The background is a dark, abstract composition of numerous thin, parallel lines and small dots. These elements are arranged to create a strong sense of three-dimensional perspective, receding from the bottom left towards the top right. The lines and dots are color-coded in a gradient: bright green and yellow on the left, transitioning through blue and purple in the center, and ending in red and orange on the right. The overall effect is reminiscent of a digital or data visualization space.

# Tema 2.5

## Introducción a GIT

Entornos de desarrollo

# ¿Qué es GIT?



- Git es un sistema de control de versiones que se utiliza para gestionar el código.
- Se utiliza para registrar los cambios de las diferentes revisiones del código y así permitir que tanto un desarrollador individual como un equipo puedan trabajar juntos en un proyecto

# Repositorio

- 
- Un repositorio es un espacio centralizado donde se almacena, organiza, mantiene y difunde información digital, habitualmente archivos informáticos, que pueden contener trabajos científicos, conjuntos de datos o software .
  - Estos repositorios están almacenados en servidores, no en ordenadores locales o en un disco duro que tengamos.
  - Normalmente utilizan sistemas de control de versiones.

# Repositorios



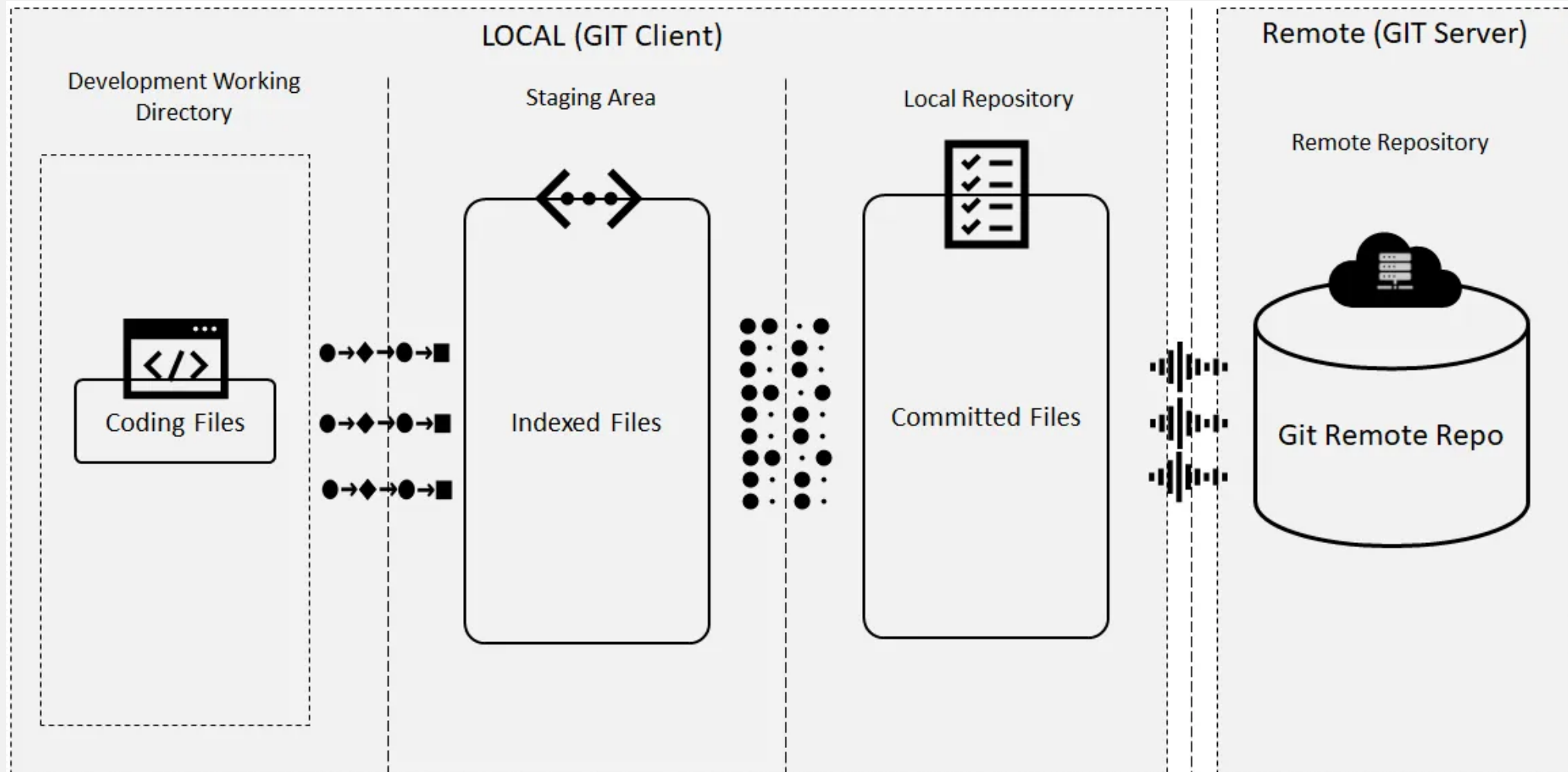
- Un repositorio es un espacio centralizado donde se almacena, organiza, mantiene y difunde información digital, habitualmente archivos informáticos, que pueden contener trabajos científicos, conjuntos de datos o software .
- Estos repositorios están almacenados en servidores, no en ordenadores locales o en un disco duro que tengamos.

