

# Git y GitHub

# Git maximiza la eficiencia del proceso de desarrollo de *software*

Participar en un proyecto de *software* desafiante junto a un equipo de desarrolladores altamente capacitados **requiere herramientas efectivas** para que todos puedan trabajar en el mismo código fuente.

Git es una herramienta esencial de **control de versiones**. Permite rastrear cambios, colaborar de manera eficiente y mantener la integridad del código.



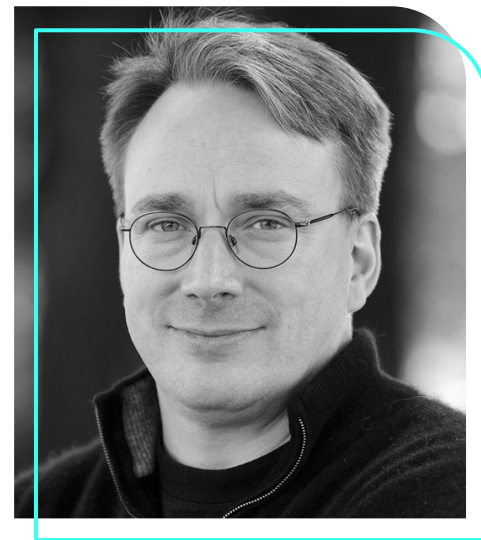
# ¿Qué es el control de versiones?

- También se conoce como **Version Control System (VCS)**.
- Facilita la **integración de cambios** realizados por varios miembros del equipo.
- Facilita la **recuperación de versiones anteriores** y el **trabajo colaborativo**.
- **Rastrea cambios** en archivos: permite ver quién realizó cambios y cuándo.
- Ayuda a mantener la **integridad** del proyecto y a evitar conflictos.



# ¿Qué es Git?

- Git es un **VCS de código abierto**.
- Creado por Linus Torvalds, autor principal de Linux.
- Rápido, versátil, escalable y gratuito.
- Permite la **creación de ramas** para el desarrollo experimental.
- Ofrece **etiquetado de versiones** para marcar hitos importantes.
- Es un **sistema de control de versiones distribuido**: cada usuario tiene una copia completa del repositorio en su máquina local.



Linus Torvalds

## Breve reseña histórica

Anteriormente, existían otros sistemas como **CVS o Subversion (SVN)**. Pero la revolución en la gestión de versiones de código comienza con la creación de **Git** por Linus Torvalds, como se

mencionó en el *slide* anterior, en **2005**. A partir de ahí, su uso se fue propagando en la industria.



# Control de versiones distribuido

- **Antes de Git**, el control de versiones con CVS o SVN usaba un **servidor centralizado** para almacenar el historial de un proyecto. Esto implicaba un único punto de error potencial.
- **Git:**
  - Es un **sistema distribuido**, lo que significa que el historial completo se almacena **tanto en el cliente como en el servidor**.
  - Permite **editar archivos sin conexión de red, protegerlos localmente y sincronizarlos** con el servidor posteriormente.

Incluso si el servidor falla, se conserva una copia local del proyecto, lo que garantiza la **disponibilidad continua de los datos**.



# Git y GitHub: Diferencias clave

## No confundir Git con GitHub

Mientras que Git es un sistema de control de versiones distribuido (DVCS), GitHub es una **plataforma en la nube** basada en Git.

**Github es una implementación de Git.**

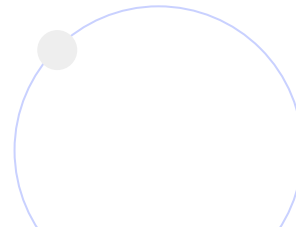
## Importancia de GitHub:

- Plataforma líder para alojar proyectos de código abierto.
- Facilita la colaboración entre desarrolladores.
- Principal lugar donde se alojan y colaboran en proyectos de código actualmente.



