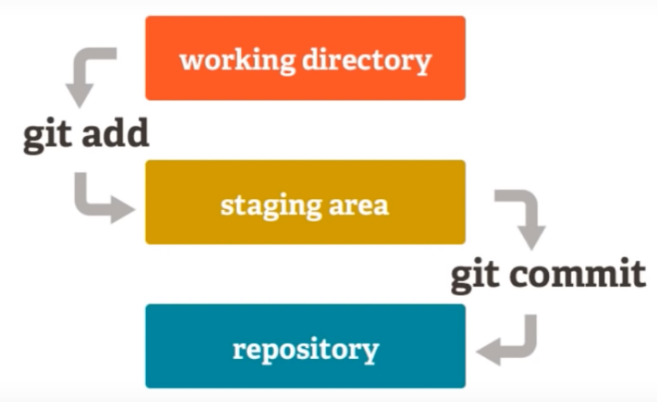
**APUNTES PARA GITHUB**

Estados de un fichero en Git



Comandos básicos

* **git init**: se utiliza para indicar que se va a empezar a usar git en un proyecto nuevo.
* **git add <file>** : es para pasar los archivos desde el working directory al staging area.
* **git status**: indica el estado en el que están los archivos (working directory, staging area, repository).
* **git commit** : es para pasar los archivos desde el staging area hasta el repositorio. Es para crear un primer snapshot de la versión del código.
* **git push** : se utiliza para subir los archivos a un repositorio remoto para que pueda ser accedido por los desarrolladores que tienen permiso para modificar el código.
* **git pull** : si estamos trabajando con otros desarrolladores, este comando trae los cambios que hayan realizado los demás desarrolladores en el código.
* **git clone** : hace una copia desde el servidor central donde está el código al computador local para trabajar.

Pasos para empezar a usar git en un proyecto

1. Abrimos un terminal git bash y nos dirigimos usando el comando **cd** a la carpeta donde se encuentra el proyecto. Un atajo para esto es hacer click derecho en la carpeta del proyecto y nos aparecerá la opción de abrir un terminal git bash.
2. Una vez dentro de la carpeta, ejecutamos **git init** y se nos creará una carpeta llamada .git dentro de nuestro proyecto. Esta carpeta se encarga de gestionar las versiones, pero nosotros no tenemos que tocar nada dentro de ella. Al hacer todo esto, convertimos la carpeta de nuestro proyecto en el working directory.
3. Si tenemos un proyecto con archivos ya creados, estos archivos se verán dentro de git como untracked porque no los hemos agregado al staging area. Podemos ver el estado de los archivos con **git status**.
4. Para agregar archivos al staging area, usamos **git add <File>**. Podemos agregar todos los archivos sin tener que nombrarlos uno a uno usando el comando seguido de un punto: **git add .** .
5. Una vez tenemos todos los archivos en el staging area, podemos hacer nuestro primer snapshot del proyecto (nuestra primera versión), ejecutando **git add**.

Es posible que se produzca un error si es la primera vez que usamos git. Git nos pedirá que nos identifiquemos con nuestro correo y nombre de usuario. Debemos tener en cuenta que en un mismo proyecto de git puede haber varios desarrolladores y es importante saber quién ha realizado cada cambio.

Para indicar estos datos, ejecutamos los comandos:

* 1. **git config --global user.email “**[**your@example.com**](mailto:your@example.com)**”**
  2. **git config --global user.name “Your name”**

1. Cuando ejecutamos **git commit** nos aparecerá una consola nueva en la que podremos escribir. Si al abrirlo no nos permite escribir, tendremos que pulsar la tecla I. Una vez escrito lo que queramos añadir, podemos guardar los cambios pulsando ESC y escribiendo :wq. Cuando ejecutamos el commit nos aparecerá un mensaje indicando:

* Los archivos que han sido modificador.
* Las inserciones que se han hecho (número de líneas de código añadidas).
* Los borrados que se han hecho (número de líneas de código borradas).

Podemos ejecutar el git commit con la opción -m para escribir el mensaje y no tener que entrar en la consola. Por ejemplo: **git commit -m “Este es el mensaje”.**

1. Si escribimos **git log** podemos obtener un hash que contiene el registro de este primer snapshot que hemos hecho junto con datos del autor, fecha, hora y lo que escribimos en el fichero a la hora de hacer commit.
2. Si hacemos ahora **git status**, nos dirá que no hay nada para confirmar.
3. Si hacemos algún cambio en uno de los ficheros y ejecutamos **git status**, nos indicará que hay archivos que han sido modificados. Podemos volver a la versión anterior haciendo **git checkout -- <File>**.
4. Si hacemos algún cambio, podemos ver qué había antes en comparación con lo que hay ahora haciendo **git diff <File>**. En rojo nos pondrá aquello que se ha eliminado y en verde lo que se encuentra ahora mismo en el archivo.

Indicar los archivos a ignorar

En git podemos indicar qué archivos y carpetas no queremos que se tengan en cuenta a la hora de hacer los add y comit. Para ello, tenemos que crear un fichero con el nombre .gitignore y dentro de este fichero, escribir los nombres de las carpetas o archivos que queremos que se ignoren.

Crear ramas (branch) del proyecto

El proyecto principal se denomina master. Podemos crear versiones alternativas al proyecto principal para realizar cambios y luego, si queremos, aplicar estos cambios al proyecto principal (master) o descartarlos.

Para hacer esto, tenemos que seguir los siguientes pasos:

1. Para crear una rama, tenemos que ejecutar **git branch nombre\_branch**.
2. Para cambiar nuestro working directory al del branch, tenemos que ejecutar **git** **checkout nombre\_branch**.

Para combinar una rama con la rama master:

1. Nos tenemos que colocar en la rama master usando **git checkout master**.
2. Ejecutamos **git merge nombre\_rama**.

Para eliminar una rama se ejecuta **git branch -d nombre\_branch**. Si la rama a borrar tiene cambios que no han sido confirmados en la rama master, nos saldrá un error. Si queremos eliminarla de todas formas, el comando será **git branch -D nombre\_branch**

Asociar nuestro proyecto con Github

Podemos subir nuestro proyecto a Github para compartirlo con otros usuarios y tener una interfaz gráfica de nuestro proyecto. Para ello, tenemos que crear una cuenta en Github, agregar un nuevo repositorio y quedarnos con la URL del repositorio para usarla en los siguientes comandos:

* **git remote add origin URL**
* **git push -u origin master**

Descargar o clonar el proyecto completo desde Github

Si presionamos en la opción Clone or download, podemos descargar un ZIP con todo el proyecto o podemos copiar la dirección URL y ejecutamos **git clone URL**. Este comando nos va a descargar el proyecto en la carpeta en la que se encuentre la consola de git bash.

Descartar cambios

* Para un solo archivo: git checkout -- <FILE>
* Para todos los archivos y volver a la versión del último commit: git reset --hard