

CONTROL DE VERSIONES

Desarrollo **colaborativo**

Se suele utilizar de manera muy habitual para llevar un control de las versiones o revisiones de un determinado programa y poder tener un repositorio accesible con las diferentes versiones creadas.

Uso de control de versiones modo exclusivo, donde para poder realizar cambios se debe marcar previamente cuál es el elemento sobre el que se va a trabajar, de este modo, el control de versiones impedirá que otro usuario pueda modificar ese elemento hasta que se elimine la restricción.

De este modo, se evita la aparición de conflictos o inconsistencias que suelen aparecer en el desarrollo colaborativo al poder desarrollar de modo asíncrono y simultáneo sobre el código del repositorio en detrimento de la libertad que ofrece el desarrollo colaborativo completo.

REPOSITARIOS

Un **repositorio** es básicamente un **servidor de archivos** típico, con una gran diferencia: lo que hace a los repositorios especiales en comparación con esos servidores de archivos es que recuerdan todos los cambios que alguna vez se hayan escrito en ellos, de este modo, cada vez que actualizamos el repositorio, éste recuerda cada cambio realizado en el archivo o estructura de directorios. Además, permite establecer información adicional por cada actualización, pudiendo tener por ejemplo un **changelog** de las versiones en el propio repositorio.

Cada herramienta de control de versiones tiene su propio repositorio, y, por desgracia, no son **interoperables**, es decir, no puedes obtener los datos del repositorio o actualizarlo si el repositorio y el control de versiones no coinciden.

REPOSITORIOS

También podríamos asociar una **copia de trabajo** (la que se encuentra en nuestro disco duro) a varios repositorios del mismo control de versiones siempre y cuando éste soporte replicación de repositorios.

Ejemplos



git





VISUAL SVN SERVER

Para poder hacer uso del control de versiones, antes necesitamos de un **repositorio** con el que sincronizar nuestra copia de trabajo. Para ello, vamos a montar uno de los repositorios más populares y utilizados, el conocido como **SubVersion (SVN)**.

El proceso de instalación es muy sencillo, el único aspecto clave que tenemos que tener en cuenta es marcar la opción de instalación **Visual SVN Server and Management Console**.

Una vez instalado, sólo necesitamos crear un **repositorio** (un directorio) y un **usuario** para acceder a él, una vez realizados esos sencillos pasos ya tendríamos al repositorio listo y dispuesto para ser usado.



Visual Studio

Team Foundation Server

Microsoft tiene a su disposición un sistema de control de versiones llamado **Team Foundation Server**, que utiliza una serie de bases de datos **SQL Server** para almacenar los datos y el **IIS** para publicarlos.

Este control de versiones es exclusivo de productos de Microsoft como Visual Studio, lo cual nos ofrece varias ventajas, como la perfecta integración con plataformas de trabajo como **Sharepoint**, pero también muchas restricciones, sobre todo si se trata de trabajo colaborativo, y cuestiones de licencias.

El propio Visual Studio tiene integrado el **cliente de TFS**, por lo que en caso de que se disponga de un **repositorio de TFS** se podría utilizar de un modo similar al de cualquier cliente de control de versiones.



ANKH-SVN

AnkhSVN es un cliente disponible como extensión para Visual Studio que nos permite trabajar con los repositorios de **SubVersion**.

[AnkhSVN2019 - Visual Studio Marketplace](#)



Git es un proyecto de código abierto

Funciona a la perfección en una amplia variedad de sistemas operativos e IDE

Arquitectura distribuida, es un ejemplo de DVCS (sistema de control de versiones distribuido)

- Rendimiento
- Seguridad
- Flexibilidad
- Control de versiones con Git



- Rendimiento

A diferencia de algunos programas de software de control de versiones, Git no se deja engañar por los nombres de los archivos a la hora de determinar cuál debería ser el almacenamiento y el historial de versiones del árbol de archivos; en lugar de ello, se centra en el contenido del propio archivo.

Al fin y al cabo, los archivos de código fuente se cambian de nombre, se dividen y se reorganizan con frecuencia. El formato de objeto de los archivos del repositorio de Git emplea una combinación de codificación delta (que almacena las diferencias de contenido) y compresión, y guarda explícitamente el contenido de los directorios y los objetos de metadatos de las versiones.

