CONTROL DE VERSIONES

GIT: software de control de versiones distribuido

CONTROL DE VERSIONES

¿Qué es el control de versiones?

En un proyecto donde trabajan diferentes personas sobre diferentes ficheros necesitamos saber:

- ¿Quién ha modificado los ficheros?
- ¿Qué modificaciones ha hecho?
- ¿Còmo volver a una versión anterior?
- ¿Cuál recueperamos?
- ¿Los cambios son adecuados? ¿Los aceptamos?

¿Qué es el control de versiones?

- El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.
- Es aconsejable disponer de herramientas que faciliten esta gestión dando lugar a los llamados sistemas de control de versiones o SVC.

Funciones

Un control de versiones nos tiene que permitir

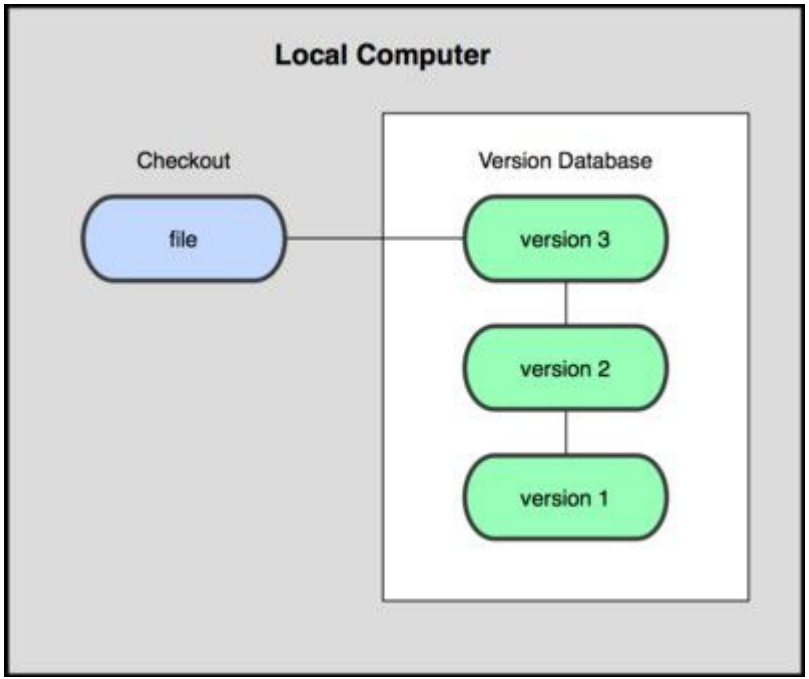
- Volver a la versión anterior de los ficheros y mantener un histórico de los cambios
- Garantizar la integridad de los datos
- Controlar usuarios y permisos
- Crear ramas (branch) de un proyecto en un momento determinado

TIPOS DE SISTEMAS DE CONTROL DE VERSIONES

Sistemas de control de versiones locales

- El método de control de versiones usado por mucha gente es copiar los archivos a otro directorio (quizás indicando la fecha y hora en que lo hicieron, si son avispados). Este enfoque es muy común porque es muy simple, pero también tremendamente propenso a errores.
- Es fácil olvidar en qué directorio te encuentras, y guardar accidentalmente en el archivo equivocado o sobrescribir archivos que no querías.

