Introducción a GIT

GitHub es una plataforma para almacenar, compartir y trabajar junto con otros usuarios para escribir código.

Almacenar tu código en un "repositorio" en GitHub te permite:

- **Presentar o compartir** el trabajo.
- **Seguir y administrar** los cambios en el código a lo largo del tiempo.
- Dejar que otros usuarios **revisen** el código y realicen sugerencias para mejorarlo.
- **Colaborar** en un proyecto compartido, sin preocuparse de que los cambios afectarán al trabajo de los colaboradores antes de que esté listo para integrarlos.

El trabajo colaborativo, una de las características fundamentales de GitHub, es posible gracias al software de código abierto Git, en el que se basa GitHub.

Git es un sistema de control de versiones (VCS) que realiza un seguimiento de los cambios en los archivos. Es especialmente útil cuando un grupo de personas cuando se están haciendo cambios en los mismos archivos al mismo tiempo.

Normalmente, para hacerlo en un flujo de trabajo basado en Git, harías lo siguiente:

- **Crear una rama**: a partir de la copia principal de archivos.
- **Realizar modificaciones** en los archivos de forma independiente y segura en tu propia rama personal.
- **Fusiona mediante combinación** y de forma inteligente los cambios específicos en la copia principal de archivos, de modo que los cambios no afecten a las actualizaciones de otras personas.
- **Realiza un seguimiento** de tus cambios y los de otras personas, por lo que todos siguen trabajando en la versión más actualizada del proyecto.

El VCS rastrea el historial de los cambios y cualquier versión anterior puede recuperarse en cualquier momento, así como también proporciona una vista unificada de un proyecto.

Comandos Básicos en GIT

1. Inicio de un repositorio nuevo:

```
git init
```

Para comienzar un nuevo proyecto, puedes inicializar un repositorio Git en el directorio del proyecto utilizando este comando.

2. Clonar un repositorio remoto:

```
git clone [url]
```

Si deseas trabajar en un proyecto existente, puedes clonar un repositorio Git remoto hacia tu máquina local utilizando su URL.

3. Agregar archivos al área de preparación:

```
git add [archivo(s)]
```

Se utiliza cuando realizamos cambios en los archivos de tu proyecto, puedes agregar esos cambios al área de preparación para incluirlos en el próximo commit.

4. Realizar un commit de los cambios:

```
git commit -m "Mensaje del commit"
```

Una vez que has agregado los cambios al área de preparación, puedes confirmarlos permanentemente en el repositorio utilizando un commit, junto con un mensaje descriptivo.

5. Verificar el estado del repositorio:

```
git status
```

Este comando permite ver qué archivos han sido modificados, añadidos o pendientes de confirmación en tu repositorio.

6. Obtener cambios desde un repositorio remoto:

git pull

Para repositorios que son compartido con otros, puedes usar `git pull` para obtener los últimos cambios del repositorio remoto y fusionarlos con tu rama local.

7. Enviar tus commits hacia un repositorio remoto:

git push

Después de haber confirmado tus cambios localmente, puedes enviar tus commits hacia un repositorio remoto para compartir tus contribuciones con otros miembros del equipo.

8. Crear una nueva rama:

git branch [nombre de la rama]

Se utiliza para crear una nueva rama en tu repositorio.

9.Navegación de ramas

git checkout

Este comando sirve para navegar entre ramas o para restaurar archivos a un estado especifico.

10. Congifuración.

git congif

Se utiliza para establecer las opciones de configuración para instalar git, y se utiliza para instalar en equipos de desarrollo.

11. Descargar una rama.

git fetch

Descarga una rama de otro repositorio junto con todas sus confirmaciones y archivos asociados, pero no se debe de integrar nada en el repositorio locas. Permite inspeccionar cambios antes de fusionarlos en tu proyecto.

12. Revisión de versiones.

git log

Permite explorar las revisiones anteriores de un proyecto y proporciona varias opciones de formato para mostrar las instantáneas confirmadas.

13. Ramas divergentes

git merge

Se utiliza para integrar los cambios de ramas divergentes: Pero cuando bifurcamos el historial del proyecto con git branch , git merge permite unirlo de nuevo.

14. Anadir,editar o eliminar.

git rebase –i

La marca –i se usa para iniciar una sesión de cambio de base interactivo. El cual ofrece ventajas de cambio de base normal, pero da la oportunidad de añadir, editar o eliminar confirmaciones de marca.

15. *Seguimiento de actualizaciones*

git reflog

Realiza el seguimiento de las actualizaciones en el extremo de las ramas con el registro de referencia o reflog, que permite volver a los conjuntos de cambios, aunque no se haga referencia a ellos en ninguna rama o etiqueta.

16. *Conexiones remotas*

git remote

Es un comando para administrar conexiones remotas. En lugar de pasar la URL completa a los comandos fetch, pull y push, que permite usar un atajo significativo.

