CONTROL DE VERSIONES

DAW RETO 1

¿QUÉ ES UN CONTROL DE VERSIONES?

- Antes de definir → Pensamos en un proyecto enorme en el que trabajan en equipo muchas personas.
- Es la gestión de los cambios que se realizan sobre un producto o configuración del mismo.
- Una versión (revisión o edición) es el estado en el que se encuentra un producto en un momento concreto.
- Como puede haber muchas → Necesito hacer un registro

¿POR QUÉ LO NECESITAMOS?

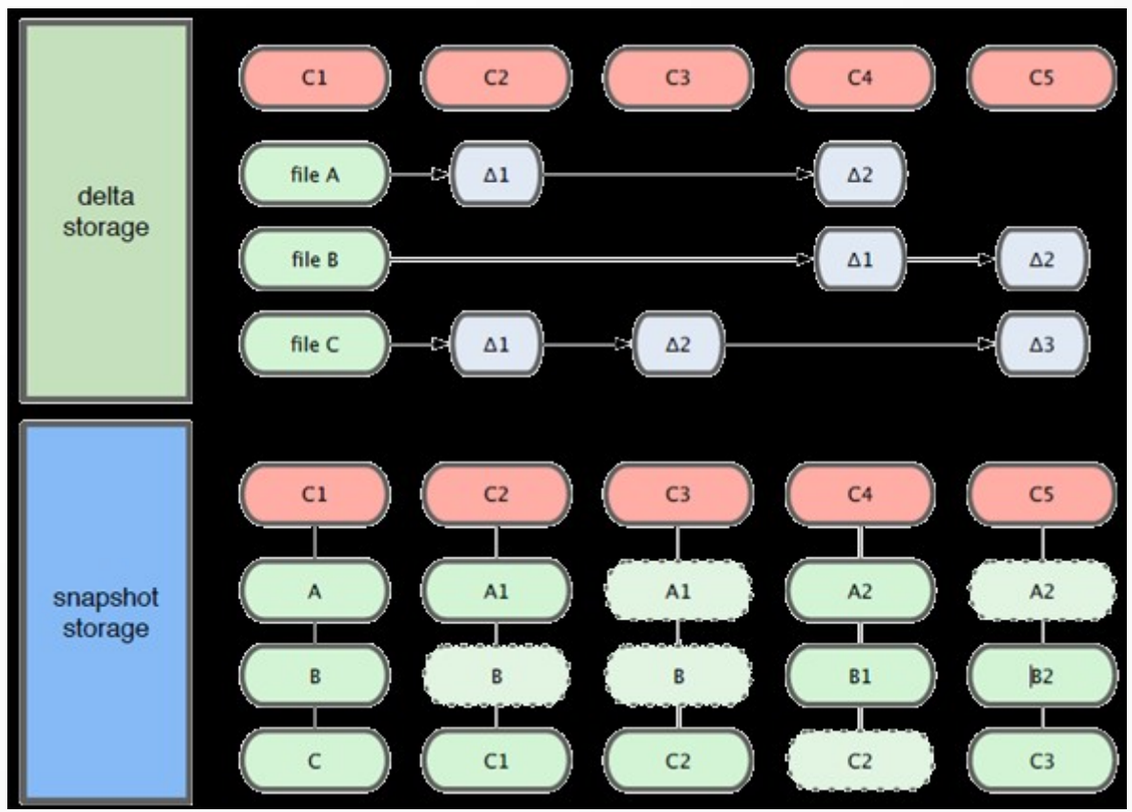
- Proporcionar copias de seguridad automáticas de los ficheros.
- Permite volver a un estado anterior.
- Ayuda a trabajar más organizados.
- Permite trabajar en local.
- Permite que trabajen varias personas a la vez.
- Permite trabajar en paralelo (ramas).

