educación IT
CODE YOUR FUTURE

Comandos para trabajar en el repositorio local

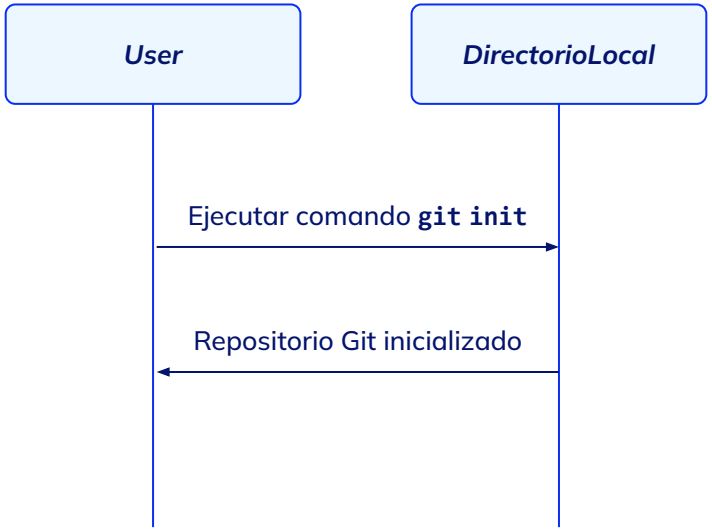
git init

Uso básico:

```
git init
```

El comando `git init` **crea un nuevo repositorio Git en el directorio actual. Se crea un nuevo subdirectorio llamado *.git*** que contiene todos los archivos necesarios para el control de versiones.

Convertir el directorio actual en un repositorio Git permite el **seguimiento de cambios** en los archivos y la utilización de otros comandos de Git para gestionar el historial de versiones.



git add

Uso básico (para **agregar un archivo**):

```
git add <archivo>
```

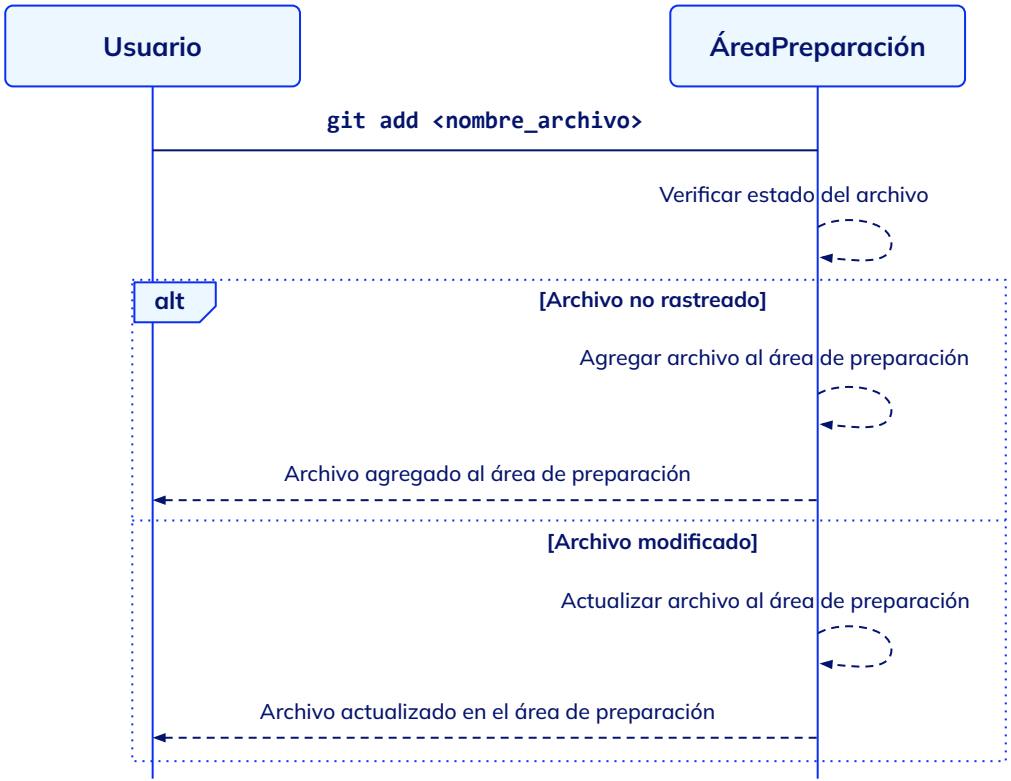
Uso básico (para **agregar todos los archivos**):

```
git add .
```

