GIT Y GITHUB

¿Qué es 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).

¿Qué es un SCM?

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:

1. Ficheros HTML / CSS / Javascript
2. Ficheros PHP
3. Ficheros de configuración
4. Documentación
5. Etc.

Si te das cuenta, todos los ejemplos que he puesto tienen una cosa en común: todo son ficheros de texto plano y todos son ejemplos de ficheros que encontrarás en tu WordPress.

¿Y qué problema resuelve un SCM?

Lo explicare con el siguiente ejemplo:

Acabas de comprarte un nuevo tema o plugin para tu WordPress. Está muy bien, te has gastado S/ 180 (o S/ 0 si es gratis) y con ese plugin tienes resuelto el 80% de las necesidades del proyecto que estás haciendo. Ahora sólo te falta desarrollar el 20% restante. Para hacerlo, tendrás que modificar el código fuente que has adquirido:

1. Seguramente tengas que modificar la hoja de estilos CSS y adaptarla a tu diseño
2. Tendrás que utilizar alguno de los hooks para implementar la funcionalidad que te falta
3. Quizás incluso tengas que añadir tu propio tipo de post, crear una interfaz de administración para él y seguramente hacer tu propio de shortcode
4. Cuando terminas, subes tu código al servidor y publicas tu nuevo proyecto o se lo entregas al cliente. En este momento, el código fuente ya no es el mismo que adquiriste, lo has cambiado.

Es ahora cuando te hago las siguientes preguntas:

1. ¿Sabrías decirme dentro de 6 meses qué es exactamente lo que has cambiado del código del plugin del que partiste?
2. Resulta que tu plugin inicial se ha actualizado y ha pasado de la versión 1.5.1 a la versión 1.6 ¿Serías capaz de incorporar todas las mejoras y novedades a tu versión personalizada (en un tiempo corto, no hacerlo todo otra vez)?
3. Con el paso del tiempo el cliente te pide una actualización, se la subes a su servidor y dos días después te dice que lo dejes como estaba porque no funciona bien ¿Podrías volver atrás de forma rápida? ¿Te acordarás de qué líneas de código has modificado con precisión para depurar rápidamente dónde está el error que has introducido en el sistema?
4. El cliente llama a los tres meses diciendo que no funciona el WordPress ¿serías capaz de determinar de forma rápida si el cliente ha modificado el código fuente del proyecto por su cuenta y se lo ha cargado él? ¿o realmente es un bug que no ha salido en tres meses? La pregunta es importante porque el coste del arreglo no sería el mismo según el tipo de mantenimiento o garantía que hayas acordado con tu cliente
5. Un SCM es la herramienta que nos permite responder positivamente a estas preguntas, y a muchas más.

¿Qué nos aporta Git?

1. Auditoría del código: saber quién ha tocado qué y cuándo.
2. Control sobre cómo ha cambiado nuestro proyecto con el paso del tiempo.
3. Volver hacia atrás de una forma rápida.
4. 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.
5. Seguridad: todas las estructuras internas de datos están firmadas con SHA1. No se puede cambiar el código sin que nos enteremos.
6. Mejora nuestra capacidad de trabajar en equipo.
7. Merging y branching extremadamente eficientes.

Git para Windows, Herramientas y Características

Git para Windows se centra en ofrecer un conjunto ligero, nativa de herramientas que traen el conjunto completo de características de la Git SMC a Windows mientras que proporciona interfaces de usuario adecuadas para los usuarios Git experimentados y novatos por igual.

1. Git BASH

Git para Windows proporciona una emulación BASH utilizada para ejecutar Git desde la línea de comandos. \* NIX usuarios deben sentirse como en casa, como la emulación BASH se comporta igual que el comando "git" en entornos Linux y UNIX.

1. Git GUI

Como los usuarios de Windows comúnmente esperan interfaces gráficas de usuario, Git para Windows también proporciona la interfaz gráfica de usuario de Git, una poderosa alternativa a Git Bash, ofreciendo una versión gráfica de casi todas las funciones de línea de comandos de Git, así como herramientas completas diff visuales.

1. Integración Shell

Simplemente haga clic derecho en una carpeta en el Explorador de Windows para acceder a la BASH o GUI. El plugin Git-Guepardo también proporciona una interfaz de TortoiseSVN similar que muestra las funciones Git directamente en el menú contextual.

Git Configuración

Usted debe tiene para hacer estas cosas sólo una vez en cualquier equipo dado, se quedan entre las actualizaciones, también puede cambiar en cualquier momento mediante la ejecución a través de los comandos de nuevo.

