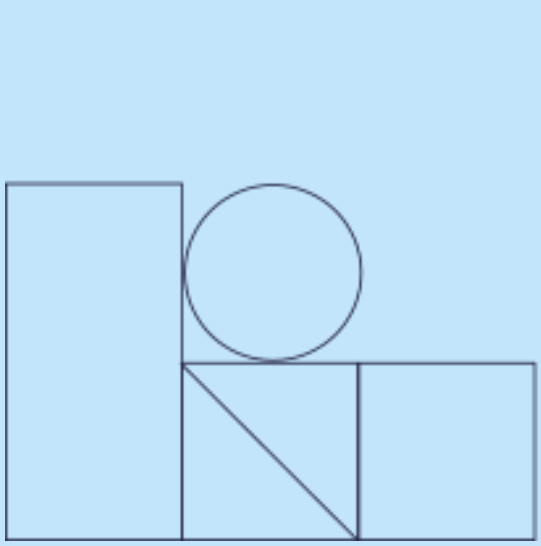
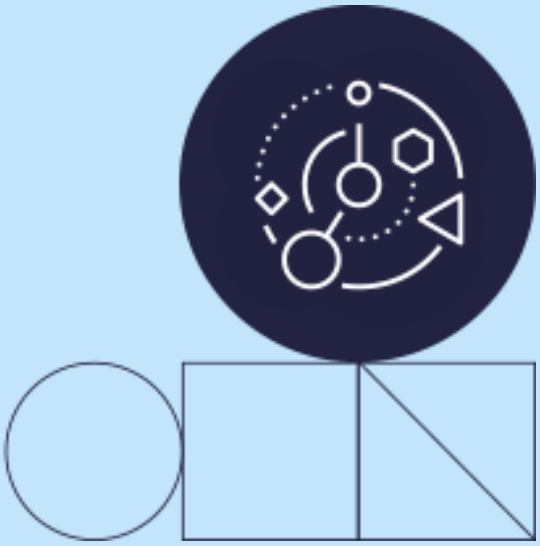
IBM SkillsBuild

IBM

Git y Github

Qué es Git y Github





Índice

Introducción

Git es un proyecto de código abierto que se inició en 2005 y creció hasta convertirse en uno de los VCS más populares del mercado: cerca del **87% de los desarrolladores** utilizan Git para sus proyectos.

Git es un sistema de control de versiones. Una herramienta que sirve para almacenar versiones de nuestro código.

A diferencia de los **sistemas de control de versiones centralizados**, *Git* ofrece **ramas de características**. Esto significa que cada ingeniero de software en el equipo puede dividir una rama de características que proporcionará un repositorio local aislado para hacer cambios en el código.

Las ramas de características no afectan a la rama maestra, que es donde se encuentra el código original del proyecto. Una vez que se hayan realizado los cambios y el código actualizado esté listo, la rama de características puede fusionarse de nuevo con la rama maestra, que es la forma en que se harán efectivos los cambios en el proyecto.

Además, nos permite mantener un historial de versiones, es decir, varias versiones en paralelo de un mismo software, arreglar simultáneamente bugs, etc. Cuando varias personas trabajan a la vez en un mismo software es cuando realmente *GIT* adquiere especial utilidad.

Antes de la aparición de esta herramienta la forma de trabajar era guardando manualmente en local las diferentes versiones que teníamos de nuestro código. De forma que almacenábamos en nuestro disco duro la versión V1.0 del programa, la V2.0, etc. De esta forma podíamos tener un historial de versiones de un código y a la hora de trabajar en grupo, cada equipo tenía una forma de compartir contenidos e implementar el código.

A menudo los programadores se “pisaban” el trabajo unos a otros y siempre estaba la amenaza de que un mal paso, o un error por parte de un programador, acabase borrando el trabajo de todos.