Su arquitectura distribuida también permite disfrutar de importantes ventajas en términos de rendimiento.



- Seguridad

Git se ha diseñado con la principal prioridad de conservar la integridad del código fuente gestionado.

El contenido de los archivos y las verdaderas relaciones entre estos y los directorios, las versiones, las etiquetas y las confirmaciones, todos ellos objetos del repositorio de Git, están protegidos con un algoritmo de hash criptográficamente seguro llamado "SHA1".

De este modo, se salvaguarda el código y el historial de cambios frente a las modificaciones accidentales y maliciosas, y se garantiza que el historial sea totalmente trazable.

Con Git, puedes tener la certeza de contar con un auténtico historial de contenido de tu código fuente.



- Flexibilidad

Uno de los objetivos clave de Git en cuanto al diseño es la flexibilidad.

Git es flexible en varios aspectos:

- La capacidad para varios tipos de flujos de trabajo de desarrollo no lineal
- Su eficiencia en proyectos tanto grandes como pequeños y en su compatibilidad con numerosos sistemas y protocolos.

Git se ha ideado para posibilitar la ramificación y el etiquetado como procesos de primera importancia (a diferencia de SVN) y las operaciones que afectan a las ramas y las etiquetas (como la fusión o la reversión) también se almacenan en el historial de cambios.

No todos los sistemas de control de versiones ofrecen este nivel de seguimiento.



Control de versiones con Git

Git es la mejor opción para la mayoría de los equipos de software actuales.

Es un software de control de versiones, su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

Existe la posibilidad de trabajar de forma remota y una opción es GitHub.

Tenemos nuestro directorio local

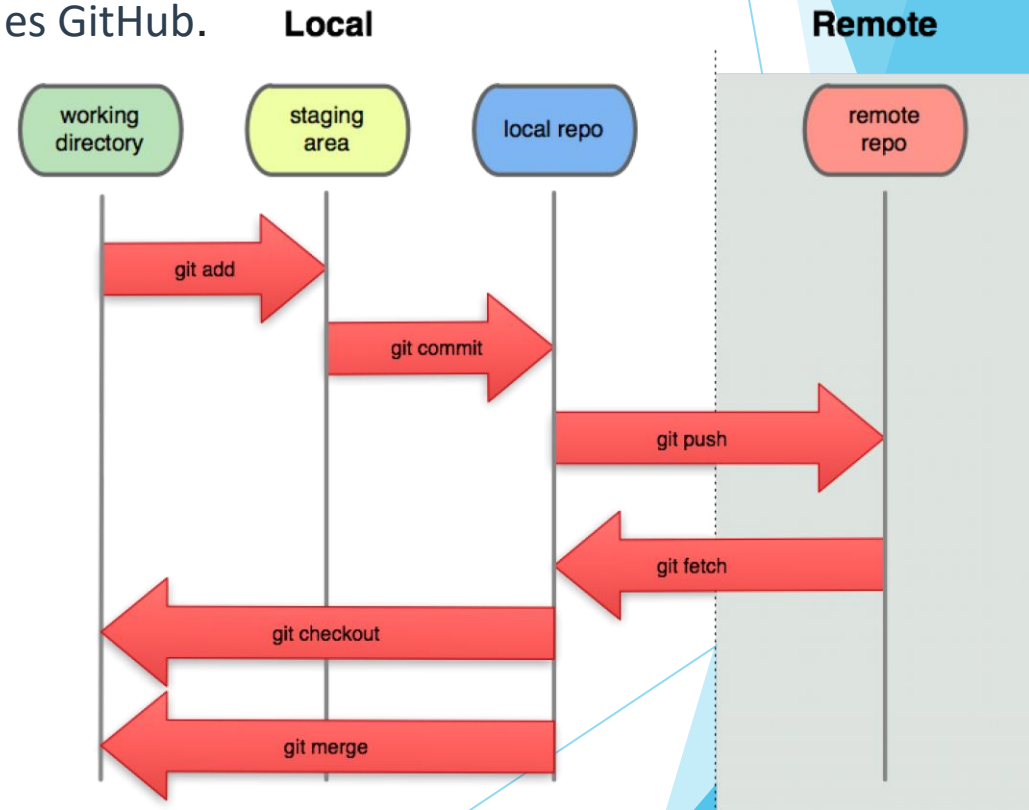
(una carpeta en nuestro pc) con muchos archivos,

