Luis David Sanchez Benavente

Que es GIT?

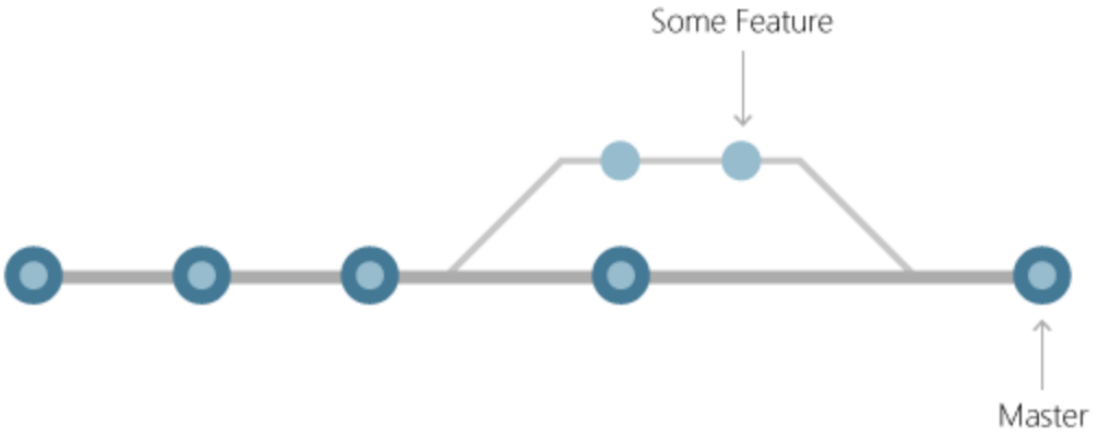
GIT es un sistema de control de versiones distribuido ampliamente utilizado en el desarrollo de software. Fue creado por Linus Torvalds en 2005 para administrar el desarrollo del kernel de Linux, pero desde entonces se ha convertido en una herramienta estándar en la industria del desarrollo de software.

En pocas palabras, GIT es una herramienta que permite a los desarrolladores realizar un seguimiento de los cambios en el código fuente a lo largo del tiempo. Permite a los equipos de desarrollo colaborar de manera eficiente, coordinando los cambios realizados por diferentes personas en un proyecto y fusionando esos cambios de manera ordenada.

Al utilizar GIT, los desarrolladores pueden trabajar en paralelo en diferentes ramas (branch) del código fuente sin interferir entre sí. Cada desarrollador puede realizar cambios en su rama y luego combinarlos (merge) con el código principal cuando estén listos.

Ramas

Cada desarrollador guarda los cambios en su propio repositorio de código local. Como resultado, puede haber muchos cambios diferentes en función de la misma confirmación. Git proporciona herramientas para aislar los cambios y volver a combinarlos posteriormente. Las ramas, que son punteros ligeros para trabajar en curso, administran esta separación. Una vez finalizado el trabajo creado en una rama, se puede combinar de nuevo en la rama principal (o troncal) del equipo.



Control de versiones con GIT

Las ramas permiten el desarrollo flexible y simultáneo. La rama principal contiene código estable y de alta calidad desde el que se publica. Las ramas de características contienen trabajo en curso, que se combinan en la rama principal tras la finalización. Al separar la rama de versión del desarrollo en curso, es más fácil administrar código estable y enviar actualizaciones más rápidamente.

Estados de un archivo en GIT

En GIT, los archivos pueden pasar por diferentes estados a medida que se realizan cambios en ellos. Los principales estados de un archivo en GIT son los siguientes:

1. Untracked (No rastreado): Cuando un archivo se crea o se agrega a un directorio que está dentro de un repositorio GIT, inicialmente se encuentra en estado "Untracked". Esto significa que GIT no está monitoreando ni registrando los cambios en ese archivo.

2. Tracked (Rastreado): Un archivo se convierte en "Tracked" cuando ha sido agregado al índice de GIT mediante el comando "git add". Los archivos rastreados son monitoreados por GIT para detectar cambios y registrarlos en los commits.

3. Modified (Modificado): Un archivo rastreado se marca como "Modified" cuando se realizan cambios en él después de que ha sido agregado al índice. Es decir, se han realizado modificaciones en el archivo desde el último commit registrado.

