# CHEATSHEET GIT

**$ git init:** crea un repositorio local vacía en la carpeta donde tenemos los archivos(local: en la pc, remoto: en github)

**$ git config user.name “fpicco”:** para asignar el autor. Usar el user de gitHub

**$ git config user.name:** da el autor

**$ git config user.email “flor.picco@gmail.com”:** el de github

**$ git config user.email:** da el mail

**$ git config --global user.name “fpicco”:** todos los repos de la pc tendrán este nombre

**$ git config --global user.email “flor.picco@gmail.com”:** idem

**$ git config --global unset user.name “fpicco”:** elima el registro que refiere al usuario

**$ git config --global unset user.email “flor.picco@gmail.com”:** elima el registro que refiere al email

Si en la carpeta con el repo ponemos el comando ls -a vamos a ver los archivos ocultos de git

Para conectar el repo local con el remoto:

**$ git remote add origin https://github.com/fpicco/CTD-II-C3.git :** conecta con el repo en github previamente creado, para que el local y el remoto se sincronicen

**$ git remote -v:** para corroborar que estén conectados correctamente

Ya podremos crear archivos en el repositorio local y enviarlos posteriormente al repositorio remoto en la nube

Agregando archivos al repo:

Esto funciona si estamos dentro de una carpeta con un repo inicializado

**$ git add:** indicando el o los archivos que queremos agregar

**$ git status:** devuelve los archivos y el estado respecto al repo.

Avisa cuando hay archivos nuevos o cuando hay archivos modificados con respecto al repo remoto; para el control de versiones.

**$ git add . :** para agregar todos los archivos del repo local

Eliminar archivos del repo

**$ git rm archivo**

**$ git rm -r directorio:** elimina la carpeta y sus archivos

Confirmando archivos:

Commit: confirmación a través de la cual le decimos al repo que los archivos que fuimos agregando los deseamos oficialmente como un pequeño paquete con esa versión.

Primero debemos agregar los archivos modificados al repositorio (crear - agregar - commitear)

**$ git commit -m “mensaje descriptivo” :** crea el paquetito

**$ git log** : registra el historial de los commits

**$ git config --list:** lista la configuración actual del repo

**$ git log:** muestra el historial de commits

**$ git log --URL:** muestra el historial de un archivo especifico

**$ git log --author=USUARIO:** muestra el historial de un usuario en particular

**$ git checkout nombreDeArchivo:** deshace un cambio local que no se haya añadido al área a de trabajo temporal por ej, un archivo borrado
**$ git reset HEAD nombreDeArchivo:** deshace el cambio local cuando el archivo ya fue agregado al área temporal

Subiendo los proyectos a gitHub:

Para pasar archivos del repo local al repo en github: antes de subir los archivos a github, deben estar commiteados.

**$git push origin* master**:** envía los archivos que tenemos en el repo local al remoto

**O la rama que deseemos

*para git, origin es el repo remoto

Bajando los proyectos de gitHub:

Queremos bajar a nuestra pc la última versión del repo remoto

**$ git clone https://github.com/fpicco/prueba-node.git:** permite crear una copia exacta en la pc de todos los archivos existentes en el repo remoto. Se ejecuta una sola vez, cuando no tenemos esos archivos en la pc, si ya los tenemos y los queremos actualizar, es otro comando:

**$ git pull origin main/master**:** no descarga nuevamente los archivos, solo descarga y actualiza los que hayan sufrido modificaciones o sean nuevos.

**O la rama que deseemos

Ramas

Rama: dentro de un repo es una copia alternativa del mismo hasta el momento para poder agregar nuevas funcionalidades sin modificar la línea original de tiempo ni el código de master/main, sería una versión 2 del proyecto en la cual podemos probar cosas nuevas y si quedan bien, la podemos fusionar con la rama principal.

**$ git branch :** lista todas las ramas del repo

**$ git branch nombreDeNuevaBranch :** Crea una nueva rama

**$ git branch -d nombreDeUnaBranch:** elimina la rama, git evita que la eliminemos si tiene cambios que no se fusionaron con main

**$ git branch -D nombreDeUnaBranch:** fuerza la eliminación de la rama aunque tenga cambios no fusionados

HEAD es un puntero que indica cual es la branch actual, por defecto HEAD

Git checkout

**$git checkout nombreDeUnaBranch:** para moverse de una rama a otra (si no tenemos cambios, si hay cambios hay que deshacerlos o commitearlos)

**$git checkout -b nombreDeOtraNuevaBranch:** crea una nueva branch y apunta a ella

Comandos para el repo remoto

**$ git remote show origin:** ver info de los repos remotos

**$ git remote rename origina nombreNuevoParaElRepo:** renombra el repo remoto

**$ git remote rm nombreRepoEnGit:** desvincula un repo remoto

Para modificar un commit que acabamos de hacer:

**$ git commit --amend:** nos abrirá un editor (bin) para que modifiquemos el mensaje del commit. Lo modificamos con insert, con escape volvemos al modo lector y luego para salir :wq

**$ git branch -M main:** para cambiarle el nombre de master a main

**$ git branch nuevaRama:** creamos la rama nueva

**$ git branch -a:** nos va a dar las ramas que tenemos

**$ git checkout dev:** nos cambiamos a la rama dev

**$ git merge dev:** desde main, ponemos este comando y va a traer el contenido de dev a main

**$ git push -u origin dev:** crea una rama en el repo remoto (si la creamos en local, la subimos)