Git nos irá registrando los cambios de archivos o códigos

cuando nosotros le indiquemos, así podremos viajar en el

tiempo retrocediendo cambios o restaurando versiones de

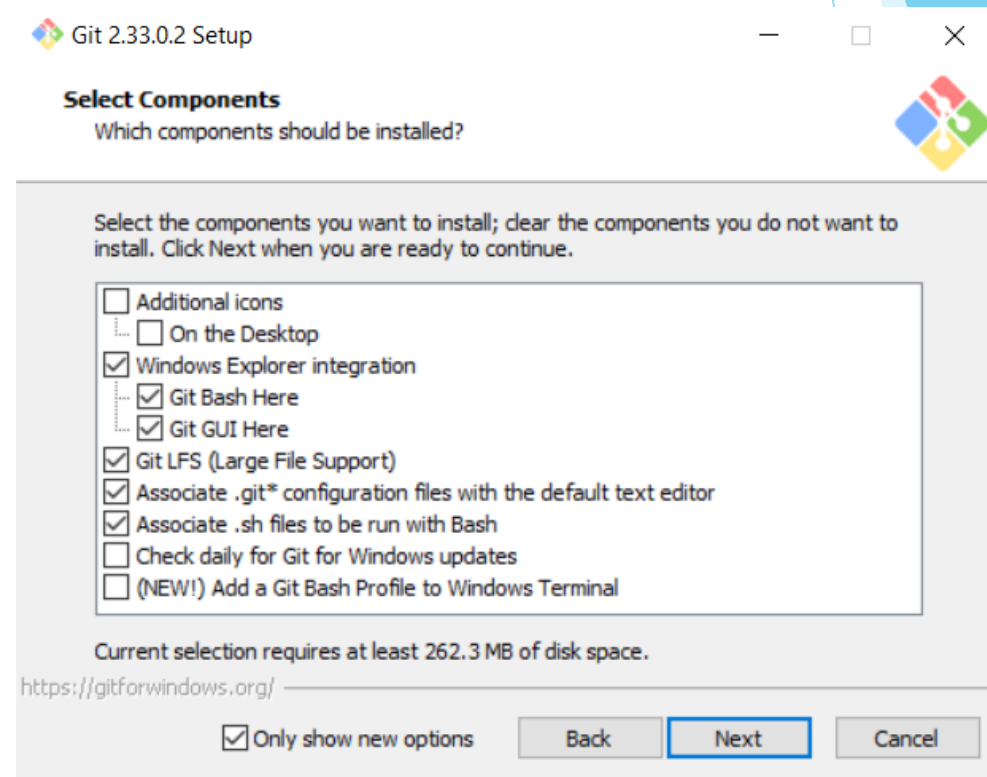
código, ya sea en Local o de forma Remota (servidor externo)

