Según el informe State Of Devops de 2017, las empresas de alto rendimiento como Amazon y Netflix implementan miles de veces al día. La gestión de la configuración permite la reversión del código anterior para el desarrollador. Sin embargo, varios factores impiden que las organizaciones adopten estos enfoques, como la falta de una prueba de aceptación totalmente automatizada, una metodología de retroceso deficiente, un control de calidad manual, etc. Este documento de investigación proporciona varias metodologías y herramientas que pueden constituir una canalización de CD / CI eficaz.

2.2.2 Nacionales

Titulo	Autor	Resumen
Prácticas devops de entrega continua de software para la transformación digital de los negocios	(GARCIA, 2019)	En la universidad EAFIT ubicada en la ciudad de Medellín Colombia se llevó a cabo una investigación en las prácticas DevOps; cuando se analizan los departamentos, áreas, direcciones, vicepresidencias de tecnología y sistemas informáticos, con el propósito de determinar su razón de ser, la mayoría de los elementos llevan a concluir que su misión es la de crear y entregar soluciones capaces de proporcionar valor al negocio, es decir, servicios de TI con diferentes habilidades, para la entrega continua; Las organizaciones hoy enfrentan una nueva dinámica económica: un mundo híper conectado, tecnologías accesibles a la mayor parte de la población y empresas haciendo grandes disrupciones y creando experiencias digitales que establecen un nuevo estándar para la interacción de los clientes con las marcas. En respuesta a esto todas las compañías, sin excepción, deben iniciar un proceso de transformación digital, donde el software es uno de los elementos claves. Así es relevante contar con capacidades para crear y evolucionar software con criterios de velocidad, calidad y eficiencia para preservar la existencia de las compañías. El movimiento DevOps, particularmente en las capacidades de Entrega Continua de Software define las prácticas que debe incorporar el proceso de ingeniería de software de una empresa para lograr producir y mantener software bajo las exigencias mencionadas. Esta investigación centra su atención en los procesos y prácticas del ciclo Construcción (Build), Pruebas (Test), despliegue (Deployment) y liberación en producción (Release).

		Aborda conceptos, definiciones y prácticas, y analiza la importancia de los procesos de gestión de la configuración, integración continua, automatización de pruebas y automatización del despliegue y la liberación en producción. Así mismo revisa las herramientas necesarias para implementar, simplificar, automatizar y administrar las prácticas de cada proceso. Finalmente se hace una propuesta para guiar la adopción y mejoramiento de las prácticas de entrega continua de software y sugiere un conjunto de métricas para evaluar el desempeño del equipo responsable del ciclo.
Configuración y despliegue automático de aplicaciones en unacloud	(Valencia, 2017)	En este trabajo de grado realizado por un estudiante de la universidad de los Andes en Colombia se refleja una profunda investigación en cuanto a la configuración correspondiente para los despliegues automáticos, teniendo en cuenta la nube, dicho esto nos describen el proyecto de la siguiente forma: UnaCloud es una implementación a medida del modelo IaaS de cloud computing, capaz de entregar servicios computacionales fundamentales (procesamiento, memoria RAM, almacenamiento, networking) a través del uso oportunista de los recursos de cómputo actualmente disponibles en el Campus de la Universidad de Los Andes. El objetivo de UnaCloud es buscar los proyectos de investigación de los grupos de la Universidad de Los Andes mediante el ofrecimiento de estos recursos computacionales para procesamiento científico. El proyecto hace uso de la virtualización para desplegar instancias de máquinas virtuales (configuradas por los usuarios) de forma individual y/o en clústeres sobre una infraestructura no dedicada soportada sobre equipos de escritorio en salas de informática en el campus universitario.

2.2.3 Regional

Titulo	Autor	Resumen
Diseño del proceso operacional para la implementación de contenedores en el despliegue de aplicaciones sobre la infraestructura tecnológica del área de desarrollo de la universidad de pamplona.	(SANCHEZ, 2020)	Este trabajo consiste en analizar y determinar los distintos tipos de contenedores y de plataformas de orquestación de código abierto más utilizadas en la actualidad, para diseñar un documento que indique cuál de estos es el más adecuado para la automatización en los procesos de despliegue de aplicaciones sobre la infraestructura tecnológica del área de desarrollo de la Universidad de Pamplona.

CAPÍTULO 3

3 Análisis preliminar

Esta sección se enfoca en una búsqueda detallada sobre las herramientas más conocidas para la realización del despliegue continuo de aplicaciones web y a su vez se determina cada una de las características de estas herramientas, esto con el fin de reunir una mayor cantidad de información que permita tomar una decisión que se adapte a las necesidades tecnológicas de la empresa. Cabe aclarar que dentro del mercado existen bastantes herramientas que pueden permitir o ayudar a implementar la integración y el despliegue continuo.

3.1 Herramientas que ayudan a implementar el despliegue continuo

3.1.1 Travis CI

Travis CI permite conectar un repositorio de Github y probar después de cada push que se haga, regenerando el proyecto. A demás soporta múltiples lenguajes como Clojure, Erlang, Node.js, PHP y Ruby. Esta herramienta planea soportar más lenguajes como Scala, Python y Java. (Rodríguez, 2012)

3.1.1.1 Características

- **Configuración en segundos:** Se puede iniciar sesión con la aplicación web en la nube, seguido a esto se le indica a Travis CI que haga pruebas en el proyecto, finalmente se inicia el proceso.
- **Prueba solicitudes de extracción:** Durante esta característica es importante asegurarse de que todas las solicitudes de extracción del proyecto se prueben antes de la fusión.

- **Apoya la aplicación web:** Muchas bases de datos y servicios están preinstalados y pueden habilitarse en la configuración de compilación.
- **Se puede implementar en cualquier lugar:** Permite actualizar la puesta en escena o la producción tan pronto como pasen las pruebas. (Palma, 2021)

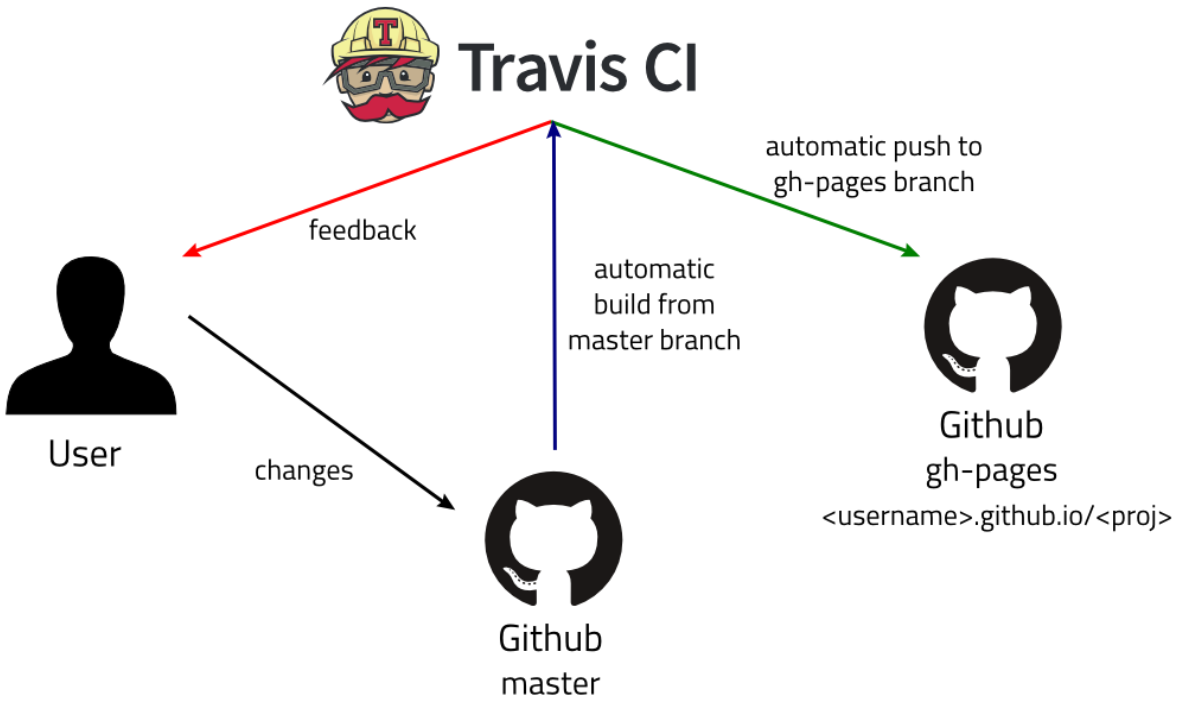


Ilustración 9. Funcionamiento de Travis CI *(Palma, 2021)*

3.1.2 AWS

Amazon Web Services (AWS) es una plataforma de servicios en la nube, esta herramienta proporciona una variedad de servicios de infraestructura tales como almacenamiento, redes, bases de datos, servicios de aplicaciones, potencia de cómputo, mensajería, inteligencia artificial, servicios móviles, seguridad, identidad y conformidad, entre otros, los cuales permiten el crecimiento de las empresas, en la ilustración 10 se puede apreciar los procesos que esta herramienta ejecuta durante el ciclo de vida del software. (Barrera, s.f.)

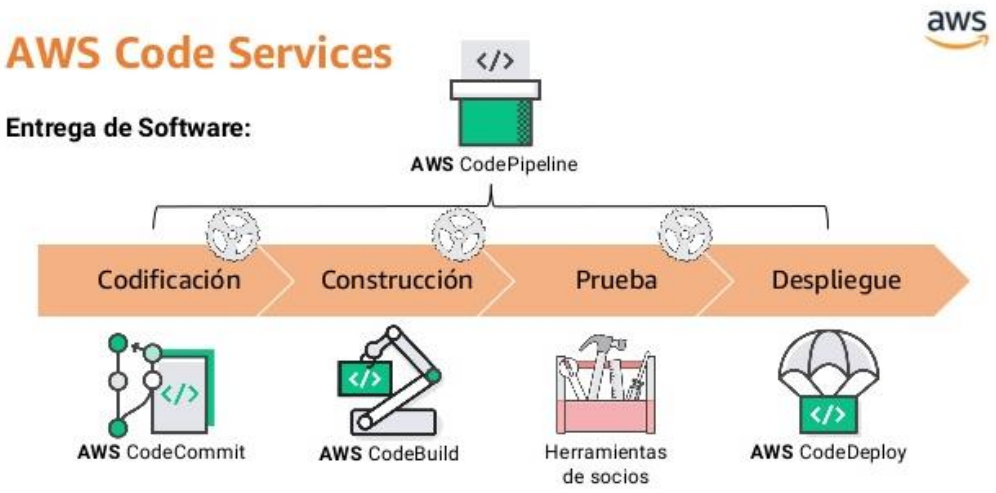


Ilustración 10. Amazon Web Services *(Barrera, s.f.)*

3.1.3 Jenkins

Jenkins es un aplicativo para la automatización del proceso completo de desarrollo de software tal y como se observa en la **Ilustración 11**, con tareas comunes como la integración continua, pero sobretodo potenciando los equipos para implementar la parte técnica de la entrega continuas.

Es un poco confuso para una persona que no está en contacto con el mundo tecnológico ya que esto puede parecer un juego de palabras, pero básicamente es tan simple como decir que disponemos de un software que nos permite automatizar los procesos de despliegue, la actualización de entornos, revisión de calidad de código y testing. Esto finalmente se traduce en entregas periódicas a cliente, en tiempo y forma con resultados que garantizan un producto funcional y listo para su explotación.

Jenkins debe su potencial a su amplio y versátil sistema de plugins que permiten realizar diferentes conjuntos de acciones, que a su vez permiten modularizar y extender la funcionalidad. (Royo, 2016)

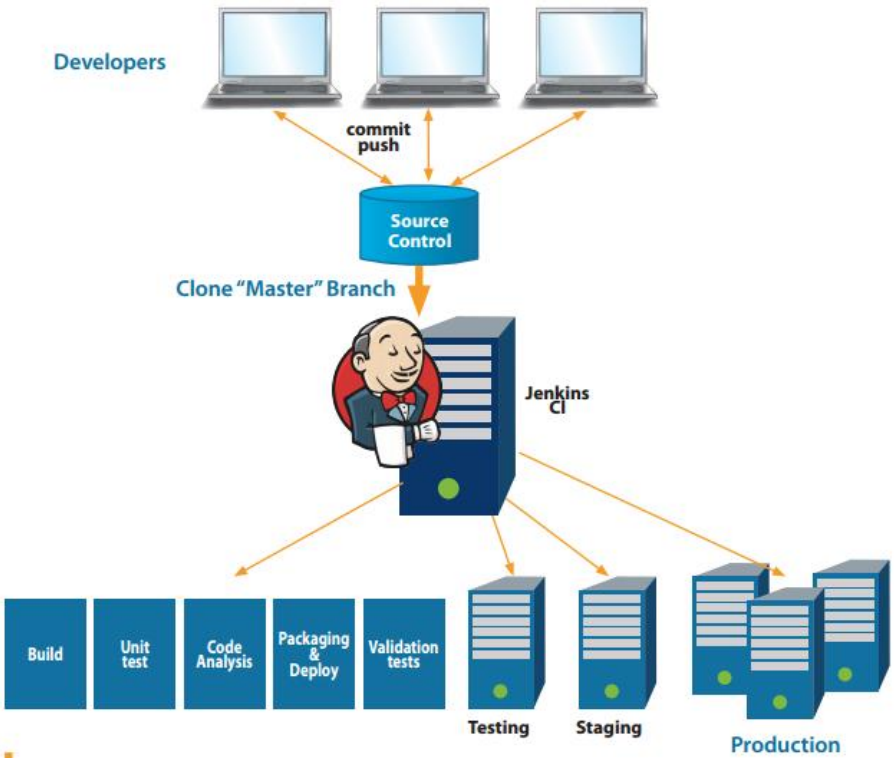


Ilustración 11. Implementación de Jenkins *(Royo, 2016)*

3.1.3.1 Características

- **Plugins**

Una de las grandes características de Jenkins es que está respaldado por una amplia comunidad, dónde cualquier desarrollador puede programar su propio plugin y ofrecer una nueva funcionalidad a nuestra aplicación de Jenkins. Algunos de los plugins más interesantes y a la vez más conocidos que usamos en nuestro día a día son: FTP, SSH, GIT Conector, Tas, Etc.

- **Publicación de contenido con FTP**

Este plugin permite publicar contenido en una máquina dónde dispongamos de acceso mediante el protocolo FTP. Este plugin es muy usado en máquinas dónde no se dispone de un usuario SSH con el que se puede acceder a la máquina. (Royo, 2016)

- **Publicación de contenido con SSH**

Este sin duda es uno de los plugins preferidos por la comunidad, ya que es realmente potente. Este plugin permite conectarnos a una máquina remota mediante SSH y realizar operaciones básicas como limpiar la cache de una aplicación, cambiar permisos a unas carpetas o ejecutar scripts en el servidor de destino. (Royo, 2016)

- **Control de calidad**

Más que a través de un plugin en concreto, el control de calidad y pruebas emplea toda una suite la cual nos permite realizar la ejecución de pruebas funcionales y unitarias y que valida cada una de las entregas entrega o cambios realizados a una aplicación.

