Introducción a Git y GitHub

Procesamiento Digital de Señales

I. Mieza / A. Greco / D. Leguizamon / G. Marzik / N. Passano

¿Que es Git?

- **Git** es un **sistema de control de versiones local**, se utiliza en proyectos con cambios constantes en los scripts y en el desarrollo colaborativo.

¿Que es un sistema de control de versiones?

- Un sistema de control de versiones **registra los cambios del codigo**. En base a una versión incial del código, y se guardan unicamente los cambios del código, **no se genera un nuevo archivo actualizado**. Esto permite localizar los cambios en el código, quien hizo los cambios y la posibilidad de volver a un estado anterior del codigo actual.

Link de descarga: https://git-scm.com/

¿Que es GitHub?

- **GitHub** es un sitio web que **almacena repositorios de git** en un **servidor remoto.**

- Esto facilita el trabajo colaborativo, donde **varios desarrolladores trabajan sobre el mismo repositorio remoto pero cada uno edita en sus versiones locales**.

- GitHub amplía aún más su funcionalidad con respecto a Git ya que permite:
  - Host de páginas web estaticas
  - Previsualización de archivos .ipynb
  - etc.

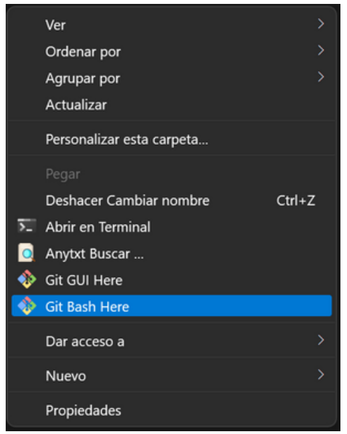
- Tip: Con el mail institucional (..@estudiantes.untref.edu.ar) tienen la versión PRO de GitHub

Link del sitio: http://github.com

¿Como se usa Git?

- Hay distintas formas de implementarlo (ej: GitHub Desktop), pero la que se presentará en esta clase es mediante la **consola de bash.**

Comandos básicos de la consola de bash



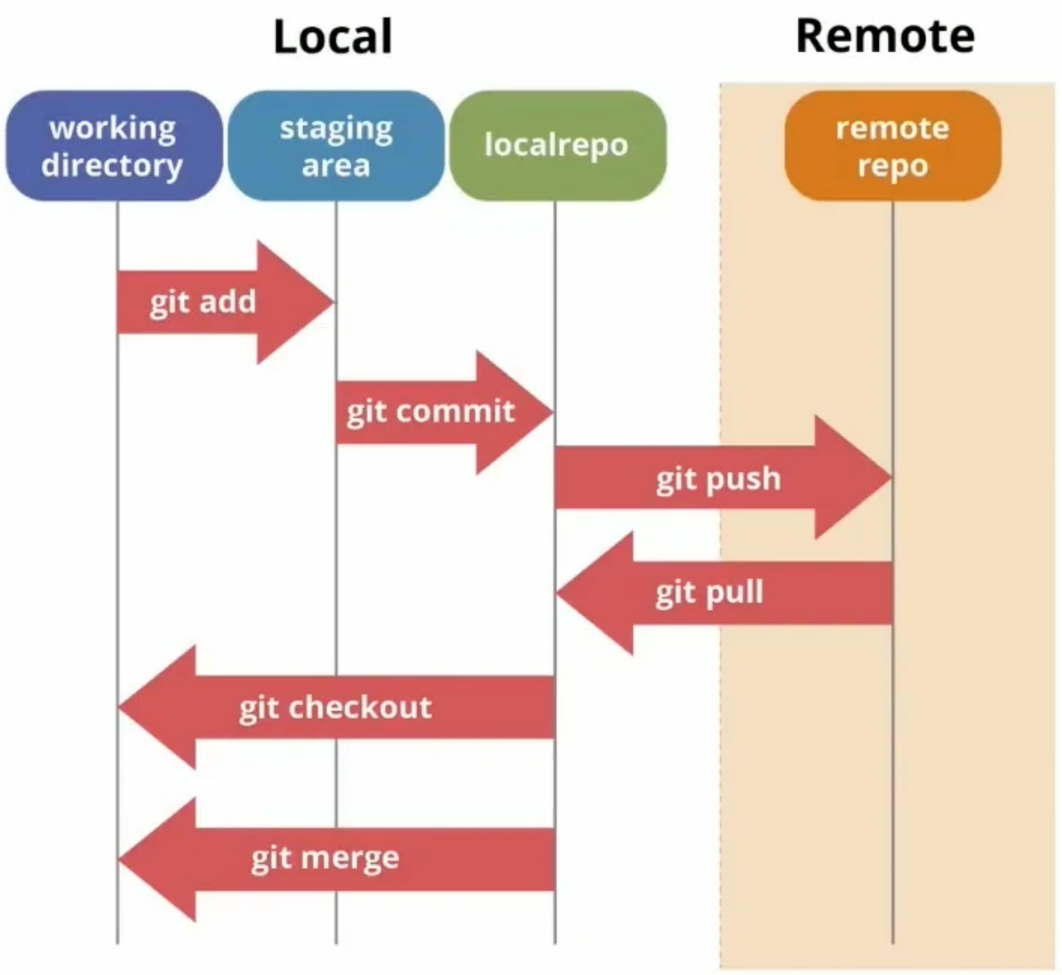
- **pwd**: Devuelve el directorio actual.
- **cd <path>**: Cambiar al directorio que se le indique.
- **ls**: Devuelve todos los archivos contenidos en el directorio actual

```
NPass@Nahuel-Laptop MINGW64 /c/python_w11/dsp
$ pwd
/c/python_w11/dsp

NPass@Nahuel-Laptop MINGW64 /c/python_w11/dsp
$ cd dsp_git_lesson/

NPass@Nahuel-Laptop MINGW64 /c/python_w11/dsp/dsp_git_lesson
$ ls
README.md  test.py

NPass@Nahuel-Laptop MINGW64 /c/python_w11/dsp/dsp_git_lesson
$
```

¿Como se manejan los cambios dentro de Git?

