“UNIVERSIDAD NACIONAL DE INGENIERIA”

“Facultad de Ingeniería Industrial y de Sistemas”

Alumno: Vidal Flores, Jose Carlos

20130078D

Dirigido: HANCCO CARPIO, RONY JORDAN

Curso: LENGUAJES DE PROGRAMACIÓN ORIENTADOS A OBJETOS (ST-232 V)

“2015-I”

SISTEMA CONTROLADOR DE VERSIONES

Los sistemas de control de versiones son programas que tienen como objetivo controlar los cambios en el desarrollo de cualquier tipo de software, permitiendo conocer el estado actual de un proyecto, los cambios que se le han realizado a cualquiera de sus piezas, las personas que intervinieron en ellos, etc.

VARIANTES DE SISTEMAS DE CONTROL DE VERSIONES

Se tiene principalmente dos tipos de variantes:

Sistemas centralizados: En estos sistemas hay un servidor que mantiene el repositorio y en el que cada programador mantiene en local únicamente aquellos archivos con los que está trabajando en un momento dado. Yo necesito conectarme con el servidor donde está el código para poder trabajar y enviar cambios en el software que se está programando. Ese sistema centralizado es el único lugar donde está todo el código del proyecto de manera completa. Subversión o CVS son sistemas de control de versiones centralizados.

Sistemas distribuidos: En este tipo de sistemas cada uno de los integrantes del equipo mantiene una copia local del repositorio completo. Al disponer de un repositorio local, puedo hacer commit (enviar cambios al sistema de control de versiones) en local, sin necesidad de estar conectado a Internet o cualquier otra red. En cualquier momento y en cualquier sitio donde esté puedo hacer un commit. Es cierto que es local de momento, luego podrás compartirlo con otras personas, pero el hecho de tener un repositorio completo me facilita ser autónomo y poder trabajar en cualquier situación. Git es un sistema de control de versiones distribuido.

GIT

Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.

Git es un sistema distribuido de control de código fuente o SCM (en inglés Source Code Management).

Un SCM es una herramienta que nos resuelve una serie de problemas a todos aquellos que tenemos que trabajar código fuente. “Código fuente” pueden ser muchas cosas:

- Ficheros HTML / CSS / Javascript
- Ficheros PHP
- Ficheros de configuración
- Documentación

Git es multiplataforma, por lo que se puede usarlo y crear repositorios locales en todos los sistemas operativos más comunes, Windows, Linux o Mac. Existen multitud de GUIs (Graphical User Interface o Interfaz de Usuario Gráfica) para trabajar con Git a golpe de ratón, no obstante para el aprendizaje se recomienda usarlo con línea de comandos, de modo que puedas dominar el sistema desde su base, en lugar de estar aprendiendo a usar un programa determinado.

¿Qué nos aporta git?

- ✓ Auditoría del código: saber quién ha tocado qué y cuándo
- ✓ Control sobre cómo ha cambiado nuestro proyecto con el paso del tiempo
- ✓ Volver hacia atrás de una forma rápida
- ✓ Control de versiones a través de etiquetas: versión 1.0, versión 1.0.1, versión 1.1, etc. Sabremos exactamente que había en cada una de ellas y las diferencias entre cualquiera de ellas dos
- ✓ Seguridad: todas las estructuras internas de datos están firmadas con SHA1. No se puede cambiar el código sin que nos enteremos
- ✓ Mejora nuestra capacidad de trabajar en equipo
- ✓ Merging y branching extremadamente eficientes

Características

Entre las características más relevantes se encuentran:

- Fuerte apoyo al desarrollo no lineal, por ende rapidez en la gestión de ramas y mezclado de diferentes versiones. Git incluye herramientas específicas para navegar y visualizar un historial de desarrollo no lineal.
- Gestión distribuida. Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramas adicionales y pueden ser fusionados en la misma manera que se hace con la rama local.
- Los almacenes de información pueden publicarse por HTTP, FTP, rsync o mediante un protocolo nativo, ya sea a través de una conexión TCP/IP simple o a través de cifrado SSH.
- Los repositorios Subversion y svk se pueden usar directamente con git-svn.
- Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución.
- Todas las versiones previas a un cambio determinado, implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio.
- Realmacenamiento periódico en paquetes (ficheros). Esto es relativamente eficiente para escritura de cambios y relativamente ineficiente para lectura si el reempaquetado (con base en diferencias) no ocurre cada cierto tiempo.

Ordenes Básicas

- git fetch:

Descarga los cambios realizados en el repositorio remoto.