Como complemento contamos con plugins de control de calidad de código, que nos dan brindan una serie de reportes basado en estándares de métricas que permiten aseverar el desarrollo de aplicaciones robustas y seguras. (Royo, 2016)

- **Gestión de usuarios**

Otro aspecto muy importante a tener en cuenta a la hora de usar Jenkins es la configuración o ejecución de las diversas tareas por usuario, lo que permite realizar una segmentación del acceso a cada uno de los proyectos y entornos a los miembros que estén autorizados. (Royo, 2016)

- **Control de lo que desplegamos**

A veces es difícil mantener informado al cliente de todo lo que va a recibir en un sprint o entrega. Con Jenkins y su integración con Git (el sistema de control de versiones que empleamos), este reporte es tan sencillo como darle al botón de ejecutar una tarea y obtendremos los cambios realizados con respecto al despliegue anterior. Vale la pena aclarar que en este aspecto es importantísimo que nuestros desarrolladores realicen trabajos atómicos, que no mezclen funcionalidades en cada trabajo que realizan y que se sigan algunas recomendaciones sobre escritura de mensajes. Estos mensajes finalmente pueden ser extraídos por el encargado de despliegues o el jefe de proyecto y se emplean para la elaboración de un documento más formal para entregar a los clientes con las tareas realizadas desde la última versión del aplicativo. (Royo, 2016)

3.1.4 Azure DevOps

Azure DevOps es un conjunto de herramientas y servicios que ayudan en la administración de los proyectos de desarrollo de software en cada una de sus fases. Esta herramienta es la evolución de Visual Studio Team Services. Además, una de las características más importantes es que soporta cualquier lenguaje de programación y cualquier plataforma de desarrollo. (Linkedin, s.f.)

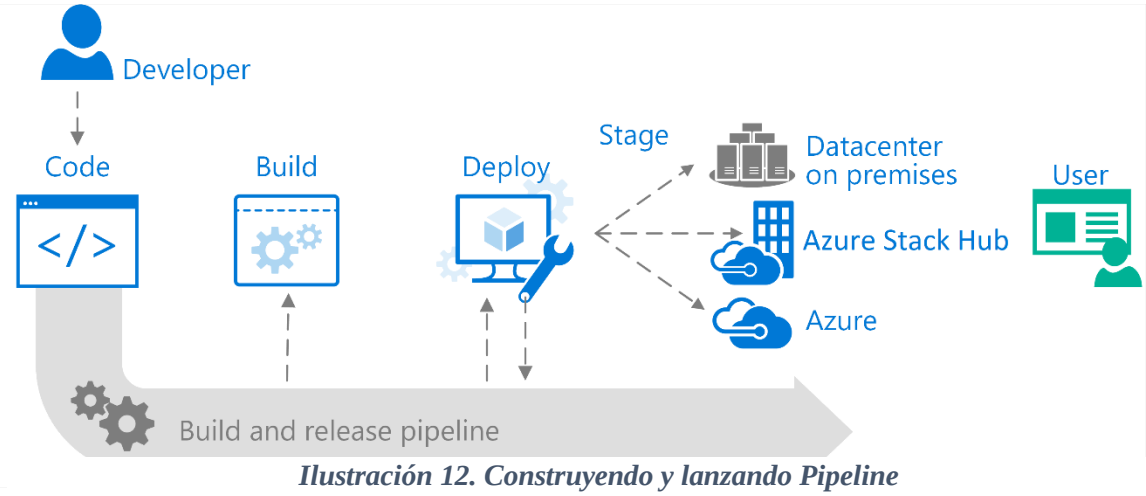


Ilustración 12. Construyendo y lanzando Pipeline

El patrón DevOps permite compilar, probar e implementar una aplicación que se ejecuta en varias nubes. Este patrón une las prácticas de integración continua y de entrega continua. Con la integración continua, el código se compila y se prueba cada vez que un miembro del equipo confirma un cambio en el control de versiones. La entrega continua automatiza cada paso de una compilación a un entorno de producción. Juntos, estos procesos crean un proceso de versión que admite la implementación en diversos entornos. Con este patrón, puede preparar el código y, a continuación, implementarlo en un entorno local, en nubes privadas diferentes y en nubes públicas. Las diferencias en el entorno requieren realizar un cambio en un archivo de configuración en lugar de cambios en el código. (Lamos, 2019)

3.1.4.1 Características

- una de sus potentes características, es su capacidad de integrarse con otras plataformas, tales como GitHub, para descargar repositorios de código, o incluso sincronizarlos hacia GitHub.
- Tiene integración con diferentes plataformas.

3.1.5 GitHub Actions

Esta herramienta permite crear flujos de trabajo (workflows) que se pueden utilizar para compilar, probar y desplegar código, dando la posibilidad de crear flujos de integración y despliegue continuo dentro del propio repositorio de git. (Márquez, 2020).

En noviembre de 2019, GitHub anunció la disponibilidad general de GitHub Actions para todos los usuarios. La función GitHub Actions permite ejecutar fragmentos de código en un contenedor en una amplia variedad de llamadas a la API de GitHub. Esto tiene la promesa de permitir a los usuarios orquestar sus flujos de trabajo en función de cualquier evento.

Ilustración 13. Flujo de trabajo de GitHub Actions *(Fraile, 2021)*

GitHub Actions es la herramienta para automatizar flujos de CI/CD en los repositorios de GitHub. (Fraile, 2021)

3.1.5.1 Características de GitHub Actions

- Brindan automatización que puede realizar una integración continua y una implementación continua.
- Pueden ejecutarse directamente en máquinas de ejecutor o en contenedores Docker.
- Permiten el acceso a un clon de un repositorio.
- No necesitan que se implemente un código o que se ejecute una aplicación.
- Tienen una interfaz simple para crear y usar secretos, que permite que las acciones interactúen con servicios de terceros sin la necesidad de almacenar las credenciales de la persona que utiliza la acción. (GitHub)

3.1.6 GitLab

GitLab es una herramienta de ciclo de vida y repositorio de Git. Es una completa plataforma DevOps, que permite a los profesionales gestionar y realizar diversas tareas del proyecto. Las tareas incluyen la planificación del proyecto, la gestión del código fuente, el mantenimiento de la seguridad y el seguimiento. (Pathak, 2021)

Por defecto Gitlab define tres etapas las cuales son: build, test y deploy tal y como se observa en la **Ilustración 14.**

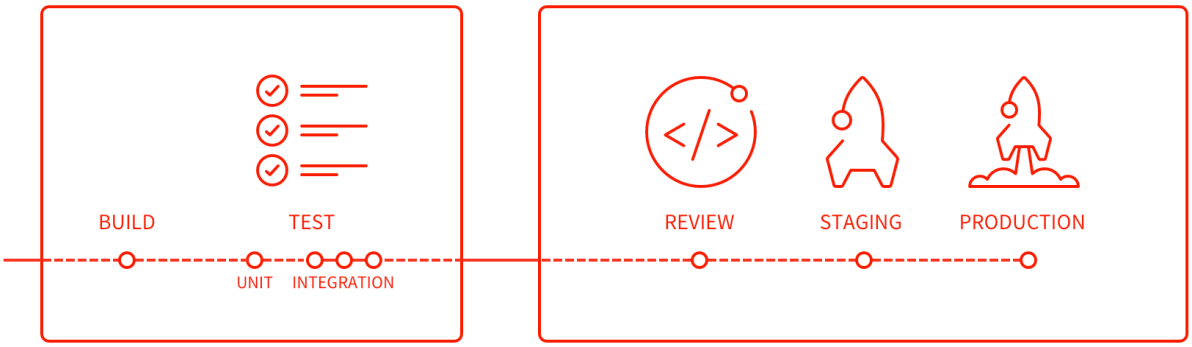


Ilustración 14. Compilación e implementación de GitLab en un servidor *(Pathak, 2021)*

Por ello, podemos crear distintos trabajos en distintas estapas. Cabe decir, que si ponemos dos trabajos en una misma estapa, se ejecutará de forma paralela. (MARTÍN, s.f.)

3.1.6.1 Características

- Pipelines CI / CD potentes y bien definidos
- Contenedores Docker
- Registro incorporado implementado instantáneamente sin configuración.
- Admite servidores de terceros para administrar imágenes de Docker
- Seguimiento de problemas
- Escrito en Ruby and Go

3.1.7 Docker

Esta herramienta es una de las más comunes, Docker es una plataforma abierta para desarrollar, enviar y ejecutar aplicaciones. Docker le permite separar sus aplicaciones de su infraestructura para que pueda entregar software rápidamente. Con Docker, puede administrar su infraestructura de la misma manera que administra sus aplicaciones. Al aprovechar las metodologías de Docker

para enviar, probar e implementar código rápidamente, puede reducir significativamente el retraso entre la escritura del código y su ejecución en producción.

Docker brinda la capacidad de empaquetar y ejecutar una aplicación en un entorno poco aislado llamado contenedor. El aislamiento y la seguridad le permiten ejecutar muchos contenedores simultáneamente en un host determinado. Los contenedores son livianos porque no necesitan la carga adicional de un hipervisor, sino que se ejecutan directamente dentro del kernel de la máquina host. Esto significa que puede ejecutar más contenedores en una combinación de hardware determinada que si estuviera utilizando máquinas virtuales

El funcionamiento de este herramienta proviene del motor de Docker o tambien llamado Docker Engine, que es una aplición cliente-servidor con estos componentes principales:

- Un servidor que es un tipo de programa de larga duracion llamado proceso demonio, tambien llamado “the dockerd command”.
- Una API REST que especifica interfaces que los programas pueden usar para hablar con el demonio e indicarle que hacer.
- Un cliente de interfaz de linea de comandos (CLI)(the docker command).

Como se evidencia en la **ilustración 15** (Docker docs, 2020), el CLI utiliza la API REST de Docker para controlar o interactuar con el demonio de Docker a través de secuencias de comandos o comandos directos de la CLI. Muchas otras aplicaciones de Docker utilizan la API y la CLI subyacentes. El Daemon crea y administra objetos Docker, como imágenes, contenedores, redes y volúmenes.

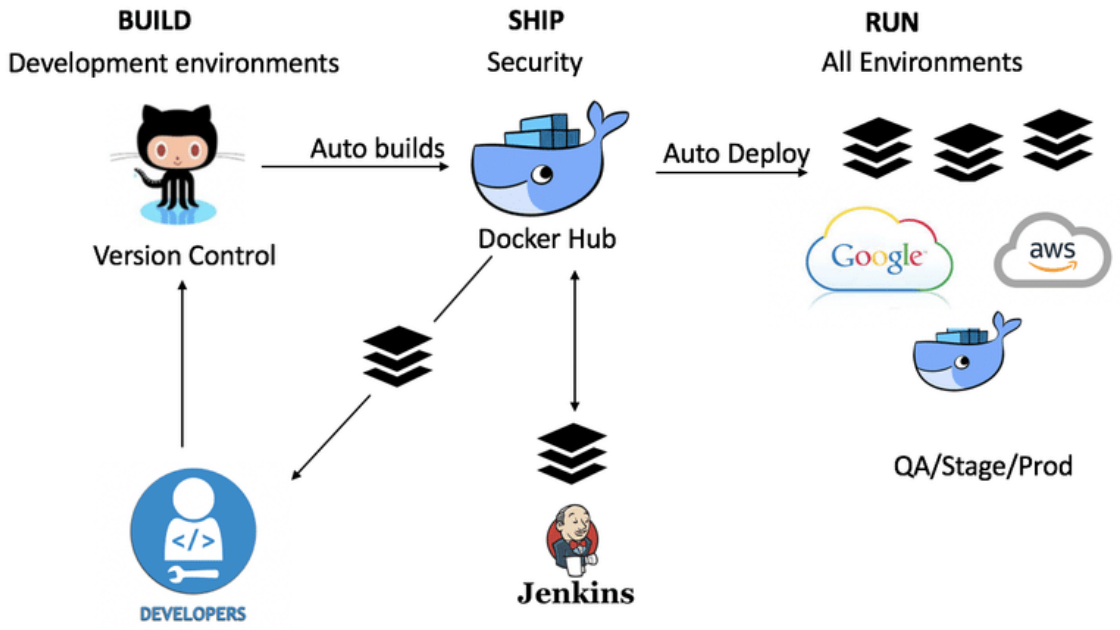


Ilustración 15. Arquitectura CI/CD con Docker (Docker docs, 2020),

CAPÍTULO 4

4 Definición base

En este capítulo se llevó a cabo la toma de decisiones para determinar la herramienta de despliegue continuo más conveniente para la ejecución actual del proyecto en la empresa BeGo, teniendo presente la información entregada en el capítulo anterior, de igual modo esta decisión se toma teniendo en cuenta el proceso de pruebas e integración continua ya estandarizado en la empresa realizado en los trabajos de (VILLAMIZAR, 2020) y (Carrillo, 2020). En este capítulo se describe cuáles fueron los factores se tuvieron en cuenta para la selección de las herramientas, esto con la realización de cuadros comparativos.

4.1 Ventajas y Desventajas de los tipos de herramientas para el despliegue continuo de aplicaciones web.

Para la selección de esta, se realizó una previa exploración de las principales herramientas utilizadas en este campo que cumplieran con los requisitos que se contaban para el proyecto, como lo es:

- Open source
- Fácil implementación
- Automatización del proyecto
- Que no necesite de muchas herramientas adicionales.
- Que se adapte a cualquier tipo de proyecto tanto actual como futuro.
- Que maneje la mayor cantidad de lenguajes.

