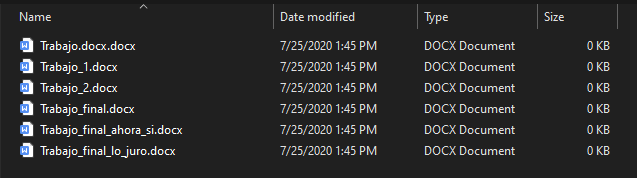
# INTRODUCCIÓN A SISTEMAS DE VERSIONES Y GIT

Te ha pasado que estás realizando un trabajo o documento, en un momento quieres realizar un cambio pero no quieres dañar el original, por si acaso, guardas una copia verdad? Al cabo de un tiempo tienes algo como:

En algún momento debes recuperar un cambio que quitaste, querías ver cómo estaba y qué cambió, en qué archivo estaba? ¿Debes revisarlos todos? ¿No guardaste la copia que tenía la información que buscas? Llevas tiempo sin trabajar en el documento y no sabes cuál es la versión final real o la última modificación.

Qué tal si el trabajo era en equipo? Eran cuatro compañeros, todos modificaron su parte y lo mandaron por correo a sus compañeros, lo reciben, modifican y lo vuelven a enviar, les suena familiar? Quién tiene la última versión? Quién une el trabajo? Quien borró mi trabajo? Cómo sabes si Juan y Pedro trabajaron uno sobre la versión del otro o ambos desde una versión anterior? Cómo hacen para que no se les pierda nada? Aún existe el mismo problema de múltiples archivos y no saben dónde están los cambios que necesitan.

Después de eso aparecen soluciones como Dropbox y Google Docs, una gran ayuda, ahora todos ven el mismo documento, en tiempo real y todos lo modifican a la vez. ¿Un gran alivio no?

Pero, oye, ¿quién quitó o cambió tu parte? Lo están moviendo todos a la vez, es confuso, tiene limitaciones. Necesitas que tu equipo pueda trabajar colaborativamente, que todos sepan qué ha cambiado, cuándo y quién lo hizo. Necesitan dar un paso más, hay una herramienta muy útil para superar todas esas complicaciones y tener un control confiable sobre su trabajo, se llaman sistemas de control de versiones, este nos registra los cambios realizados en uno o varios archivos.

## ¿Software de control de versiones? ¿Para qué quiero controlar las versiones de mi código?

Trabajar con un software de control de versiones, nos permite poder trabajar más tranquilos sabiendo que todos los cambios y modificaciones que vayamos haciendo en nuestro código están siendo “vigilados” por nuestro controlador de versiones, permitiéndonos revertir archivos a un estado anterior, revertir el proyecto entero a un estado anterior, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que puede estar causando un problema, quién introdujo un error y cuándo, y mucho más.

En la actualidad cualquier proceso de desarrollo debe incluir un control de versiones, para poder facilitar el trabajo en grupo y llevar un estricto seguimiento en los cambios de cada uno de los archivos de código de nuestras aplicaciones. Esta necesidad es tan vital que hace de Git una herramienta indicada incluso para desarrollos donde participa una única persona.

## ¿Que es git?

GIT es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

~ Wikipedia

~ Globant Acamica

Puedes ver un poco más en

* [Introducción a Git](https://www.youtube.com/watch?v=zH3I1DZNovk) (Vídeo)
* [Sobre el Control de Versiones](https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones)
* [Una breve historia de Git](https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Una-breve-historia-de-Git)

## ¿Línea de comandos o interfaz gráfica?

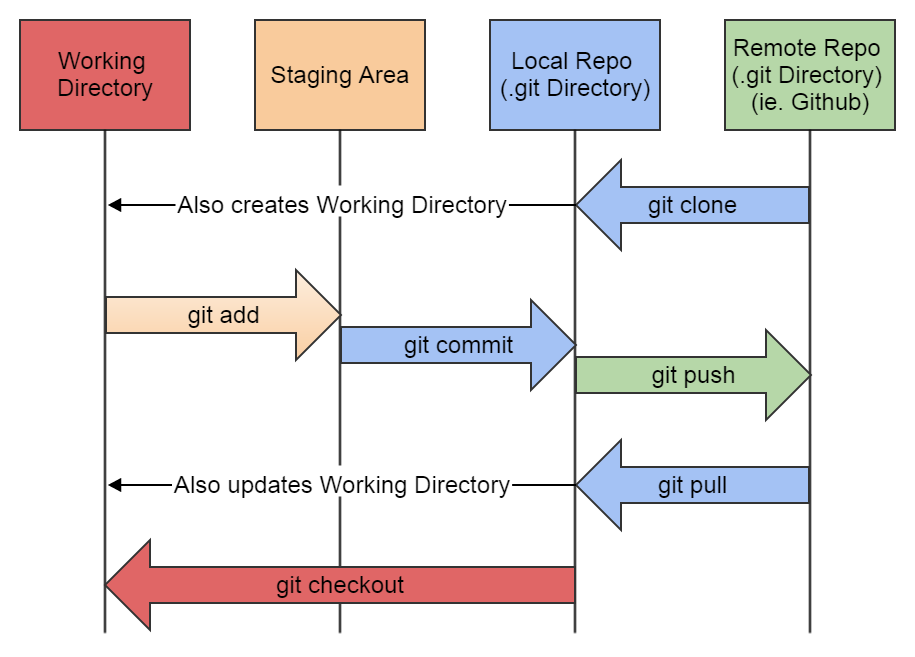
Aunque es posible utilizar GIT desde diversas interfaces gráficas integradas en las herramientas de desarrollo, es importante conocer los comandos básicos que nos permiten interactuar con nuestros archivos.

* La mayor parte de la documentación que se encuentra en internet, explica cómo usar la línea de comandos, por eso es importante conocerla además que es independiente de su herramienta de desarrollo o de su sistema operativo.

## [Comandos básicos:](http://rogerdudler.github.io/git-guide/index.es.html)

Explico un poco de los comandos más básicos, no te preocupes si no los entiendes, por el momento sólo queremos que tengas idea de qué hacen.

* **git init** (Iniciar) para crear un nuevo repositorio local de git.
* **git clone** (Clonar, copiar) crea una copia local del repositorio
* **git remote add** (Agregar remoto) conecta tu repositorio local a un repositorio remoto
* **git add** (Agregar) añade uno o varios archivos al Staging
* **git commit** (Confirmar) añade archivos al repositorio local desde el Staging
* **git push** (Empujar) añade archivos al repositorio remoto desde el repositorio local
* **git pull** (Halar) trae archivos desde el repositorio remoto al local, es igual a hacer fetch + merge (Comandos que veremos más adelante)
* **git checkout** (Sacar o salir) Revertir cambios y cambiar ramas



Si deseas ampliar tu información puedes visitar los siguientes recursos:

* [git - la guía sencilla](http://rogerdudler.github.io/git-guide/index.es.html)
* [Tutorial de Git. Manual básico con ejemplos](https://www.diegocmartin.com/tutorial-git/)
* [Documentación de Git](https://git-scm.com/book/es/v2)
* [Curso de Git Gratis](https://codigofacilito.com/cursos/git) (Vídeo-curso)

## ¿Qué es GitHub?

GitHub es un sitio web y un servicio en la nube que ayuda a los desarrolladores a almacenar y administrar su código, al igual que llevar un registro y control de cualquier cambio sobre este código. <https://kinsta.com/es/base-de-conocimiento/que-es-github/>