💻 Git & GitHub Cheatsheet – Using Git Locally

> **Curso:** Introduction to Git and GitHub
> **Módulo:** Using Git Locally
> **Tema:** Instalación y manejo de Git en tu máquina local

📓 Introducción

Git es un **sistema de control de versiones distribuido** que permite:

- Registrar cambios en archivos
- Colaborar con otros desarrolladores
- Mantener historial de un proyecto

GitHub es una **plataforma remota** para alojar repositorios Git y colaborar en línea.

🔷 Instalación de Git

- **Linux (Debian/Ubuntu):**

```
sudo apt update
sudo apt install git
```

- **macOS:**

```
brew install git
```

- **Windows:** Descargar desde git-scm.com y seguir instalador.

- Verificar versión:

```
git --version
```

🔷 Configuración Inicial

```
git config --global user.name "Tu Nombre"
git config --global user.email "tuemail@ejemplo.com"
git config --global core.editor nano   # Cambiar editor
git config --list                      # Ver configuración
```

Crear y Clonar Repositorios

- Crear un repositorio local:

```
mkdir proyecto
cd proyecto
git init
```

- Clonar un repositorio remoto:

```
git clone https://github.com/usuario/repositorio.git
```

Estado del Repositorio

```
git status          # Estado de archivos
git log             # Historial de commits
git log --oneline   # Historial resumido
git diff            # Cambios sin stage
```

Añadir y Confirmar Cambios

```
git add archivo.txt       # Añadir archivo al stage
git add .                 # Añadir todos los cambios
git commit -m "Mensaje"   # Confirmar cambios
```

- **Buen hábito:** Mensajes claros y descriptivos
- **Atajo:** `git commit -am "Mensaje"` → añade y confirma cambios modificados (no nuevos archivos)

Ramas (Branches)

```
git branch                # Ver ramas locales
git branch nueva-rama     # Crear nueva rama
git checkout nueva-rama   # Cambiar de rama
git checkout -b rama2     # Crear y cambiar de rama en un paso
```

- Ramas permiten trabajar en **características nuevas** sin afectar la rama principal (`main` o `master`)

Deshacer Cambios

```
git checkout -- archivo.txt      # Revertir cambios no confirmados
git reset HEAD archivo.txt       # Quitar del stage
git revert <commit_id>           # Deshacer commit con nuevo commit
git reset --hard <commit_id>    # Volver al estado exacto de un commit
```

> ⚠ Usar `--hard` con precaución, ya que elimina cambios locales sin posibilidad de recuperación.

Tips y Buenas Prácticas

- Hacer commits frecuentes y pequeños
- Mantener rama principal (`main`) estable
- Usar mensajes de commit claros: “Agregar login de usuario” en lugar de “Cambios”
- Revisar estado con `git status` antes de añadir o confirmar cambios

Referencias oficiales

- Pro Git Book
- Git Reference – Git SCM
- GitHub Docs – Git Basics

```
---
```