Versions - General

29-11-2022

Leandro Baier

Metas

Temas a revisar:

1. ¿Qué es Version Control Coding?
2. ¿Cuáles son los principales tipos de sistemas de Version Control?
3. ¿Cuáles son los conceptos básicos? (Trackeo de cambios, commits, revisión, updates, conflictos, diferenciación, branches y merges, etc)

1. ¿Qué es Version Control Coding?

Es la práctica de rastrear y gestionar los cambios en el código de software.

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 hacia 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.

2. ¿Cuáles son los principales tipos de sistemas de Version Control?

- Git: es una de las mejores herramientas de control de versiones disponible en el mercado actual. Es un modelo de repositorio distribuido compatible con sistemas y protocolos existentes como HTTP, FTP, SSH y es capaz de manejar eficientemente proyectos pequeños a grandes.
- CVS: es otro sistema de control de versiones muy popular. Es un modelo de repositorio cliente-servidor donde varios desarrolladores pueden trabajar en el mismo proyecto en paralelo. El cliente CVS mantendrá actualizada la copia de trabajo del archivo y requiere intervención manual sólo cuando ocurra un conflicto de edición.
- Apache Subversion (SVN): abreviado como SVN, apunta a ser el sucesor más adecuado. Es un modelo de repositorio cliente-servidor donde los directorios están versionados junto con las operaciones de copia, eliminación, movimiento y cambio de nombre.
- Mercurial: es una herramienta distribuida de control de versiones que está escrita en Python y destinada a desarrolladores de software. Los sistemas operativos que admite son similares a Unix, Windows y macOS. Tiene un alto rendimiento y escalabilidad con capacidades avanzadas de ramificación y fusión y un desarrollo colaborativo totalmente distribuido. Además, posee una interfaz web integrada.

- Monotone: está escrito en C ++ y es una herramienta para el control de versiones distribuidas. El sistema operativo que admite incluye Unix, Linux, BSD, Mac OS X y Windows. Brinda un buen apoyo para la internacionalización y localización. Además, utiliza un protocolo personalizado muy eficiente y robusto llamado Netsync.

3. ¿Cuáles son los conceptos básicos? (Trackeo de cambios, commits, revisión, updates, conflictos, diferenciación, branches y merges, etc)

- Trackeo de cambios: En muchos sistemas de control de versiones con commits multi-cambio atómicos, una lista de cambios identifica el conjunto de cambios hechos en un único commit. Esto también puede representar una vista secuencial del código fuente, permitiendo que el fuente sea examinado a partir de cualquier identificador de lista de cambios particular.
- Commits: Un commit sucede cuando una copia de los cambios hechos a una copia local es escrita o integrada sobre el repositorio.
- Revisión: Una revisión es una versión determinada de la información que se gestiona. Hay sistemas que identifican las revisiones con un contador (Ej. subversión). Hay otros sistemas que identifican las revisiones mediante un código de detección de modificaciones (Ej. Git usa SHA1). A la última versión se le suele identificar de forma especial con el nombre de HEAD. Para marcar una revisión concreta se usan los rótulos o tags.
- Updates: Una actualización integra los cambios que han sido hechos en el repositorio (por ejemplo por otras personas) en la copia de trabajo local.
- Conflictos: Un conflicto ocurre cuando el sistema no puede manejar adecuadamente cambios realizados por dos o más usuarios en un mismo archivo. Por ejemplo, si se da esta secuencia de circunstancias:
    - Los usuarios X e Y despliegan versiones del archivo A en que las líneas n1 hasta n2 son comunes.
    - El usuario X envía cambios entre las líneas n1 y n2 al archivo A.
    - El usuario Y no actualiza el archivo A tras el envío del usuario X.
    - El usuario Y realiza cambios entre las líneas n1 y n2.
    - El usuario Y intenta posteriormente enviar esos cambios al archivo A.
    - El sistema es incapaz de fusionar los cambios. El usuario Y debe resolver el conflicto combinando los cambios, o eligiendo uno de ellos para descartar el otro.
- Diferenciación:
- Branches: Un módulo puede ser branched o 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. El

módulo tiene entonces 2 (o más) "ramas". La ventaja es que se puede hacer un "merge" de las modificaciones de ambas ramas, posibilitando la creación de "ramas de prueba" que contengan código para evaluación, si se decide que las modificaciones realizadas en la "rama de prueba" sean preservadas, se hace un "merge" con la rama principal. Son motivos habituales para la creación de ramas la creación de nuevas funcionalidades o la corrección de errores.

- Merges: Una integración o fusión une dos conjuntos de cambios sobre un fichero o un conjunto de ficheros en una revisión unificada de dicho fichero o ficheros.
    - Esto puede suceder cuando un usuario, trabajando en esos ficheros, actualiza su copia local con los cambios realizados, y añadidos al repositorio, por otros usuarios. Análogamente, este mismo proceso puede ocurrir en el repositorio cuando un usuario intenta check-in sus cambios.
    - Puede suceder después de que el código haya sido branched, y un problema anterior al branching sea arreglado en una rama, y se necesite incorporar dicho arreglo en la otra.
    - Puede suceder después de que los ficheros hayan sido branched, desarrollados de forma independiente por un tiempo, y que entonces se haya requerido que fueran fundidos de nuevo en un único trunk unificado.