trabajo_final2_finalfinal_bueno - copia.py

Un taller de introducción a Git y GitHub

Delegación de Estudiantes EPS *delegeps@uc3m.es*

Grupo de Usuarios de Linux *info@gul.uc3m.es*

Universidad Carlos III de Madrid

Jorge Lázaro Ruiz

19 de octubre de 2023

Índice

Git es tu amigo

Git es una herramienta de **control de cambios**.

Su función es llevar un registro de versiones anteriores de un mismo archivo de texto plano (ideal para archivos de código).

Hay alternativas, pero la hegemonía de Git se debe a su modelo de ramificación (*branches*), que permite monitorear varias ramas independientes de un mismo repositorio y crearlas, mezclarlas y eliminarlas fácilmente. Ya hablaremos de esto.

La carpeta donde guardamos todos los archivos que queremos que Git controle se llama **repositorio**.
Cada una de las versiones que guardamos se registran como una **confirmación** (o ***commit***, en inglés).

¿Cómo descargo Git?

Desde la página oficial

La web oficial de Git detecta tu sistema operativo y te redirige a su correspondiente página de descargas.

`https://git-scm.com/downloads`

Desde la terminal

Git está disponible para su instalación mediante los principales gestores de paquetes.

- Con APT (Debian, Ubuntu...): `sudo apt install git`
- En MacOS suele venir preinstalado, pero puedes comprobarlo con: `git --version`
  - También puedes descargar una versión distinta con Homebrew: `brew install git`

¿Para qué quiero usar Git?

- Porque así, si la lías en una práctica, puedes volver a una versión anterior.

  ¡Se acabaron los nombres como `trabajo_final2_finalfinal_bueno - copia.py`! ¡Se acabó escribirle a tu profe porque subiste la versión equivocada de la práctica!
- Porque puedes hacer tu parte del trabajo mientras tu compa de laboratorio hace la suya (y no morir en el intento).
- Porque en cualquier equipo de desarrollo se usa muchísimo, así que vas a tener que aprender a usarlo sí o sí.

Inicializar un repositorio

Pasos

1. Creamos una carpeta
2. Abrimos la carpeta en la terminal
3. Inicializamos el repo con: `git init`

Un **repositorio** no es más que la **carpeta** donde guardaremos los archivos que queremos que Git controle.

Esto sería el equivalente a la típica carpeta de Drive que haces para una práctica.

Añadir archivos

Pasos

1. Creamos/modificamos archivos
2. Añadimos los archivos
   - `git add <archivo>`
   - `git add .`
     (Añade todos los archivos modificados del directorio)
3. Confirmamos cambios
   - `git commit -m "<mensaje>"`
   - `git commit -m "<mensaje corto>" -m "<descripción larga>"`
   - El mensaje es una descripción de lo que hemos cambiado o un nombre para el *commit*.

Cuando hemos hecho cambios en nuestro repositorio y queremos guardarlos, primero tenemos que **añadir** los archivos que hemos cambiado y luego **confirmar** esos cambios.

Cada una de estas confirmaciones o *commits* son las versiones de los archivos. Podemos comparar el estado de un mismo archivo en distintos *commits*, revertir los cambios...

Otras funciones de Git

Ramas

Las **ramas** permiten crear una «copia» del repositorio donde podemos modificar lo que queramos sin miedo a que esto afecte a la versión principal del repositorio.

Una vez las cosas funcionan en la rama nueva, podemos integrarla con la rama principal (*main* o *master*) **mezclándolas** (en inglés, haciendo un *merge*).

Crear una rama

```
git branch <rama>
```

Cambiar a otra rama

```
git checkout <rama>
```

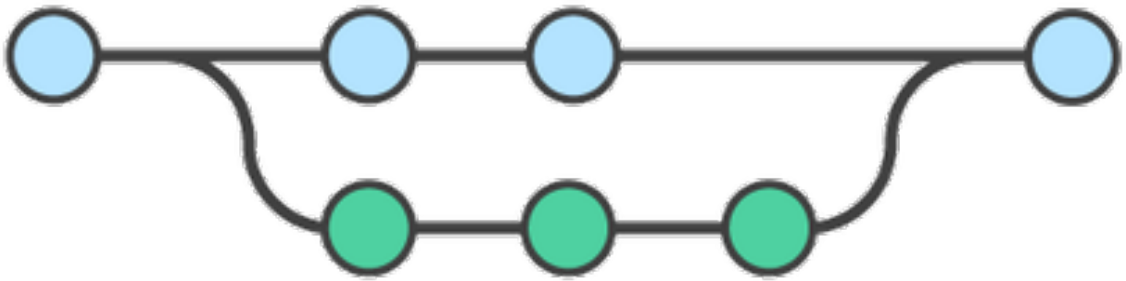


Figura: La rama secundaria se mezcla con la principal en el último *commit*.

Otras funciones de Git

Mezclar ramas (*merge*)

Rama `feature`

```
Esta línea es A.
```

Rama `main`

```
Esta línea es B.
```

¿Qué pasa si intentamos mezclar `feature` con `main`?

