educación IT
CODE YOUR FUTURE

Git:
Desarrollo colaborativo

Módulo 1

educación IT
CODE YOUR FUTURE

Introducción a Git

¿Qué es y para qué sirve Git?

Git es un software de control de versiones diseñado por Linus Torvalds. Fue pensado para que el mantenimiento de versiones de aplicaciones - cuando tienen un gran número de archivos de código fuente - sea eficiente y confiable.

Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

Linus Torvalds

Pero, ¿qué es todo esto de control de versiones?

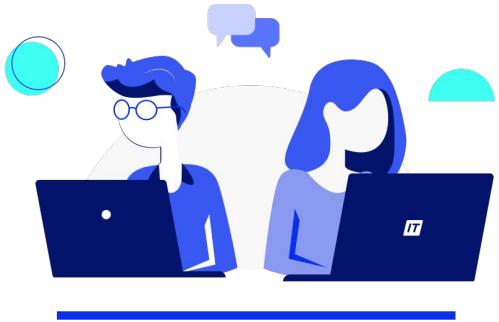
Imaginemos que comenzamos un nuevo proyecto que consiste en solo un archivo fuente. Lo más probable es que se puedan deshacer los cambios que se hicieron durante esa jornada.

Al día siguiente, se continúa con el desarrollo y se mejora el archivo pero, en el intento, el programa comienza a presentar errores. Necesitamos volver a una versión anterior del proyecto, donde no existía el error. Sin embargo, al no contar con un Control de Versiones, no contamos con respaldo de las versiones previas.

Git, entonces, ayuda con este aspecto del desarrollo. **Ofrece herramientas para poder gestionar cada una de las etapas y versiones** por las que va transitando un proyecto de desarrollo. Si bien el control de versiones es la principal característica de Git, actualmente, **es un gran aliado en el Desarrollo Colaborativo**.

¿Qué es el desarrollo colaborativo de software?

Es un modelo de desarrollo basado en la **disponibilidad pública del código y la comunicación vía Internet**. Este modelo se hizo popular a raíz de su uso para el desarrollo de Linux en 1991.

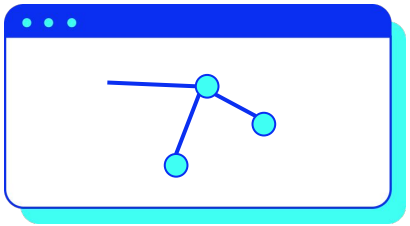


Tomando como contexto a Git, podríamos decir que **el desarrollo colaborativo proporciona herramientas para que un gran número de individuos puedan hacer desarrollos en conjunto de una manera más fácil, menos propensa a errores y rápida de implementar.**

De esta manera, siempre tenemos la opción de, por medio de algún cliente, publicar nuestro código junto con todas las etapas y versiones que nos llevó el proyecto para que otras personas puedan sumar y aportar nuevas ideas a nuestro repositorio.

¿Qué es un Grafo?

En matemáticas y ciencias de la computación, un grafo (del griego *grafos:* dibujo, imagen) es un **conjunto de objetos llamados *vértices* o *nodos* unidos por enlaces llamados *aristas* o *arcos*** que permiten **representar relaciones binarias** entre elementos de un conjunto. Son objeto de estudio de la Teoría de Grafos.



Un grafo **se representa gráficamente como un conjunto de puntos (vértices o nodos) unidos por líneas (aristas)**.

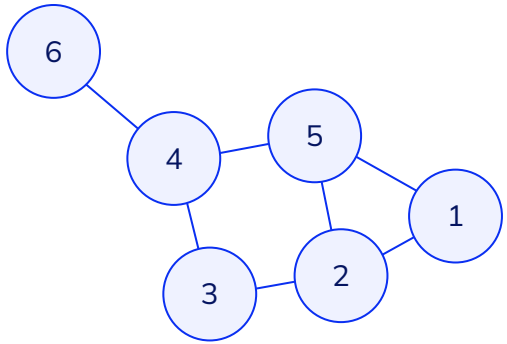
Desde un punto de vista práctico:

Los grafos **permiten estudiar las interrelaciones entre unidades que interactúan unas con otras.**

Por ejemplo: una red de computadoras puede representarse y estudiarse mediante un grafo, en el que los vértices representan terminales y las aristas representan conexiones (las cuales, a su vez, pueden ser cables o conexiones inalámbricas).