4.1.1 Travis

Tabla 1 Ventajas y desventajas de Travis

Ventajas	Desventajas
➢ Ligero y fácil de configurar ➢ Gratis para proyectos de código abierto ➢ No se necesita servidor dedicado ➢ Función de matriz de construcción (Rodríguez, 2012)	➢ Los planes empresariales tienen un costo ➢ Opciones limitadas de personalización

4.1.2 AWS

Tabla 2. Ventajas y desventajas de Amazon Web Services

Ventajas	Desventajas
➢ AWS ofrece más de 50 servicios de manera rápida y sencilla a las empresas. Con tan solo un par de clics y sin gastos adelantados, los clientes pueden tener acceso a los elementos básicos que necesitan para responder rápidamente a las diferentes necesidades empresariales. ➢ La seguridad en la nube es mejor que cualquier infraestructura física y es prioridad para AWS. Las certificaciones y acreditaciones, cifrado de datos en reposo y en tránsito, módulos de	➢ No apto para amateurs: Amazon Web Services no es para principiantes ya que el sistema es muy complejo, si no se tiene la experiencia previa y si lo que se busca es montar un blog o una web sencilla, se debe saber que la cantidad de plugins a usar es tal, que probablemente puede conducir a la frustración. En ese sentido, hay otras plataformas enfocadas al pequeño usuario que pueden funcionar.

seguridad hardware y una fuerte seguridad física, permiten implementar funciones de seguridad para administrar la infraestructura de TI de las empresas de manera segura y confiada. Amazon Web Services permite que sus clientes crezcan e innoven simultáneamente, garantizándoles la seguridad. ➢ Esta plataforma integra capacidades para controlar, auditar y administrar la identidad, configuración y uso, lo cual permite satisfacer los requisitos de conformidad, gobernanza y normativa. ➢ Bajo costo: Amazon ha pensado esta plataforma con la mentalidad bajo costo. Cuantos más usuarios la usen, más económico es. ➢ Es escalable: cada usuario puede contratar lo que quiera y configurar su servicio como quiera. No hay un solo modo de utilizar AWS, sino múltiples y casi ilimitados. (Barrera, s.f.)	➢ No hay excusas: El nivel de seguridad de AWS es tal, que muchas empresas se han tenido que poner las pilas para cumplir con los estándares de cumplimiento de esta plataforma. ➢ Es una plataforma genérica que sirve para todo tipo de negocios pero que no está especializada en entornos regulados, es decir, sector salud, farmacéutica o productos sanitarios. (Ambit, 2020)

4.1.3 Jenkins

Tabla 3. Ventajas y desventajas de Jenkins

Ventajas	Desventajas
➢ Es de código abierto y es fácil de usar, fácil de instalar y no requiere instalaciones o componentes adicionales. ➢ Es gratis. ➢ Fácilmente configurable. Jenkins se puede modificar y extender fácilmente. Implementa código de forma instantánea, genera informes de prueba. Jenkins se puede configurar de acuerdo con los requisitos de integraciones continuas y entrega continua. ➢ Plataforma independiente. Jenkins está disponible para todas las plataformas y diferentes sistemas operativos, ya sea OS X, Windows o Linux. ➢ Rich Plugin ecosystem. El amplio conjunto de complementos hace que Jenkins sea flexible y permita construir,	➢ Una de las razones por las que muchas personas no implementan Jenkins se debe a su dificultad para instalar y configurar. ➢ Las integraciones continuas se rompen regularmente debido a algunos pequeños cambios de configuración. La integración continua se detendrá y, por lo tanto, requiere la atención del desarrollador ➢ Su interfaz está desactualizada y no es fácil de usar en comparación con las tendencias actuales de la interfaz de usuario. ➢ Aunque Jenkins es amado por muchos desarrolladores, su mantenimiento no es tan fácil porque Jenkins se ejecuta en un servidor y requiere algunas habilidades

implementar y automatizar en varias plataformas. ➢ Fácil soporte debido a que es de código abierto y ampliamente utilizado, no hay escasez de soporte de grandes comunidades en línea de equipos ágiles. ➢ El desarrollador escribe las pruebas para detectar los errores de su código lo más rápido posible. De modo que el tiempo del desarrollador se guarda sin desperdiciar integraciones plagadas de errores a gran escala. ➢ Los problemas se detectan y resuelven casi de inmediato, lo que mantiene el software en un estado en el que se puede liberar en cualquier momento de forma segura. ➢ La mayor parte del trabajo de integración está automatizado. Por lo tanto, los problemas de integración son menores. Esto ahorra tiempo y dinero	como administrador del servidor para monitorear su actividad

durante la vida útil de un proyecto. (Pérez, apiumhub.com, 2017)	

4.1.4 Azure DevOps

Tabla 4. Ventajas y desventajas de Azure DevOps

Ventajas	Desventajas
➢ incluye diferentes servicios, que son Azure Repos; el cual es un servicio para el almacenamiento de repositorios de código fuente, de cualquier lenguaje de programación y de cualquier plataforma de desarrollo. ➢ Soporta dos sistemas de control de versiones o VCS, por sus siglas en inglés de Version Control System, las cuales son Git y Team Foundation Version Control. ➢ Con Azure Pipelines, puedes crear definiciones de pipelines, de integración continua y de despliegue continuo para automatizar las fases de desarrollo,	➢ Azure es diferente de los servidores locales y requiere experiencia para asegurar que todas las partes móviles funcionen de la manera más eficiente posible. Incluso un error común cometido por un administrador de TI de una empresa puede crear problemas. (Henson, 2017)

compilación, pruebas y despliegue de tus proyectos de software. ➢ Cuenta con una gran cantidad de tareas preconstruidas para prácticamente hacer cualquier cosa que requiera tu software para compilar y desplegarse. Ppuede integrarse con visualstudio.net para que los desarrolladores puedan rastrear sus tareas asignadas, así como la asignación nueva de elementos. (Linkedin, s.f.)	

4.1.5 GitHub Actions

Tabla 5. Ventajas y desventajas de GitHub Actions

Ventajas	Desventajas
➢ Es bastante sencillo de configurar ➢ No se necesita instalar infraestructura (está basado en un entorno cloud). ➢ La principal configuración se realiza mediante un fichero YML el cual está basado en el lenguaje de programación YAML. El código contenido en un archivo YML es legible por el ser	➢ Poco claro en la documentación disponible

humano y generalmente se utiliza para serializar datos. ➢ Ofrece una versión community edition donde podemos jugar con nuestros proyectos de prueba o usarlo para nuestros proyectos open source (sólo proyectos públicos). ➢ Ofrece una versión enterprise para nuestros proyectos privados. (Diez, 2020)	

4.1.6 Git Lab

Tabla 6. Ventajas y desventajas de GitLab

Ventajas	Desventajas
➢ GitLab es fácil de configurar y administrar con sus códigos disponibles gratuitamente. ➢ La aplicación es única y crea un flujo de trabajo optimizado con colaboración y eficiencia.	➢ Interfaz comparativamente lenta. ➢ A menudo surgen problemas habituales con los repositorios. ➢ No es absolutamente abierto. ➢ Tiene limitaciones de espacio, ya que no puedes exceder de 100MB en un solo archivo, mientras que los repositorios

➢ Las revisiones de código junto con las solicitudes de extracción son fáciles de usar y compactas. ➢ Al ser una aplicación nativa de la nube con fuertes medidas de seguridad, ofrece características de seguridad como restricciones granulares, autenticación de usuario con Kerberos. ➢ Integración mínima para reducir el ciclo de vida del desarrollo mientras aumenta la productividad Facilita una organización adecuada de una orquestación de contenedoresy la integración ➢ Permite una gestión de proyectos amplia y adaptable para acelerar el flujo de trabajo (Isaac, 2021)	están limitados a 1GB en la versión gratis. (Isaac, 2021)

4.1.7 Docker

Tabla 7. Ventajas y desventajas de Docker

Ventajas	Desventajas
➢ Docker es compatible con diversos sistemas operativos y plataformas como: Linux, Windows, MacOS, Microsoft Azure, Amazon Web Services (AWS). ➢ Con las herramientas Docker Swarm y Docker Compose, la plataforma Docker ya ofrece herramientas de gestión de clústers. ➢ Los usuarios cuentan con un amplio repositorio de imágenes, ya sea imágenes oficiales o no oficiales, gracias al apoyo de empresas externas y la comunidad. ➢ Docker está en constante crecimiento, brinda una gran documentación y pone a disposición de sus usuarios diversas herramientas, plugins y componentes de infraestructura.	➢ El motor de Docker solo es compatible con su propio formato de contenedor. ➢ Los contenedores Docker están enfocados a aislar procesos entre sí, pero no virtualizan sistemas operativos completos. ➢ No proporciona una opción de almacenamiento. ➢ Mal seguimiento. ➢ Sin reprogramación automática de nodos inactivos. ➢ Las acciones se deben realizar en CLI. ➢ Gestión manual de múltiples instancias. ➢ Necesita soporte para otras herramientas. ➢ Implementación manual de clúster complicada. ➢ Sin soporte para controles de seguridad

➢ Configuración inicial eficiente y sencilla. ➢ Permite rastrear las versiones de contenedores para examinar variaciones. ➢ Rapidez. ➢ Buen aislamiento entre apps. (Isaac, blog.desdelinux.net, 2021)	➢ Docker es una empresa con fines de lucro y algunos de sus componentes críticos, como Docker Engine y Docker Desktop, no son de código abierto. (Isaac, blog.desdelinux.net, 2021)

4.2 Diferencias entre GitHub Actions y GitLab

GitHub también es un repositorio basado en Git como GitLab. Lanzado en 2008, es el más grande de la categoría con más de 40 millones de usuarios.

Los proyectos de GitHub son tanto públicos como privados. Los códigos compartidos públicamente son abiertos y gratuitos (según la licencia) para todos en Internet. Por lo tanto, muchos desarrolladores usan repositorios públicos para compartir su software de código abierto en GitHub.

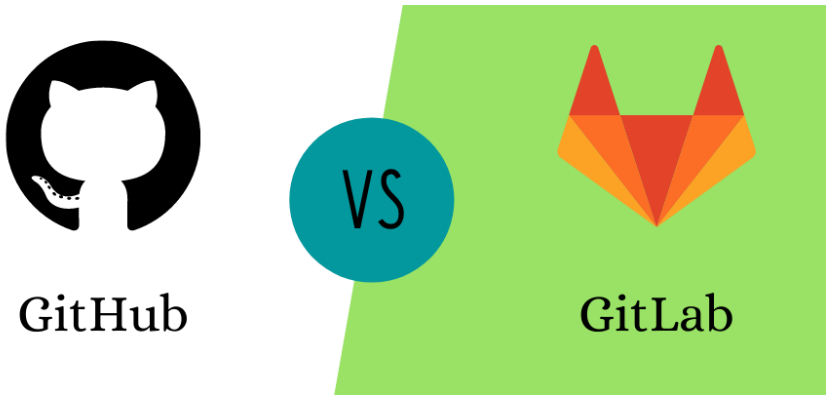


Ilustración 16. GitHub vs GitLab

4.2.1 Diferencias GitHub vs GitLab

Tabla 8. Diferencias entre GitHub y GitLab

Características	GitLab	GitHub
Inicio	Septiembre de 2011	Abril de 2008
Plan gratuito	Repositorios públicos y privados ilimitados	Gratis solo para repositorios públicos
Planes pagados	Desde 19$ por usuario por año para el Plan Premium. O 99$ por usuario y año para el Ultimate.	Desde $4 por usuario y año para el Team, 21$ para el Enterprise, o más para el One.
Funciones de revisión de código	SI	SI
Wiki	SI	SI
Seguimiento de errores y problemas	SI	SI
Rama privada	SI	SI

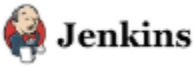
Sistema de construcción	SI	sí (con servicio de terceros)
Importar proyectos	SI	No
Exportar proyectos	SI	No
Seguimiento del tiempo	SI	No
Alojamiento web	SI	SI
Autohospedaje	SI	sí (con plan empresarial)
Popularidad	546.000+ proyectos	69.000.000+ proyectos

Como se ve en la tabla, no existe un claro ganador. La elección de alguno de los dos no es tan fácil, sí queremos determinar por cual nos inclinamos más deberíamos volver a ver las ventajas y desventajas de cada uno para poder identificar cuál se adapta mejor a las necesidades de la empresa.

Para tomar una buena decisión se puede tener en cuenta lo siguiente: sí se quiere disponer de un entorno totalmente abierto, es mejor se usa GitLab. Por otro lado, sí se buscan más facilidades y usar el servicio web con más presencia, entonces se puede escoger GitHub.

4.3 Comparación resumida de lo que pude ofrecer cada una de las herramientas expuestas en el capítulo 3

Tabla 9. Comparativo de todas las herramientas-primera parte

Herramienta	Travis	AWS	Jenkins	Azure DevOps
				
Configuración	Configuración de despliegue por medio de archivos YML	Configuración de pipeline por medio de archivo YML o por medio de la interfaz gráfica (GUI)	Configuración de pipeline por medio de Jenkinsfile	Configuración de pipeline por archivo YAML ó por editor clásico (GUI)
Ejecución	Versión open source hasta 5 pipelines al tiempo	Una sola ejecución de un pipeline al tiempo	Múltiples ejecuciones de pipelines al tiempo	Una sola ejecución de un pipeline al tiempo
Compatibilidad	Alta compatibilidad con GitHub y BitBucket	Compatible con GitHub, BitBucket pero prioriza AWS CodeCommit	Compatible con múltiples sistemas de control de versiones	Compatible con diferentes sistemas de control de versiones, pero

				prioriza a Azure Repos
Uso	Proporciona una versión gratuita con ciertas limitaciones, superadas estas se incurre en gastos.	Proporciona una versión gratuita con ciertas limitaciones, superadas estas se incurre en gastos.	Proporciona una versión gratuita con ciertas limitaciones, superadas estas se incurre en gastos.	Proporciona una versión gratuita con ciertas limitaciones, superadas estas se incurre en gastos.

Tabla 10. Comparativo de todas las herramientas-Segunda parte

Herramienta	**GitHub Actions**	**GitLab**	**Docker**
			
Configuración	Configuración de despliegue por medio de archivos YML	Configuración de despliegue por medio de archivos YML	Configuración de despliegue por medio de Contenedores
Ejecución	Múltiples ejecuciones de pipelines al tiempo	Múltiples ejecuciones de pipelines al tiempo	Múltiples contenedores al tiempo
Compatibilidad	Cuenta con un sistema de control de versiones	Tiene su propio sistema de control de versiones	Alta compatibilidad con GitHub y Jenkins

Uso	Proporciona una versión gratuita con ciertas limitaciones, superadas estas se incurre en gastos.	Proporciona una versión gratuita con ciertas limitaciones, superadas estas se incurre en gastos.	Es de código abierto. Esto significa que cualquiera puede contribuir a Docker y ampliarlo para satisfacer sus propias necesidades si necesita funciones adicionales que no están disponibles de forma inmediata.

4.4 Selección de herramientas

Después de indagar en profundidad en cada una de estas herramientas y teniendo en cuenta la etapa de integración continua que actualmente se lleva a cabo en las instalaciones de BeGo, se puede realizar la toma de decisiones para la herramienta con la que se busca automatizar el procedimiento de despliegue continuo dentro de la empresa y dar así cabalidad al objetivo general del proyecto.

Vale la pena resaltar que el procedimiento de despliegue continuo va muy ligado al procedimiento de integración continua el cual ya fue automatizado anteriormente.

