Control de versiones

¿En qué consiste el control de versiones? ¿Cuál es su utilidad?

¿Qué es un control de versiones?

- El control de versiones es la gestión de todos los cambios que se producen en un producto o en una configuración del mismo por parte del equipo de desarrollo.
- La versión o edición del producto es el estado en que se encuentra el mismo en un momento dado de su desarrollo.
- El control de versiones se puede hacer de dos maneras:
  - Manualmente. Opción muy poco recomendable, pues es extremadamente proclive al error humano.
  - Herramientas de software. Opción a elegir si queremos hacer un control de versiones correcto y controlado.

¿Qué es un software de control de versiones?

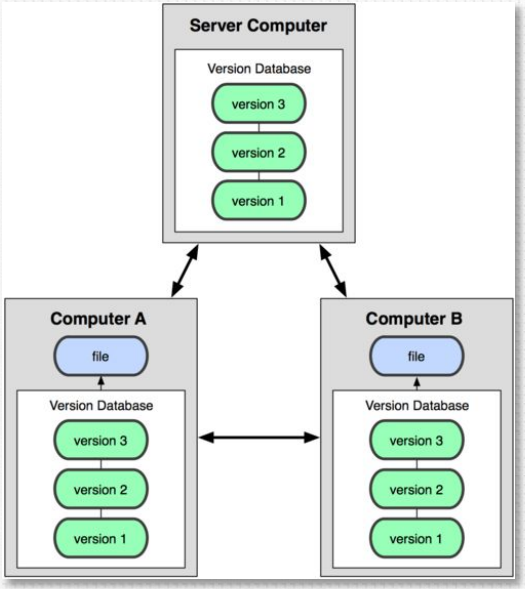
- El software de control de versiones es un programa, que en algunos sistemas como **MacOS** de **Apple** viene preinstalado (**Git**) que nos permite hacer un control de versiones de un producto de software.
- Este tipo de software es conocido también por sus siglas en inglés **VCS** (**V**ersion **C**ontrol **S**ystem)
- Existe una amplia variedad de software de control de versiones, en modalidades de pago, código abierto y código propietario, siendo **Git** uno de los más extendidos y utilizados, por su robustez y por ser de código abierto.
- **Git** viene preinstalado en los sistemas **Apple**, pero en caso de necesitar instalarlo, podemos ir a las fuentes del software y bajar la versión necesaria:
  - **Página oficial de Git**

¿Cómo funciona Git?

- Los archivos del código inicial de un proyecto se alojan en un servidor.
- Este conjunto de archivos alojados en el servidor se denomina **repositorio**.
- Ese servidor puede ser gratuito, de pago o una mezcla de ambas modalidades:
  - **GitHub**: Plataforma en la que los proyectos se alojan de forma pública y gratuita. Si queremos versión privada es necesario pagar una cuota.
  - **GitLab**: Otra plataforma de alojamiento de repositorios, que si bien tiene opción gratuita, es restrictiva y conviene pagar la cuota, pues con ello ofrece muchas más posibilidades.
  - **BitBucket**: Plataforma de alojamiento de repositorios también de pago, aunque ofrece una opción gratuíta con restricciones.
- Para gestionar los repositorios, lo podemos hacer bien usando la consola escribiendo los comandos de Git directamente, o bien mediante un UI para controlarlo como pueden ser:
  - **Sourcetree**
  - **Smartgit**

¿Cómo funciona Git?

- Git es un software de control de versiones distribuido, en inglés **DVCS** (**D**istributed **V**ersion **C**ontrol **S**ystem).
- Cada ordenador local de cada desarrollador tiene almacenada una réplica de todo el repositorio.
- Así, aunque el servidor fallase de forma catastrófica, se podría recuperar el código desde cualquiera de los ordenadores.
- Esto agiliza el desarrollo, ya que nadie depende directamente del código alojado en el servidor principal, sino que pueden ir subiendo sus cambios a su copia local del repositorio, para luego solicitar una subida de su código al servidor.
- Cada desarrollador crea una rama a partir de la rama principal de desarrollo para que sus cambios sólo afecten a ese desarrollo paralelo y no a la rama principal.
- La desventaja es que al no estar todos los cambios en todo momento en el servidor central, no se tiene una instantánea exacta del estado del código en cada momento.



