

\*El numero de #Gatitos es para dar enfasis a los titulos o marcas

\*Si tenemos problemas con /bin/sh usar el siguiente comando

```
chsh -s /bin/bash
```

---

## Acordeon GIT

### \$ git init

- Inicializa un nuevo repositorio.

### \$ git --version

- Muestra la version que tenemos instalada

### \$ git config --global user.name "Tanya Ricci"

- Configura el nombre de usuario de nuestros archivos

### \$ git config --global user.email "tiaricci@gmail.com"

- Configura el correo electronico de nuestros archivos

### \$ git config --global core.editor

# “vi”

- Configura el editor de textos que sera por default

## \$ git status

- Muestra el estado actual de nuestro repositorio, nos permite saber si el repositorio contiene archivos sin seguimiento, o si estan listos para ser registrados.

## \$ git add <archivo> / -A

- Este comando empiza a seguir a uno o mas archivos y los agrega al area de preparacion, generando un nuevo estado de nuestro proyecto.

La bandera -A agrega todos los archivos del repositorio.

## \$ git commit

- Este comando registra nuestro nuevo estado y lo registra en la historia de nuestro repositorio.
- Por lo general este comando se usa con la bandera -m y un pequeno texto que describa lo que hicimos.
  - git commit -m “first commit”

## \$ git log

- Este comando nos muestra el historial de commits que hemos

hecho en nuestro proyecto.

- La bandera --oneline muestra cada entrada en una sola linea.
- Tambien es posible ver la historia de un solo archivo, pasando como argumento el nombre de este.

## **.gitignore**

- Este archivo nos permite ignorar archivos o directorios los cuales no queramos que entren en el seguimiento de nuestro repositorio.
  - nombres de archivos o carpetas
  - wildcards\*

## **\$ git checkout**

- Este comando nos permite movernos entre commits o incluso ramas de nuestro repositorio.
  - Lleva como argumento el id del commit o parte de.  
=== Para regresar a mi estado original o mi rama original  
=== git checkout master (Regreso a mi rama principal)  
=== git checkout "numero de la rama a la que quiero regresar"  
=== git checkout -b inicio (Con esto creo una nueva rama llamada inicio y me cambia a esta nueva rama automaticamente)

## **\$ git revert**

- Este comando nos revierte un cambio ya registrado, creando un nuevo commit.
  - Lleva como argumento el id del commit a revertir.

## \$ git reset

- Regresa al ultimo estado guardado, borrando permanentemente cualquier cambio en el area de pruebas.
  - Lleva la bandera --hard

## \$ git clean

- Borra permanentemente los archivos no seguidos
  - Lleva la bandera -f (force)

## \$ git branch

- Lista las ramas existentes en el repositorio.
  - Si se le agrega [<nombre>] de rama como argumento, se creara una rama con ese nombre.

## \$ git merge

- Este comando fusiona dos ramas, fusiona una rama objetivo con la rama donde nos encontramos actualmente.
  - Recibe como parametro la rama objetivo.

## \$ git diff

- Este comando me muestra las diferencias del ultimo archivo y del que estas modificando actualmente, debe de ser antes de usar add, ya que despues se aceptan los cambios y no se ve que haya diferencia.

## Conflictos en el merging

- Hay que tener cuidado, a veces dos ramas pueden tener conflictos entre ellas y es necesario corregirlo.

— TAREA: Hacer dos branch y hacer dos cambios y luego un merge con proyecto.

## \$ git remote

- Este comando nos permite hacer diversas operaciones sobre repositorios remotos.
    - add: agrega un nuevo remoto
    - remove: elimina un remoto
- ```
$ git remote add <nombre> <url>
```
- ```
$ git remote remove <nombre>
```

## \$ git pull

- Este comando nos permite recibir el ultimo estado registrado del repositorio remoto.
  - Recibe como argumentos el nombre del remoto y del branch al que queremos hacer pull.

# \$ git push

- Este comando nos permite mandar los cambios registrados de nuestro repositorio local a un repositorio remoto.
  - Recibe como argumentos el nombre del remoto y del branch al que queremos hacer push.

\*\*\* Una buena practica es hacer el pull antes del push, para sincronizar cambios con el servidor \*\*\*

# \$ git clone

- Este comando nos permite clonar un repositorio desde la web
    - Recibe como argumentos la url que queremos clonar
- \$ git clone <https://github.com/annkymFI/annkymFI.github.io.git>

# Fork

- Github ofrece una opcion, la cual nos permite copiar el repositorio de algun otro usuario y hacerlo nuestro.

# Pull Request

- Este metodo permite a los equipos de desarrollo control que cambios se realizan en un repositorio.
- El propietario del repositorio recibe peticiones de cambio, las cuales deben ser aprobadas por el.

- Tipicamente un colaborador hace fork del proyecto maestro, y al realizar sus cambios realiza el pull request.
- Tambien es posible hacer pull request entre ramas.

## Issues

- Github ofrece la capacidad de crear entradas en nuestro repositorio, las cuales pueden ser usadas para especificar varias cosas, desde mejoras al sistema, problemas a resolver o nuevas ideas que implementar.