Herramientas Computacionales para la Astroinformática

Cristian A. Vega-Martínez (oficina: IMIP, Académicos 2)
Facundo A. Gómez

Administrador de versiones

Introducción a GIT

¿Qué es GIT?

Git es un **sistema de control de versiones distribuido** de **código abierto** diseñado para manejar proyectos de diferentes tamaños con velocidad y eficiencia. Fue creado por Linus Torvalds en 2005 para el desarrollo del kernel de Linux.

Permite a desarrolladores y equipos rastrear y gestionar cambios en el código fuente de un proyecto a lo largo del tiempo.

Git es distribuido (a diferencia de otros): cada persona tiene **una copia completa** del historial del proyecto en su máquina local.

Es una herramienta que nos permite llevar un registro de los cambios que se realizan en un directorio a elección.

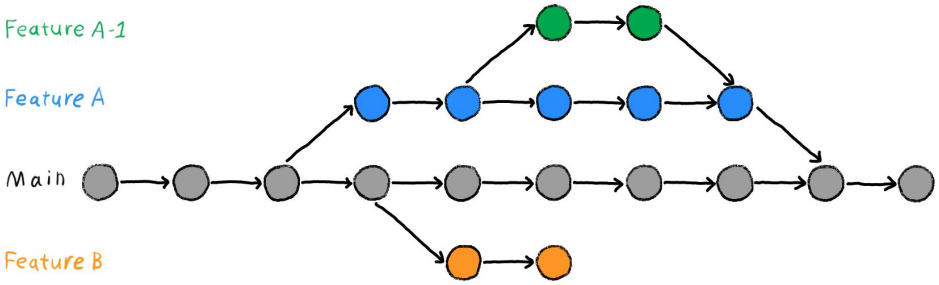
¿Para qué se puede usar GIT?

- Mantener **historial de cambios** de los archivos de un directorio personal:
  - códigos desarrollados (cálculos, análisis, ejemplos, parámetros, etc).
  - archivos de texto generales (libros, papers, tesis, datos < 20 MB)
- Crear **ramas** de desarrollo **derivadas**, administradas y accesibles en el mismo directorio.
- **Sincronización** entre diferentes computadoras.
- **Colaboración** con otras personas.
- Identificación de cambios realizados por c/u.
- Comparar diferencias entre distintas revisiones.
- Recuperar información antigua (descartada).

Con git, cada registro histórico se denomina **revisión**, y deben crearse **manualmente**.

Esquema de trabajo

utilizando git



© The Turing Way

1) Inicializar git en un directorio, creando un **repositorio**.
2) **Modificar**, **agregar** o **borrar** archivos.
3) Marcar archivos como **staged**.
4) Crear revisiones con **commit**.
5) Crear, descartar o fusionar **ramas** de desarrollo.

Primeros pasos

Es recomendable tener conocimientos básico de **Bash** para utilizar git en modo texto. (**optativo**: revisar clases de Bash compartidas)

- **Instalar** git en la computadora donde lo utilizarán.
- Verificar la instalación (e.g. **`git --version`**)
- **Configurar** sus datos personales (nombre y correo) que identificarán sus cambios/revisiones:

```
> git config --global user.name "Nombre Apellido"
> git config --global user.email "sucorreo@algunmail.com"
```

Esto crea un archivo **.gitconfig** en el **$HOME** con sus datos.

En general, el uso de git sigue la siguiente estructura:

```
> git comando <opciones>
```

Crear un repositorio

Para crear un repositorio de un directorio específico (*working directory*), es necesario inicializar git:

```
> cd ruta/al/directorio
> git init
```

Con ello se crea un directorio oculto `.git` que almacena toda la información del repositorio.

Los repositorios git llevan un registro propio de su contenido, que **no es equivalente** al contenido del *working directory*.

Realizar modificaciones

A partir de ahora se pueden realizar modificaciones al contenido del directorio:

- crear archivos,
- borrar archivos, y
- eliminar archivos.

Luego, git puede analizar el estado del *working directory* y compararlo con la última revisión mediante:

```
> git status
```

Recomendable conocer de forma básica un **editor de texto** en terminal.
(e.g. tutorial del NLHPC en este link)

Creación de una revisión

(un proceso de **tres** pasos)

Git requiere que todos los archivos con modificaciones a considerar en la nueva revisión, sean indicados explícitamente (marcados como **staged**). Por ejemplo, para un archivo nuevo README.txt:

```
> git add README.txt
```

También se puede utilizar `-a` o `--all` para marcar todos.

Luego, la revisión se registra utilizando commit:

```
> git commit
```

Automáticamente se abrirá un editor de texto con los detalles de la revisión, donde será necesario ingresar una descripción de la revisión (que también puede ser ingresado con la opción `-m`).

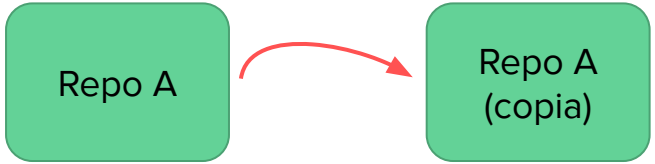
Si no hay archivos nuevos, todo este proceso se puede realizar en una sola instrucción:

```
> git commit -a -m "descripción personal"
```

Revisiones

Además del registro de cambios, cada revisión tiene: un identificador único, autor, fecha y descripción. La historia de revisiones se puede consultar con **`git log`**.

Clonación local



La clonación es el proceso de copiar un repositorio de una localización a otra.

```
> git clone ruta/a/src ruta/a/dst
```

- Cada copia clonada contiene la historia completa del repositorio. Desde ese momento, el repositorio se comportará de forma independiente a la fuente.
- Cada copia clonada recuerda la información de la fuente (**`ruta/a/src`**). Esto queda almacenada en el archivo `.git/config` del repositorio.

Es posible re-sincronizar un repositorio copiado con su fuente luego de cambios realizados (revisiones nuevas) en cualquiera de ellos utilizando: **`git push`** y **`git pull`**. No obstante, la actualización en la fuente por defecto solo admite nuevas ramas.

Más *comandos* útiles de git (desarrollo individual)

- `git switch`, `git branch`: cambiar de **rama**.
- `git diff`: visualizar diferencias.
- `git restore`: para **deshacer** cambios realizados.
- `git merge`: para fusionar ramas locales.
- `git tag`: para establecer una **etiqueta**.

Las etiquetas corresponden a nombres que se le pueden asignar a revisiones. La etiqueta HEAD siempre se asigna a la revisión más actual de la rama activa.

- `git comando -h`: verificar opciones disponibles del comando
- `git comando --help`: manual específico del comando.

Los manuales específicos también se pueden acceder con `man git-comando`, y el manual general del software: `man git`

Tema para profundizar

Investigar sobre el uso de **ramas** en git.

- ¿Para qué se utilizan las ramas en git?
- ¿Cuáles son los beneficios de utilizarlas de forma regular?