GIT realiza por nosotros una instantánea de nuestro proyecto tal y como está hasta la fecha de forma que, si se presenta algún problema, podemos volver atrás y recargar dicha instantánea (u otras anteriores) y no perder así el trabajo. A dicha instantánea *GIT* le adjunta un código de registro alfanumérico y la descripción que nosotros le digamos. De forma que tendremos un historial de versiones de nuestro código con sus respectivas descripciones para volver atrás en caso necesario.

Además, permite que varios programadores trabajen a la vez sobre el mismo código, de forma que el trabajo de uno no interfiera en el de los demás. Es capaz de detectar que dos o más programadores están trabajando en la misma parte del código y gestiona mediante ramas la fusión de ambos trabajos de forma que no existan conflictos entre ambos códigos.

Git dispone de tres "áreas" diferentes para nuestro código:

- **Directorio de trabajo**: el área en la que realizará todo nuestro trabajo (creación, edición, eliminación y organización de archivos).
- **Área de almacenamiento**: el área donde se enumerarán los cambios que hayamos realizado en el directorio de trabajo.
- **Repositorio**: donde *Git* almacena permanentemente los cambios que hemos realizado, como diferentes versiones del proyecto.

¿Por qué se utiliza Git?

Git y *Github* se utilizan en la vida diaria de las personas que crean software por una razón muy simple: tener una manera fácil de administrar el código fuente de la aplicación, del sistema y del producto.

En un equipo pequeño, algunas personas todavía intentan cuidar estos archivos de formas algo cuestionables: compartir directorios en la red, usar herramientas como *Dropbox* o mantener todo en un servidor *FTP*. Prácticas totalmente desaconsejadas hoy en día.

No basta con poder acceder al código de otros colaboradores. Necesitamos mantener el **historial** de nuestros archivos. Más: de nuestras modificaciones, ya que a menudo cambiamos archivos en grupo, en un solo movimiento (un ***commit***). De esa manera, podemos volver atrás y recuperar el estado del sistema como estaba ayer o el año pasado, y comparar los cambios, encontrar un bug, estudiar optimizaciones...etc.

Todos nuestros archivos, así como sus historiales, están en un **repositorio** y existen varios sistemas que administraban dichos repositorios, como *CVS* y *SVN*.

Git es una alternativa que funciona aún más interesante: se distribuye y todos tienen una copia completa del repositorio, no solo el "servidor principal".

***Git* es un sistema de control de versión distribuido y ampliamente adoptado**. *Git* nació y tomó el espacio de otros sistemas de control. Su principal creador es el mismo que el de *Linux*: Linus Torvalds y se ganó el corazón de las personas que trabajan con *open source*.

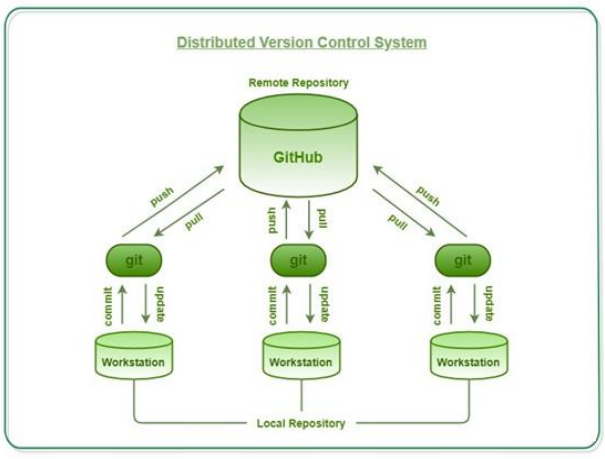
Características de Git

Sistema distribuido

Los sistemas distribuidos son aquellos que permiten a los usuarios realizar trabajos en un proyecto de todo el mundo. Un sistema distribuido contiene un repositorio central al que pueden acceder muchos colaboradores remotos mediante un sistema de control de versiones. *Git* es uno de los sistemas de control de versiones más populares que se utilizan en la actualidad.

