relevant.

Git & GitHub

git

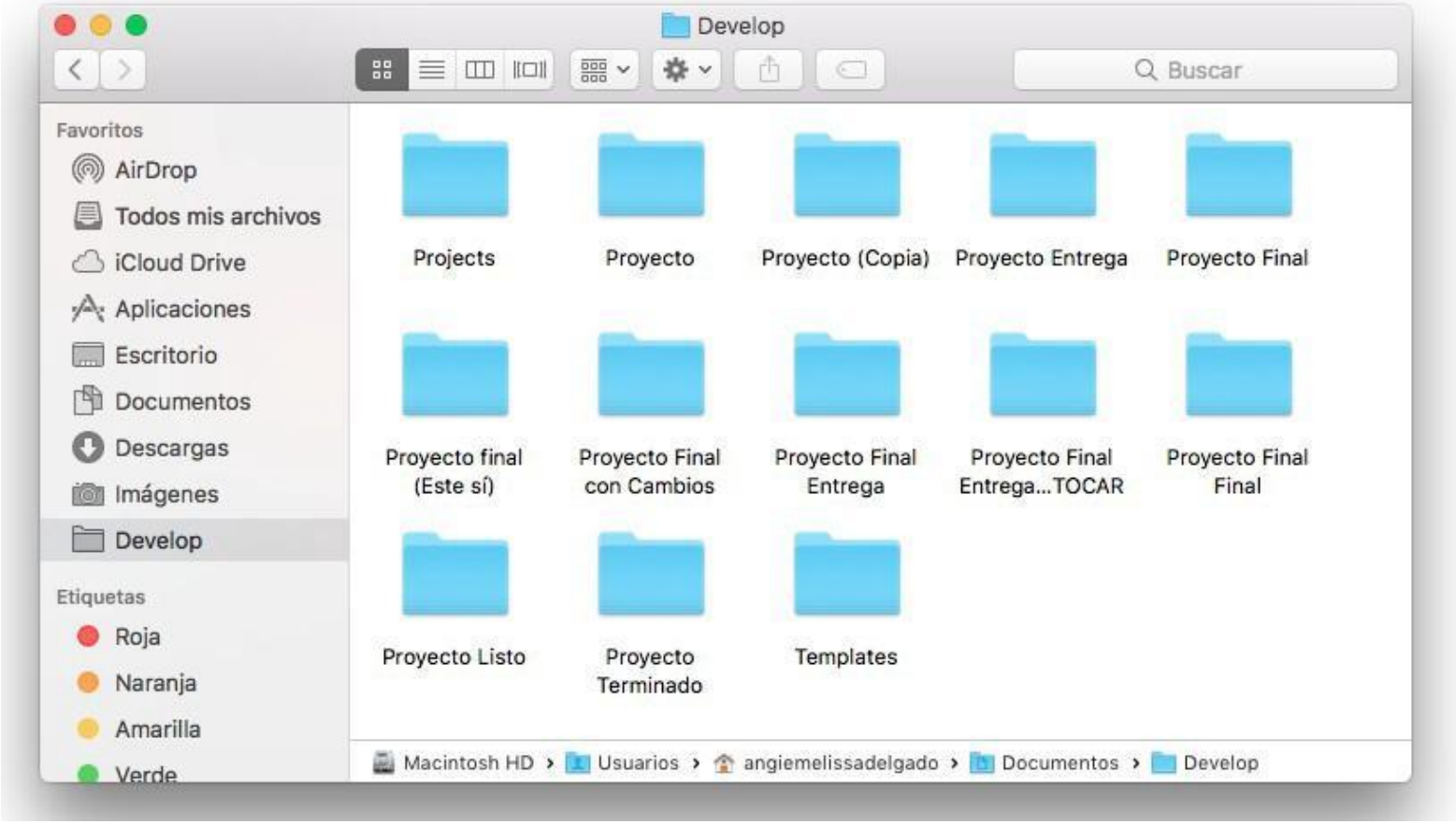
Índice Sesión 1

releevant.

