



Manual de Uso Profesional de Git

Este manual proporciona una guía detallada para gestionar repositorios Git con buenas prácticas, optimizando flujos de trabajo y asegurando la correcta organización del código.



Clonar un repositorio

Para obtener una copia local de un repositorio remoto:

```
git clone <URL_DEL_REPOSITORIO>
```

Ejemplo:

```
git clone https://github.com/user/repository.git
```



Ver el estado del repositorio

Para conocer el estado actual del repositorio, archivos modificados o no rastreados:

```
git status
```



Listar las ramas disponibles

Para ver todas las ramas locales:

```
git branch
```

Para ver todas las ramas remotas:

```
git branch -r
```



Cambiar de rama con **switch**

En lugar de **checkout**, Git recomienda **switch** para cambiar de rama:

```
git switch <nombre_rama>
```

Ejemplo:

```
git switch main
```

Crear una nueva rama de forma correcta

Para evitar problemas con datos no actualizados, sigue estos pasos:

```
git switch main
git pull origin main # Asegurar que la rama esté actualizada
git switch -c feature/<nombre_de_la_rama>
```

Ejemplo:

```
git switch -c feature/user-login
```








Nota: Siempre crea nuevas ramas desde `main`, a menos que necesites copiar cambios específicos.

Buenas prácticas con Conventional Commits

El formato recomendado es [Conventional Commits](#), usando la estructura:

```
git commit -m "<tipo>(<área>): <mensaje corto y claro>"
```

Tipos más comunes:

-  **feat**: Nueva funcionalidad
-  **fix**: Corrección de errores
-  **docs**: Cambios en documentación
-  **style**: Cambios de formato (sin afectar código)
-  **refactor**: Reestructuración del código sin cambiar su funcionalidad
-  **perf**: Mejoras de rendimiento
-  **test**: Agregar o modificar pruebas

Ejemplo de commit bien estructurado:

```
git commit -m "feat(auth): add user login with JWT"
```

Subir cambios al repositorio remoto

```
git push origin feature/<nombre_de_la_rama>
```

Ejemplo:

```
git push origin feature/user-login
```

Mantener el repositorio actualizado

Antes de iniciar nuevas tareas, es fundamental actualizar la base del código:

```
git pull origin main
```

Si trabajas en una rama específica y necesitas actualizarla con **main**:

```
git switch feature/<nombre_de_la_rama>  
git merge main
```

Reglas clave para manejar ramas

- ☒ Siempre crea nuevas ramas desde **main** (salvo excepciones justificadas).
- ☒ Mantén la rama **main** siempre limpia y funcional.
- ☒ Usa nombres claros en las ramas, como **feature/login** o **fix/bug-123**.
- ☒ Sincroniza tu rama con **main** regularmente para evitar conflictos.
- ☒ Usa **git rebase** en lugar de **merge** cuando sea necesario mantener un historial limpio.

Siguiendo estos principios, tu flujo de trabajo en Git será más eficiente, organizado y profesional. 🚀