Tener un servidor central da como resultado un problema de pérdida de datos o desconexión de datos en caso de una falla del sistema del servidor central. Para abordar este tipo de situación, *Git* refleja todo el repositorio en cada instantánea de la versión que está extrayendo el usuario. En este caso, si el servidor central falla, la copia de los repositorios se puede recuperar de los usuarios que han descargado la última instantánea del proyecto.

Al tener un sistema distribuido, *Git* permite a los usuarios trabajar simultáneamente en el mismo proyecto, sin interferir con el trabajo de otros. Cuando un usuario en particular termina con su parte del código, envía los cambios al repositorio y estos cambios se actualizan en la copia local de todos los demás usuarios remotos que extraen la última copia del proyecto.



Compatibilidad

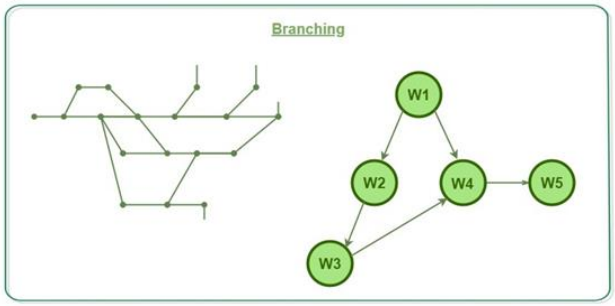
Git es compatible con todos los sistemas operativos que se están utilizando en estos días. Los repositorios de *Git* también pueden acceder a los repositorios de otros sistemas de control de versiones como *SVN*, *CVK*, etc. Git puede acceder directamente a los repositorios remotos creados por estos *SVN*. Por lo tanto, los usuarios que no estaban usando *Git* en primer lugar también pueden cambiar a *Git* sin pasar por el proceso de copiar sus archivos de los repositorios de otros *VCS* en *Git-VCS*. Git también puede acceder a los repositorios centrales de otros *VCS*. Por lo tanto, uno puede trabajar en *Git-SVN* y usar el repositorio central como el mismo. Git tiene una emulación de servidor *CVS*, que permite el uso de clientes *CVS* existentes y complementos *IDE* para acceder a los repositorios de *Git*.

Desarrollo no lineal

Git permite a los usuarios de todo el mundo realizar operaciones en un proyecto de forma remota. Un usuario puede seleccionar cualquier parte del proyecto y realizar la operación requerida y luego actualizar el proyecto. Esto se puede hacer mediante el comportamiento de desarrollo no lineal del *Git*. Git admite la ramificación y la fusión rápidas e incluye herramientas específicas para visualizar y navegar por un historial de desarrollo no lineal. Una suposición importante en *Git* es que un cambio se fusionará con más frecuencia de lo que está escrito. *Git* registra el estado actual del proyecto en forma de árbol. Se puede agregar una nueva rama al árbol en cualquier momento y se fusionará con el proyecto final una vez que esté completo.

Ramificación

Git permite a sus usuarios trabajar en una línea que corre paralela a los archivos principales del proyecto. Estas líneas se llaman ramas (o *branchs*). Las sucursales en *Git* proporcionan una función para realizar cambios en el proyecto sin afectar la versión original. La rama maestra de una versión siempre contendrá el código de calidad de producción. Cualquier característica nueva se puede probar y trabajar en las ramas y, además, se puede fusionar con la rama maestra.
La ramificación y la fusión se pueden hacer muy fácilmente con la ayuda de algunos comandos de *Git*. Una sola versión de un proyecto puede contener *n* número de ramas según los requisitos del usuario.



Ligero

Git almacena todos los datos del repositorio central en el repositorio local mientras se realiza la clonación. Puede haber cientos de usuarios trabajando en el mismo proyecto y, por lo tanto, los datos en el repositorio central pueden ser muy grandes. Uno podría estar preocupado de que clonar tantos datos en máquinas locales pueda resultar en fallas en el sistema, pero *Git* ya se ha ocupado de ese problema. *Git* sigue el criterio de compresión sin pérdidas que comprime los datos y los almacena en el repositorio local ocupando un espacio mínimo. Siempre que sea necesario para estos datos, sigue la técnica inversa y ahorra mucho espacio en la memoria.