GIT Y GITHUB

- GIT → CONTROL DE VERSIONES
- GITHUB → NOS PERMITE ALMACENAR GIT

SNAPSHOTS

- SNAPSHOTS (INSTANTANEAS)
  - En un punto x ¿Cómo están los ficheros?
  - Puedo ir atrás o adelante.
  - Los ficheros son individuales a las versiones:
    - Si tengo V1 y V2 con 3 ficheros y solo cambio 1. Solo guardo ese archivo en la V2 y las referencias a la V1.



INSTALAR GIT

- UBUNTU → sudo apt-get install git-core
- MAC → home brew → brew install git
- WIN → Descargar e instalar git
  - Cmder → Permite utilizar comandos linux a la vez que integra git
  - EXISTEN ENTORNOS GRÁFICOS: EJ: Gitkraken
  - Recomendación: **aprender por comandos.**

VOCABULARIO INICIAL

- **HEAD:** Commit en el que está posicionado el repositiorio en el momento actual.
- **COMMIT:** Guarda una instantánea (snapshot) de los cambios añadidos al index.
- **INDEX:** Zona intermedia entre el directorio local y el HEAD. Aquí se almacenan los cambios antes de hacer commit.
- **ORIGIN:** Nombre que recibe por defecto el repositorio remoto sobre el que trabajamos.
- **MASTER:** Nombre que recibe por defecto la rama principal del proyecto.

ESTADOS DE GIT

- GIT DIRECTORY: Directorio local .git donde se almacenan los cambios.
  - **Git clone** → clonar proyectos
  - **Git init** → iniciar repositorios
- WORKING DIRECTORY: Copia más reciente almacenada en nuestro git.
- STAGGING AREA: Archivo que contiene todo lo que se ha hecho.

HOSTING GIT

- Existen varios lugares donde almacenar en la nube proyectos de git, de tal manera que pueda favorecer el trabajo colaborativo y la distribución de los proyectos.
  - **GITHUB** → Gratuíto en públicas y código abierto.
  - **BIT BUCKET** → Gratuíto en público y privado con límite.
  - **GITLAB** → Autogestión.

COMANDOS DE GIT

- CHULETA DE COMANDOS DE GIT
- Como se puede comprobar, hay multitud de comandos para utilizar en GIT, a continuación vamos a comentar los más básicos.

1-INICIAR REPOSITORIO GIT

- EN LOCAL:
  - Creamos una carpeta que sera nuestro working directory.
  - Dentro de la carpeta utilizamos → ***git init***
- DESDE REMOTO:
  - Creamos un repositiorio nuevo en github.
  - En la carpeta donde queremos que se cree la carpeta del proyecto utilizamos → **git clone <url>**
  - ***Problema: Nos pedirá credenciales**

2-CREDENCIALES

- EN LOCAL:
  - Indicaremos nuestro nombre de usuario:
    - **git config --global user.name “Nombre”**
  - Indicaremos nuestro correo electrónico:
    - **git config --global user.email “Mail”**
  - *La primera vez, o si no hemos iniciado sesión en github, nos pedirá el usuario y contraseña de github.*
- Existen multitud de configuraciones desde cambiar los colores hasta indicar que modificaciones se pueden permitir o no.
  - *MORE INFO*: https://git-scm.com/docs/git-config

3-AÑADIR Y COMMITEAR

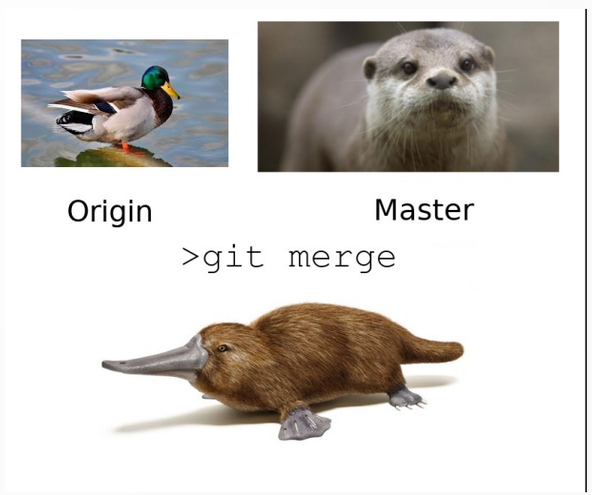
- ADD: Las modificaciones tenemos que añadirlas al index antes de guardarlas en nuestro control de versiones.
  - **git add <fichero>**
- COMMIT: Agregar los ficheros que hemos añadido previamente a nuestro control de versiones.
  - **git commit -m “Mensaje que indique lo que hemos hecho”**
- STATUS: Nos indica si hay algo que hemos añadido pero no commiteado.
  - **git status**

