INTRODUCCIÓN A LA PROGRAMACIÓN

CONTROL DE VERSIONES

Dra. Ana Julia Velez Rueda

SBG UNQ - CONICET

Control de Versiones

Ana Julia Velez Rueda
SBG UNQ - CONICET

Los sistemas de control de versiones comienzan con una versión base del documento y luego registran los cambios que realiza en cada paso del camino.

Se podría pensar como un video: puede retroceder para comenzar en el documento inicial y reproducir cada estado o cambio que realizó, llegando finalmente a su versión más reciente.

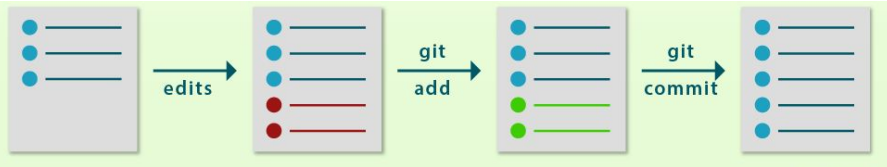
¿Qué es un sistema de control de versiones?

Ana Julia Velez Rueda
SBG UNQ - CONICET

Un **sistema de control de versiones** es una herramienta que realiza un seguimiento de los cambios de un documento o directorio de forma automática, creando efectivamente diferentes versiones de nuestros archivos.

Cada registro de estos cambios se denomina commit y mantiene metadatos útiles sobre ellos.

El historial completo de commits para un proyecto en particular y sus metadatos forman un repositorio.



Source: National Ecological Observatory Network (NEON)

¿Cómo funciona?

Ana Julia Velez Rueda
SBG UNQ - CONICET

Siguiendo con la analogía del video, podemos pensar cada **commit** como un fotograma en nuestro video, siendo este el historial completo de cambios de un archivo o directorio.

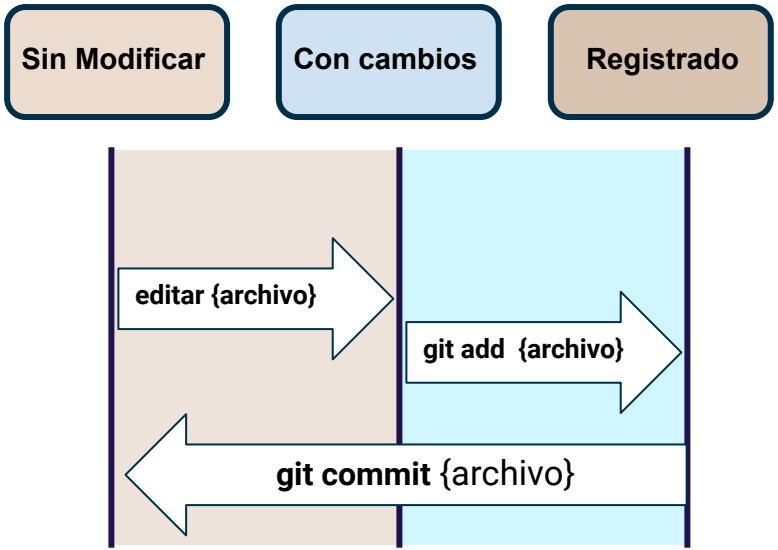
Cada **commit** funciona como un “paquete” de cambios realizados, que se pueden ir agregando al **stage** (estado intermedio con cambios) mediante el comando ***git add***.

Estos cambios se gestionan como una unidad, al generar un **commit**, y quedan registrados en una “foto” al hacer ***git commit***.

Es muy importante especificar los cambios realizados en cada commit, esto nos ayudará a rastrear cualquier cambio al querer volver atrás.

¿Cómo funciona?

Ana Julia Velez Rueda
SBG UNQ - CONICET



Al hacer **commit** se obtiene un ID del mismo, que luego puede usarse en otros comandos para referenciar este bloque de cambios.

Git al infinito y más allá...

Ana Julia Velez Rueda
SBG UNQ - CONICET

Git trabaja con un repositorio local que está en tu computadora, donde vas a ir agregando tus commits y uno remoto (en la nube) en el cual podes subir tus commits, compartirlos con alguien más o bajarte los commits que haya subido alguien.