- git merge <nombre_rama>:

Impacta en la rama en la que te encuentras parado, los cambios realizados en la rama “nombre_rama”.

- git pull:

Unifica los comandos fetch y merge en un único comando.

- git commit -am "<mensaje>":

Confirma los cambios realizados. El “mensaje” generalmente se usa para asociar al commit una breve descripción de los cambios realizados.

- git push origin <nombre_rama>:

Sube la rama “nombre_rama” al servidor remoto.

- git status:

Muestra el estado actual de la rama, como los cambios que hay sin commitear.

- git add <nombre_archivo>:

Comienza a trackear el archivo “nombre_archivo”.

- git checkout -b <nombre_rama_nueva>:

Crea una rama a partir de la que te encuentres parado con el nombre “nombre_rama_nueva”, y luego salta sobre la rama nueva, por lo que quedas parado en ésta última.

- git checkout -t origin/<nombre_rama>:

Si existe una rama remota de nombre “nombre_rama”, al ejecutar este comando se crea una rama local con el nombre “nombre_rama” para hacer un seguimiento de la rama remota con el mismo nombre.

- git branch:

Lista todas las ramas locales.

- git branch -a:

Lista todas las ramas locales y remotas.

- git branch -d <nombre_rama>:

Elimina la rama local con el nombre “nombre_rama”.

- git push origin :<nombre_rama>:

Elimina la rama remote con el nombre “nombre_rama”.

- git remote prune origin:

Actualiza tu repositorio remoto en caso que algún otro desarrollador haya eliminado alguna rama remota.

- git reset --hard HEAD:

Elimina los cambios realizados que aún no se hayan hecho commit.

- git revert <hash_commit>:

Revierte el commit realizado, identificado por el “hash_commit”.

GITHUB

Github es un servicio para alojamiento de repositorios de software gestionados por el sistema de control de versiones Git. Por tanto, Git es algo más general que nos sirve para controlar el estado de un desarrollo a lo largo del tiempo, mientras que Github es algo más particular: un sitio web que usa Git para ofrecer a la comunidad de desarrolladores repositorios de software. En definitiva, Github es un sitio web pensado para hacer posible el compartir el código de una manera más fácil y al mismo tiempo darle popularidad a la herramienta de control de versiones en sí, que es Git.

Cabe destacar que Github es un proyecto comercial, a diferencia de la herramienta Git que es un proyecto de código abierto. No es el único sitio en Internet que mantiene ese modelo de negocio, pues existen otros sitios populares como Bitbucket que tienen la misma fórmula. No obstante, aunque Github tenga inversores que inyectan capital y esté movido por la rentabilidad económica, en el fondo es una iniciativa que siempre ha perseguido (y conseguido) el objetivo de hacer más popular el software libre. En ese sentido, en Github es gratuito alojar proyectos Open Source, lo que ha posibilitado que el número de proyectos no pare de crecer, y en estos momentos haya varios millones de repositorios y usuarios trabajando con la herramienta.

¿Para qué sirve?

GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto.

Además de eso, puedes contribuir a mejorar el software de los demás. Para poder alcanzar esta meta, GitHub provee de funcionalidades para hacer un fork y solicitar pulls.

Realizar un fork es simplemente clonar un repositorio ajeno (genera una copia en tu cuenta), para eliminar algún bug o modificar cosas de él. Una vez realizadas tus modificaciones puedes enviar un pull al dueño del proyecto. Éste podrá analizar los cambios que has realizado fácilmente, y si considera interesante tu contribución, adjuntarlo con el repositorio original.

¿Qué herramientas proporciona?

En la actualidad, GitHub es mucho más que un servicio de alojamiento de código. Además de éste, se ofrecen varias herramientas útiles para el trabajo en equipo. Entre ellas, cabe destacar:

- ✓ Una wiki para el mantenimiento de las distintas versiones de las páginas.
- ✓ Un sistema de seguimiento de problemas que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.
- ✓ Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.
- ✓ Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.

¿Qué uso le daremos?

En nuestra especialidad “Programación”, fuimos aprendiendo cosas y creando programas de código abierto, fomentando el software libre; es por eso que presentamos esta gran herramienta enfocada al crecimiento de proyectos comunitarios y libres.

En esta página podremos crear una cuenta gratuita y comenzar a subir repositorios de código (o crearlos desde 0), para que con la ayuda de todos ese proyecto mejore; así como también fortalecerlos proyectos de los demás para crecer como grupo.