Control de Versiones. Características

Fecha entrega: 13/04/15
Autor: Pedro J. Ramos
Estela Muñoz Cordón

Control de versiones.

- Introducción
- ¿Qué es? Definición
- Directorio de trabajo
- Repositorio
  - Local
  - Remoto
  - Con Ramas
- Características
  - Mecanismo de almacenamiento
  - Realizar cambios
  - Registro histórico
  - Control de cambios
    - Quien
    - Que
    - Donde
  - Unificar
  - Volver atrás
- Tipos
  - Centralizados
  - Distribuidos
- Conclusiones
- Bibliografía

1. Introducción

En los tiempos en los que vivimos, prácticamente todo el mundo maneja ficheros, archivos, directorios, etc… Pues estamos inmersos en un mundo tecnológico y el uso de sistemas informáticos está a la orden del día.

Por lo tanto, necesitamos gestionar y manejar estos archivos, lo que podemos hacer de forma manual o con el uso de herramientas para este fin, lo cual es más recomendable. Y es que, cuando manejamos un archivo, éste sufre varias modificaciones a lo largo del tiempo, algunas de las cuáles nos podría interesar acceder en algún momento. Para ello es conveniente usar un sistema de control de versiones.

2. Control de versiones. ¿Qué es? Definición

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 se puedan recuperar versiones específicas más adelante.

Para realizar este control se utilizan herramientas de software, que se encargan de manera automática de su gestión.

El control de versiones se realiza mediante tecnologías y técnicas (que veremos más adelante) sobre un proyecto (un código fuente de un programa, una documentación, una página web…).

3. ¿Qué es un directorio de trabajo?

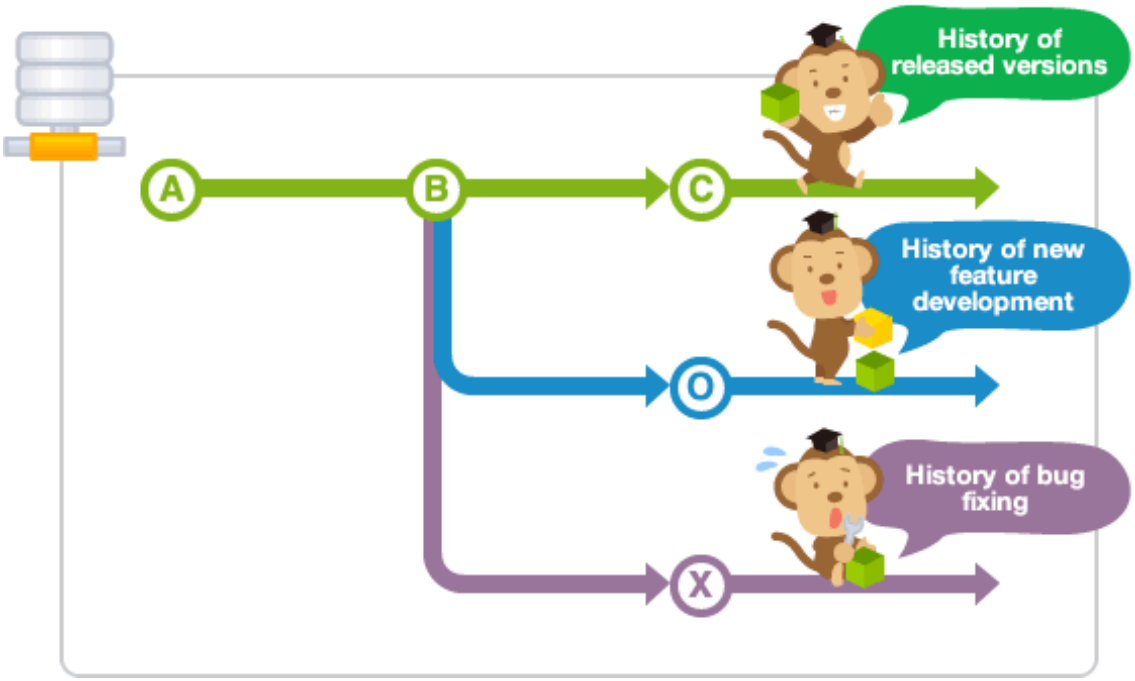
El directorio de trabajo es el espacio físico donde se almacenan los elementos que forman un proyecto. Sólo puede acceder a él el usuario que lo posee y no se comparte, como es el caso de un repositorio.