1. **Sistema de control de versiones (SCV o VCS)**
   - ¿Porqué?
   - ¿Qué es?
   - Tipos: exclusivos / colaborativos, centralizados / distribuidos
   - Conceptos básicos / Definiciones
2. **Git**
   - Definición
   - Repositorio

Sistema de control de versiones: ¿Por qué?

releevant.



Sistema de control de versiones: ¿Por qué?

releevant.

- SCV: Sistema de Control de Versiones
- VCS: Version Control System

¿Por qué?

Es muy probable que en algún momento de nuestras vidas nuestra carpeta de documentos luciera como la de la imagen y nos haya tocado recurrir a tener muchas copias de nuestros proyectos, copias que requerían de toda nuestra creatividad para nombrarlas con etiquetas super útiles para poder reconocer cuál era nuestro ansiado “proyecto final”.

Antes de la masificación de los sistemas de control de versiones, se solían guardar los hitos (una especie de versión) del proyecto en archivos comprimidos y medios de almacenamiento como disquetes y CD’s.

¿Se imaginan lo insostenible que resultaba esta práctica en proyectos de gran envergadura?. Los sistemas de control de versiones nacen de la necesidad de solventar y facilitar este tedioso proceso.

Sistema de Control de Versiones: ¿Qué es?

- Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.
- Una versión, revisión o edición de un producto, es el estado en el que se encuentra dicho producto en un momento dado de su desarrollo o modificación.
- Es muy aconsejable disponer de herramientas que faciliten esta gestión dando lugar a los llamados sistemas de control de versiones.
- Ejemplos de este tipo de herramientas: CVS, Subversion, SourceSafe, ClearCase, Darcs, Bazaar , Plastic SCM, Git, Mercurial, Perforce.

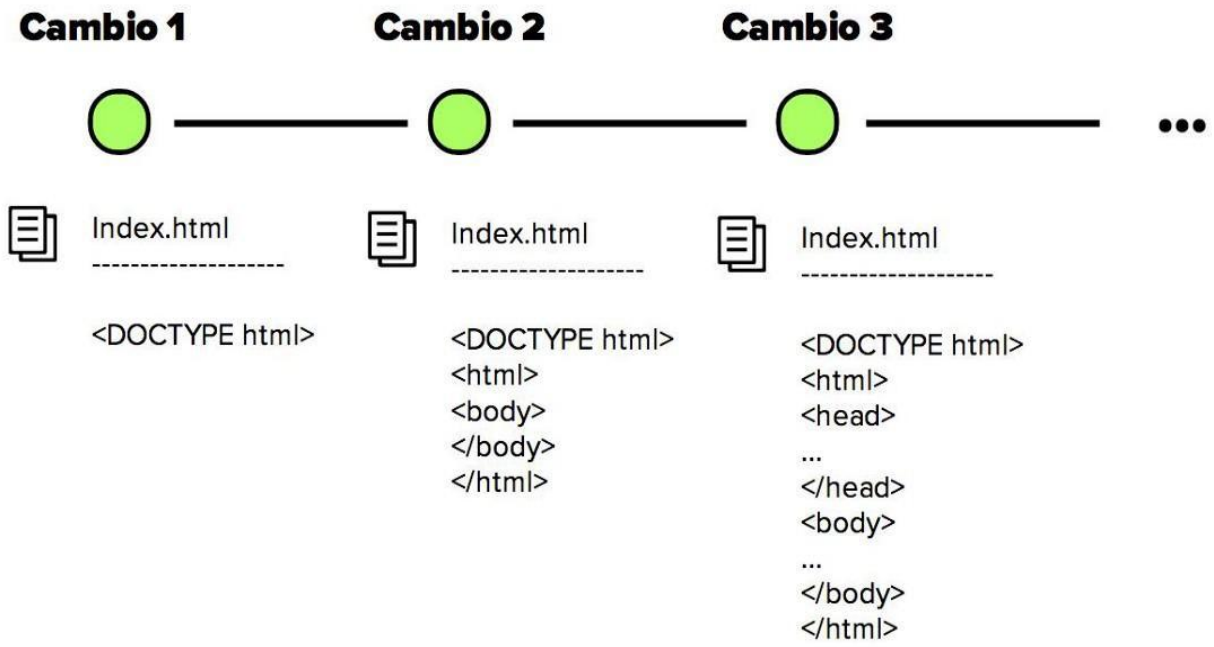
releevant.

