Universidad Instituto Tecnológico de las Américas (ITLA)

Presentado Por:
Adriana Cunillera Feliz

Matricula:
2023-0993

Presentado a:
Kelyn Tejada Belliard

Materia
programación III

Fecha

2/4/2025

Índice

1. ¿Qué es Git?

Git es un sistema de control de versiones de código fuente. Es una herramienta que se utiliza para llevar un registro de los cambios en el código fuente de un proyecto y permitir a varios desarrolladores trabajar en el mismo proyecto de manera simultánea.

Git es una herramienta muy popular en la comunidad de desarrollo de software debido a su flexibilidad, velocidad y capacidad para manejar grandes proyectos. También es ampliamente utilizado en proyectos de código abierto, ya que permite a cualquier persona contribuir al proyecto y hacer seguimiento de los cambios realizados en el código fuente.

Con Git, es posible crear un repositorio que contenga el código fuente de un proyecto y llevar un registro de todos los cambios realizados en el repositorio. Los desarrolladores pueden colaborar en el mismo proyecto al hacer «commits» (guardar cambios) en el repositorio y luego «pushear» (enviar) estos cambios a un servidor centralizado para que puedan ser utilizados por otros desarrolladores.

2. ¿Para qué sirve el comando git init?

En comparación con SVN, el comando git init permite crear de manera increíblemente sencilla nuevos proyectos con control de versiones. Con Git, no es necesario que crees un repositorio, importes los archivos ni extraigas una copia de trabajo. Basta con que utilices el comando cd en el subdirectorio de tu proyecto y ejecutes git init para que tengas un repositorio de Git totalmente funcional.

Si ya has ejecutado git init en el directorio de un proyecto y este contiene un subdirectorio de .git, puedes volver a ejecutar git init de forma segura en el mismo directorio del proyecto. No se reemplazará la configuración de. git existente.

3. ¿Qué es una rama en Git?

Una rama Git es simplemente un apuntador móvil apuntando a una de esas confirmaciones. La rama por defecto de Git es la rama master. Con la primera confirmación de cambios que realicemos, se creará esta rama principal master apuntando a dicha confirmación.

4. ¿Cómo saber en cuál rama estoy trabajando?

Para saber qué ramas están disponibles y cuál es el nombre de la rama actual, ejecuta git branch . En el ejemplo anterior, se muestra cómo ver una lista de ramas disponibles ejecutando el comando git branch y cómo cambiar a una rama específica, en este caso, la rama feature_inprogress_branch.

5. ¿Quién creó Git?

Git es, con diferencia, el sistema de control de versiones moderno más utilizado en el mundo. Git es un proyecto de código abierto maduro y con un mantenimiento activo, desarrollado originalmente en 2005 por Linus Torvalds, el famoso creador del núcleo del sistema operativo Linux.

Un número asombroso de proyectos de software, tanto comerciales como de código abierto, dependen de Git para el control de versiones. Los desarrolladores que han trabajado con Git cuentan con una amplia experiencia en desarrollo de software y funciona correctamente en una amplia gama de sistemas operativos e IDE (Entornos de Desarrollo Integrados).

6. ¿Cuáles son los comandos esenciales de Git?

git add: Mueve los cambios del directorio de trabajo al área del entorno de ensayo. Así puedes preparar una instantánea antes de confirmar en el historial oficial.

rama de git: Este comando es tu herramienta de administración de ramas de uso general. Permite crear entornos de desarrollo aislados en un solo repositorio.