Mezclar ramas desde la terminal

```
git checkout main
git merge feature
```

Git nos avisará de que hay un **conflicto** en el archivo y que no ha podido mezclarlos automáticamente.

Resolver conflictos

Si abrimos el archivo conflictivo, nos encontraremos con esto:

Interfaz de resolución de conflictos

```
<<<<<<< HEAD
Esta línea es B.
=======
Esta línea es A.
>>>>>>> feature
```

Tendremos que decidir manualmente con qué versión queremos quedarnos.

Cuando hayamos resuelto el conflicto, podemos añadir el archivo y, por fin, hacer un *commit*.

Otras funciones de Git

Reservar (*stash*)

Podemos almacenar temporalmente (o guardar en un *stash*) los cambios en el código para poder trabajar en otra cosa y, más tarde, regresar y volver a aplicar los cambios.

Es práctico cuando estamos en medio de un cambio en el código y no lo tenemos todo listo para confirmar, pero queremos ponernos a trabajar en otra cosa.

- `git stash`
  Reserva los cambios no confirmados.
- `git stash pop`
  Recupera los cambios previamente reservados (y elimina la reserva).
- `git stash apply`
  Recupera los cambios previamente reservados (sin eliminar la reserva).

GitHub: el modo *PvP* de Git

GitHub es una plataforma en línea que permite a los desarrolladores compartir, gestionar y colaborar en repositorios de Git.
Los repositorios que tenemos subidos a GitHub son **repositorios remotos**, y podemos sincronizarlos con repositorios locales.

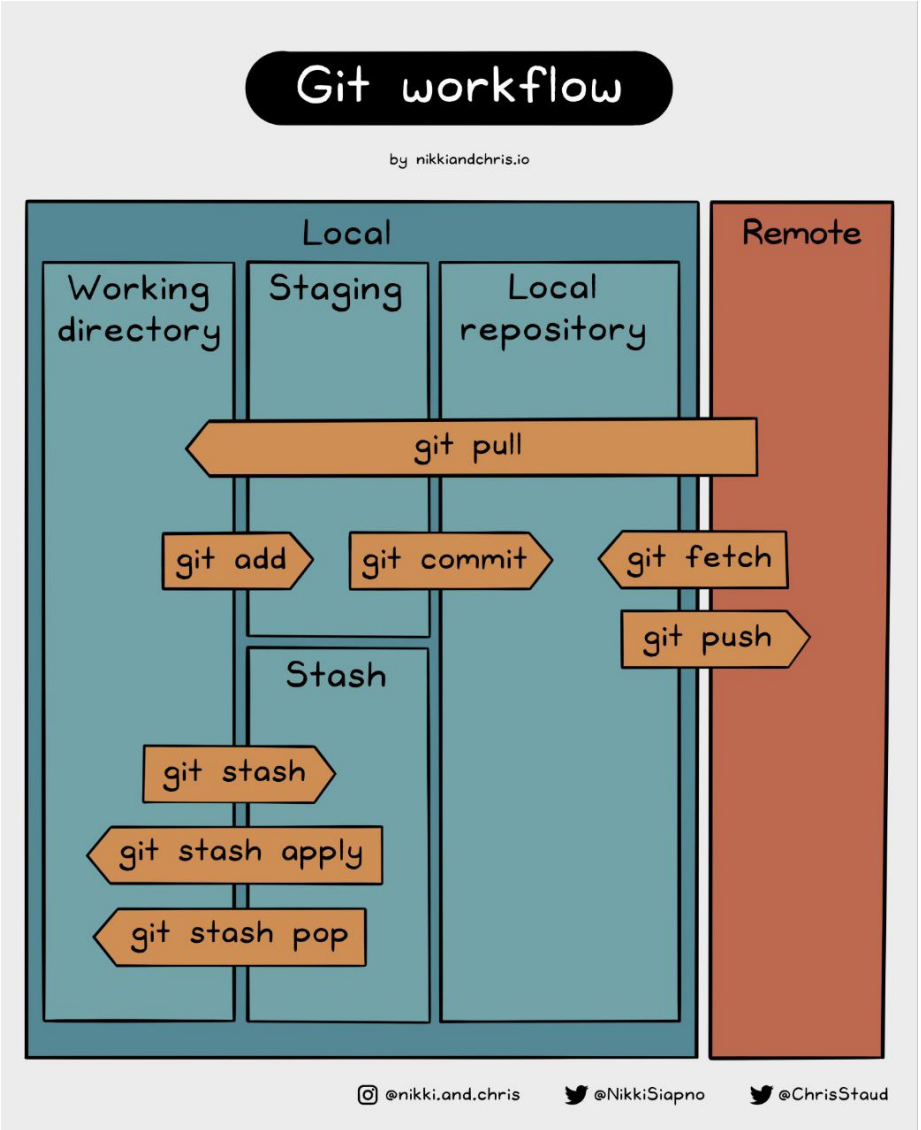
```
https://www.github.com
```

Superconsejito: Podéis solicitar GitHub Pro si utilizáis vuestro correo institucional de la UC3M.