Sistema de Control de Versiones: ¿Qué es?

Un sistema de control de versiones debe proporcionar:

- Mecanismo de almacenamiento de los elementos que deba gestionar (ej. archivos de texto, imágenes, documentación...).
- Posibilidad de realizar cambios sobre los elementos almacenados (ej. modificaciones parciales, añadir, borrar, renombrar o mover elementos).
- Registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente pudiendo volver o extraer un estado anterior del producto).

Sistema de Control de Versiones: ¿Qué es?



Ventajas:

- Regresar a versiones anteriores de tus archivos
- Comparar cambios a lo largo del tiempo
- Ver quién modificó por última vez algo
- Recuperar archivos fácilmente si se pierden

SVC: Tipos

Todos los sistemas de control de versiones se basan en disponer de un repositorio, que es el conjunto de información gestionada por el sistema. Este repositorio contiene el historial de versiones de todos los elementos gestionados.

Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso. Es posible duplicar la última versión o cualquier versión almacenada en el historial. Este proceso se suele conocer como checkout o desproteger. Para modificar la copia local existen dos formas de colaborar:

SVC: Tipos

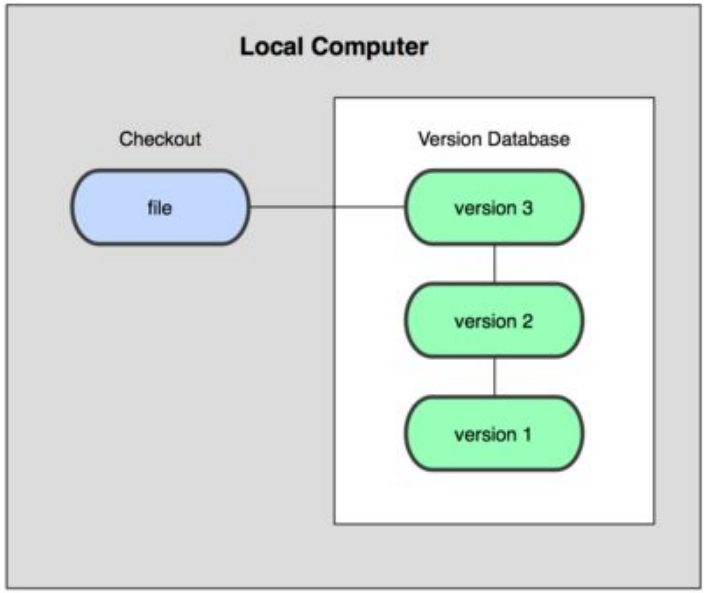
Según la forma de colaborar:

- **Exclusivo**: Para poder realizar un cambio es necesario comunicar al repositorio el elemento que se desea modificar y el sistema se encargará de impedir que otro usuario pueda modificar dicho elemento. Una vez hecha la modificación, esta se comparte con el resto de colaboradores. Si se ha terminado de modificar un elemento entonces se libera ese elemento para que otros lo puedan modificar.
- **Colaborativo:** Cada usuario modifica la copia local y cuando el usuario decide compartir los cambios el sistema automáticamente intenta combinar las diversas modificaciones. El principal problema es la posible aparición de conflictos que deban ser solucionados manualmente o las posibles inconsistencias que surjan al modificar el mismo fichero por varias personas no coordinadas.