# Commits

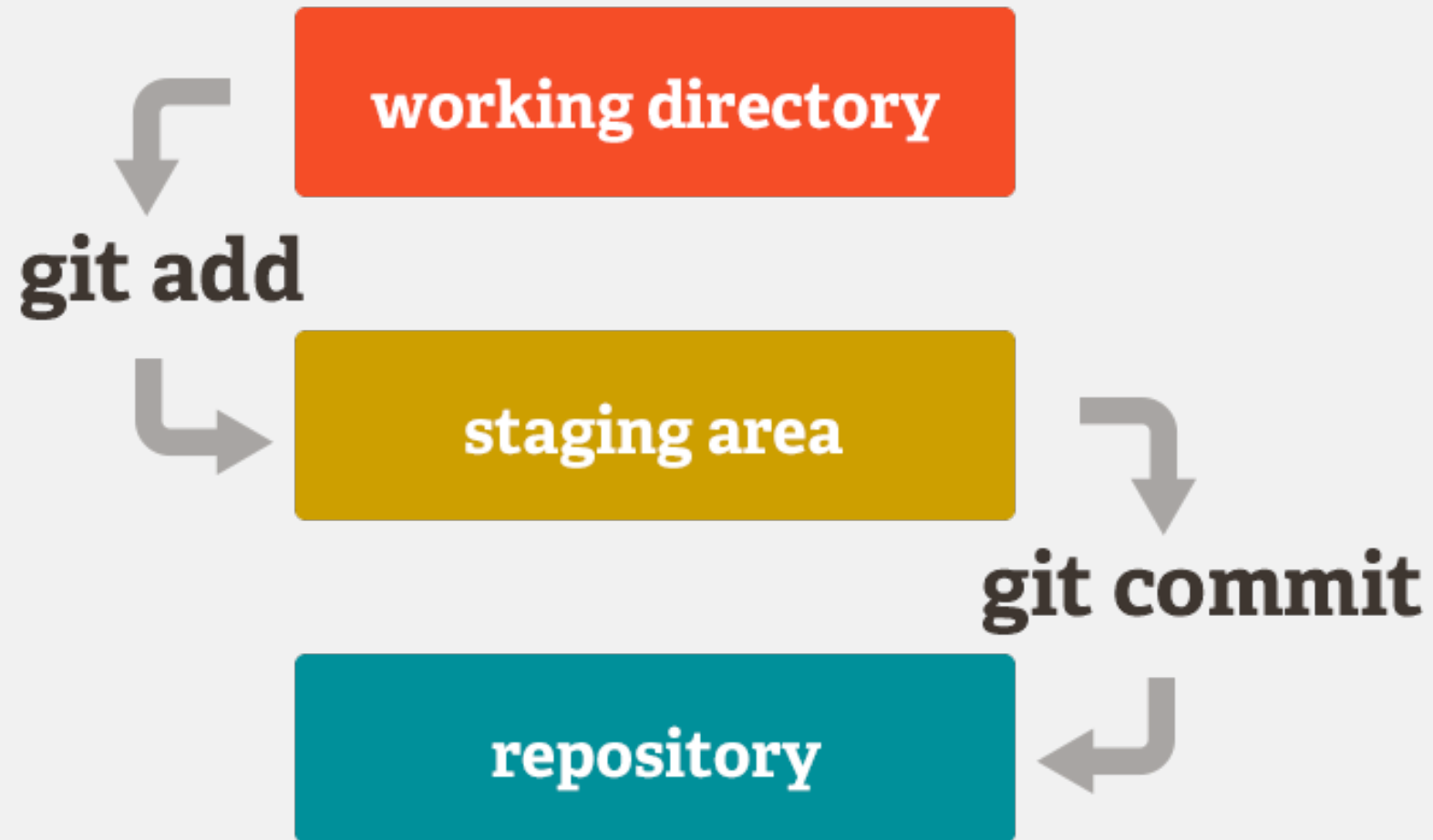
---

- Un commit es una instantánea de los cambios preparados en ese momento del proyecto.
- Las instantáneas confirmadas pueden considerarse como versiones "seguras" de un proyecto: Git no las cambiará nunca a no ser que se haga expresamente.