Git Checkout: Además de extraer las confirmaciones y las revisiones de archivos antiguas, git checkout también sirve para navegar por las ramas existentes. Combinado con los comandos básicos de Git, es una forma de trabajar en una línea de desarrollo concreta.

git clean: Elimina los archivos sin seguimiento de tu directorio de trabajo. Es la contraparte lógica de git reset, que normalmente solo funciona en archivos con seguimiento.

git clone: Crea una copia de un repositorio de Git existente. La clonación es la forma más habitual de que los desarrolladores obtengan una copia de trabajo de un repositorio central.

git commit: Confirma la instantánea preparada en el historial del proyecto. En combinación con git add, define el flujo de trabajo básico de todos los usuarios de Git.

git commit –amend: Pasar la marca --amend a git commit permite modificar la confirmación más reciente. Es muy práctico si olvidas preparar un archivo u omites información importante en el mensaje de confirmación.

git config: Este comando va bien para establecer las opciones de configuración para instalar Git. Normalmente, solo es necesario usarlo inmediatamente después de instalar Git en un nuevo equipo de desarrollo.

git fetch: Con este comando, se descarga una rama de otro repositorio junto con todas sus confirmaciones y archivos asociados. Sin embargo, no intenta integrar nada en el repositorio local. Esto te permite inspeccionar los cambios antes de fusionarlos en tu proyecto.

7. ¿Qué es Git Flow?

Gitflow es un modelo alternativo de creación de ramas en Git en el que se utilizan ramas de función y varias ramas principales. Fue Vincent Driessen en nvie quien lo publicó por primera vez y quien lo popularizó.

Gitflow puede utilizarse en proyectos que tienen un ciclo de publicación programado, así como para la práctica recomendada de DevOps de entrega continua. Este flujo de trabajo no añade ningún concepto o comando nuevo, aparte de los que se necesitan para el flujo de trabajo de ramas de función. Lo que hace es asignar funciones muy específicas a las distintas ramas y definir cómo y cuándo deben estas interactuar.

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

El desarrollo basado en troncos (TBD) es una estrategia de desarrollo de software en la que los ingenieros integran cambios menores con mayor frecuencia en el código base principal y trabajan a partir de la copia principal en lugar de trabajar en ramas de características de larga duración . Este modelo de desarrollo se suele utilizar como parte de un flujo de trabajo de desarrollo de integración continua.

Con muchos ingenieros trabajando en la misma base de código, es importante contar con una estrategia para el control del código fuente y la colaboración entre los ingenieros. Para evitar anular los cambios de los demás, los ingenieros crean su propia copia del código base, denominadas ramas. Siguiendo una analogía con un árbol, la copia maestra a veces se denomina línea principal o tronco. El proceso de incorporar los cambios de la copia de un individuo al tronco maestro principal se denomina fusión.

Bibliografía

https://davinciti.com/que-es-y-para-que-sirve-git/#:~:text=Git%20es%20un%20sistema%20de,mismo%20proyecto%20de%20manera%20simult%C3%A1nea.

https://www.atlassian.com/es/git/tutorials/setting-up-a-repository/git-init#:~:text=Leer%20tutorial-,Uso,extraigas%20una%20copia%20de%20trabajo.

https://git-scm.com/book/es/v2/Ramificaciones-en-Git-%C2%BFQu%C3%A9-es-una-rama%3F#:~:text=Una%20rama%20Git%20es%20simplemente,master%20apuntando%20a%20dicha%20confirmaci%C3%B3n.

https://git-scm.com/book/es/v2/Ramificaciones-en-Git-%C2%BFQu%C3%A9-es-una-rama%3F#:~:text=Una%20rama%20Git%20es%20simplemente,master%20apuntando%20a%20dicha%20confirmaci%C3%B3n.

https://www.atlassian.com/es/git/tutorials/using-branches/git-checkout#:~:text=Para%20saber%20qu%C3%A9%20ramas%20est%C3%A1n,rama%20actual%2C%20ejecuta%20git%20branch%20.&text=En%20el%20ejemplo%20anterior%2C%20se,este%20caso%2C%20la%20rama%20feature_inprogress_branch%20.

https://www.atlassian.com/es/git/glossary#commands

https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow#:~:text=%C2%BFQu%C3%A9%20es%20Gitflow%3F,vez%20y%20quien%20lo%20populariz%C3%B3.

https://www.optimizely.com/optimization-glossary/trunk-based-

development/