El comando `git add` se utiliza para **agregar cambios, en el directorio de trabajo, al área de preparación *(staging area)*.** Esto prepara los cambios para ser incluidos en el próximo **`commit`**.

Este comando **permite una mayor granularidad y control sobre los cambios se registran**. Esta funcionalidad facilita la gestión del historial de versiones de nuestro proyecto.

Representación gráfica:



git commit

Uso básico:

```
git commit -m "Mensaje descriptivo"
```

El comando `git commit` permite al usuario **registrar los cambios realizados en el repositorio de manera permanente.** Cuando se ejecuta este comando, **se toma una instantánea *(snapshot)* de los cambios** realizados en el área de preparación *(staging area)* y se guarda en el historial del repositorio.

Es muy importante poner un mensaje descriptivo en el *commit* para comprender rápidamente el propósito y el alcance de cada *commit*.

Representación gráfica:

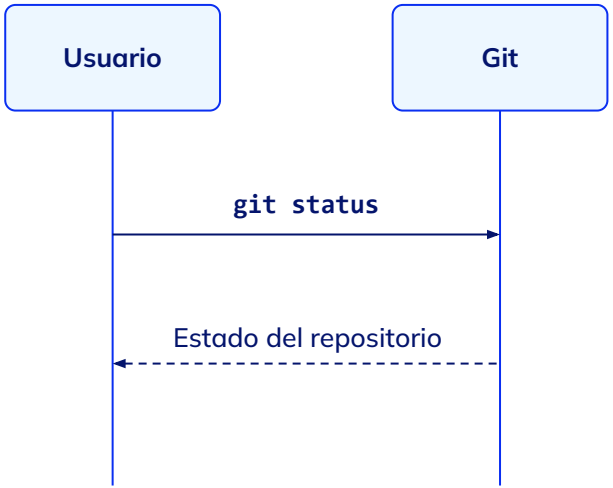
git status

Uso básico:

```
git status
```

El comando `git status` **muestra el estado actual del repositorio,** incluyendo información sobre los **archivos modificados, archivos no rastreados y el estado del área de preparación** *(staging area).*

Proporciona una instantánea rápida y útil del estado actual del proyecto.



Otros comandos para trabajar localmente

`git log`	Muestra el **historial de commits en el repositorio local,** incluyendo detalles como el autor del *commit,* la fecha y la hora, y el mensaje del *commit.* Es útil para revisar el historial de cambios y entender la evolución del proyecto.
`git diff`	Muestra las **diferencias entre los archivos en el directorio de trabajo y el área de preparación *(staging area),* o entre el área de preparación y la última confirmación *(commit).*** Es útil para revisar los cambios realizados antes de agregarlos al área de preparación o realizar un *commit.*
`git reset`	Se utiliza para **deshacer cambios en el repositorio local.** Puede usarse para deshacer *commits,* mover archivos del área de preparación de vuelta al directorio de trabajo, o incluso revertir el repositorio a un estado anterior.
`git rm`	Se utiliza para **eliminar archivos del repositorio y del sistema de archivos local.** Al igual que git add, es necesario realizar un *commit* después de usar `git rm` para confirmar los cambios en el repositorio.