4- PULL Y PUSH

- PULL (Tirar): Actualizamos nuestro directorio con los ultimos cambios del remoto.
  - **git pull**
- PUSH (Empujar): Añadimos los cambios que hemos hecho en local al remoto (El commit solamente deja los cambios en local).
  - **git push origin <rama que queremos subir>**
- ORIGIN: La url del lugar remoto donde está el proyecto.
  - En caso de que iniciemos el repositorio desde local, es decir, en caso de que no lo hayamos clonado → **git remote add origin <URL>**

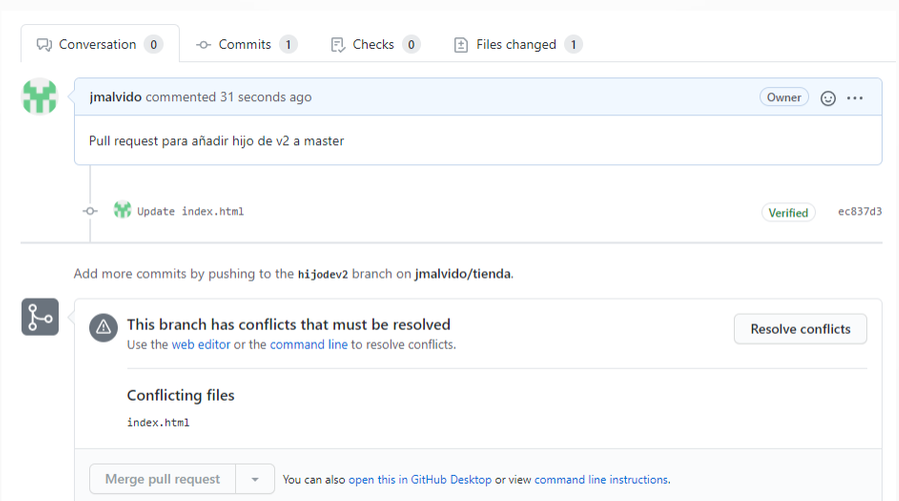
5- BRANCHES

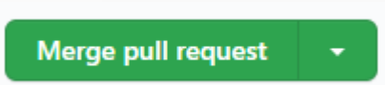
- BRANCH = RAMA
- Creo branches para realizar cambios en paralelo sobre un código.
- CREAR: **git branch <nombre>**
- MOSTRAR: **git branch [-a]**
- CAMBIAR: **git checkout <rama>**
- UNIR (MERGE): **git merge <rama>**
  - *IMPORTANTE: Para unir hay que situarse en la rama destino*



5- BRANCHES

- PULL REQUEST
- Muchas veces, antes de hacer un merge entre ramas, querremos compararlas, o que otra persona del equipo las compare.
- Para ello generemoas una pull request.





6- VOLVER ATRÁS

- Volvemos atrás moviéndonos entre los commits realizados.
- CHECKOUT: Crear una rama auxiliar en un commit anterior.
  - **git checkout <idRama>**
  - Para saber el id → **git log**
- RESET: Borra todas las ramas hasta la que queremos llegar:
  - **git reset --hard HEAD~1** → Resetea 1 commit

TRABAJO 1

VAMOS A REALIZAR EL TRABAJO 1 DE MOODLE