Velocidad

Dado que *Git* almacena todos los datos relacionados con un proyecto en el repositorio local mediante el proceso de clonación, es muy eficiente obtener datos del repositorio local en lugar de hacer lo mismo desde el repositorio remoto. *Git* es muy rápido y escalable en comparación con otros sistemas de control de versiones, lo que da como resultado el manejo de grandes proyectos de manera eficiente.
El poder de obtención de un repositorio local es aproximadamente 100 veces más rápido de lo que es posible con el servidor remoto.
Según una prueba realizada por *Mozilla*, *Git* es un orden de magnitud más rápido, que es aproximadamente 10 veces más rápido que otras herramientas *VCS*. Esto se debe a que *Git* está escrito en lenguaje *C*, a diferencia de otros lenguajes, muy parecido al lenguaje de máquina y, por lo tanto, hace que el procesamiento sea muy rápido.

Código abierto

Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños hasta muy grandes, con velocidad y eficiencia. Se llama de código abierto porque proporciona la flexibilidad de modificar su código fuente de acuerdo con las necesidades del usuario. A diferencia de otros sistemas de control de versiones que brindan funcionalidades de pago, como el espacio de repositorio, privacidad de códigos, precisión y velocidad, etc. *Git* es todo un software de código abierto que proporciona estas funcionalidades de forma gratuita e incluso mejor que otros sistemas de pago.
Al ser *Git* de código abierto, permite que varias personas trabajen en el mismo proyecto al mismo tiempo y colaboren entre sí de manera muy fácil y eficiente. Por lo tanto, se considera que *Git* es el mejor sistema de control de versiones disponible en la actualidad.

Fiabilidad

Al proporcionar un repositorio central que se clona cada vez que un usuario realiza la operación de extracción, los datos del repositorio central siempre se respaldan en el repositorio local de cada colaborador. Por lo tanto, en caso de falla del servidor central, los datos nunca se perderán, ya que cualquiera de las máquinas locales del desarrollador puede recuperarlos fácilmente. Una vez que el servidor central está completamente reparado, cualquiera de los múltiples colaboradores puede recuperar los datos. Existe una probabilidad muy baja de que los datos no estén disponibles con ningún desarrollador porque el que ha trabajado en el proyecto por última vez definitivamente tendrá la última versión del proyecto en su máquina local. Igual es el caso al final del cliente. Si un desarrollador pierde sus datos debido a alguna falla técnica o cualquiera de los motivos imprevistos, pueden extraer fácilmente los datos del repositorio central y obtener la última versión de los mismos en su máquina local. Por lo tanto, enviar datos al repositorio central hace que *Git* sea más fiable para trabajar.

Seguro

Git mantiene un registro de todas las confirmaciones realizadas por cada uno de los colaboradores en la copia local del desarrollador. Se mantiene un archivo de registro y se envía al repositorio central cada vez que se realiza la operación de inserción. Por lo tanto, si surge un problema, el desarrollador puede rastrearlo y manejarlo fácilmente.
Git usa SHA1 para almacenar todos los registros en forma de objetos en el *Hash*. Cada objeto colabora entre sí con el uso de estas claves Hash. SHA1 es un algoritmo criptográfico que convierte el objeto de confirmación en un código hexadecimal de 14 dígitos. Ayuda a almacenar el registro de todas las confirmaciones realizadas por cada uno de los desarrolladores. De ahí que sea fácilmente diagnosticable aquello que ha producido un fallo en el código.

