INTRO A GIT JCYL 20’

¿Qué es Control de Versiones?

Contenido

Historial completo

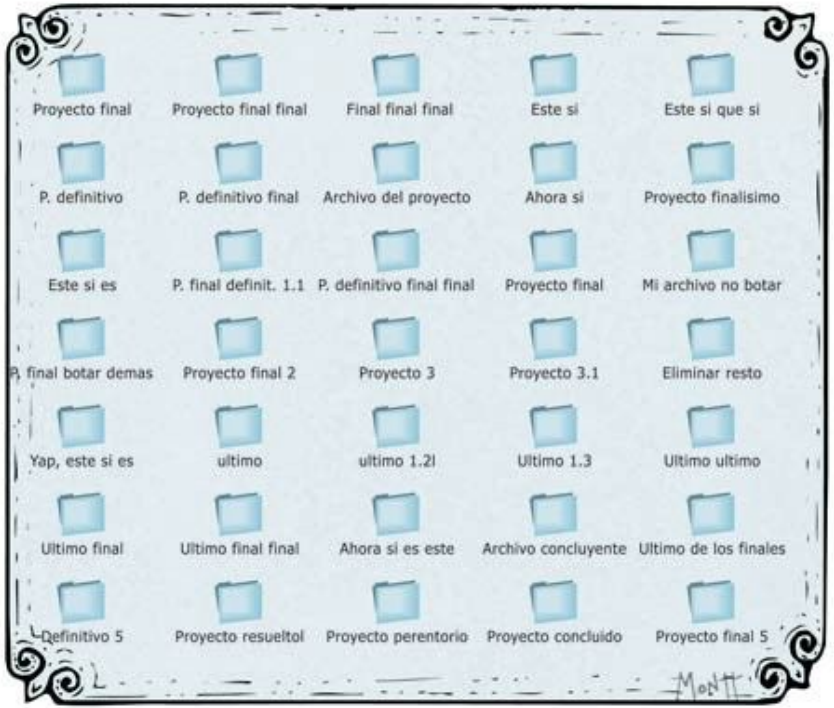
Trabajo en equipo

Herramienta colaborativa

Agilidad

Gestión de cambios, tests...

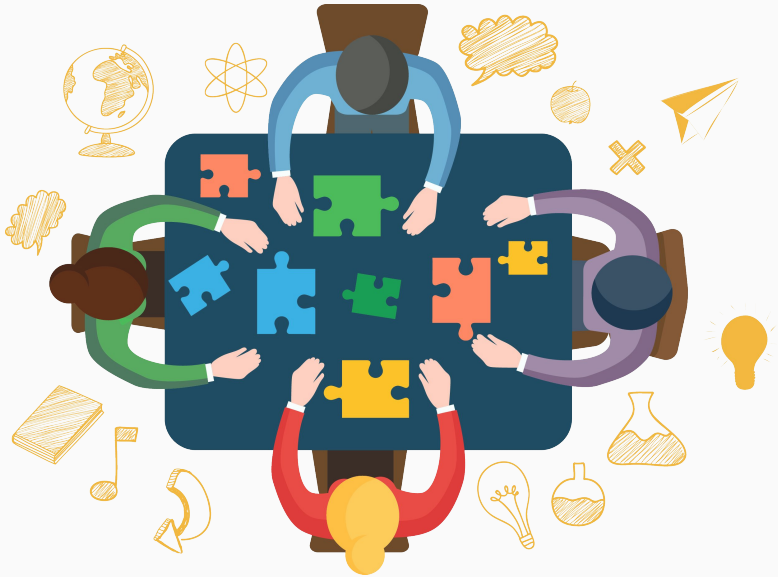
¿Por qué usar control de versiones?



Un proyecto para controlarlos a todos

Registro de cambios (deshacer, rehacer, comparar..)

¿Por qué usar control de versiones?



Indispensable para trabajo en **equipo**:

- **Método** de trabajo
- **Historial de cambios**

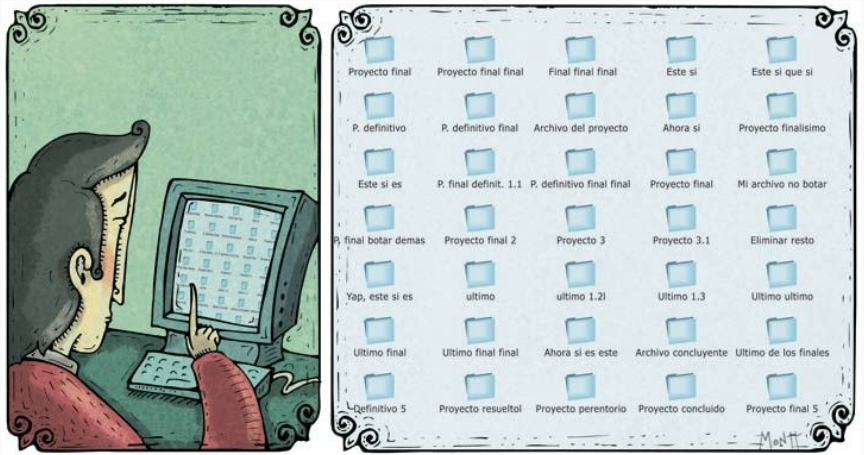
- Trabajar **en paralelo** (en ramas)

¿Tiene sentido para trabajo individual?

1. Permite trabajar en **diferentes funcionalidades sin mezclarlas**

   Por ej: **Cambio de contexto ante un bug crítico**

2. Evito trabajar con muchas carpetas:
   Uso **una sola carpeta**



¿Qué es git?

(para 80% de la gente)

¿Qué es git?

(para el que sabe usarlo)

