Introducción a Git

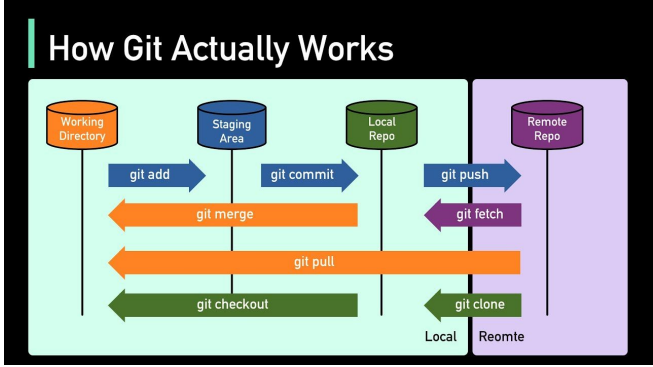
Cómo funciona Git, uso básico, trabajo con ramas y gestión de conflictos

Gerardo Andres Almaguer Ramos

¿Qué es Git?

- Sistema de control de versiones distribuido.
- Permite gestionar versiones de archivos y coordinar trabajo en equipo.
- Creado por Linus Torvalds en 2005 para el desarrollo del núcleo de Linux.

Funcionamiento Básico de Git



Repositorio Local: Almacena el historial completo de cambios en tu máquina.

Repositorio Remoto: Almacena el historial en un servidor central (e.g., GitHub, GitLab).

Commits: Registra los cambios en el repositorio local.

Ramas: Permiten trabajar en diferentes versiones del proyecto simultáneamente.

Instalación y Configuración Inicial

Instalación: `sudo apt-get install git` (Debian/Ubuntu) o `brew install git` (macOS).

Configuración Inicial:

- `git config --global user.name "Tu Nombre"`
- `git config --global user.email "tu.email@example.com"`

Comandos Básicos de Git

`git init`: Inicializa un nuevo repositorio.

`git clone [url]`: Clonar un repositorio remoto.

`git add [archivo]`: Agrega archivos al área de preparación.

`git commit -m "mensaje"`: Registra los cambios en el repositorio.

`git status`: Muestra el estado del repositorio.

`git log`: Muestra el historial de commits.

Uso de Ramas en Git

Crear una Rama: `git branch [nombre-rama]`

Cambiar de Rama: `git checkout [nombre-rama]` o `git switch [nombre-rama]`

Fusionar Ramas: `git merge [nombre-rama]` para integrar cambios de una rama a otra.

Eliminar una Rama: `git branch -d [nombre-rama]`

Gestión de Conflictos en Git

Conflictos al Fusionar: Ocurren cuando dos ramas modifican la misma parte de un archivo.

Detectar Conflictos: Git te avisa durante un merge.

Resolver Conflictos:

- Editar manualmente los archivos conflictivos.
- Marcar el conflicto como resuelto con `git add [archivo]`.
- Finalizar el merge con `git commit`.

Estrategias para Trabajar en Equipo

Pull Requests: Solicitudes para revisar e integrar cambios.

Revisión de Código: Asegura calidad y consistencia en el código.

Sincronización Regular: Mantén tu repositorio actualizado con `git pull`.

Conclusión

En resumen, Git es una herramienta esencial para el control de versiones y la colaboración en proyectos de desarrollo. Hemos cubierto desde los comandos básicos para gestionar archivos y commits hasta el trabajo con ramas y la resolución de conflictos. La comprensión de estas funcionalidades te permitirá manejar eficazmente tus proyectos y colaborar con otros de manera más eficiente.