Económico

Git se publica bajo la Licencia para el público general (GPL) y, por lo tanto, está disponible de forma gratuita. *Git* crea un clon del repositorio central en la máquina local y, por lo tanto, todas las operaciones se realizan en la máquina local del desarrollador antes de enviarlo al repositorio central. La inserción se realiza sólo después de que la versión en la máquina local esté funcionando perfectamente y esté lista para ser insertada en el servidor central. No se experimenta con los archivos del servidor central. Esto ayuda a ahorrar mucho dinero en servidores costosos. Todo el trabajo pesado se realiza en el lado del cliente y, por lo tanto, no es necesario tener máquinas pesadas para el lado del servidor.

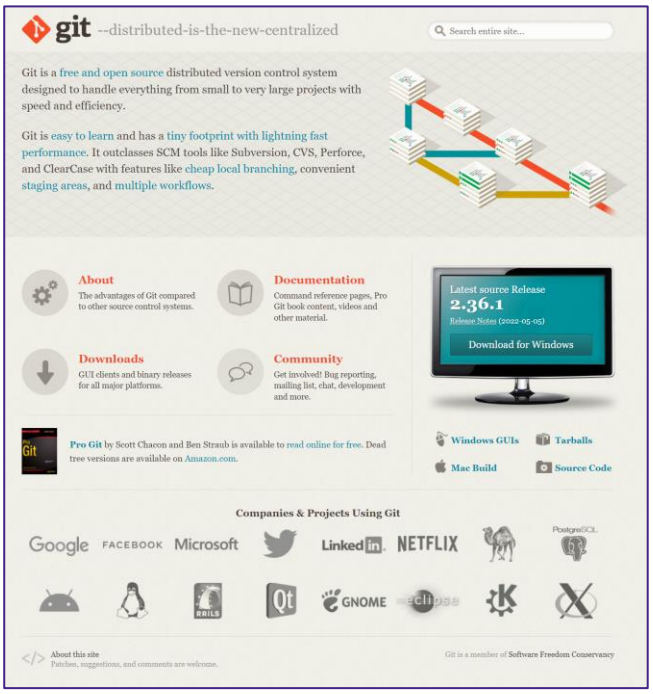
Estas características han convertido a *Git* en el sistema de control de versiones más fiable y más utilizado de todos los tiempos. El repositorio central de *Git* se denomina *GitHub*. *GitHub* permite realizar todas las operaciones *Push* (subir archivos) y *Pull* (bajar archivos) con el uso de *Git*. Estos comando y otros complementarios se verán en posteriores temas.

Instalación de GIT

Descargaremos la versión en curso de *GIT* desde su página oficial:

https://git-scm.com/

Las versiones disponibles son *Mac, Windows, Linux/Unix*.



Una vez descargado e instalado, tendremos disponibles varias herramientas de *GIT*, entre las que destaca la consola *Git Bash*:

Con *GIT* podremos trabajar tanto desde la consola, como desde nuestro propio *IDE*, ya que hoy en día, la mayoría de *IDEs* presentan integración con *GIT*, bien de forma nativa, como es el caso de *Visual Studio Code* o bien mediante *plug-ins*.

¿Qué es el control de versiones?

Hemos mencionado anteriormente que *Git* es una utilidad de gestión de versiones. Pero ¿qué es exactamente el control de versiones?

El control de versiones es un sistema que ayuda a rastrear y gestionar los cambios realizados en un archivo o conjunto de archivos. Utilizado principalmente por ingenieros de software para hacer un seguimiento de las modificaciones realizadas en el código fuente, el sistema de control de versiones les permite analizar todos los cambios y revertirlos sin repercusiones si se comete un error.

En otras palabras, el control de versiones permite a los desarrolladores trabajar en proyectos simultáneamente. Les permite hacer tantos cambios como necesiten sin infringir o retrasar el trabajo de sus colegas.

Si esos cambios en el código fuente arruinan el proyecto cuando se implementan, GitHub hace que sea fácil revertirlos con unos pocos clics, y se recuperará la versión anterior del proyecto.