4. ¿Qué es un repositorio?

Como en el caso anterior, es el espacio físico donde se almacenan los elementos de un proyecto, pero la diferencia con el directorio de trabajo es que el repositorio puede ser compartido, siendo utilizado por varias personas y que incluye un historial de versiones de los elementos guardados, modificados y eliminados.

4.1. Tipos de repositorios:

- **Local:** es el repositorio que se encuentra en el sistema informático con el que se trabaja, desde el que se modifica y se ve su contenido, pudiendo ser su contenido original o copia de un repositorio no local.
- **Remoto:** es el repositorio que se encuentra físicamente en un sistema informático situado en la red, diferente del que se usa para trabajar con él. Los usuarios se conectan a través de la red a éste para trabajar con los datos contenidos en el repositorio, creando una copia de este en sus propias computadoras.



- **Con Ramas (branch):** existe un repositorio central donde se contiene el código y luego cada usuario tiene en su máquina local una copia de ese código, parte de él o incluso una modificación del mismo. Pero siempre lo que se tiene es una copia. Se requiere de un usuario que se encargará de centralizar y unir todas las ramas o, incluso, desechar el trabajo hecho en alguna de ellas.

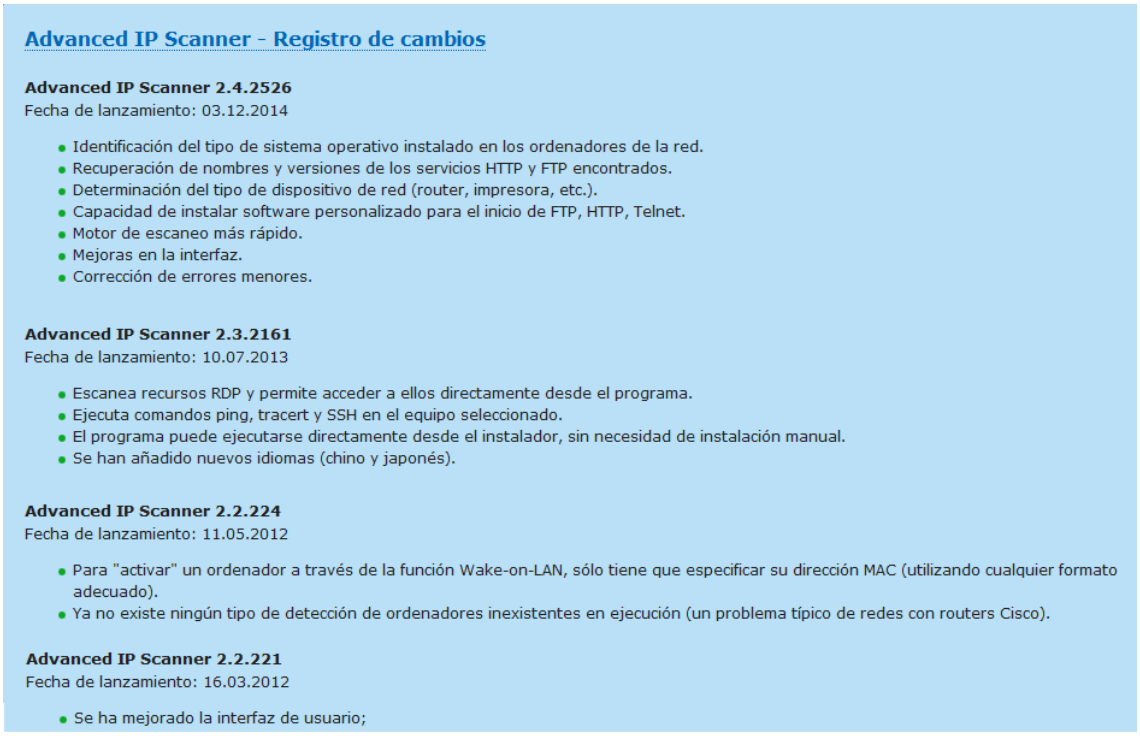
5. Características:

- **Mecanismo de almacenamiento:** en los repositorios se almacenan los elementos que queremos gestionar (archivos de texto, imágenes, documentación…).