Conceptos Básicos de Git: LOCAL

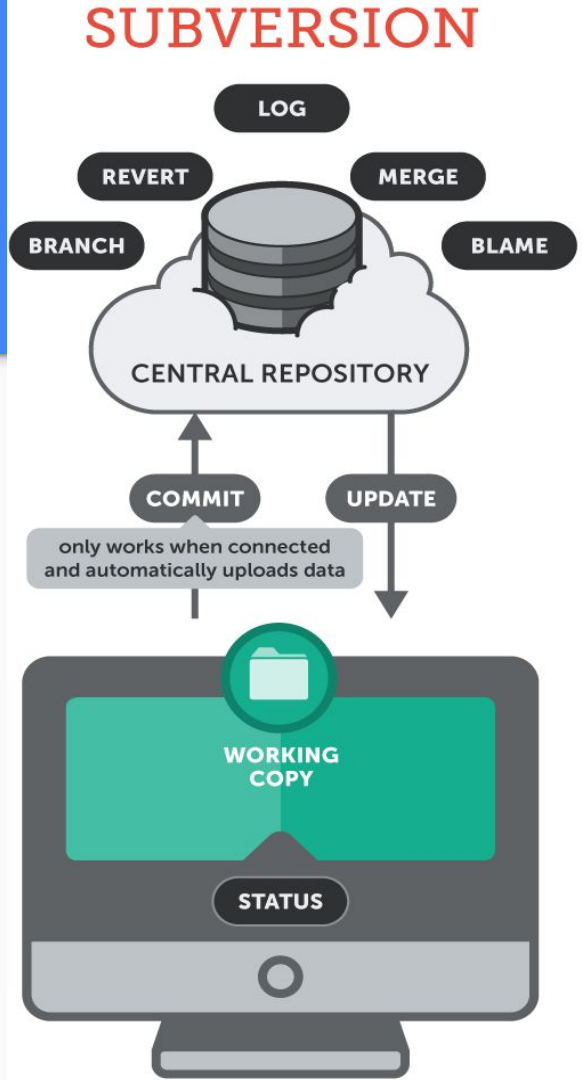
Las operaciones son **LOCALES**

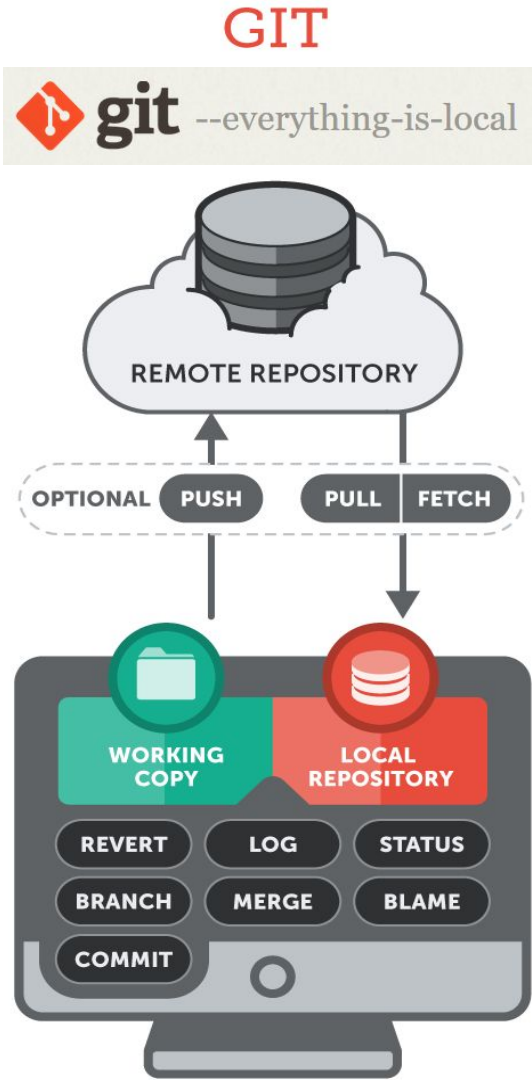
Offline: No necesitamos conexión para trabajar

