Nombre:

Janiel

Apellido:

Geraldo Abreu

Matt:

2022-0296

Maestro:

Kelyn Tejada

Fecha:

4/3/2025

Índice

¿Qué es Git?

Hoy en día, Git es, con diferencia, el sistema de control de versiones moderno más utilizado del mundo. Git es un proyecto de código abierto maduro y con un mantenimiento activo que desarrolló originalmente Linus Torvalds, el famoso creador del kernel del sistema operativo Linux, en 2005. Un asombroso número de proyectos de software dependen de Git para el control de versiones, incluidos proyectos comerciales y de código abierto. Los desarrolladores que han trabajado con Git cuentan con una buena representación en la base de talentos disponibles para el desarrollo de software, y este sistema funciona a la perfección en una amplia variedad de sistemas operativos e IDE (entornos de desarrollo integrados).

Git, que presenta una arquitectura distribuida, es un ejemplo de DVCS (sistema de control de versiones distribuido, por sus siglas en inglés). En lugar de tener un único espacio para todo el historial de versiones del software, como sucede de manera habitual en los sistemas de control de versiones antaño populares, como CVS o Subversion (también conocido como SVN), en Git, la copia de trabajo del código de cada desarrollador es también un repositorio que puede albergar el historial completo de todos los cambios.

Además de contar con una arquitectura distribuida, Git se ha diseñado teniendo en cuenta el rendimiento, la seguridad y la flexibilidad.

¿Para qué sirve el comando git init?

El comando git init crea un nuevo repositorio de Git. Puede utilizarse para convertir un proyecto existente y sin versión en un repositorio de Git, o para inicializar un nuevo repositorio vacío. La mayoría de los demás comandos de Git no se encuentran disponibles fuera de un repositorio inicializado, por lo que este suele ser el primer comando que se ejecuta en un proyecto nuevo.

¿Qué es una rama en Git?

En términos técnicos, una rama en Git es un puntero que señala un commit específico dentro del historial**.** Desde ahí, puedes crear nuevos commits y desarrollar funcionalidades de manera aislada del resto del proyecto.

Para que sea más claro de entender, vamos a imaginar que estás escribiendo un libro donde tienes un capítulo principal (que sería tu rama main). De repente, se te ocurre una idea genial para un giro argumental, pero no estás seguro de incluirlo. ¿Qué sería lo mejor en ese caso? Crear una copia del capítulo, trabajar en esta nueva idea y, si funciona, la incluyes en el libro original. O, en caso de que no, simplemente la descartas y conservas la versión original de tu trabajo. Bueno, las ramas en Git funcionan igual: son copias del proyecto donde puedes experimentar sin la necesidad de preocuparte porque algo pueda dejar de funcionar o dañar tu proyecto en general.

¿Cómo saber en cuál rama estoy trabajando?

- Use el comando git branch para determinar la rama actual
- Use el comando git status para determinar la rama actual

¿Quién creó Git?

Git 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.

¿Cuáles son los comandos esenciales de Git?

Configuración:

```
git config --global user.name "Tu Nombre"

git config --global user.email "tuemail@example.com"
```

Inicialización y Clonación:

git init

git clone <url>

Estados y Seguimiento:

git status

git add <archivo>

git add .

Confirmación de Cambios:

git commit -m "Mensaje"

git commit -am "Mensaje"

Historial y Revisión
git log

git log --oneline

Ramas (Branches):

git branch

git branch <nombre>

git checkout <rama>

git checkout -b <rama>

git merge <rama>

git branch -d <rama>

Sincronización con Remoto:

git remote add origin <url>

git push origin <rama>

git pull origin <rama>

git fetch

Revertir Cambios:

git reset <archivo>

git reset --hard <commit>

git revert <commit>

Otros Útiles:

git stash

git stash pop

git tag <nombre>

¿Qué es Git Flow?

Git Flow es una metodología de desarrollo basada en Git que define un conjunto de reglas y estrategias para organizar el flujo de trabajo en proyectos de software. Fue propuesta por Vincent Driessen en 2010 y es ampliamente utilizada en equipos que trabajan con integración y despliegue continuo.

Este modelo se basa en el uso estructurado de ramas para gestionar el desarrollo, nuevas características, correcciones de errores y lanzamientos de software de manera eficiente.

Estructura de Ramas en Git Flow

- **Main**: Contiene la versión estable del proyecto.
- **Develop**: Rama de desarrollo donde se integran nuevas características antes de pasarlas a main.
- **Feature**: Ramas para desarrollar nuevas funcionalidades. Se crean desde develop y se fusionan de vuelta cuando están listas.
- **Reléase**: Ramas para preparar una nueva versión antes de pasarla a main.
- **Hotfix**: Ramas para corregir errores críticos en main y luego fusionarlos en develop.

¿Qué es el desarrollo basado en trunk (Trunk Based Development)?

El Trunk Based Development (TBD) es una estrategia de desarrollo en la que todos los desarrolladores trabajan en una única rama principal (trunk, generalmente main o master), en lugar de mantener múltiples ramas de larga duración como en Git Flow. En este enfoque, las ramas de características (feature branches) son temporales y de corta duración, integrándose rápidamente en main para evitar acumulación de cambios y minimizar conflictos de fusión.

Este modelo promueve la integración continua (CI/CD), asegurando que el código se fusione frecuentemente y se mantenga siempre en un estado desplegable. Para gestionar funcionalidades en desarrollo sin afectar la estabilidad del sistema, se utilizan Feature Flags, que permiten habilitar o deshabilitar características sin necesidad de mantener ramas separadas.

El Trunk Based Development es ampliamente utilizado en entornos ágiles y equipos que implementan despliegue continuo, ya que facilita la entrega rápida de software y mejora la colaboración al evitar la fragmentación del código en múltiples ramas.

Bibliografía:

https://es.wikipedia.org/wiki/Git

https://www.delftstack.com/es/howto/git/git-which-branch-am-i-on/

http://keepcoding.io/desarrollo-web/que-son-las-ramas-en-git/

https://www.atlassian.com/es/git/tutorials/what-is-git