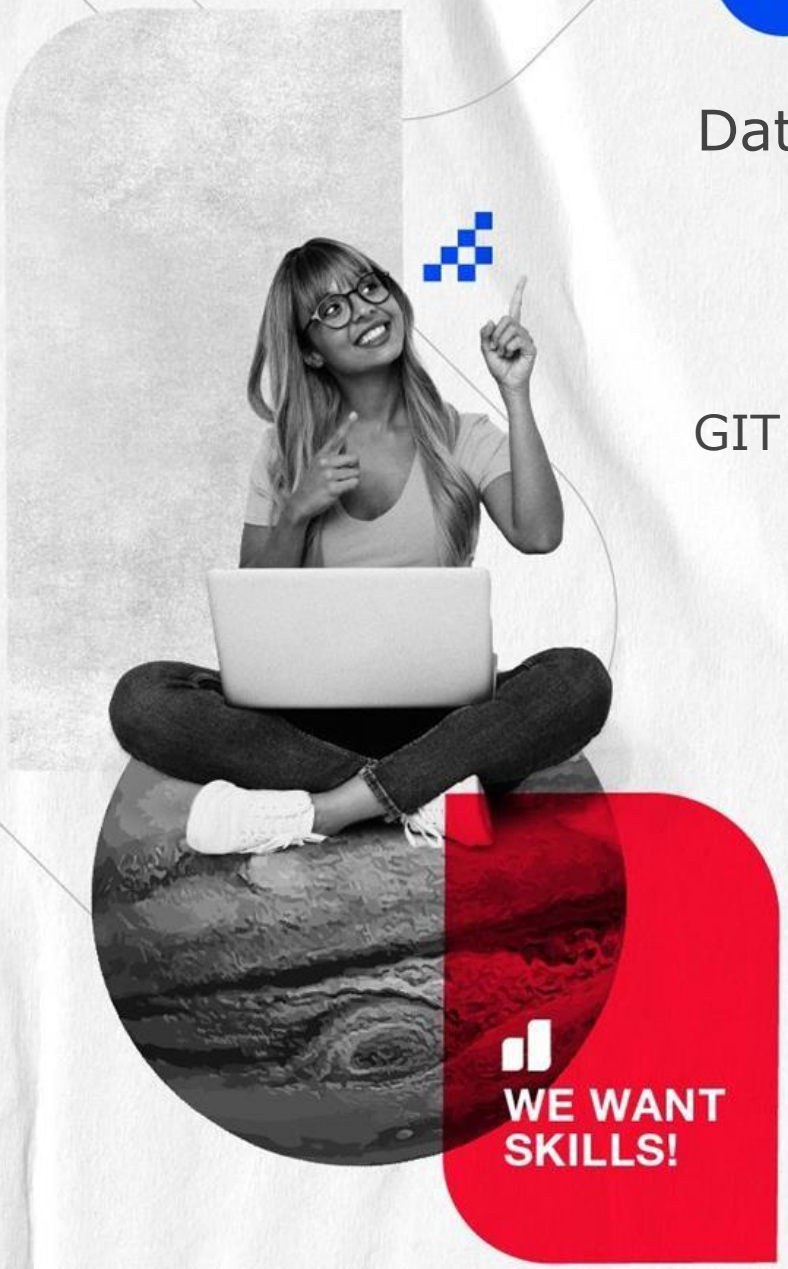
CERTIFICACIÓN UNIVERSITARIA

UNC | FCEFyN

Data Science



GIT

mundosE

Sistemas de versionamiento — ¿Qué son y cómo nos ayudan?

Los sistemas de versionamiento, son herramientas (software) que pueden gestionar cambios dentro de archivos, de forma tal, que cada versión sea un fotografía del momento, pero a su vez, puedas tener múltiples fotos y estas fotos puedan ser comparadas, unificadas y consolidadas, siempre sabiendo qué y quién realizó cada cambio. Existen sistemas de versionamiento centralizados y distribuidos.

Ventajas de los sistemas de versionamiento

- Historial completo de cambios.
- Creación de ramas y fusiones.
- Trazabilidad.