# Flujo de trabajo

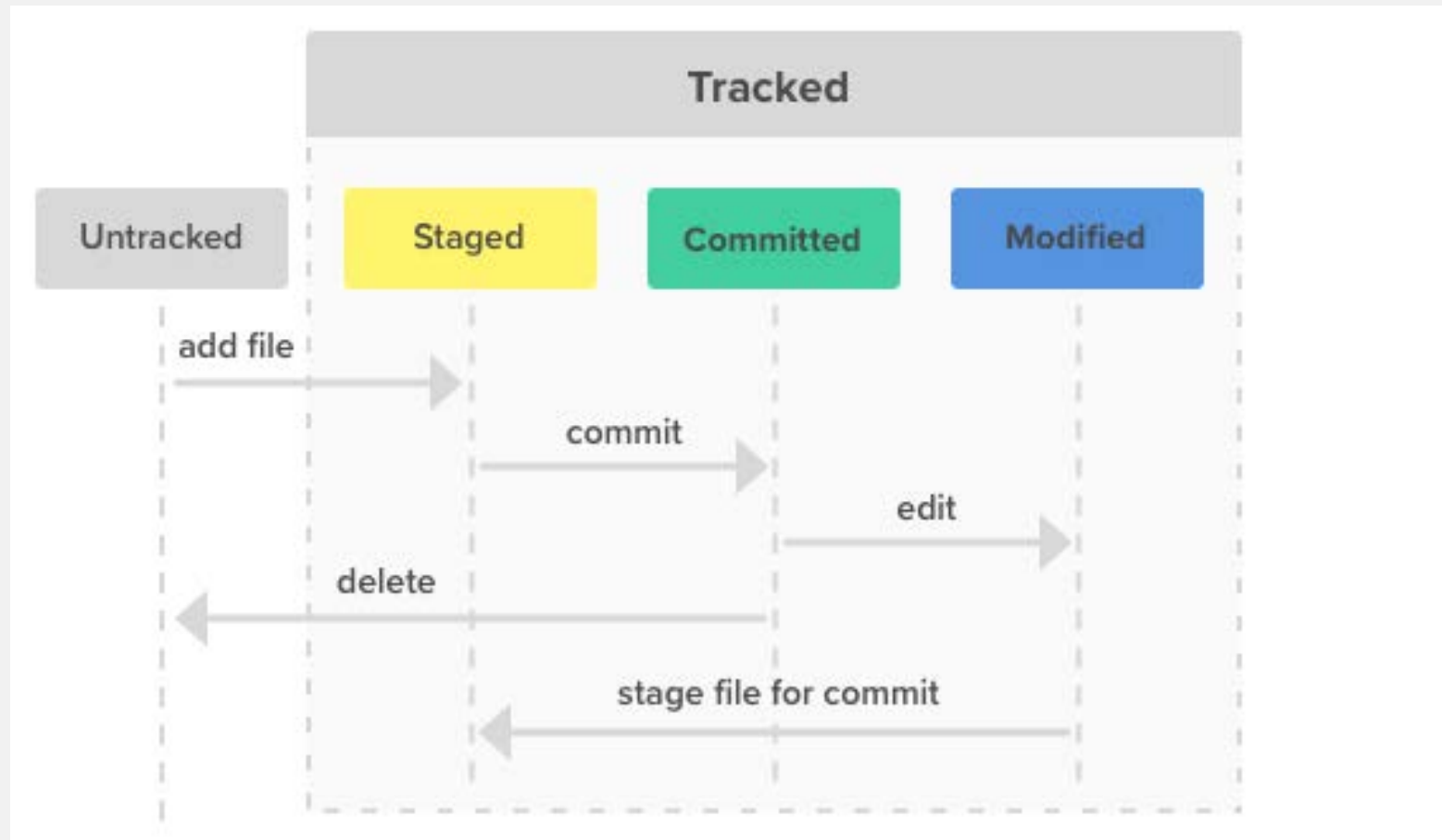


# Flujo de trabajo



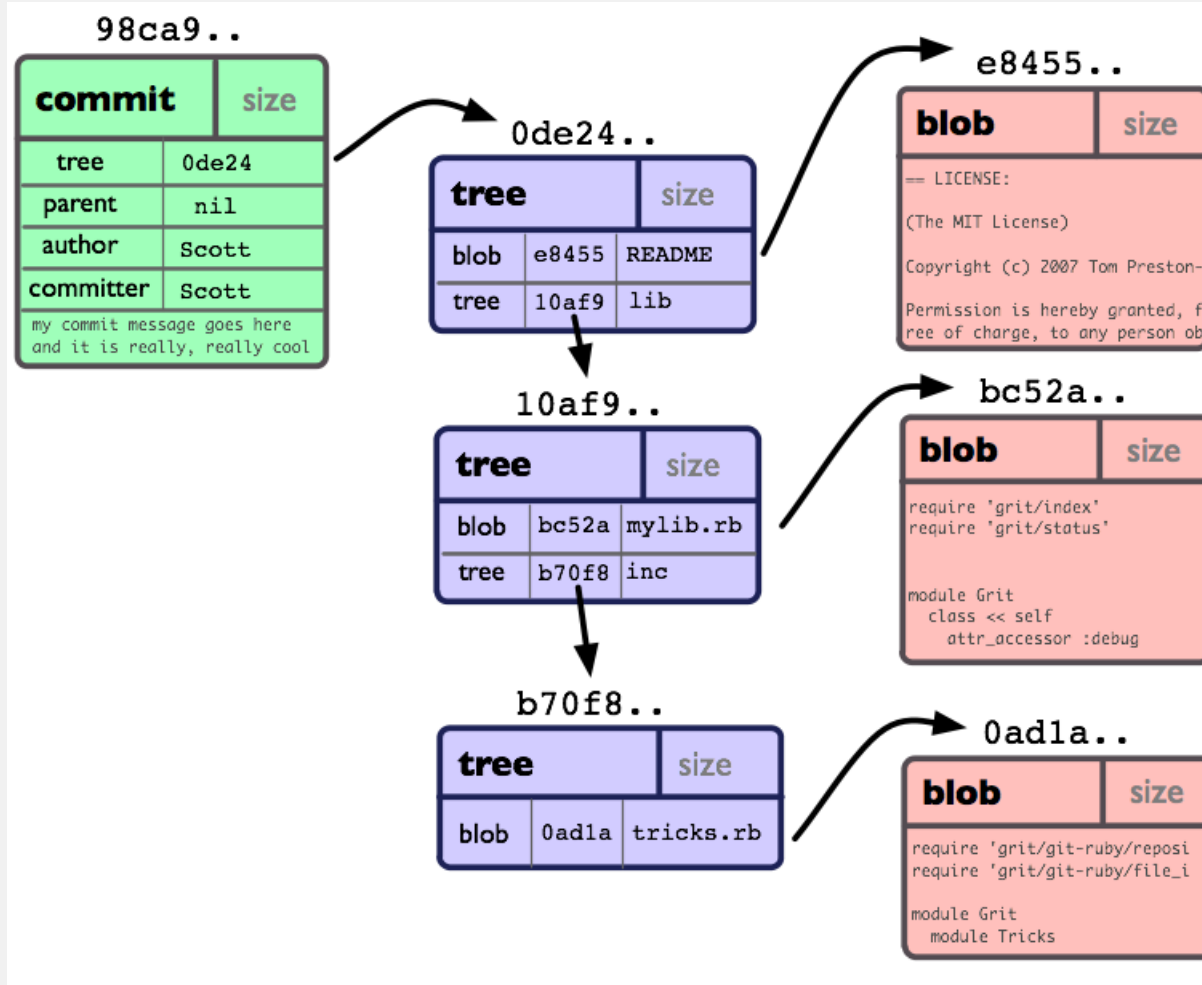


# Flujo de trabajo





# Flujo de trabajo

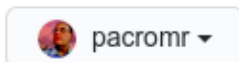


# Crear repositorio remoto

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*



Repository name \*



Great repository names xmldemo is available. able. Need inspiration? How about [turbo-lamp](#)?

Description (optional)



**Public**

Anyone on the internet can see this repository. You choose who can commit.



**Private**

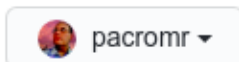
You choose who can see and commit to this repository.

# Crear repositorio remoto

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*



Repository name \*

/ xmldemo ✓

Great repository names xmldemo is available. able. Need inspiration? How about [turbo-lamp](#)?

Description (optional)



**Public**

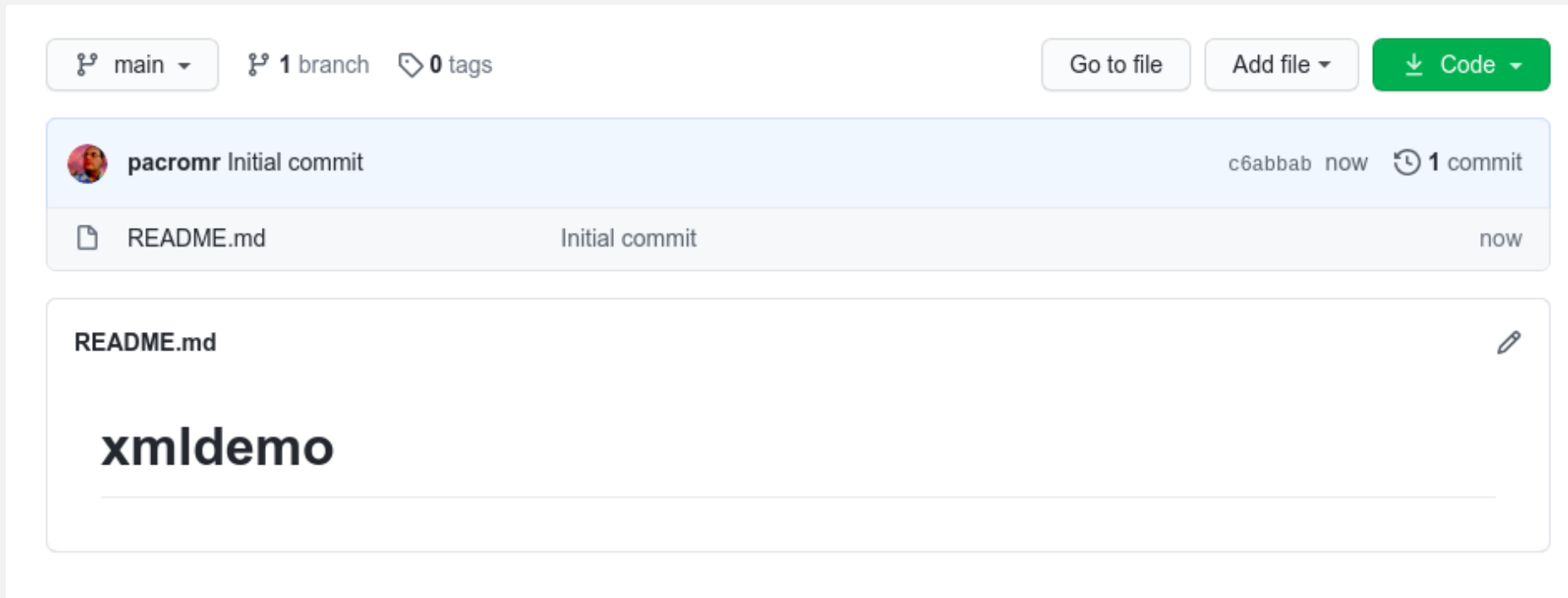
Anyone on the internet can see this repository. You choose who can commit.