Git viene con una herramienta llamada *git config* que permite obtener y establecer las variables de configuración que controlan todos los aspectos de cómo se ve y funciona Git. Estas variables pueden ser almacenadas en tres lugares diferentes:

1. */ Etc / gitconfig* archivo: Contiene los valores para todos los usuarios en el sistema y todos sus repositorios. Si pasa la opción *--system* a *git config*, lee y escribe desde este archivo concreto.
2. *~ / .gitconfig o ~ / .config / git / config file*: Específico para el usuario. Usted puede hacer Git leer y escribir a este archivo concreto pasando la *--global* opción.
3. *config* archivo en el directorio Git (es decir, .*git / config* ) de cualquier repositorio que está utilizando actualmente: Específico para que solo repositorio.

Cada nivel prevalece sobre los valores en el nivel anterior, por lo que los valores en *.git / config* triunfo aquellos en */ etc / gitconfig*.

En los sistemas Windows, Git busca el *.gitconfig* archivo en el *$ HOME* directorio (C*: \ Users \ $ USER* para la mayoría de la gente). También todavía se ve a */ etc / gitconfig* , aunque es relativo a la raíz MSys, que es donde decide instalar Git en el sistema Windows al ejecutar el instalador.

Su Identidad

La primera cosa que debe hacer al instalar Git es establecer el nombre de usuario y dirección de correo electrónico. Esto es importante porque cada Git cometer utiliza esta información, y está inmutablemente al horno en las confirmaciones de empezar a crear:

*$ git config user.name --global "WilmerJelko"*

*$ git config --global user.email wilague@hotmail.com*

Una vez más, es necesario hacer esto solamente una vez si pasas el --global opción, porque entonces Git siempre utilizará esa información para cualquier cosa que hagas en ese sistema. Si desea reemplazar este con un nombre o dirección de correo electrónico diferente para proyectos específicos, se puede ejecutar el comando sin la --global opción cuando estás en ese proyecto.

Muchas de las herramientas de la GUI le ayudará a hacer esto la primera vez que los ejecute.

Su editor

Ahora que su identidad está configurada, puede configurar el editor de texto predeterminado que se utilizará cuando Git te necesita para escribir un mensaje. Si no se configura, Git utiliza editor por defecto del sistema, que es generalmente Vim. Si desea utilizar un editor de texto diferente, como Emacs, se puede hacer lo siguiente:

$ git config emacs core.editor --global

Comprobación de los ajustes

Si desea comprobar la configuración, puede utilizar el *git config --list* comando para listar todos los ajustes Git puede encontrar en ese punto:

$ git config --list

user.name = Wilmer Jelko

user.email=WilmerJelko@hotmail.com

color.status=auto

color.branch=auto

color.interactive=auto

color.diff=auto

...

Usted puede ver las claves más de una vez, ya que Git lee la misma clave de diferentes archivos ( */ etc / gitconfig y ~ / .gitconfig* , por ejemplo). En este caso, Git utiliza el último valor para cada clave única que ve.

También puede comprobar lo Git piensa el valor de una clave específica está escribiendo git *config <key>* :

$ git config user.name

John Doe

¿Qué es Github?

GitHub es una forma para alojar proyectos utilizando el sistema de control de versiones Git, utiliza el framework Ruby on Rails por GitHub, Inc. (anteriormente conocida como Logical Awesome); Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc.

El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

Alcance

GitHub se utiliza sobre todo para el código.

Además de código fuente, GitHub es compatible con los siguientes formatos y características:

1. Documentación, incluyendo automáticamente prestados- README archivos en una variedad de Markdown formatos de archivo -como (ver archivos README en GitHub ).
2. Seguimiento de problemas (incluyendo peticiones de características).
3. Pequeños sitios web pueden ser organizadas desde los repositorios públicos en GitHub.
4. Anidadas tareas listas dentro de los archivos.
5. Visualización de geo espacial de datos.
6. Diagramas de Gantt.

“Git es un sistema distribuido de control de código fuente o SCM”. Que Git sea distribuido quiere decir que está preparado para poder trabajar en equipos distribuidos (es decir, cada uno en su casa) de forma eficiente. Imagínate que tú estás en España y yo en Rusia ¿Cómo hacemos para coordinarnos? ¿Cómo sé yo qué código has tocado tú y viceversa?.

Este problema Git lo resuelve con herramientas un poco complicadas de configurar si no tienes la experiencia y conocimientos adecuados (Servidores SSH, claves públicas y privadas, etc). Si eres administrador de sistemas y tienes tu propio servidor, no tardarías mucho en hacerlo. En caso contrario, Github te facilita toda la infraestructura para trabajar en equipos distribuidos a través de una interfaz web la mar de cómoda.