DESARROLLO DE SISTEMAS II

Herramientas de Versionamiento de Código

9 DE SEPTIEMBRE DE 2024

DANIEL ROMAN TERRAZAS
UNIVERSIDAD DOMINGO SAVIO

Contenido

Introducción

El versionamiento de código es una práctica esencial en el desarrollo de software moderno. Permite gestionar y rastrear los cambios en el código fuente a lo largo del tiempo, ofreciendo un historial detallado de todas las modificaciones realizadas. Las herramientas de control de versiones (VCS, por sus siglas en inglés), como Git, Mercurial y Subversion, facilitan la colaboración entre desarrolladores, minimizan conflictos, y aseguran que siempre se pueda restaurar el código a un estado anterior si es necesario.

¿Por qué es importante el versionamiento de código?

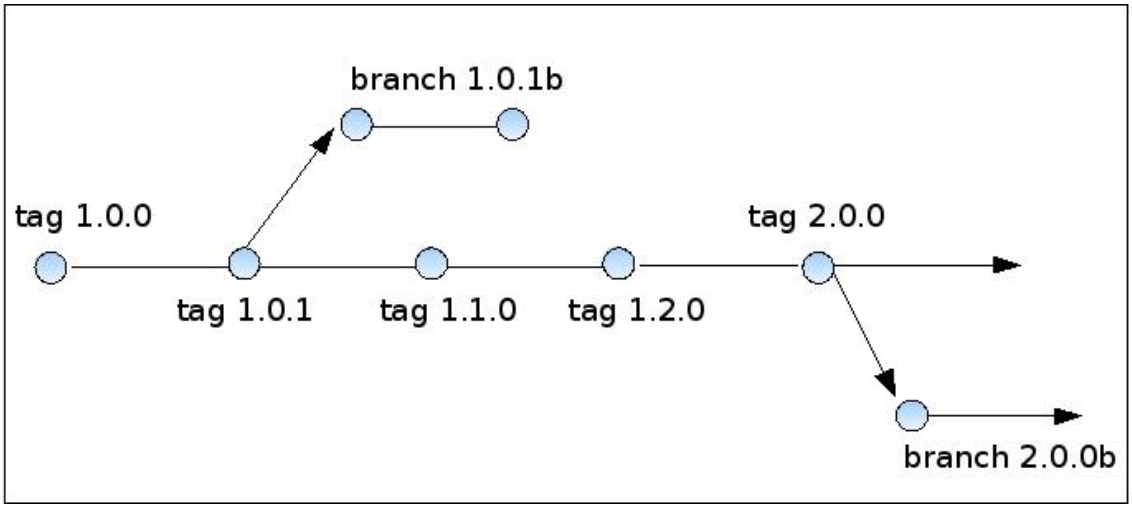
1. **Colaboración eficiente**: En proyectos donde participan varios desarrolladores, es fundamental poder trabajar de forma simultánea en diferentes partes del código sin interferir con el trabajo de otros. Las herramientas de versionamiento permiten que los cambios se integren de manera organizada y controlada.
2. **Historial de cambios**: El versionamiento guarda un registro detallado de cada modificación, incluyendo qué cambios se hicieron, quién los realizó y cuándo. Esto permite rastrear errores o revisar cómo ha evolucionado el código con el tiempo.
3. **Facilidad para revertir cambios**: Si un error grave se introduce en una actualización o algún cambio no deseado se comete, puedes fácilmente regresar a una versión anterior del código.
4. **Manejo de ramas (branches)**: Las herramientas de control de versiones permiten crear ramas para trabajar en diferentes características o correcciones sin afectar el código principal. Esto facilita la experimentación sin poner en riesgo la estabilidad del proyecto.

Herramientas más comunes

1. **Git**: Es la herramienta de control de versiones más popular y utilizada. Ofrece una gran flexibilidad para manejar múltiples ramas y cambios simultáneos. Git funciona de manera distribuida, lo que significa que cada desarrollador tiene una copia completa del historial del proyecto. GitHub, GitLab y Bitbucket son algunas de las plataformas que alojan repositorios Git.
2. **Subversion (SVN)**: Es un sistema de control de versiones centralizado que permite el acceso y control desde un único repositorio maestro. Aunque ha perdido popularidad frente a Git, sigue siendo útil en proyectos con estructuras más simples.
3. **Mercurial**: Similar a Git, es un sistema de control de versiones distribuido. Aunque no es tan popular como Git, es conocido por su simplicidad y rendimiento en proyectos grandes.

Conceptos Clave

1. **Commit**: Un commit representa un cambio realizado en el código. Este incluye un mensaje descriptivo que ayuda a entender qué se ha modificado.
2. **Branch (rama)**: Una rama es una línea paralela de desarrollo en la que se puede trabajar de manera aislada sin afectar la rama principal (generalmente llamada "main" o "master").
3. **Merge**: El proceso de unir cambios de una rama a otra.
4. **Pull Request (o Merge Request)**: Es una solicitud para revisar y, eventualmente, fusionar los cambios realizados en una rama a la rama principal del proyecto.
5. **Repositorio**: Es el espacio donde se almacena el código y su historial de versiones.



Buenas Prácticas

1. **Commits frecuentes y pequeños**: Realiza commits con regularidad y mantén los cambios pequeños y manejables. Esto facilita la revisión y reduce la posibilidad de conflictos.
2. **Mensajes descriptivos en los commits**: Siempre incluye mensajes claros y específicos en tus commits. Esto ayudará a otros (y a ti mismo) a entender qué cambios se realizaron y por qué.
3. **Uso de ramas para cada nueva funcionalidad**: Es recomendable crear una nueva rama para cada nueva funcionalidad o corrección. Esto te permite trabajar de forma aislada y mantener la rama principal libre de errores hasta que se haya probado el nuevo código.
4. **Revisiones de código**: Al realizar una solicitud de pull o merge, asegúrate de que otro desarrollador revise el código antes de fusionarlo. Esto ayuda a detectar errores y mejora la calidad del código.