ACTIVIDAD 1 SEMANA 3 – INTEGRACIÓN CONTINUA

Estudiantes

Martha Yaneth Mera Velasco

Erick Alexander Salamanca Borda

Jesús David Mendivelso Lizarazo

Tutor

Natalia Martínez

POLITÉCNICO GRANCOLOMBIANO

FACULTAD INGENIERÍA - INGENIERÍA DE SOFTWARE

BOGOTÁ D.C

NOVIEMBRE 2024

TABLA DE CONTENIDO

# INTRODUCCION

En el desarrollo de software actual, la agilidad y la colaboración son claves para mantener la calidad y la velocidad en cada proyecto. La integración continua es una práctica que nos permite automatizar la entrega de código, permitiendo que las actualizaciones sean constantes, seguras y eficientes. Dos herramientas esenciales en este proceso son los controladores de versiones y los contenedores, que facilitan el control y la escalabilidad del desarrollo.

GitHub es uno de los sistemas de control de versiones más utilizados en la actualidad. A través de su sistema de almacenamiento y manejo de versiones, permite a los equipos trabajar de forma simultánea, mantener un historial detallado de cambios y colaborar sin conflictos en cada paso del desarrollo.

Por otro lado, los contenedores, con herramientas como Docker, permiten encapsular aplicaciones en entornos controlados y replicables. Gracias a esto, es posible asegurar que el software funcione de manera consistente en cualquier entorno, desde el desarrollo hasta la producción.

En este documento, se explicará la forma de utilizar las herramientas de GitHub y el uso de los contenedores los cuales contribuyen a una integración continua eficiente, promoviendo un flujo de trabajo optimizado y minimizando errores a lo largo del ciclo de vida del software.

# DESARROLLO ACTIVIDAD 1

## Contextualización

## **Github**.

GitHub es una plataforma de desarrollo colaborativo basada en la nube que permite a los desarrolladores gestionar, almacenar y colaborar en proyectos de software utilizando Git, un sistema de control de versiones. GitHub facilita el trabajo en equipo en proyectos de código al permitir que los miembros contribuyan, revisen y combinen cambios de forma organizada, llevando un registro detallado de cada modificación.

Algunas de sus principales funciones incluyen:

* **Control de versiones**: GitHub permite a los desarrolladores crear y gestionar versiones de su código, lo que facilita el seguimiento de cambios, la reversión a versiones anteriores y la colaboración sin conflictos.
* **Repositorio centralizado**: Cada proyecto tiene un "repositorio" en el que se almacena el código. Desde este repositorio, los desarrolladores pueden clonar (copiar) el código a sus propias máquinas, trabajar en él y luego subir los cambios.
* **Colaboración**: GitHub facilita la colaboración mediante herramientas como *pull requests*, que permiten a los colaboradores sugerir cambios que pueden ser revisados y aceptados por los administradores del proyecto. También cuenta con comentarios y revisiones de código, que fomentan el trabajo en equipo.
* **Integraciones y automatización**: GitHub se integra con otras herramientas, como sistemas de integración continua y plataformas de despliegue, que permiten automatizar pruebas y lanzamientos de software.

Además, GitHub ofrece funcionalidades para gestionar proyectos, como issues (para reportar y seguir problemas o mejoras) y milestones (para organizar los desarrollos en etapas), facilitando tanto el desarrollo como el mantenimiento de proyectos de software.

## **Contenedores**.

Los contenedores son una tecnología que permite empaquetar una aplicación junto con todas sus dependencias y configuraciones necesarias en un entorno aislado y portátil. Esto significa que una aplicación en un contenedor puede ejecutarse de manera consistente en diferentes entornos, ya sea en la computadora del desarrollador, en servidores de pruebas o en producción.

**Características de los contenedores:**

|  |  |
| --- | --- |
| Aislamiento | Los contenedores crean un entorno separado para cada aplicación, evitando conflictos entre dependencias o configuraciones de distintas aplicaciones en un mismo sistema |
| Portabilidad | Como el contenedor incluye todas las dependencias y configuraciones necesarias, puede ejecutarse en cualquier sistema compatible, lo cual es ideal para mover aplicaciones entre entornos locales, servidores o la nube |
| Eficiencia | A diferencia de las máquinas virtuales, que emulan sistemas completos con su propio sistema operativo, los contenedores comparten el núcleo del sistema operativo del anfitrión, por lo que consumen menos recursos y son más rápidos de iniciar y detener |
| Escalabilidad | Es fácil crear múltiples instancias de un contenedor para adaptarse a la demanda. Esto hace que los contenedores sean ideales para aplicaciones que necesitan escalar rápidamente en entornos de producción |

**Docker** es una de las herramientas más populares para trabajar con contenedores. Con Docker, los desarrolladores pueden definir todo lo que necesita una aplicación en un archivo llamado Dockerfile, que luego se utiliza para crear una "imagen". Esta imagen se convierte en un contenedor al ejecutarse, conteniendo la aplicación con todas sus dependencias.

### Beneficios en el desarrollo y despliegue de software

Los contenedores son especialmente útiles en la integración y entrega continua (CI/CD), ya que permiten ejecutar pruebas automáticas en un entorno estable y desplegar aplicaciones en producción de forma rápida y confiable. Esto asegura que el software funcione de manera consistente en cualquier entorno y facilita la colaboración y escalabilidad en los equipos de desarrollo.

## Creación de repositorio en GitHub

## Creación de contenedores

## Comunicación entre los contenedores

# CONCLUSIONES

# BIBLIOGRAFIA

---

Esta introducción presenta los temas de manera general, destacando la importancia de ambas herramientas en el flujo de trabajo de integración continua.