## Características de Github

- **Colaboración:** Mediante *incidencias* y *solicitudes de incorporación de cambio* los usuarios pueden reportar problemas, discutir soluciones y proponer cambios. Estas acciones fomentan la participación y el desarrollo colaborativo.
- **Comunicación:** Con *notificaciones*, *etiquetas* y *acciones* proporcionan funciones para mantener a los colaboradores informados sobre actividades importantes y automatizar acciones repetitivas para una eficiencia mejorada.
- **Organización:** Se pueden crear proyectos y bifurcaciones que facilitan la creación de copias independientes para experimentar o contribuir sin afectar la versión principal.
- **Visualización:** Ofrece herramientas para organizar y visualizar el progreso del trabajo en un tablero de proyecto dedicado.





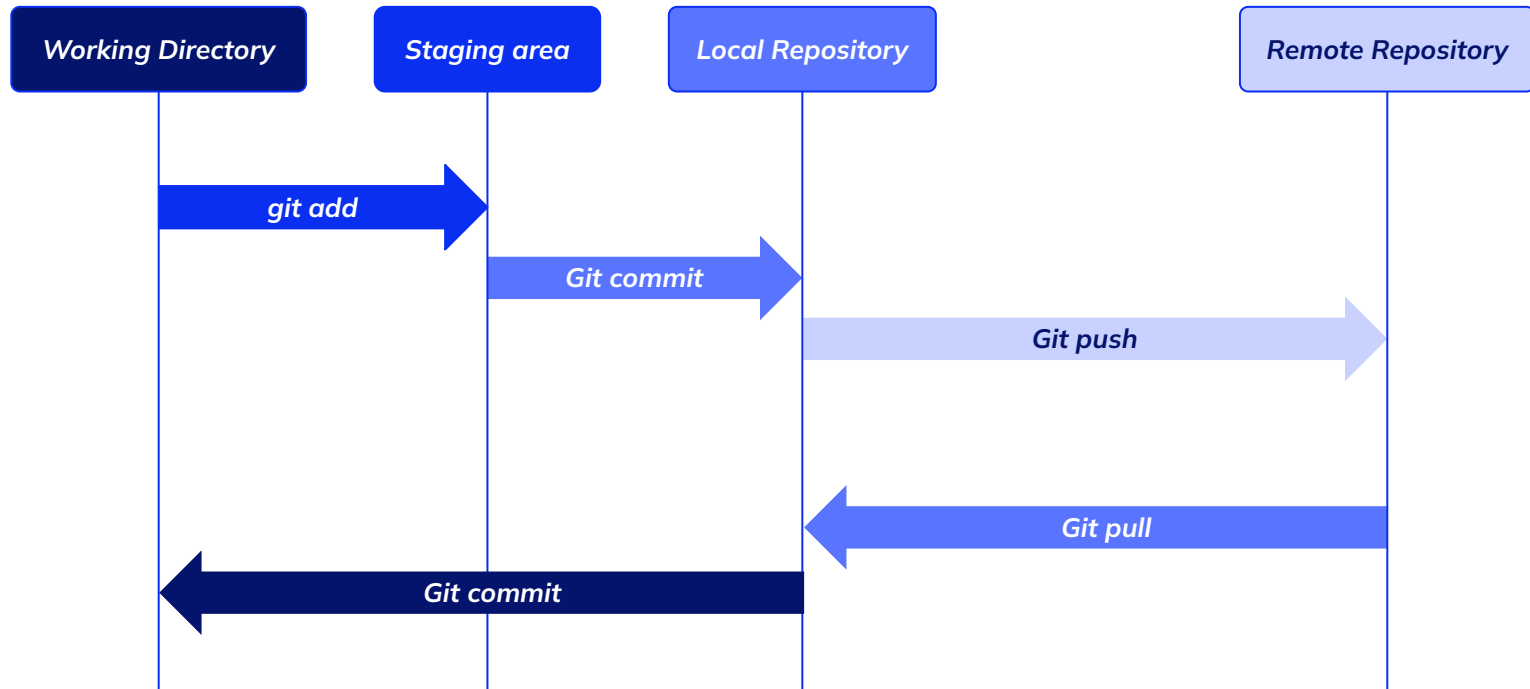
# Guía de conceptos básicos

<b>Directorio de trabajo</b> ( <i>Working Directory</i> )	Es donde se encuentran los archivos y carpetas del proyecto en la computadora local.
<b>Área de preparación</b> ( <i>Staging area</i> )	Es donde se colocan los archivos modificados antes de confirmarlos en el repositorio local.
<b>Repositorio local</b> ( <i>Local Repository</i> )	Es donde se almacenan las versiones controladas del proyecto en el dispositivo de almacenamiento local.
<b>Repositorio remoto</b> ( <i>Remote Repository</i> )	Es una copia del repositorio local alojada en un servidor remoto, permitiendo la colaboración y el respaldo del proyecto en línea.
<b>Inicializar un repo o hacer un <i>init</i></b>	Esto significa crear un repositorio local lo que implica crear una carpeta oculta <i>.git</i> con los metadatos del repositorio.

(Continúa en el siguiente *slide*)

Continuación:

<b>Clonar un <i>repo</i></b>	Significa descargar un repositorio de GitHub, localmente, a partir de la URL del repositorio que termina con <i>.git</i> (ejemplo: <a href="https://github.com/educacionit/repo.git">https://github.com/educacionit/repo.git</a> ).