**Private**

You choose who can see and commit to this repository.


# Crear repositorio remoto





The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with a branch selector set to 'main', indicating 1 branch and 0 tags. Action buttons include 'Go to file', 'Add file', and a green 'Code' button. Below this, a commit history table shows a single commit by user 'pacromr' with the message 'Initial commit', hash 'c6abbab', and timestamp 'now'. The file list below the commit shows 'README.md' as the only file. The content area for 'README.md' displays the text 'xmldemo' in a large, bold font, with an edit icon in the top right corner.

main 1 branch 0 tags

Go to file Add file Code


 <b>pacromr</b>	Initial commit	c6abbab	now	🕒 1 commit
--	----------------	---------	-----	------------

 README.md	Initial commit	now
---	----------------	-----

README.md 

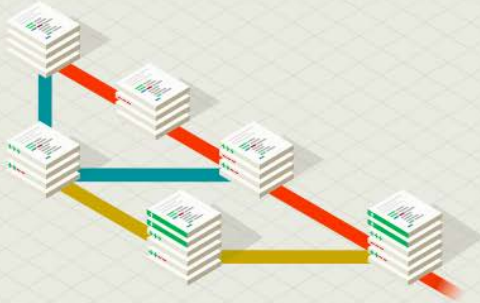
## xmldemo


# Consola de GIT

 **git** --everything-is-local

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.


Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.






### About

The advantages of Git compared to other source control systems.




### Documentation

Command reference pages, Pro Git book content, videos and other material.




### Downloads

GUI clients and binary releases for all major platforms.




### Community


Get involved! Bug reporting, mailing list, chat, development and more.





**Pro Git** by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).




Latest source Release  
**2.30.0**  
[Release Notes \(2020-12-27\)](#)  
[Downloads for Linux](#)

 [Linux GUIs](#)

 [Tarballs](#)

 [Mac Build](#)

 [Source Code](#)

# Comandos básicos

---

- **git status:** Comprueba el estado de un repositorio.
- **git add:** Sirve para agregar los nuevos archivos al repositorio, llevando así un control sobre ellos.
- **git commit:** Se utiliza para agregar al repositorio los cambios en los archivos que han sido modificados.
- **git stash:** Deshacer los cambios no guardados.

# Comandos básicos



- **git push:** Se utiliza para subir los archivos al repositorio origen.
- **git pull:** Se utiliza para descargar los archivos del repositorio junto con los cambios que otros usuarios hayan realizado en él.
- **Git diff:** Se utiliza para visualizar los cambios entre repositorios.



# Comandos básicos

---

- **git config** --global user.name "Nombre"
- **git config** --global user.email [usuario@dominio.tld](mailto:usuario@dominio.tld)
- **git init**: Se utiliza para crear un nuevo repositorio vacío.
- **git clone**: Se utilizar para clonar un repositorio remoto en local.
- **git reset**: elimina commits ya realizados

# Clonar repositorio remoto

---

```
paco@paco-mountain:~/xmldemo$ git clone https://github.com/pacromr/xmldemo.git
Clonando en 'xmldemo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Desempaquetando objetos: 100% (3/3), 586 bytes | 146.00 KiB/s, listo.
paco@paco-mountain:~/xmldemo$ ls
xmldemo
paco@paco-mountain:~/xmldemo$ cd xmldemo
paco@paco-mountain:~/xmldemo/xmldemo$ ls
README.md
paco@paco-mountain:~/xmldemo/xmldemo$
```

# Comprobar el estado del entorno local

---

```
paco@paco-mountain:~/xmldemo/xmldemo$ git status
En la rama main
Tu rama está actualizada con 'origin/main'.

nada para hacer commit, el árbol de trabajo está limpio
paco@paco-mountain:~/xmldemo/xmldemo$ nano readme.md
paco@paco-mountain:~/xmldemo/xmldemo$ git status
En la rama main
Tu rama está actualizada con 'origin/main'.

Archivos sin seguimiento:
  (usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
    readme.md

no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa
"git add" para hacerles seguimiento)
```

# Preparar los cambios a subir

```
paco@paco-mountain:~/xmldemo/xmldemo$ git add readme.md
paco@paco-mountain:~/xmldemo/xmldemo$ git status
En la rama main
Tu rama está actualizada con 'origin/main'.

Cambios a ser confirmados:
  (usa "git restore --staged <archivo>..." para sacar del área de stage)
    nuevo archivo:  readme.md

paco@paco-mountain:~/xmldemo/xmldemo$ git commit
[main ded9e8b] First change
 1 file changed, 1 insertion(+)
 create mode 100644 readme.md
paco@paco-mountain:~/xmldemo/xmldemo$ git status
En la rama main
Tu rama está adelantada a 'origin/main' por 1 commit.
  (usa "git push" para publicar tus commits locales)

nada para hacer commit, el árbol de trabajo está limpio
```

# Repositorio remoto

---

- ***origin*** es simplemente el nombre predeterminado que recibe el repositorio remoto principal contra el que trabajamos.
- Cuando clonamos un repositorio por primera vez desde GitHub o cualquier otro sistema remoto, el nombre que se le da a ese repositorio "maestro" es precisamente *origin*.

# Subir cambios al repositorio remoto

---

```
paco@paco-mountain:~/xmldemo/xmldemo$ git push origin main
Username for 'https://github.com': pacromr
Password for 'https://pacromr@github.com':
Enumerando objetos: 4, listo.
Contando objetos: 100% (4/4), listo.
Compresión delta usando hasta 8 hilos
Comprimiendo objetos: 100% (2/2), listo.
Escribiendo objetos: 100% (3/3), 283 bytes | 283.00 KiB/s, listo.
Total 3 (delta 0), reusado 0 (delta 0)
To https://github.com/pacromr/xmldemo.git
   c6abbab..ded9e8b  main -> main
```

# Cambios en el remoto antes de subir...

```
paco@paco-mountain:~/repos/Tarea1-XML$ git push origin
FRUsername for 'https://github.com' FRomero999
Password for 'https://FRomero999@github.com':
To https://github.com/FRomero999/Tarea1-XML.git
 ! [rejected]          main -> main (fetch first)
error: falló el push de algunas referencias a 'https://github.com/FRomero999/Tarea1-XML.git'
ayuda: Actualizaciones fueron rechazadas porque el remoto contiene trabajo que
ayuda: no existe localmente. Esto es causado usualmente por otro repositorio
ayuda: realizando push a la misma ref. Quizás quiera integrar primero los cambios
ayuda: remotos (ej. 'git pull ...') antes de volver a hacer push.
ayuda: Vea 'Notes about fast-forwards' en 'git push --help' para detalles.
```