- **Posibilidad de hacer cambios:** sobre los elementos almacenados pueden realizarse cambios, totales o parciales, como por ejemplo: añadir, borrar, renombrar o mover elementos.

- **Registro histórico:** todas las acciones realizadas con cada elemento son registradas de forma cronológica.

- **Control de cambios:** Cuando estamos hablando de proyectos de código fuente, normalmente se trabaja en equipo, que estará formado por varios programadores y coordinado por un jefe de proyecto. Pues bien se registra quién ha modificado cada elemento, qué ha modificado y cuándo, lo que es imprescindible para mantener el control sobre el proyecto. Las etiquetas (tags) marcan las diferentes versiones creadas con cada modificación.



Advanced IP Scanner - Registro de cambios

Advanced IP Scanner 2.4.2526
Fecha de lanzamiento: 03.12.2014

- Identificación del tipo de sistema operativo instalado en los ordenadores de la red.
- Recuperación de nombres y versiones de los servicios HTTP y FTP encontrados.
- Determinación del tipo de dispositivo de red (router, impresora, etc.).
- Capacidad de instalar software personalizado para el inicio de FTP, HTTP, Telnet.
- Motor de escaneo más rápido.
- Mejoras en la interfaz.
- Corrección de errores menores.

Advanced IP Scanner 2.3.2161
Fecha de lanzamiento: 10.07.2013

- Escanea recursos RDP y permite acceder a ellos directamente desde el programa.
- Ejecuta comandos ping, tracert y SSH en el equipo seleccionado.
- El programa puede ejecutarse directamente desde el instalador, sin necesidad de instalación manual.
- Se han añadido nuevos idiomas (chino y japonés).

Advanced IP Scanner 2.2.224
Fecha de lanzamiento: 11.05.2012

- Para "activar" un ordenador a través de la función Wake-on-LAN, sólo tiene que especificar su dirección MAC (utilizando cualquier formato adecuado).
- Ya no existe ningún tipo de detección de ordenadores inexistentes en ejecución (un problema típico de redes con routers Cisco).

Advanced IP Scanner 2.2.221
Fecha de lanzamiento: 16.03.2012

- Se ha mejorado la interfaz de usuario;

- **Unificar:** la decisión final de unificar las diferentes partes, incluso cuando dos personas han realizado cambios incompatibles, es la tarea del director del proyecto.

- **Volver atrás:** existe la posibilidad de volver atrás en caso de que los cambios realizados no vayan por el camino adecuado.

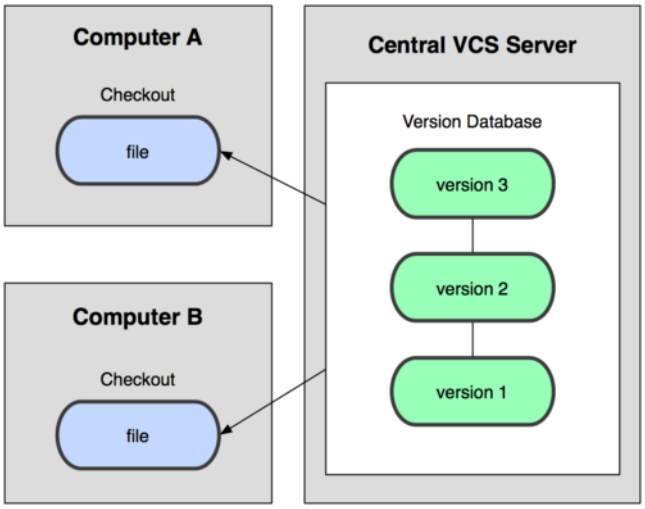
6. Tipos de control de versiones

- **Centralizados:** como mencionamos anteriormente, un repositorio puede ser de tipo remoto, lo que significa que utiliza un enfoque tipo cliente-servidor.