17. *Deshacer cambios*

git reset

Se utiliza para deshacer cambios en los archivos del directorio. Lo que permite limpiar o eliminar por completo los cambios que no se han enviado a un repositorio público.

18. *Deshacer instantánea confirmada.*

git revert

Si se descubre una confirmación errónea, revertirla es fácil y segura de eliminar por completo del código base.

19. *Estado del directorio*

git status

Muestra es estado del directorio en el que se está trabajando y la instantánea prepara y lo mejor es ser ejecutado con un git add y git commit para ver exactamente que se va a incluir en la próxima instantánea.

Trabajar con repositorios en GitHub (Branches, Merge, Conflicts)

Los *repositorios* son un espacio donde se almacenan y gestionan los archivos de tu proyecto. En GitHub, los repositorios pueden ser públicos o privados y proporcionan herramientas para colaborar con otros desarrolladores.

Branches(ramas) son es una versión separada de tu código que se puede desarrollar y modificar independientemente de la rama principal (usualmente llamada `main` o `master`). Crear branches te permite trabajar en nuevas características o arreglos de errores sin afectar el código en la rama principal. Puedes fusionar (merge) los cambios de un branch de vuelta a la rama principal una vez que ¡Claro! Aquí tienes una guía básica para crear branches en GitHub utilizando Git en tu terminal:

1. Clonar el repositorio:

```
git clone <url_del_repositorio>
```

2. Navegar al directorio del repositorio:

```
cd <nombre_del_repositorio>
```

3. Crear un nuevo branch: Utiliza el comando `git checkout -b` seguido del nombre que deseas darle a tu nuevo branch. Por ejemplo:

```
git checkout -b mi-nuevo-branch
```

Esto creará un nuevo branch llamado `mi-nuevo-branch` y cambiará tu entorno de trabajo para que estés en ese branch.

4. Hacer cambios y confirmarlos: utiliza los comandos `git add` y `git commit` para agregar y confirmar tus cambios.

```
git add .
```

```
git commit -m "Mensaje descriptivo de los cambios"
```

5. Empujar el branch al repositorio remoto (opcional): Si deseas compartir tu nuevo branch con otros colaboradores o trabajar en él desde otro dispositivo, puedes empujar el branch al repositorio remoto:

```
git push -u origin mi-nuevo-branch
```

Esto creará el branch en el repositorio remoto y establecer estés listo para integrarlos.

Merge(fusionar) es el proceso de combinar los cambios de un branch en otro que implica tomar los cambios de un branch (por ejemplo, una nueva función o arreglo de errores) y aplicarlos a otro branch (como la rama principal). GitHub proporciona herramientas para fusionar branches de manera fácil y segura.

Para fusionar branches en GitHub utilizando Git en tu terminal, sigue estos pasos:

1. Asegúrate de estar en el branch destino: usa el comando `git checkout` para cambiar al branch destino:

```
git checkout main
```

Reemplaza `main` con el nombre del branch destino si es diferente.

2.Antes de fusionar cambios, es una buena práctica obtener los últimos cambios del repositorio remoto para asegurarte de que tu rama esté actualizada. Puedes hacerlo usando el comando `git pull`:

```
git pull origin main
```

Esto asegurará que tengas los últimos cambios del branch `main` (o tu rama principal) antes de fusionar.

3.Ahora, puedes fusionar el branch que deseas incorporar en el branch destino. Utiliza el comando `git merge` seguido del nombre del branch que deseas fusionar. Por ejemplo, si deseas fusionar un branch llamado `mi-nuevo-branch` en el branch `main`, usa el siguiente comando:

```
git merge mi-nuevo-branch
```

Esto fusionará los cambios de `mi-nuevo-branch` en el branch `main`.

4. Si hay conflictos durante la fusión, Git te notificará y te pedirá que los resuelvas. Deberás abrir los archivos afectados, resolver los conflictos manualmente y luego confirmar los cambios.

5.Una vez que hayas resuelto los conflictos (si los hubiera), confirma los cambios fusionados utilizando el comando `git commit`. Si no hay conflictos, este paso puede omitirse.

6.Finalmente, empuja los cambios fusionados al repositorio remoto utilizando el comando `git push`:

```
git push origin main
```

Reemplaza `main` con el nombre de tu rama principal si es diferente.

Conflicts(conflictos) son los que ocurren cuando hay cambios en diferentes partes del mismo archivo en dos branches que estás intentando fusionar. GitHub te notifica si hay conflictos y te proporcionará herramientas para resolverlos y esto puede implicar revisar los cambios conflictivos, decidir qué cambios mantener, y fusionar manualmente los cambios.

Trabajar con repositorios en GitHub implica crear branches para trabajar en nuevas características o arreglos de errores, fusionar los cambios de vuelta a la rama principal cuando estén listos y resolver conflictos si surgen al fusionar branches.