- De manera local, **hay 3 estados.**
  - **Working directory**
  - **Staging area**
  - **Local repository**

- De manera remota, unicamente está el repositorio que se **pushea** desde **Local Repository**



Comandos básicos de Git

- **git status**
  - Se utiliza para saber **en que estado** estan los archivos (Working Directory, Staging Area o Local Repository)

- **git add <file>**
  - Se utiliza para **pasar** los archivos del **Working Directory** al **Staging Area.** Aqui es donde Git comienza a trackear los archivos y cambios.

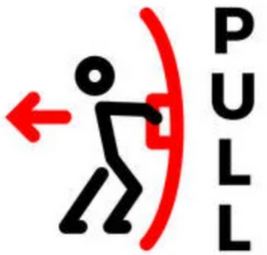
- **git commit -m "<msg>"**
  - Se utiliza para **pasar** los archivos del **Staging Area** al **Local Repository**. Se utiliza para crear snapshots o checkpoints del código.

Comandos básicos de Git

- **git clone <url>**
  - Clona un repositorio remoto a la pc de manera local a través de la url del remoto.

- **git push origin branch_name**
  - Se utiliza para **subir los cambios** (pushear) de **Local Repository** al **Remote Repository**

- **git pull origin branch_name**
  - **Trae los cambios** (pullea) que han hecho los otros desarrolladores.

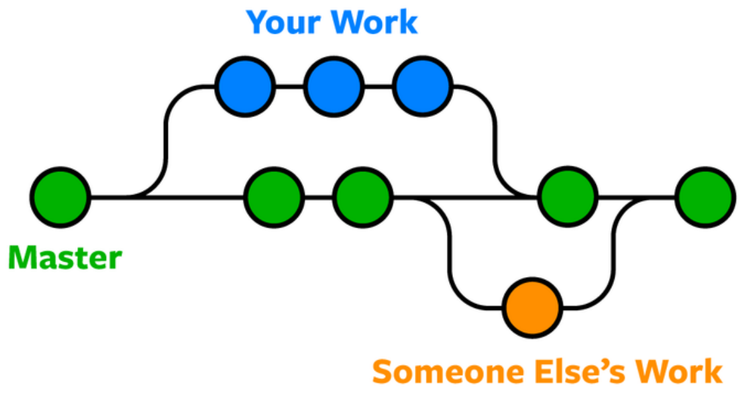


Versiones alternativas

- El manejo de versiones alternativas es un punto fuerte en el control de versiones de un código o proyecto. En Git se denominan como **ramas** o **branchs**

- Por default, Git y GitHub al crear un repositorio generan la rama **main** o **master**

- Al momento de crear una nueva rama, se **genera con el mismo contenido de la rama donde fue generada**, pero sus **cambios** y posteriores **commits son independientes y no afectaran al resto de ramas.**



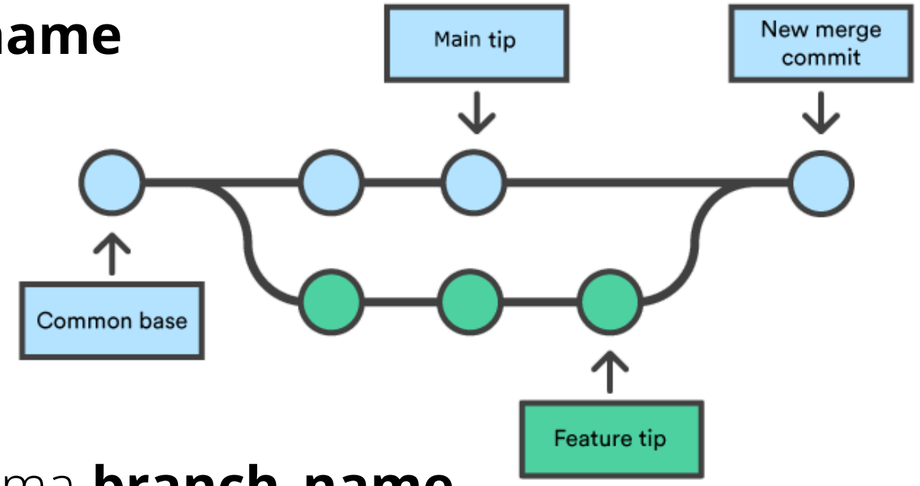
Comandos básicos para manipular ramas

- **git branch**
  - Muestra las ramas que hay en el repositorio

- **git branch branch_name**
  - **Crea** una **rama** o **branch** con el nombre **branch_name**

- **git checkout branch_name**
  - **Cambia** la rama actual a la rama **branch_name**

- **git merge branch_name**
  - **Fusiona** la rama donde se está trabajando con la rama **branch_name**




CONOCIDO
FULBO AL
INSTANT


Mucho texto

Primeros pasos

1) Crear un **repositorio remoto** en **GitHub**

2) Clonar el repositorio en nuestra computadora
    - **git clone <url>**

3) Añadir todos los scripts a utilizar, y desarrollar de manera comun y corriente.

4) Pasar todos los archivos con los que se estuvo trabajando al **Staging Area**
    - **git add <file>**

5) Commitear los cambios hechos del **Staging Area** al **Local Repository**
    - **git commit -m "<msg>"**

6) Subir mis cambios para actualizar el repositorio remoto
    - **git push origin branch_name**