Diferencias CVS/SVN

SVN/CVS: todo en servidor

GIT: todo en local

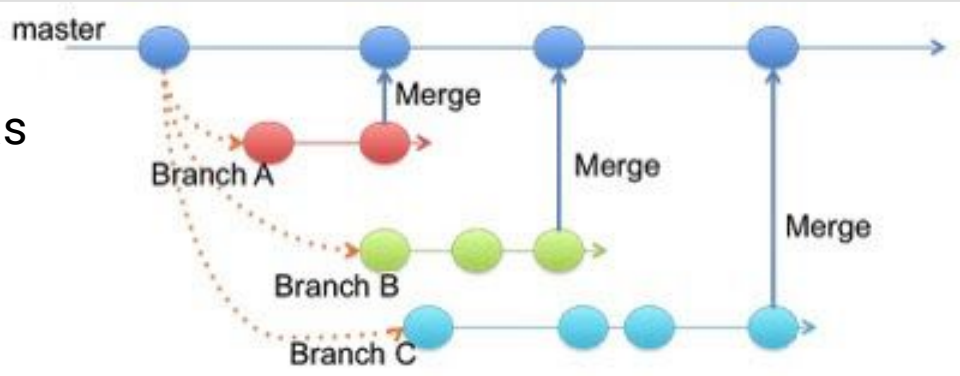




Ramas en Git

Líneas de trabajo aisladas unas de otras

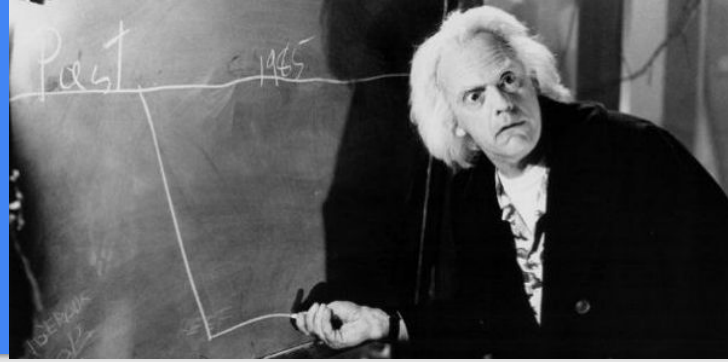
Rama principal por defecto: **MASTER**

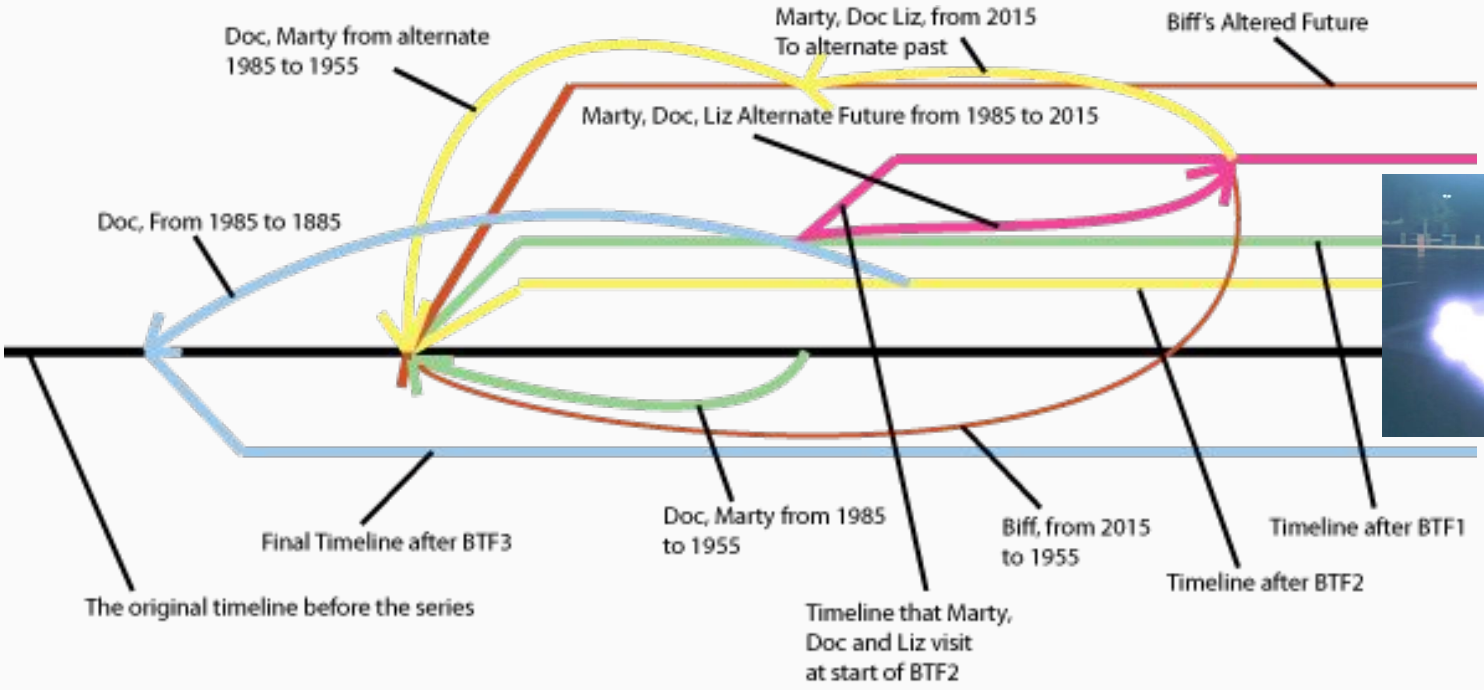


Creamos **nuevas ramas** durante el desarrollo (una por cada **funcionalidad**)

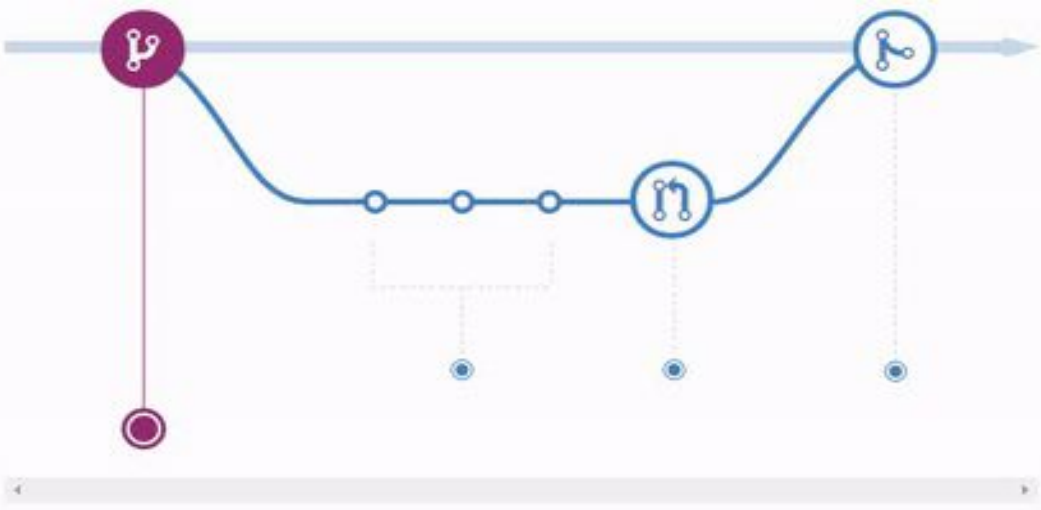
Cuando terminamos las fusionamos (**merge**) con la rama principal

Ramas (Regreso al Futuro)





Trabajo diario



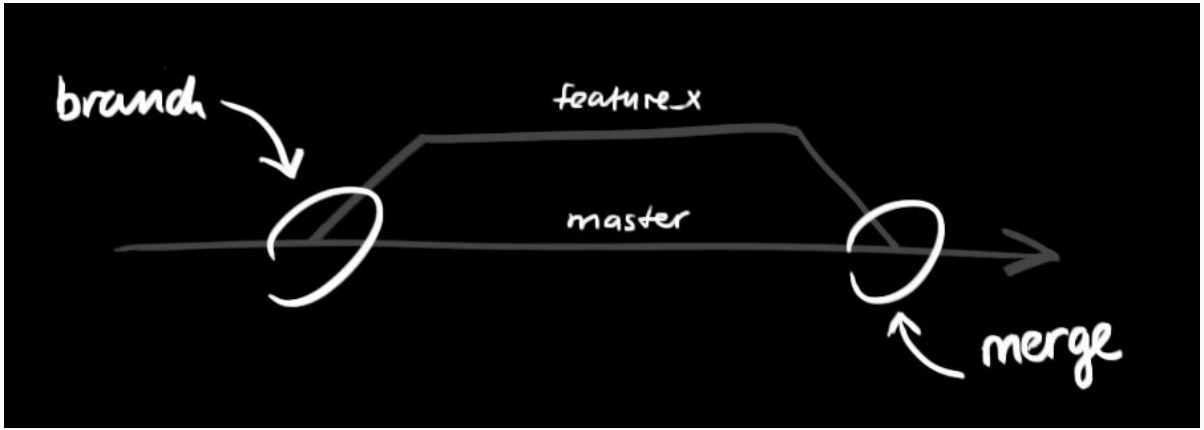
Create a branch

Para cada rama de trabajo

1. **Crear** una **rama** y **cambiar a ella**
2. Hacemos **commits** de nuestros cambios
3. Cuando acabamos, **Sincronizamos** con el **repositorio remoto**:
   a. **pull** → incluir los cambios que hayan subido otros
   b. **push** → **subir** nuestra **rama** al repositorio remoto
4. Solicitar **fusionar** nuestros cambios (**merge request**) en la rama principal

