HISTORIA Y COMANDOS DE GIT

DANIEL FUQUENE

JULIANA TIQUE ORTIZ

SENA

24/02/2025

Historia del GIT

Git comenzó con un poco de destrucción creativa y una gran polémica. El kernel de Linux es un proyecto de software de código abierto con un alcance bastante amplio. Durante la mayor parte del mantenimiento del kernel de Linux (1991-2002), los cambios en el software se realizaban a través de parches y archivos. En el 2002, el proyecto del kernel de Linux empezó a usar un DVCS propietario llamado BitKeeper.

En el 2005, la relación entre la comunidad que desarrollaba el kernel de Linux y la compañía que desarrollaba BitKeeper se vino abajo y la herramienta dejó de ser ofrecida de manera gratuita. Esto impulsó a la comunidad de desarrollo de Linux (y en particular a Linus Torvalds, el creador de Linux) a desarrollar su propia herramienta basada en algunas de las lecciones que aprendieron mientras usaban BitKeeper. Algunos de los objetivos del nuevo sistema fueron los siguientes:

* Velocidad
* Diseño sencillo
* Gran soporte para desarrollo no lineal (miles de ramas paralelas)
* Completamente distribuido
* Capaz de manejar grandes proyectos (como el kernel de Linux) eficientemente (velocidad y tamaño de los datos).

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.

Comandos de GIT

Git add

Mueve los cambios del directorio de trabajo al área del entorno de ensayo.asi puedes preparar una instantánea antes de conformar en el historial oficial.

Gir Branch

Este comando es tu herramienta de administración de ramas de uso general. Permite crear entornos e desarrollo asilados 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.

Git init

Inicializa un nuevo repositorio de Git. Si quieres poner un proyecto bajo un control de revisiones, este es el primer comando que debes aprender.

Git log

Permite explorar las revisiones anteriores de un proyecto. Proporciona varias opciones de formato para mostrar las instantáneas confirmadas.

Git merge

Es una forma eficaz de integrar los cambios de ramas divergentes. Después de bifurcar el historial del proyecto con git branch, git merge permite unirlo de nuevo.

Git pull

Este comando es la versión automatizada de git fetch. Descarga una rama de un repositorio remoto e inmediatamente la fusiona en la rama actual. Este es el equivalente en Git de svn update.

Git push

Enviar (push) es lo opuesto a recuperar (fetch), con algunas salvedades. Permite mover una o varias ramas a otro repositorio, lo que es una buena forma de publicar contribuciones. Es como svn commit, pero envía una serie de confirmaciones en lugar de un solo conjunto de cambios.

Git rebase

Un cambio de base con git rebase permite mover las ramas, lo que ayuda a evitar confirmaciones de fusión innecesarias. El historial lineal resultante suele ser mucho más fácil de entender y explorar.

Git rebase -i

La marca -i se usa para iniciar una sesión de cambio de base interactivo. Esto ofrece todas las ventajas de un cambio de base normal, pero te da la oportunidad de añadir, editar o eliminar confirmaciones sobre la marcha.

Git reflog

Git realiza el seguimiento de las actualizaciones en el extremo de las ramas mediante un mecanismo llamado registro de referencia o reflog. Esto permite volver a los conjuntos de cambios aunque no se haga referencia a ellos en ninguna rama o etiqueta.

Git remote

Es un comando útil para administrar conexiones remotas. En lugar de pasar la URL completa a los comandos fetch, pull y push, permite usar un atajo más significativo.

Git reset

Deshace los cambios en los archivos del directorio de trabajo. El restablecimiento permite limpiar o eliminar por completo los cambios que no se han enviado a un repositorio público.

Git revert

Permite deshacer una instantánea confirmada. Si descubres una confirmación errónea, revertirla es una forma fácil y segura de eliminarla por completo del código base.

Git status

Muestra el estado del directorio en el que estás trabajando y la instantánea preparada. Lo mejor es que lo ejecutes junto con git add y git commit para ver exactamente qué se va a incluir en la próxima instantánea.