  Este tipo de sistema de control de versiones, tiene un único servidor que contiene todos los archivos, y varios clientes que descargan los archivos de ese lugar central.

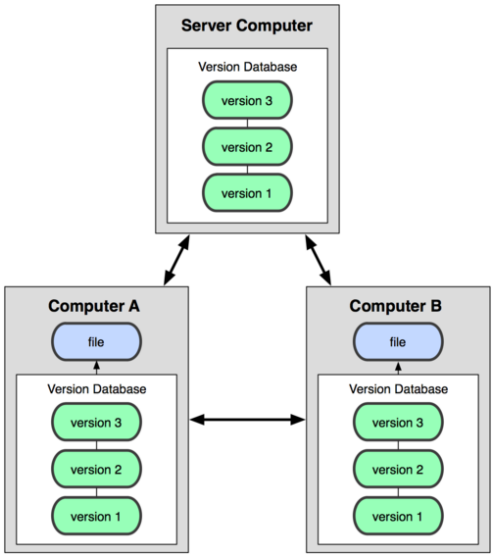
  Durante muchos años, éste ha sido el estándar para el control de versiones, es decir, de este repositorio centralizado que contiene todo el código del que es responsable un usuario, de modo que todas las decisiones importantes recaen sobre este usuario. Software de ejemplo de este tipo serían: Subversion y Perforce.

Esta configuración también tiene desventajas: si ese servidor deja de funcionar durante una hora, entonces durante esa hora nadie puede colaborar o guardar cambios sobre los archivos. Por otro lado, si el disco duro en el que se encuentra se corrompe, y no se han realizado copias de seguridad, se pierde absolutamente todo, salvo lo que esté en las máquinas locales.



- **Distribuidos:** Utiliza el enfoque tipo p2p (punto a punto). Los clientes no sólo descargan la última versión de los archivos, sino también las versiones antiguas y el historial de cambios (replican completamente el repositorio). Así, si un servidor deja de funcionar, cualquiera de los repositorios de los clientes puede copiarse en el servidor para restaurarlo. Como ejemplos de este tipo son Git, Mercurial, Bazaar o Darcs.

  Cada vez que se descarga una versión, en realidad se hace una copia de seguridad completa de todos los datos. Cada usuario tiene su propio repositorio.



7. CONCLUSIONES:

Como conclusión podemos decir que los repositorios locales están en desventaja con los repositorios remotos, ya que estos últimos ofrecen la posibilidad de tener varias copias del mismo código y en caso de momento fatal pueden restaurarse.

Trabajar en ramas agiliza la gestión del código ya que la carga de trabajo es repartida entre varios programadores y se gana velocidad a la hora de generar nuevas versiones. Podemos ver que en los trabajos colaborativos mediante este tipo de herramientas los diferentes programadores a veces trabajan por porciones de código diferente y otras con todo el código en común.

Para centralizar todo este trabajo existe una figura que es el administrador que se encarga de tomar las decisiones importantes y también de unificar el código.

A la hora de hablar de los distribuidos y de los centralizados el tipo de funcionamiento de cada uno de ellos nos ofrece unas posibilidades más amplias ya que la configuración de estos mismos nos ofrece más seguridad, menos perdida de información y al ser herramientas colaborativas especializan mucho el trabajo de cada uno de los miembros.

Respecto a las velocidades de cada uno de estos sistemas, en diversos foros y páginas especializadas se habla de que por ejemplo herramientas como Git, Mercurial, Bazaar o Darcs son mejores que otras como por ejemplo Subversion o Perforce; pero esto todo a base de experiencias y opiniones personales de usuarios.

8. Bibliografía

- Librosweb.es: Pro Git, el libro oficial de Git
- Iesgrancapitan.org: Temario Entornos de Desarrollo
- Wikipedia.org: Control de Versiones
- Monografias.com: Control de Versiones