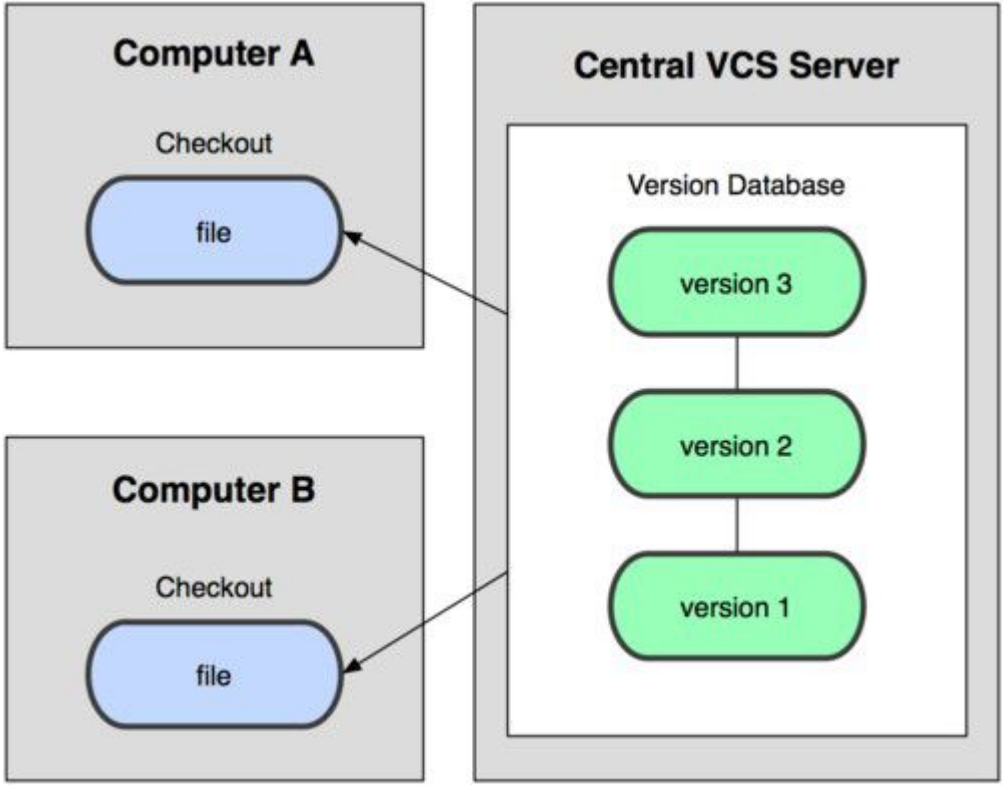
Sistemas de control de versiones locales



Sistemas de control de versiones centralizados

- El siguiente gran problema que se encuentra la gente es que necesitan colaborar con desarrolladores en otros sistemas.
- Para solventar este problema, se desarrollaron los sistemas de control de versiones centralizados (Centralized Version Control Systems o CVCSs en inglés). Estos sistemas, como CVS, Subversion, y
- Perforce, tienen un único servidor que contiene todos los archivos versionados, y varios clientes que descargan los archivos de ese lugar central.
- Durante muchos años, éste ha sido el estándar para el control de versiones

Sistemas de control de versiones centralizados



Sistemas de control de versiones centralizados

Ventajas

- Esta configuración ofrece muchas ventajas, especialmente frente a VCSs locales.
- Todo el mundo sabe hasta cierto punto en qué está trabajando el resto de gente en el proyecto. Los administradores tienen control detallado de qué puede hacer cada uno.

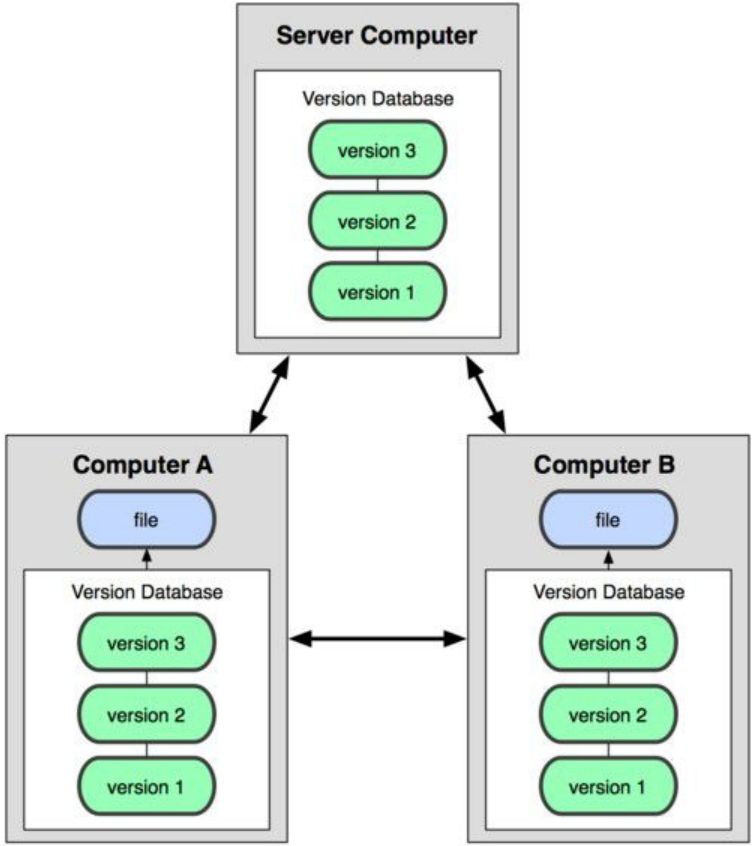
Inconvenientes

- El servidor centralizado: Si ese servidor se cae durante una hora, entonces durante esa hora nadie puede colaborar o guardar cambios versionados de aquello en que están trabajando.
- Si el disco duro en el que se encuentra la base de datos central se corrompe, y no se han llevado copias de seguridad adecuadamente, pierdes absolutamente todo —toda la historia del proyecto salvo aquellas instantáneas que la gente pueda tener en sus máquinas locales

