Reporte de Laboratorio

Control de versiones y despliegue continuo (CI/CD)

06/04/2025

Departamento de Electrónica, Sistemas e Informática (DESI)

Arquitectura en la Nube (Cloud Architecture)



Mtro. Rodolfo Luthe Ríos

Mauricio Federico Casillas

759945

Maestría en sistemas computacionales

# *Palabras clave: computo en la nube, CI/CD, validación y pruebas, DevOps*

# Introducción

La automatización del despliegue de aplicaciones es esencial para optimizar la eficiencia y acelerar el tiempo de comercialización. Se explorará cómo la automatización mediante CI/CD permite a los equipos de desarrollo implementar cambios de código de manera rápida y segura, garantizando despliegues consistentes en diversos entornos. Este enfoque no solo reduce el riesgo de errores humanos, sino que también facilita la integración continua de nuevas características y correcciones, mejorando la calidad del software y la satisfacción del cliente. Además, el uso de herramientas de control de versiones como Git es crucial para la colaboración en equipo y la gestión eficiente del código fuente

# Marco teórico

La integración continua es un proceso en el que los desarrolladores integran su código en un repositorio central varias veces al día. Cada integración es verificada por un proceso automatizado de construcción y pruebas, lo que permite detectar y corregir errores de manera temprana. Esto ayuda a los equipos de desarrollo a evitar problemas de integración significativos y a mantener un flujo constante de mejoras en el código [1]. Alde Alanda et al. en [1] destacan que CI es fundamental para automatizar el proceso de desarrollo de software, asegurando que cada commit de los desarrolladores sea detectado y probado automáticamente. Despliegue continuo por otro lado es una extensión de CI que automatiza la entrega del software a los entornos de producción de manera segura y rápida. CD asegura que el software esté siempre en un estado listo para ser desplegado, lo que permite a las organizaciones entregar nuevas características y correcciones de errores a los usuarios de manera continua. Abhishek Goyal en [2] explica que CD mejora la productividad de los desarrolladores al automatizar procesos, mejorar la calidad del código y permitir la entrega rápida de actualizaciones a los clientes.

La validación y las pruebas son componentes críticos en el ciclo de vida del desarrollo de software, garantizando que los cambios en el código no introduzcan errores y que el software cumpla con los requisitos de calidad antes de su despliegue. La automatización de pruebas dentro de los pipelines de CI/CD permite a los equipos de desarrollo validar continuamente el software, reduciendo el tiempo de comercialización y mejorando la satisfacción del cliente. DevOps juega un papel crucial al integrar las prácticas de desarrollo de software (Dev) con las operaciones de TI (Ops) para mejorar la colaboración y la productividad. DevOps fomenta una cultura de colaboración y responsabilidad compartida entre los equipos de desarrollo y operaciones, lo que es esencial para la implementación efectiva de CI/CD.

