Sistemas de control de Versiones

Definición

Un sistema de control de versiones **es una herramienta que registra todos los cambios hechos en uno o más proyectos**, guardando así versiones del producto en todas sus fases del desarrollo. Las versiones son como fotografías que registran su estado en ese momento del tiempo y se van guardando a medida que se hacen modificaciones al código fuente.

Un poco de contexto

Antes de la masificación de los sistemas de control de versiones, se solían guardar los hitos (una especie de versión) del proyecto en archivos comprimidos.

Los sistemas de control de versiones nacen de la necesidad de solventar y facilitar este tedioso proceso.

Para un desarrollador esta herramienta es muy valiosa porque permite viajar atrás en el tiempo (hacer rollback) si los cambios aplicados no resultaron de la manera que se esperaba, pudiendo restaurar en cualquier momento una versión previa. Es que tener la posibilidad de volver atrás no tiene precio

Diferentes Modelos

Modelo cliente-servidor

Los desarrolladores usan un repositorio central al que acceden mediante un cliente en su máquina

Modelo distribuido

En el modelo distribuido, cada desarrollador trabaja directamente con su repositorio local, y los cambios se comparten entre repositorios en un paso posterior.

Algunos Versionadores

Modelo Cliente Servidor

Código abierto

- Concurrent Versions System (CVS): basado originalmente en RCS
- Subversion (svn): inspirado en CVS.
- Vesta: sistema de construcción con soporte para versionado de ficheros en repositorios distribuidos.

Propietario

- Vault: herramienta de control de versiones de SourceGear (primer uso gratis).
- Visual SourceSafe: herramienta de control de versiones de Microsoft.
- Visual Studio Team Foundation Server (anteriormente Team System): orientada a Plataforma Microsoft .Net.

Modelo distribuido

Código abierto

- DCVS: CVS descentralizado.
- Fossil: (Richard Hipp), presenta un control de versiones distribuido, wiki y seguimiento de fallos.
- Git: escrito en Perl, C y varios scripts de shell, diseñado por Linus Torvalds según las necesidades de Linux; con los requisitos de descentralización, rápido, flexible y robusto.

Propietario

- BitKeeper: usado en el desarrollo del Núcleo de Linux
- Code Co-op: sistema de control de versiones P2P
- Sun WorkShop TeamWare: retirado, reemplazado por BitKeeper.
- Plastic SCM: por Codice Software, Inc.

Git

Git es un sistema de control de versiones distribuído (scvd) escrito en C.

Git realiza los commits a tu repositorio local y es posible sincronizar ese repositorio con otros (tal vez remotos) repositorios.

Git te permite clonar los repositorios, por ejemplo, crear una copia exacta de un repositorio incluyendo la historia completa del código fuente.

Los dueños de los repositorios pueden sincronizar los cambios vía push (transfiere los cambios al repositorio remoto) o pull (obtiene los cambios desde un repositorio remoto).

Git

Git soporta el concepto de branching. Si querés desarrollar una nueva característica, podés abrir un branch en tu código fuente y hacer los cambios en este branch sin afectar el principal desarrollo de tu código.

Git puede ser usado desde la línea de comandos o bien desde gráficas.

Git - Terminología

Repositorio:
Un repositorio contiene la historia, las diferentes versiones en el tiempo y todas los distintos branch y etiquetas. En Git, cada copia del repositorio es un repositorio completo. El repositorio te permite obtener revisiones en tu copia actual.

Etiqueta - Tag
Un tag (una etiqueta) apunta a un cierto punto en el tiempo en un branch específico. Con un tag, es posible tener un punto con un tiempo al cual siempre sea posible revertir el código.

Git - Terminología

Branch

Un branch es una línea separada de código con su propia historia. Es posible crear un nuevo branch de a partir de uno existente y cambiar el código independientemente de otros branches. Uno de los branches es el original (generalmente llamado master). El usuario selecciona un branch y trabaja en ese branch seleccionado, el cual es llamado copia actual (working copy). Seleccionar un branch es llamado "obtener un branch" (checkout a branch)

Git - Terminología

Commit (Check-in)
Se hace un commit de los cambios a un repositorio. Esto crea una revisión nueva la cual puede ser obtenida después, por ejemplo si querés ver el código fuente de una versión anterior.

URL
Una URL en Git determina la ubicación de un repositorio.

Revisión
Representa una versión del código fuente. Git identifica revisiones con un id SHA1. Los id son de 160 bits de largo y son representados en hexadecimal.

Git - Terminología

Clon

Cuando hacemos un clon de un repositorio, te bajas una copia del mismo a tu máquina. Se hacen las modificaciones y se hace un push. Cuando se hace el push se modificando el repositorio clonado

Fork

La palabra fork se traduce al castellano, dentro del contexto que nos ocupa, como bifurcación. Cuando hacemos un fork de un repositorio, se hace una copia exacta en crudo del repositorio original que podemos utilizar como un repositorio git cualquiera.

Git - Terminología

Para entender mejor……..

Cuando haces un **fork** de un repositorio, se crea un nuevo repositorio, con una URL diferente (fork). Acto seguido tienes que hacer un **clon** de esa copia sobre la que empiezas a trabajar de forma que cuando haces push, estás modificando TU COPIA (fork). El repositorio original sigue intacto.