Comandos de Git remotos

git clone

Uso básico:

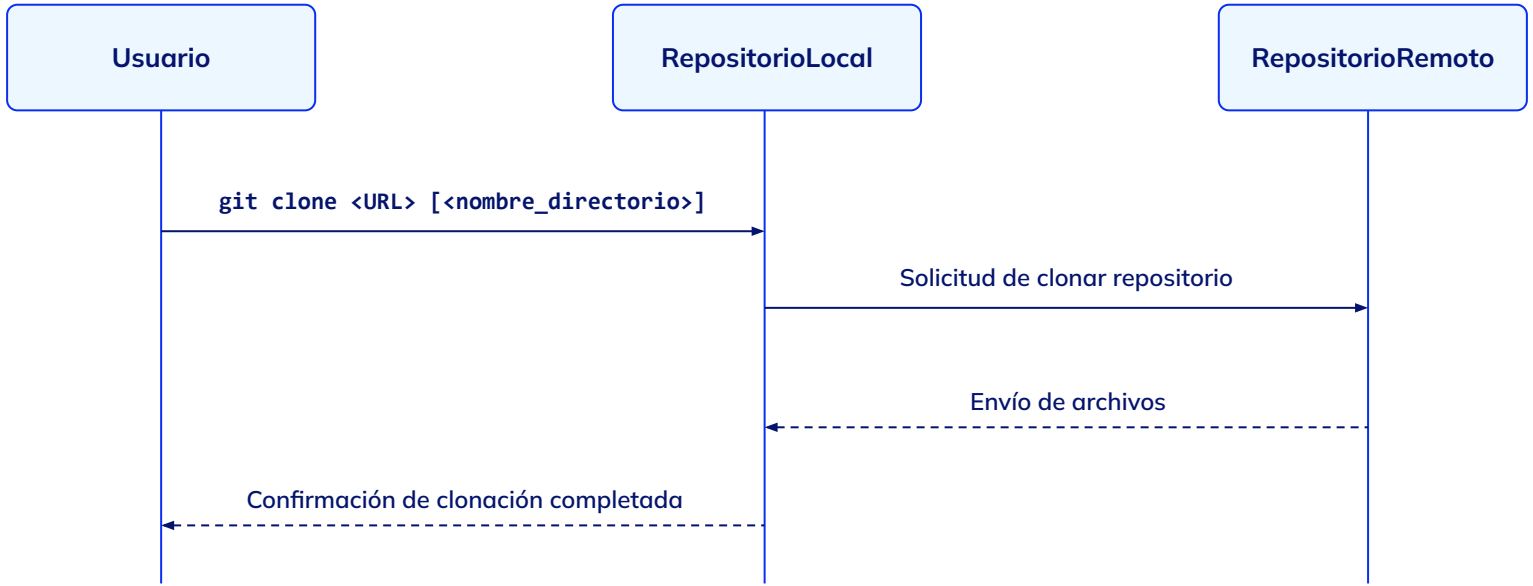
```
git clone <url-repo>
```

Este comando se utiliza para **crear una copia local de un repositorio remoto.**

Este comando **copia los archivos del repositorio con toda su historia.** De esta manera, permite trabajar con el proyecto en la máquina local.

Veamos el esquema
de la próxima pantalla.

Representación gráfica:



git push

Uso básico:

```
git push <nombre_remoto> <nombre_rama>
```

Este comando se utiliza para **subir *commits* locales a un repositorio remoto en Git.**

Este comando es esencial para colaborar en proyectos donde **múltiples desarrolladores trabajan en conjunto**, ya que permite **sincronizar cambios** hechos en un repositorio local con su contraparte remota.

git pull

Uso básico:

```
git pull <nombre_remoto> <nombre_rama>
```

Este comando es crucial en Git para **actualizar el repositorio local con los cambios de un repositorio remoto.**

Combina dos comandos:

- **`git fetch`** (descargar cambios desde el remoto).
- **`git merge`** (integrar esos cambios en la rama actual).

Veamos el esquema de la próxima pantalla.

Representación gráfica de *git push* y de *git pull:*