Qué es el control de versiones

Ventajas del control de versiones

Los sistemas de control de versiones (o VCS por sus siglas en inglés) son una categoría de herramientas de software que ayudan a un equipo de software a gestionar los cambios en el código fuente a lo largo del tiempo.

El software de control de versiones realiza un seguimiento de todas las modificaciones en el código en un tipo especial de base de datos. Si se comete un error, los desarrolladores pueden ir atrás en el tiempo y comparar las versiones anteriores del código para ayudar a resolver el error al tiempo que se minimizan las interrupciones para todos los miembros del equipo.

Para todos los proyectos de software, el código fuente es un activo muy valioso que debe protegerse. Para la mayoría de los equipos de software, el código fuente es un repositorio de conocimiento de valor incalculable y de la comprensión sobre el dominio del problema que los desarrolladores han recopilado y perfeccionado con un esfuerzo cuidadoso. El control de versiones protege el código fuente tanto de las catástrofes como del deterioro casual de los errores humanos y las consecuencias accidentales.

Los desarrolladores de software están continuamente modificando el código fuente, agregando nuevo código y cambiando el que ya existe. El código fuente de un proyecto, una aplicación o un componente de software, normalmente se organiza en una estructura de carpetas o "árbol de archivos". Un desarrollador del equipo podría estar trabajando en una nueva función mientras otro desarrollador podría estar solucionando un error no relacionado que también requiere cambiar código. Cada desarrollador podría hacer sus cambios en varias partes del árbol de archivos.

El control de versiones ayuda a los equipos a resolver este tipo de problemas al realizar un seguimiento de todos los cambios individuales de cada colaborador y al contribuir a evitar que el trabajo concurrente entre en conflicto. Los cambios realizados en una parte del software pueden ser incompatibles con los que ha hecho otro desarrollador que está trabajando al mismo tiempo. Este problema debería detectarse y solucionarse de manera ordenada sin bloquear el trabajo del resto del equipo.

Además, en todo el desarrollo de software, cualquier cambio puede introducir nuevos errores por sí mismo y el nuevo software no es fiable hasta que se prueba. De este modo, las pruebas y el desarrollo van de la mano hasta que está lista una nueva versión.

Un buen software de control de versiones soporta el flujo de trabajo preferido de un desarrollador o equipo de desarrollo sin imponer una forma determinada de trabajar. Idealmente, también funciona en cualquier plataforma, en vez de ordenar qué sistema operativo o cadena de herramientas deben utilizar los desarrolladores. Los sistemas de control de versiones excepcionales facilitan un flujo sencillo y continuo de cambios en el código en vez del mecanismo frustrante y burdo del bloqueo de archivos, que da luz verde a un desarrollador a expensas de bloquear el progreso de los demás.

Los equipos de software que no utilizan ninguna forma de control de versiones a menudo se encuentran con problemas como no saber qué cambios que se han hecho están disponibles para los usuarios o la

creación de cambios incompatibles entre dos partes no relacionadas que tienen que desvincularse y revisarse exhaustivamente.

Si nunca has utilizado el control de versiones, puede que hayas añadido versiones a tus archivos, quizás con sufijos como "final" o "más reciente", y que después hayas tenido que enfrentarte con una nueva versión final. Quizás has convertido en comentarios bloques de código, porque quieres desactivar una determinada función sin eliminar el código, con el miedo de que pueda utilizarse más adelante. El control de versiones es una forma de solucionar estos problemas.

El software de control de versiones es una parte esencial del día a día de las prácticas profesionales del equipo de software moderno. Los desarrolladores de software individuales que están acostumbrados a trabajar con un sistema de control de versiones potente en sus equipos suelen reconocer el increíble valor que el control de versiones también les da incluso en los proyectos pequeños en los que trabajan solos. Una vez acostumbrados a las potentes ventajas de los sistemas de control de versiones, muchos desarrolladores no se plantearían trabajar sin ellos incluso para los proyectos que no son de software.

Ventajas de los sistemas de control de versiones

Desarrollar software sin utilizar el control de versiones es arriesgado, equiparable a no tener copias de seguridad. El control de versiones también puede permitir que los desarrolladores se muevan más rápido y posibilita que los equipos de software mantengan la eficacia y la agilidad a medida que el equipo se escala para incluir más desarrolladores.

Los sistemas de control de versiones (VCS) han experimentado grandes mejoras en las últimas décadas y algunos son mejores que otros. Los VCS a veces se conocen como herramientas de SCM (Gestión del código fuente) o RCS (Sistema de control de revisiones). Una de las herramientas de VCS más populares hoy en día se llama Git. Git es un VCS distribuido, una categoría conocida como DVCS, que explicaremos con más detalle después. Al igual que muchos de los sistemas de VCS más populares disponibles hoy en día, Git es gratuito y de código abierto. Independientemente del nombre que tengan, o del sistema que se utilice, las principales ventajas que deberías esperar del control de versiones son las siguientes.

