**Paso 1: Working Directory (*Directorio de trabajo*)**

Antes de empezar a registrar cambios, necesitamos inicializar un proyecto para que Git comience a rastrear nuestros archivos.

**Git Init:** Esto crea una carpeta oculta llamada .git, que es donde se guardará toda la información sobre el historial de los proyectos. Este paso **convierte cualquier carpeta** en un repositorio de Git. A partir de ahí, puedes comenzar a trabajar con control de versiones localmente.

**Git Status:** Este comando te muestra en qué estado se encuentran tus archivos dentro del proyecto.

Es nuestra herramienta principal para **verificar qué está pasando** en el repositorio en cualquier momento. Con este comando puedes saber:

* Qué archivos cambiaron.
* Cuáles están listos para guardar.
* Si hay archivos nuevos sin seguimiento (untracked).

**Paso 2: Staging Area (*Área de preparación*)**

Acá vamos a decidir los archivos que vamos a incluir en el próximo

commit. Es una bandeja donde vamos a colocar solo lo que vamos a guardar.

**Mover un archivo al staging:** git add práctica\_semana\_1.txt

**Mover todos archivo al staging:** git add .

**Paso 3: Local Repository (Repositorio local)**

En este paso, vamos a registrar en el historial del repositorio los cambios de los archivos que preparamos en el paso anterior.

Para ello vamos a realizar un commit, lo que captura el estado actual del repositorio y crea un identificador único junto con un mensaje descriptivo que agreguemos.

Esto te permite:

* Recuperar versiones anteriores del código.
* Saber quién cambió qué.
* Dejar mensajes que expliquen por qué se hicieron esos cambios.

**Crear un commit con mensaje:** git commit -m "Agrega ejercicios de práctica de semana 1 a 3"

**Publicar repositorio para compartir el código**

Para que tu trabajo en Git esté respaldado y actualizado tanto en tu máquina como en la nube, necesitas mantener sincronizados el repositorio local y el remoto.

Diagrama

El contenido generado por IA puede ser incorrecto.

**Sincronizar desde local hacia remoto**

Para lograrlo, cuando quieras publicar una nueva versión de tu código en GitHubo GitLaby ya hiciste los commits puedes ejecutar el siguientecomando: **git push origin nombre-de-la-rama # Cualquier nombre que quieras poner a la rama**

Esto sube los cambios de tu rama local al repositorio remoto.

**Sincronizar desde remoto hacia local**

Antes de empezar a trabajar, es buena práctica traer los cambios más recientes que hayan hecho otros. Para ello vamos a sincronizar los cambios que estén en el repositorio remoto con el siguiente comando: **git pull origin nombre-de-la-rama**

Esto evita conflictos y asegura que trabajas con la versión más actual del proyecto.

**¿Qué es un branch?**

Un branch(*rama*) es una **línea paralela de trabajo** que parte de una versión del código y permite hacer cambios sin alterar el original. Cuando los cambios están listos, podemos unir esa rama con otra mediante un proceso llamado merge.

Imagen que contiene competencia de atletismo

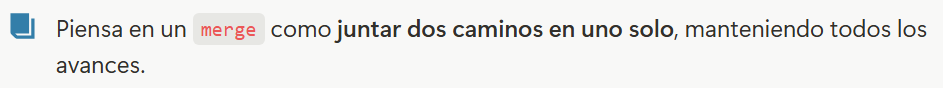
El contenido generado por IA puede ser incorrecto.

Piensa en las ramas como caminos alternativos donde puedes experimentar, construir nuevas ideas o corregir errores. Luego puedes decidir si esos cambios se integran al camino principal (main) o no.

* **git checkout -b nueva-rama** - Crear nueva rama y cambiar a ella
* **git checkout nombre-de-la-rama** - Cambiar de rama
* **git branch -** Lista de ramas del repositorio

**¿Cómo hacer un merge? 🛠️**

Una vez que terminas de trabajar en una **branch**, puedes **unir sus cambios** con otra rama . Este proceso se llama **merge**.



Para hacer un mergedesde la terminal, primero debes estar **ubicado en la rama que va a recibir los cambios** (por ejemplo, main), y luego ejecutar el comando git merge.

* **git checkout main # Nos ubicamos en la rama principal**
* **git merge nueva-rama # Traemos los cambios desde 'nueva-rama'**

Este proceso **no borra** la branchoriginal (nueva-rama). Solo une sus cambios a la rama actual.

**Pull Request desde GitHub**

Cuando trabajamos en equipo, lo más común es hacer el merge atraves de una Pull Request. Esto permite revisar, discutir y aprobar los cambios antes de integrarlos. Para lograrlo puedes seguir estos pasos:

1. Ve a la pestaña Pull requests en tu repositorio.
2. Haz clic en **New pull request**.
3. Selecciona la branch de origen (la que tiene los cambios) y la de destino.
4. Escribe un título y descripción clara.
5. Solicita revisión de otros colaboradores.
6. Haz clic en **Merge pull request** cuando todo esté listo.

**Flujo común de trabajo (Git Flow)**

Git Flow es un flujo de trabajo estandarizado que define **cómo organizar las ramas** en un proyecto. Es utilizado para mantener procesos de desarrollo ordenados, seguros y colaborativos.

En Lyfter, utilizamos una versión **simplificada** este modelo. Aprender este flujo te prepara para entornos profesionales reales. 💼

**Estructura principal de ramas main (o master)**

La rama main representa el **código en producción**. Solo contiene versiones **estables y verificadas** del proyecto. Es como el historial oficial de entregas.

* Cada commit en main es una versión que ya fue probada y lanzada.
* No se desarrolla directamente en esta rama.
* Cada "punto" es un commit (*una versión estable*).

**develop**

La rama develop se crea a partir de main y sirve como el **espacio de trabajo principal** para desarrolladores. Aquí se integran todas las nuevas funcionalidades antes de que estén listas para producción.

* Aquí se prueba todo antes de liberar una nueva versión.
* Es como un "*taller*" donde se ensamblan los cambios.
* La rama develop nace desde main.

feature

Para trabajar en una nueva funcionalidad, como **crear un botón**, se crea una rama feature/nombre desde develop. Esto permite trabajar de forma aislada.

* Cuando el botón está listo y probado, se fusiona de vuelta a develop.
* Así no se interrumpe el trabajo de otros ni se rompe el proyecto.
* Se crea la rama feature/boton desde develop.
* Se hacen varios commits (diseño, lógica, prueba).
* Al final, se hace un merge a develop.

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

¿Cómo fluye el trabajo?

1. Se crea una rama feature desde develop para trabajar en algo nuevo.
2. Al finalizar la tarea, esa rama se **fusiona** (merge) de nuevo con develop.
3. Cuando develop ha acumulado suficientes funcionalidades, se hace merge a main para una nueva versión en producción

Gráfico, Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.