Prácticamente, cualquier problema puede representarse mediante un grafo, y su estudio trasciende a las diversas áreas de las ciencias exactas y las ciencias sociales.

Ejemplo de grafo:



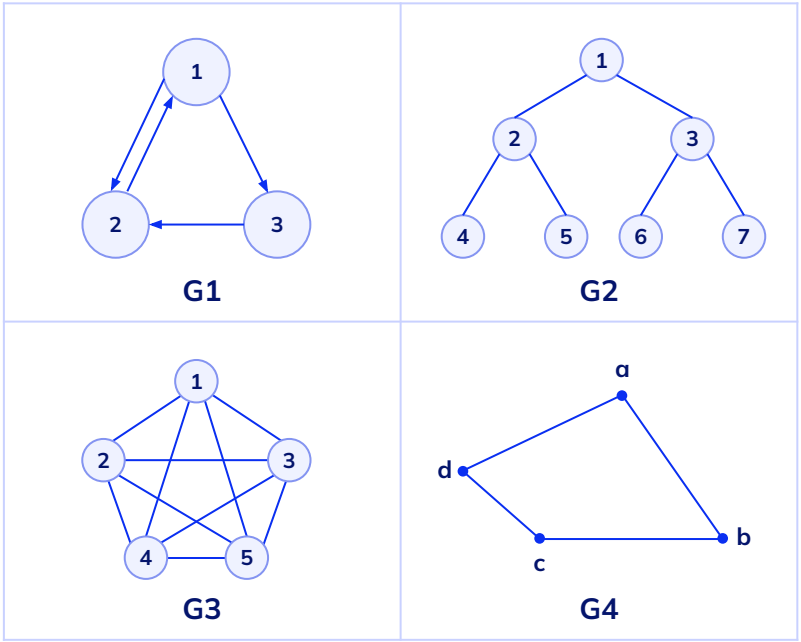
Un grafo G es un par ordenado G(V,N) donde:

- **V** es un conjunto de vértices o nodos, y...
- **N** es un conjunto de aristas o arcos, que relacionan estos nodos.

Teoría de grafos

Formalmente, un grafo **G=(V,E)** es una pareja ordenada en la que **V** es un conjunto no vacío de vértices y **E** es un conjunto de aristas. Donde **E** consta de pares no ordenados de vértices, tales como **{x,y}** in **E** entonces se dice que **x** e **y** son adyacentes; **y** en el grafo se representa mediante una línea no orientada que une dichos vértices.

Si el grafo es dirigido se le llama dígrafo, se denota **D**, y entonces el par **(x,y)** es un par ordenado, esto se representa con una flecha que va de **x** a **y** y se dice que **(x,y) in E**.



Un **grafo** está compuesto por:

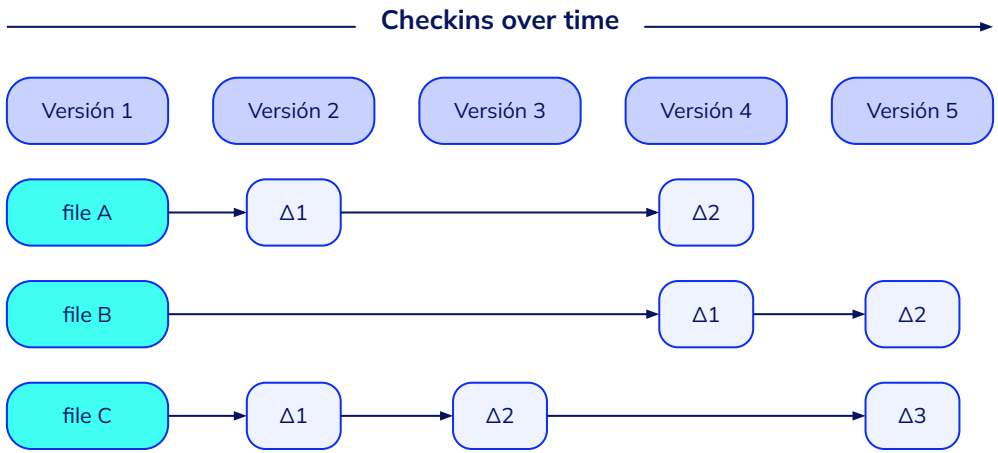
- **Aristas:** son las líneas con las que se unen los vértices de un grafo.
- **Aristas adyacentes:** 2 aristas son adyacentes si convergen en el mismo vértice.
- **Aristas paralelas:** son dos aristas conjuntas si el vértice inicial y final son el mismo.
- **Arista cíclicas:** es la arista que parte de un vértice para entrar en sí mismo.
- **Cruce:** son 2 aristas que cruzan en un mismo punto.
- **Vértices:** son los elementos que forman un grafo. Cada uno lleva asociada una valencia característica según la situación, que se corresponde con la cantidad de aristas que confluyen en dicho vértice.
- **Camino:** se denomina *camino de un grafo* a un conjunto de vértices interconectados por aristas. Dos vértices están conectados si hay un camino entre ellos.