Después de algunas reuniones con el equipo de desarrollo de la empresa, se llegó a la conclusión de que la mejor herramienta que se podía implementar en este momento para agilizar los procesos internos y tener algo de ventaja era la misma herramienta con la que se venía trabajando el proceso de integración continua. Esto debido a que incurrir en un cambio de herramienta provocaría un retroceso en lo que ya está construido, en teoría el cambio de herramienta implicaría nuevamente generar las pruebas y la integración continua. Por tal razón y aprovechando que GitHub Actions permite la automatización de integración y despliegue continuo, seguiremos trabajando en este campo, tal y como se venía realizando en la empresa para cumplir en totalidad con los criterios que se piden.

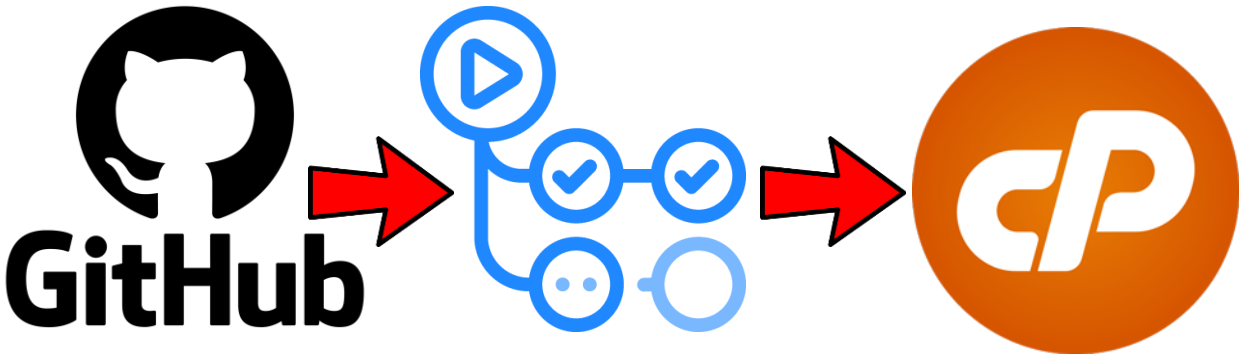


Ilustración 17. Selección de herramientas (Autor)

Hasta el momento la empresa ha logrado automatizar de forma correcta algunos procesos, estos procesos hacen parte del ciclo de vida del software es por ello que en algunas de las etapas ya pasadas en la empresa se eligió trabajar con GitHub.

De igual manera se ha venido trabajando en la integración continua, hasta el momento se logró automatizar con la herramienta GitHub Actions, la cual en esta ocasión permitirá dar continuidad a esa completa automatización que se busca incluyendo el despliegue automático.

Por otro lado, la empresa actualmente cuenta con los servicios contratados de Cpanel, este es uno de los paneles de control basados en Linux más populares para cuentas de hosting web.

Permite administrar de manera más cómoda todos los servicios en un solo lugar. Actualmente, cPanel es el estándar de la industria y lo mejor es que la mayoría de los desarrolladores web lo conocen bien, es por esta razón que los desarrolladores decidieron trabajar con este servicio para el alojamiento de los sitios web.

4.4.1 Seguridad en cPanel

La seguridad en cPanel se controla con algunas herramientas que este mismo ofrece, las cuales permiten proteger directorios y diferentes partes de los sitios web, de esta manera se evitan accesos no autorizados.

Las herramientas más utilizadas para brindar seguridad al servidor son las siguientes:

Bloqueador de Ip:

Esta herramienta permite bloquear un rango de direcciones IP para prevenir el acceso de ellas al sitio web. Se pueden poner dominios completos calificados, el administrador de negación de IP intentará resolver el dominio a la dirección de IP correspondiente para el sitio web. Si se desea agregar una IP a la cual se le quiere negar el acceso al sitio, será necesario que se ingrese la IP en el campo de escritura.

Protección Leech:

La Protección de Leech permite prevenir a los usuarios dar la contraseña públicamente a un área restringida del sitio. Esta característica direccionará y suspenderá si es necesario cuentas que han sido comprometidas a una url.

SSL/TLS:

Las funciones SSL y TLS mejoran la seguridad del sitio web que se tenga montado en el servidor. Estas funciones generan y administran los certificados SSL, solicitudes de firma y claves. Estas son útiles para sitios web que trabajan regularmente con información sensitiva, como las

credenciales de inicio de sesión y números de tarjetas de crédito. Esta información sensitiva será enviada hacia y desde su sitio web como datos codificados en vez de como texto sin formato. Esta codificación hace que sea difícil para que los usuarios maliciosos intercepten comunicaciones entre los visitantes y el sitio web.

Protección de enlace directo:

La protección Hotlink impide que otros sitios web establezcan vínculos directos a archivos del sitio web cargado al servidor. Un ejemplo de hotlinking sería usar una etiqueta <img> para desplegar una imagen desde la página web cargada a otra parte del Internet. El resultado final es que el otro sitio está consumiendo el ancho de banda del servidor de cPanel.

Protocolo FTP

Cuando se suben archivos al servidor utilizando el protocolo FTP se suele insertar los datos del servidor, nombre de usuario, contraseña, puerto y también se puede elegir el tipo de cifrado para la conexión, las opciones serán si requiere FTP explícito o implícito sobre TLS o utilizar FTP plano.

Si se utiliza la última opción los datos transferidos no serán encriptados incluyendo los datos de acceso. Esto no es lo más recomendable además se puede estar susceptible a hackers que pueden captar la información confidencial y modificarla a su antojo, esto porque no hay ninguna protección SSL.

Por esta razón se hace conveniente utilizar el protocolo FTP vía SSH o con una conexión segura ya que con solo cambiar y proteger estos 2 datos se está dando un gran paso en la protección FTP del servidor.

CAPÍTULO 5

5 Implementación

De acuerdo a la selección de herramientas en el capítulo 4, en esta sección del proyecto se dará una explicación detallada de la forma correcta en la que se deben usar estas herramientas para lograr automatizar los despliegues en cualquier empresa, teniendo en cuenta cada uno de los estándares en este documento y la forma en la que se venía trabajando.

Cabe aclarar que en la empresa el equipo de desarrollo hace aproximadamente seis meses logró automatizar dos procesos necesarios para la realización del despliegue continuo; por ende, se mencionarán y describirán brevemente las herramientas implicadas ya explicadas en documentos pasados de la empresa, estos para contextualizar y llegar al proceso de despliegue continuo.

En la **ilustración 18** se muestra el diagrama de flujo que refleja una serie de procesos que deberán estar ya configurados para llegar a la automatización del proceso de despliegue.

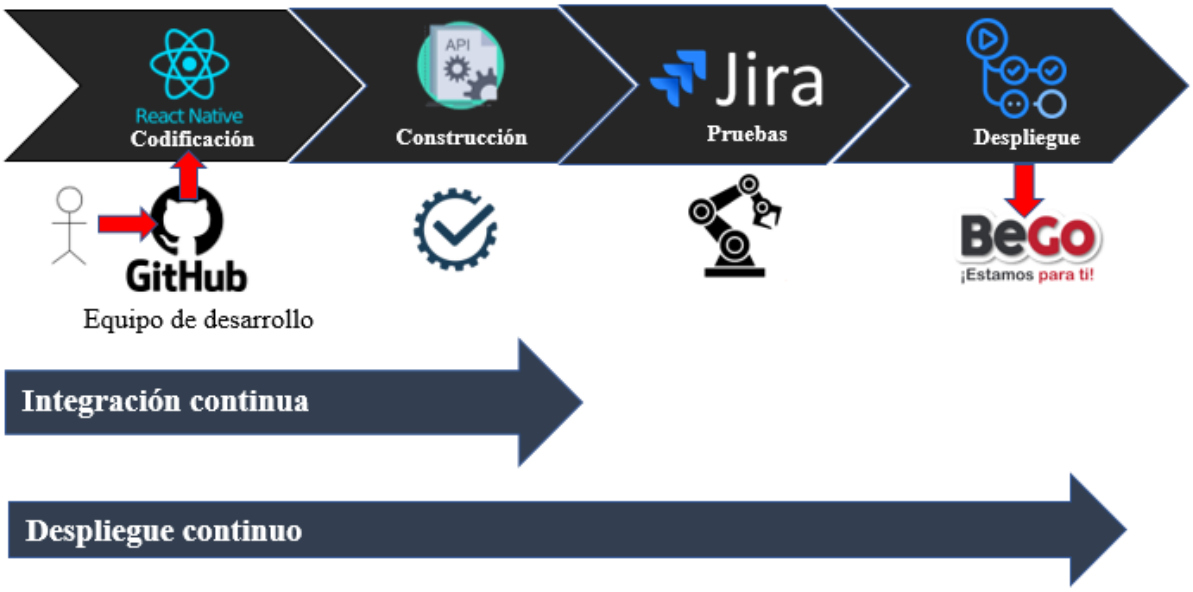


Ilustración 18. Procesos previos al despliegue. (Autor)

El flujo de trabajo para poder configurar de manera general cada una de las herramientas aquí mostradas en el capítulo anterior se ve en la **ilustración 19**

Ilustración 19. Diagrama general del procedimiento. (Autor)

5.1 Jira

En primer lugar, Jira (Software Cloud) fue seleccionada por el equipo de desarrollo de BeGo ya que proporciona herramientas de planificación y hojas de ruta para que los equipos puedan gestionar a los interesados, los presupuestos y los requisitos de las funciones desde el primer día.

A demás esta herramienta se integra en una amplia variedad de herramientas de CI y CD para facilitar la transparencia durante el ciclo de vida de desarrollo de software.

5.2 GitHub

GitHub fue estandarizada por el equipo de desarrollo teniendo en cuenta el trabajo grupal que se lleva a cabo día tras día en la empresa, fue seleccionada sin duda alguna , ya que es muy conocida por los desarrolladores, porque cuenta con una amplia comunidad, es decir que hay bastante documentación, es de código abierto, maneja diferentes tipos de lenguaje, tiene opciones para repositorios públicos o privados, es una herramienta muy fácil de usar y se adapta por completo a la forma de trabajo de la empresa, teniendo en cuenta que siempre se trabaja por equipos.

5.2.1 Implementación de GitHub

Para la implementación de GitHub en la empresa como repositorio de código, se decidió separar el proyecto en dos repositorios, en el primer repositorio se encuentra todo el código relacionado a desarrollo móvil con react native, y en el segundo repositorio el desarrollo web con react, para este proyecto solo nos centramos en el segundo repositorio. En cada uno de los repositorios cada programador cuenta con su propia rama en la que se trabaja de acuerdo a las funcionalidades asignadas; se cuenta con una serie de reglas para que no haya choques entre ramas o errores en la rama principal.

Las ramas se trabajan de la siguiente manera:

- **Rama Master:** Ninguno de los programadores puede trabajar sobre esta rama, ya que es donde se encuentra el proyecto principal.
- **Rama Dev:** cada desarrollador puede sacar su propia rama y trabajar en lo que corresponde,

- Sí existen conflictos se deben corregir generalmente el encargado de esta tarea es el líder del proyecto o de la integración continua
- Sí una funcionalidad está correctamente implementada y probada se deberá integrar inmediatamente con master, para que la rama principal siempre este actualizada

5.3 GitHub Actions

Basados en la documentación oficial de Git, se llegó a la conclusión que las acciones de GitHub ayudan a automatizar tareas dentro del ciclo de vida del desarrollo del software. Teniendo presente que todas las acciones de GitHub están controladas por eventos, esto significa que puede ejecutar una serie de comandos después de que se haya producido un evento específico. Lo primero que se debe hacer es crear una carpeta en la cual se añadirá el fichero de configuración del workflow o evento. Esta carpeta por norma tiene que llamarse. **github/workflows/** Aquí se crea el fichero workflow.yml en donde se encontrarán todos los eventos los cuales activarán automáticamente los flujos de trabajo, que contiene un trabajo. Luego, el trabajo usa estos pasos para controlar el orden en que se ejecutan cada una de las acciones. Estas acciones son los comandos que automatizan las pruebas de software, el procedimiento de integración y el despliegue en el servidor.

5.3.1 Listado de componentes de GitHub Actions

A continuación, se describen cada uno de los componentes de GitHub Actions, los cuales tienen mucha importancia en la configuración de los archivos con extensión. yml para la automatización de procesos.

- **Workflows:** Estos indican el flujo de trabajo, son procedimientos automatizados que se agregan a los repositorios, para generar un tipo de instrucciones. Los flujos de trabajo se componen de uno o más trabajos y pueden ser programados o activados por un tipo de evento.

El flujo de trabajo se puede usar para construir, probar, empaquetar, lanzar o implementar un proyecto en GitHub.

- **Events:** Un evento es una actividad en específico que permite el desencadenamiento de un flujo de trabajo. Por ejemplo, la actividad puede originarse en GitHub cuando alguien envía una confirmación a un repositorio o cuando se crea un problema o una solicitud de extracción. También puede utilizar el webhook de envío del repositorio para activar un flujo de trabajo cuando se produce un evento externo. (VILLAMIZAR, 2020)
- **Jobs:** Un trabajo es un conjunto de pasos que se ejecutan en el mismo corredor. De forma predeterminada, un flujo de trabajo con varios trabajos ejecutará esos trabajos en paralelo. También puede configurar un flujo de trabajo para ejecutar trabajos de forma secuencial. Por ejemplo, un flujo de trabajo puede tener dos trabajos secuenciales que compilan y prueban código, donde el trabajo de prueba depende del estado del trabajo de compilación. Si el trabajo de compilación falla, el trabajo de prueba no se ejecutará.
- **Steps:** Conocidos como pasos, estos no son más que tareas individuales que pueden ejecutar comandos (conocidos como acciones). Cada paso de un trabajo se ejecuta en el mismo corredor, lo que permite que las acciones de ese trabajo compartan datos entre sí.
- **Actions:** Las acciones son comandos independientes que se combinan en pasos para crear un trabajo. Las acciones son el bloque de construcción portátil más pequeño de un flujo de trabajo. Puede crear sus propias acciones o utilizar acciones creadas por la comunidad de GitHub. Para usar una acción en un flujo de trabajo, debe incluirla como un paso.
- **Runners:** Un corredor es un servidor que tiene instalada la aplicación de ejecución de acciones de GitHub. Puede utilizar un corredor alojado en GitHub, o puede alojar el suyo propio. Un corredor escucha los trabajos disponibles, ejecuta un trabajo a la vez e informa el progreso, los

registros y los resultados a GitHub. Para los ejecutores alojados en GitHub, cada trabajo de un flujo de trabajo se ejecuta en un entorno virtual nuevo

5.3.2 Implementación de GitHub Actions para el Despliegue web

A continuación, se dará a conocer el flujo de trabajo, la configuración y cada uno de los pasos que se llevaron a cabo para la completa automatización del despliegue de aplicaciones web para la empresa desarrolladora de software BeGo, se mostrara cómo desplegar automáticamente un repositorio privado de GitHub en cPanel con actualizaciones 100% en vivo.