# Cambios en el remoto antes de subir...

—

Antes de actualizar los cambios locales en el repositorio remoto hay que comprobar si hay cambios que no se hayan descargado aún:

- **git fetch origin** --> descargamos el repositorio
- **git diff origin** --> vemos los cambios que hay entre el local y el remoto
- **git pull** --> aceptamos los cambios que se han descargado

# Comprobar historial de cambios

—

History for [xmldemo](#) / [readme.md](#)

🔗 Commits on Jan 9, 2021

**First change**



**paco** authored and **paco** committed 12 minutes ago



[ded9e8b](#)



[Newer](#)

[Older](#)

# Actualizar repositorio actual con el remoto

```
paco@paco-mountain:~/xmldemo/xmldemo$ git pull origin main
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Desempaquetando objetos: 100% (3/3), 918 bytes | 918.00 KiB/s, listo.
Desde https://github.com/pacromr/xmldemo
* branch                main          -> FETCH_HEAD
   ded9e8b..214242c      main          -> origin/main
Actualizando ded9e8b..214242c
Fast-forward
 sample.svg | 8 ++++++++
1 file changed, 8 insertions(+)
create mode 100644 sample.svg
```

# Cambios locales y remotos

---

```
paco@paco-mountain:~/xmldemo/xmldemo$ git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Desempaquetando objetos: 100% (3/3), 742 bytes | 185.00 KiB/s, listo.
Desde https://github.com/pacromr/xmldemo
* branch          main          -> FETCH_HEAD
  214242c..2552ba9  main          -> origin/main
Actualizando 214242c..2552ba9
error: Los cambios locales de los siguientes archivos serán sobrescritos al fusionar:
      readme.md
Por favor, confirma tus cambios o agrédalos antes de fusionar.
Abortando
```

# Resolución de conflictos

---

```
paco@paco-mountain:~/xmldemo/xmldemo$ git add readme.md
paco@paco-mountain:~/xmldemo/xmldemo$ git commit -m "cambio local"
[main 25febd2] cambio local
 1 file changed, 1 insertion(+)
paco@paco-mountain:~/xmldemo/xmldemo$ git pull origin main
Desde https://github.com/pacromr/xmldemo
* branch          main      -> FETCH_HEAD
Auto-fusionando readme.md
CONFLICTO (contenido): Conflicto de fusión en readme.md
Fusión automática falló; arregle los conflictos y luego realice un commit con el
resultado.
paco@paco-mountain:~/xmldemo/xmldemo$
```

# Resolución de conflictos



```
GNU nano 4.8                readme.md
Primer commit
<<<<<< HEAD
segundo
=====
Segunda edición
desde Github
directamente
>>>>>> 2552ba917f9e1bab3035f01bd5a03ea38ecad71a
[ Cancelado ]
^G Ver ayuda  ^O Guardar    ^W Buscar     ^K Cortar Text ^J Justificar
^X Salir      ^R Leer fich. ^\ Reemplazar ^U Pegar       ^T Ortografía
```

Una vez resuelto, hay que hacer un add y un commit para que los cambios formen parte del repositorio.

# Colaboradores del repositorio

FRomero999 / Tarea1-XML

Unwatch 1 Star 0

<> Code ⓘ Issues 🔗 Pull requests ▶ Actions 📁 Projects 📖 Wiki 🛡 Security 📈 Insights ⚙ Settings

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Actions

Environments

Secrets

Moderation settings

Who has access

PUBLIC REPOSITORY

👁

This repository is public and visible to anyone.


Manage

DIRECT ACCESS

🔑

0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access



You haven't invited any collaborators yet

Invite a collaborator



# Ramas

- 
- Una rama es una versión paralela a la principal en la que se suelen agregar nuevas funcionalidades, manteniendo así la estabilidad de todo el repositorio.
  - Internamente, una rama no es más que un puntero hacia un commit concreto.
  - Son muy útiles si se quieren añadir nuevas funcionalidades al proyecto sin que interfieran con lo desarrollado hasta ahora.

# Ramas principales

---

- **Master/main:** Es la rama principal del repositorio. Normalmente es la que está activa en producción.
- **HEAD:** Se refiere al último commit de la rama activa.

