Explicación de los comandos del documento

Inicialización y configuración

- **git init**: Inicializa un nuevo repositorio Git en el directorio actual.
- **git config --global user.email "xxx@gmail.com"**: Configura el correo electrónico global del usuario para los commits.
- **git config --global user.name "xxx"**: Configura el nombre global del usuario para los commits.

Añadir archivos al área de preparación (staging area)

- **git add .**: Añade todos los archivos modificados o nuevos al área de preparación.
- **git add nombreDelArchivo**: Añade un archivo específico al área de preparación.

Ver el estado del repositorio

- **git status -s**: Muestra un resumen corto del estado del repositorio (archivos modificados, añadidos, etc.).

Commits

- **git commit -m "xxx"**: Crea un commit con un mensaje descriptivo ("xxx").
- **git log --oneline**: Muestra un historial compacto de los commits, mostrando solo la primera línea de cada mensaje.

Resetear cambios

- **git reset --hard**: Revierte todos los cambios locales al último commit.
- **git reset --hard 012a3b4**: Revierte todos los cambios locales a un commit específico identificado por su hash (**012a3b4**).

Trabajar con GitHub

Nuevo repositorio:

1. **git remote add origin https://github.com/XXXX.git**: Vincula el repositorio local con uno remoto en GitHub.
2. **git push -u origin master**: Sube el contenido del repositorio local a la rama principal (**master**) del remoto y establece seguimiento.

Subir cambios:

- **git push**: Sube los cambios locales al repositorio remoto.
- **git push -u origin master**: Igual que el anterior, pero establece la rama remota como predeterminada para futuros pushes.

Descargar cambios:

- **git pull**: Descarga y fusiona los cambios del repositorio remoto con el local.

Tags

- **git tag nombre -m "xxx"**: Crea una etiqueta (tag) con un nombre y un mensaje descriptivo.
- **git push --tags**: Sube las etiquetas al repositorio remoto.

Clonar un repositorio

- **git clone https://github.com/xxxt.git**: Clona un repositorio remoto en tu máquina local.

Ramas

- **git branch nombreRama**: Crea una nueva rama llamada **nombreRama**.
- **git branch**: Lista todas las ramas existentes en el repositorio.
- **git checkout nombreRama**: Cambia a una rama específica.
- **git branch -d nombreRama**: Elimina una rama local llamada **nombreRama**.

Fusionar ramas

- **git merge ramaAFusionar**: Fusiona la rama especificada (**ramaAFusionar**) con la rama actual.

Comandos útiles adicionales

Aquí tienes algunos comandos adicionales que pueden ser muy útiles:

Ignorar archivos

- Crear un archivo **.gitignore** para excluir ciertos archivos o carpetas del control de versiones. Ejemplo:
  text

```
# Ignorar archivos temporales
*.log
/node_modules
```

Deshacer cambios antes de hacer commit

- **git restore archivo.txt**: Revierte los cambios no guardados en un archivo.
- **git restore --staged archivo.txt**: Quita un archivo del área de preparación sin eliminar los cambios.

Ver diferencias

- **git diff**: Muestra las diferencias entre los archivos modificados y el último commit.
- **git diff --staged**: Muestra las diferencias entre los archivos preparados para commit y el último commit.

Trabajo con ramas remotas

- **git fetch**: Descarga información sobre las ramas remotas sin fusionarlas.
- **git branch -r**: Lista las ramas remotas disponibles.

Trucos y consejos para lanzarte con Git/GitHub

1. **Usa mensajes descriptivos en tus commits**
   Los mensajes deben explicar claramente qué cambios hiciste. Ejemplo:
   text
   ```
   git commit -m "Corrige error en la función de login"
   ```
2. **Mantén tu repositorio limpio**
   Usa **.gitignore** para evitar subir archivos innecesarios como logs, dependencias o configuraciones locales.
3. **Crea ramas para nuevas funcionalidades**
   Trabaja en una nueva funcionalidad o corrección de errores en una rama separada:
   text
   ```
   git checkout -b nueva-funcionalidad
   ```
4. **Sincroniza regularmente**
   Antes de empezar a trabajar, asegúrate de tener la última versión del código:
   text
   ```
   git pull origin master
   ```
5. **Revisa antes de fusionar**
   Antes de hacer un merge, revisa posibles conflictos usando:
   text
   ```
   git diff ramaActual ramaAFusionar
   ```
6. **Usa GitHub Desktop o extensiones**
   Si prefieres interfaces gráficas, puedes usar herramientas como GitHub Desktop o extensiones como GitLens para Visual Studio Code.
7. **Documenta tu proyecto**
   Incluye un archivo README.md bien estructurado para explicar tu proyecto. Puedes usar Markdown para formatearlo fácilmente.
8. **Prueba comandos interactivos**
   Usa comandos como:
   text
   ```
   git rebase -i HEAD~3
   ```
   Para reordenar, combinar o editar commits recientes (avanzado).