


## HOMEWORK 1

### ¿Qué es GIT?

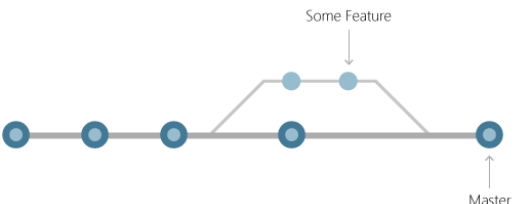
GIT es un estándar mundial de control de versiones distribuido, lo que permite que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan su copia del repositorio con la copia del servidor. GIT ofrece ramas de características, esto significa que cada desarrollador de software en el equipo puede dividir una rama de características que proporcionará un repositorio local aislado para hacer cambios en el código.

### Control de versiones con GIT



Cada vez que se guarda un trabajo en GIT se crea una confirmación. Una confirmación es una instantánea de todos los archivos en un momento dado. Si un archivo no ha cambiado de una confirmación a la siguiente, Git usa el archivo almacenado anteriormente.

Las confirmaciones crean vínculos a otras confirmaciones, formando un gráfico del historial de desarrollo. Es posible revertir el código a una confirmación anterior, inspeccionar cómo cambiaron los archivos de una confirmación a la siguiente y revisar información como dónde y cuándo se realizaron los cambios. Las confirmaciones se identifican en GIT mediante un hash criptográfico único del contenido de la confirmación. Dado que todo está has, es imposible realizar cambios, perder información o archivos dañados sin que GIT lo detecte.



Cada desarrollador guarda los cambios en su propio repositorio de código local. Como resultado, puede haber muchos cambios diferentes en función de la misma confirmación. GIT proporciona herramientas para aislar los cambios y volver a combinarlos posteriormente.

Las ramas, que son punteros ligeros para trabajar en curso, administran esta separación. Una vez finalizado el trabajo creado en una rama, se puede combinar de nuevo en la rama principal o troncal del equipo.

## *Estados de un archivo en GIT*

GIT tiene tres estados principales en los que se pueden encontrar los archivos: confirmado (committed), modificador (modified) y preparado (staged). Confirmado: significa que los datos están almacenados de manera segura en tu base de datos local. Modificador: significa que has modificado el archivo, pero todavía no lo has confirmado a tu base de datos. Preparado: significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

## *¿Cómo se configura un repositorio?*

Para crear un nuevo repositorio, debemos posicionarnos en la carpeta raíz de nuestro proyecto y ejecutar el comando: `git init`, al ejecutar este comando tendremos una nueva carpeta oculta llamada `.git` con toda la base de datos con cambios atómicos de nuestro proyecto.

Git está optimizado para trabajar en equipo, por lo que debemos proporcionar información de nosotros. Ejecutar una única vez los comandos con nuestra información:

```
git config --global user.email tu@email.com
```

```
git config --global user.name "tu nombre"
```

Los comandos para iniciar un repositorio en GIT son:

- `git init`: para inicializar el repositorio git y el staged
- `git add nombre_del_archivo.txt`: enviar el archivo al staged
- `git status`: ver el estado, si se requiere agregar al staged o si se requiere commit
- `git config`: para ver las posibles configuraciones
- `git config --list`: para ver la lista de configuraciones hechas
- `git config --list --show-origin`: para mostrar las configuraciones y sus rutas
- `git rm --cached nombre_del_archivo.txt`: para eliminar el archivo del staged(ram)
- `git rm nombre_del_archivo.txt`: para eliminar del repositorio

## *Comandos en GIT*

- `git config`  
Establece valores de configuración para tu usuario, email, entre otros.  
`git config --global user.name "mi nombre"`  
`git config --global user.mail usuario@dominio.com`
- `git init`  
Inicializar un repositorio git – crea el directorio `.git` inicial en un proyecto nuevo o existente.
- `git clone`  
Crea una copia de repositorios GIT de una fuente externa. También añade la ubicación original como remota.
- `git add`  
Añade cambios de archivos en tu directorio de ensayo a tu index.

- [git rm](#)  
Elimina archivos de tu index y de tu directorio de ensayo para que no se rastreen.
- [git commit](#)  
Toma todos los cambios escritos en el index, crea un nuevo objeto de confirmación que apunta a él y establece la rama para que apunte a esa nueva confirmación.
- [git status](#)  
Muestra el estado de los archivos en el index en comparación con los del directorio de trabajo. Enumera los archivos que no están rastreados (directorio de trabajo), modificados (rastreados pero aún no actualizados en el index) y almacenados (añadidos al index y listos para comprometerse).
- [git branch](#)  
Para listar las ramas existentes, incluyendo las ramas remotas, si se proporciona "-a". Crea una nueva rama si se proporciona un nombre.
- [git merge](#)  
Fusiona una o más ramas con otra rama activa y crea automáticamente un nuevo commit si no hay conflictos.
- [git reset](#)  
Resetea el index y directorio de trabajo al último estado comprometido.
- [git pull](#)  
Obtiene los archivos del repositorio remoto y los combina con el local.
- [git push](#)  
Envía todos los objetos modificados localmente al repositorio remoto.
- [git remote](#)  
Muestra todas las versiones remotas del repositorio.
- [git log](#)  
Muestra una lista de confirmaciones en una rama que incluye los detalles correspondientes.
- [git diff](#)  
Genera archivos parche o estadísticas de diferencias entre rutas o archivos en el repositorio git, o índice o directorio de ensayo.
- [git archive](#)  
Crea un archivo tar o zip que incluye el contenido de un solo árbol desde el repositorio.
- [git gc](#)  
Recolector de basura para el repositorio. Optimiza el repositorio. Debe ejecutarse ocasionalmente.
- [git fsck](#)  
Realiza una comprobación de integridad del sistema de archivos Git, identificando objetos corruptos.
- [git prune](#)  
Elimina objetos ya que no apuntan a ningún objeto en ninguna rama accesible.