2️⃣ Git

▼ ¿Qué es un sistema de control de versiones y cómo afecta directamente a nuestra felicidad?

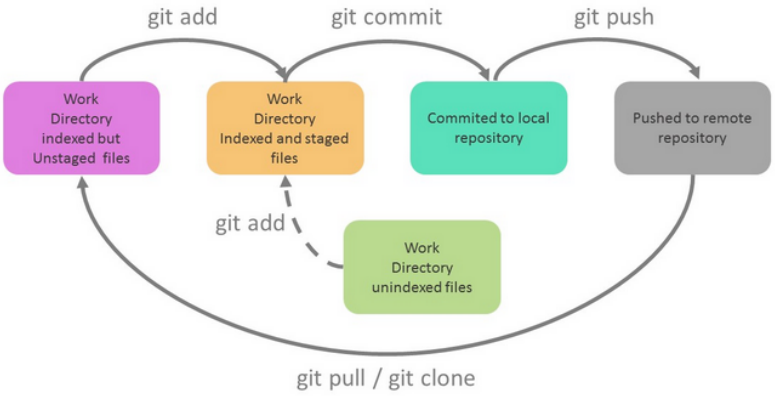
- Proporciona copias de seguridad automáticas de los ficheros
- Permite volver a un estado anterior de nuestros ficheros
- Ayuda a trabajar de una forma más organizada
- Permite que varias personas trabajen en los mismos ficheros (Fijando regiones críticas)
- Permiten trabajar en varias funcionalidades en paralelo por separado (Ramas)

▼ Conceptos generales

- Repositorios: contaremos con repositorio local (una carpeta en nuestro ordenador) y un repositorio remoto (alojada en plataformas como GitHub o GitLab)
- Commits: puntos de guardado (o hitos) que se van realizando en los que fijamos los cambios realizados en el código. Podremos volver a ellos para recuperar el estado del código en un determinado momento.
- Remote: conexión que establecemos entre el repositorio local y remoto para la sincronización
- Sincronización (Pull y Push): son los procesos para sincronizar el estado de nuestros sistemas locales con el repositorio remoto, tanto para subir los cambios locales como para descargar las últimas actualizaciones
  - Pull → Servidor a local
  - Push → Local a servidor
- Branches o ramas: son herramientas proporcionadas por los sistemas de control de versiones para poder trabajar paralelamente en distintas versiones del código sin interferir entre si. Suelen utilizarse remas de largo recorrido

para las principales versiones de la aplicación (master para la aplicación en producción, staging para aplicación en pruebas…) y ramas puntuales para distintos desarrolladores que participen en el proyecto.

▼ Control de versiones para trabajo individual



`git add .` → Añadir todos los archivos de la ruta actual a la fase de stage

`git commit -m "mensaje"` → Definir un commit o hito

`git push "remote" "rama"` → Sincronizar los cambios con el repositorio remoto

`git pull` → Sincronizar el repositorio local con el remoto

`git clone "https_url"` → Clonar repositorio en la ruta actual

`git branch -m "nombre" "nuevo_nombre"` → Cambiar el nombre de una rama

`git init` → Inicializar proyecto de git en la ruta actual

`git remote add "remote" "https_url"` → Añadir remote

`git remote remove "nombre_remote"` → Eliminar remote

`git remote -v` → Muestra los remotes con la dirección a la que apuntan

▼ Guía para trabajar colaborativamente en Git con ramas

Colaboración entre administrador y alumno

1. Administrador
    a. Crea un proyecto

b. Clona el proyecto en local → `git clone "url"`
c. Añade contenido
d. Añade los cambios → `git add .`
e. Hace commit de los cambios → `git commit -m "mensaje"`
f. Sube los cambios al repositorio remoto → `git push origin master`

2. Usuario
   a. Hace fork al repositorio remoto del administrador
   b. Elige el destino del fork
   c. Clona su proyecto en local → `git clone "url"`
3. Administrador
   a. Añade contenido
   b. Añade los cambios → `git add .`
   c. Hace commit de los cambios → `git commit -m "mensaje"`
   d. Sube los cambios al repositorio remoto → `git push origin master`
4. Usuario
   a. Añade el remote del repositorio remoto del administrador → `git remote add "nombre" "url"`
   b. Comprueba el remote añadido → `git remote -v`
   c. Sincroniza el repositorio local con el repositorio remoto del administrador → `git pull "nombre_remote" master`
   d. Crea la rama cambios → `git branch cambios`
   e. Cambia de rama a la de cambios → `git checkout cambios`
   f. Añade contenido
   g. Añade los cambios → `git add .`
   h. Hace commit de los cambios → `git commit -m "mensaje"`
   i. Sube los cambios al repositorio remoto → `git push origin cambios`
5. Administrador
   a. Crea una rama en el repositorio remoto para los cambios del usuario

6. Usuario
    a. Crea una merge request desde la rama cambios del repositorio remoto
    b. Cambia la rama raíz y la rama de destino (rama creada por el administrador) de la solicitud de fusión (merge request)
    c. Cambia el titulo de la solicitud
    d. Envía la solicitud de fusión
7. Administrador
    a. Acepta la merge request del usuario
    b. Crea una rama en el repositorio local con el mismo nombre que la rama que se creo en el repositorio remoto → `git branch "nombre_rama"`
    c. Cambia de rama a la recién creada → `git ckeckout "nombre_rama"`
    d. Sincroniza los cambios del repositorio remoto con el repositorio local → `git pull origin "nombre_rama"`
    e. Cambia de rama a la master → `git checkout master`
    f. Fusiona la rama master con la de cambios → `git merge "nombre_rama"`
    g. Sube los cambios al repositorio remoto → `git push origin master`
8. Usuario
    a. Cambia de rama a la master → `git checkout master`
    b. Sincroniza el repositorio local con el repositorio remoto del administrador → `git pull "nombre_remote" master`
    c. Sube los cambios al repositorio remoto → `git push origin master`