ISFDyT N°70

Tecnicatura Superior en Análisis de Sistemas

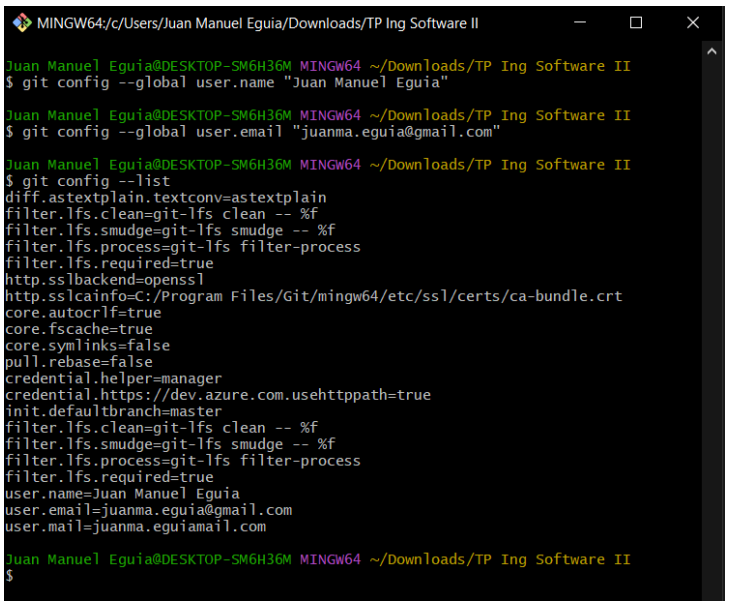
Ingeniería de Software II

- Trabajo Práctico 1 – Control de versiones con Git y GitHub
- Eguia Juan Manuel; Falabella Valentino
- 3° año
- **Nombre de la profesora:** Marina Caseres
- **Fecha de entrega:** 11/07/2025
- Enlace al repositorio GitHub del grupo:
- juanma.eguia@gmail.com ; valentinofala@gmail.com

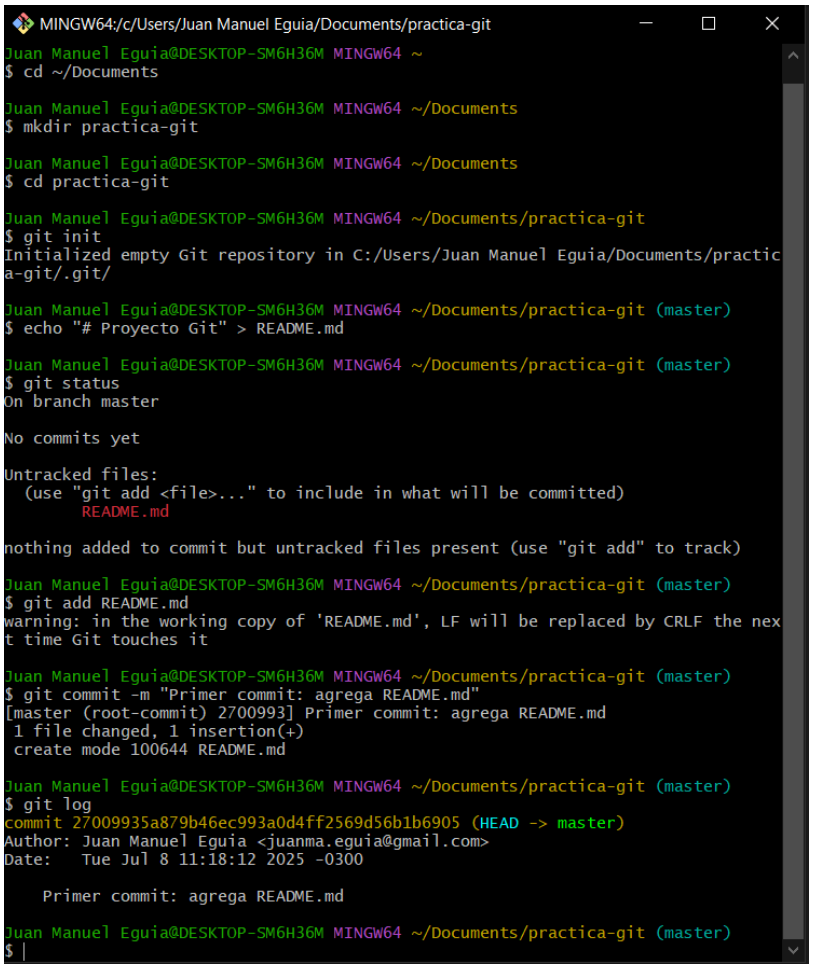
Trabajo Practico 1 – Control de versiones con Git y GitHub

