



Introducción a los sistemas de control de versiones GIT y GITHUB


¿Que son los sistemas de control de versiones?

Un sistema de control de versiones es el encargado de registrar los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que en cualquier momento se puedan recuperar versiones específicas. En ese orden de ideas los sistemas de control de versiones permiten restablecer versiones anteriores de un archivos o proyecto, comparar cambios a lo largo del desarrollo y tener estricto control sobre las modificaciones implementadas.

De acuerdo con lo anterior y para mencionar un ejemplo, contratiempos como perder un archivo, no representan un problema cuando se trabaja con un sistema de control de versiones ya que resulta sencillo recuperarlos.

El control de versiones se realiza principalmente en la industria informática para controlar las distintas versiones del código fuente dando lugar a los sistemas de control de código fuente. Sin embargo, los mismos conceptos son aplicables a otros ámbitos como documentos, imágenes, sitios web, etc.

Entre otras podemos mencionar las siguientes ventajas:

- ✓ Rastrear el desarrollo y los cambios de un archivo, de una manera que puedas entender posteriormente.
 - ✓ Registrar los cambios que realizados y experimentar con versiones distintas de un archivo al mismo tiempo que conservas la más antigua.
 - ✓ Fusionar dos versiones de un archivo y administrar los conflictos existentes entre distintas versiones.
 - ✓ Revertir cambios y restablecer versiones anteriores de un archivo.
- 

Sistemas de control de versiones locales

Los sistemas de control de versiones locales centran su desarrollo en una simple base de datos en la que se lleva registro de todos los cambios realizados sobre los archivos.

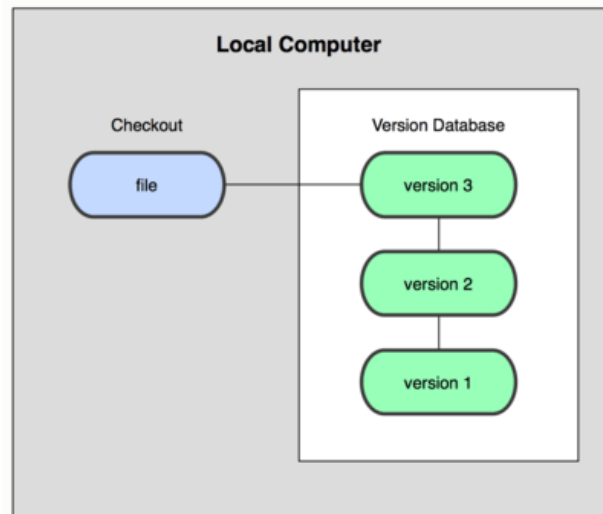


Figura 1. control de versiones local. [Imagen] Recuperada de: <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones>

Sistemas de control de versiones centralizados

Tienen un único servidor que contiene todos los archivos versionados y varios clientes que descargan los archivos desde ese lugar central. Este ha sido el estándar para el control de versiones por muchos años.

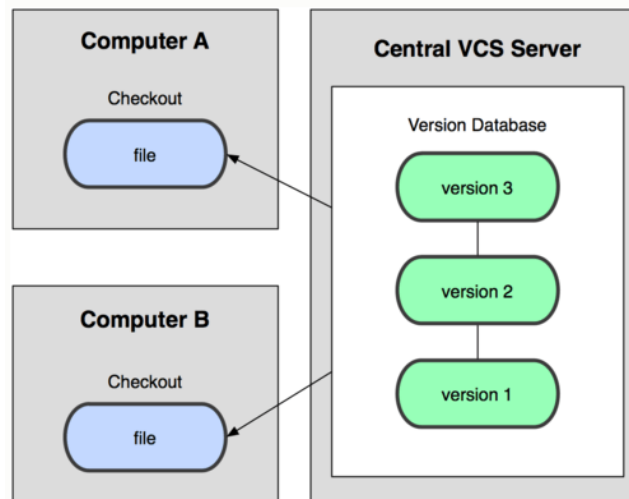


Figura 2. control de versiones centralizado. [Imagen] Recuperado de: <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones>

Sistemas de control de versiones distribuidos

A través de este tipo de control de versiones, se permite que si un servidor deja de funcionar y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios disponibles en los clientes puede ser copiado al servidor con el fin de restaurarlo.

