**Apuntes**

GIT es un sistema de control de versiones. Guarda todas las versiones de un mismo fichero. **Guarda sólo los cambios no el fichero de nuevo.** Así como quien ha realizado los cambios, la fecha y otros datos sobre los cambios.

**$ git init** -> Inicia el repositorio en la carpeta

**$ git add ‘archivo’** -> Añade un fichero al repositorio

**$ git add .** -> Añade todos los ficheros que contengan un cambio al repositorio

**$ git rm – cache ‘archivo’** -> Elimina el add del fichero

**$ git commit** -> Sube los archivos añadidos al repositorio

**$ git commit -m ‘Comentario’** -> Hace el commit con un comentario

**$ git status** -> Muestra el estado del repositorio

**$ git show ‘x’** -> Muestra todos los cambios del repositorio o el archivo si se le pone ‘x’. Cuando se hizo, quien lo hizo, lo que se ha cambiado, los comentarios.

**$ git log ‘nombre’** -> Es lo mismo que **show** pero para un fichero, no todo el repositorio

**$ git push** -> Sube los cambios a un repositorio externo

**$ git pull** -> Trae los cambios de un repositorio externo

$ **git --list** -> Muestra la configuración por defecto de GIT

$ **git --list --origin** -> Muestra donde están los archivos de configuración

**$ git config** -> Muestra la configuración por defecto de GITy muestra como configurarla

**$ git config --global user.name o user.email ‘nombre’** -> Cambia la configuración del usuario de GIT ya sea el nombre o email

**Instalar GIT en Windows**

1. Ir a la web de GIT: <https://git-scm.com/downloads>
2. Descargar
3. Abrir e instalarlo (En principio siguiente, siguiente…)
4. En el apartado Path enviroment da a elegir entre usar GIT con la terminal emulada de Linux (Primera opción) que se instala o desde la terminal de Windows (Segunda opción)
5. En el apartado Choosing HTTPS te da a elegir el tipo de certificado que se va a usar (Mejor usar OPENSSL)
6. En el apartado line-ending conversions da a elegir que tipo de salto de línea se quiere elegir en el entorno de desarrollo. (Mejor usar la primera opción)
7. Una vez terminado todo, install y se selecciona la opción lauch para abrir la consola de GIT
8. Cada vez que se quiera abrir la consola de GIT solo hay que buscar GIT BASH

**Instalar GIT en MacOS**

1. Ir a la web de GIT: <https://git-scm.com/downloads>
2. Descargar
3. Se abre el archivo dmg y se abre el archivo pkg
4. Se instala y se mueve el instalador a la basura si lo desea
5. Para saber si se ha instalado git correctamente se escribe git en la consola. Si sale la lista de instrucciones está instalado.

**Instalar GIT en Linux**

1. sudo apt-get install git (Recomendable hacer primero el update y upgrade para actualizar los repositorios de actualización)
2. Ya está instalado. Para comprobarlo escribir git en la consola y comprobar que muestra las instrucciones

**Tipos de archivos y sus diferencias**

**Archivos de texto (.txt):** Es texto normal y sin nada especial. Se ve igual en todas partes sin importar el editor de texto que sea.

**Archivos RTF (.rtf):** Se puede guardar en diferentes tamaños, estilos o colores. Al abrirlo desde un editor de código se pueden ver todos esos estilos del texto en código.

**Archivos de Word (.docx):** Podemos guardar imágenes y texto con diferentes tamaños, estilos… Cuando se abre desde un editor de código se ve realmente como archivos binarios

**Terminal**

La estructura de datos de Windows y Linux es muy diferente.

En Linux la raíz del disco es /

En Windows C:\

El terminal en Windows no es sensitive case

**ls -al ->** Muestra todos los archivos, incluido los ocultos de manera ordenada.

**Touch ->** crea un archivo vacío

**cd ..** -> Cambia al directorio anterior

**cat ‘nombre del archivo’** -> Muestra de manera resumida lo que tiene el archivo dentro

**history ->** Muestra el historial de comandos

**rm ‘nombre del archivo’** -> Comando usado para borrar archivo

**Cómo funciona GIT**

Cuando un archivo se agrega (add) al repositorio, todavía no ha sido subido al repositorio. Está en estado de staging. Una vez que se hace el commit, se sube al repositorio. **staging es la memoria RAM.**

Un archivo añadido no está traqueado. Todavía no tiene un seguimiento.

Cuando se hace un commit a los archivos cambiado se añade a la rama Master. Cada commit es una nueva versión de cambios hacia el repositorio.

Los números raros que aparecen en los commit cuando se usa el comando **git log ‘x’** son el nombre que GIT usa internamente para identificar los commit.

**Ramas**

Las ramas son copias en un punto de la rama Master (principal) que se usan para realizar variaciones en el código. Por ejemplo, para corregir bugs, a esta rama se le llama **hotfix.** O para realizar experimentos en el código, la rama **development.**

Un **Merge** es unir dos ramas para implementar los cambios de una rama en la otra.

Al iniciar un repositorio (**git init**) se crea la carpeta oculta **.git** que contiene los cambios atómicos en la carpeta (significa que cada cambio se guarda de manera independiente).

Con **git status** podemos ver el estado de la carpeta donde está nuestro repositorio en todo momento.

Si hacemos **git rm ‘x’** el fichero se ha dejado fuera del commit pero al hacer **git status** todavía sigue apareciendo listo para el commit. Esto ocurre porque está en memoria RAM. La mejor forma de hacerlo es **git rm –cached ‘x’** y así directamente no aparece en el commit.

**Staged** ->Estado del archivo. Listo para hacer commit pero no hecho todavía.

Cuando se hace un commit hay que configurar quien ha hecho el cambio o de lo contrario lo buscará en el sistema.

Para configurar git se usa **git config**