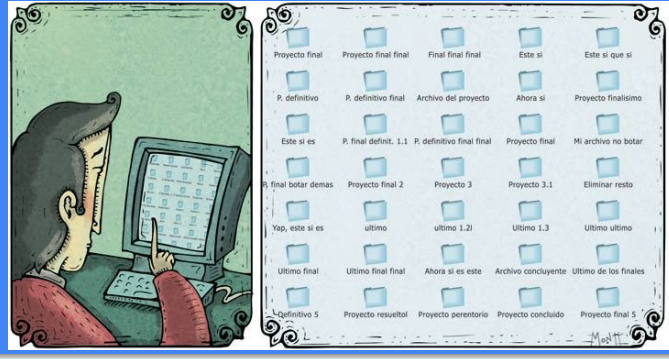
Crear una rama: branch

Una rama = Una funcionalidad

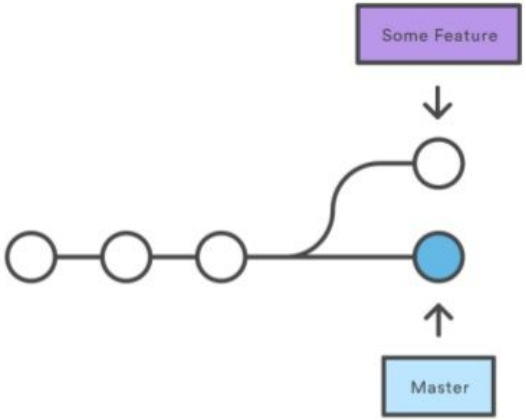


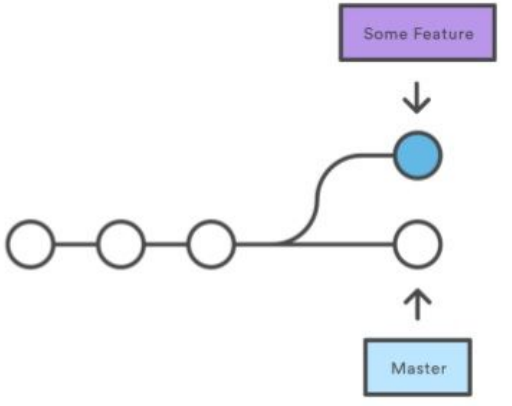
git branch nombre_de_nueva_rama

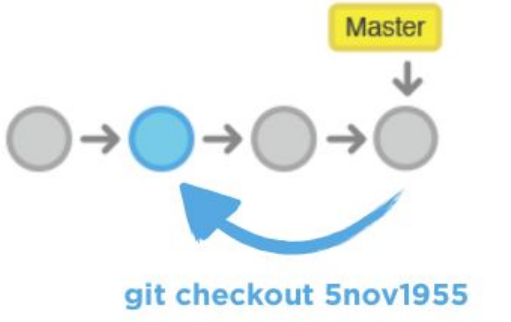
Cambiar de rama: switch / checkout



Para **cambiar a** una rama: **git checkout** rama







git fetch: Informarse de cambios remotos

git fetch se **entera** de los cambios en remoto

git --everything-is-local

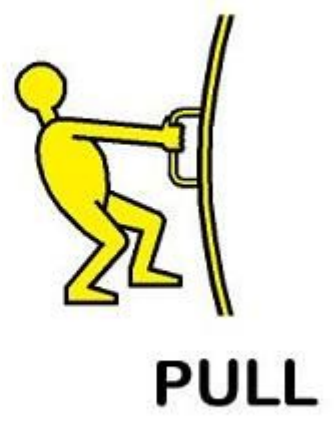
No sobreescribe nuestro directorio de trabajo

Algunos IDEs implementan **autoFetch**

git pull: Bajar y fusionar cambios remotos

git pull ⇒ git merge de rama remota indicada

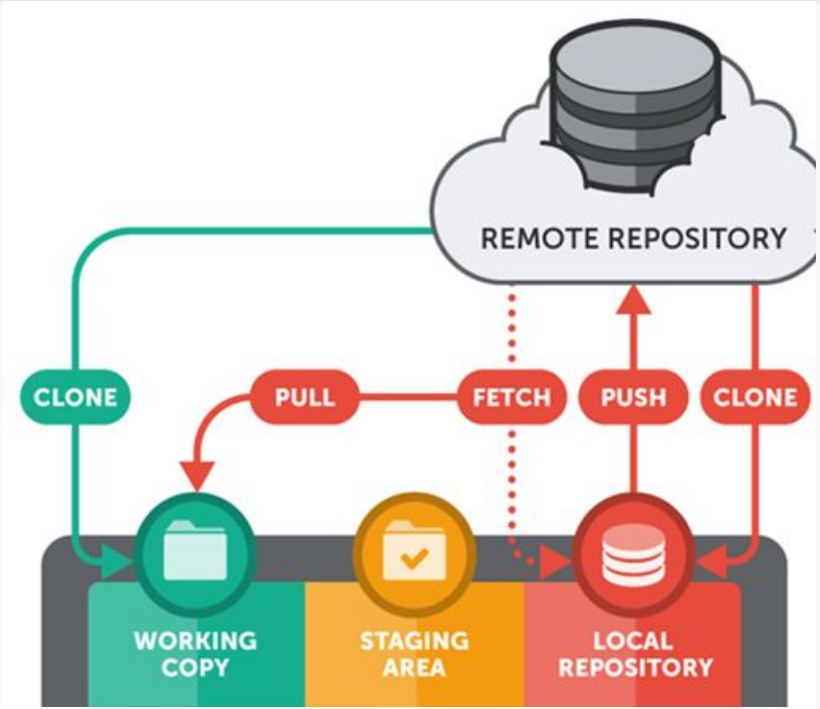
Sobreescribe nuestro directorio de trabajo



Fetch vs Pull

Pull

Sobrescribe directorio de trabajo
(más agresivo: **pulla**)



Fetch

Se informa de cambios remotos
No sobreescribe

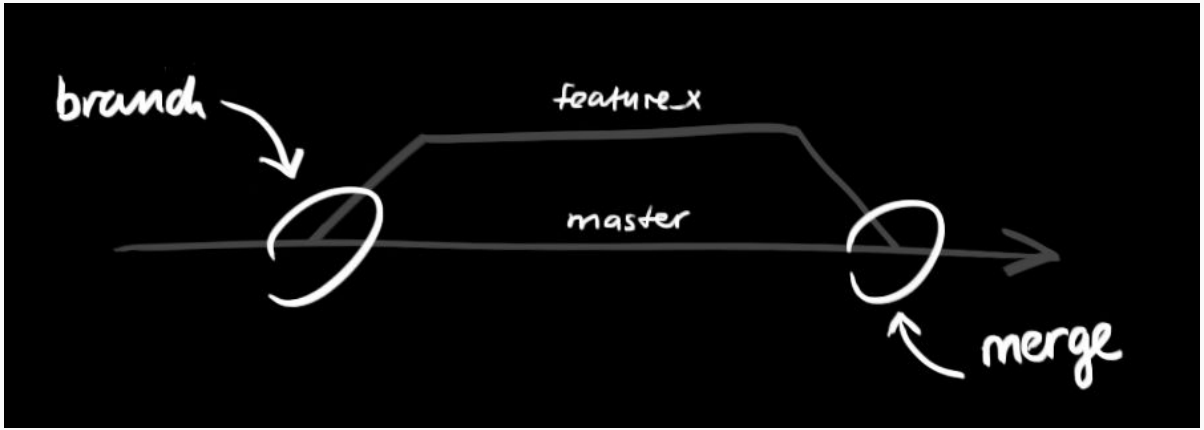
Hacer Push (subir rama a remoto)