Dichos pasos se llevaron a cabo teniendo en cuenta que ya se había creado una acción que automatizaba todo el proceso de pruebas e integración, lo que se hizo fue dar continuidad a la configuración que allí se llevaba, esto teniendo en cuenta que las pruebas que se habían realizado eran de forma local, lo que implicaba agregar algunos comandos que nos permitieran desplegar todo directamente en el servidor contratado por la empresa, en este caso Cpanel.

5.3.2.1 Flujo de trabajo

El flujo de trabajo para poder configurar de manera correcta cada una de las herramientas aquí mostradas es el siguiente:

- Se deben llevar a cabo algunos cambios en la máquina local.
- Se debe crear una rama para realizar algunos cambios a los repositorios
- Se realizarán los cambios en la nueva rama de github.
- Una vez que se haya terminado con la etapa pruebas y todo parezca perfecto, se buscará integrar estos nuevos cambios con la rama maestra, ahora bien, supongamos que la rama maestra es la rama de producción, técnicamente sabemos que todo lo que envíe a la rama maestra estará listo para funcionar.

- A continuación, lo que se busca es Desplegar todo el proyecto probado e integrado previamente, se despliega la rama maestra remota y se implementa automáticamente en el servidor que utiliza la empresa, en este caso cPanel, con una característica especial y esta permitirá que los usuarios aprecien los cambios en vivo.

5.3.2.2 Dependencias para la implementación

Para lograr este proceso de automatización se necesitará disponer de algunas configuraciones previas, tanto en el servidor como en GitHub.

- Una cuenta FTP o SFTP creada en la cuenta de cPanel
- Un subdominio creado y configurado en la cuenta de cPanel, este con el objetivo de realizar las pruebas antes de tocar el proyecto principal.
- Un proyecto que se encuentre bajo el control de versiones de Git y alojado en GitHub
- Una configuración de Action secrets en la que se incluirán algunos datos para la configuración FTP con el servidor
- Una configuración que permita el flujo de trabajo en GitHub Actions

5.3.2.3 Pasos para la implementación de GitHub Actions en el despliegue automático con cPanel

1. Inicialmente se crea un subdominio con el que se van a realizar cada una de las pruebas para la validación del despliegue automático, se crea una carpeta llamada en este caso **prueba_despliegue_general** en la que se cargará el proyecto de prueba, a este dominio le vamos a asignar la ruta: **/prueba_despliegue_general** la cual se encuentra en el directorio raíz del servidor, específicamente en: **/home/begocomc**

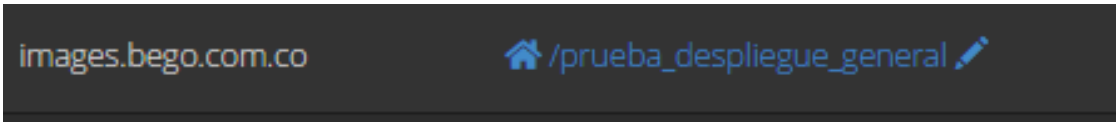


Ilustración 20. Subdominio creado para las pruebas. (Autor)

2. Se procede a iniciar sesión en la cuenta de cPanel, dirigirse al apartado archivos y agregar ingresar a la opción cuentas de FTP tal y como se aprecia en la **ilustración 21**

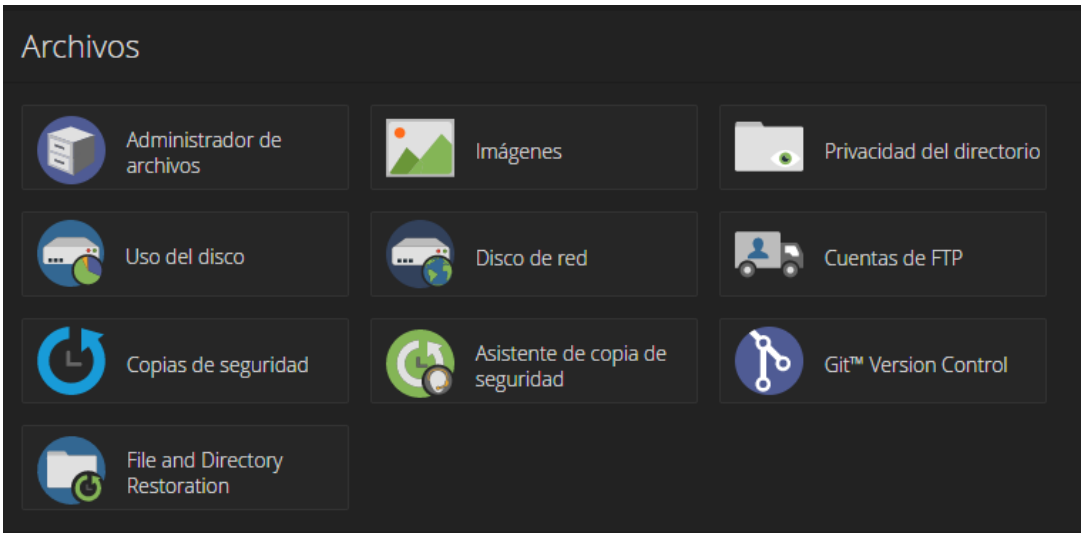


Ilustración 21. Menú Archivos en cPanel (Autor)

3. En este apartado se llena el formulario con la información correspondiente para la creación de la cuenta FTP que se va a utilizar.

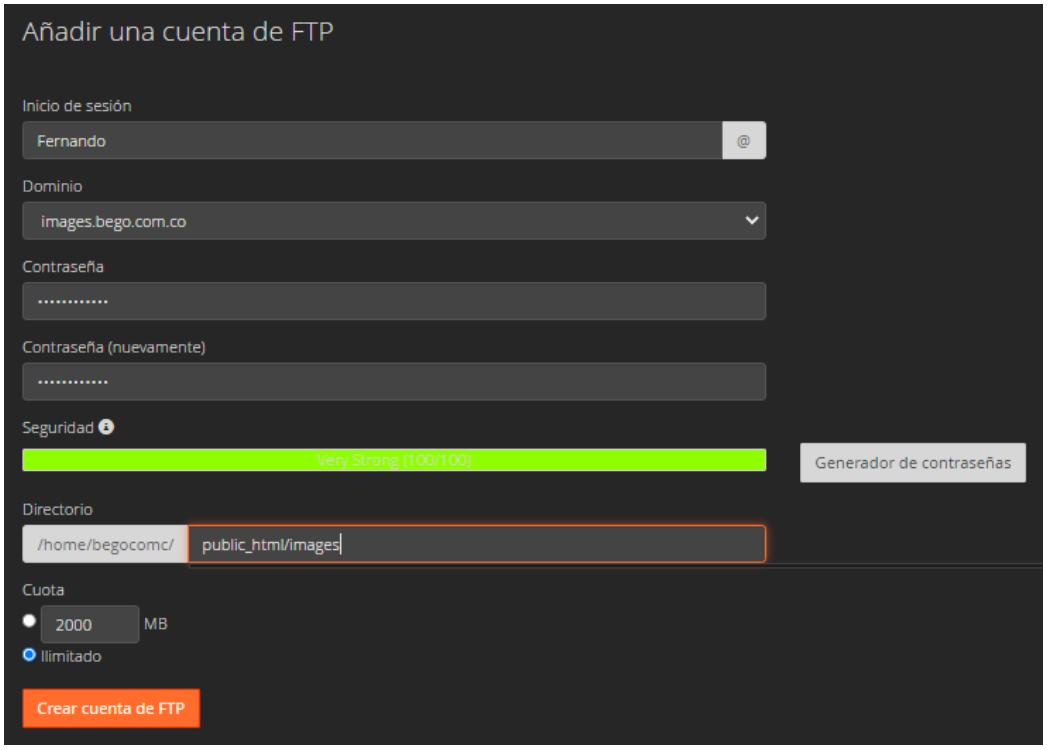


Ilustración 22. Creación de cuenta FTP en cPanel (Autor)

Como se aprecia en la **ilustración 22** es importante tener en cuenta el nombre de usuarios y la contraseña que se seleccionan, ya que estos datos serán usados en la conexión de nuestro GitHub Actions con el servidor cPanel; El dominio que se ve reflejado en la ilustración 19 es uno que se ha creado con anticipación tal y como se evidencia en el primer paso, de igual modo el directorio será el lugar donde quedarán almacenados los repositorios que se desplieguen continuamente.

4. En este paso se debe contar con la dirección IP del servidor, en este caso se puede evidenciar en la parte lateral derecha, si no se puede ver de manera correcta la dirección IP ó tiene algún problema lo que se recomienda es comunicarse con el proveedor de servicios de host. En la ilustración 23 se aprecia el aspecto de la barra lateral derecha que contiene la información general del servidor.

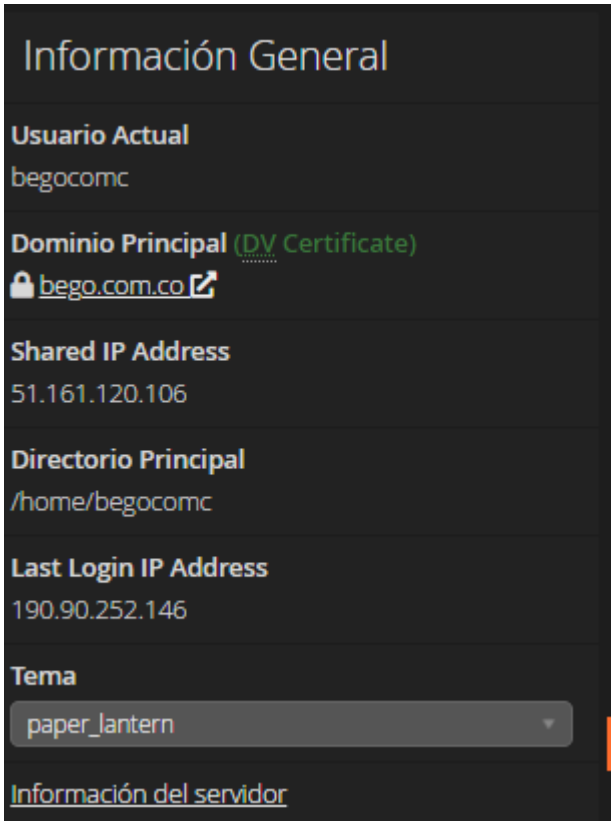


Ilustración 23. Información general del servidor (Autor)

5. A continuación, se procede a dirigirse al repositorio privado en GitHub, en este caso el repositorio tiene como nombre Team-BeGo y este está compuesto por cinco desarrolladores que se desempeñan en las instalaciones de la empresa.

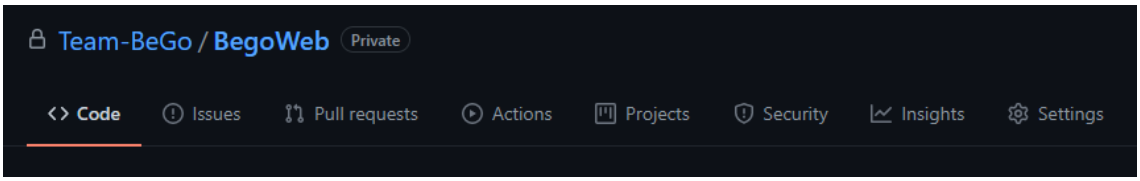


Ilustración 24. Repositorio de BeGo (Autor)

6. Seguido a esto hay que dirigirse a la configuración del repositorio e ingresar a la configuración de los secrets

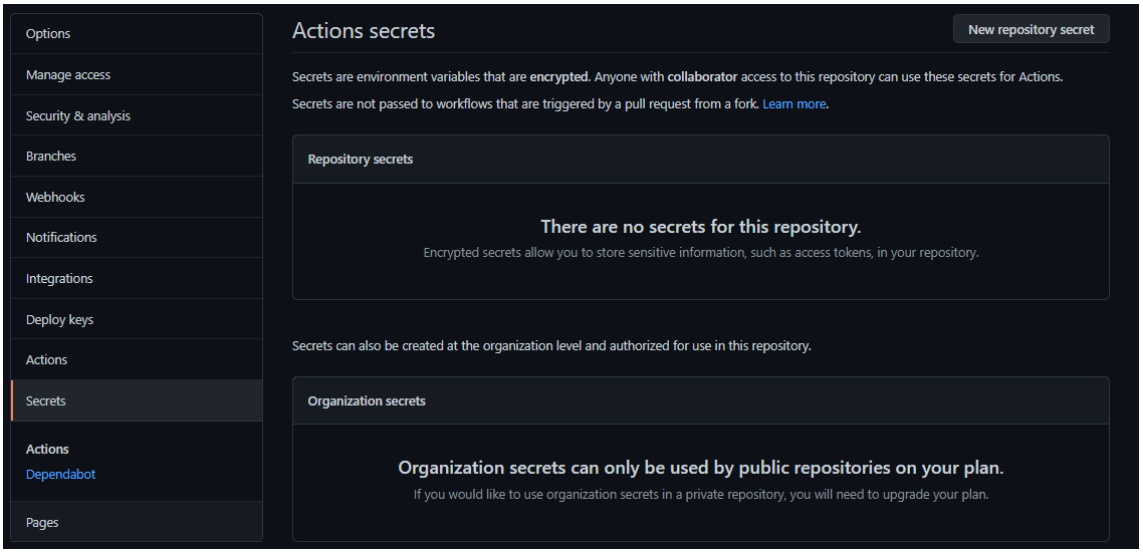


Ilustración 25. Configuración de Secrets (Autor)

7. En la pestaña de Action secrets se oprime el botón que dice **New repository secret** para agregar un nuevo Action secret

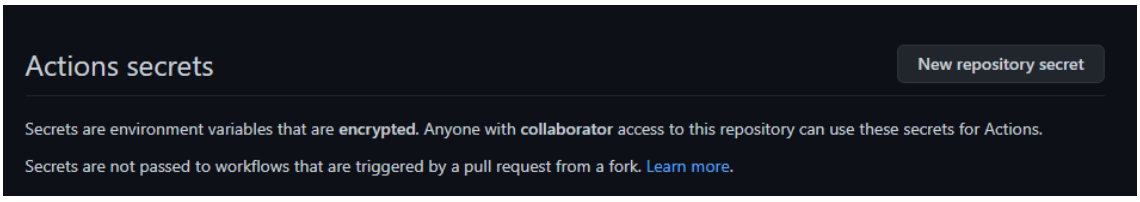


Ilustración 26. Creando nuevo Secret (Autor)

8. El paso a seguir es agregar los detalles que generamos en el paso 3, en este caso vamos a agregar un **FTP_SERVER** que me indicará el servidor, un **FTP_USUARIO** indicará el nombre del usuario y un **FTP_PASSWORD**, vale la pena aclarar que estos nombres pueden ser asignados de manera libre, tan solo se debe tener en cuenta que sí se realiza un cambio a los nombres de igual manera se deben hacer los cambios respectivos al archivo de GitHub Action que se mostrará en el paso 9. El ejemplo se aprecia en la **ilustración 27 y 28.**