Algunas funciones de GitHub

Repositorios remotos

- `git add remote`
  Vincula el repositorio local con uno remoto.
- `git pull`
  Descarga (y fusiona con la rama actual) los cambios más recientes de un repositorio remoto.
  Si no queremos fusionarlos, podemos usar `git fetch`.
- `git push`
  Sube los cambios del repositorio local al remoto.



- **Solicitudes de incorporación de cambios (*pull request* o PR)**
  Cuando hemos terminado nuestro trabajo en una rama, podemos abrir una *pull request* para pedir que mezclen nuestra rama con la rama principal.
  Esto es útil al dividir un trabajo en grupo. Cada integrante trabaja en su rama y, cuando ha acabado, hace una PR. Así, nadie se carga el trabajo de nadie.
- **Bifurcaciones (*fork*)**
  Hacer una copia de un repositorio que no sea nuestro para poder modificarlo como queramos.
  Más adelante, podemos solicitar que incorporen los cambios de nuestro *fork* mediante una PR al repo original.

GitHub con SSH

Generar nueva clave SSH

```
ssh-keygen -t ed25519 -C <email@ejemplo.com>
```

Añadir la clave SSH

```
https://github.com/settings/ssh/new
```

Clonar un repositorio

Clonar un repositorio es descargar todo su contenido.

Clonar con SSH

```
git clone git@github.com:<usuario>/<repositorio>.git
```

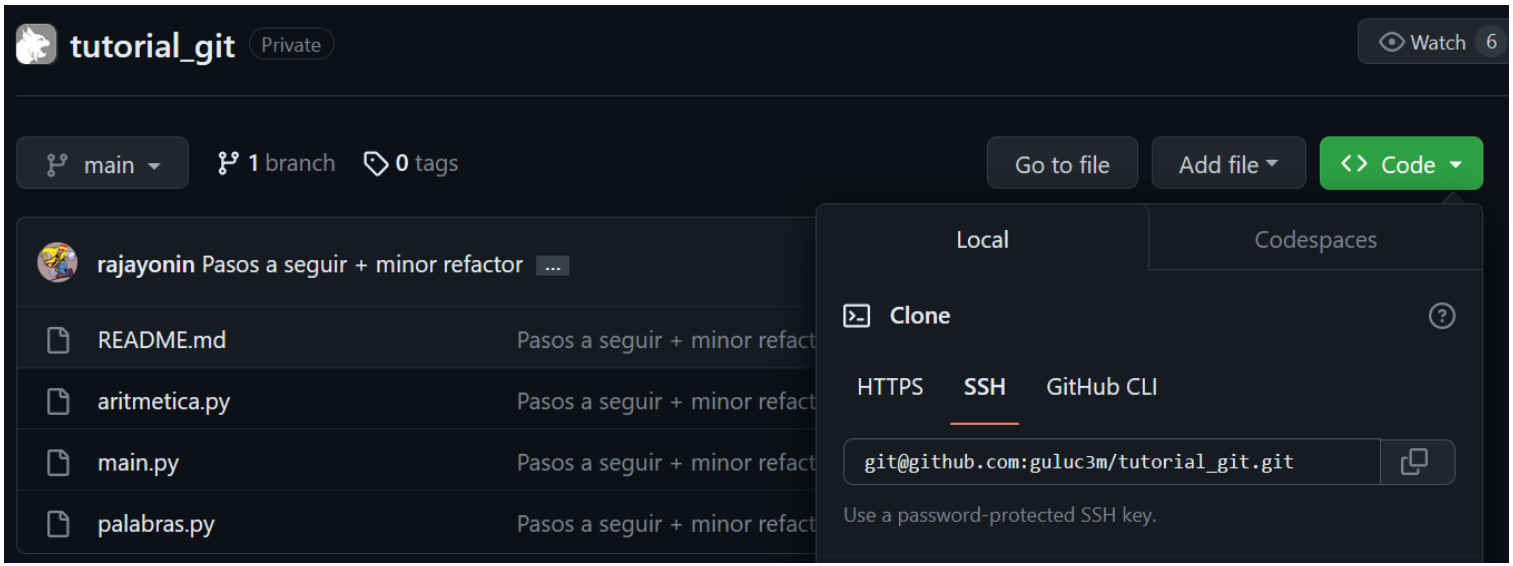


Figura: Desplegando *Code* encontramos la dirección SSH del repo.

Manos a la obra

Ahora que habéis clonado el repositorio, podéis seguir los pasos que aparecen descritos en el `README.md` y practicar lo que hemos aprendido.

Si tenéis cualquier pregunta, no dudéis en levantar la mano.

Para más información

Daniel Rodríguez (2021)
Taller Introducción a Git (GUL)
https://www.youtube.com/watch?v=jvsneGS00Tw

Bilal Arslan (2021)
arslanbilal/git-cheat-sheet
https://github.com/arslanbilal/git-cheat-sheet