SVC: Tipos

Según la arquitectura de almacenamiento:

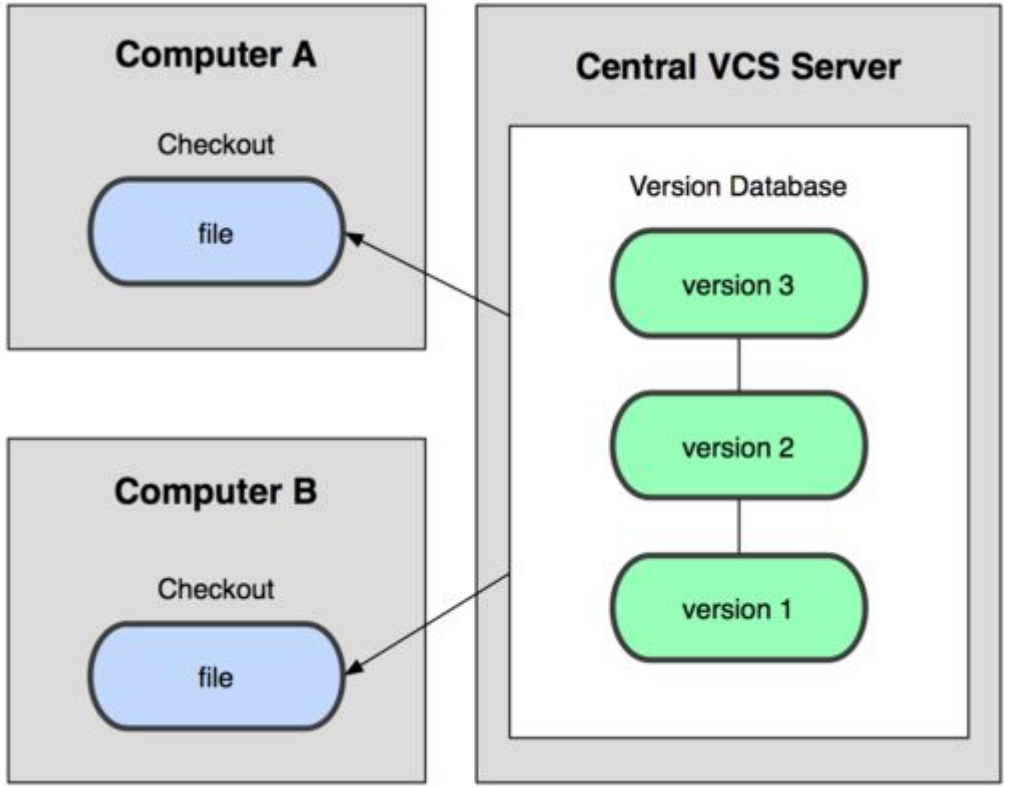
- **Locales**: Los cambios son guardados localmente y no se comparten con nadie. Esta arquitectura es la antecesora de las dos siguientes. En este punto el control de versiones se llevaba a cabo en el computador de cada uno de los desarrolladores y no existía una manera eficiente de compartir el có



releevant.

SVC: Tipos

- **Centralizados**: Existe un repositorio centralizado de todo el código, del cual es responsable un único usuario (o conjunto de ellos). Se facilitan las tareas administrativas a cambio de reducir flexibilidad, pues todas las decisiones fuertes (como crear una nueva rama) necesitan la aprobación del responsable.



releevant.

