Angela María Bethancourt López
999012804

Que es GIT

Git es un sistema de control de versiones distribuido que permite gestionar y realizar un seguimiento de los cambios en el código fuente y otros archivos de un proyecto de desarrollo de software. Proporciona un historial completo de todos los cambios realizados, quién los hizo y cuándo se hicieron. Esto resulta especialmente útil en la ingeniería, donde se trabaja en equipos y se realizan múltiples modificaciones en paralelo.
Los proyectos se almacenan en repositorios, que pueden ser locales o remotos (en un servidor). Cada desarrollador tiene una copia completa del repositorio en su máquina local, lo que permite trabajar de forma independiente y sin necesidad de una conexión constante a un servidor central.
La unidad básica en Git es el commit, este representa un conjunto de cambios realizados en el repositorio. Cada commit tiene un identificador único, un mensaje descriptivo y una referencia a su commit padre, lo que permite seguir la historia de los cambios.
Git utiliza ramas para permitir un desarrollo paralelo y la implementación de nuevas características sin afectar la versión principal del proyecto. Las ramas facilitan la colaboración entre diferentes desarrolladores, ya que cada uno puede trabajar en su propia rama y luego fusionar los cambios en la rama principal cuando estén listos. Además, ofrece herramientas para resolver conflictos que pueden surgir cuando se fusionan cambios en diferentes ramas que afectan al mismo archivo o línea de código. Esto permite manejar de manera efectiva los escenarios en los que múltiples personas han modificado la misma parte de un proyecto.

Control de versiones con GIT

Básicamente el funcionamiendo del control de versiones con Git funciona de la siguiente manera:

1. La inicialización del repositorio: Se crea un repositorio Git en una carpeta del proyecto con el comando "git init".
2. Los Commits: Cada cambio se guardan en commits, estos representan diferentes versiones de un proyecto. Se utiliza "git add" para preparar los cambios y "git commit" para crear un commit con un mensaje descriptivo.
3. Ramas: Git permite trabajar en diferentes ramas separadas para desarrollar funcionalidades sin afectar la rama principal. Se pueden crear, cambiar y fusionar ramas con distintos comandos
4. Repositorios remotos: Git facilita la colaboración con repositorios remotos. Se pueden clonar, enviar y recibir cambios.
5. Gestión de conflictos: Git ofrece herramientas para resolver conflictos cuando se fusionan cambios conflictivos.

El control de versiones con Git brinda un mayor control sobre el desarrollo de software, permite realizar un seguimiento de los cambios, colaborar con otros desarrolladores y revertir a versiones anteriores si es necesario.

Estados de un archivo en GIT

Un archivo puede pasar por varios estados a medida que se trabaja con él en un repositorio. Los principales estados de un archivo en Git son los siguientes:

- Untracked (No rastreado): El archivo no está siendo seguido por Git.
- Modified (Modificado): Se han realizado cambios en el archivo desde el último commit.
- Staged (Preparado): Los cambios en el archivo se han agregado a la zona de preparación para ser incluidos en el próximo commit.
- Committed (Confirmado): Los cambios en el archivo se han guardado de forma permanente en el repositorio de Git.

Estos estados permiten tener un mayor control y seguimiento de los cambios en los archivos a medida que se trabaja en un proyecto con Git.

Como se configura un repositorio

Para poder configurar un repositorio en git se pueden seguir los siguientes pasos:

1. Inicializar un repositorio local: Ejecutar el comando **git init** en el directorio raíz del proyecto para crear un nuevo repositorio Git.
2. Configurar el nombre de usuario: Utilizar **git config user.name "nombre"** para establecer el nombre de usuario en los commits.
3. Configurar la dirección de correo electrónico: Utilizar el comando **git config user.email "email@email.com"** para establecer la dirección de correo electrónico.
4. Verificar la configuración: Con el comando **git config --list** se verificar la configuración actual del repositorio.

Una vez se haya completado estos pasos, el repositorio Git estará configurado y listo para empezar a trabajar. Se pueden agregar archivos, hacer commits y utilizar todas las funcionalidades de control de versiones que ofrece Git.

Comandos en GIT

Los comandos más importantes que se utilizan con frecuencia son:

1. **git init**: Inicializa un nuevo repositorio Git en el directorio actual.
2. **git clone**: Clona un repositorio Git existente en un nuevo directorio.
3. **git add**: Agrega archivos al área de preparación (staging area) para ser incluidos en el próximo commit.
4. **git commit**: Crea un nuevo commit con los cambios preparados en el área de preparación.
5. **git status**: Muestra el estado actual del repositorio, incluyendo los archivos modificados y los cambios pendientes de ser confirmados.
6. **git log**: Muestra el historial de commits realizados en el repositorio.
7. **git branch**: Muestra las ramas existentes en el repositorio o crea una nueva rama.

8. **git checkout**: Cambia entre ramas o restaura archivos a una versión específica.
9. **git merge**: Fusiona los cambios de una rama en otra.
10. **git pull**: Obtiene los cambios más recientes desde un repositorio remoto y los fusiona con la rama actual.
11. **git push**: Envía los commits locales a un repositorio remoto.
12. **git remote**: Gestiona las conexiones con repositorios remotos.
13. **git diff**: Muestra las diferencias entre versiones o ramas.
14. **git stash**: Guarda cambios temporales sin realizar commit.
15. **git reset**: Deshace commits o deshace cambios en el repositorio.

Existen muchos otros comandos y opciones disponibles que permiten realizar tareas más avanzadas. Se puede obtener más información sobre cada comando utilizando **git help <comando>** o consultando la documentación oficial de Git.