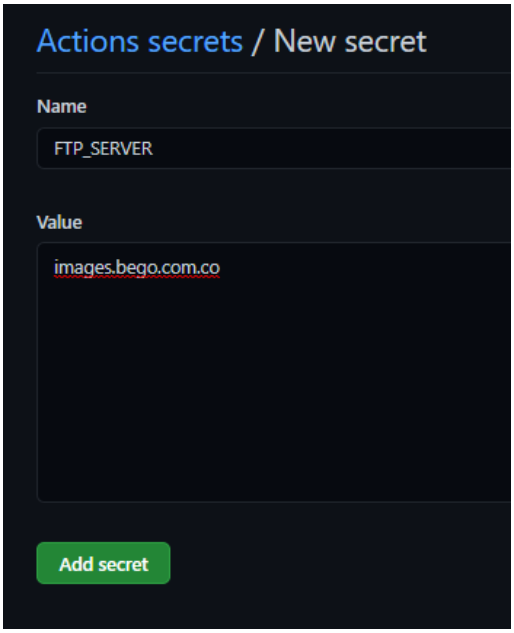


Ilustración 27. Agregando el servidor (Autor)

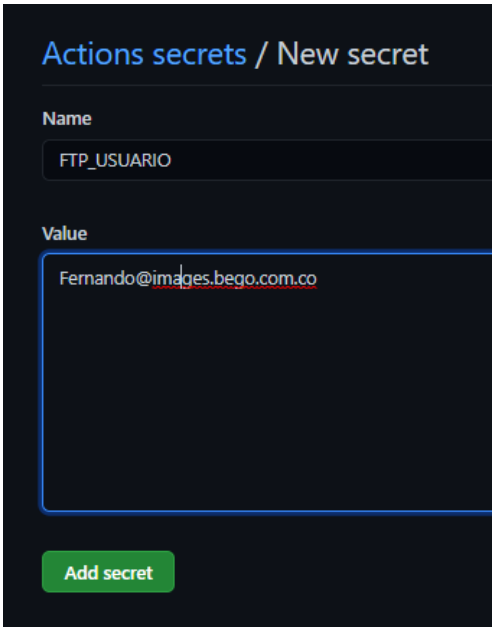


Ilustración 28. Agregando el usuario (Autor)

9. El siguiente paso es crear un flujo de trabajo de acción de GitHub. Las acciones de GitHub tienen lugar en el servidor de GitHub; en el repositorio de GitHub, se debe hacer clic en Actions y crear un nuevo flujo de trabajo tal y como se aprecia en la **ilustración 29.**

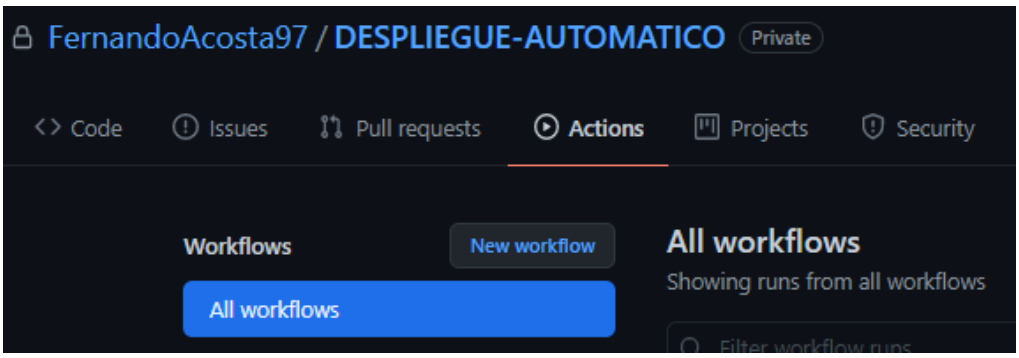


Ilustración 29. Creación de un Workflow (Autor)

10. Seguido a esto se debe hacer clic en New workflow y seleccionar cualquier plantilla, en este caso seleccionaremos la primera y eliminaremos todas las instrucciones que salen en ella, esto con el objetivo de crear reglas personalizadas.

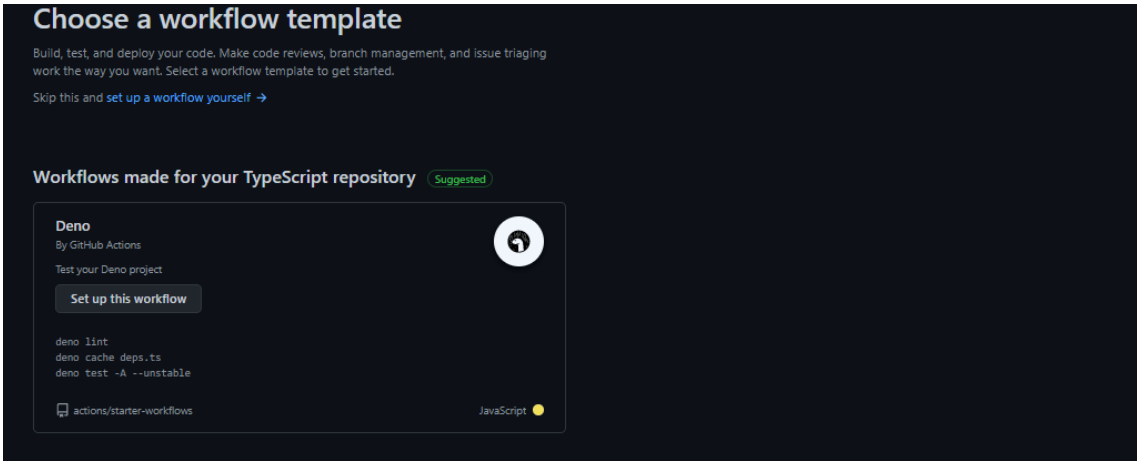


Ilustración 30. Plantilla para workflow (Autor)

11. Una vez seleccionada la plantilla se debe cambiar el nombre del archivo por el que se desee tal y como se evidencia en la ilustración 31, importante recordar que la extensión debe ser .yml

En este caso ese archivo se llamó DespliegueAutomatico_por_Fernando.yml

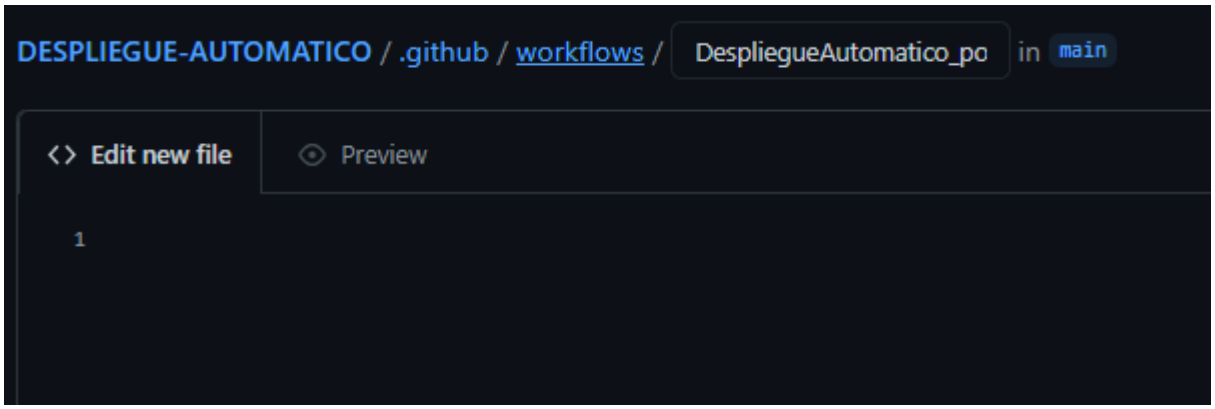


Ilustración 31. Nombre del workflow (Autor)

12. En el espacio que hemos dejado en la **ilustración 32** vamos a poner las respectivas instrucciones para la automatización del despliegue.

```
19 lines (19 sloc)  478 Bytes
1  name: DespliegueAutomatico_por_Fernando
2  on:
3    push:
4      branches:
5        - main
6  jobs:
7    FTP-Deploy-Action:
8      name: FTP-Deploy-Action
9      runs-on: ubuntu-latest
10     steps:
11     - uses: actions/checkout@v2.1.0
12       with:
13         fetch-depth: 2
14     - name: FTP-Deploy-Action
15       uses: SamKirkland/FTP-Deploy-Action@3.1.1
16       with:
17         ftp-server: ${{ secrets.FTP_SERVER }}
18         ftp-username: ${{ secrets.FTP_USUARIO }}
19         ftp-password: ${{ secrets.FTP_PASSWORD }}
```

Ilustración 32. Configuración del workflow (Autor)

Es importante Asegurarse de cambiar la rama desde la que se desea que se realicen los despliegues, en este caso para la empresa BeGo, estos se implementarán desde la rama maestra debido a que durante el proceso de integración se carga cada cambio que paso por cada una de las pruebas, lo que significa que en la rama maestra esta todo el código sin ningún error.

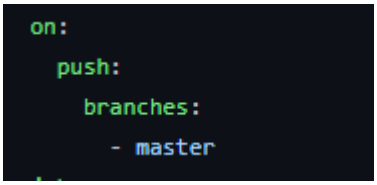


Ilustración 33. Rama principal (Autor)

13. Una vez que se hayan terminado estas configuraciones, se procede a dar clic en el botón Start Commit el cual se encuentra en el lado derecho

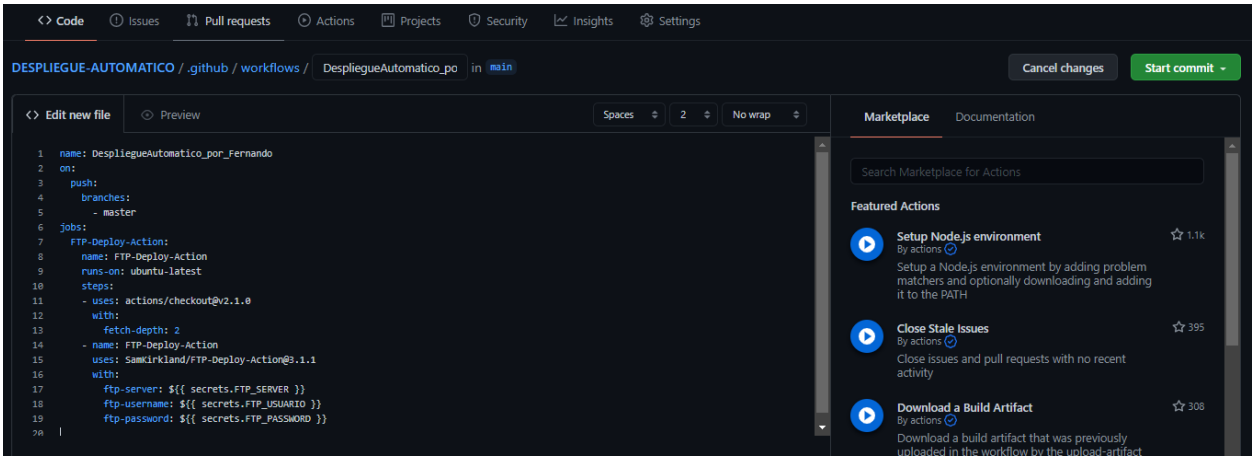


Ilustración 34. Start Commit (Autor)

Esto creará automáticamente una ruta /.github/workflows/ en la raíz del repositorio, dentro de esta ruta se evidenciara la previa configuración que se ejecutará de manera automática cada vez que se realice un cambio en la rama master como se aprecia en la **ilustración 35**.

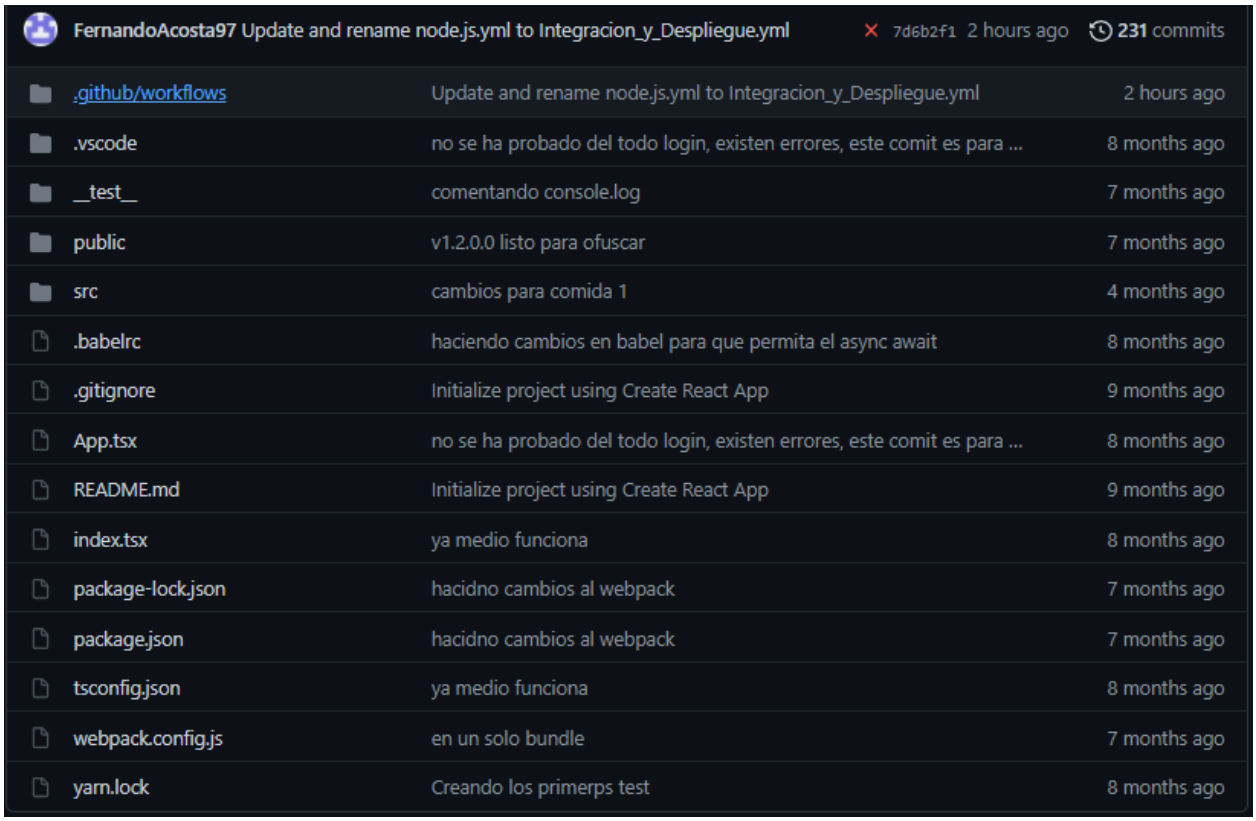


Ilustración 35. Carpeta creada después del workflow (Autor)

