➔ **Ejemplos prácticos para gestionar ramas, manejar conflictos y colaborar en equipo utilizando Git y GitHub.**

Ejemplo 1: Creación y Fusión de Ramas

Supongamos que estás trabajando en un proyecto y deseas desarrollar una nueva funcionalidad sin afectar la rama principal.

1. Crear una nueva rama:

   Imagina que estás desarrollando una nueva característica.

2. Realizar cambios en la nueva rama:

   Modifica algún archivo, por ejemplo `archivo.txt`, y luego añádelo al área de staging y haz un commit:

3. Volver a la rama principal:

   Después de haber terminado tu desarrollo, vuelve a la rama principal.

4. Fusionar la rama:

   Ahora puedes fusionar la rama `nueva-funcionalidad` con la rama principal.

5. Eliminar la rama (opcional):

   Si ya no necesitas la rama, puedes eliminarla.

Ejemplo 2: Resolución de Conflictos de Fusión

Supongamos que tú y otro desarrollador modifican el mismo archivo y tratan de fusionar sus ramas. Esto puede generar conflictos.

1. Crear un conflicto:

   - Tú modificas una línea en `archivo.txt` en la rama `main` y haces commit.

   - Otro desarrollador crea y trabaja en la rama `otra-funcionalidad`, modifica la misma línea y también hace commit.

2. Fusionar y gestionar el conflicto:

Ahora intentas fusionar la rama del otro desarrollador.

Git te notificará sobre el conflicto y te mostrará qué líneas están en conflicto en el archivo. Tendrás que editar manualmente el archivo para resolverlo. Por ejemplo:

Elimina los delimitadores `<<<<<`, `=====`, `>>>>>` y decide cuál versión o combinación de ambas quieres mantener.

3. Añadir el archivo corregido y continuar:

Una vez que hayas resuelto el conflicto, añade el archivo al área de staging y realiza el commit.

Ejemplo 3: Trabajo Colaborativo en GitHub

Un flujo colaborativo típico usando Git y GitHub podría ser el siguiente:

1. Clonar un repositorio remoto:

Cada miembro del equipo puede clonar el repositorio remoto a su máquina local.

2. Crear una rama para desarrollar una tarea específica:

Cada desarrollador debería crear una nueva rama para su tarea.

3. Subir los cambios al repositorio remoto:

Después de realizar los cambios, el desarrollador puede hacer commit y enviar los cambios al repositorio remoto.

4. Crear un Pull Request (PR) en GitHub:

En la plataforma de GitHub, el desarrollador puede crear un Pull Request para que otros revisen el código antes de fusionarlo con la rama principal.

5. Revisar y fusionar el PR:

Después de la revisión, el equipo puede fusionar el PR a la rama `main` directamente desde GitHub.

6. Actualizar la rama local:

Los otros miembros del equipo deben actualizar su rama `main` local para asegurarse de tener los cambios más recientes.

Ejemplo 4: Rebase para Mantener un Historial de Commits Limpio

El rebase se usa para "rebasar" una rama sobre otra y mantener un historial de commits más lineal.

1. Crear una rama y hacer commits:

Estás trabajando en una nueva rama:

2. Rebase para actualizar tu rama con los últimos cambios de la rama `main`:

En lugar de hacer un `merge`, puedes hacer un `rebase` para rebasar tu rama sobre la más reciente versión de `main`.

Si hay conflictos, Git te pedirá que los resuelvas, como en el ejemplo anterior.

3. Finalizar el rebase:

Una vez resueltos los conflictos, puedes continuar el proceso de rebase:

4. Empujar los cambios:

Si ya habías empujado tu rama antes, es posible que necesites usar `--force` para actualizar el historial:

Este flujo ayuda a mantener el historial del repositorio más limpio y organizado.