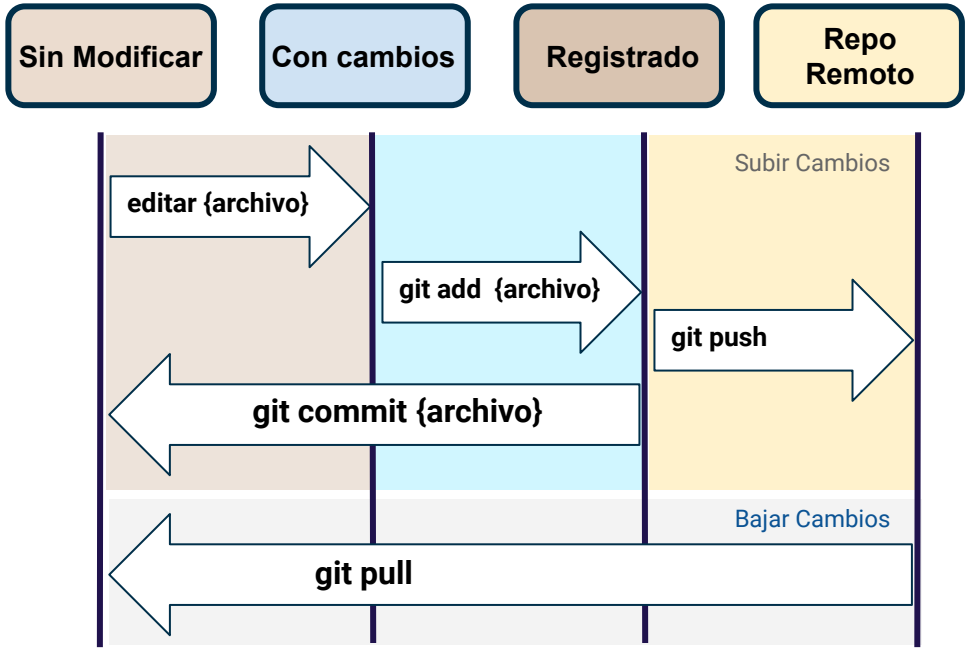
Existen varios varios servicios para almacenar repositorios remotamente:

- **Github** (https://github.com)
- **Bitbucket** (https://bitbucket.com)
- **Gitlab** (https://gitlab.com/)

Para usarlos deberás registrarte y crear una cuenta.

El más allá...

Ana Julia Velez Rueda
SBG UNQ - CONICET



Podemos hacer una sincronización saliente del repo local al remoto (*origin*), haciendo ***git push***. Este comando envía los commits generados localmente que no se hayan enviado anteriormente.

Podemos también descargar los cambios del repositorio remoto utilizando el comando ***git pull***.

¿Cómo trabajar?

1) Necesitaremos un repositorio local y uno remoto. Desde el local haremos cambios, que luego vamos a agregar al repositorio remoto.

2) La primera vez que usas GIT vas a tener que configurar tu nombre completo y tu email con los siguientes comandos:

- git config --global user.name "TU NOMBRE"
- git config --global user.email "TU DIRECCION DE EMAIL"

3) Una vez creada la cuenta y un repositorio en alguno de estos servicios, tenés que bajarte la información del repositorio remoto a tu computadora

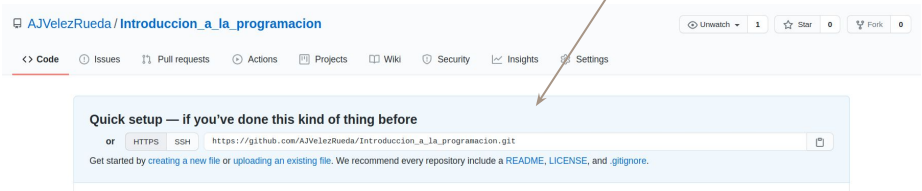
¡Hora de programar!