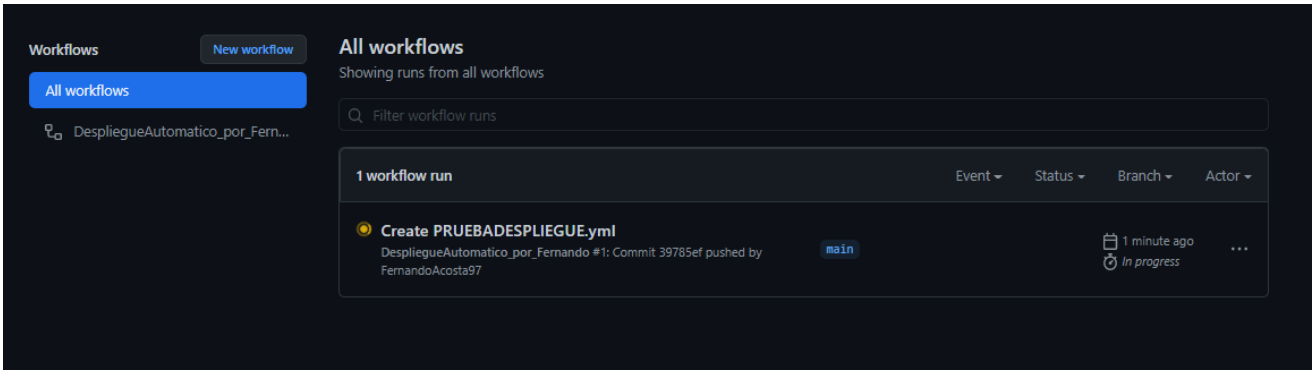


Ilustración 36. Ejecucion del Workflow (Autor)

En la ilustración 36 se evidenció la ejecución del action, cuando sale el color ambar (Naranja) quiere decir que el despliegue está en curso, en teoría se están leyendo las instrucciones que hemos configurado previamente, cuando sale en verde quiere decir que el despliegue se realizó de manera correcta. Ver en **ilustración 37.**

Ilustración 37. Despliegue correcto (Autor)

Sí por el contrario un despliegue nos falla este se tornará de color rojo tal como en la ilustración que veras a continuación

Ilustración 38. Despliegue Fallido (Autor)

CAPÍTULO 6

6 Validación

Dentro de los objetivos planteados al inicio del proyecto se encontraba el de la validación, para cumplir con dicho objetivo se ha destinado este capítulo, en el que se verá reflejado el procedimiento de despliegue para una nueva versión del aplicativo web, el cual esta soportado en programación híbrida mediante el uso de la librería React Native. Este se concluyó con la herramienta GitHub Actions en el servidor con el que la empresa de BeGo tenía contrato.

Se mostrarán los resultados y cada uno de los avances que se realizaron con el trabajo conjunto de las herramientas, para evidenciar los cambios y el aporte que se le dio a la empresa, se realizará una tabla donde se mostrará el antes y el después de la automatización del proceso de despliegue continuo en las instalaciones de BeGo.

6.1 Validación del proceso de Despliegue automático en el servidor de BeGo

Durante el capítulo 5 en el apartado 5.3 se pudo evidenciar el funcionamiento de la herramienta con la que se decidió implementar el despliegue automático en la empresa, allí se incluyeron cada uno de los pasos que se debieron seguir para llegar a generar un completo despliegue en el servidor cPanel, que como se explicaba en dicho capitulo era el servidor con el que la empresa tenía convenio para dicho momento.

Para validar que el despliegue se está haciendo de manera correcta y automática en el servidor de cPanel después de realizar las aceptaciones en los cambios de la rama maestra, se probará creando un nuevo repositorio e incluyendo un index.html sencillo el cual se redirigirá al subdominio que se había creado en el **capítulo 5** específicamente en la **ilustración 20,** desplegando

desde GitHub con GitHub Actions hacia el servidor de cPanel por medio de la conexión remota que se logró con el protocolo FTP y los Secrets configurados previamente.

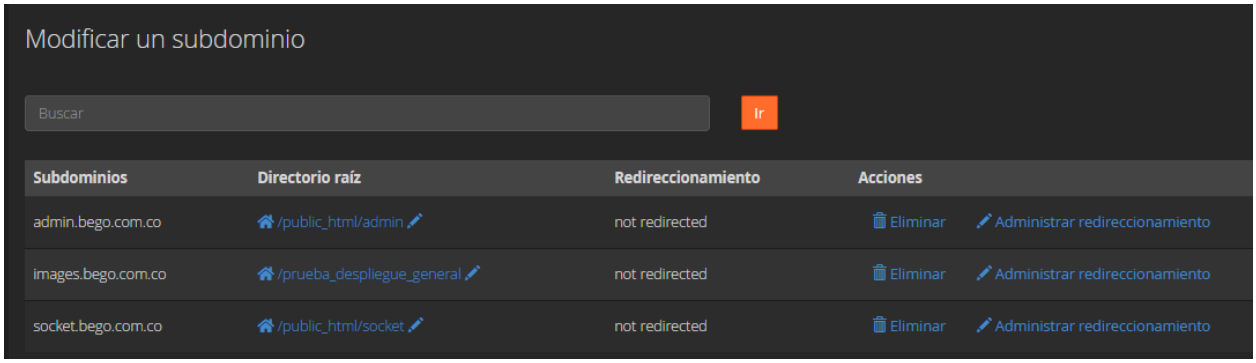


Ilustración 39. Subdominios creados en cPanel (Autor)

En la **ilustración 39** se puede apreciar el subdominio que se había creado en el capítulo 5, este subdominio es: **images.bego.com.co** y la ruta que tiene destinada para consulta en el servidor es: **/prueba_despliegue_general** la cual se encuentra en el directorio raíz del servidor, específicamente en: **/home/begocomc**

Al ver la cuenta FTP creada anteriormente se puede de igual manera evidenciar la ruta a la que se está dando el acceso, por ende, es lógico que el despliegue automático apunte a dicha carpeta.

Ilustración 40. Cuentas FTP (Autor)

se procede a validar ejecutando un nuevo cambio en el index.html sencillo que se destinó para realizar las pruebas de despliegue.

Ilustración 41. Índex de prueba (Autor)

Se guardarán los cambios de este index.html en la rama principal para validar así la respectiva prueba de despliegue en el servidor.

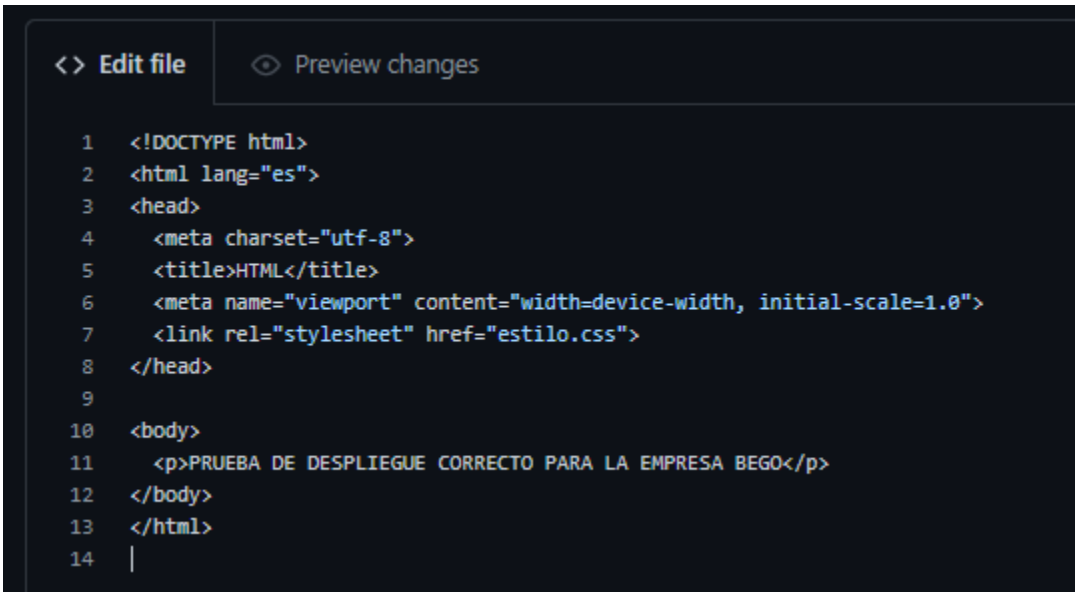


Ilustración 42. Contenido del índex de prueba (Autor)

Hacemos un Commit para guardar los cambios en el repositorio, en teoría después de hacer esto el Worflow configurado anteriormente se debe disparar.

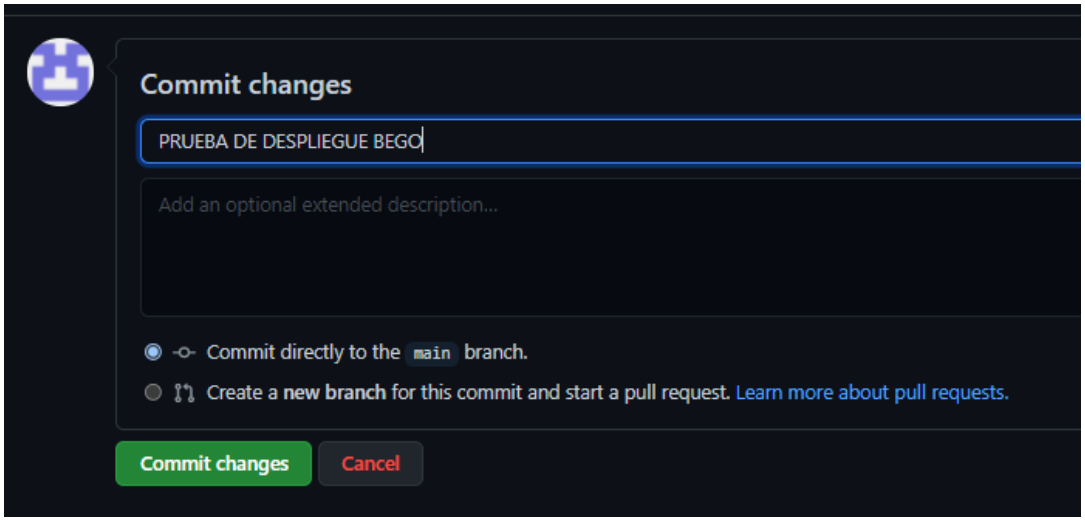


Ilustración 43. Commit changes (Autor)

La prueba de despliegue bego como la hemos llamado fue correcta como se puede apreciar en la **ilustración 44**

Ilustración 44. Prueba de despliegue Correcta (Autor)

A continuación, toca dirigirse al servidor cPanel, específicamente a la carpeta: **/prueba_despliegue_general** con el fin de observar que el despliegue se haya efectuado de manera correcta

Ilustración 45. Comprobando el despliegue en el servidor (Autor)

A continuación, se procede a consultar el subdominio creado para pruebas, en la ilustración se evidencia el índex corriendo en el navegador.

Esto quiere decir que el despliegue se ha realizado de manera automática y correcta, ahora cuando se realice cualquier cambio o se haga MERGE a la rama maestra todo se desplegará en el servidor cPanel de la empresa BeGo.

Ilustración 46. Consulta a subdominio (Autor)

Antes de gestionar estos cambios en la configuración de los repositorios privados de BeGo en GitHub se tenía el proceso de integración continúa automatizado de manera muy similar a la forma en la que se automatizó el despliegue, a continuación, se mostrará en la **ilustración 47.** En la cual se agregó esta configuración en un mismo archivo, evidenciando así primeramente la integración y seguido a esto el Deploy (Despliegue)el cual recibe como parámetro el build.

Ilustración 47. Configuración de integración y despliegue (Autor)

6.2 Aportes a la empresa

Tabla 11. Aportes de la automatización del despliegue

Concepto	Antes	Después
Despliegues Rutinarios	Se olvidaban hacer los despliegues en las fechas programadas ya que no eran automáticos.	El despliegue se genera de manera inmediata, apenas se realizan o aprueban los cambios en la rama maestra en los repositorios privados de BeGo
Seguridad	Se tenía que compartir la clave del servidor con algunos programadores para que realizarán el respectivo despliegue ya que todos estaban cargados de tareas.	Se logró una conexión directa entre el servidor y GitHub mediante el protocolo FTP, lo que evita que sea necesaria una clave, está ya se encuentra configurada en los actions secrets de GitHub Actions en el repositorio correspondiente
Profesionalismo	Antes el despliegue era muy artesanal, la forma en la	Ahora el despliegue estaba configurado de manera

	que se subían los repositorios ya probados e integrados al servidor se hacían por medio de un .zip para descomprimir en la carpeta correcta, quiere decir que siempre se llevaba de forma manual y poco profesional.	automática, de modo que después de cada cambio validado e integrado, este se desplegará en el servidor sin necesidad de estar comprimido, la conexión FTP nos ayuda por medio de un Workflow configurado con Actions.
Tiempo	Antes cada que se realizaba un cambio en la web de BeGO se tenía que estar haciendo el despliegue manual, en ocasiones volviendo a cargar todos los archivos pertenecientes al repositorio, esto consumía bastante tiempo que e podía utilizar para otras cosas en desarrollo.	En este momento se cargan los archivos de manera automática, tan solo con realizar un commit en la rama master, los archivos se sobrescriben, pero al ser automáticamente no hay esfuerzo humano.

Estrategia	Antes se subía archivo por archivo al servidor, esto debido a que el servidor no permitía cargar carpetas con varios archivos, esto quitaba bastante tiempo y disminuía la producción. Por esta misma razón se cargaban archivos comprimidos para extraer todo el proyecto en una carpeta del servidor.	Ahora los archivos se cargan completos, carpetas como deben ser y ficheros en su lugar, lo mejor es que no se debe hacer absolutamente nada, solo es esperar la confirmación en la acción del GitHub Actions.
Actualizaciones	Las versiones tardaban más en llegar ya que cada programador se enfocaba en las funciones asignadas, el despliegue se dejaba para cuando todo estuviera listo.	Plazos más cortos entre actualizaciones y nuevas versiones, ahora se pueden ir actualizando todo por funciones.

