Comando:

git init

Función:

El comando git init **inicializa un nuevo repositorio de Git** en el directorio actual. Esto significa que crea las estructuras y configuraciones necesarias para que Git comience a rastrear cambios en ese directorio.

Paso a Paso:

1. **Crea un directorio de Git oculto**:
    - Cuando ejecutas git init, se crea un subdirectorio oculto llamado .git en el directorio actual.
    - Este directorio contiene todos los archivos y metadatos que Git necesita para gestionar el repositorio, como:
        - Información de las ramas (refs).
        - Historial de cambios (objects).
        - Configuraciones del repositorio (config).
2. **Convierte el directorio actual en un repositorio de Git**:
    - El directorio actual pasa a estar listo para que Git comience a rastrear los cambios en los archivos.
3. **No afecta archivos existentes**:
    - Si el directorio ya contiene archivos, git init no los modifica ni los elimina.
    - Sin embargo, Git no comenzará a rastrearlos automáticamente. Necesitas agregarlos explícitamente usando comandos como git add.
4. **Establece un repositorio vacío**:
    - Después de la inicialización, el repositorio está vacío y no tiene commits, ramas, ni archivos rastreados.

Ejemplo práctico:

a) Crear un nuevo proyecto desde cero:

mkdir mi-proyecto

cd mi-proyecto

git init

- **Resultado**: Se crea un repositorio vacío en el directorio mi-proyecto.

b) Inicializar un directorio existente:

cd proyecto-existente

git init

- **Resultado**: Convierte proyecto-existente en un repositorio de Git sin alterar los archivos actuales.

Verificación:

Después de ejecutar git init, puedes verificar que el directorio .git fue creado:

ls -a

Deberías ver algo como:

archivo1.txt

archivo2.txt

Comando:

git remote add origin https://github.com/christiangithub24/expotec25.git

Paso a paso:

1. **git**:
    - Indica que se está utilizando Git, el sistema de control de versiones.
2. **remote**:
    - Gestiona las conexiones a repositorios remotos.
    - Este subcomando te permite ver, añadir, cambiar o eliminar las referencias a repositorios remotos.
3. **add**:
    - Especifica que deseas añadir un nuevo repositorio remoto.
    - Aquí se está registrando una nueva dirección URL para conectar tu repositorio local con uno remoto.
4. **origin**:
    - Es el **nombre del repositorio remoto** que estás agregando.
    - Por convención, el nombre origin se utiliza como alias para la ubicación principal del repositorio remoto. Sin embargo, puedes usar cualquier nombre (por ejemplo, main-remote).
    - Este alias facilita las operaciones posteriores, ya que puedes referirte al remoto con origin en lugar de escribir toda la URL cada vez.
5. **https://github.com/christiangithub24/expotec25.git**:
    - Es la **URL del repositorio remoto** alojado en GitHub.
    - Este es el repositorio remoto al que se conectará tu repositorio local.
    - Puede ser una URL HTTP/HTTPS, SSH, o incluso un archivo local.

¿Qué hace este comando en conjunto?

- Añade un nuevo repositorio remoto llamado origin, con la URL https://github.com/christiangithub24/expotec25.git.
- Después de ejecutar este comando, tu repositorio local sabrá que tiene una conexión con el repositorio remoto especificado, y podrás:
    - Subir cambios (git push).
    - Descargar actualizaciones (git pull o git fetch).

- Sincronizar tu trabajo con el remoto.

Verificación:

Para comprobar que el remoto fue agregado correctamente:

git remote -v

Esto mostrará algo como:

origin https://github.com/christiangithub24/expotec25.git (fetch)

origin https://github.com/christiangithub24/expotec25.git (push)

Nota importante:

Si decides usar otro nombre en lugar de origin, asegúrate de usarlo en los comandos posteriores. Por ejemplo:

git push origin main

Este comando empujará tus cambios locales de la rama main al remoto origin.

comando:

git status

Función general:

El comando git status muestra el estado actual del repositorio. Te proporciona información sobre los cambios en los archivos, las ramas y el estado general del repositorio.

Paso a paso:

1. **git**:
    - Es el sistema de control de versiones que estás usando.
2. **status**:
    - Es un comando que consulta el estado del repositorio local.
    - Muestra información sobre:
        - Archivos **modificados**.
        - Archivos **nuevos** no rastreados.
        - Archivos listos para ser confirmados (en el área de preparación o "staging area").
        - Rama actual y su relación con el repositorio remoto.

comando:

```
git add .
```

Función general:

El comando git add . **añade al área de preparación (staging area)** todos los archivos y cambios del directorio actual y sus subdirectorios, preparándolos para el siguiente commit.