git push origin rama

Sube la rama de trabajo (local) al repositorio remoto origin

Si hay conflictos no nos dejará hacerlo

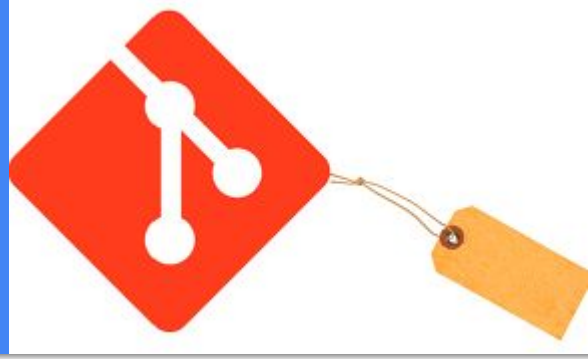
Fusionar (merge) una rama



git merge rama

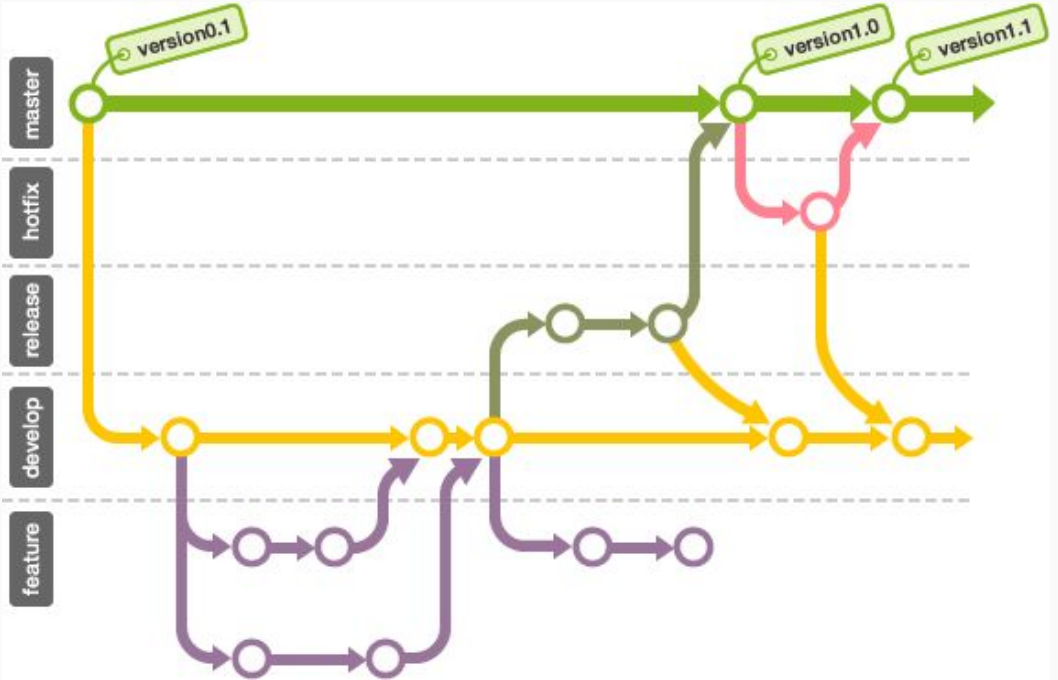
Fusiona la rama indicada en la rama donde estamos trabajando

git tag: Etiquetar versiones



Son **marcas fijas**
(no evolucionan)

Para etiquetar **versiones**



Flujo de trabajo: GitLab Flow

RAMAS estandarizadas:

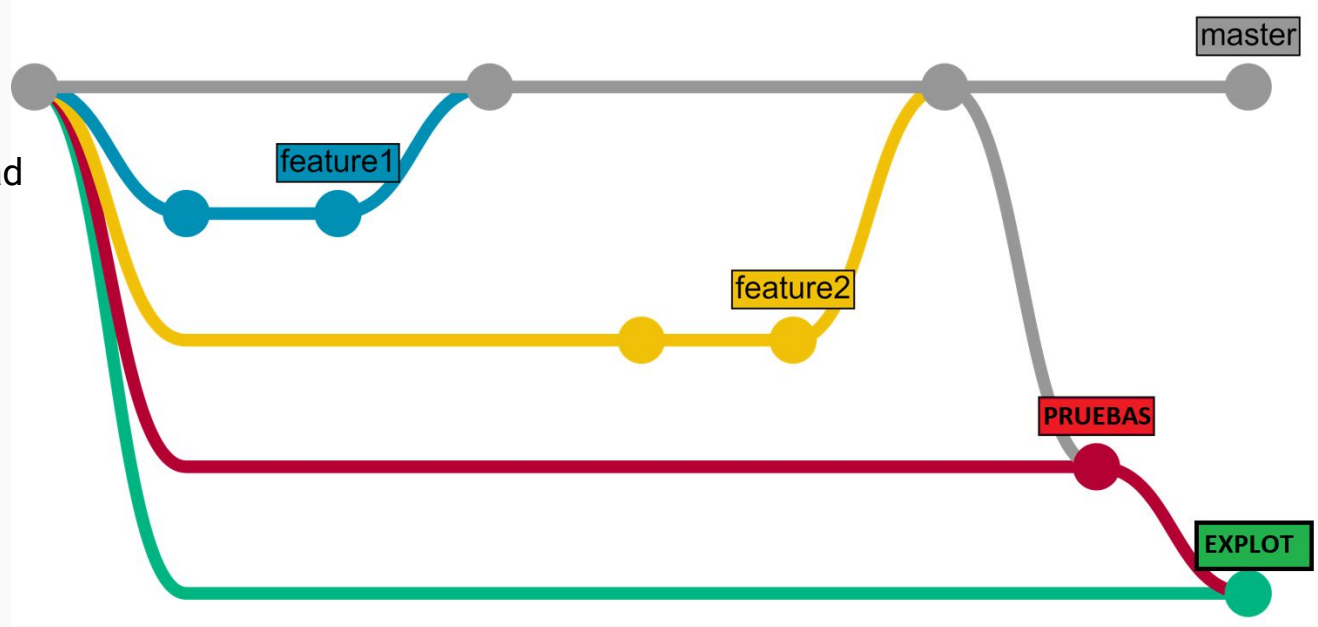
* **master**: desarrollos parten de ella

features: una por cada funcionalidad

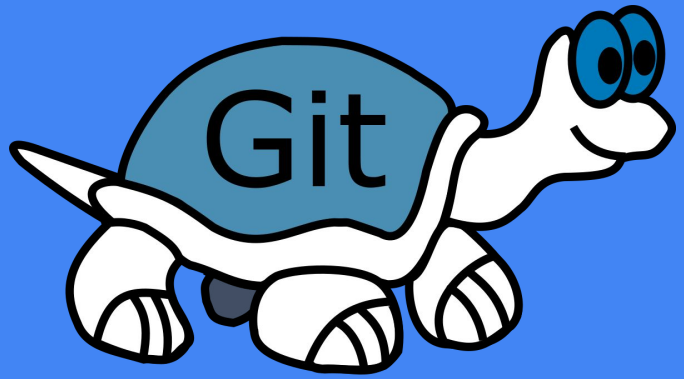
ejemplos: Nuevo_menu
Corregir_error...

