Git

¿Qué es Git?
Git es un *software* de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

Primeros Pasos

Es necesario tener instalado **Git**

Todos los comandos se deben ejecutar en la terminal (del sistema operativo o de vs code) estando dentro del directorio del proyecto.

Creamos un repositorio local en la carpeta del proyecto:
`git init`
Podemos ver el estado del repositorio con el comando:
`git status`

Git Stage

Git funciona haciendo seguimiento a los archivos del proyecto para detectar cambios en los mismos. Git no sigue automáticamente a los archivos del proyecto donde se inicializó, sino que tenemos que agregarlos a su lista de seguimiento manualmente con el comando:
`git add nombre_archivo`

Por suerte podemos añadir todos los archivos/carpetas del directorio actual con un solo comando:
`git add .`

.gitignore

Git no rastreará archivos/carpetas listados en el archivo `.gitignore`, este debe ir en la carpeta principal del proyecto.

Para quitar del seguimiento archivos/carpetas que incluímos en gitignore pero git ya los tenía en siguiendo usamos:

1. `git rm -r --cached .`
2. `git add .`
3. `git commit -am "quitamos los archivos ignorados"``

Esto no borra los archivos físicamente, git simplemente dejará de seguirlos.

Comandos alternativos avanzados:
`git update-index --skip-worktree carpeta/nombre_archivo`
`Volver a agregarlo al seguimiento:`
`git update-index --no-skip-worktree carpeta/nombre_archivo`

Git Branch

Una rama (branch) es una bifurcación del código inicial del proyecto tal como lo es una rama para un árbol. Crear ramas nos permite realizar distintas modificaciones al proyecto sin modificar la rama principal y, una vez terminadas y probadas las modificaciones, aplicarlas a la rama principal. Esto nos permite trabajar en equipo o solos sin preocuparnos por "romper" la rama principal del proyecto. Git nos permite crear infinitas ramas de trabajo.
Listar ramas locales:
`git branch`
Crear rama:
`git branch nombre_rama`
Cambiar nombre a una rama:
`git branch -m nuevo_nombre`
Eliminar una rama:
`git branch -D nombre_rama`

Git Checkout

Nos permite cambiar entre ramas:
`git checkout nombre_rama`

Es posible crear y cambiarnos a una rama nueva con un solo comando:
`git checkout -b rama_nueva`

La rama a la que nos cambiemos será la rama "activa": todos los cambios que hagamos se le aplicarán a esta y no a otras ramas.

Git Stash

El comando `git stash` almacena temporalmente (o guarda en un stash) los cambios realizados en el código de la rama activa para que podamos trabajar en otra rama y, más tarde, regresar y aplicar los cambios que veníamos haciendo.

Para volver a cargar los cambios que hicimos stash previamente usamos:
`git stash pop`

Git no guarda en un stash los cambios efectuados en archivos sin seguimiento o ignorados, por lo tanto si creamos archivos y queremos guardarlos en un stash tendremos que agregarlos al seguimiento y luego hacer el stash o bien incluir el parámetro **-u** con el cual git si va a incluirlos en el stash:
`git stash -u`

Git Commit

Confirma (guarda) los cambios realizados. El "mensaje" generalmente se usa para asociar al commit una breve descripción de los cambios realizados.
`git commit -m "mensaje"`

Corregir mensaje de un commit anterior:
`git commit --amend -m "mensaje nuevo de commit"`

Git Log

Nos muestra el historial de commits y sus códigos de referencia:
`git log`
`git reflog`

Los códigos de referencia nos sirven para realizar distintas operaciones

Git Revert

Nos permite volver a un commit anterior deshaciendo los commits posteriores al indicado:
`git revert codigo_referencia_commit`

Si el revert nos abre un editor de texto pulsamos escape y escribimos :wq y damos enter.

Git Merge

Fusiona la rama indicada con la rama activa:
`git merge nombre_rama`

Si el merge genera conflictos podemos resolverlos con las opciones que nos brinda o bien cancelarlo con el comando:
`git merge --abort`

Trabajando con un repositorio remoto

Un repositorio remoto no es más que un repositorio local pero alojado en otra computadora. Github, gitlab, bitbucket, etc… Son “nubes” que permiten alojar repositorios git.

Hay dos formas de “conectar” a un repositorio remoto:

1 - Tenemos un repositorio local y queremos subirlo a la nube
Para esto necesitamos crear un repositorio remoto en cualquier nube de repositorios git (github, gitlab, bitbucket, etc) y obtener su URL (dirección). Con esta URL procedemos a conectar nuestro repositorio local al repositorio remoto:
`git remote add origin url-repositorio-remoto.git`

“origin” puede ser cualquier nombre, se usa para indicar a cual repositorio nos referimos, en este caso “origin” es el nombre que le dimos al repositorio remoto.

Git push

Envía la rama local con sus confirmaciones (commits) hacia el repositorio remoto**:**
`git push origin rama_local`

Si la rama no existe en el repositorio remoto (ni referenciamos una existente), se creará una rama en el repositorio remoto con el mismo nombre de la rama local y quedarán “referenciadas”.

2 - El repositorio remoto ya existe y necesitamos trabajar con él localmente
El proceso de descarga de un repositorio remoto a nuestro entorno local se llama clonado de repositorio. Para clonar un repositorio remoto a nuestro entorno local ejecutamos el comando:
`git clone url-repositorio-remoto.git`

Git Fetch
Obtiene nuevas ramas y cambios en ramas de un repositorio remoto:
`git fetch origin`

Git Pull
Trae el contenido de una rama remota actualizando el contenido de la rama local que tengamos activa:
`git pull origin rama_remota`

No importa la forma en que conectemos nuestro repositorio local al repositorio remoto, todos los comandos anteriores se utilizan para enviar o descargar ramas y sus contenidos.

Las ramas locales y remotas se identifican y enlazan por un código de referencia.
Así es que podemos tener una rama local con un nombre distinto al nombre de la rama remota a la que está “conectada”.

Para cambiar la referencia de la rama local activa usamos:
`git branch -u origin/rama_remota_a_referenciar`