

# ¿Qué es el **control de versiones**?



El **control de versiones** es un sistema que registra cambios en un archivo o proyecto a lo largo del tiempo, permitiendo recuperar versiones específicas más adelante.

Un sistema de control de versiones puede **registrar cambios** en cualquier tipo de archivo dentro de un ordenador pero se utiliza principalmente para archivos de código fuente en proyectos de desarrollo de software.

En inglés se les conoce como **Version Control System (VCS)**.

# Ventajas del control de versiones



Un sistema de control de versiones permite **múltiples ventajas** a la hora de trabajar con archivos en un proyecto. Un sistema de control de versiones permite:

- Mantener una **versión de cada cambio** introducido en los archivos.
- **Revertir cambios** en un archivo o el proyecto entero.
- **Comparar cambios** para visualizar la evolución de un archivo.
- Ver qué **usuario** ha introducido qué **cambios** y en qué **fecha** lo hizo. También visualizar un mensaje descriptivo asociado a los cambios.
- **Trabajar de forma colaborativa** en equipo al permitir trabajar sobre los mismos archivos y fusionar los cambios introducidos por cada persona.
- Creación de **ramas** o flujos de trabajo donde podemos tener versiones diferentes de la aplicación.

# ¿Qué es Git?

---

**Git** es un sistema de control de versiones distribuido de código abierto y gratuito creado por Linus Torvalds en 2005.

Git está diseñado para el control de versiones desde proyectos simples a muy complejos de manera rápida y eficiente, aportando integridad de los datos.

En la actualidad es el sistema de control de versiones **más popular**, superando a otros sistemas de control de versiones como Subversion, CVS, Perforce y ClearCase.



# Instalación de Git

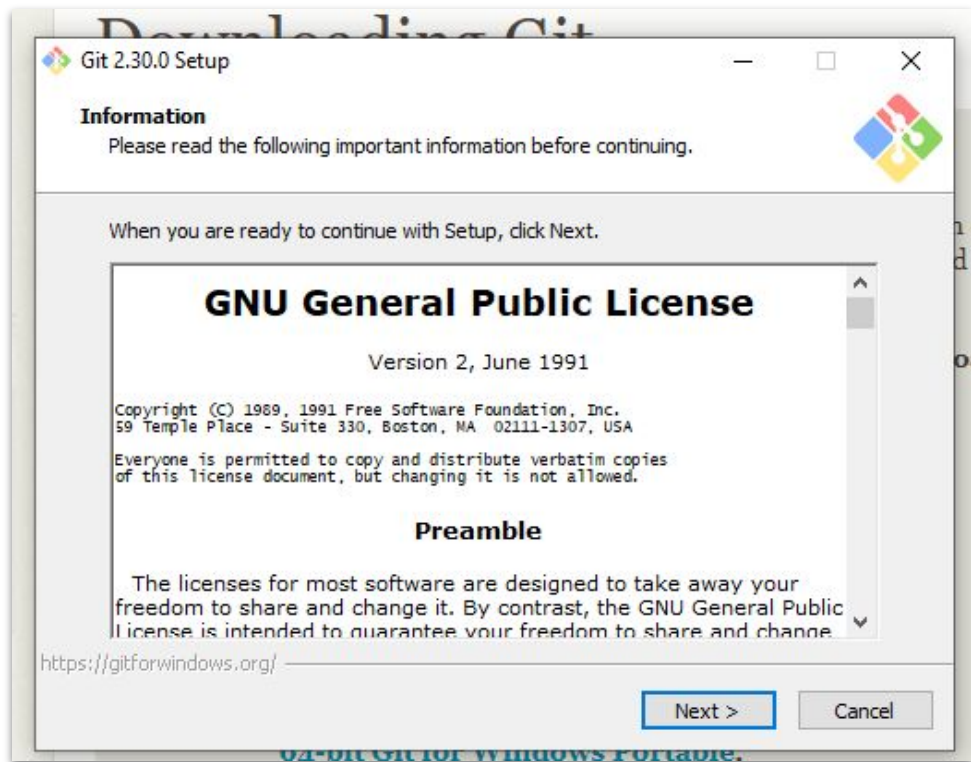
La última **versión estable** de Git se puede descargar desde la [página oficial](#).



# Instalación de Git

Una vez descargado ejecutamos el archivo exe y seguimos los pasos de la instalación. Pulsamos en todos Next.