* **rama para cada entorno**:
PRUEBAS

EXPLOTACIÓN

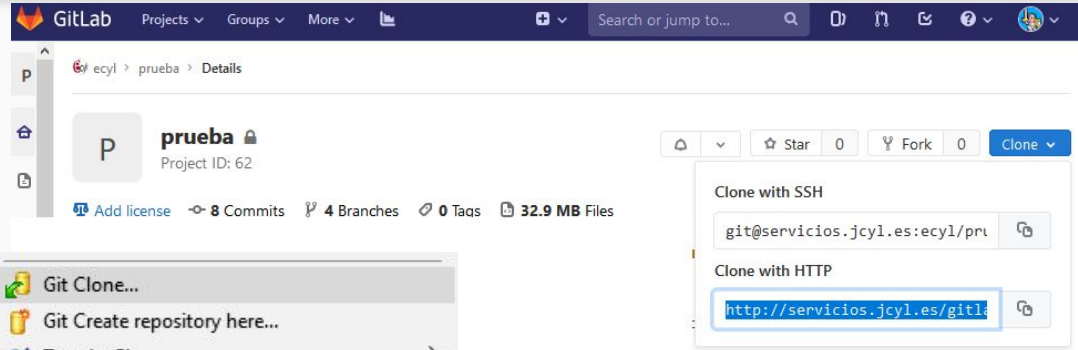


EJEMPLO PRÁCTICO CON TORTOISEGIT



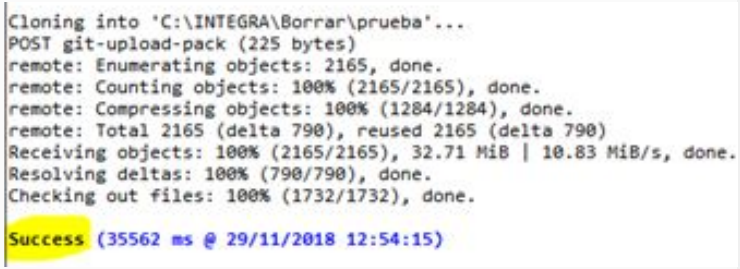
Clone (sólo la primera vez)

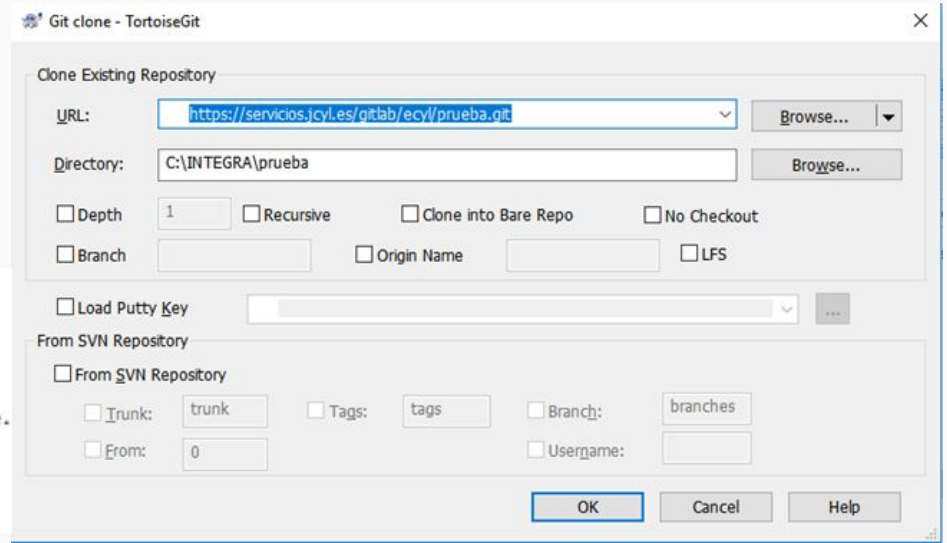
1. En **GitLab** copiamos la **url** (**.git**) del proyecto (**ojo**: http**S**)



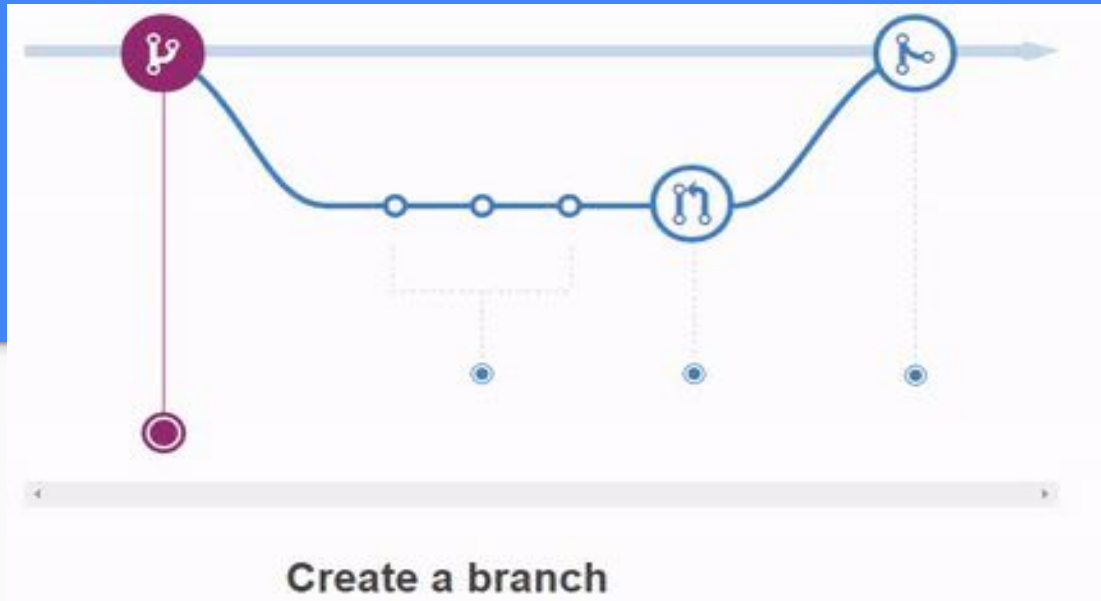
2. En **Explorador de Windows** botón derecho → **git clone**