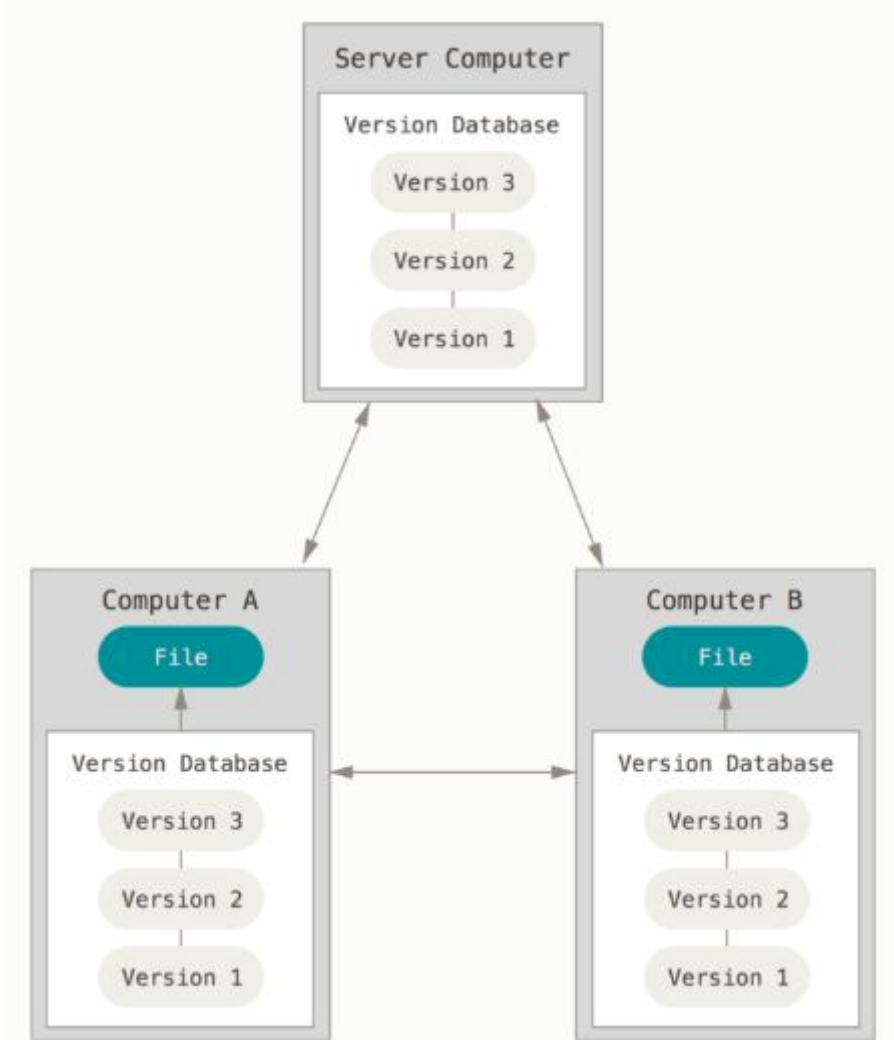
SVC: Tipos

- Todas las personas saben hasta cierto punto en qué están trabajando los otros colaboradores del proyecto.
- Los administradores tienen control detallado sobre qué puede hacer cada usuario.
- Si ese servidor se cae durante una hora, entonces durante esa hora nadie podrá colaborar o guardar cambios en archivos en los que hayan estado trabajando.
- Si el disco duro en el que se encuentra la base de datos central se corrompe, y no se han realizado copias de seguridad adecuadamente, se perderá toda la información del proyecto,
- Cuando tienes toda la historia del proyecto en un mismo lugar, te arriesgas a perderlo todo.

SVC: Tipos

releevant.

- **Distribuidos**: Los clientes no solo descargan la última copia instantánea de los archivos, sino que se replica completamente el repositorio. De esta manera, 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. Cada clon es realmente una copia completa de todos los datos. Los distintos repositorios pueden intercambiar y mezclar revisiones entre ellos. Es frecuente el uso de un repositorio, que está normalmente disponible, que sirve de punto de sincronización de los distintos repositorios locales.



releevant.

