¿Para qué sirve GIT?

1. **Control de versiones:**
    - Permite llevar un registro de los cambios realizados en los archivos a lo largo del tiempo.
    - Puedes volver a versiones anteriores si algo sale mal.
2. **Colaboración:**
    - Facilita el trabajo en equipo al permitir que varias personas trabajen en los mismos archivos sin sobrescribir los cambios de los demás.
    - Proporciona herramientas para combinar cambios y resolver conflictos cuando diferentes personas editan e
3. **Ramas:**
    - Puedes crear ramas separadas para trabajar en características específicas o correcciones de errores sin afectar el código principal.
    - Facilita la experimentación, ya que puedes probar ideas sin riesgo de dañar la versión estable.
4. **Historial detallado:**
    - Almacena un historial completo de todos los cambios, lo que permite identificar quién hizo qué y por qué.
    - Facilitar l
5. **Despliegue continuo:**
    - GIT se integra con herramientas de integración continua y despliegue continuo (CI/CD), ayudando a automatizar pruebas, construcciones y despliegues.
6. **Trabajo local y remoto:**
    - Puedes trabajar en tu proyecto localmente sin necesidad de conexión a internet.
    - Al sincronizar con repositorios remotos como **GitHub** ,**GitLab** o ****Bitbucket** ,

¿Por qué es importante?

En proyectos complejos, es común que haya múltiples personas trabajando en diferentes partes al mismo tiempo. Sin GIT, gestionar esos cambios sería caótico. Con GIT, puedes trabajar de manera ordenada, minimizar errores y mantener un flujo de

Características principales de GIT:

1. **Distribuido:**
    - Cada copia de un repositorio en GIT es independiente y contiene toda la historia del proyecto. Esto significa que puedes trabajar sin estar
2. **Control de versiones:**
    - GIT guarda un historial detallado de cada cambio realizado en los archivos del proyecto. Esto permite volver a estados anteriores, analizar quién realizó qué cambio y cuándo.
3. **Rendimiento:**
    - Es muy rápido al
4. **Ramas y fusiones eficientes:**
    - GIT permite crear ramas para trabajar en nuevas características o solucionar problemas sin interferir con el código principal. Estas ramas pueden combinarse fácilmente cuando están listas.
5. **Integridad:**
    - GIT asegura la integridad de los datos mediante el uso de un sistema de "hashes" (SHA-1) para identificar los cambios. Esto garantiza que los datos no podrán ser modificados sin que se detecten.
6. **Colaboración:**
    - Facilita el trabajo en equipo al permitir que múltiples desarrolladores trabajen en el mismo proyecto simultáneamente y sin conflictos mayores.

¿Cómo funciona?

1. **Repositorios:**
    - Un repositorio es donde se almacena todo el proyecto y su historial de cambios. Puede ser local (en tu computadora) o remoto (en plataformas como GitHub, GitLab o Bitbucket).
2. **Compromisos:**
    - Son "instantáneas" de los cambios que se han realizado en el proyecto. Cada commit tiene un mensaje que explica el cambio y está vinc
3. **Ramas:**
    - Representan diferentes líneas de trabajo. Por ejemplo, puedes tener una rama principal (*main*o *maestro*maestro) y
4. **Puesta en escena y Área de preparación:**
    - Antes de confirmar un cambio (commit

¿Por qué es importante?

- GIT es esencial en el desarrollo de software moderno, especialmente en proyectos de colaboración. Su uso permite a los equipos trabajar juntos de manera eficiente, gestionar versiones y reducir riesgos de errores.

1.`git init`

¿Qué hace?

- Inicializa un nuevo repositorio GIT en el directorio actual. Esto crea una carpeta oculta llamada `.git`que contiene

Ejemplo de uso:

intento

Copiar código

```
git init
```

- Usado al comenzar un proyecto nuevo para que GIT pueda se

---

2. `git add`

¿Qué hace?

- Agrega archivos al "Área de preparación". Esto significa que los archivos están listos para ser confirmados en el historial de GIT.

Ejemplo de uso:

intento

Copiar código

```
git add archivo.txt
```

```
git add .
```

- `git add archivo.txt`:Antes
- `git add .`:Antes

---

3. `git commit`

¿Qué hace?

- Confirma los cambios del

Ejemplo de uso:

intento

Copiar código

```
git commit -m "Descripción breve del cambio"
```

- El mensaje entre comillas explica el propósito del cambio (por ejemplo: "Arreglar el bug en el fo

---

4.`git status`

¿Qué hace?

- Muestra el estado actual del repositorio, incluyendo:
  - Archivos que han sido modificados.
  - Archivos que están en el área de preparación.
  - Archivos no rastreados.

Ejemplo de uso:

intento

Copiar código

```
git status
```

- Útil para saber qué cambios se han hecho antes de añadirlos o confirmarlos.

---

5. `git push`

¿Qué hace?

- Envía los cambios confirmados al repositorio remoto (por ejemplo, en GitHub o GitLab). Esto es necesario para compartir tu trabajo con otros o para respaldarlo.

Ejemplo de uso:

intento

Copiar código

```
git push origin main
```

- **`origin`**: Nombre
- **`main`**:`master`es

Resumen en una frase:

Estos comandos te permiten inicializar un proyecto (`git init`),`git add`),`git commit`), verificar`git status`) y Compartir`git push`).