Unidad 3: Git

# Introducción

## Ventajas de GIT

1. Facilita el trabajo colaborativo
2. Reduce los tiempos de despliegue de un proyecto
3. Permite regresar a versiones anteriores de forma sencilla y rápida
4. Permite generar flujos de trabajo que facilitan el desarrollo y mantenimiento
5. El ecosistema de Git agrega un montón de herramientas a nuestra disposición
6. Las ramas de Git, permiten trabajar con una base de código paralela al proyecto
7. Empieza a trabajar desde otro entorno es tan fácil como “clonar” a un nuevo entorno
8. Sistema de etiquetas, para etiquetar las distintas versiones del proyecto

# Configuración

## Primeros comandos

Saber la versión del Git:

**Git --version**

Muestra la ayuda sobre los comandos:

**Git help**

Muestra la ayuda sobre el comando específico:

**Git help <comando>**

## Saber la rama de nuestra ubicación

Con el comando git Branch



# Cambiar el nombre de la rama “Master a Main”

Para cambiar el nombre de la rama usamos el comando



Esto es una configuración, para dejar en “**main**” todos los repositorios por defecto a partir del momento que ejecutemos



Para mostrar la configuración del git, ejecutamos el comando

**git config --global -e**

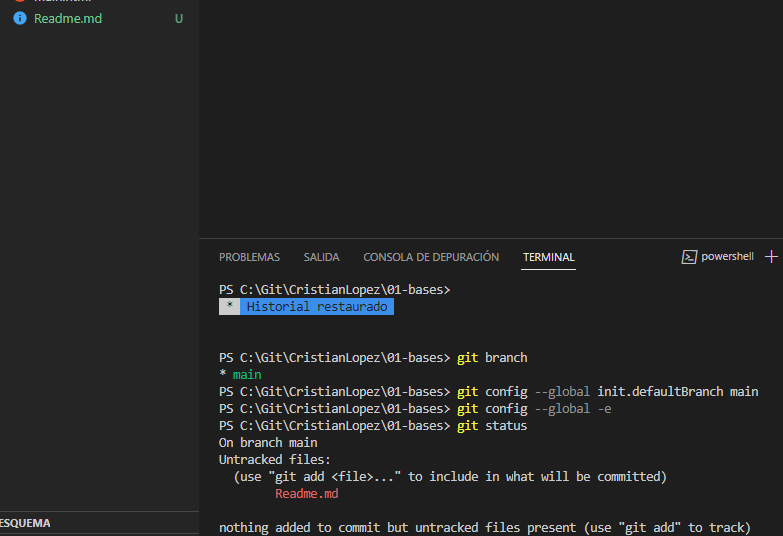
Y mostrará una ventana como a continuación

Texto

Descripción generada automáticamente

# Archivo Readme.md y comando log

Creamos un archivo de información, llamado **Readme.md** y si lo visualizamos en Git, está sin seguimiento.



Añadimos contenido al fichero, como en la siguiente captura

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Y ahora, lo añadimos al seguimiento, con el comando

**git add .\Readme.md**

Texto

Descripción generada automáticamente

Con esto, añadimos un comentario

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Si borramos el fichero, y usamos el comando

**Git checkout -- .**

Se restaurará el fichero, de vuelta como en la captura

Texto

Descripción generada automáticamente

Si modificamos el archivo, con otro contenido, aparecerá que está modificado, como aquí



Y para guardar los datos modificados, ejecutaremos el comando:

**Git commit -am “<Mensaje>”**

Texto

Descripción generada automáticamente

*Y volverá verse, como antes.*

## Ver los commits realizados

Para ver los commits que hemos hecho hasta ahora, podemos usar el siguiente comando:

**Git log**

Y aparecerá esto:



Incluye el autor de quién realizó el cambio, la fecha de modificación. Esta ordenado, del más reciente por la parte superior, y la parte inferior, la última.

# Adds y commits con VSC

Volvemos modificador el archivo de texto

Nos aparecerá un circulo azul en VSC, y lo abrimos



1. Abre el archivo del fichero
2. Descarta los cambios
3. Aplica los cambios que hayamos modificado

Y si luego, lo aplicamos desde VSC correctamente. Lo podremos, ver si se guardó correctamente, con el comando:

**Git log**

## Diferencia de agregar archivos en Git

Ahora vamos abrir el bases-02

Para inicializarlo usamos lo mencionado anteriormente, el **git init.** En este caso, agregamos el index y el main:

Texto

Descripción generada automáticamente

De esta manera podemos agregar archivos, en un mismo comando.

Con el comando git reset, podemos eliminarlos del Git.

Texto

Descripción generada automáticamente

También el la consola, se pueden usar los comodines. En este caso agregamos todos los archivos que acaben el .html:

Texto

Descripción generada automáticamente

Ahora en este caso, añadiremos los archivos que acaben en .js, y se puede usar el \*

Como en este caso:

Texto

Descripción generada automáticamente

## Archivo .gitkeep

Cuando queremos añadir al menos un archivo a una carpeta vacía para que git la tenga en cuenta, en vez de crear cualquier archivo tenga la existencia .

Vamos a crear un nuevo archivo en una nueva carpeta, y el Git lo detecta, en caso contrario, no lo detecta.

Ahora, en la nueva carpeta, creamos un archivo **.gitkeep**, se verá algo así:

Texto

Descripción generada automáticamente

Ahora, lo añadimos dentro del Git.

Y después lo agregamos, en el commit, como en este caso:

Texto

Descripción generada automáticamente

Pantalla de computadora con letras

Descripción generada automáticamentePara comprobar que funciona, borramos el directorio “nueva carpeta”, y para probar que el Git lo restaura, usamos el comando hasta ahora:

Git checkout -- .

Y volverá en la raíz del proyecto:

Si queremos poner todo los ficheros, de un directorio, usamos el nombre de su carpeta y acompañado de la barra **\**. Es similar, como si usamos el **test\test\\***

Texto

Descripción generada automáticamente

Para comprobar sus commits, usamos el comando git log. Como en este captura:

Texto

Descripción generada automáticamente

# Formas de agregar archivos al escenario

## Añadir archivos sueltos

Para añadir archivos a la vez, con un mismo comando. Usando el siguiente comando:

**Git add index.html main.html**

Y se verá de esta manera:



# Creando alias para nuestros comandos

Para mostrar el estado de los archivos del repositorio con una descripción corta podemos usar el siguiente comando:

**Git status--short**