Tutorial básico de GIT – Introducción

Introducción

Bien este va a ser el primero de un seguido de posts sobre GIT. A través de ellos aprenderemos el uso de Git e instalación.

Este extendido sistema de versiones nos permite trabajar conjuntamente en equipo sobre un mismo programa y guardar revisiones sobre nuestro código, siendo después fácilmente recuperables. Git fue creado por Linus Torvalds (el creador del Kernel de Linux)

En este primer post trataremos:

- ¿Qué es Git?
- Principios de Git
- Beneficios de Git
- Diferencias con otros sistemas de control de versiones
- Repositorios
- Working Tree i Index

¿Qué es Git?

Git es un sistema de control de versiones distribuido (Version

Control System») o herramienta de gestión de código («Code Management tool «). Creado por Lnuis Torval y actualmente utilizado por el equipo de desarrollo del Kernel de GNU\Linux.

Utilizando Git fácilmente puedes revisar el histórico de los códigos fuentes de tu aplicativo y revertir cambios volviendo una versión anterior o comprobar diferencias entre versiones de tus ficheros.

Si la última versión de un archivo se encuentra en un repositorio compartido, Git evitará una sobrescritura no intencionada por lo que mantienen una versión anterior del archivo.

Principios de Git

El código fuente se guarda en un «working directory», donde tienes ficheros «tracked» (ficheros que estan controlados por versiones) y ficheros «untracked» (exluidos del control de versiones, por ejemplo ficheros class generados a partir del código fuente que no son necesarios).

La línea de tiempo de desarrollo se compone de las revisiones («commits» en terminología de Git) del código fuente. Cada commit tiene conocimiento de sus predecesores. Mediante la comparación de un commit con su padre (es decir, la comparación de una revisión a la anterior), uno puede ver los cambios introducidos por el commit hijo.

Un commit contiene los nombres y contenidos de los archivos bajo control de versión, información sobre el autor y el committer (quien realizo el commit y que no tiene por qué ser el autor,), el momento en que se hizo el commit y un mensaje en el se deben indicar cuáles son las razones para los cambios realizados.

Para hacer un commit, primero se debe indicar a Git qué

cambios deben formar parte del nuevo commit mediante un comando add y luego de realizar el commit en sí, seguidamente abrirá un editor en el que podremos escribir un mensaje para indicar la razón por la cual se hicieron dichos cambios.

Beneficios de Git

Git no sólo permite realizar un histórico de tus cambios, permite realizar un seguimiento fácil del desarrollo de otros programadores e integrar esos cambios («merge» en terminología Git).

Aunque nos encontremos en un punto en que nuestro programa ha dejado de funcionar por completo mediante Git podemos saber fácilmente que cambios hemos hecho respecto al commit en el que trabajamos.

Otras ventajas:

- Estudiar el historico de un proyecto para entender no sólo lo que hicieron los desarrolladores, sino también por qué al leer sus mensajes en los commits (es decir, el registro de los cambios).
- Recuperar cualquier revisión anterior (es decir, «regression»), por ejemplo, cuando la versión más reciente contiene bugs que no están presentes en una versión anterior.
- Encontrar fácilmente el commit que introdujo los errores para realizar la regresión.
- Asegurarse de que el código y su histórico nunca se perderán, incluso si nuestro disco duro muere. Cualquier persona con un «clon» del repositorio tiene una copia de toda la historia.
- El trabajo en múltiples funcionalidades o corrección de errores, organizándolas fácilmente y cambiando entre ellas utilizando branches.

El único aspecto negativo de Git es el tiempo de aprendizaje. Una vez se llega a dominar, Git es una gran herramienta para cualquier proyecto de desarrollo y su equipo.

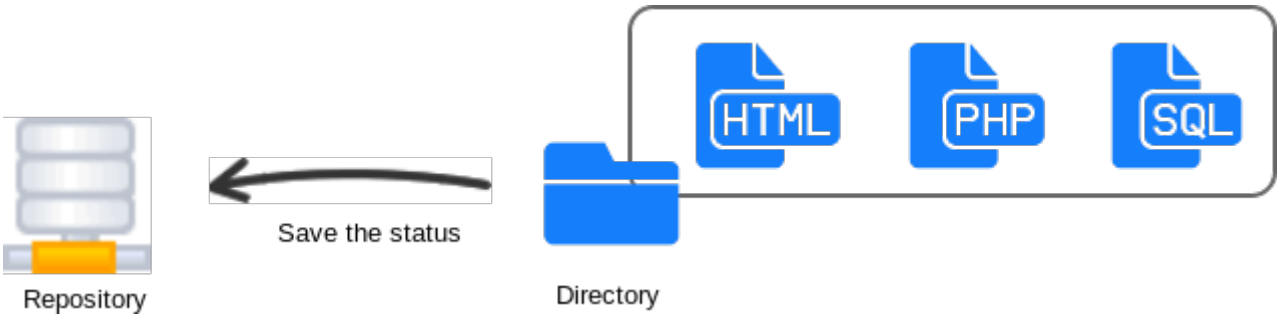
Diferencias con otros sistemas de control de versiones