3. En **TortoiseGit** pegamos la **url**
   Nos pide credenciales (DA o ext)
   Si todo va bien dirá **Success**:





Trabajo diario



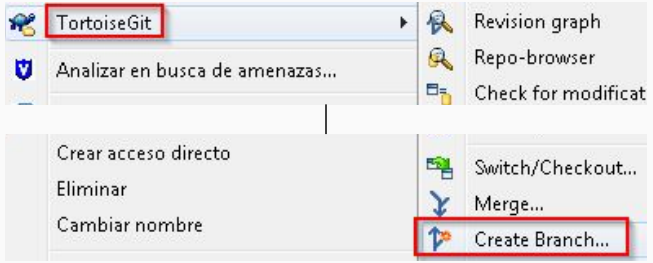
Para cada rama de trabajo

1. **Crear** una **rama** y **cambiar a ella**
2. Hacemos **commits** de nuestros cambios
3. Cuando acabamos, **Sincronizamos** con el **repositorio remoto**:
   a. **pull** → incluir los cambios que hayan subido otros
   b. **push** → **subir** nuestra **rama** al repositorio remoto
4. Solicitar **fusionar** nuestros cambios (**merge request**) en la rama principal

Crear nueva rama

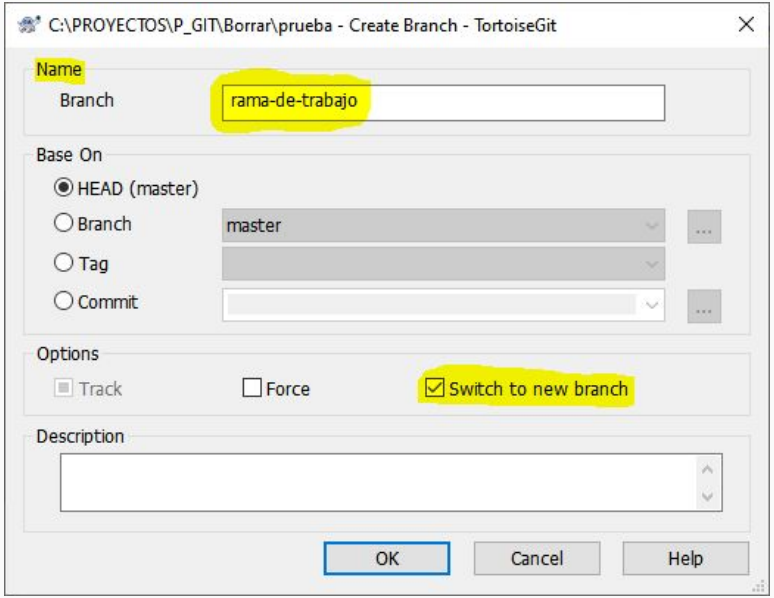
1. En **Explorador de Windows**

   Botón derecho → **TortoiseGit** → **create Branch**



2. En **TortoiseGit**:
   a. Ponemos un **nombre**

   b. Marcamos “**Switch to new branch**” para trabajar ya sobre la rama



Si todo va bien dirá **Success**

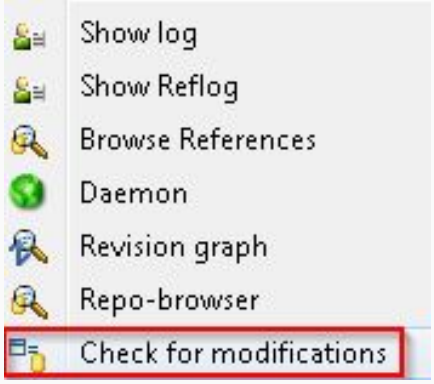
```
git.exe checkout rama-de-trabajo --

Switched to branch 'rama-de-trabajo'

Success (1078 ms @ 29/11/2019 10:41:19)
```

Comprobar modificaciones

Trabajamos en la rama

Git registra todos los cambios

Para ver los cambios que hemos hecho:
Tortoise git → check for modifications

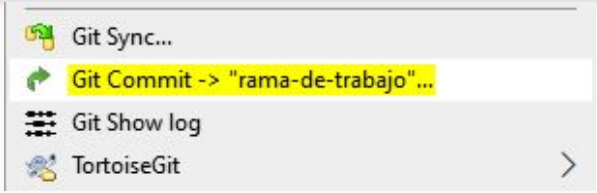


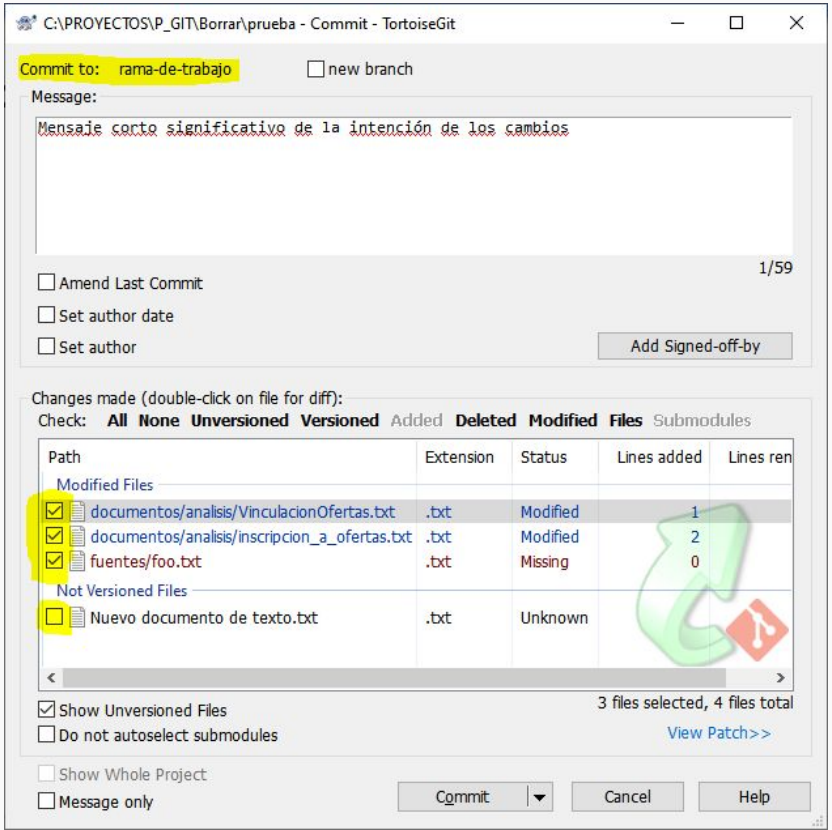
C:\PROYECTOS\P_GIT\Borrar\prueba - Working Tree - TortoiseGit