Paso a paso:

1. **git**:
    - Indica que estás utilizando Git, el sistema de control de versiones.
2. **add**:
    - Este comando añade cambios al área de preparación.
    - El área de preparación es donde se colocan los cambios que deseas incluir en el próximo commit.
3. **. (punto)**:
    - Representa **todos los archivos y directorios** en el directorio actual.

- Esto incluye:
  - Archivos nuevos (no rastreados) que aún no están en el repositorio.
  - Archivos modificados.
  - Archivos eliminados.

Lo que sucede cuando ejecutas git add .

1. **Archivos no rastreados (untracked files)**:
   - Los archivos nuevos que nunca han sido añadidos al repositorio se moverán al área de preparación.
2. **Archivos modificados (modified files)**:
   - Los cambios realizados en archivos previamente rastreados por Git se preparan para el commit.
3. **Archivos eliminados (deleted files)**:
   - Si has eliminado archivos, esos cambios también se añadirán al área de preparación.
4. **Archivos ignorados (ignored files)**:
   - Los archivos definidos en un archivo .gitignore no se verán afectados por git add ..

comando:

git commit -m "primer repositorio"

Función general:

Este comando **crea un commit** (un registro de los cambios en el repositorio) con un mensaje descriptivo específico.

Un commit es como una "fotografía" del estado actual de los archivos en el área de preparación (staging area). Este comando guarda permanentemente estos cambios en el historial del repositorio.

Paso a paso:

1. **git**:
    - Indica que estás usando Git, el sistema de control de versiones.
2. **commit**:
    - Crea un nuevo commit con los cambios que están en el área de preparación.
    - Si no has añadido cambios con git add, el comando no funcionará, ya que no hay nada preparado para ser confirmado.
3. **-m**:
    - Este indicador (flag) permite añadir un mensaje al commit directamente desde la línea de comandos.
    - Evita abrir el editor de texto para escribir el mensaje del commit.
4. **"primer repositorio"**:
    - Es el mensaje del commit.
    - Este mensaje debe ser una descripción breve y clara de los cambios que estás registrando en este commit.

Qué sucede al ejecutar este comando:

1. **Git toma una instantánea**:
    - Git guarda una instantánea de los archivos que están en el área de preparación.
    - Cualquier archivo no preparado (sin git add) **no será incluido en el commit**.
2. **El mensaje se guarda con el commit**:
    - El mensaje "primer repositorio" se adjunta al commit como una descripción. Esto ayuda a identificar los cambios realizados en ese punto del historial.
3. **Se actualiza el historial del repositorio**:
    - El commit se añade al historial del repositorio. Si ejecutas git log, podrás ver este commit registrado.

comando:

git push origin main

Función general:

Este comando **envía los commits de la rama main desde el repositorio local al repositorio remoto identificado como origin**.

En otras palabras, sincroniza los cambios confirmados localmente con el repositorio remoto, haciendo que los demás puedan acceder a ellos si tienen acceso al repositorio.

Paso a paso:

1. **git**:
    - Indica que estás utilizando Git, el sistema de control de versiones.
2. **push**:
    - Este subcomando **sube los cambios** desde el repositorio local al repositorio remoto.
3. **origin**:
    - Es el alias del repositorio remoto al que se enviarán los cambios.
    - Por convención, cuando configuras un repositorio remoto con git remote add, su alias suele ser origin. Si usaste otro nombre, deberías reemplazar origin por ese alias.
4. **main**:
    - Es la rama que deseas enviar al repositorio remoto.
    - En este caso, estás subiendo los commits realizados en la rama main.

Qué sucede al ejecutar este comando:

1. **Autenticación con el remoto**:
    - Git se conecta al repositorio remoto identificado como origin. Si es la primera vez que haces push, puede solicitarte autenticación (por ejemplo, con un token de acceso personal o credenciales SSH).
2. **Sincronización de commits**:
    - Todos los commits que están en la rama local main y que no están aún en el remoto se envían al repositorio remoto.
3. **Actualización de la rama remota**:
    - La rama main en el repositorio remoto se actualiza para reflejar el estado actual de la rama local.

comando:

git pull origin main

Función general:

El comando **git pull origin main** sincroniza tu repositorio local con la rama main del repositorio remoto llamado origin. Combina dos operaciones: **descargar cambios (fetch)** y **fusionar cambios (merge)**.

Paso a paso:

1. **git**:
    - Indica que estás utilizando Git, el sistema de control de versiones.
2. **pull**:
    - Combina dos operaciones:
        - **Fetch**: Descarga los cambios más recientes de la rama especificada en el remoto.
        - **Merge**: Fusiona esos cambios en tu rama local actual.
3. **origin**:
    - Es el alias del repositorio remoto desde donde se descargan los cambios.
    - Por convención, el repositorio remoto principal se llama origin.
4. **main**:
    - Es la rama del repositorio remoto que deseas sincronizar con tu rama local.
    - Los cambios de origin/main se fusionarán con la rama local actual.

