MEDAC
Instituto Oficial de Formación Profesional

Sistemas de Control de Versiones - Git y Subversion

Ramón Montoya Romero

Sistemas de Control de Versiones - Version Control System (VCS)

Su función principal es acoger a un proyecto y realizar el seguimiento de su desarrollo a lo largo del tiempo.

Un VCS, permite a los desarrolladores comparar cambios en archivos, realizar un seguimiento de los commits (confirmaciones de cambios), proponer cambios, ver el historial del proyecto, volver a versiones anteriores y más.

Un buen VCS es esencial para los desarrolladores software, ya trabajen en equipo o no, las metodologías de desarrollo ágiles como Scrum o ExtremeProgramming, cascada o espiral entre otras, se basan en la entrega continua de los distintos estados del proyecto a medida que progresa, el uso de vcs ayuda a mantener un flujo de trabajo productivo, pues se lleva detalladamente el historial del proyecto…..

Algunos de los sistemas de control de versiones más utilizados son Git, SVN, Mercurial y Perforce.

Optar por un vcs senta las bases en el cómo el equipo va construir, desarrollar el proyecto; por lo que es crucial y los desarrolladores tienen sus opiniones firmes al respecto.

Git

Lo creó Linus Torvalds en 2005, fue diseñado para mantener el desarrollo del Kernel de Linux.

Git se construyó sobre los pilares de la **distribución completa**, velocidad, diseño simple, capacidad para manejar grandes proyectos y un **fuerte soporte para el desarrollo no lineal**.

Es muy importante comprender que Git se creó para abordar problemas comunes del momento de una manera nueva y perfeccionar el modelo distribuido.

Subversion(SVN)

Subversion , fue fundada en el año 2000 por *CollabNet, Inc*. bajo la licencia Apache.

SVN se distribuye como código abierto, y su **modelo** utiliza un sistema de control de versiones **centralizado**, lo que significa que toda la información y los archivos se almacenan en un servidor central.

Afirma que su propósito es ser “universalmente reconocido y adoptado como un VCS centralizado de código abierto, caracterizado por su confiabilidad como refugio para datos valiosos; la sencillez de su modelo y uso; y su capacidad para satisfacer las necesidades de una amplia variedad de usuarios y proyectos, desde individuos hasta operaciones empresariales a gran escala”.

Modelo Centralizado vs Distribuido

La distinción más polarizadora entre *Git* y *SVN* se encuentra en la ***arquitectura*** central: ***distribuida*** y ***centralizada*** respectivamente.

Esta diferencia crítica, tanto en el proceso como en la ideología a la hora de desarrollar un proyecto, es a menudo el quid que determina qué *VCS* utiliza un individuo o equipo.

¿En qué se diferencian y cuáles son las ventajas de cada uno?

Modelo Centralizado

Tener un **sistema centralizado,** como *SVN,* significa que *hay una única "copia maestra" o tronco (**trunk**) de un proyecto, y esta reside en el servidor. Sólo aquellos archivos que se modifican o revisan, se almacenan localmente en el equipo del desarrollador*.

Cuando los desarrolladores reclaman o retiran esos archivos del servidor, pueden restringir el acceso de otros miembros del equipo a ellos hasta que sean retornados. Esto puede ayudar a reducir el riesgo de conflictos de fusión, pero también puede interrumpir los flujos de trabajo si varias personas necesitan realizar cambios en un solo archivo.

Cuando los desarrolladores estén listos para enviar sus cambios, deben enviar sus versiones del código al servidor. Este enfoque controlado del flujo de trabajo de codificación atrae a algunos debido a su naturaleza segura y central, su proceso claro para publicar código y la necesidad de menos almacenamiento de archivos.

Modelo Distribuido

En un sistema distribuido, todos los desarrolladores pueden tener mismo acceso a todos los archivos de un proyecto. *Cualquiera con los permisos adecuados puede realizar una copia del repositorio principal (**git clone**) y tener acceso a todo el historial del proyecto en su equipo local*, incluidos todos los metadatos y confirmaciones de cambios (***commits***) asociados.

Esto también significa que, excepto empujar (***push***) cambios al repositorio remoto y tirar (***pull***) traer el repositorio al entorno local , todas las demás acciones solo afectan a los archivos en el disco local del desarrollador, en lugar de a los del servidor remoto.