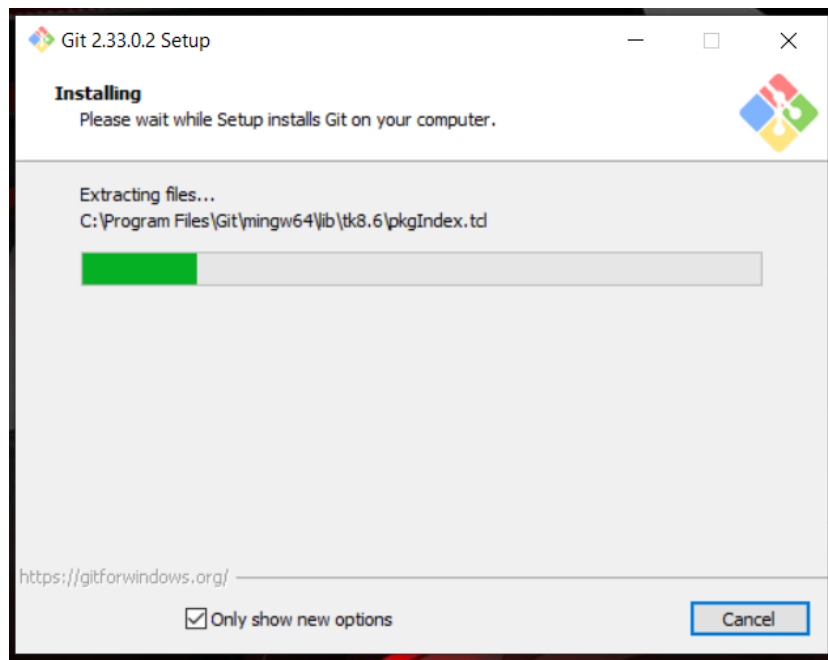
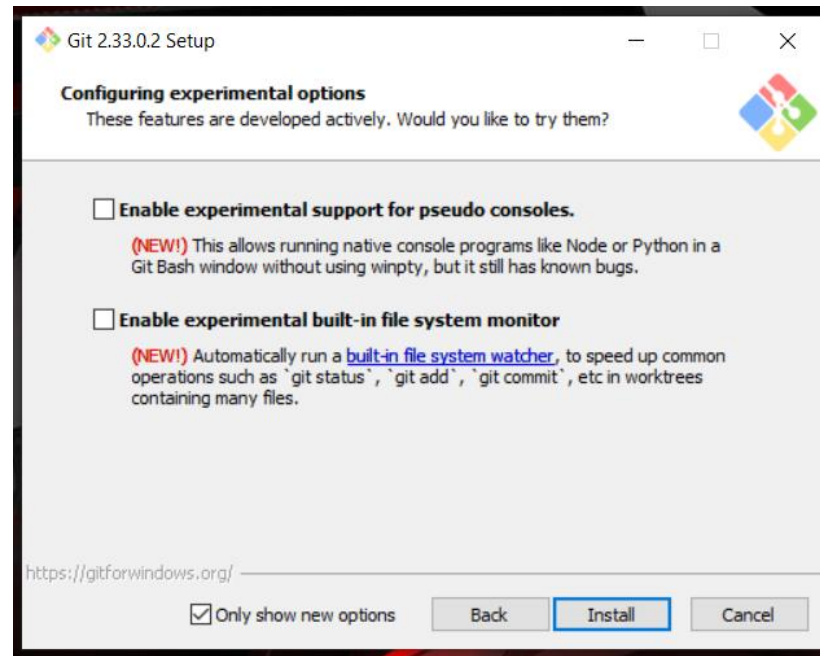
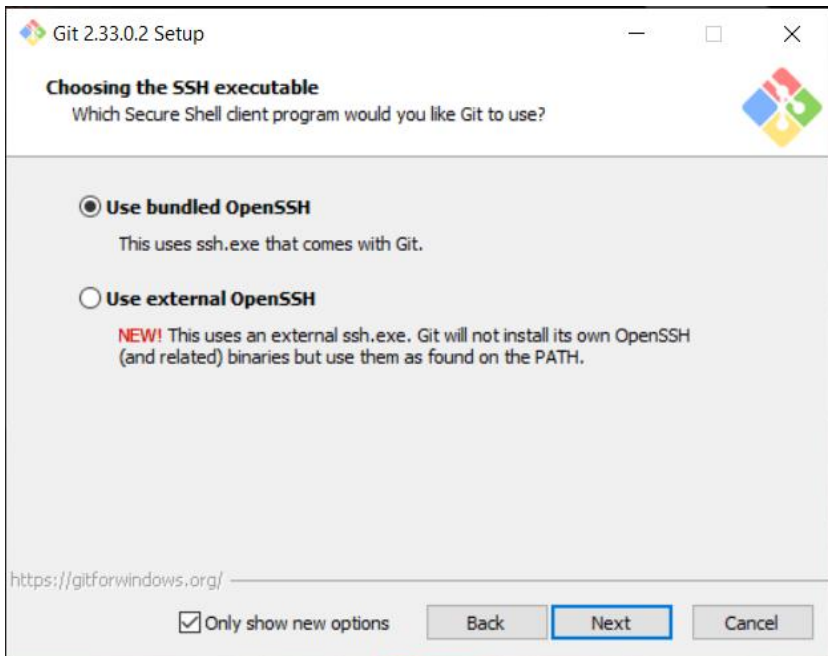


Instalación Git - Windows

[Git - Downloading Package \(git-scm.com\)](https://git-scm.com)

[Git for Windows](https://gitforwindows.org)





```
C:\Users\rafac>git version
git version 2.33.0.windows.2

C:\Users\rafac>
```

¿Qué es GitHub?