Características fundamentales

Resolución de conflictos

Es muy probable que los miembros del equipo tengan la necesidad de realizar cambios en el mismo archivo de código fuente al mismo tiempo. Un VCS monitoriza y ayuda a resolver los conflictos entre varios desarrolladores.

Revertir y deshacer los cambios en el código fuente

Al monitorizar un sistema de archivos de códigos fuente, existe la posibilidad de revertir y deshacer rápidamente a una versión estable conocida.

Copia de seguridad externa del código fuente

Se debe crear una instancia remota del VCS que se puede alojar de forma externa con un tercero de confianza y con ello, se conservará una copia del código fuente.

Entre los sistemas más populares está GIT.

¿Qué es GIT?

Es el sistema de versionamiento de código más utilizado actualmente. Es multiplataforma, por lo que puede ser ejecutado en sistemas Windows, Linux y Mac. Es de código abierto por lo que no es necesario el pago de licencias para su uso.

El uso de git principalmente es a través de comandos. Usando sólo git tenemos ya disponible el sistema de versionamiento de forma local.

Los tres estados en GIT

Git tiene tres estados principales en los que se pueden encontrar tus archivos: confirmado (committed), modificado (modified), y preparado (staged). *Confirmado* significa que los datos están almacenados de manera segura en tu base de datos local. *Modificado* significa que has modificado el archivo pero todavía no lo has confirmado a tu base de datos. *Preparado* significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

Esto nos lleva a las tres secciones principales de un proyecto de Git: el directorio de Git (Git directory), el directorio de trabajo (working directory), y el área de preparación (staging area).

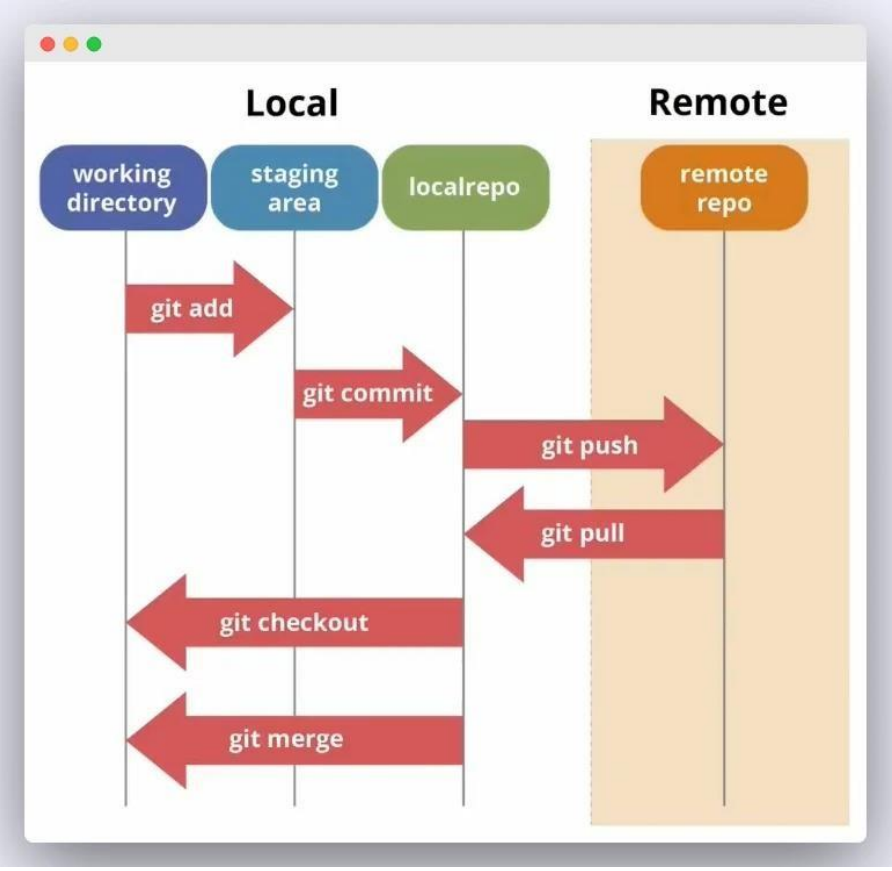
Hablemos ahora de la herramienta complementaria a GIT. Hablemos de GitHub.

¿Qué es Github?

