Git

**Git** (pronunciado "guit" es un software de [control de versiones](https://es.wikipedia.org/wiki/Control_de_versiones) diseñado por [Linus Torvalds](https://es.wikipedia.org/wiki/Linus_Torvalds), pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de [código fuente](https://es.wikipedia.org/wiki/C%C3%B3digo_fuente). Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o [front end](https://es.wikipedia.org/wiki/Front-end_y_back-end) como [Cogito](https://es.wikipedia.org/wiki/Cogito_(software)) o [StGIT](https://es.wikipedia.org/w/index.php?title=StGIT&action=edit&redlink=1).  ​Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena.  ​Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de [programación](https://es.wikipedia.org/wiki/Programaci%C3%B3n) del [núcleo Linux](https://es.wikipedia.org/wiki/N%C3%BAcleo_Linux).

El [mantenimiento del software](https://es.wikipedia.org/wiki/Mantenimiento_de_Software) Git está actualmente (2009) supervisado por Junio Hamano, quien recibe contribuciones al código de alrededor de 280 programadores. En cuanto a derechos de autor Git es un [software libre](https://es.wikipedia.org/wiki/Software_libre) distribuible bajo los términos de la versión 2 de la [Licencia Pública General de GNU](https://es.wikipedia.org/wiki/GNU_General_Public_License).

Que implica el mantenimiento de software Git

El ciclo de vida de software tiene varias etapas: Análisis, Diseño, manufactura, pruebas implantación y mantenimiento.

El despliegue o implantación de un programa se basa en tres acciones entrega, soporte y retroalimentación. Como el software moderno es evolutivo se despliega varias veces, por lo que cada que se entrega se agregan nuevas funcionalidades y se debe de dar la documentación adecuada y el soporte necesario para todas las funciones dadas.

El mantenimiento es el proceso de efectuar cambios en un software después de entregado, el cual se presenta cuando el software se adapta a un nuevo ambiente, se corrigen errores, el cliente presenta funciones nuevas o cuando la aplicación presenta una reingeniería.

Por lo que existen cuatro tipos de mantenimiento:

Mantenimiento correctivo

Mantenimiento adaptativo

Mantenimiento de perfeccionamiento

Mantenimiento preventivo

Características

El diseño de Git se basó en [BitKeeper](https://es.wikipedia.org/wiki/BitKeeper) y en [Monotone](https://es.wikipedia.org/wiki/Monotone).  Originalmente fue diseñado como un motor de sistema de control de versiones de bajo nivel sobre el cual otros podrían codificar interfaces frontales, tales como [Cogito](https://es.wikipedia.org/wiki/Cogito_(software)) o [StGIT](https://es.wikipedia.org/w/index.php?title=StGIT&action=edit&redlink=1). Desde ese entonces hasta ahora el núcleo del proyecto Git se ha vuelto un sistema de control de versiones completo, utilizable en forma directa.

[**Linus Torvalds**](https://es.wikipedia.org/wiki/Linus_Torvalds) buscaba un sistema distribuido que pudiera usar en forma semejante a BitKeeper, pero ninguno de los sistemas bajo software libre disponibles cumplía con sus requerimientos, especialmente en cuanto a desempeño. El diseño de Git mantiene una enorme cantidad de código distribuida y gestionada por mucha gente, que incide en numerosos detalles de rendimiento, y de la necesidad de rapidez en una primera implementación.

Entre las características más relevantes se encuentran:

* Fuerte apoyo al desarrollo no lineal, por ende rapidez en la gestión de ramas y mezclado de diferentes versiones. Git incluye herramientas específicas para navegar y visualizar un historial de desarrollo no lineal. Una presunción fundamental en Git, es que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente, conforme se pasa entre varios programadores que lo revisan.
* Gestión distribuida al igual que Darcs, BitKeeper, Mercurial, SVK, Bazaar y Monotone, Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramas adicionales y pueden ser fusionados en la misma manera que se hace con la rama local.
* Los almacenes de información pueden publicarse por [HTTP](https://es.wikipedia.org/wiki/HTTP), [FTP](https://es.wikipedia.org/wiki/File_Transfer_Protocol), [rsync](https://es.wikipedia.org/wiki/Rsync) o mediante un protocolo nativo, ya sea a través de una conexión TCP/IP simple o a través de cifrado [SSH](https://es.wikipedia.org/wiki/SSH). Git también puede emular servidores [CVS](https://es.wikipedia.org/wiki/CVS), lo que habilita el uso de clientes CVS pre-existentes y módulos IDE para CVS pre-existentes en el acceso de repositorios Git.
* Los repositorios Subversion y svk se pueden usar directamente con git-svn.
* Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución.
* Todas las versiones previas a un cambio determinado, implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio (denominado autenticación criptográfica de historial). Esto existía en [Monotone](https://es.wikipedia.org/wiki/Monotone).
* Resulta algo más caro trabajar con ficheros concretos frente a proyectos, eso diferencia el trabajo frente a CVS, que trabaja con base en cambios de fichero, pero mejora el trabajo con afectaciones de código que concurren en operaciones similares en varios archivos.
* Los renombrados se trabajan basándose en similitudes entre ficheros, aparte de nombres de ficheros, pero no se hacen marcas explícitas de cambios de nombre con base en supuestos nombres únicos de nodos de sistema de ficheros, lo que evita posibles y desastrosas coincidencias de ficheros diferentes en un único nombre.
* Realmacenamiento periódico en paquetes (ficheros). Esto es relativamente eficiente para escritura de cambios y relativamente ineficiente para lectura si el reempaquetado (con base en diferencias) no ocurre cada cierto tiempo.