Ana Julia Velez Rueda
SBG UNQ - CONICET

Hasta aquí no parece tan complejo. ¡Es hora de que pruebes vos!

👉 **RETO I: Generá tu usuario en GitHub (https://github.com/) y armá el repositorio de la materia.**

👉 **RETO III: Cloná tu repositorio de forma local con el comando *git clone <url repositorio remoto>* para poder comenzar a trabajar.**



¡¡20 MINUTOS!!

Ana Julia Velez Rueda
SBG UNQ - CONICET

Conflictos

Ana Julia Velez Rueda
SBG UNQ - CONICET

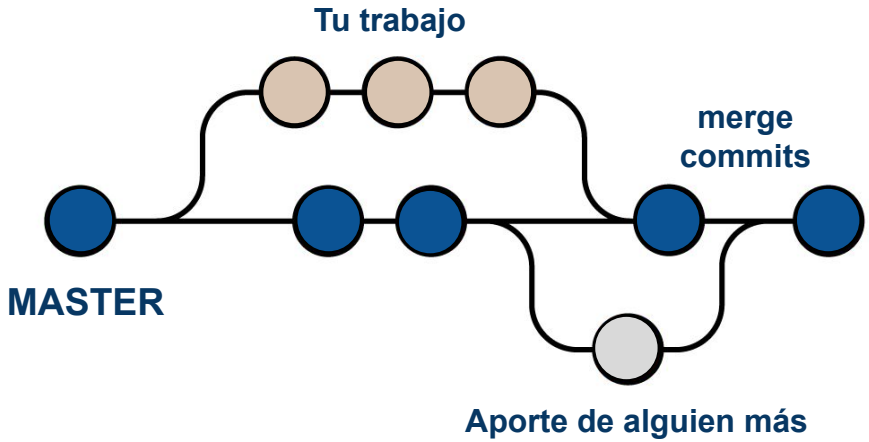
Suele suceder cuando trabajamos en equipo o desde más de una computadora sobre el mismo proyecto, que se sobreescriba o modifique una misma porción de código dentro de un archivo.

En general sucede cuando hemos introducido cambios (commits) localmente que sobrescriben los cambios que se encuentran remotos y git no sabe qué versión es la correcta.

Estos conflictos pueden ser solucionados o tomando la versión remota o bien generando un nuevo ***commit*** de forma local y subiéndolos al repositorio remoto.

Nos vamos por las ramas

Ana Julia Velez Rueda
SBG UNQ - CONICET



El sistema de “**branches**” o **ramas** nos permite desarrollar en paralelo diferentes funcionalidades (cada una en un branch distinto), sin mezclar el código, hasta tener una versión final y estable. Luego, cuando se pretenden integrar los cambios de cada branch para obtener un producto más completo, se fusionan mediante un **merge**.

Ver más en: shorturl.at/jzC17

Nos vamos por las ramas

Ana Julia Velez Rueda
SBG UNQ - CONICET

Podemos movernos entre las diferentes ramas de nuestro proyecto mediante la sentencia ***git checkout <nombreBranch>*** , de este modo podemos trabajar en cambios que tienen como base el commit al que hace referencia dicho branch

Además con el parámetro **-b** puede crearse un nuevo branch antes de moverse al mismo. Por ejemplo para crear el branch dev y comenzar a trabajar en el mismo, debemos hacer:

git checkout -b dev

Y podemos corroborar en qué rama nos encontramos haciendo:

git branch --show-current

¡HORA DE LA PRÁCTICA!

Ana Julia Velez Rueda
SBG UNQ - CONICET

¡Hora de programar!

Ana Julia Velez Rueda
SBG UNQ - CONICET

Hasta aquí no parece tan complejo. ¡Es hora de que pruebes vos!

👉 **RETO IV: Agregá a tu repositorio local una carpeta para cada trabajo práctico o tema que hayamos visto hasta aquí en la materia. Registrá y subí los cambios a tu repositorio remoto.**

👉 **RETO IV: Agregá a tu repositorio remoto como colaboradora a l@s docentes de la materia, cuyos usuarios son: AJVelezRueda y BenitezG**