Comandos Principales de Configuración:

1. Establecer el Nombre de Usuario:

```
git config --global user.name "Tu Nombre"
```

1.1.Chequear que el nombre se haya guardado bien

```
git config user.name
```

2. Establecer el Correo Electrónico del Usuario:

```
git config --global user.email "tuemail@dominio.com"
```

3. Chequear que el email se haya guardado bien

```
git config user.email
```

4. Comando para mostrar configuracion actual de Git:

```
git config --list
```

5. Comando para obtener ayuda de Git

```
git help <comando>
```

Principales comandos para trabajar con proyectos:

1. **Iniciar un repositorio local**

```
git init mi_repositorio
```

2. **Chequear el estado del repositorio**

```
git status
```

-Observamos el mensaje **untracked files** (archivos sin seguimiento)
estado: Working directory - **Directorio de trabajo**

3. **Si hay archivos que no deseamos agregar usamos un archivo *.*gitignore** para especificar los mismos.**

```
*concidencia comodin
/ se usa para ignorar las rutas relativas al archivo .gitignore.
# es usado para agregar comentarios
```

4. **Agregamos los archivos al staging Area**

```
git add
```

5. **Verificamos que los archivos efectivamente se hayan agregados**

```
git status
```

Observamos por consola el mensaje Changes to be commited, y el archivo pintado de verde

6. **Confirmamos cambios y enviamos los archivos a nuestro repositorio local**

```
git commit -m "mi comentario¨
```

agregamos un comentario descriptivo de lo que venimos haciendo.

7. **Ver el historial de commits**

```
git log
```

Este comando mostrará una lista de commits en el historial del repositorio, incluyendo:
con la tecla "q" matamos el proceso y continuamos usando la terminal.

- El hash SHA-1 del commit.
- El autor del commit.
- La fecha del commit.
- El mensaje del commit.

1. **Comando para eliminar un repositorio.(🔪Mucho cuidado con este comando)**

```
git rm -rf nombre-del-repositorio
```

Trabajo con ramas.

---

1. **Crear una nueva rama.**

```
git branch nombre-de-la-rama
```

Crea una nueva rama llamada nombre-de-la-rama basada en la rama actual.

2. **Cambiar a una rama diferente**

```
git checkout nombre-de-la-rama
```

O usando la nueva sintaxis con **switch**

```
git switch nombre-de-la-rama
```

Cambia a la rama nombre-de-la-rama.

3. **Crear y cambiar a una nueva Rama:**

```
git checkout -b nombre-de-la-rama
```

Crea una nueva rama y cambia a ella inmediatamente.

4. **Listar todas las ramas:**

```
git branch
```

Muestra todas las ramas locales. La rama actual se indica con un asterisco `*`

5. **Eliminar una Rama:**

```
git branch -d nombre-de-la-rama
```

Elimina la rama nombre-de-la-rama si ha sido completamente integrada en la rama actual.

Para forzar la eliminación de una rama (incluso si no está completamente integrada):

```
git branch -D nombre-de-la-rama
```

6. **Renombrar una rama:**

```
git branch -m nuevo-nombre
```

Renombra la rama actual a nuevo-nombre.

7. **Combinar (Merge) una Rama en la Rama Actual:**

```
git merge nombre-de-la-rama
```

8. **Rebase de una Rama:**

```
git rebase nombre-de-la-rama
```

Rebase la rama actual sobre la rama nombre-de-la-rama.

Comandos avanzados para Ramas en Git.

---

1. **Mostrar la rama actual:**

```
git symbolic-ref --short HEAD
```

Muestra el nombre de la rama actual.

2. **Mostrar el historial de Commits por rama:**

```
git log nombre-de-la-rama
```

Muestra el historial de commits de la rama nombre-de-la-rama.

3. **Comparar dos ramas:**

```
git diff rama1..rama2
```

Muestra las diferencias entre rama1 y rama2.

4. **Listar ramas remotas:**

```
git branch -r
```

Muestra todas las ramas remotas.

5. **Listar todas las ramas (Locales y remotas)**

```
git branch -a
```

6. **Eliminar una rama remota.**

```
git push origin --delete nombre-de-la-rama
```

Elimina la rama nombre-de-la-rama del repositorio remoto.

7. **establecer la rama por defecto en remoto**

```
git push -u origin nombre-de-la-rama
```

Establece nombre-de-la-rama como la rama por defecto en el repositorio remoto y configura el seguimiento.

Revertir Cambios.

---

1. **Revertir un commit Confirmado (Usando git reset)** Estos son cambios que han sido añadidos al historial del repositorio mediante un commit.

git reset --hard

Este comando deshace todos los cambios realizados desde el commit especificado y restablece el estado del árbol de trabajo al commit especificado.
Usar **--hard** elimina permanentemente todos los cambios no confirmados, así que asegúrate de que realmente quieres deshacer todos los cambios.

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

git reset --soft Mueve el puntero de HEAD al commit especificado, manteniendo los cambios en el área de preparación.

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

1.3. **git reset --mixed** Mueve el puntero de HEAD al commit especificado, manteniendo los cambios en el árbol de trabajo.

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

2. **Revertir un commit Confirmado (Usando git revert)**

```
git revert <commit_hash>
```

Este comando crea un nuevo commit que deshace los cambios del commit especificado. Es útil porque no altera el historial del commit.

3. **Restablecer al último commit confirmado**

```
git reset --hard HEAD
```

Si quieres simplemente deshacer todos los cambios no confirmados y volver al último commit confirmado:

4. **Reestablecer cambios No Confirmados (Uncommitted Changes)** (Estos son cambios que has realizado en los archivos de tu repositorio, pero que aún no has confirmado con un commit.)
   Cambios en el Árbol de Trabajo (Working Directory)
   Cambios Añadidos al Área de Preparación (Staging Area)

4.1 **git reset**

```
git reset
```

Si los cambios han sido añadidos (staged) pero no confirmados, puedes utilizar git reset para deshacer esta acción.

4.2 **git checkout**

```
git checkout -- <archivo>
```

Si los cambios no han sido añadidos al área de preparación, puedes usar git checkout para restablecerlos al último commit.