GIT

* ¿Qué es GIT?

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

Git es un sistema de control de versiones.

Un sistema de control de versiones nos va a servir para trabajar en equipo de una manera mucho más simple y optima cuando estamos desarrollando software.

Con Git vamos a poder controlar todos los cambios que se hacen en nuestra aplicación y en nuestro código y vamos a tener control absoluto de todo lo que pasa en el código, pudiendo volver atrás en el tiempo, pudiendo abrir diferentes ramas de desarrollo, etc.

* Para que sirve GIT

Para poder trabajar en equipo de una manera muy sencilla y optimizada, de forma que si tenemos dos o tres personas trabajando en ciertas funcionalidades del proyecto y nosotros podemos estar trabajando en nuestra parte del código. Cuando acabamos de desarrollar nuestro código, utilizamos Git para mezclar los cambios con los otros compañeros. De forma que el código se mezcla de manera perfecta sin generar ningún tipo de fallo y de forma rápida.

* ¿Dónde implementamos GIT?

Cloud, SaaS, Web, Mac (desktop), Windows (desktop)

* Comandos básicos para GIT
* Git clone

Git clone es un comando para descargarte el código fuente existente desde un repositorio remoto (como Github, por ejemplo). En otras palabras, Git clone básicamente realiza una copia idéntica de la última versión de un proyecto en un repositorio y la guarda en tu ordenador.



* Git Branch

Las ramas (branch) son altamente importantes en el mundo de Git. Usando ramas, varios desarrolladores pueden trabajar en paralelo en el mismo proyecto simultáneamente. Podemos usar el comando git branch para crearlas, listarlas y eliminarlas.



* Git checkout

Este es también uno de los comandos más utilizados en Git. Para trabajar en una rama, primero tienes que cambiarte a ella. Usaremos git checkout principalmente para cambiarte de una rama a otra. También lo podemos usar para chequear archivos y commits.

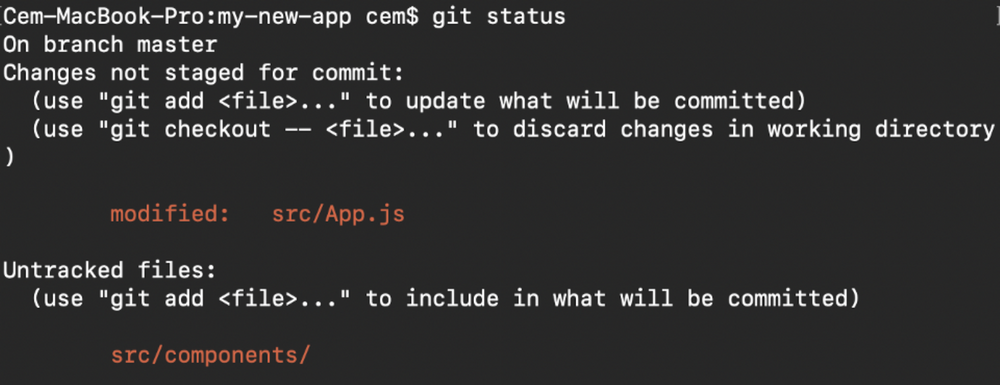


* Git status

El comando de git status nos da toda la información necesaria sobre la rama actual.

Podemos encontrar información como:

* Si la rama actual está actualizada
* Si hay algo para confirmar, enviar o recibir (pull).
* Si hay archivos en preparación (staged), sin preparación(unstaged) o que no están recibiendo seguimiento (untracked)
* Si hay archivos creados, modificados o eliminados



* Git add

Cuando creamos, modificamos o eliminamos un archivo, estos cambios suceden en local y no se incluirán en el siguiente commit (a menos que cambiemos la configuración).

Necesitamos usar el comando git add para incluir los cambios del o de los archivos en tu siguiente commit.

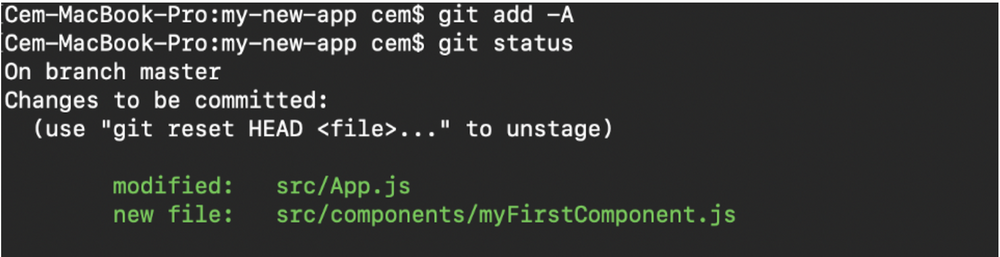
* Añadir un único archivo



* Añadir todo de una vez



* Para añadirlos, necesitas usar el git add



* Repositorio (Directorio de trabajo)

Se trata de la carpeta principal donde se encuentran almacenados los archivos que componen el proyecto. El directorio contiene metadatos gestionados por Git, de manera que el proyecto es configurado como un repositorio local.

* Rama (branch)

Una rama es una línea alterna del tiempo, en la historia de nuestro repositorio. Funciona para crear features, arreglar bugs, experimentar, sin afectar la versión estable, la línea principal, del proyecto. La rama principal por defecto es master.

* Pull Request

Usualmente en proyectos con un equipo de trabajo, cada persona puede trabajar en una rama distinta, pero llegara el momento que dicha rama se tenga que unir a a la rama principal master. Para eso se crea un pull request (abreviado como PR) donde comunicas a tus compañeros de trabajo el código que incluye tu cambio. Usualmente revisan tu código, agregan comentarios y finalmente lo aprueban para darle merge. En el contexto de GIT, merge significa unir dos trabajos, en este caso tu branch con master.