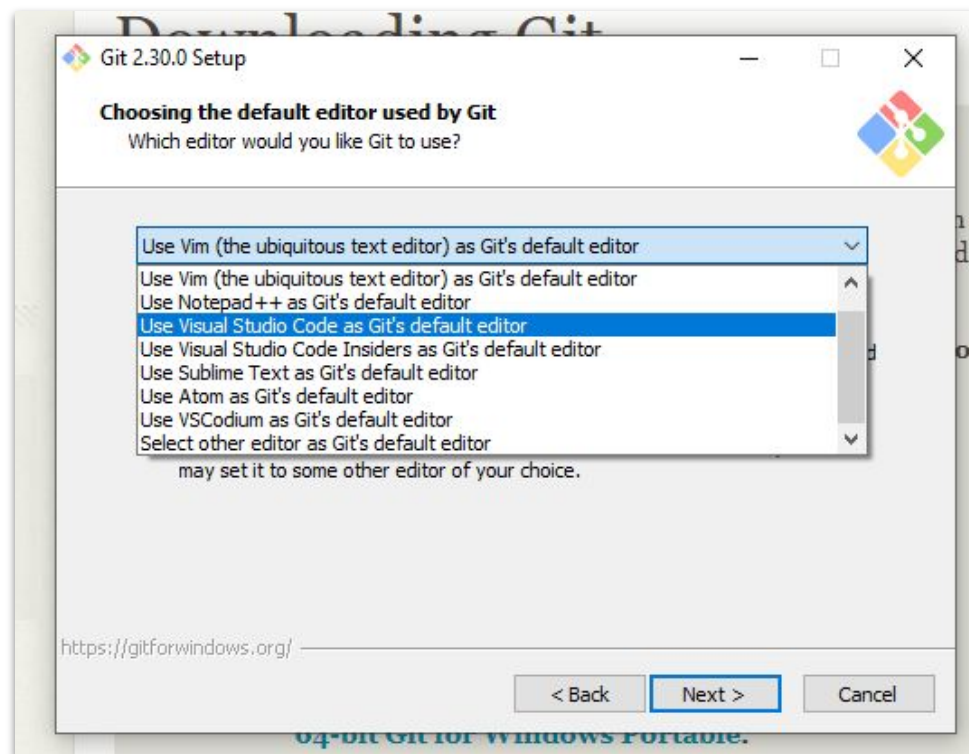
**Importante:** en el paso Default Editor seleccionaremos Visual Studio Code como se indica en la siguiente diapositiva.



# Instalación de Git

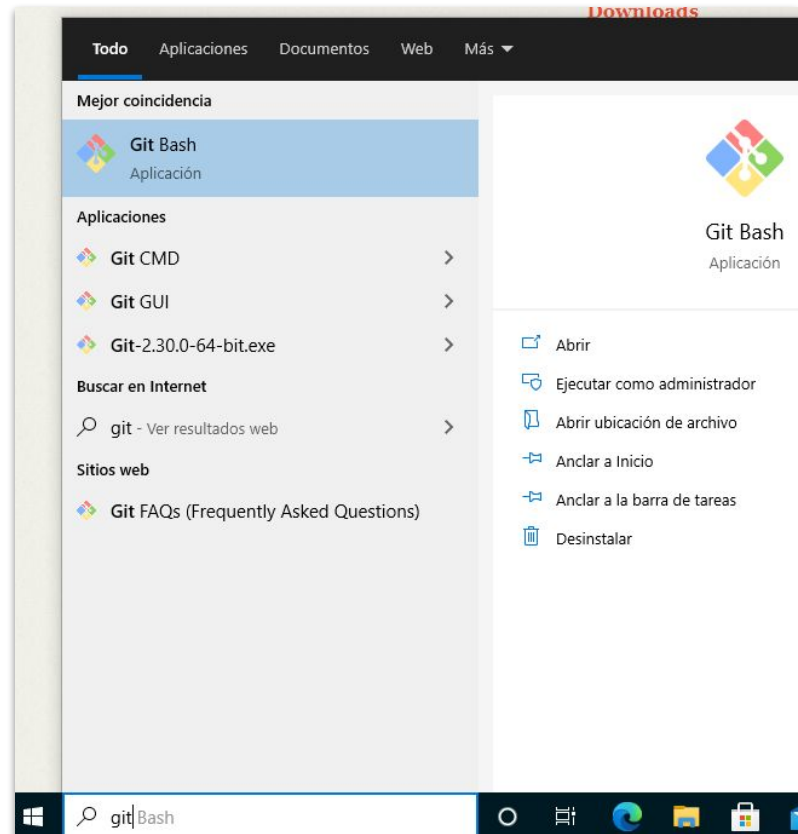
**Importante:** en el paso Default Editor seleccionaremos Visual Studio Code como se indica en la siguiente diapositiva.

El resto de pasos utilizaremos la configuración por defecto, pulsando Next o siguiente.



# Instalación de Git

Una vez instalado si buscamos Git en el explorador de Windows aparecerá el programa **Git Bash**.



# Configuración Git

Abrimos la aplicación **Git Bash** y configuraremos las propiedades **user.name** y **user.email** globales por defecto, de esta manera cada vez que creemos un proyecto Git asignará estos datos por defecto.