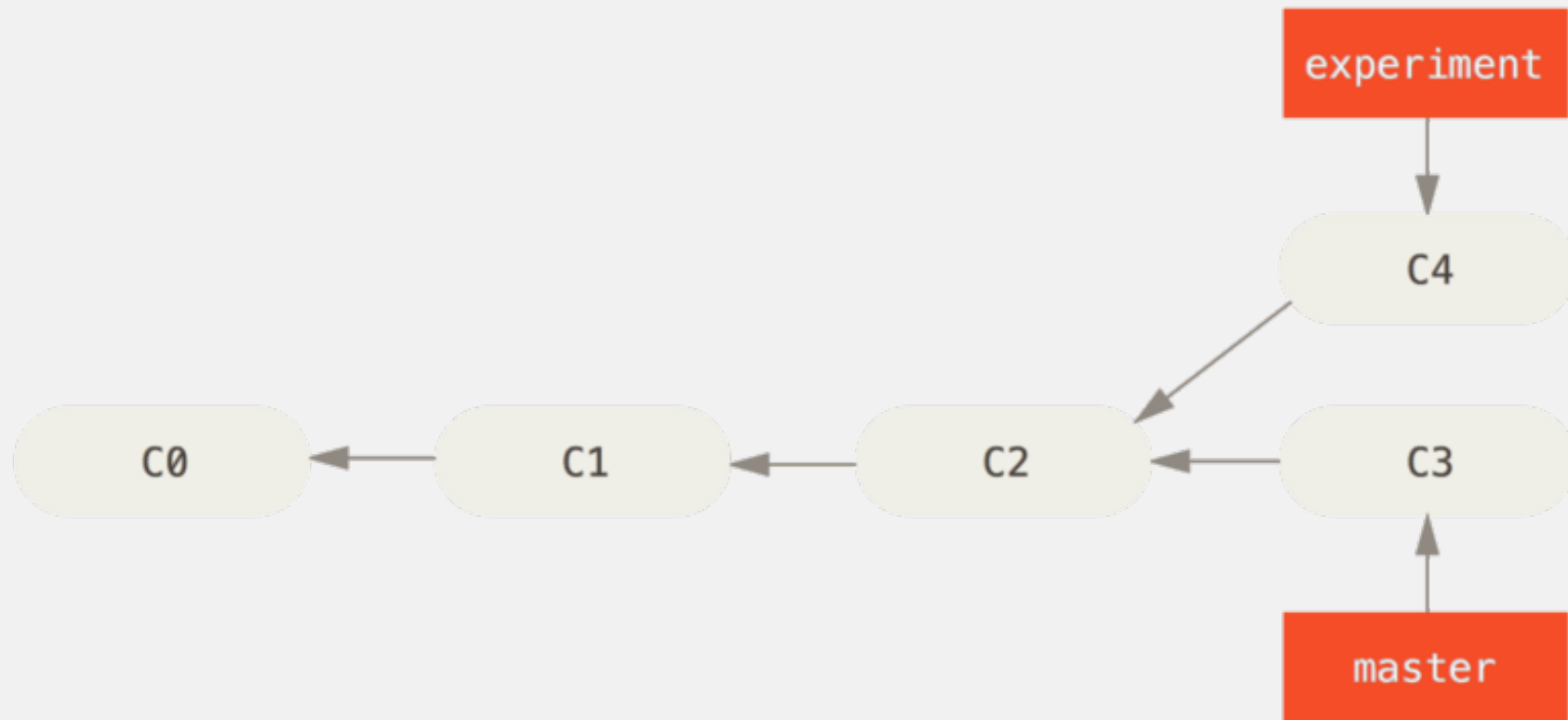
# Creación de ramas

---

**git branch <rama>** crea una nueva rama con el nombre <rama> en el repositorio a partir del último commit.

Al crear una rama a partir de un commit, el flujo de commits se bifurca en dos de manera que se pueden desarrollar dos versiones del proyecto en paralelo.

# Creación de ramas



# Trabajar con ramas

---

- **git branch:** muestra las ramas activas de un repositorio indicando con \* la rama activa en ese momento.  
`git log --graph --oneline`
- **git checkout <rama>:** actualiza los ficheros del directorio de trabajo a la última versión del repositorio correspondiente a la rama <rama>, y la activa, es decir, HEAD pasa a apuntar al último commit de esta rama.

# Trabajar con ramas

---

- **git branch:** muestra las ramas activas de un repositorio indicando con \* la rama activa en ese momento.  
`git log --graph --oneline`
- **git checkout <rama>:** actualiza los ficheros del directorio de trabajo a la última versión del repositorio correspondiente a la rama <rama>, y la activa, es decir, HEAD pasa a apuntar al último commit de esta rama.