Esto *permite que varios miembros del equipo de desarrollo accedan al mismo archivo remoto, descarguen (**pull**), y realicen cambios de forma local e independiente; pero esto conlleva el riesgo de conflictos en fusión (**merge**), cuando con el archivo remoto y objeto de cambios, se intenta fusionar alguna versión que pueda ser incompatible. Es entonces cuando el administrador del proyecto, deberá decidir qué cambios incorporar y cuáles descartar, para cumplir con los requisitos del proyecto*.

Ramificación en Git y SVN - ***Branches***

En SVN las ramificaciones, bifurcaciones, etiquetas y troncales (***trunk***) se crean como directorios dentro del repositorio central (*remoto*).

Todas ellas son públicas, por lo que los miembros del equipo de desarrollo tienen acceso y pueden verlas. Esto incluye ramas que solo se crearon para probar ideas o realizar modificaciones menores, lo que puede ser problemático por ejemplo, al buscar un archivo específico y tener que navegar por ramas que no vuelven a la principal, o los archivos de prueba de cada miembro del equipo para obtener lo que se está buscando.

Un flujo de trabajo genérico en SVN:

- ➔ El directorio trunk representa la versión más estable y reciente de un proyecto.
- ➔ El desarrollo de nuevas características se desarrolla dentro de subdirectorios (branches).
- ➔ Una vez finalizada una característica, el directorio branches se combina

```
sample_project/trunk/README.md
sample_project/trunk/lib/widget.rb
sample_project/branches/new_feature/README.md
sample_project/branches/new_feature/lib/widget.rb
sample_project/branches/another_new_feature/README.md
sample_project/branches/another_new_feature/lib/widget.rb
```

Ramificación en Git y SVN - ***Branches***

Git crea las ramas a partir de confirmaciones de cambio (***commits***) específicos. Esto significa que, cuando un desarrollador crea una rama ,puede actualizar, alterar o eliminar código sin tener que preocuparse de cómo sus cambios afectan otras confirmaciones en el proyecto principal.

Los proyectos en Git también se almacenan dentro de un directorio único. Pero Git oculta los detalles de sus referencias almacenándolos en un directorio ***.git*** especial.

Un flujo de trabajo genérico Git:

- ➔ El directorio ***.git*** almacena el historial completo de todas sus ramas y etiquetas.
- ➔ El último lanzamiento estable se encuentra en la rama predeterminada (main)
- ➔ El desarrollo de nuevas características se desarrolla en ramas separas.
- ➔ Cuando una característica finaliza, la rama sobre la que se ha desarrollado se fusiona en la rama predeterminada.

Git y SVN - **Disponibilidad sin conexión**

Una ventaja de Git es su funcionalidad offline, debido a que los desarrolladores tienen acceso a todo el proyecto en su equipo local, pueden trabajar localmente y ver cómo sus versiones interactúan y encajan con el proyecto en su conjunto antes de afectar el código base compartido en remoto.

Git no requiere acceso constante al servidor principal y produce un tráfico de red más liviano si trabajas en local, mitiga el riesgo de perder cambios. Solo necesita estar conectado al repositorio principal cuando realiza ***push*** y ***pull***.

SVN, por otro lado, requiere que cada cambio, confirmación y acción pase por el servidor central. Entonces, si el proyecto se basa en este modelo centralizado, y el repositorio central falla, no se puede trabajar hasta que se solucione; todo se ejecuta en el servidor principal, creando un precario punto único de falla.

Git y SVN - **Comparativa de comandos y referencias**

Command	Operation	Subversion
git clone	Copy a repository	svn checkout
git commit	Record changes to file history	svn commit
git show	View commit details	svn cat
git status	Confirm status	svn status
git diff	Check differences	svn diff
git log	Check log	svn log
git add	Addition	svn add
git mv	Move	svn mv
git rm	Delete	svn rm
git checkout	Cancel change	svn revert[1]

Command	Operation	Subversion
git reset	Cancel change	svn revert[1]
git branch	Make a branch	svn copy[2]
git checkout	Switch branch	svn switch
git merge	Merge	svn merge
git tag	Create a tag	svn copy[2]
git pull	Update	svn update
git fetch	Update	svn update
git push	It is reflected on the remote	svn commit[3]
gitignore	Ignore file list	.svnignore

- ➔ Git vs Subversion en Gitkraken.
- ➔ Git vs Subversion según Github.

MEDAC
Instituto Oficial de Formación Profesional