Relación de grafos con Git

La principal diferencia entre Git y cualquier otro SCV (Ej.: CVS, Subversion, Perforce, Bazaar, y otros) **es la forma en que procesa sus datos**.

Conceptualmente, la mayoría de los otros sistemas almacenan información como una lista de cambios basados en archivos. Git no piensa ni almacena sus datos de esta manera.

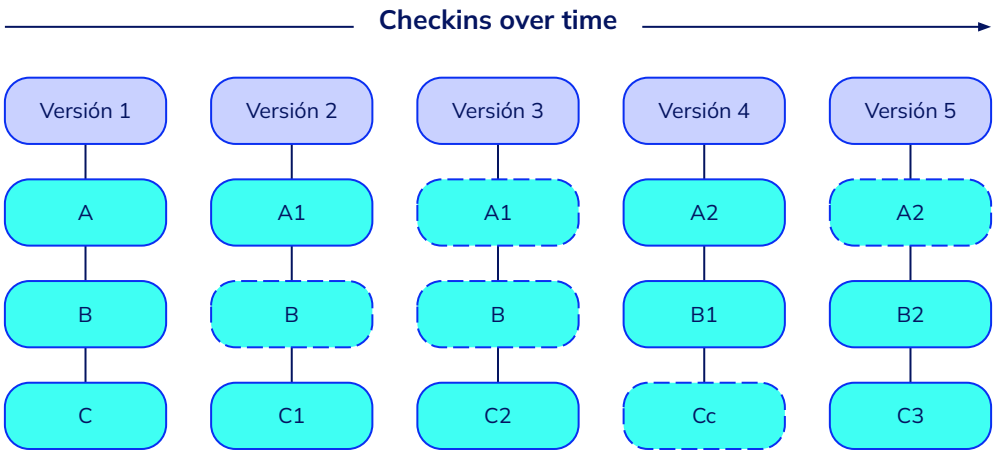
Teoría de grafos aplicada a Git



En cambio, Git piensa en sus datos más como un **conjunto de *snapshots* (instantáneas) de un mini sistema de archivos**. Cada vez que se confirma o se guarda el estado de un proyecto en Git, básicamente se toma una fotografía de cómo se ven todos los archivos en ese momento y se almacena una referencia a esa instantánea.

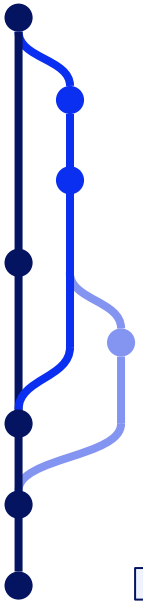
Para ser eficiente, **si los archivos no han cambiado, Git no almacena el archivo nuevamente, solo un enlace** al archivo idéntico anterior que ya ha almacenado.

Teoría de grafos aplicada a Git



Esta es una distinción importante entre Git y casi todos los demás SCV. Hace que Git reconsidere casi todos los aspectos del control de versiones que la mayoría de los otros sistemas copian de la generación anterior y que además se parezca más a un mini sistema de archivos con algunas herramientas increíblemente poderosas construidas sobre él, en lugar de simplemente un SCV.

Relación de grafos con Git



[master] 6c6faa5 My first commit - John Doe

[develop] 3e89ec8 Develop a feature - part 1 - John Doe

[develop] e188fa9 Develop a feature - part 2 - John Doe

[master] 665003d Fast bugfix - John Fixer

[myfeature] 6af618c New cool feature- John Feature

[master] 8f1e0e7 Merge branch `develop´ into ´master´ - John Doe

[master] 6a3dacc Merge branch `myfeature into ´master´ - John Doe

[master] abcdef0 Release of version 0.1 - John Doe

¡Sigamos trabajando!