Github es el complemento ideal para Git. Github es un plataforma que sirve como repositorio en la nube, de esta forma, además de tener nuestro sistema de versionamiento local, con *giĞhub* podemos contar de forma distribuida con nuestro código y, podemos tener desarrollo colaborativo, donde distintos desarrolladores pueden trabajar en el mismo proyecto a la vez de forma individual y luego, es posible realizar la consolidación y puesta en producción del producto.

En esencia la dupla: git + github nos permite tener un sistema de versionamiento de código local + distribuido en nube.

Arquitectura de trabajo de git-github



Descripción básica de un proceso de versionamiento con git-github

Vamos a describir le paso a paso de un proceso de versionamiento, desde el punto de vista local:

- **El estado actual de la versión.** Esto lo sabemos usando el comando git status. Allí se nos mostrará qué archivos han sido creados, modificados y eliminados. Con ello sabemos que ha pasado desde la última versión.
- **Agregar los cambios.** Esto significa que queremos crear una versión con lo que hemos cambiado desde la última versión o desde inicio del proyecto, si es la primera versión. Podemos agregar de forma individualizada archivos, carpetas o todos los cambios. Esto se realiza con el comando *giĞ add*.
- **Cerrando o confirmando versiones locales.** Cuando queremos tomar una fotografía de lo que tenemos, es decir, cerrar una versión, le indicamos a git que haga el *commiĞ*, esto se hace con el comando *giĞ commiĞ* -m "Nombre

descriptivo". De esta forma está diciendo que esos cambios son permanentes localmente. Si fuese necesario tener 2 o más versiones del código en simultáneo, debemos crear ramas, que realizan una copia de la versión actual incluyendo los últimos cambios no cerrados, esto lo realizamos con el comando *giǴ branch* "nombre rama", pero eso solo crea la foto, aún debemos indicar explícitamente que queremos trabajar en esa fotografía, para ello usamos *giǴ checkouǴ* "nombre rama"

Ahora veamos los pasos relacionados con el proceso de interacción entre mi código y github.

- **Para llevar una versión desde el ambiente local** gerenciado por git hacia la nube, ejecutamos el comando *giǴ push "desde" "para"* . Generalmente el "desde" será origen, que es la relación entre tu repositorio y la nube, aunque es posible tener en un mismo proyecto más de una relación con la nube, "para" será la rama o fotografía que vamos a subir a la nube.

- **Para traernos versiones desde la nube**, usaremos el comando *giǴ pull "desde" "para"*, este comando funciona de forma similar al comando *giǴ push* pero en sentido inverso. Es de hacer notar que si por alguna razón git detecta que nuestro código puede generar un problema que ponga el riesgo a la versión, no va a dejar realizar el proceso.

- **Todas las veces que interactuamos con la nube**, llevando o trayendo, git ejecuta un proceso interno llamado merge, que permite consolidar el código de la versión, es decir, unifica lo que tenemos en local con lo de la nube dejando una versión unificada, pero este comando también es posible ejecutarlo entre diferentes ramas, es decir, tengo 2 fotografías de mi código y quiero unificar una en otra, esto lo hacemos con el comando *giǴ merge* "desde", donde "desde" es la versión que quieres unificar con tu versión actual. Esto significa que siempre vas a estar en una versión. Desde que haces la creación del repositorio indicas una versión. ,

- **Y cómo creamos el repositorio,** lo hacemos con el comando *giǴ iniǴ*. Todos estos comandos que hemos visto se deben ejecutar en el directorio donde se encuentra lo que queremos versionar.

Autenticándonos en Github

Para poder conectar nuestro repositorio local a un repositorio de nube, debemos primero tener una cuenta en github y luego, por seguridad debemos autenticarnos en Github.

Existen 2 formas para realizar esto. Antiguamente solo era necesario indicar nuestro nombre de usuario y contraseña, que definimos al crear la cuenta en el portal de Github, pero con el tiempo se han mejorado la seguridad y es necesario realizar los siguientes pasos:

- **Crear llaves de autenticación SSH.** Esto significa que debemos crear una identificación única de nuestro computador y luego, esa identificación la registramos en github, con ello ya el sistema realizará la validación.
- **Creación de tokens de acceso personales**, que trabajan o funcionan como contraseñas pero con un mayor grado de seguridad