Sistemas de control de versiones distribuidos

- Es aquí donde entran los sistemas de control de versiones distribuidos (Distributed Version Control Systems o DVCSs en inglés). En un DVCS (como Git, Mercurial, Bazaar o Darcs), los clientes no sólo descargan la última instantánea de los archivos: replican completamente el repositorio.
- Así, si un servidor muere, y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios de los clientes puede copiarse en el servidor para restaurarlo. Cada vez que se descarga una instantánea, en realidad se hace una copia de seguridad completa de todos los datos.

Sistemas de control de versiones distribuidos



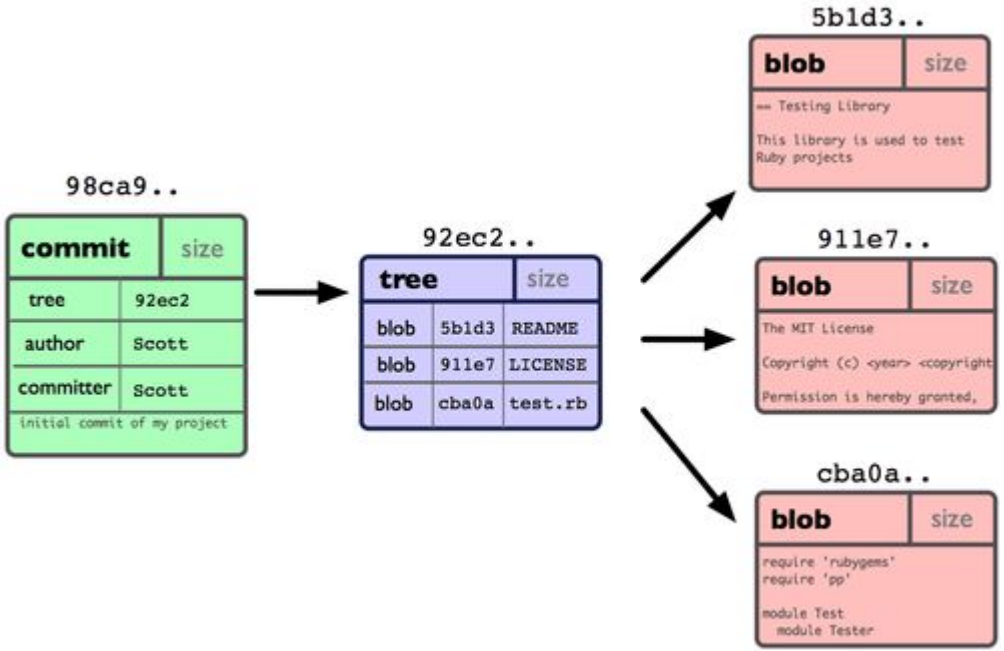
GIT

¿Qué es GIT?

- Es un software de control de versiones distribuido.
- Lo diseñó Linus Torvalds.
- El diseño se basó en BitKeeper y en Monotone.
- Historia: En 2005, la relación entre la comunidad que desarrollaba el núcleo de Linux y la compañía que desarrollaba BitKeeper se vino abajo, y la herramienta dejó de ser ofrecida gratuitamente. Esto impulsó a la comunidad de desarrollo de Linux (y en particular a Linus Torvalds, el creador de Linux) a desarrollar su propia herramienta basada en algunas de las lecciones que aprendieron durante el uso de BitKeeper.

GIT

- Totes les operacions generen una signatura SHA-1 que servirà per comprovar-ne la integritat