- Historial completo de cambios a largo plazo de todos los archivos. Esto incluye todos los cambios realizados por muchas personas a lo largo de los años. Los cambios incluyen la creación y la eliminación de los archivos, así como los cambios de sus contenidos. Las diferentes herramientas de VCS difieren en lo bien que gestionan el cambio de nombre y el movimiento de los archivos. Este historial también debería incluir el autor, la fecha y notas escritas sobre el propósito de cada cambio. Tener el historial completo permite volver a las versiones anteriores para ayudar a analizar la causa raíz de los errores y es crucial cuando se tiene que solucionar problemas en las versiones anteriores del software. Si se está trabajando de forma activa en el software, casi todo puede considerarse una "versión anterior" del software.
- Creación de ramas y fusiones (merge). Si se tiene a miembros del equipo trabajando al mismo tiempo, es algo evidente; pero incluso las personas que trabajan solas pueden beneficiarse de la capacidad de trabajar en flujos independientes de cambios. La creación de una “rama” o “Branch” en las herramientas de VCS mantiene múltiples flujos de trabajo independientes los unos de los otros al tiempo que ofrece la facilidad de volver a fusionar ese trabajo, lo que permite que los desarrolladores verifiquen que los cambios de cada rama no entran en conflicto. Muchos equipos

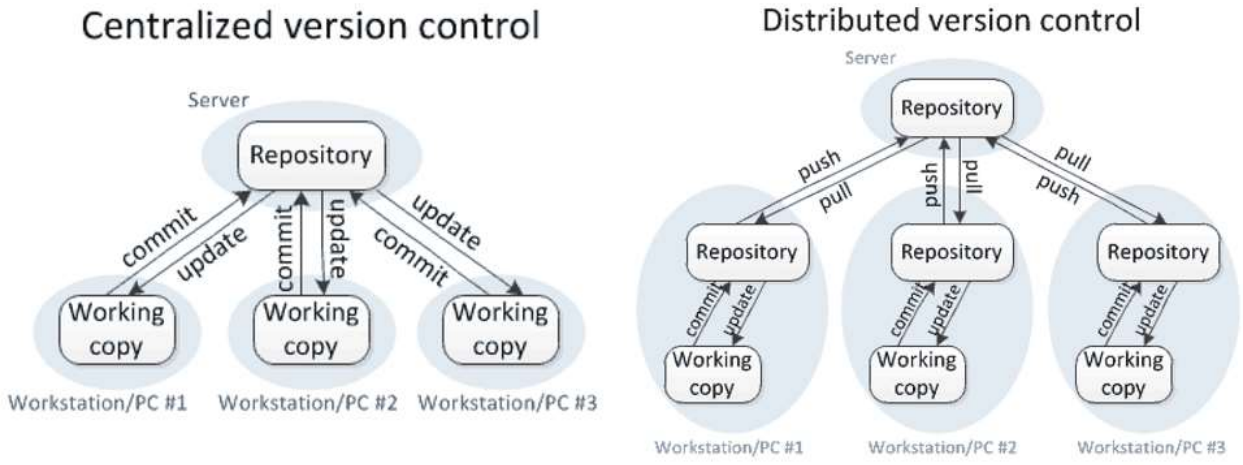
de software adoptan la práctica de crear ramas para cada función o quizás para cada publicación, o ambas. Existen muchos flujos de trabajo diferentes que los equipos pueden elegir cuando deciden cómo utilizar la creación de las ramas y las fusiones en VCS.

- Trazabilidad. Ser capaz de trazar cada cambio que se hace en el software y conectarlo con un software de gestión de proyectos y seguimiento de errores como Jira, además de ser capaz de anotar cada cambio con un mensaje que describa el propósito y el objetivo del cambio, no solo te ayuda con el análisis de la causa raíz y la recopilación de información. Tener el historial anotado del código a tu alcance cuando estás leyendo el código, intentando entender lo que hace y por qué se ha diseñado así, puede permitir a los desarrolladores hacer cambios correctos y armoniosos que estén en línea con el diseño previsto a largo plazo del sistema. Esto puede ser especialmente importante para trabajar de manera eficaz con código heredado y es esencial para que los desarrolladores puedan calcular el trabajo futuro con precisión.

Aunque se puede desarrollar software sin utilizar ningún control de versiones, hacerlo somete al proyecto a un gran riesgo que ningún equipo profesional debería aceptar. Así que la pregunta no es si utilizar el control de versiones, sino qué sistema de control de versiones usar.

Tipos de VCS

Las herramientas del VCS se dividen en dos tipos principales de arquitectura remota. Estos tipos de arquitectura son: centralizados y distribuidos.



Cuando hablamos de los pros y los contras de cada arquitectura, la función de copia de seguridad externa es el principal punto de debate. Un VCS centralizado cuenta con un solo punto de error, que es la instancia remota del VCS central. Si se pierde dicha instancia, puede producir la pérdida de datos y de la productividad, y se deberá sustituir por otra copia del código fuente (pero la historia se pierde).

Asimismo, si se vuelve temporalmente no disponible, evitará que los desarrolladores envíen, fusionen o reviertan código.

Un modelo distribuido de arquitectura evita estos obstáculos manteniendo una copia total del código fuente en cada instancia de VCS. Si en el modelo distribuido se produce cualquiera de los casos de error

centralizados antes mencionados, se puede designar una instancia del VCS como principal mitigando cualquier caída grave de productividad, ya que todas las instancias son clones completos entre sí.