Asignatura:

Programación III

Participante:

Ariadna Massiel Reynoso Pérez

Matrícula:

20220330

Maestro:

Kelyn Tejada

Fecha:

25 de noviembre del 2023

1-Que es Git?

Git es un sistema de control de versiones distribuido diseñado para rastrear cambios en archivos a lo largo del tiempo. Su principal función es mantener un registro completo y detallado de todas las modificaciones realizadas en un proyecto, lo que permite a los desarrolladores trabajar en colaboración de manera eficiente. Git almacena metadatos sobre archivos, incluyendo quién hizo qué cambio, cuándo y por qué. Además, permite ramificar el trabajo, fusionar ramas y colaborar en proyectos de manera distribuida, lo que lo convierte en una herramienta esencial para el desarrollo de software.

Dicho de forma simplificada git es un un control de versions distribuido, siendo una herramienta muy práctica para nuestro proyecto de software.

2-Para que funciona el comando Git init?

El comando git init se utiliza para inicializar un repositorio Git en un directorio específico. Cuando ejecutas este comando en un directorio vacío o existente, Git crea un nuevo repositorio local. Este repositorio contendrá toda la información necesaria para realizar un seguimiento de los cambios en los archivos del proyecto. Después de ejecutar git init, Git comenzará a rastrear los cambios en los archivos de ese directorio.

3-Que es una rama?

En Git, una rama es una línea independiente de desarrollo. Las ramas permiten a los desarrolladores trabajar en diferentes características o arreglos por separado sin afectar directamente el código en la rama principal (generalmente llamada main o master). Cada rama puede contener sus propios cambios y desarrollos, y posteriormente se pueden fusionar con otras ramas para combinar el trabajo realizado en diferentes líneas de desarrollo.

4-Como saber es que rama estoy?

Puedes utilizar el comando git branch con la opción -a para listar todas las ramas, y la rama actual tendrá un asterisco (*) junto a ella. Esto indica en qué rama te encuentras trabajando actualmente.

5-Quien creo git?

Git fue creado por Linus Torvalds en 2005. Torvalds es conocido por ser el creador y principal desarrollador del kernel de Linux. Git fue concebido originalmente para el desarrollo del kernel de Linux, pero su flexibilidad y eficiencia lo han convertido en una herramienta de control de versiones ampliamente utilizada en muchos otros proyectos.

Linus Torvalds buscaba un sistema distribuido que pudiera usar en forma semejante a BitKeeper, pero ninguno de los sistemas bajo software libre disponibles cumplía con sus requerimientos, especialmente en cuanto a desempeño. El diseño de Git mantiene una enorme cantidad de código distribuida y gestionada por mucha gente, que incide en numerosos detalles de rendimiento, y de la necesidad de rapidez en una primera implementación.

6-Cuales son los comandos más esenciales de Git?

- git init: para inicializar un repositorio.
- git clone: para clonar un repositorio existente.

