Pro Git, el libro oficial de Git

1.1. Acerca del control de versiones

¿Qué es el control de versiones, y por qué debería importarte? 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. Los ejemplos de este libro utilizan el control de versiones para el código fuente, pero puedes emplearlo para casi cualquier tipo de archivo que encuentres en un ordenador.

Si eres diseñador gráfico o web, y quieres mantener cada versión de una imagen o diseño (algo que sin duda quieres), un sistema de control de versiones (*Version Control System* o VCS en inglés) es una elección muy sabia. Te permite 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. Usar un VCS también significa que si fastidias o pierdes archivos, puedes recuperarlos fácilmente. Además, obtienes todos estos beneficios a un coste muy bajo.

1.1.1. 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). Esta técnica es muy común porque es muy simple, pero también tremendamente propensa a cometer errores. Es fácil olvidar en qué directorio te encuentras, y guardar accidentalmente en el archivo equivocado o sobrescribir archivos que no querías.

Para hacer frente a este problema, los programadores desarrollaron hace tiempo VCSs locales que contenían una simple base de datos en la que se llevaba registro de todos los cambios realizados sobre los archivos (véase Figura 1-1).

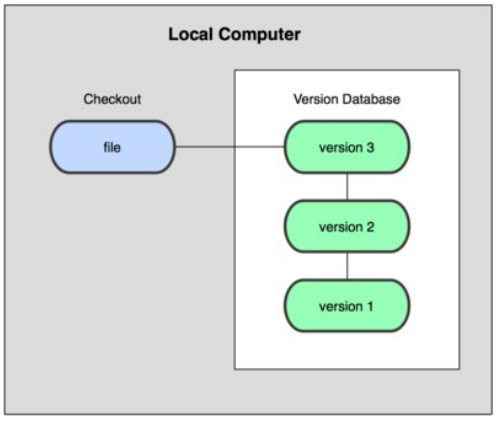


Figura 1.1 Diagrama de control de versiones local

Una de las herramientas de control de versiones más popular fue un sistema llamado `rcs`, que todavía podemos encontrar en muchos de los ordenadores actuales. Hasta el famoso sistema operativo Mac OS X incluye el comando `rcs` cuando instalas las herramientas de desarrollo. Esta herramienta funciona básicamente guardando conjuntos de parches (es decir, las diferencias entre archivos) de una versión a otra en un formato especial en disco. Así puede recrear después cómo era un archivo en cualquier momento sumando los distintos parches.

1.1.2. Sistemas de control de versiones centralizados

Indice de contenidos

Aspose.Cells for Java
Automate Excel sheets processing using powerful & intuitive API
Full-featured Java library
Free Trial
ASPOSE

Aspose.Cells for Java

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 (véase Figura 1-2).

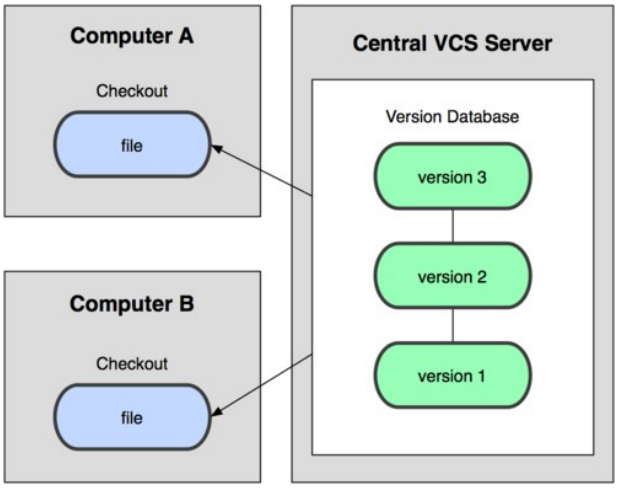


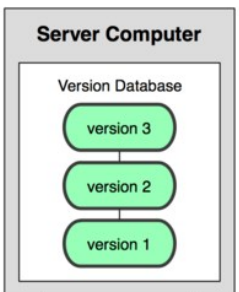
Figura 1.2 Diagrama de control de versiones centralizado

Esta configuración ofrece muchas ventajas, especialmente frente a VCSs locales. Por ejemplo, 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; y es mucho más fácil administrar un CVCS que tener que lidiar con bases de datos locales en cada cliente.

Sin embargo, esta configuración también tiene serias desventajas. La más obvia es el punto único de fallo que representa 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. Los VCSs locales sufren de este mismo problema — cuando tienes toda la historia del proyecto en un único lugar, te arriesgas a perderlo todo.

1.1.3. 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 (véase Figura 1-3).



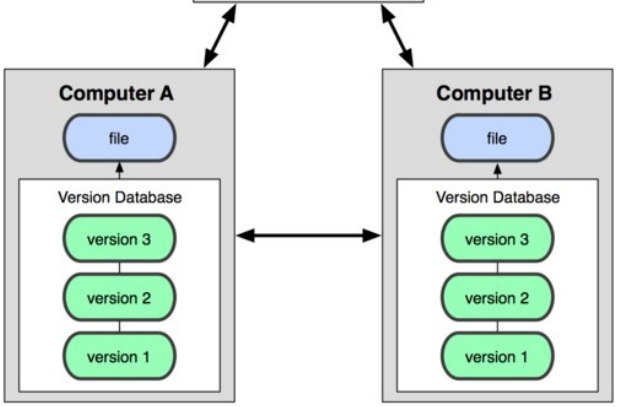


Figura 1.3 Diagrama de control de versiones distribuido

Es más, muchos de estos sistemas se las arreglan bastante bien teniendo varios repositorios con los que trabajar, por lo que puedes colaborar con distintos grupos de gente de maneras distintas simultáneamente dentro del mismo proyecto. Esto te permite establecer varios tipos de flujos de trabajo que no son posibles en sistemas centralizados, como pueden ser los modelos jerárquicos.

Anterior
Capítulo 1. Empezando

Siguiente
1.2. Una breve historia de Git