Introducción a Git y GitHub

Un material de consulta para aprender a versionar y colaborar en tus proyectos.

Un sistema de control de versiones te permite:

- Guardar *snapshots* de tu proyecto.
- Volver a cualquier versión anterior si algo sale mal.
- Colaborar con otras personas sin pisarse el trabajo.

¿Qué es Git?

Git es el **software** de control de versiones más popular del mundo. Es una herramienta que se instala en tu computadora y funciona desde la terminal. Es rápido, potente y el estándar de la industria.

¿Qué es GitHub?

GitHub es una **plataforma web** que utiliza Git. Es como una red social para alojar tus proyectos de código. Te ofrece:

- Un lugar seguro en la nube para guardar tus repositorios (proyectos).
- Herramientas para colaborar en equipo, revisar código y gestionar proyectos.
- Un portafolio público para mostrar tu trabajo.

> **Piénsalo así:** Git es el programa para controlar las versiones en tu PC. GitHub es el sitio web donde guardas y compartes esos proyectos con el mundo.

Git

Glosario

- **Repositorio:** Es la carpeta que contiene todo tu proyecto y su historial de cambios (una base de datos oculta en una carpeta `.git`).
- **Commit:** *Un snapshot* o punto de guardado permanente en la historia de tu proyecto. Cada commit tiene un identificador único y un mensaje que describe los cambios.
- **Stage:** Una zona intermedia donde agrupas los cambios que quieres incluir en tu próximo commit. Es como el carrito de compras donde añades cosas antes de pagar.
- **Push:** El acto de *enviar* tus commits locales (que están en tu PC) al repositorio remoto (GitHub, por ejemplo).
- **Pull:** El acto de *traer* los cambios del repositorio remoto a tu máquina local para mantenerte actualizado.

- **Branch (rama):** Una línea de desarrollo independiente. Permite trabajar en nuevas ideas o funcionalidades sin afectar la rama principal (usualmente llamada `main` o `master`).
- **Merge:** El proceso de combinar los cambios de una rama en otra.

Comandos de Git

La terminal es la herramienta más poderosa para usar Git. Todos los clientes gráficos ejecutan estos comandos por debajo.

Paso 0: Configuración inicial (sólo una vez)

Git necesita saber quién eres para firmar tus commits.

Sustituye con tu nombre y correo (el mismo de GitHub).

```
git config --global user.name "Tu Nombre"
git config --global user.email "tu_correo@ejemplo.com"
```

Paso 1: Crear un repositorio local

Crea una carpeta para tu proyecto y entra en ella

```
mkdir mi-proyecto-genial
cd mi-proyecto-genial
```

Inicializa Git, esto crea la carpeta oculta .git

```
git init
```

Paso 2: Realizar el primer commit

Crea un archivo README.md (es una buena práctica)

```
echo "# Mi proyecto es genial" > README.md
```

Revisa el estado. Git te dirá que hay un archivo *untracked*

```
git status
```

Añade el archivo al stage

```
git add README.md
```

ó usa el siguiente comando para añadir al stage todos los archivos nuevos y modificados

```
git add .
```

Confirma los cambios con un mensaje descriptivo

```
git commit -m "Creación inicial del proyecto con README"
```

Paso 3: Conectar tu repositorio local con GitHub

Ve a GitHub.com y crea un nuevo repositorio.

Importante: no lo inicialices con ningún archivo (README, license, etc.), ya que ya lo hemos hecho localmente.

Copia la url que te proporciona GitHub y enlaza tu repositorio local al remoto; 'origin' es el nombre estándar.

```
git remote add origin URL_DE_TU_REPO_DE_GITHUB
```

(Opcional) Verifica que se añadió correctamente

```
git remote -v
```

Paso 4: Subir tu código a GitHub (push)
Envía tus commits a la rama 'main' de 'origin'. La bandera -u establece la conexión para futuros pushes.

```
git push -u origin main
```

Nota: Si la rama se llama 'master', usa 'master' en lugar de 'main'.

Y ¡listo! tu código ahora está en GitHub. Ahora, tu flujo diario de trabajo sería

1. Modificas tus archivos
2. Añades los cambios: `git add .`
3. Confirmas los cambios: `git commit -m "Un mensaje que describa lo que hiciste"`
4. Subes los cambios: `git push`

Git con interfaz gráfica (GUI)

Las GUIs nos ayudan a visualizar el historial y los cambios. Usaremos **GitHub Desktop** como ejemplo, ya que es muy intuitivo.

Paso 1: Clonar un repositorio existente
"Clonar" significa descargar una copia de un repositorio de GitHub a tu PC, incluyendo todo su historial.

1. Abre GitHub Desktop
2. Ve a File > Clone Repository...
3. Selecciona el repositorio desde tu cuenta de GitHub o pega la URL
4. Elige la carpeta local donde quieres guardarlo y haz clic en Clone

Paso 2: Hacer cambios y commit

1. Abre la carpeta del proyecto y modifica o crea algún archivo
2. Vuelve a GitHub Desktop. En la pestaña Changes, verás una lista de todos los archivos modificados

   Las líneas en rojo son las que eliminaste

   Las líneas en verde son las que añadiste
3. Escribe un mensaje de commit en el cuadro de texto de la esquina inferior izquierda
4. Haz clic en el botón 'Commit to main'. ¡Listo!

Paso 3: Subir cambios (push)
Tras hacer un commit, GitHub Desktop te mostrará un botón en la parte superior 'Push origin'. Púlsalo para enviar tus cambios a GitHub.

Paso 4: Ramas y Pull Requests (para flujo colaborativo)
Las ramas te permiten trabajar en algo nuevo sin arriesgar la versión estable de tu proyecto.

1. Crear una rama: En GitHub Desktop, haz clic en la pestaña ‘Current Branch’ y luego en ‘New Branch’. Dale un nombre descriptivo.
2. Trabajar en la rama: Haz commits en esta nueva rama como lo harías normalmente. Estos cambios solo existen en esta rama.
3. Publicar la rama: Cuando estés listo, haz clic en ‘Publish branch’.
4. Crear un Pull Request (PR): Después de publicar, GitHub Desktop te mostrará un botón ‘Create Pull Request’. Esto te llevará a la web de GitHub. Un PR es una “solicitud para fusionar” tus cambios en la rama principal. Es el momento de que otros revisen tu trabajo.
5. Fusionar (Merge): Una vez que el PR es aprobado, haces clic en el botón ‘Merge pull request’ en GitHub. Listo, tus cambios ahora son parte del proyecto principal.

Resumen de comandos

Comando	**Descripción**
git init	Inicializa un repositorio en la carpeta actual
git status	Muestra el estado de tus archivos (modificados, en stage, etc.)
git add [archivo]	Añade un archivo al área de preparación (stage)
git commit -m [descripción]	Crea un punto de guardado (commit) con los archivos del stage y les asigna una descripción
git push	Sube tus commits locales al repositorio remoto (GitHub)
git fetch	Descarga los cambios del remoto sin integrarlos; te permite revisar antes de fusionar
git merge [rama]	Combina los cambios de otra rama en tu rama actual
git pull	Descarga los últimos cambios del repositorio remoto y los fusiona automáticamente en tu rama actual (es un fetch + merge)
git clone [URL]	Crea una copia local de un repositorio remoto