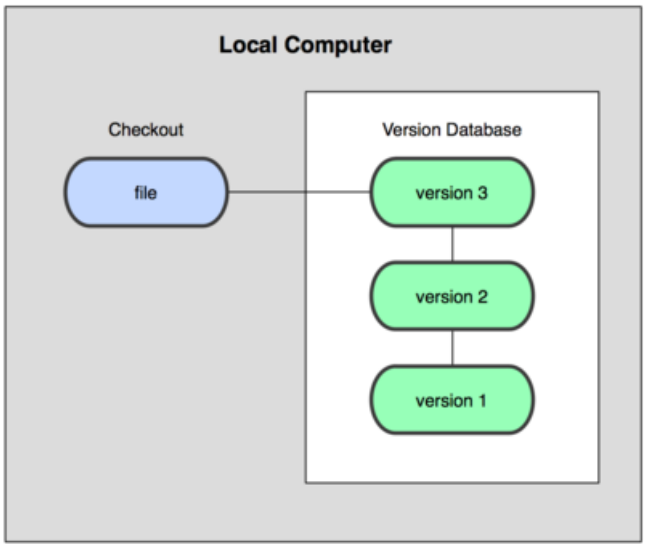
En síntesis, el control de versiones elimina los riesgos y el miedo a cometer demasiados errores. En cambio, proporciona la libertad de colaborar y desarrollar sin demasiadas preocupaciones.

Clasificación de los sistemas de control de versiones

Dentro del campo de control de versiones, nos podemos encontrar de distintos tipos. Veamos a continuación una clasificación de los más importantes.

Sistemas de control de versiones locales

Uno de los métodos más utilizados por la gente a la hora de realizar algún tipo de control de versión de sus cambios, consistía en copiar en un directorio de su equipo local el archivo que iba a ser modificado indicando la fecha de modificación, para que en caso de error se supiese cuál era la última versión guardada.

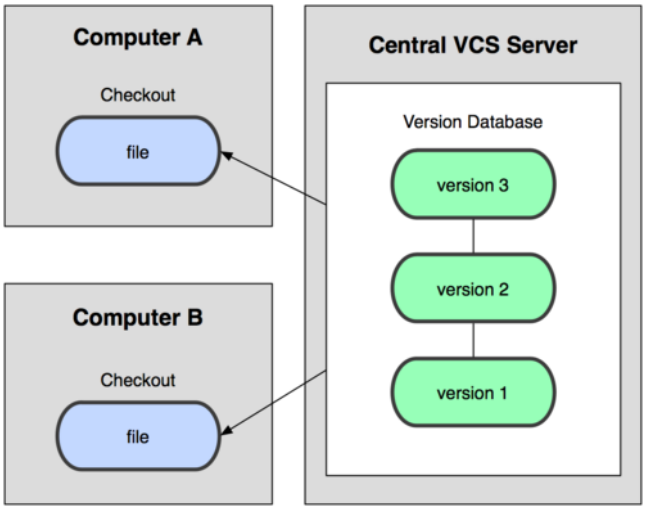


Este sistema podría valer para el desarrollo de una aplicación pequeña, pero ofrece ciertos problemas como el de no recordar dónde hemos guardado la copia o simplemente olvidarnos de hacerla.

Para hacer frente a estos problemas, los programadores desarrollaron los conocidos como *VCSs* locales, que consistían en una base de datos donde se llevaba un registro de los cambios realizados sobre los archivos.

Sistemas de control de versiones centralizados

El sistema comentado anteriormente nos puede valer en el caso de que estemos trabajando solos, pero en un proyecto suelen intervenir varias personas y cada una de ellas se encarga de realizar una tarea determinada. En este caso, es necesario poder contar con un sistema colaborador y es aquí donde entran en juego los sistemas de control de versiones centralizados.