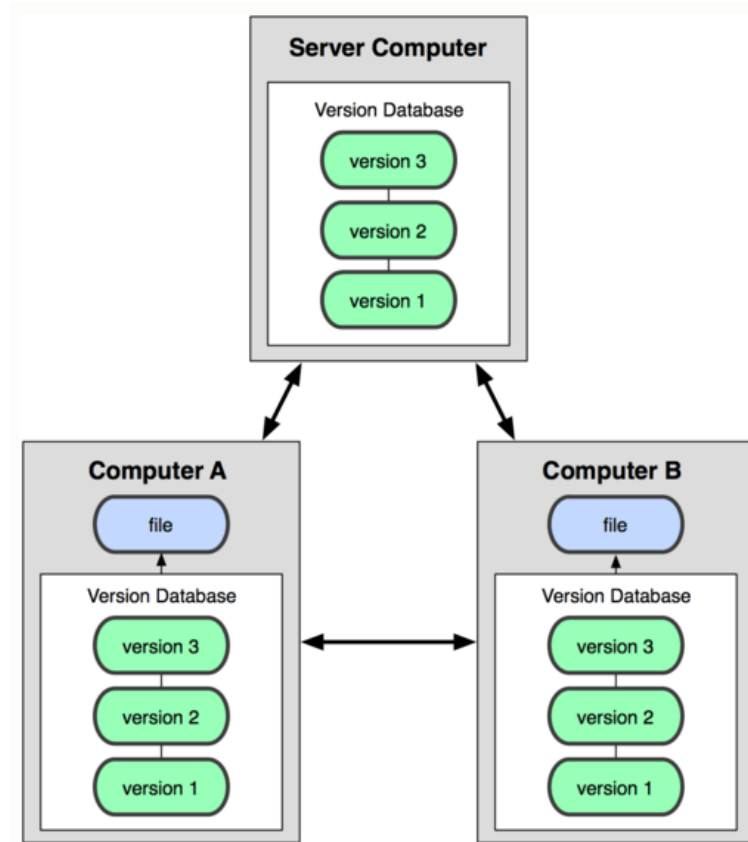


Figura 3. control de versiones distribuido. [Imagen] Recuperado de: <https://git-scm.com/book/es/v2/Inicio--Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones>



¿Que es Git?

Git, es un software de control de versiones diseñado por Linus Torvalds. Git maneja sus datos como un conjunto de copias de un pequeño sistema de archivos. Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en Git, este básicamente hace lo que podemos llamar como una captura de pantalla o foto del aspecto actual de los archivos y guarda dicha referencia. Si los archivos no se han modificado Git no almacena el archivo de nuevo, sino un enlace al archivo anterior idéntico que ya tiene almacenado. Git maneja sus datos como una secuencia de copias instantáneas.


¿Qué es GitHub?

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. GitHub permite alojar repositorios de código y brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto.

Además de eso, los usuarios pueden contribuir a mejorar el software de los demás. Para ello, GitHub provee diversas funcionalidades para hacer un fork y solicitar pulls.

Realizar un fork es simplemente clonar un repositorio de otro usuario, procedimiento que realizan los desarrolladores para eliminar o modificar. Una vez realizadas las modificaciones se envía lo que se denomina un pull (petición al propietario del repositorio original para que incorpore cambios propuestos) al dueño del proyecto. Éste podrá analizar los cambios realizados, y si considera interesante la contribución, adjuntarlo con el repositorio original.

En la actualidad, GitHub es mucho más que un servicio de alojamiento de código. Además de éste, se ofrecen varias herramientas útiles para el trabajo en equipo. Entre ellas, caben destacar:

- ✓ Una wiki para el mantenimiento de las distintas versiones de las páginas.
 - ✓ Un sistema de seguimiento de problemas que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.
 - ✓ Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.
 - ✓ Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.
- 

Diferencias entre Git y GitHub

Las diferencias básicamente radican en que Git es un sistema específico diseñado para controlar versiones en un entorno Linux; fue desarrollado con el objetivo primordial de gestionar código fuente.

En cambio, GitHub es una compañía que aloja repositorios Git y que proporciona un programa específico para usar Git. Entre las modalidades de uso, destaca el programa 'GitHub Desktop'. Actualmente GitHub es la plataforma más popular para alojar en abierto el código de proyectos digitales.

Pese a que GitHub está diseñado originalmente para publicar código fuente, algunos proyectos, lo utilizan para controlar las versiones y para gestionar el flujo de trabajo de sus publicaciones, libros de texto, etc.

En el siguiente esquema se plantean algunos conceptos mencionados.

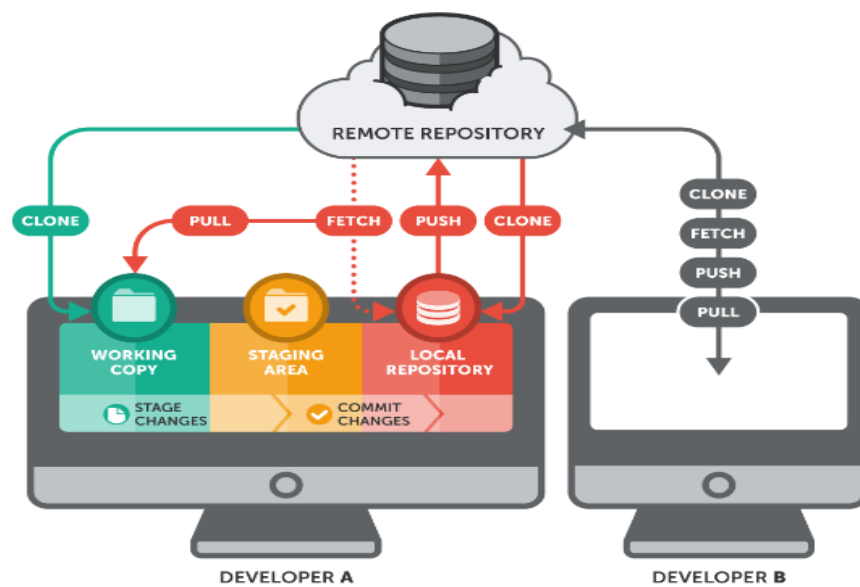


Figura 4.

GitHub.

[Imagen] Recuperado de: <https://gist.github.com/dirkdunn/2ff2784e8c780f0f7c2cc2a909fc299c>



Referencia Bibliográficas

Chacon, S. & Straub, B. (2014). *Pro Git* (2nd ed.). Mountain View: Apress. Recuperado de:
<https://git-scm.com/book/es/v2>