7 Conclusiones

- Se logró determinar que el despliegue continuo ayuda a la liberación de carga a los equipos de operaciones de procesos manuales, que son una de la principal causa de retrasos en la distribución de aplicaciones.
- Con el estudio de la automatización de procesos en las empresas de desarrollo de software y la puesta en marcha de estas técnicas se logra aportar una serie de beneficios como la entrega o liberación inmediata de código, una simplificación del trabajo colaborativo o de equipo y se permite llegar a detecciones tempranas de errores, ayudando de esta manera a la disminución de costos de desarrollo, a su vez el ahorro de tiempo y esfuerzo físico.
- Se diseñó un procedimiento estructurado de Despliegue continuo automático que permitió agilizar, disminuir costos de tiempo y personal en la empresa, y lo mejor es que este procedimiento se puede aplicar a cualquier otro proyecto web.
- Se complementó el proceso de integración continua, contribuyendo con un despliegue automático una vez se integra y prueba todo en GitHub Actions.
- Se determinó que las necesidades actuales a la hora de desarrollar y entregar software de calidad a los clientes de forma periódica, ha incitado a que metodologías como DevOps hayan sido adoptadas en la industria del desarrollo de software esto con el objetivo de automatizar procesos y aplicar técnicas como la integración continua, la entrega continua y el despliegue continuo.
- Los resultados serán un software de mayor calidad libre de errores que podrá ser entregado al cliente de forma más rápida y periódica.

8 Recomendaciones y trabajos futuros

- Se propone que se automatice el despliegue continuo de una App móvil implementando DevOps, ya que la empresa cuenta con un aplicativo móvil el cual se encuentra en producción
- Se recomienda que el proceso de automatización continúe con el despliegue IOS en la App Store y para Android en Play Store
- Se propone realizar una investigación para que el proceso de despliegue continuo de aplicaciones móviles se pueda ejecutar de manera local o en algún tipo de simulador, esto con el objetivo de reducir costos y tiempo.
- Se recomienda una investigación para la implementación de contenedores en el despliegue de aplicaciones de la infraestructura de desarrollo de BeGo.

9 Bibliografía

Ambit. (04 de 06 de 2020). *ambit-bst.com*. Obtenido de https://www.ambit-bst.com/blog/amazon-web-services-ventajas-desventajas

Barrera, A. (s.f.). *nextu.com*. Obtenido de https://www.nextu.com/blog/6-cosas-que-debes-saber-de-amazon-web-services-y-sus-beneficios/

Carrillo, H. J. (2020). *DISEÑO DE UN PROCEDIMIENTO DE PRUEBAS PARA LOS PROYECTOS DE DESARROLLO DE SOFTWARE DE LA EMPRESA BEGO.* Pamplona.

Chávez, G. (11 de 10 de 2018). *smartnodus.cl*. Obtenido de https://www.smartnodus.cl/integracion-continua/

Chuidiang. (02 de 11 de 2020). *http://blog.chuidiang.org/*. Obtenido de http://blog.chuidiang.org/2020/11/02/maven-gradle/

DEVOPS. (10 de 12 de 2020). *ilimit.com*. Obtenido de https://www.ilimit.com/blog/integracion-continua-entrega-continua-despliegue-continuo/

Diez, B. (12 de 02 de 2020). *LemonCode.net*. Obtenido de https://lemoncode.net/lemoncode-blog/2020/2/12/hola-docker-ci-cd-github-actions

Docker docs. (6 de 10 de 2020). Obtenido de Docker docs: https://docs.docker.com/get-started/overview/

Everac99. (23 de 06 de 2011). *everac99.wordpress.com*. Obtenido de https://everac99.wordpress.com/2011/06/23/despliegue-continuo-continuous-delivery/

Fraile, L. (11 de 03 de 2021). *plainconcepts.com*. Obtenido de https://www.plainconcepts.com/es/github-actions-introduccion/

GARCIA, P. A. (2019). *PRÁCTICAS DEVOPS DE ENTREGA CONTINUA DE SOFTWARE PARA LA.* Medellin .

García, R. (11 de 05 de 2019). *openexpoeurope.com.* Obtenido de https://openexpoeurope.com/es/open-source-stack-integracion-despliegue/

Garg, S. (02 de 2019). *www.researchgate.net*. Obtenido de https://www.researchgate.net/figure/CI-CD-Architecture-using-Docker_fig1_331131851

GIL, L. A. (09 de 11 de 2015). *code2read*. Obtenido de https://code2read.wordpress.com/2015/11/09/jenkins-instalacion-ci-integracion-continua/

GitHub. (s.f.). *GitHub Docs.* Obtenido de https://docs.github.com/es/actions/creating-actions/about-actions#further-reading

git-scm.com. (s.f.). *git.* Obtenido de https://git-scm.com/

Gutierrez, C. (21 de 10 de 2020). *csl.com.co.* Obtenido de https://csl.com.co/sonarqube-auditando-al-auditor-parte-i/

Henson. (2017). *blog.hensongroup.com*. Obtenido de https://blog.hensongroup.com/es/pros-and-cons-of-microsoft-azure-cloud-service-for-businesses/#:~:text=Contras%20de%20Microsoft%20Azure,-Necesitar%C3%A1%20experiencia%20en&text=Azure%20es%20diferente%20de%20los,una%20empresa%20puede%20crear%20problemas.

ILIMIT. (05 de 19 de 2020). *ilimit.com.* Obtenido de https://www.ilimit.com/blog/automatizacion-despliegue-continuo/

Isaac. (02 de 03 de 2021). *blog.desdelinux.net*. Obtenido de https://blog.desdelinux.net/github-vs-gitlab/

Isaac. (26 de 02 de 2021). *blog.desdelinux.net*. Obtenido de https://blog.desdelinux.net/docker-vs-kubernetes/#Ventajas

Kiprosh. (21 de 01 de 2020). *blog.kiprosh.com*. Obtenido de https://blog.kiprosh.com/automate-deployment-with-github-actions/

Lamos, B. (05 de 11 de 2019). *Microsoft Build*. Obtenido de https://docs.microsoft.com/es-es/hybrid/app-solutions/pattern-cicd-pipeline

Laurenti, B. (01 de 08 de 2018). *slideshare.net*. Obtenido de https://www.slideshare.net/AmazonWebServicesLATAM/tcnicas-para-implementacin-de-continuous-delivery-en-aws

Linkedin. (s.f.). *es.linkedin.com*. Obtenido de puede integrarse con visualstudio.net para que los desarrolladores puedan rastrear sus tareas asignadas, así como la asignación nueva de elementos.

Márquez, C. (07 de 05 de 2020). *Styde.net*. Obtenido de https://styde.net/construyendo-un-flujo-ci-cd-para-laravel-con-github-actions/#:~:text=GitHub%20Actions%20permite%20crear%20flujos,contener%20al%20menos%20un%20job.

MARTÍN, Á. (s.f.). *_CODETAKERS*. Obtenido de https://codetakers.team/blog/despliegue-continuo-gitlab

Montoya, S., & Ocampo , M. (12 de 05 de 2020). *pragma.com.co*. Obtenido de https://www.pragma.com.co/blog/conoce-5-herramientas-para-integracion-y-entrega-continua-con-devops

Omoregbee, T. (22 de 06 de 2018). *codeburst.io*. Obtenido de https://codeburst.io/gitlab-build-and-push-to-a-server-via-ssh-6d27ca1bf7b4

Oquendo, F. (03 de 02 de 2021). *castor.com.co.* Obtenido de https://castor.com.co/automatizacion-del-pipeline-de-despliegue/

Orias, M., & Marfil, A. J. (2018). *Despliegue ágil en proyectos de gran escala.* La Plata .

Palma, I. (12 de 04 de 2021). *freecodecamp.org*. Obtenido de https://www.freecodecamp.org/espanol/news/como-automatizar-el-despliegue-en-github-pages-con-travis-cl/#:~:text=Travis%20CI%20es%20un%20servicio,de%20software%20alojados%20en%20GitHub.&text=yml%20%2C%20Travis%20CI%20ejecutar%C3%A1%20los,o%20ejecutar%20scr

Pathak, A. (25 de 02 de 2021). *geekflare.com*. Obtenido de https://geekflare.com/es/gitlab-hosting/

Pérez, S. (10 de 2017). Obtenido de https://apiumhub.com/es/tech-blog-barcelona/ventajas-de-jenkins/

Pérez, S. (23 de 10 de 2017). *apiumhub.com*. Obtenido de https://apiumhub.com/es/tech-blog-barcelona/ventajas-de-jenkins/

project, T. J. (13 de 01 de 2012). *Wikipedia.* Obtenido de https://es.m.wikipedia.org/wiki/Archivo:Jenkins_logo_with_title.svg

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

Rodríguez, T. (09 de 02 de 2012). *genbeta.com.* Obtenido de https://www.genbeta.com/desarrollo/travis-ci-sistema-distribuido-de-integracion-continua-libre-integrado-con-github

Royo, D. (24 de 11 de 2016). *hiberus.com.* Obtenido de https://www.hiberus.com/crecemos-contigo/jenkins-como-automatizar-tareas-despliegues-testing/

Sánchez, L. A. (2017). *ARQUITECTURA TECNOLÓGICA PARA LA ENTREGA.* Cuenca.

Seta, L. d. (22 de 01 de 2009). *dosideas.com.* Obtenido de https://dosideas.com/noticias/java/391-automatizacion-de-despliegues-ino-mas-dolores-de-cabeza

Turrado, J. (10 de 03 de 2020). *campusmvp.es.* Obtenido de https://www.campusmvp.es/recursos/post/que-son-las-pruebas-de-software.aspx#:~:text=Las%20pruebas%20de%20software%20son,hasta%20su%20puesta%20en%20producci%C3%B3n.

Valencia, J. E. (2017). *CONFIGURACIÓN Y DESPLIEGUE AUTOMÁTICO DE.* Bogotá.

Vargas , C., & Marcos, M. (17 de 07 de 2019). *DDigital - UMSS.* Obtenido de http://ddigital.umss.edu.bo:8080/jspui/handle/123456789/14967

VILLAMIZAR, A. L. (2020). *DISEÑO DE PROCESO DE INTEGRACIÓN PARA LOS PROYECTOS DE.* Pamplona.

ANEXO 1

Pamplona, 08 de Junio del 2021

Cumplimiento de objetivos

Señores
Universidad de Pamplona
Programa de Ingeniería de Sistemas

Por medio de la presente **OSCAR FERNEY ARDILA CONTRERAS**, Responsable de la Práctica Profesional del estudiante **LUIS FERNANDO ARCILA ACOSTA**, hace constar el cumplimiento de los objetivos planteados al inicio de la misma.

LUIS FERNANDO ARCILA ACOSTA estudiante de la Universidad de Pamplona, con numero de identificación **1015467593** perteneciente al programa de **INGENIERIA DE SISTEMAS**, ha cumplido satisfactoriamente con las actividades asignadas para la prestación de la Práctica Profesional en el proyecto **CREACIÓN DE UN PROCEDIMIENTO DE DESPLIEGUE AUTOMÁTICO DE APLICACIONES SOBRE LA INFRAESTRUCTURA TECNOLÓGICA DEL ÁREA DE DESARROLLO DE LA EMPRESA BEGO.**

La presente se extiende a petición del interesado para los fines legales que a él convengan, en la ciudad de PAMPLONA NORTE DE SANTANDER siendo el día 08-06-2021.

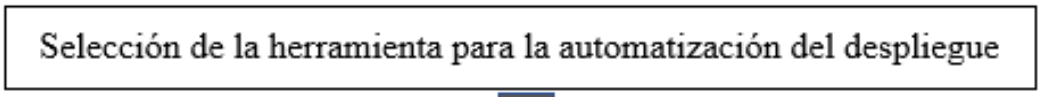
A t e n t a m e n t e

Oscar Ardila

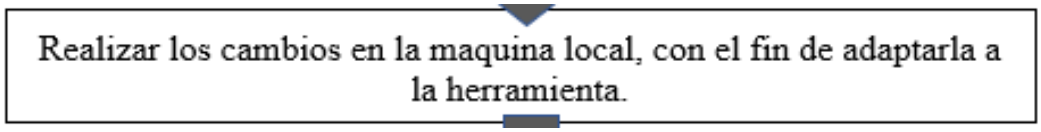
Oscar Ferney Ardila Contreras

ANEXO 2

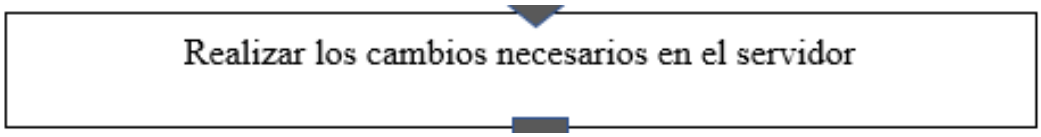
(Pasos a seguir para conseguir un despliegue automático)



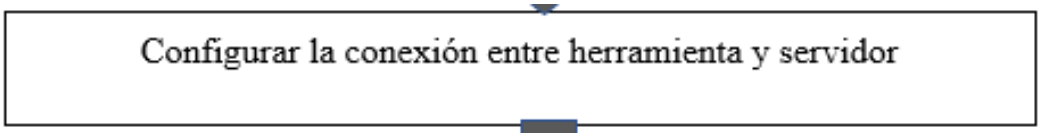
En este paso se debe tener en cuenta las ventajas de cada una de las herramientas, teniendo presente de igual manera la magnitud del sistema a desplegar.



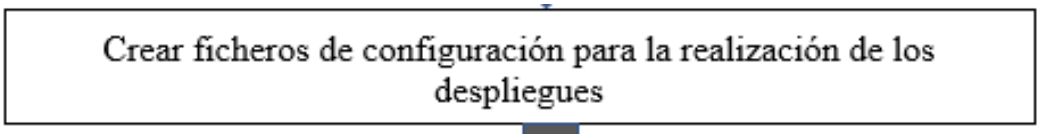
Este paso dependerá de la herramienta que se esté utilizando para el versionamiento del código, puede ser la configuración con Git u otro.



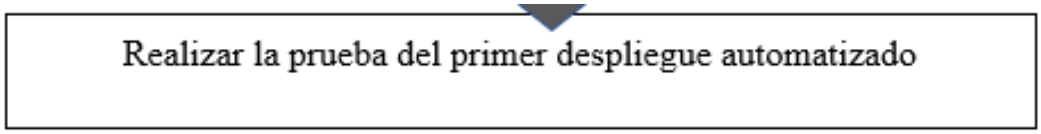
En este paso se debe configurar un subdominio o dominio principal el cual apuntará al directorio donde se cargará el compilado generado.



Se tiene en cuenta la seguridad que implementa el servidor y la herramienta que se ha escogido para la automatización del despliegue, en este paso se busca la implementación de protocolos como FTP o SSH con el fin de conectar el servidor con la herramienta y automatizar los despliegues.



Como es de esperarse en este paso del proceso se generan los scrips necesarios para la automatización de la tarea de despliegue, estos scrips y su lenguaje dependerán de la herramienta que se seleccione previamente.



Finalmente se ejecuta lo anteriormente configurado para la verificación del correcto despliegue en el servidor, si todo funciona de manera adecuada al consultar el dominio debería cargar el sitio que se ha desplegado en el directorio de nuestro servidor.