GIT

Git es un sistema de control de versiones distribuido creado por Linus Torvalds en 2005. Se desarrolló para gestionar eficientemente el control de versiones de proyectos de software. Lo que distingue a Git de otros sistemas de control de versiones es su enfoque en la velocidad, la sencillez y la escalabilidad.

En Git, cada desarrollador tiene una copia completa del repositorio en su sistema, lo que significa que pueden trabajar de manera aislada en su propia rama de desarrollo y fusionar los cambios de manera controlada. Esto facilita la colaboración y permite el seguimiento de múltiples ramas de desarrollo al mismo tiempo.

Comandos básicos en GIT

Los siguientes comandos de git son un buen punto de partida para comenzar a trabajar con un controlador de versiones y colaboración de proyectos

1. git init: Este comando inicia un nuevo repositorio Git en un directorio local. Una vez ejecutado, se creará una carpeta oculta llamada ".git" que almacena toda la información relacionada con el repositorio.
2. git clone [URL]: Utiliza este comando para copiar un repositorio remoto en tu sistema local. Esto se hace generalmente al principio del desarrollo de un proyecto.
3. git add [archivo]: Agrega cambios en un archivo específico a la "zona de preparación", lo que significa que están listos para ser confirmados en el repositorio.
4. git commit -m "Mensaje del commit": Realiza un commit con los cambios en la zona de preparación. El mensaje es una descripción breve de los cambios realizados en este commit.
5. git status: Muestra el estado de los archivos en tu repositorio local. Indica los archivos modificados, agregados o no rastreados.
6. git log: Muestra un registro de todos los commits en el repositorio, incluyendo detalles como el autor, la fecha y el mensaje del commit.
7. git pull: Actualiza tu repositorio local con los cambios del repositorio remoto. Es útil para mantener sincronizados los cambios realizados por otros colaboradores.
8. git push: Sube tus commits locales al repositorio remoto. Esto comparte tus cambios con otros colaboradores.
9. git branch: Muestra una lista de las ramas en tu repositorio local. La rama actual se resalta con un asterisco (*).
10. git checkout [nombre de la rama]: Cambia a una rama diferente en tu repositorio local. Esto te permite trabajar en una rama específica.

Trabajando con repositorios en Github (Branches, Merge, Conflicts)

Trabajar con repositorios en GitHub (ramas, fusiones y conflictos) es esencial para colaborar en proyectos de desarrollo de software.

Branches en GitHub

Las ramas en GitHub son copias independientes de un proyecto en un estado específico. Se utilizan para trabajar en nuevas características, solucionar problemas o realizar modificaciones sin afectar la rama principal del proyecto, generalmente denominada "main" o "master". Algunos comandos y acciones relacionados con las ramas en GitHub incluyen:

- Crear una rama: Puedes crear una nueva rama a través de la interfaz web de GitHub o utilizando el comando git branch [nombre de la rama] en tu repositorio local.

- Cambiar a una rama: Puedes cambiar de rama localmente con git checkout [nombre de la rama]. En GitHub, puedes seleccionar la rama en la que deseas trabajar en el menú desplegable.

Merge en GitHub

Las fusiones en GitHub se utilizan para combinar los cambios realizados en una rama en otra, como unir una rama de desarrollo a la rama principal del proyecto. Algunos aspectos clave de las fusiones en GitHub incluyen:

-Fusionar ramas:En la interfaz de GitHub, puedes iniciar una solicitud de extracción (Pull Request) para fusionar cambios de una rama en otra. Esto permite a otros colaboradores revisar los cambios antes de la fusión.

-Resolución de conflictos: Si hay conflictos entre los cambios en la rama de destino y la rama que estás fusionando, GitHub te guiará para resolver estos conflictos. Debes editar manualmente los archivos en conflicto y luego confirmar los cambios.

-Eliminar una rama: Después de una fusión exitosa, puedes eliminar la rama que ya no necesitas. Esto ayuda a mantener un repositorio limpio.

Conflicts en GitHub

Los conflictos en GitHub pueden surgir cuando se intenta fusionar cambios en una rama que han afectado las mismas líneas de código que se han modificado en la rama de destino. Aquí hay algunos puntos clave sobre cómo manejar conflictos en GitHub:

-Resolver conflictos: GitHub te mostrará las áreas en conflicto en un archivo. Debes editar el archivo para conservar las partes que desees mantener y eliminar las que no necesitas. Luego, debes marcar el conflicto como resuelto y confirmar los cambios.

-Commit de resolución: Una vez que hayas resuelto los conflictos, debes realizar un commit para registrar las modificaciones que has realizado en la resolución.

-Continuar con la fusión: Después de resolver los conflictos y realizar un commit de resolución, puedes continuar con el proceso de fusión. En la interfaz de GitHub, puedes marcar que los conflictos se han resuelto y completar la fusión.