SVC: Tipos

- Necesita menos veces estar conectado a la red para hacer operaciones. Esto produce una mayor autonomía y una mayor rapidez.
- Aunque se caiga el repositorio remoto la gente puede seguir trabajando.
- Al hacer de los distintos repositorio una réplica local de la información de los repositorios remotos a los que se conectan, la información está muy replicada y por tanto el sistema tiene menos problemas en recuperarse si por ejemplo se quema la máquina que tiene el repositorio remoto.
- Permite mantener repositorios centrales más limpios en el sentido de que un usuario puede decidir que ciertos cambios realizados por él en el repositorio local, no son relevantes para el resto de usuarios y por tanto no permite que esa información sea accesible de forma pública, ya sea porque contiene versiones inestables, en proceso de codificación o con etiquetas personalizadas y de uso exclusivo del usuario.
- Al ser los sistemas distribuidos más recientes que los sistemas centralizados, y al tener más flexibilidad por tener un repositorio local y otro u otros remotos, estos sistemas han sido diseñados para hacer fácil el uso de ramas.

releevant.

SVC: Conceptos básicos

Procedimiento de uso habitual de un sistema de control de versiones

- Descarga de ficheros inicial (Checkout)
- Ciclo de trabajo habitual:
  - Modificación de los ficheros
  - Actualización de ficheros en local (Update)
  - Resolución de conflictos (si los hay)
  - Actualización de ficheros en repositorio (Commit).

SVC: Conceptos básicos

- **Repositorio**: es el lugar en el que se almacenan los datos actualizados e históricos de cambios, a menudo en un servidor. A veces se le denomina depósito o depot. Puede ser un sistema de archivos en un disco duro, un banco de datos, etc..
- **Revisión** (“versión”): es una versión determinada de la información que se gestiona.
- **Rotular** (“tag”): darle a alguna versión de cada uno de los ficheros del módulo en desarrollo en un momento preciso un nombre común ("etiqueta" o "rótulo") para asegurarse de reencontrar ese estado de desarrollo posteriormente bajo ese nombre. Los tags permiten identificar de forma fácil revisiones importantes en el proyecto.
- **Ramificar** (“branch”): un módulo puede ser **bifurcado** en un instante de tiempo de forma que, desde ese momento en adelante se tienen dos copias (ramas) que evolucionan de forma independiente siguiendo su propia línea de desarrollo.
- **Desplegar** (“Check-out”): un despliegue crea una copia de trabajo local desde el repositorio. Se puede especificar una revisión concreta, y predeterminadamente se suele obtener la última.
- **Publicar** (“commit”): sucede cuando una copia de los cambios hechos a una copia local es escrita o integrada sobre el repositorio.
- **Conflicto**: ocurre cuando el sistema no puede manejar adecuadamente cambios realizados por dos o más usuarios en un mismo archivo.

SVC: Conceptos básicos

releevant.

- **Resolver**: acto de la intervención del usuario para atender un conflicto entre diferentes cambios al mismo archivo.
- **Revisión** (“versión”): es una versión determinada de la información que se gestiona.
- **Cambio** (“change”,”diff”): representa una modificación específica a un archivo bajo control de versiones. La granularidad de la modificación considerada un cambio varía entre diferentes sistemas de control de versiones.
- **Integración** (“merge”): une dos conjuntos de cambios sobre un fichero o un conjunto de ficheros en una revisión unificada de dicho fichero o ficheros.
- **Desplegar** (“Check-out”): un despliegue crea una copia de trabajo local desde el repositorio. Se puede especificar una revisión concreta, y predeterminadamente se suele obtener la última.
- **Actualización** (“update”): integra los cambios que han sido hechos en el repositorio (por otras personas) en la copia de trabajo local.
- **Copia de trabajo** (“workspace”): es la copia local de los ficheros de un repositorio, en un momento del tiempo o revisión específicos. Todo el trabajo realizado sobre los ficheros en un repositorio se realiza inicialmente sobre una copia de trabajo, de ahí su nombre.

releevant.