En comparación con otros sistemas, como CVS o Subversion, nos encontramos con que:

- En Git, cada repositorio es local. Para publicar cambios, es necesario tener un repositorio remoto, también, y hacer un «push» para guardar los cambios allí.
- En Git, ramas son de fáciles de usar y rápidas.
- En Git, imbécil agregar contenido, no los archivos. En otras palabras, cuando el fichero README que ya se realiza un seguimiento, git add README dirá Git que desea que los cambios a los ficheros de ser parte de la próxima confirmación.
- En Git, nunca, nunca intenta integrar los cambios remotos en un estado comprometido. En otras palabras, si usted tiene cambios no confirmados, siempre cometerlos antes de llamar git fetch origen; git merge origin / master.

Repositorios

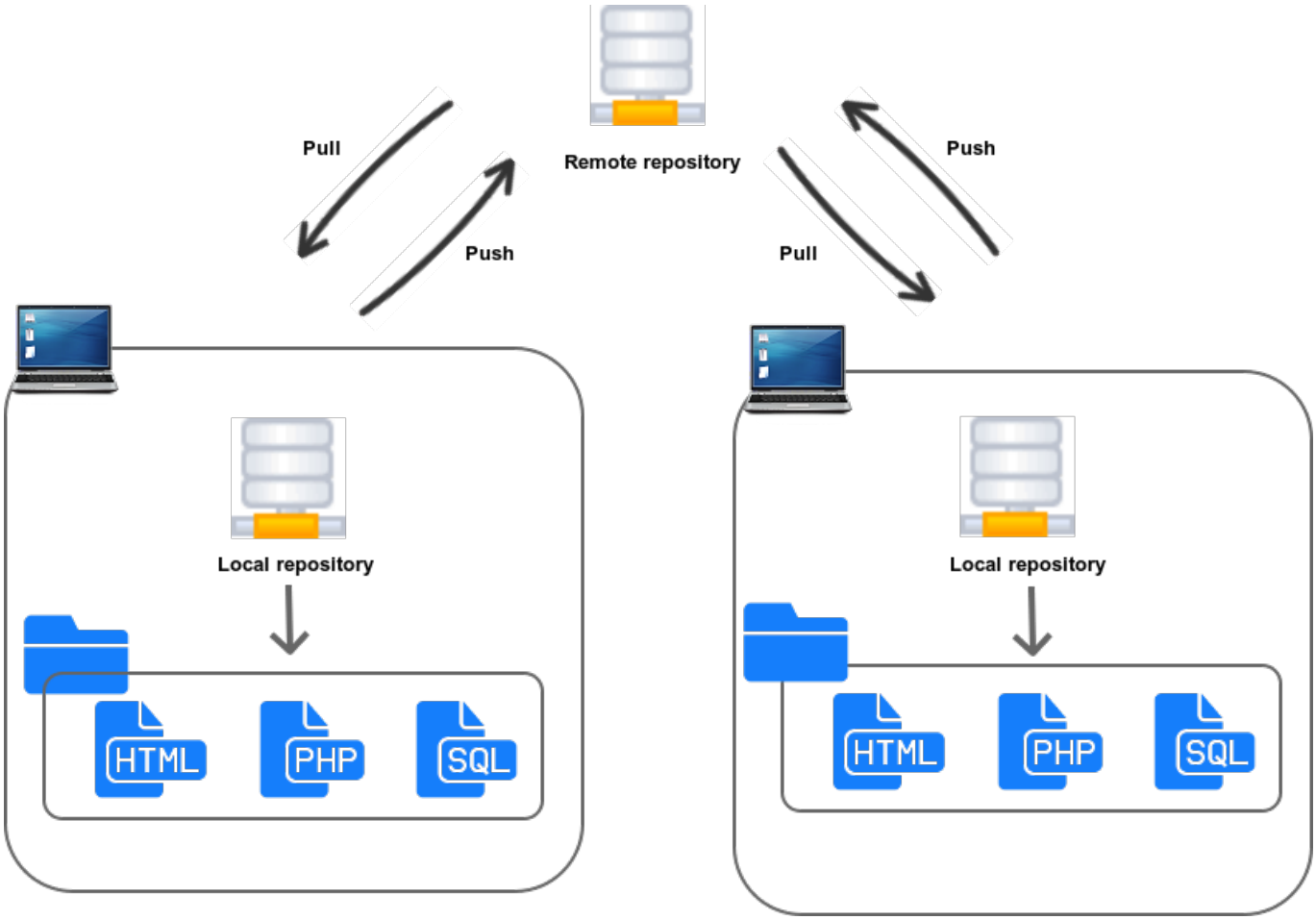
Cuando secrea un repositorio de Git con directorios y ficheros, se puedee guardar sus cambios e el historico y revisar sus estados y versiones.



