

SISTEMA DE
**CONTROL DE
VERSIONES**



66

Sistema de control de versiones

Es una herramienta que registra los cambios en archivos a lo largo del tiempo.

Permite colaborar con otros desarrolladores de manera eficiente.

Facilita la recuperación de versiones anteriores de un archivo.

Según <https://www.atlassian.com>

El control de versiones, también conocido como "control de código fuente", es la práctica de rastrear y gestionar los cambios en el código de software. Los sistemas de control de versiones son herramientas de software que ayudan a los equipos de software a gestionar los cambios en el código fuente a lo largo del tiempo. A medida que los entornos de desarrollo se aceleran, los sistemas de control de versiones ayudan a los equipos de software a trabajar de forma más rápida e inteligente.

Tipos de Sistemas de Control de Versiones

Locales

Control de versiones en una sola máquina.

Ejemplo: Guardar copias manuales de archivos.

Centralizados **CVCS**

Un servidor central almacena todas las versiones del proyecto.

Distribuidos **DVCS**

Cada usuario tiene una copia completa del repositorio.
Ejemplo: Git, Mercurial.

¿Por qué usar un Sistema de Control de Versiones?

1

**Historial
de
cambios.**

2

**Trabajo
colaborati
vo.**

3

**Respaldo
seguro de
archivos.**

4

**Manejo
eficiente
de
versiones.**

5

**Facilita la
recuperación
de errores.**



Es un ingeniero de software finlandés-estadounidense, conocido por iniciar y mantener el desarrollo del kernel Linux. Actualmente es responsable de la coordinación del proyecto. También ha desarrollado el software de control de versiones Git.

Git

Git es un sistema de control de versiones de código fuente, creado por Linus Torvalds en 2005. Es una herramienta que se utiliza para llevar un registro de los cambios en el código fuente de un proyecto y permitir a varios desarrolladores trabajar en el mismo proyecto de manera simultánea.

Conceptos Básicos de Git

Repository

Lugar donde se almacenan los archivos y su historial de cambios.

Merge

Une cambios de diferentes ramas.

Commit

Registro de cambios en el repositorio.

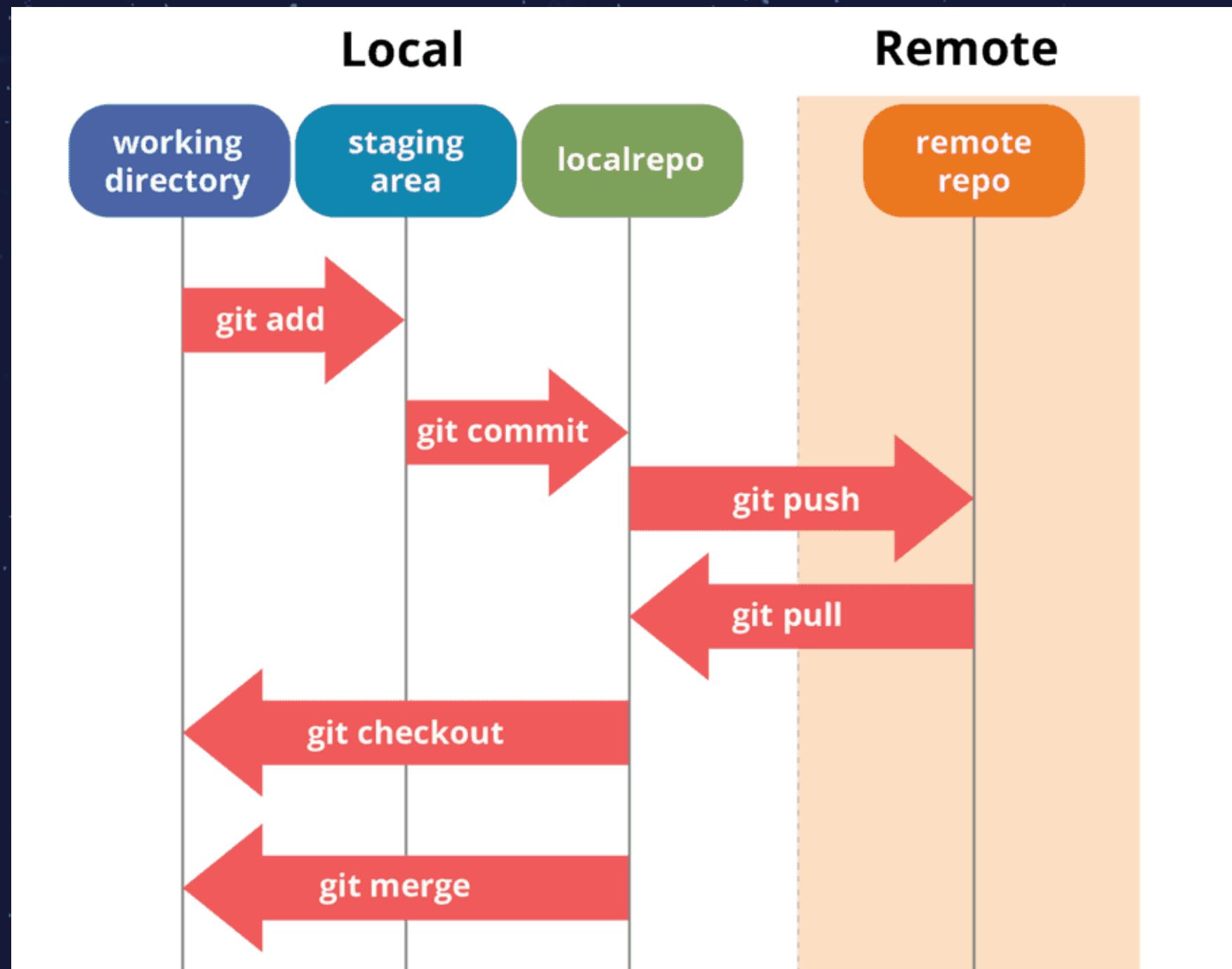
Branch (Rama)

Permite desarrollar nuevas funcionalidades sin afectar el código principal.

Push & Pull

Enviar y recibir cambios desde un repositorio remoto.

Flujo de Trabajo en Git



1. Inicializar un repositorio: **git init**
2. Agregar archivos al área de preparación: **git add .**
3. Confirmar cambios: **git commit -m "Mensaje"**
4. Conectar con un repositorio remoto: **git remote add origin URL**
5. Subir cambios: **git push origin main**

Subir Cambios a un Repositorio Existente

Agregar y confirmar los cambios
`git commit -m "Agregando nueva línea al README"`

Subir los cambios a GitHub
`git push origin main`

Descargar Cambios desde GitHub

Clonar el repositorio si aún no lo tienes

```
git clone https://github.com/mjdiazdev/prueba.git
```

Obtener la última versión del repositorio

```
git pull origin main
```