Qué sucede al ejecutar este comando:

1. **Conexión con el remoto**:
    - Git se conecta al repositorio remoto identificado como origin.
2. **Descarga de los cambios**:
    - Git obtiene los commits más recientes de la rama main en el repositorio remoto y los almacena temporalmente en tu máquina local.
3. **Fusión automática**:
    - Git intenta fusionar automáticamente los cambios del remoto (origin/main) con tu rama local actual.
    - Si no hay conflictos, la fusión se realiza automáticamente.
    - Si hay conflictos, Git te informará para que los resuelvas manualmente.

Comando

git pull origin main

Función general:

El comando **git pull origin main** sincroniza tu repositorio local con la rama main del repositorio remoto llamado origin. Combina dos operaciones: **descargar cambios (fetch)** y **fusionar cambios (merge)**.

Paso a paso:

1. **git**:
    - Indica que estás utilizando Git, el sistema de control de versiones.
2. **pull**:
    - Combina dos operaciones:
        - **Fetch**: Descarga los cambios más recientes de la rama especificada en el remoto.
        - **Merge**: Fusiona esos cambios en tu rama local actual.
3. **origin**:
    - Es el alias del repositorio remoto desde donde se descargan los cambios.
    - Por convención, el repositorio remoto principal se llama origin.
4. **main**:
    - Es la rama del repositorio remoto que deseas sincronizar con tu rama local.
    - Los cambios de origin/main se fusionarán con la rama local actual.

Qué sucede al ejecutar este comando:

1. **Conexión con el remoto**:
    - Git se conecta al repositorio remoto identificado como origin.
2. **Descarga de los cambios**:
    - Git obtiene los commits más recientes de la rama main en el repositorio remoto y los almacena temporalmente en tu máquina local.
3. **Fusión automática**:
    - Git intenta fusionar automáticamente los cambios del remoto (origin/main) con tu rama local actual.
    - Si no hay conflictos, la fusión se realiza automáticamente.
    - Si hay conflictos, Git te informará para que los resuelvas manualmente.

Comando

git push -u origin main

Paso a paso:

1. **git push**: Enviar commits desde la rama local al remoto.
2. **-u o (--set-upstream)**: Configura una relación de seguimiento entre la rama local (main) y la rama remota (origin/main). Esto permite que los comandos como git pull y git push se ejecuten sin especificar el remoto y la rama en el futuro.
3. **origin**: Es el alias del repositorio remoto.
4. **main**: Es la rama local/remota que se sincroniza.

Si tu objetivo:

1. Sincronizar tu rama local con los últimos cambios de la rama main del remoto (origin).
2. Configurar una relación de seguimiento entre ambas ramas.

Comando

git push origin main

Función general:

Este comando **envía los commits** de la rama local main al repositorio remoto identificado como origin, actualizando la rama main del remoto con los cambios más recientes.

Paso a paso:

1. **git**:
   - Indica que estás utilizando Git, el sistema de control de versiones.
2. **push**:
   - Este subcomando **sube los cambios** del repositorio local al repositorio remoto.
3. **origin**:
   - Es el alias que identifica el repositorio remoto. Este alias se crea típicamente cuando usas el comando git remote add origin <URL> para vincular tu repositorio local con uno remoto.

4. **main**:
    - Es la rama del repositorio local cuyos cambios deseas enviar al repositorio remoto.
    - Si la rama main no existe en el remoto, Git la creará automáticamente al ejecutar este comando.

Qué sucede al ejecutar este comando:

1. **Conexión con el remoto**:
    - Git establece una conexión con el repositorio remoto identificado como origin.
2. **Sincronización de commits**:
    - Git compara los commits de la rama local main con la rama main del remoto (origin/main).
    - Solo los commits que están en tu repositorio local pero no en el remoto son enviados.
3. **Actualización de la rama remota**:
    - Si el remoto ya tiene una rama llamada main, se actualiza con los nuevos commits.
    - Si la rama main no existe en el remoto, Git la crea automáticamente.
4. **Confirmación exitosa**:
    - Si el envío es exitoso, verás un mensaje como este:

Enumerating objects: 5, done.

Counting objects: 100% (5/5), done.

Delta compression using up to 4 threads

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 346 bytes | 346.00 KiB/s, done.

Total 3 (delta 2), reused 0 (delta 0)

To https://github.com/tuusuario/tu-repositorio.git

* [new branch] main -> main