Es una plataforma de desarrollo colaborativo para alojar proyectos (en la nube) utilizando el sistema de control de versiones Git.

Además cuenta con una herramienta muy útil que es GitHub Pages donde podemos publicar nuestros proyectos estáticos (HTML, CSS y JS) .

Comandos básicos

Git versión

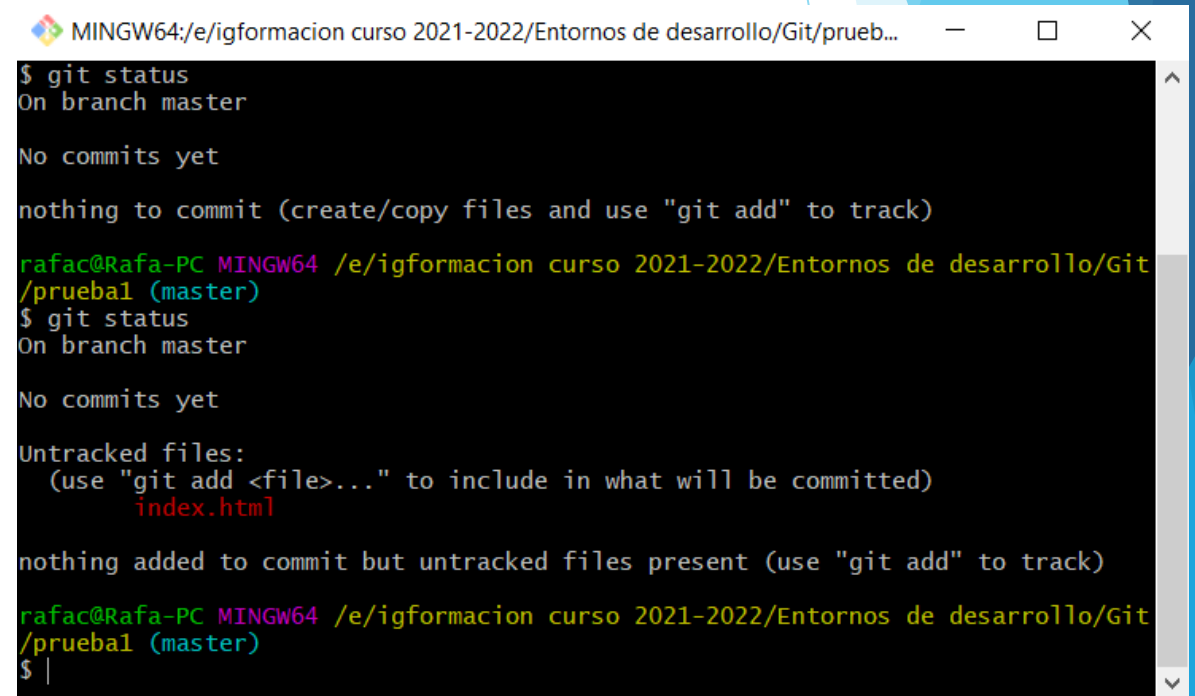
Git help

Registrar nuevo usuario asociado a git:

Git config --global user.name "Jhon"

Git config --global user.email "jhon@doe.com"

\$ git config --global user.name "John Doe" git config --global user.email "john@doe.com"



```
MINGW64:/e/igformacion curso 2021-2022/Entornos de desarrollo/Git/prueb...
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

rafac@Rafa-PC MINGW64 /e/igformacion curso 2021-2022/Entornos de desarrollo/Git
/prueba1 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

nothing added to commit but untracked files present (use "git add" to track)

rafac@Rafa-PC MINGW64 /e/igformacion curso 2021-2022/Entornos de desarrollo/Git
/prueba1 (master)
$ |
```

Instalación Git - Linux

Desde tu shell, instala Git mediante apt-get:

```
$ sudo apt-get update
```

```
$ sudo apt-get install git
```

Escribe `git --version` para verificar la versión y que esta se haya instalado correctamente:

```
$ git --version
```

```
git version 2.9.2
```

Configura tu nombre de usuario y tu correo electrónico de Git mediante los siguientes comandos

Esta información se asociará a todas las confirmaciones que crees:

```
$ git config --global user.name "Jhon Doe"
```

```
$ git config --global user.email "jhon@doe.com"
```