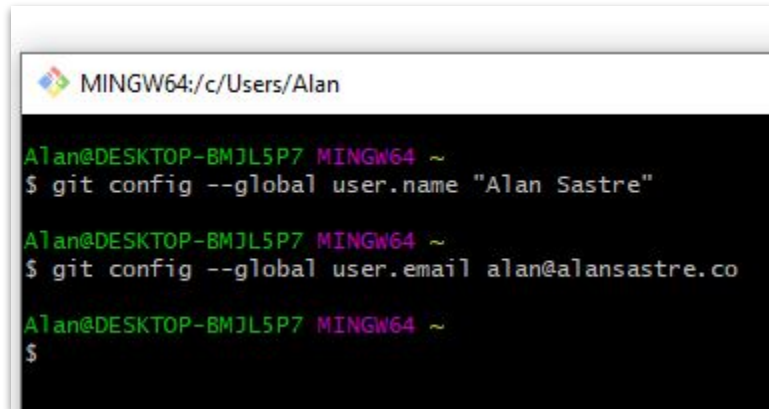
Una vez abierta la aplicación ejecutaremos los comandos:

```
git config --global user.name "John Doe"
```

```
git config --global user.email johndoe@example.com
```

**Nota:** si en un proyecto concreto queremos utilizar otro nombre o email podemos ejecutar estos mismos comandos sobre el proyecto sin poner **--global**.

**Nota:** se recomienda utilizar el mismo email con el que nos registremos en GitHub.com



```
MINGW64:/c/Users/Alan
Alan@DESKTOP-BMJL5P7 MINGW64 ~
$ git config --global user.name "Alan Sastre"
Alan@DESKTOP-BMJL5P7 MINGW64 ~
$ git config --global user.email alan@alansastre.co
Alan@DESKTOP-BMJL5P7 MINGW64 ~
$
```

Para visualizar las configuraciones actuales utilizamos el comando:

```
git config --list --show-origin
```



# Funcionamiento básico de Git



Para incluir git en un proyecto necesitamos inicializarlo sobre el mismo, para ello se utiliza el comando **git init**.

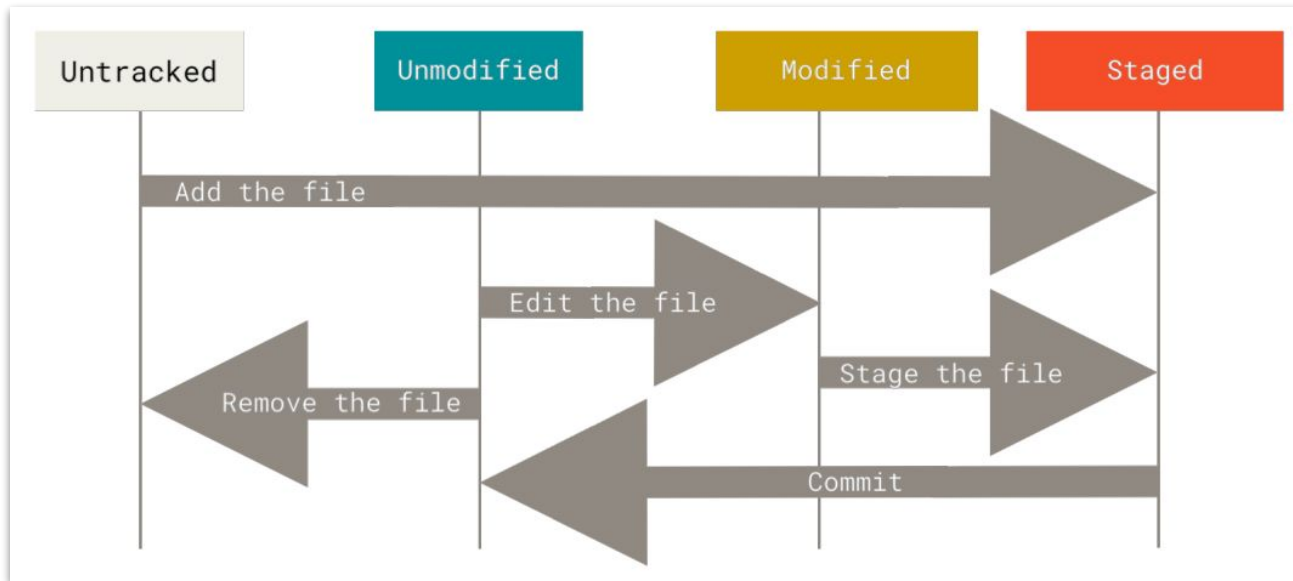
Una vez inicializado git crea dos áreas de trabajo:

- **Área de ensayo (staging area):** es temporal, nos muestra el estado de cada archivo
- **Área de repositorio local:** el área donde se versionan los cambios.

Para dar seguimiento a los archivos del proyecto utilizamos el comando **git add**, en ese momento estos archivos forman parte del staging area y git los tendrá bajo seguimiento.

Cuando se quiere registrar los cambios realizados se utiliza el comando **git commit**, en ese momento los archivos del staging area pasan al área de repositorio local. Es en este punto cuando quedan registrados los cambios.

# Funcionamiento básico de Git



Ciclo de vida del estado de los archivos en un proyecto con git. Fuente: Libro Pro Git.