¿Cómo funciona Git?

- Para gestionar **Git** de forma básica en el proceso de desarrollo, los pasos habituales son los siguientes:
  - Para crear un repositorio nuevo, en un directorio nuevo ejecutamos ***git init***
  - Para clonar un repositorio existente, en un directorio vacío ejecutamos ***git clone <camino del repositorio>***
  - Al clonar el repositorio, en tu ordenador local se crean tres "árboles" administrados por **Git**:
    - El **directorio de trabajo**. Es el que contiene los archivos del proyecto.
    - El **Index**. Zona intermedia.
    - El **HEAD**. Apunta al último commit realizado
  - Para registrar los cambios realizados en el Index, ejecutamos:
    - ***git add <nombre del archivo>*** si queremos añadir un archivo concreto
    - ***git add -A*** si queremos añadir todos los cambios realizados
    - ***git add .*** Si queremos añadir archivos nuevos y modificaciones, **sin borrados**
    - ***git add -u*** Si queremos añadir modificaciones y borrados, **sin nuevos archivos**

¿Cómo funciona Git?

- Para pasar esos cambios al HEAD de nuestra copia local del repositorio, debemos ejecutar ***git commit -m "Mensaje del commit"***.
- Todos nuestros cambios están ahora en HEAD pero no en remoto principal.
- Para que estas modificaciones pasen de nuestra copia local del repositorio al repositorio remoto principal, debemos ejecutar ***git push origin <rama>***. La rama usualmente es la rama en la que estamos trabajando.

- Para trabajar con varias ramas podemos proceder de este modo:
  - Si queremos crear una rama y cambiar inmediatamente a ella ejecutamos ***git checkout -b nombre_rama*** o si la rama ya existía cambiar a ella directamente con ***git checkout nombre_rama***
  - Si queremos cambiar a la rama principal ejecutamos ***git checkout master***
  - Podemos borrar una rama si no estamos posicionados en ella ejecutando ***git branch -d nombre_rama***
  - Como ya vimos, para subir los cambios realizados en nuestra rama al repositorio remoto debemos ejecutar ***git push origin <nombre_rama>***

¿Cómo funciona Git?

- Para actualizar nuestra copia local del repositorio:
  - Podemos ejecutar ***git pull*** para actualizar nuestra copia de la rama a la versión más actualizada de esta rama en el remoto.
  - Para fusionar otra rama en nuestra rama activa podemos ejecutar ***git merge <rama a fusionar>***
  - En caso de haber conflictos entre el contenido de la rama remota y nuestra copia en local, deberemos solucionar esos conflictos manualmente
  - Una vez esos archivos con conflictos tienen dichos conflictos solucionados, los debemos añadir de nuevo con ***git add <nombre del archivo>***
  - Antes de fusionar se pueden revisar las diferencias entre ramas con ***git diff <rama origen> <rama destino>***
- Versionado del proyecto
  - Hay que crear una etiqueta con cada versión del producto que se publique. Para ello se utiliza el comando ***git tag <nombre de etiqueta> <id del commit>***
  - El id del commit lo podemos obtener con el comando ***git log***

¿Cómo funciona Git?

- Si algo sale mal (que puede pasar, somos humanos):
  - Se pueden reemplazar los cambios locales usando ***git checkout*** -- ***<nombre del archivo>*** esto reemplaza los cambios de tu directorio de trabajo con el último contenido del HEAD
  - Para deshacer los cambios locales y commits, podemos traer la última versión del servidor y apuntar a nuestra copia local ejecutando ***git fetch origin*** y ***git reset --hard origin/master***, donde **master** se puede sustituir por el nombre de la rama principal de desarrollo *(develop, etc.)*
  - Aunque con estos comandos se puede trabajar con **Git** de forma básica, este es un software complejo y con innumerables características en las cuales se puede profundizar consultando su documentación:
    - **Pro Git, el libro oficial de Git** (versión traducida al castellano).