GIT | Control de Versiones

Contenido

1. Introducción

Cuando desarrollamos software, el código cambia constantemente, es por ello por lo que vamos a necesitar alguna manera de controlarlo. Git es un sistema de control de versiones distribuido que te ayuda a realizar un seguimiento de los cambios en tu código y a colaborar con otros.

Los proyectos de la vida real generalmente tienen múltiples desarrolladores trabajando en paralelo, así que necesitan un sistema de control de versiones como Git para asegurarse de que no hay conflictos de código entre ellos.

¿Por qué GIT?

Por la sencillez que nos genera para gestionar los cambios. Pensemos en lo siguiente: ¿Cómo gestionarías las versiones de tu código sin esta herramienta?

Pues si te ha tocado en algún momento, de seguro sabes que se suele hacer, lo que comúnmente se denomina “definitivo definitivo 100% definitivo”, la siguiente imagen lo ilustra perfectamente:

Tenemos, además de GIT, otros controladores como CVS o Subversion.

Ventajas

- Facilita al equipo de desarrollo su labor, permitiendo que varios desarrolladores trabajen sobre el mismo proyecto de forma simultánea y controlando que no “se pisen los pies”.
- Proveen un sitio central donde almacenan el código fuente de la aplicación, así como el historial de cambios realizados a lo largo del desarrollo.
- Nos permite, en el caso de necesitarlo, volver a una versión anterior y estable en la etapa del desarrollo.

2. Componentes de un controlador de versiones

- **Repositorio:**

Un repositorio es un proyecto que contiene múltiples archivos que está siendo controlado por GIT y que cuenta con un historial en el que se están registrando los cambios

- **Ramas:**

Son nuevos caminos que toma el proyecto. La rama principal se llama “master” o “main” y es donde está el proyecto que sale a producción. Cada vez que se quiere crear una nueva característica o corregir una existente se crea una rama, de esta forma podemos trabajar en un ambiente aislado.

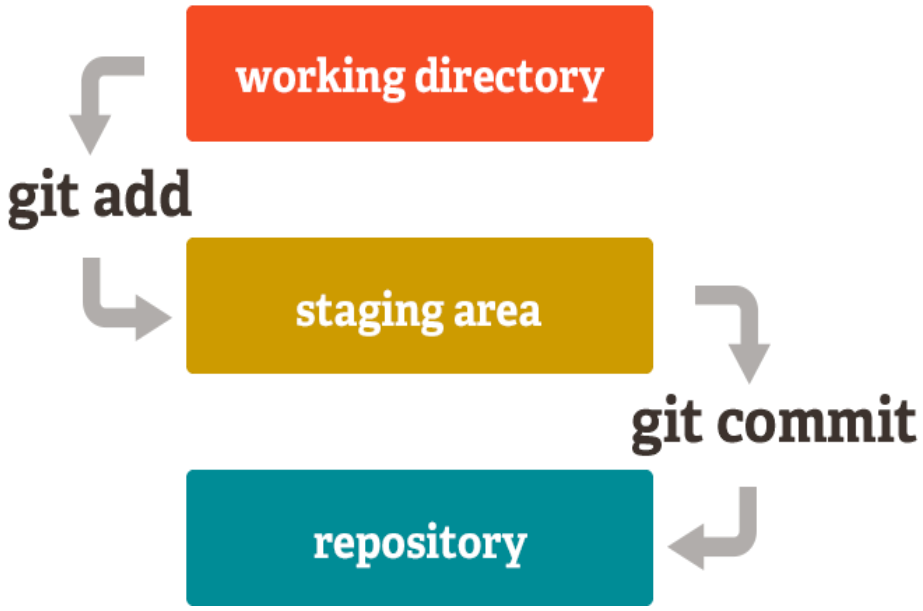
Esta contiene una copia exacta del proyecto, pero separada. De este modo trabajamos en un entorno, que en el caso de que fallara, no tendríamos más que eliminarlo sin ocasionar daños.

- **Etiquetas (Tags):**

Información textual que se añade como complemento a un conjunto de archivos o a un módulo completo para indicar alguna información importante (como la versión).

- **Módulo:**

Es un directorio específico del proyecto, puede identificar una parte o ser el proyecto completo.



3. Áreas de Trabajo de GIT.

En Git, el concepto de "áreas de trabajo" se refiere a las tres principales secciones en las que se puede dividir el ciclo de vida de los cambios en un repositorio:

- **Working area**:

Como su nombre indica, este es el directorio de trabajo donde crearemos los archivos, realizaremos las modificaciones…etc.

Esta área es la que contiene la última versión de los ficheros. Los cambios realizados en el directorio de trabajo no se registran automáticamente en Git, esto quiere decir que no estamos protegidos si perdemos esta información a menos que la guardemos y es por eso, que se le suele considerar a esta área la más volátil.

- **Staging area:**

Este es el área intermedia en la que preparamos los cambios y modificaciones de los archivos del *working area* antes de la validación y el pase al repositorio. Esto te permite seleccionar qué cambios específicos deseas incluir en el próximo commit. Ayuda a organizar tus cambios y a realizar confirmaciones más controladas y significativas.