<b>Hacer un <i>add</i></b>	Agregar un archivo al repositorio local para versionarlo.
<b>Hacer un <i>commit</i></b>	Crear una nueva versión del proyecto con un conjunto de cambios realizados. Es importante incluir un mensaje significativo de los cambios realizados.
<b>Rama principal</b>	El espacio de trabajo principal, donde se encuentra la versión primaria y estable del proyecto, generalmente llamado " <i>master</i> " o " <i>main</i> ".
<b>Hacer un <i>pull</i></b>	Traer las últimas actualizaciones desde el repositorio remoto al local, para asegurarse de trabajar con la versión más reciente del proyecto.
<b>Hacer un <i>push</i></b>	Enviar los cambios locales al repositorio remoto. De esta manera, se actualiza el historial de versiones y se permite que otros colaboradores accedan a las modificaciones.



# Herramientas para usar Git

- Existen varias herramientas disponibles para Git, como [GitHub Desktop](#) y [GitKraken](#).
- Algunos **editores de programación**, como Visual Studio Code, también tienen interfaces para Git.

Cada una tiene sus propias **características y limitaciones**, sin implementar toda la funcionalidad de Git.

Se recomienda trabajar siempre con la interfaz de línea de comandos de Git:

- Es consistente en todos los sistemas operativos y permite aprovechar toda la capacidad de Git.
- A veces, los desarrolladores que solo utilizan una interfaz visual se enfrentan a mensajes de error que **solo pueden resolver volviendo a la línea de comandos**.



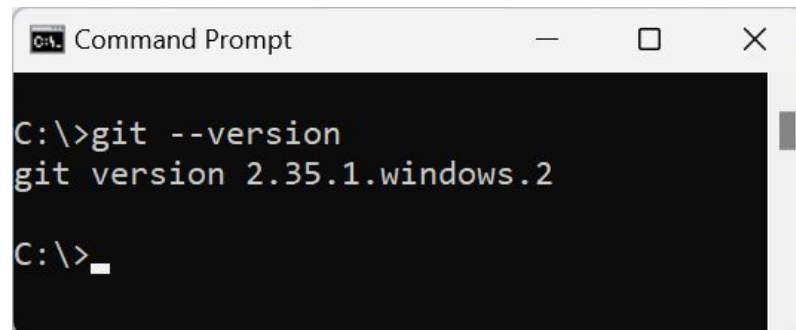
# Instalación del CLI de Git

Para instalar la línea de comando de Git en Windows, se hace en la siguiente dirección:

[Gitforwindows.org](https://gitforwindows.org)

Para verificar que se haya instalado correctamente: abrir una línea de comando y escribir:

```
git --version
```



```
C:\>git --version
git version 2.35.1.windows.2

C:\>_
```