Parte 1: Iniciación a Git y configuración básica

Consigna 1: Configurar Git en tu equipo



Consigna 2: Crear un repositorio local y realizar tu primer commit



¿Dónde queda guardado el control de versiones? ¿En la nube o en tu equipo?

El control de versiones queda guardado en el equipo, dentro de la carpeta del repositorio local.

¿Qué información te brinda git status? ¿Por qué es útil antes de hacer un commit?

Muestra el estado actual de los archivos en el repositorio. Es útil antes de hacer un commit porque permite revisar que cambios se registrarán.

¿Cuál es la diferencia entre agregar (add) y guardar un cambio (commit)? ¿Podés hacer un commit sin hacer un add antes?

Git add prepara los archivos para ser incluidos en el próximo commit, mientras que commit guarda esos cambios en el historial. No se puede hacer un commit de cambios nuevos si no se han agregado previamente con add.

¿Qué información muestra este comando? (git log)

Muestra el historial de commits del repositorio. Sirve para rastrear que cambios se hicieron y cuando.

Parte 2: Repositorio remoto y sincronización

Consigna 3: Crear un repositorio remoto y vincularlo

```
Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (master)
$ git remote add origin https://github.com/juaneguia/practica-git.git

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (master)
$ git branch -M main

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (main)
$ git push -u origin main
remote: Repository not found.
fatal: repository 'https://github.com/juaneguia/practica-git.git/' not found

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (main)
$ git remote add origin https://github.com/juaneguia/practica-git.git
error: remote origin already exists.

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (main)
$ git branch -M main

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 249 bytes | 249.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/juaneguia/practica-git.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

Juan Manuel Eguia@DESKTOP-SM6H36M MINGW64 ~/Documents/practica-git (main)
$ git pull origin main
From https://github.com/juaneguia/practica-git
 * branch            main       -> FETCH_HEAD
Already up to date.
```

**Inicialmente ejecuté los comandos antes de haber creado el repositorio en GitHub.*

¿Qué significa vincular tu repositorio local con uno remoto? ¿Podés trabajar sin conexión a internet después de hacer esto?

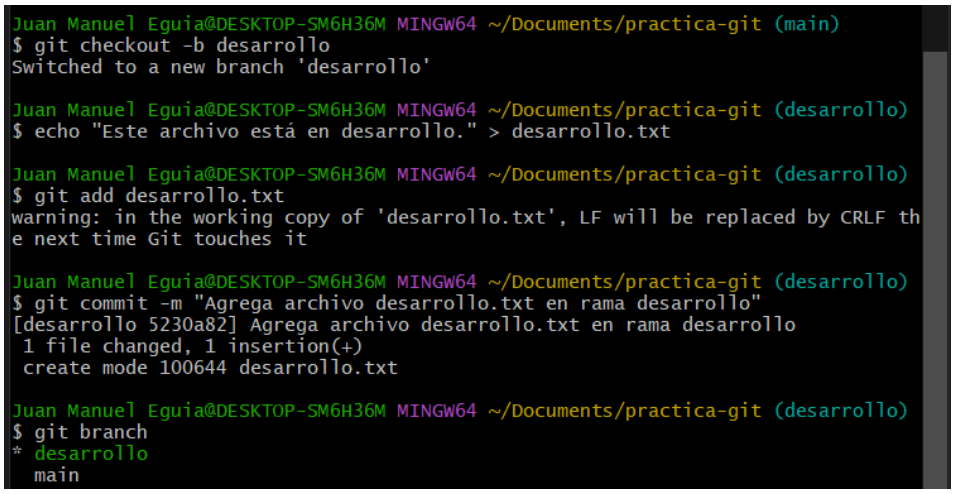
Vincular el repositorio local con uno remoto significa conectar tu proyecto en tu computadora con una copia alojada en GitHub. Esto permite subir (push) o traer (pull) cambios. Sí, se puede seguir trabajando sin conexión: los cambios se guardan localmente y se sincronizan cuando haya internet.