Para pasar nuestro contenido editado a esta área usaremos el comando "**add**" que va a mover el contenido del Working Area al Staging Area.

- **Local repository:**

Este es el directorio que crea GIT donde almacena el historial completo de cambios. Es el área más segura. Después de agregar los cambios al área de preparación, puedes confirmarlos (**commit**) para guardar una versión de esos cambios en el repositorio. Principalmente lo que veremos en esta área serán todos los puntos de control o commits. Este repositorio también almacena información sobre todas las confirmaciones anteriores, lo que permite realizar un seguimiento del historial del proyecto y revertir a versiones anteriores si es necesario.

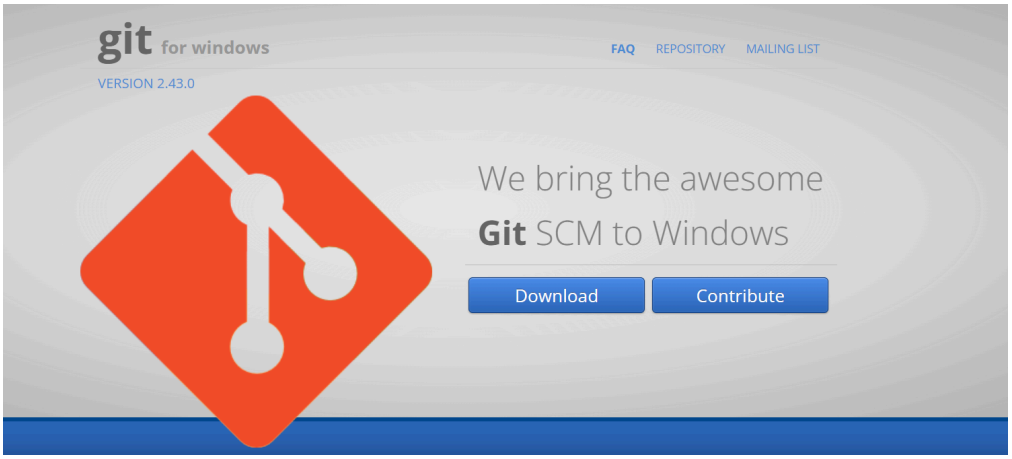
- **GIT Stash:**

Además de estas tres áreas principales de trabajo, tenemos una más, la 'Stash'.

Esta la vamos a utilizar para almacenar modificaciones de manera temporal, es decir, supongamos que estamos trabajando en el proyecto cuando de repente surge algo que tenemos que modificar urgentemente, ¿Qué haríamos con los cambios que estábamos trabajando? pues tendríamos que pasarlos al staging area y de ahí al repositorio, pero esta práctica no siempre nos puede ser útil ya que igual, los cambios con los que estamos trabajando, no son lo suficientemente significativos para crear un commit nuevo pero tampoco para descartarlos por el trabajo que han llevado. Para ello nace este cuarto área, en ella nos vamos a apoyar cuando surjan problemas de este tipo.

En resumen, git stash te permite salvar temporalmente los cambios en tu área de trabajo, limpiarla para poder cambiar de rama o realizar otras operaciones, y luego recuperar esos cambios cuando lo necesites. Es una herramienta útil cuando quieres cambiar de contexto rápidamente sin necesidad de realizar un commit permanente de tus cambios.

4. Instalando Git en Windows



Para instalar Git en Windows, sólo tienes que descargar el instalador y ejecutarlo. Sigue estos sencillos pasos para hacerlo:

1. Descarga el instalador de GIT para Windows en **este enlace.**
2. Una vez que hayas descargado el instalador, haz doble clic sobre el ejecutable para que comience el proceso de instalación y sigue las instrucciones que te aparecerán en pantalla. Al igual que cualquier otro programa, tendrás que dar "Next" (siguiente) en varias ocasiones hasta que aparezca la opción "Finish" (terminar) para completar la instalación.
3. Ahora tienes que abrir el símbolo de sistema y escribir los siguientes comandos en la terminal:
4. `git config --global user.name "Tu nombre"`
5. `git config --global user.email "ejemplo@email.com"`

Recuerda que debes de cambiar '**Tu Nombre**' y '**ejemplo@email.com**' por tu información.

¡Y listo! Ya has instalado GIT en tu equipo.

6. Practicando con Git

Práctica GIT

Revirtiendo Cambios

5. Controlando el contenido de nuestro repositorio

. gitignore

6. Repositorio Remoto

Repositorio Remoto

7. Buenas Prácticas

Tips de Buenas prácticas

8. Git Interactivo.

Learn GIT Branching

9. Extras

Pro-Git

CheatSheet GIT (1)

CheatSheet GIT (2)

Si quieres conocer más acerca de GIT aquí te dejamos un tutorial muy bueno:

TUTORIAL DE GIT de MAKIGAS

https://www.youtube.com/playlist?list=PLTd5ehIj0goMCnj6V5NdzSIHBgrIXckGU