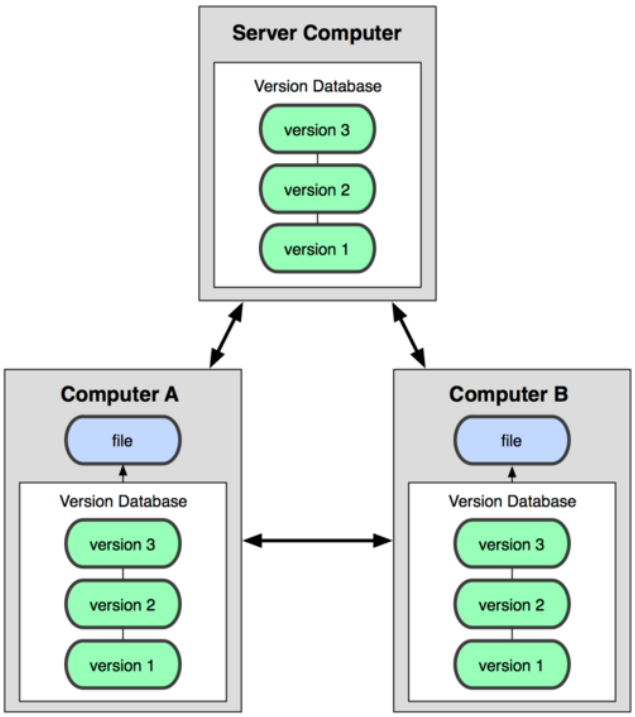


En estos sistemas nos encontramos un único servidor que contiene todos los archivos versionados, y los usuarios que forman parte del proyecto se los pueden descargar desde ese servidor centralizado.

Este sistema tiene un problema muy claro, y es que, al utilizar un único servidor centralizado, en caso de problema en ese servidor toda la información se podría perder.

Sistemas de control de versiones distribuidas

A diferencia del caso anterior, en estos sistemas no tenemos un único servidor que mantenga la información del proyecto, sino que cada usuario contiene una copia completa del proyecto de forma local. De esta forma, si un servidor muere, cualquiera de los repositorios de los clientes se podría utilizar para restaurar el servidor.



Git pertenece a este tipo de sistemas.

Github

GitHub tiene mucho que ver con *Git*. ***GitHub* es una plataforma para administrar tu código y crear un entorno colaborativo entre desarrolladores, utilizando *Git* como sistema de control.** Facilitará el uso de *Git*, ocultando algunos detalles de configuración más complicados.

Muchas personas pueden pensar que *Git* y *GitHub* son lo mismo, ya que la mayoría de las ocasiones están estrechamente relacionadas, pero a la hora de la verdad, son cosas totalmente distintas.

GitHub se puede definir como un servidor donde alojar los repositorios de los proyectos, añadiendo funcionalidades extra para la gestión del proyecto y del código fuente. A diferencia del proyecto *Git*, éste se trata de un servicio comercial, ya que, aunque tiene una parte pública gratuita, cuenta con la desventaja de que todo el código que subamos estará disponible para cualquier persona. Si queremos decidir quién puede tener acceso a nuestro repositorio, entonces sería necesario pasarse a la modalidad de pago.

El sistema web que tiene nos permite cambiar archivos online, aunque no es muy recomendable, ya que no dispondremos de las ventajas de un editor, un entorno de desarrollo y pruebas. Para comunicarnos con GitHub y modificar los archivos de nuestro repositorio, podemos usar la línea de comando, usando el comando *git* y sus directivas de *commit*, *pull* y *push*. Se verá con detalle más adelante.

Entre las principales características que nos ofrece *GitHub* podemos destacar:

- Una Wiki que opera con *Git* para el mantenimiento de las distintas versiones de las páginas.
- Sistema de seguimiento de problemas. Se trata de un sistema muy parecido al tradicional ticket, donde cualquier miembro del equipo o persona (si nuestro repositorio es público) puede abrir una consulta o sugerencia que se quiera hacer.
- Herramienta de revisión de código. Permite añadir anotaciones en cualquier punto de un fichero.
- Visor de ramas que permite comparar los progresos realizados en las distintas ramas de nuestro repositorio.

¿Por qué GitHub es tan popular?