Existen dos tipos de repositorios:

- Remote repository: Repositorio que reside en un servidor remoto, el cual se comparte con varios miembros del equipo.
- Local repository: Repositorio que reside en un equipo local para uso individual.

Se pueden usar todas las funcionalidades de control de versiones de Git en tu repositorio local (revertir cambios, seguimiento de los cambios, etc.). Sin embargo, cuando se trata de compartir cambios o obtener cambios de miembros de del equipo de desarrollo un repositorio remoto viene muy bien.

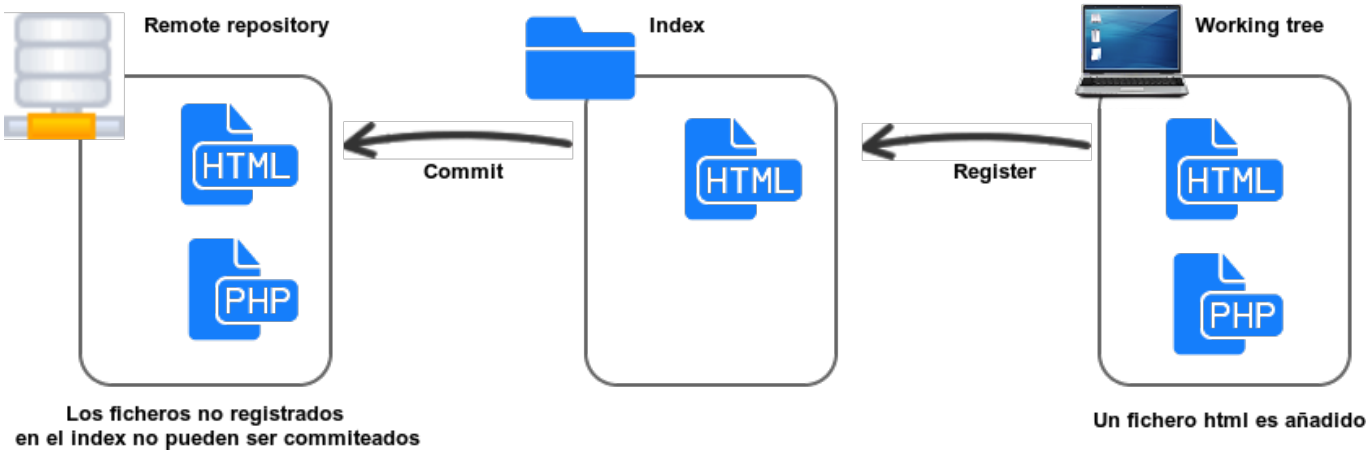


Hay dos formas de crear un repositorio local:

- Puede crear un nuevo repositorio desde cero
- Mediante la clonación obtener un repositorio remoto existente en en nuestro equipo local.

Working Tree y Index

Un «working tree» es un conjunto de archivos con los cuales estas trabajando. Un «index» es un área de ensayo donde se preparan nuevos commits. Actúa como interfaz entre un repositorio y un «working tree».



Los cambios realizados en el «working tree» no serán commiteados directamente al repositorio. Necesitan ser movidos al «index» primero. Todos los cambios que esten en el «index» serán los que realmente se commitearan al repositorio.

Un «index» permite un mejor control sobre qué archivos deben ser incluidos y permite hacer un commit de una parte especifica de un archivo que se encuentra en el «index» al repositorio.

Observaciones

Con está introducción podemos tener un acercamiento a la importancia de los sistemas de versiones y a la versatilidad de Git en concreto. Comenzamos una serie de posts para profundizar en la utilización y explotación de este sistema para mostrar también como puede ayudarnos a trabajar colaborativamente en un proyecto.

Ruben.