Objetivos de GIT

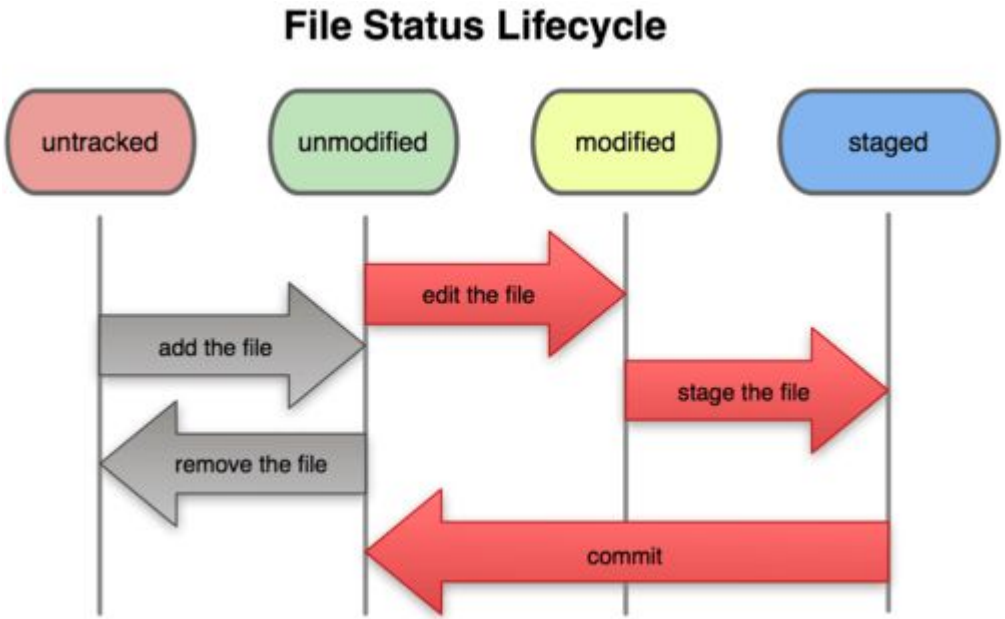
- Velocidad
- Diseño sencillo
- Fuerte apoyo al desarrollo no lineal (miles de ramas paralelas)
- Completamente distribuido
- Capaz de manejar grandes proyectos como el núcleo de Linux de manera eficiente (velocidad y tamaño de los datos)

Estados de los archivos

Git tiene tres estados principales en los que se pueden encontrar tus archivos:

- **confirmado (committed)**: los datos están almacenados de manera segura en tu base de datos local.
- **modificado (modified)**: se ha modificado el archivo pero todavía no lo has confirmado a tu base de datos.
- **preparado (staged)**: se ha marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

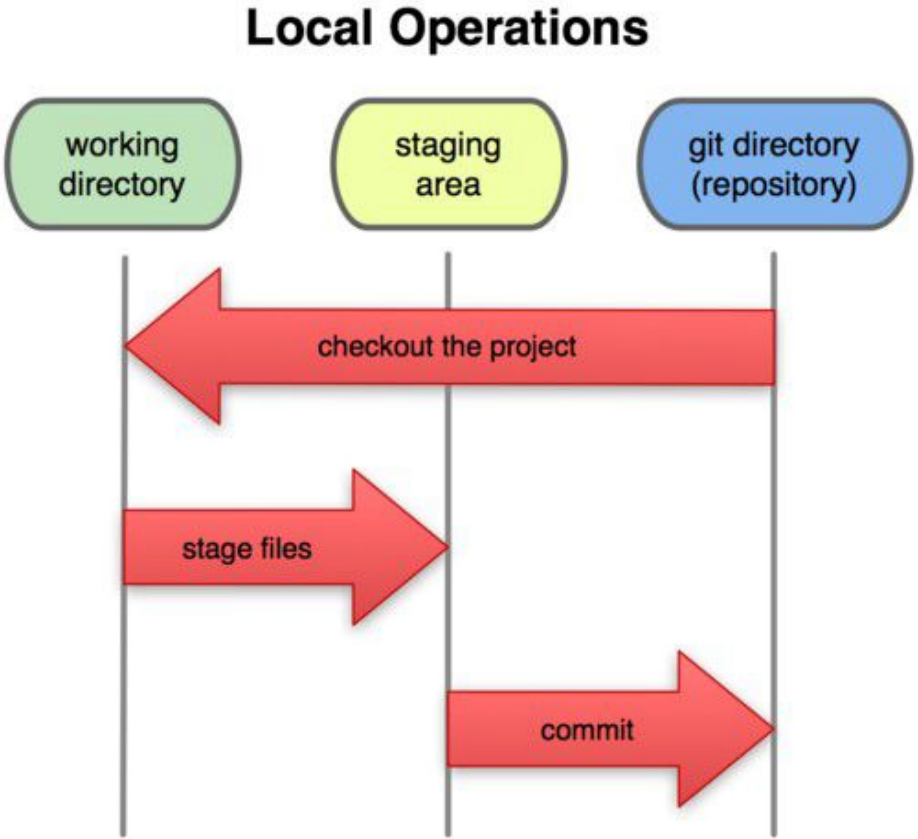
Estados de los archivos



Secciones principales de un proyecto

- **Directorio de Git (Git directory)**: es donde Git almacena los metadatos y la base de datos de objetos para tu proyecto. Es la parte más importante de Git, y es lo que se copia cuando clonas un repositorio desde otro ordenador.
- **Directorio de trabajo (working directory)**: El directorio de trabajo es una copia de una versión del proyecto. Estos archivos se sacan de la base de datos comprimida en el directorio de Git, y se colocan en disco para que los puedas usar o modificar.
- **Área de preparación (staging area)**: es un sencillo archivo, generalmente contenido en tu directorio de Git, que almacena información acerca de lo que va a ir en tu próxima confirmación.

