GIT con la terminal

Empezar con un proyecto desde cero

1. Crear una carpeta nueva y acceder a ella dersde la terminal.
2. Inicializamos un nuevo repositorio con `git init`
3. Por defecto **siempre se creará una rama principal** llamada main o master
4. Podemos usar el comando `git status` cuando queramos para saber información del estado actual del repositorio.
5. Creamos o modificamos los archivos que hagan falta.
6. Para prepararlos para hacer esa "foto" a la que llamamos commit utilizamos el comando `git add`
    1. Podemos añadir uno o varios archivos llamándolos uno a uno. `git add <nombre-archivo> <nombre-archivo>`
    2. Podemos añadir TODOS los archivos a la vez -> `git add .`
7. Ya podemos hacer un commit de los archivos que hemos "preparado" usando `git commit`
8. Podemos hacer todo esto las veces que quieras. `git add .` + `git commit -m "mensaje del commit"`

Vamos a añadir nuestros datos de github para evitar problemas

1. Para añadir tus datos de github editamos un archivo de configuración que guarda muchas cosas.
    1. Añadir el nombre de usuario en --global para que todos en todos los lugares de tu ordenador donde uses Git sepan que eres tu.

        `git config --global user.name "tu nombre de usuario"`
    2. Añadir el email en --global para que todos en todos los lugares de tu ordenador donde uses Git sepan cual es tu email de usuario.

        `git config --global user.email "email@email.com"`
    3. Para decirle a git que siempre que tenga que editar algo utilice VSCode usamos este comando:
        `git config --global core.editor "code --wait"`

Vincular tu repositorio local a uno nuevo de Github

1. Crea el nuevo repositorio en Github. Para que te muestre toda la info de los comandos que tienes que poner para vincular el repositorio a tu local, no le añadas el archivo README.md desde github.
2. Si tu rama se llama `master` por defecto le cambiamos el nombre a ==main= con el comando `git branch -M main`
3. Le decimos a nuestro repositorio local que vamos a añadir un repositorio remoto en el cual subiremos nuestros cambios.
    1. El comando para añadir un remoto es `git remote add <alias del remoto> <url del remoto>`
4. Tenemos que subir el repositorio local a github. Para eso utilizamos el comando `git push`
    1. `git push` necesita que le digamos qué rama subir y a qué remoto -> `git push <alias del remoto> <nombre de la rama>`
    2. La primera vez es recomendable (o github por lo menos te dice que lo hagas) añadirle al comando el argumentp `-u` para hacer que a partir de ahora, solo tengas que escribir `git push` para "empujar" los archivos a remoto. Por ejemplo: `git push -u origin main`
5. Podemos hacer todo esto las veces que quieras. `git add .` + `git commit -m "mensaje del commit"` + `git push`

Clonar un repositorio y obtener los archivos nuevos

Si en algún momento tienes que irte a otra ciudad, otra casa, otro ordenador y necesitas coger tu repositorio de github para seguir trabajando:

1. Clonas el repositorio en ese nuevo ordenador. `git clone <url del repositorio>`
2. Ahora ya tienes esa carpeta **IDÉNTICA** a la carpeta de Github.
3. Ya puedes trabajar como de costumbre. `git add .` + `git commit -m "mensaje del commit"`
4. Cuando ya quieres subir a Github lo nuevo haces otro `git push`
5. Cuando vuelves a tu primer ordenador, **NECESITAS LOS NUEVOS CAMBIOS** y para eso tienes que "estirarlos" usando el comando `git pull <alias del remoto> <nombre de la rama>` pero com ya hemos establecido antes con el comando `git push -u origin main` que la rama main se vincule automaticamente con la del remoto, podemos hacer simplemente un `git pull`
6. A partir de ahora todo el rato harás lo mismo:

1. Trabajar en un ordenador `add` + `commit` + `push`
2. En el otro ordenador te bajas lo nuevo `git pull`
3. Trabajas y `add` + `commit` + `push`
4. Y así constantemente.

7. Si te olvidaras de hacer un `git pull`, al hacer un `push` te saltará una alerta diciendo que no puedes hacer `push` porque hay cambios en github más modernos que los tuyos, que necesitas hacer un `git pull`.

    1. Si cuando haces el `git pull` los cambios que vienen de github no coinciden con los mismos que has hecho tu (no son las mismas líneas), todo se actualizará automáticamente sin problema y ya podrás hacer tu `git push`
    2. Si cuando haces el `git pull` sí que hay CONFLICTOS, te traerá todo lo nuevo de Github, pero los archivos con conflictos tendrás que modificarlo y decidir con qué código te quedas. Una vez lo hayas decidido vuelves a lo mismo `git add .` + `git commit` + `git push`