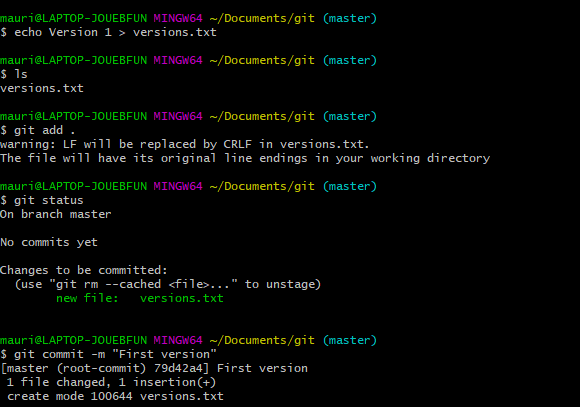
# Diagrama arquitectural

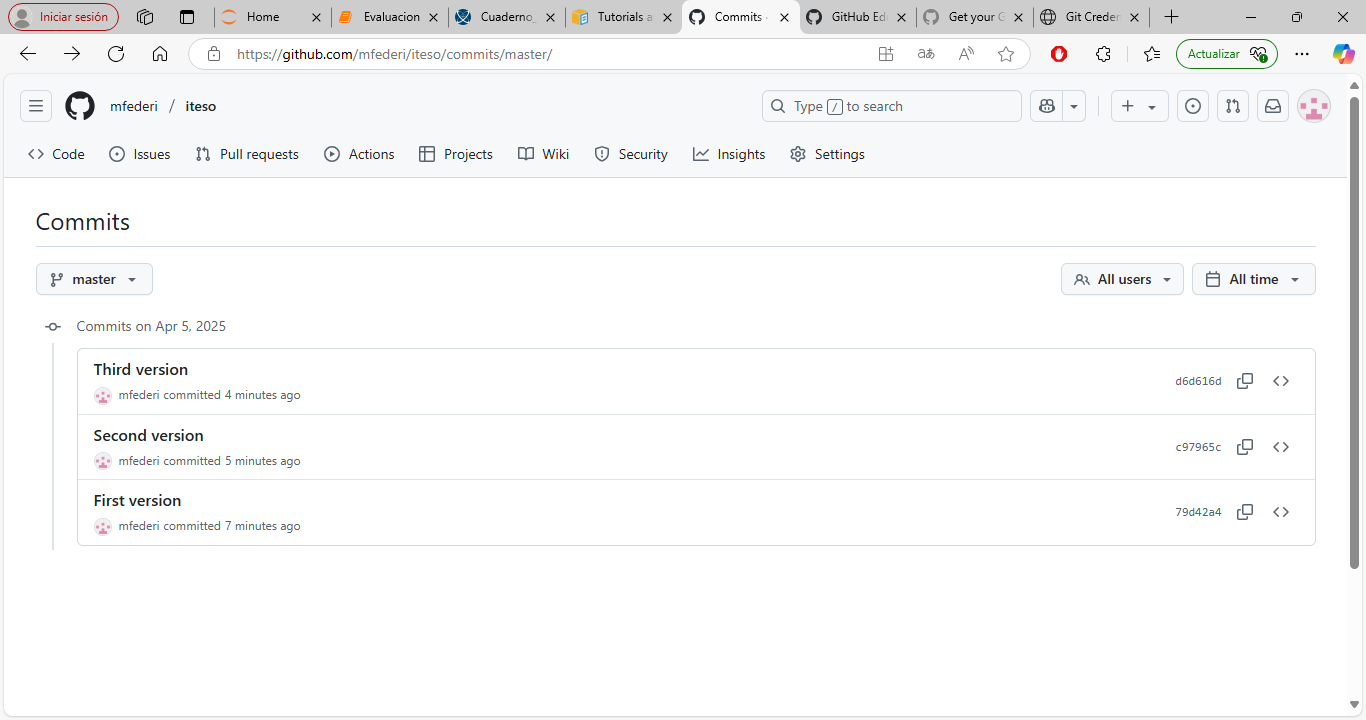
A screenshot of a computer

AI-generated content may be incorrect.

# Desarrollo de la práctica

Se inicio creando un repositorio local, en este se creó un archivo “versions.txt” en el cual se modifico 3 veces haciendo un commit por cada modificación.

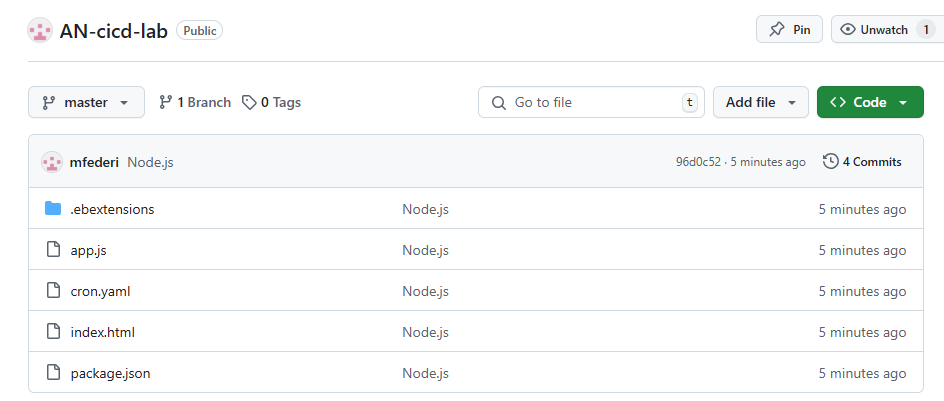


Desde una cuenta de GitHub se creó un repositorio remoto desde la página web y asignamos nuestro repositorio local al repositorio remoto que creamos y cargamos los cambios al repositorio remoto.

Para el ejercicio de despliegue automático nos conectamos a una consola de AWS. Se empezó creando una aplicación de tipo Node.js utilizando el servicio de ElasticBeanstalk de instancia única. Se accesó al URL de nuestra aplicación para asegurar que el despliegue de la aplicación se haya hecho de manera correcta.

A screenshot of a computer

AI-generated content may be incorrect.

Después se creó un repositorio en GitHub con la misma aplicación de muestra de Node.js.