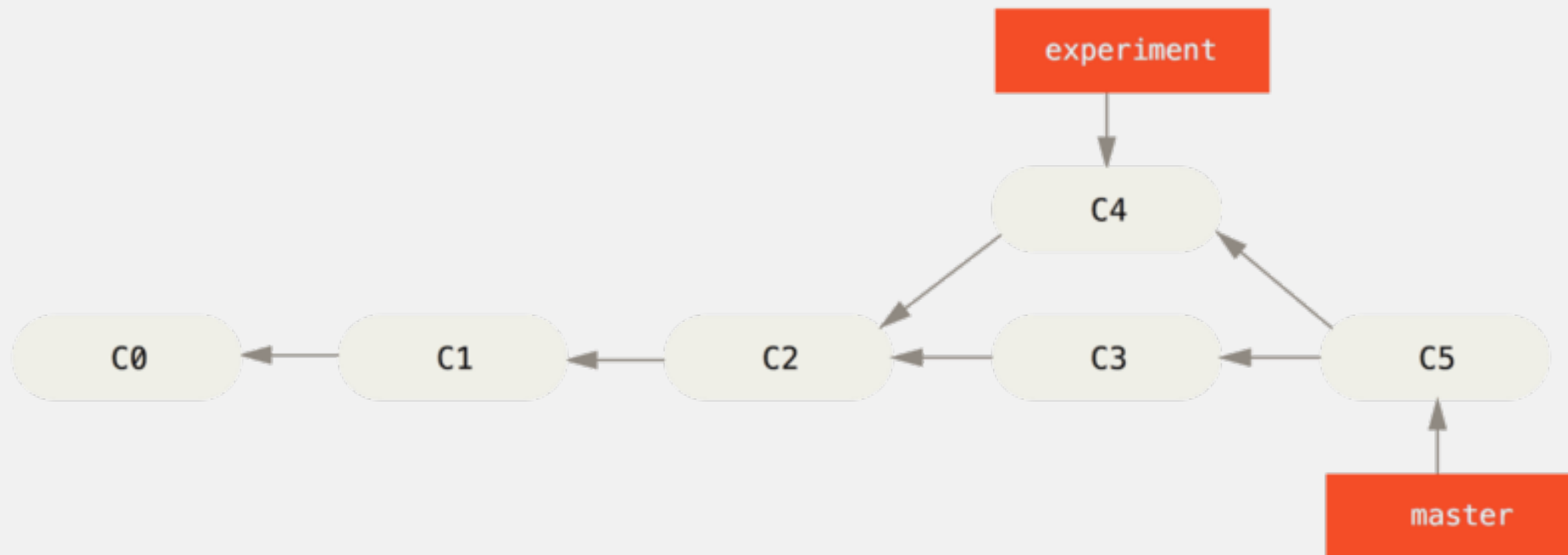
# Fusionar ramas

—

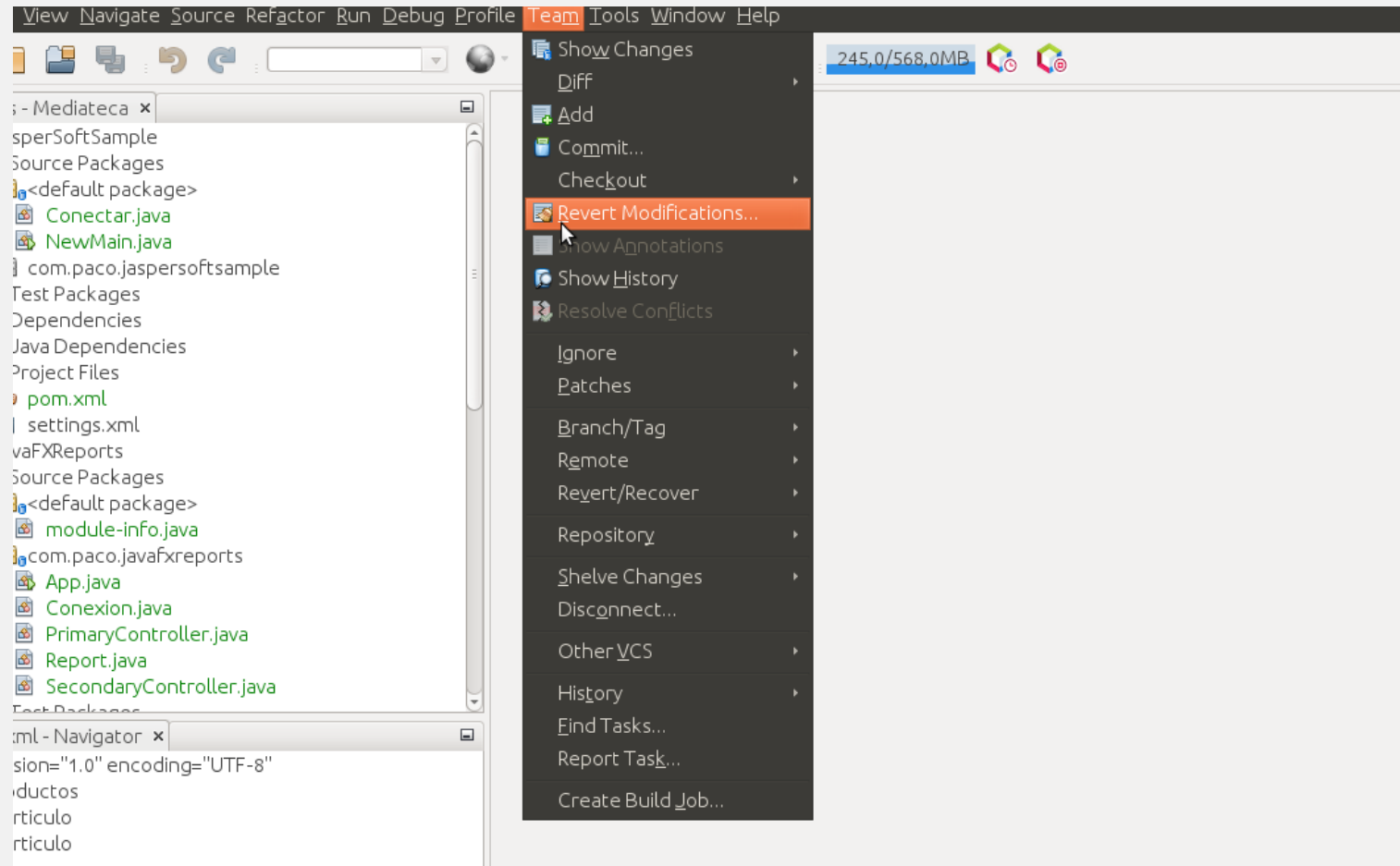
- **git merge <rama>** integra los cambios de la rama <rama> en la rama actual a la que apunta HEAD.
- Puede haber conflictos a la hora de fusionarlos, que se deben resolver de forma manual



# Fusionar ramas



# Configuración de NetBeans con Git



# Ejercicio práctico

---

**Crear un proyecto de Java de forma colaborativa entre un grupo de 2 alumnos.**

1. Se creará el repositorio remoto en GitHub
2. Se realizarán, al menos, 2 aportaciones por cada miembro del equipo en el repositorio remoto
3. Se comprobará que el funcionamiento es el correcto.