GitHub aloja más de **100 millones** de repositorios, la mayoría de los cuales son proyectos de código abierto. Esta estadística revela que *GitHub* se encuentra entre los **clientes *Git GUI*** más populares y es utilizado por varios profesionales y **grandes empresas**, como Hostinger.

Esto se debe a que *GitHub* es una plataforma de gestión y organización de proyectos basada en la nube que incorpora las funciones de control de versiones de *Git*. Es decir, que todos los usuarios de *GitHub* pueden rastrear y gestionar los cambios que se realizan en el código fuente en tiempo real, a la vez que tienen acceso a todas las demás funciones de Git disponibles en el mismo lugar.

Además, la interfaz de usuario de *GitHub* es más fácil de usar que la de *Git*, lo que la hace accesible para personas con pocos o ningún conocimiento técnico. Esto significa que se puede incluir a más miembros del equipo en el progreso y la gestión de un proyecto, haciendo que el proceso de desarrollo sea más fluido.

Diferencia entre CVS y GitHub

Sistema de versiones concurrentes (CVS)

El sistema de versiones concurrentes es un sistema de control de versiones funcional desarrollado por Dick Grune como una serie de scripts de *shell*. Esto ayuda a los equipos a estar conectados a los cambios que se miden en un repositorio cuando se trabaja en software. Esta herramienta se utilizó como sistema de control de versiones durante mucho tiempo.

Es una herramienta de software fiable, pero con nuevos desafíos, otras alternativas hicieron que se usase cada vez menos.

A continuación, se muestran algunas características de *CVS*:

- Es uno de los sistemas de control de versiones confiables que existen.
- No permite cometer errores.
- Los guiones están escritos en formato *RCS*.
- El usuario solo puede almacenar archivos directamente en el repositorio.

Ventajas

- *CVS* es uno de los software de control de versiones más fiables.
- Los cambios se confirman solo cuando hay un cambio completo

Desventajas

- Los cambios de *CVS* requieren mucho tiempo.
- *CVS* no confirma si hay un error en la confirmación.

GitHub

GitHub es una plataforma de alojamiento de repositorios que presenta colaboración y control de acceso. Es una herramienta de control de versiones para que los programadores procesen los errores juntos para contribuir y alojar proyectos de código abierto. *GitHub* está diseñado para los desarrolladores y los usuarios registrados pueden usar *GitHub* para contribuir, pero los usuarios no registrados pueden ver los repositorios.

A continuación, se muestran algunas características de *GitHub*:

- Especifica hitos y etiquetas a los proyectos.
- Las páginas de *GitHub* nos permiten publicar y alojar sitios web dentro de *GitHub*.
- Se permite la vista de comparación entre ramas.
- Permite integraciones de *API* de terceros para el seguimiento de errores y el alojamiento en la nube.

Ventajas

- Nos ayuda a almacenar los datos en forma de metadatos.
- Se utiliza para compartir el trabajo frente al público.

Desventajas

- Almacenar archivos grandes en *GitHub* lo hace lento.
- Solo admite el control de versiones de *Git*.

Diferencias entre CVS y GitHub

Parámetro	Cvs	Github
Desarrollado por	CVS fue desarrollado por Dick Grune.	GitHub fue desarrollado por Chris Wanstrath, Tom Preston-Werner, PJ Hyett y Scott Chacon.
Fuente abierta	Es de código abierto y se publica con la licencia pública general GNU.	GitHub no es de código abierto.
Confirmar ubicación	El repositorio está comprometido en el servidor central.	El repositorio está comprometido en el repositorio local.
Repositorio de clonación	CVS tiene una función para clonar el repositorio, pero requiere GIT.	GitHub permite al usuario clonar el repositorio.
Navegación	CVS no permite la navegación en el repositorio.	GitHub permite al usuario navegar por la usabilidad.
Compromiso de proyecto	CVS detiene la confirmación del error encontrado en un nodo.	GitHub permite la confirmación en el repositorio y el desarrollador corrige el error.