4. Staged (Preparado): Cuando se han realizado cambios en un archivo modificado y se agregan al índice utilizando el comando "git add", el archivo pasa al estado "Staged". Esto significa que los cambios en ese archivo se incluirán en el próximo commit.

5. Committed (Confirmado): Después de ejecutar el comando "git commit" con los cambios preparados (staged), el archivo se considera "Committed". El cambio se registra permanentemente en el repositorio GIT como parte de un commit, y el archivo vuelve a su estado "Modified" si se realizan más cambios.

Estos estados de archivo en GIT permiten a los desarrolladores realizar un seguimiento preciso de los cambios y administrar el flujo de trabajo de manera eficiente al colaborar en proyectos de desarrollo de software.

Como se configura un repositorio?

Para configurar un repositorio en GIT, se pueden seguir los siguientes pasos:

1. Inicializar un repositorio: Navega hasta el directorio raíz del proyecto en tu máquina y ejecutar el comando "git init". Esto creará un repositorio GIT vacío en ese directorio.

2. Configurar el nombre de usuario y el correo electrónico: GIT utiliza esta información para identificar al autor de los commits. Utilizar los siguientes comandos para configurar tu nombre de usuario y correo electrónico:

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

3. Opcionalmente, puedes configurar otras opciones según tus necesidades, como el editor de texto predeterminado, las opciones de formato, etc. Puedes hacerlo utilizando comandos como:

```
git config --global core.editor "nombre_editor"
git config --global core.autocrlf true
:::
```

4. Verificar la configuración: se podra verificar la configuración actual ejecutando el comando "git config --list". Esto mostrará todas las opciones de configuración establecidas en tu máquina.

5. Conectar un repositorio remoto (opcional): Si se conectar un repositorio local a un repositorio remoto, utilizar el comando "git remote add <nombre_remoto> <URL_remoto>". Esto permite la colaboración con otros desarrolladores y la sincronización de cambios. Algunos ejemplos comunes de repositorios remotos son GitHub, GitLab o Bitbucket.

```
git remote add origin https://github.com/tu_usuario/tu_repositorio.git
```

Una vez que se haya configurado el repositorio GIT, se podra comenzar a agregar archivos, realizar commits y trabajar en tu proyecto. Recuerdar que estos pasos son para configurar un repositorio nuevo. Si se trabajando en un proyecto existente, se podra clonar un repositorio remoto utilizando el comando "git clone" en lugar de inicializar uno nuevo.

Comandos en GIT

GIT ofrece una amplia gama de comandos para administrar y trabajar con repositorios. A continuación, estos son algunos de los comandos más comunes que se utilizan en GIT:

1. git init: Inicializa un nuevo repositorio GIT en un directorio.

2. git clone <URL_repositorio>: Clona un repositorio remoto y crea una copia local en tu máquina.

3. git status: Muestra el estado actual del repositorio, incluyendo los archivos modificados, agregados o eliminados.

4. git add <archivo>: Agrega un archivo específico al índice de GIT para que sea rastreado.

5. git add .: Agrega todos los archivos modificados y nuevos en el directorio actual al índice.

6. git commit -m "mensaje_de_confirmación": Confirma los cambios realizados en los archivos agregados al índice, creando un nuevo commit en el historial del repositorio.

7. git push: Envía los commits locales al repositorio remoto.

8. git pull: Descarga los cambios desde el repositorio remoto y los fusiona con tu copia local.

9. git branch: Muestra las ramas disponibles en el repositorio.

10. git checkout <nombre_rama>: Cambia a una rama específica.

11. git merge <nombre_rama>: Fusiona los cambios de una rama en la rama actual.

12. git remote add <nombre_remoto> <URL_remoto>: Conecta un repositorio local a un repositorio remoto.

13. git log: Muestra el historial de commits realizados en el repositorio.

14. git diff: Muestra las diferencias entre el estado actual del repositorio y el último commit.

15. git stash: Guarda temporalmente los cambios no confirmados en una pila de almacenamiento temporal (stash) para trabajar en otra tarea.