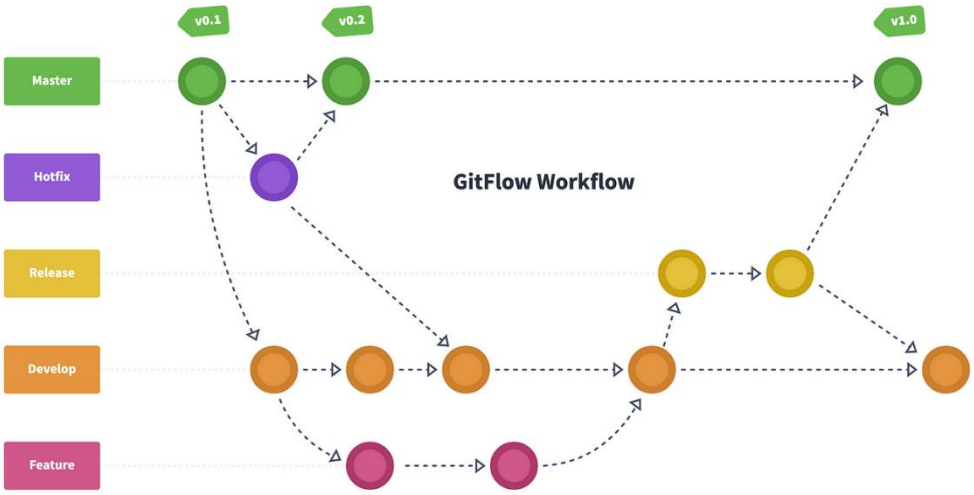
- git add: para agregar cambios al área de preparación.
- git commit: para guardar los cambios en el repositorio.
- git push: para enviar cambios locales a un repositorio remoto.
- git pull: para obtener cambios del repositorio remoto y fusionarlos con tu rama local.
- git checkout/swicht para cambiar entre ramas o versiones específicas de archivos.
- git merge: para fusionar cambios de una rama a otra.
- git branch: muestra las ramas disponibles y crea nuevas ramas.
- git status: muestra el estado actual del repositorio y los cambios pendientes.

7-Que es git Flow?

Git Flow es un modelo de ramificación y flujo de trabajo para proyectos que utilizan Git. Propone un conjunto de reglas y prácticas sobre cómo estructurar las ramas y gestionar el flujo de trabajo en un proyecto Git. Define ramas específicas para nuevas características, lanzamientos, corrección de errores y mantenimiento, con reglas claras sobre cómo fusionar y gestionar estas ramas.

El Git Flow se basa en la idea de tener diferentes tipos de ramas para administrar el desarrollo, las correcciones de errores y las versiones del software. Las ramas principales en este modelo son:

- Branch Master: Contiene el código estable y desplegable en producción.
- Branch Develop: Rama donde se integran todas las características. Representa el estado más reciente del desarrollo.
- Feature Branches: Ramas utilizadas para desarrollar nuevas características. Se crean a partir de la rama develop y se fusionan nuevamente en develop cuando se completa el trabajo.
- Release Branches: Ramas preparadas para la próxima versión del software. Se crean a partir de develop y sirven para realizar pruebas finales y ajustes antes de la publicación. Después de las pruebas, se fusionan tanto en master como en develop.
- Hotfix Branches: Ramas creadas para corregir problemas críticos en producción. Se originan desde master, solucionan el problema y luego se fusionan tanto en master como en develop.



8-Que es trunk based development?

Trunk Based Development es una metodología de desarrollo de software que se centra en mantener una única rama principal (trunk) como la línea de desarrollo principal del proyecto. En este enfoque, los desarrolladores trabajan directamente en la rama principal, evitando ramificaciones prolongadas y fusionando cambios pequeños y frecuentes. Esto promueve la integración continua y la entrega rápida de cambios al producto final, fomentando la colaboración constante entre los miembros del equipo de desarrollo.

En contraste con otros modelos de ramificación más complejos, como Git Flow, el Trunk Based Development promueve la idea de tener una única rama principal y utilizar estrategias como "feature toggles" o "feature flags" para desarrollar nuevas funcionalidades sin crear ramas separadas. Algunos puntos clave del Trunk Based Development incluyen:

- **Uso de ramas cortas:** En lugar de ramas largas y persistentes, se utilizan ramas cortas y temporales para el desarrollo de características específicas. Estas ramas se fusionan rápidamente de regreso al tronco principal una vez que la funcionalidad está completa y probada.
- **Integración continua:** Se enfoca en integrar cambios en el tronco de manera frecuente y continua. Los equipos buscan fusionar cambios pequeños y frecuentes en lugar de grandes bloques de código al final del desarrollo de una función.
- **Despliegues regulares:** Al trabajar directamente en el tronco principal, se facilita el despliegue regular y frecuente de código. Esto puede ayudar a reducir el riesgo y los conflictos al mantener siempre el código en un estado de producción viable.