Secciones principales de un proyecto



Flujo de trabajo en GIT

El flujo de trabajo básico en Git es algo así:

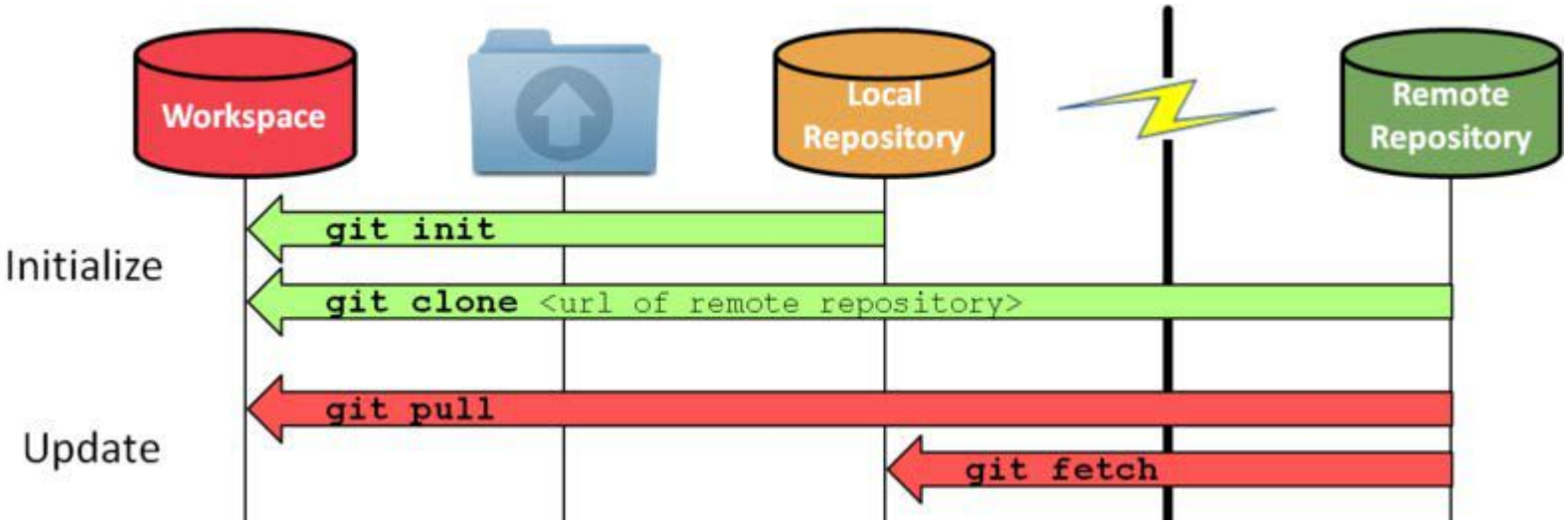
1. Modificas una serie de archivos en tu directorio de trabajo.
2. Preparas los archivos, añadiendo instantáneas de ellos a tu área de preparación.
3. Confirmas los cambios, lo que toma los archivos tal y como están en el área de preparación, y almacena esa instantánea de manera permanente en tu directorio de Git.

Flujo de trabajo en GIT

Encontraremos diferentes flujos de trabajo:

- **Enfoque centralizado**: solo existe un flujo de trabajo posible. Los desarrolladores se sincronizarán con el repositorio centralizado.
- **Enfoque distribuido**:
  - Flujo manager: Los desarrolladores no subirán código al repositorio público. Por tanto, nos va a interesar crear la figura del manager. Los desarrolladores enviarán el código al manager, y este subirá los cambios al repositorio público.
  - Flujo dictador: Utilizado para proyectos Open Source muy grandes. Permite tener jerarquías. Los desarrolladores podrán bajarse el código, pero no podrán subirlo sin pasar por el “rol dictador”.

Flujo de trabajo en GIT



Flujo de trabajo en GIT