3. Uso del Historial y Navegación en Git

En este apartado, exploraremos cómo utilizar el historial de Git y cómo navegar entre diferentes estados del repositorio. Para facilitar la comprensión, dividiremos las acciones según los tres estados principales de Git:

1. **Directorio de Trabajo (Working Directory)**
2. **Área de Preparación (Staging Area)**
3. **Repositorio (Commits)**

3.1. Acciones en el Directorio de Trabajo

El **Directorio de Trabajo** es donde realizas cambios en tus archivos. Aquí puedes:

- **Ver el estado actual de los archivos**:

  ```
  git status
  ```

  Este comando muestra qué archivos han sido modificados, cuáles están preparados para el commit y cuáles no están siendo rastreados por Git.

- **Ver diferencias entre el Directorio de Trabajo y el último commit**:

  ```
  git diff
  ```

  Muestra los cambios no preparados (unstaged changes) en comparación con el último commit.

- **Revertir cambios en el Directorio de Trabajo**:

  - Revertir cambios en un archivo modificado no preparado:

    ```
    git restore nombre_archivo
    ```

    Esto restaurará el archivo al estado del último commit.

- **Ver historial de cambios en un archivo específico**:

  ```
  git log -- nombre_archivo
  ```

  Lista todos los commits que afectaron a ese archivo.

- **Ver versiones anteriores de un archivo sin cambiar el estado actual**:

  ```
  git show commit_hash:nombre_archivo
  ```

Muestra el contenido de `nombre_archivo` en el commit especificado.

3.2. Acciones en el Área de Preparación (Staging Area)

El **Área de Preparación** es donde agregas cambios que deseas incluir en el próximo commit. Aquí puedes:

- **Ver los cambios preparados para el commit**:

```
git diff --staged
```

Muestra las diferencias entre los archivos en el Área de Preparación y el último commit.

- **Ver el estado de los archivos en el Área de Preparación**:

```
git status
```

Indica qué archivos están en el Área de Preparación listos para ser confirmados.

- **Quitar archivos del Área de Preparación**:

```
git restore --staged nombre_archivo
```

Mueve el archivo de vuelta al Directorio de Trabajo sin los cambios preparados.

- **Ver el historial de commits que afectaron a los archivos en el Área de Preparación**:

Aunque los archivos en el Área de Preparación aún no forman parte de un commit, puedes revisar su historial previo:

```
git log -- nombre_archivo
```

3.3. Acciones en el Repositorio (Commits)

El **Repositorio** contiene todos los commits confirmados. Aquí puedes:

- **Ver el historial completo de commits**:

```
git log
```

Lista todos los commits en orden cronológico inverso.

- **Ver un historial simplificado**:

```
git log --oneline
```

Muestra cada commit en una sola línea.

- **Ver detalles de un commit específico**:

```
git show commit_hash
```

Muestra los cambios introducidos en ese commit.

- **Comparar cambios entre commits**:

```
git diff commit_hash1 commit_hash2
```

Compara los cambios entre dos commits específicos.

- **Navegar a un commit específico**:

```
git checkout commit_hash
```

Cambia el estado del Directorio de Trabajo y el Área de Preparación al estado de ese commit.

- **Regresar a la rama principal**:

```
git checkout main
```

- **Crear una nueva rama desde un commit anterior**:

```
git branch nombre_rama commit_hash
```

- **Revertir un commit específico**:

```
git revert commit_hash
```

Crea un nuevo commit que deshace los cambios del commit especificado.

- **Restablecer el repositorio a un commit anterior**:

El comando `git reset [--mixed | --soft | --hard] <commit_hash>` restaura el repositorio a un estado anterior basado en el commit que especifiques, este comando se comporta de manera diferente dependiendo de las siguientes opciones:

- `git reset --soft <commit>`: Mueve el puntero de la rama actual al commit especificado. **No afecta el área de preparación** (staging area) ni los archivos del directorio de trabajo. Los cambios en los commits posteriores se mantienen en el área de preparación como si hubieras ejecutado git add para cada archivo.

  ```
  git reset --soft <commit_hash>
  ```

  Si usas git reset `--soft` en un commit anterior, todo el código en los commits posteriores aparecerá en el área de preparación como si se estuviera preparando para un nuevo commit.

- `git reset --mixed <commit>` (Opción predeterminada): Es la opción predeterminada si no especificas ningún argumento (`git reset`). Mueve el puntero de la rama actual al commit especificado y **resetea el área de preparación**. Los cambios en los commits posteriores se mantienen en el **directorio de trabajo**, pero se sacan del área de preparación.

  ```
  git reset --mixed <commit_hash>
  ```

Si usas `git reset --mixed` en un commit anterior, los archivos en los commits posteriores no se perderán, pero se mostrarán como cambios no preparados (es decir, estarán en el directorio de trabajo pero no en el área de preparación).

- `git reset --hard <commit>`: Mueve el puntero de la rama actual al commit especificado y **descarta** los cambios tanto en el **área de preparación** como en el **directorio de trabajo**. Los cambios que no se han confirmado se **pierden** y el directorio de trabajo vuelve a reflejar exactamente el estado del commit especificado.

  ```
  git reset --hard <commit_hash>
  ```

  Si usas git reset --hard, todos los cambios que hayas realizado después del commit especificado se perderán permanentemente, a menos que utilices git reflog para recuperarlos.

Precaución: Este comando es **destructivo. Todos los cambios y commits realizados después del commit especificado se perderán permanentemente** a menos que los recuperes con `git reflog`.

Resumen de Comandos y Acciones por Estado

Estado	Acción	Comando
Directorio de Trabajo	Ver estado de los archivos	`git status`

Estado	Acción	Comando
	Ver diferencias con el último commit	`git diff`
	Revertir cambios en archivos modificados	`git restore nombre_archivo`
	Ver historial de un archivo	`git log -- nombre_archivo`
Área de Preparación	Ver diferencias de archivos preparados	`git diff --staged`
	Quitar archivos del Área de Preparación	`git restore --staged nombre_archivo`
	Ver estado de los archivos	`git status`
Repositorio (Commits)	Ver historial completo	`git log`
	Ver historial simplificado	`git log --oneline`
	Mostrar detalles de un commit	`git show commit_hash`
	Comparar cambios entre commits	`git diff commit_hash1 commit_hash2`
	Navegar a un commit específico	`git checkout commit_hash`
	Regresar a la rama principal	`git checkout main`
	Crear rama desde un commit	`git branch nombre_rama commit_hash`
	Revertir un commit	`git revert commit_hash`
	Restablecer repositorio a un commit anterior	`git reset --hard commit_hash`

Consejos Adicionales

- **Usar alias para comandos frecuentes**:

  Puedes crear alias para simplificar comandos largos:

  ```
  git config --global alias.lg "log --oneline --graph --all --decorate"
  ```

  Ahora puedes usar `git lg` para ver un historial gráfico simplificado.

- **Explorar el historial de forma segura**:

  Cuando navegas a un commit específico, recuerda que cualquier cambio que hagas no estará asociado a ninguna rama a menos que crees una nueva rama o regreses a una existente.

- **Restaurar archivos específicos de un commit anterior**:

  Si deseas restaurar un archivo a como estaba en un commit anterior sin cambiar todo el repositorio:

  ```
  git checkout commit_hash -- nombre_archivo
  ```