Git

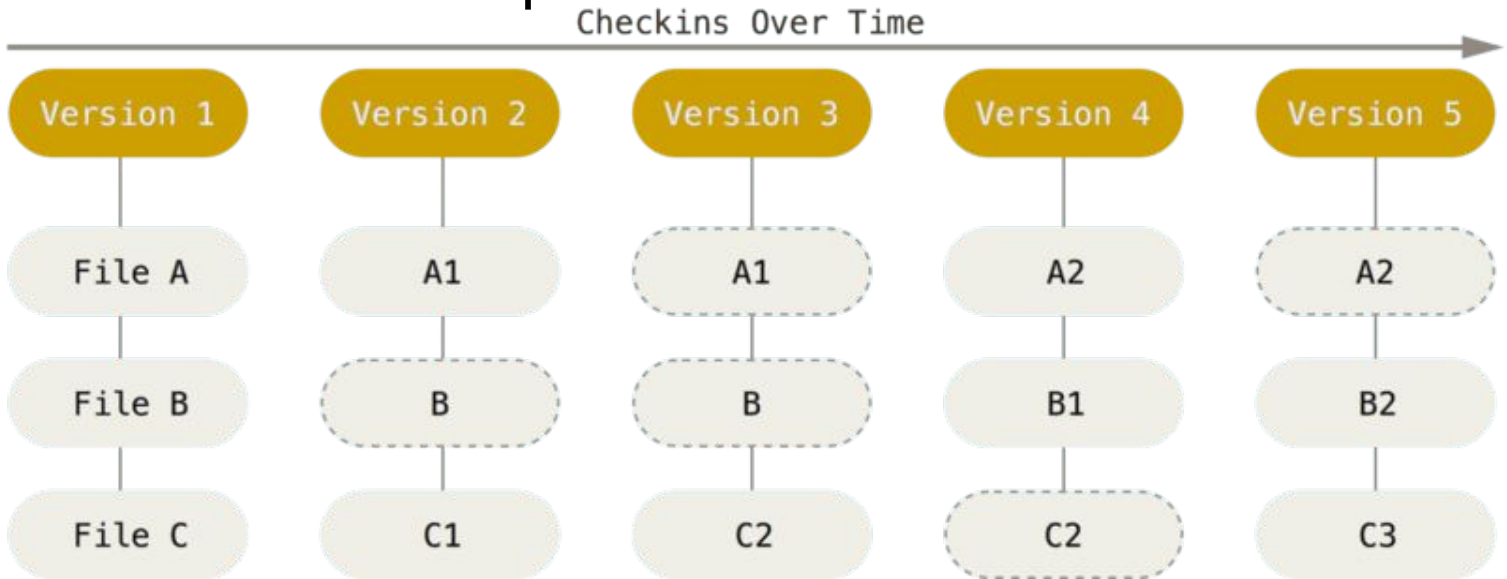
Git es un gestor de **repositorios** de versiones software

- Desarrollado por Linus Torvalds en 2005 en código libre
- Lo diseñó para soportar el desarrollo de Linux con un modelo descentralizado y abierto.

Git

releevant.

Git maneja sus datos como un conjunto de copias instantáneas de un sistema de archivos miniatura. Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en Git, él básicamente toma una foto del aspecto de todos tus archivos en ese momento, y guarda una referencia a esa copia instantánea. Para ser eficiente, 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



Git

- **Casi todas las operaciones son locales**
- **Git tiene integridad**
- **Git generalmente solo añade información**
- **Los Tres Estados**

Referencias

releevant.

- https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones
- https://medium.com/@jointdeveloper/sistemas-de-control-de-versiones-qué-son-y-por-qué-amarlos-24b6957e716e
- https://www.uco.es/aulasoftwarelibre/curso-de-git/cvs/
- https://es.wikipedia.org/wiki/Control_de_versiones#Formas_de_colaborar
- https://www.ecured.cu/Sistemas_de_control_de_versiones