Una vez que nuestra aplicación fue subida a GitHub se configuró el servicio de CodePipeline de AWS el despliegue de nuestra aplicación. Para ello se tuvo que hacer la autentificación de nuestra cuenta de GitHub para asignar nuestro repositorio de GitHub como la fuente del código de la aplicación a desplegar.

A screenshot of a computer

AI-generated content may be incorrect.

Para simular un cambio a nuestra aplicación se cambio el fondo de color verde a azul de nuestra aplicación Node.js

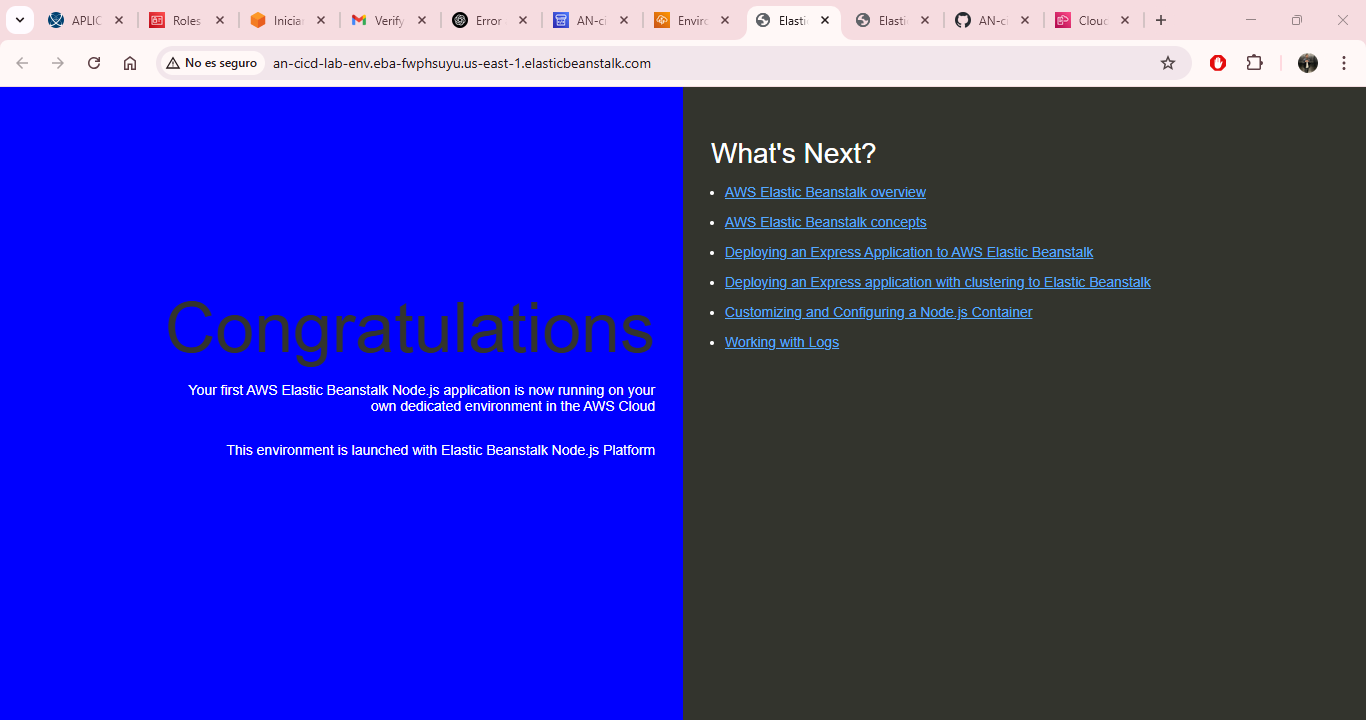
A computer screen shot of text

AI-generated content may be incorrect.

Cargamos los cambios al repositorio remoto y se observó que nuestro pipeline detectó el cambio desplegando automáticamente la nueva aplicación.

A screenshot of a computer

AI-generated content may be incorrect.



# Problemas y soluciones

Durante la configuración del servicio ElasticBeanstalk fue necesario crear roles que tuviesen los permisos para utilizar los servicios cosa que en nuestra consola educativa tenemos roles predefinidos, esto llevo algo de investigación.

# Experimentos y resultados

Se logro desplegar una aplicación web utilizando el servicio de AWS ElasticBeanstalk de manera sencilla demostrando que estos servicios son muy beneficiosos para los desarrolladores facilitándoles la automatización de pruebas y despliegue de estas así pudiéndose enfocar más en la funcionalidad del software.

# Análisis de costo

Amazon EC2 Linux t3.small On-Demand instances cost (Monthly): 5.06 USD

2 CodePipeline V1 cost (Monthly): 1.00 USD

***Total mensual 6.06 USD***