¿Qué estás logrando al hacer un push? ¿Por qué hay que subir los cambios al repositorio remoto si ya están en tu computadora?

Con git push estás enviando tus commits locales al repositorio remoto en GitHub. Subir los cambios es importante para tener una copia de respaldo en la nube, colaborar con otros y mantener actualizados todos los entornos que usan ese repositorio.

Parte 3: Trabajo con ramas

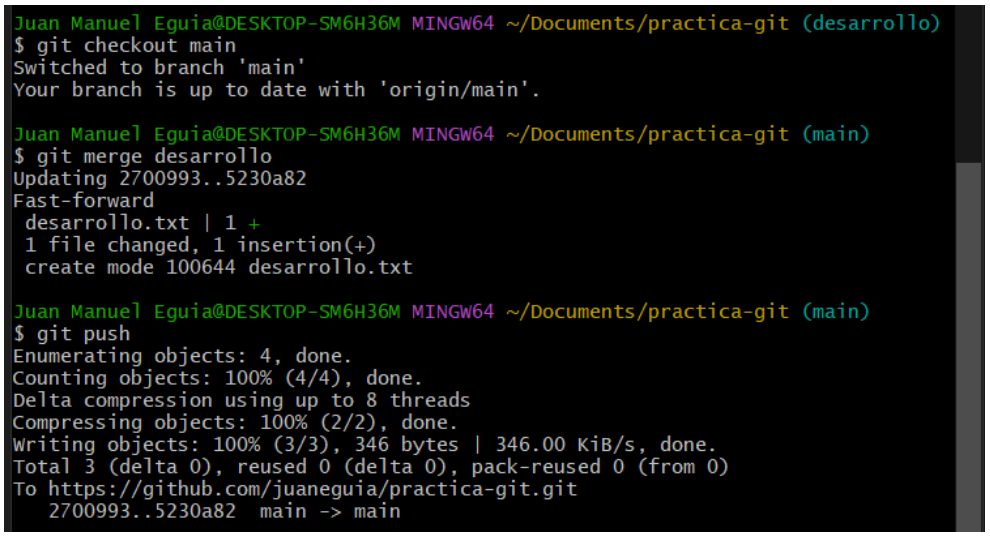
Consigna 4: Crear y trabajar en una nueva rama



¿Para qué sirve crear una nueva rama? ¿Qué ventajas tiene trabajar en ramas distintas en lugar de hacerlo siempre en la principal?

Crear una nueva rama permite desarrollar funcionalidades o hacer pruebas sin afectar la rama principal. Trabajar en ramas distintas evita errores en el código principal, facilita la colaboración y hace más fácil revisar, probar y fusionar cambios.

Consigna 5: Fusionar ramas



¿Qué pasa si hay diferencias entre las ramas al hacer un merge? ¿Podés revertir un merge si algo sale mal?)

Si hay diferencias en los mismos archivos o líneas, Git genera un conflicto que debe resolverse manualmente antes de completar el merge. Si algo sale mal, es posible revertir el merge usando git reset o git revert, aunque conviene tener un commit de respaldo antes de fusionar.