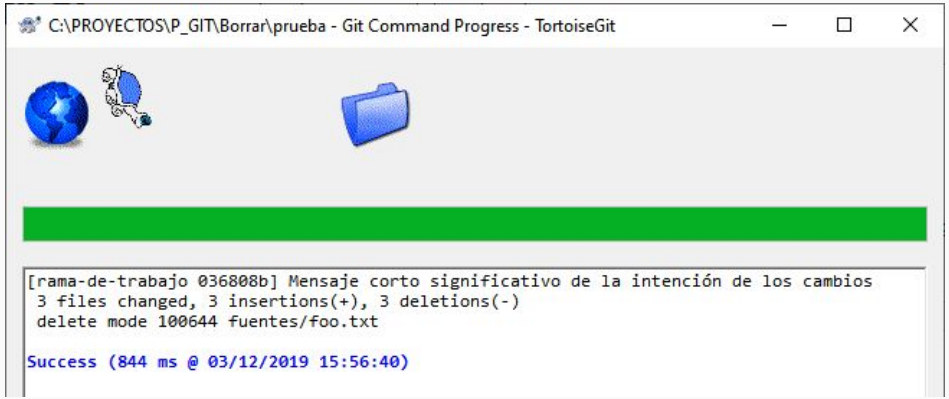
Path	Extension	Status	Lines added	Lines removed	Last Modified
Modified Files					
documentos/analisis/VinculacionOfertas.txt	.txt	Modified	1	1	29/11/2019 11:12:26
documentos/analisis/inscripcion_a_ofertas.txt	.txt	Modified	2	2	29/11/2019 11:12:13
fuentes/foo.txt	.txt	Missing	0	0	
Not Versioned Files					
Nuevo documento de texto.txt	.txt	Unknown			29/11/2019 11:11:25

Hacer commits

Para **confirmar los cambios: Git Commit → <rama>**



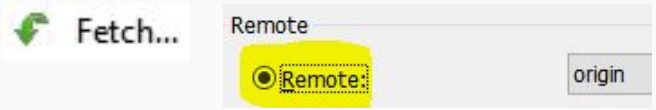




Fetch y Pull: Sincronizar con el servidor remoto

1. **TortoiseGit → Fetch**:

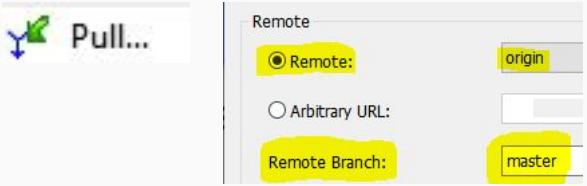
Enterarse de los cambios remotos




```
From https://servicios.jcyl.es/gitlab/ecyl/prueba
= [up to date]      2-error_listado   -> origin/2-error_listado
= [up to date]      2-error_listado-2 -> origin/2-error_listado-2
= [up to date]      corregir_error    -> origin/corregir_error
7f4fe2e..7d08b50  master              -> origin/master

Success (1969 ms @ 04/12/2019 12:25:04)
```

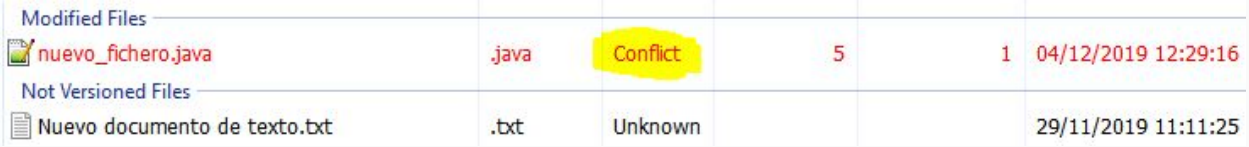
2. **TortoiseGit → Pull**:

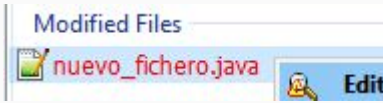
Incluir cambios que haya habido en la rama indicada (ej: **master**)




```
From https://servicios.jcyl.es/gitlab/ecyl/prueba
* branch            master     -> FETCH_HEAD
= [up to date]      master     -> origin/master
Auto-merging nuevo_fichero.java
CONFLICT (content): Merge conflict in nuevo_fichero.java
Automatic merge failed; fix conflicts and then commit the result.

git did not exit cleanly (exit code 1) (2000 ms @ 04/12/2019 12:29:17)
```

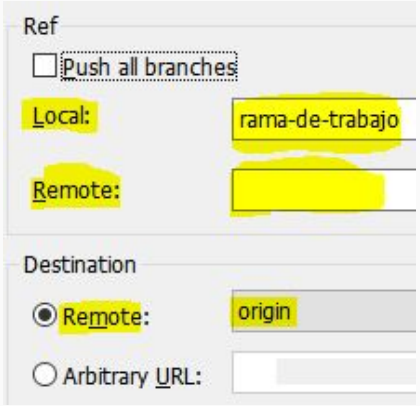
Si hay **conflictos**: **TortoiseGit → Check for modifications:** editamos + **commit**

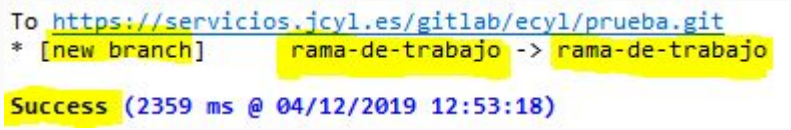




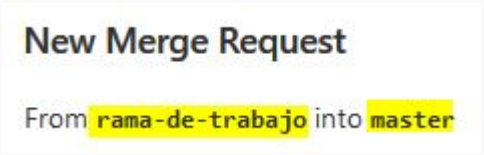
Push: Sincronizar con el servidor remoto

3. **TortoiseGit → Push:**
   Subir nuestra **rama** al repositorio remoto

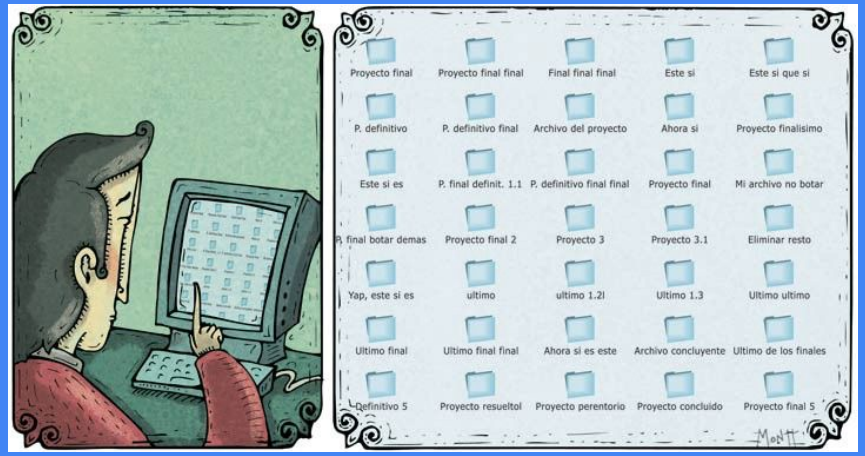




4. **En GitLab** pedimos una **merge request** para que **integren** nuestra **rama de trabajo** en **master**



Si sólo pudieras quedarte con un par de ideas:



¡Gracias!