

Programación Frontend y Backend

BLOQUE JAVA

GIT





01

Conceptos
Básicos

02

Software
de control
de
versiones

03

Workflow
Básico con
GIT

04

Comandos
GIT

05

TortoiseGit

06

Resolución
de
Conflictos



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO

EOI

Escuela de
organización
industrial



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



00

Material
necesario



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Escuela de
organización
industrial

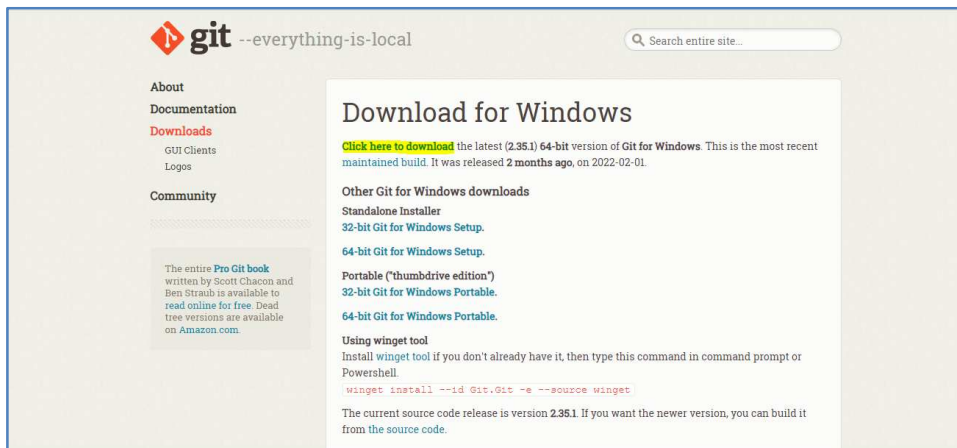


Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Software necesario – Cliente Git

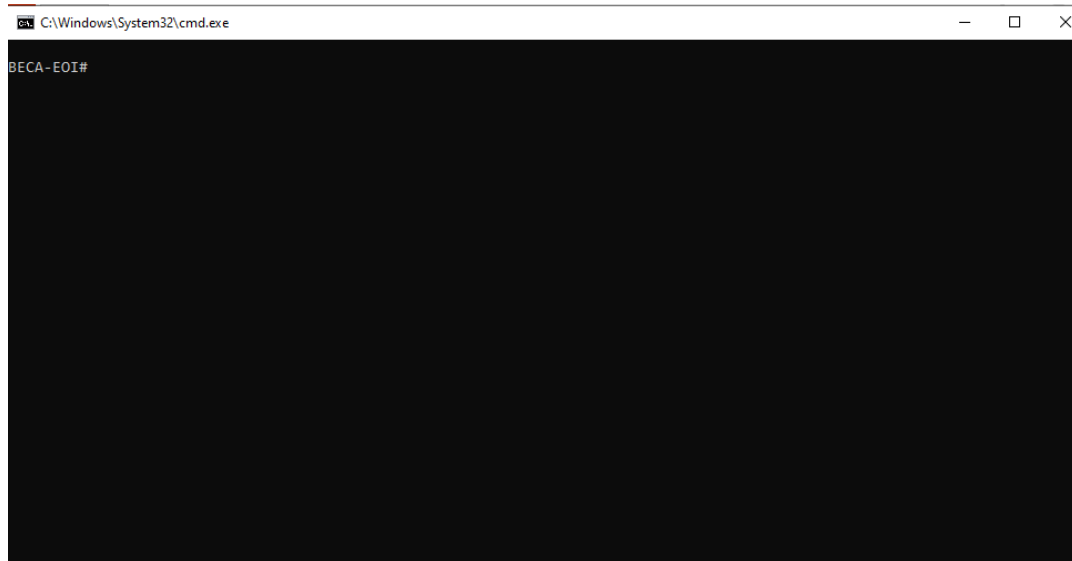
- <https://git-scm.com/download/win>
- Podéis cambiar el Prompt del CMD con el comando **prompt <texto>** , yo lo he cambiado a **BECA-EOI#** usando el comando **prompt BECA-EOI#**



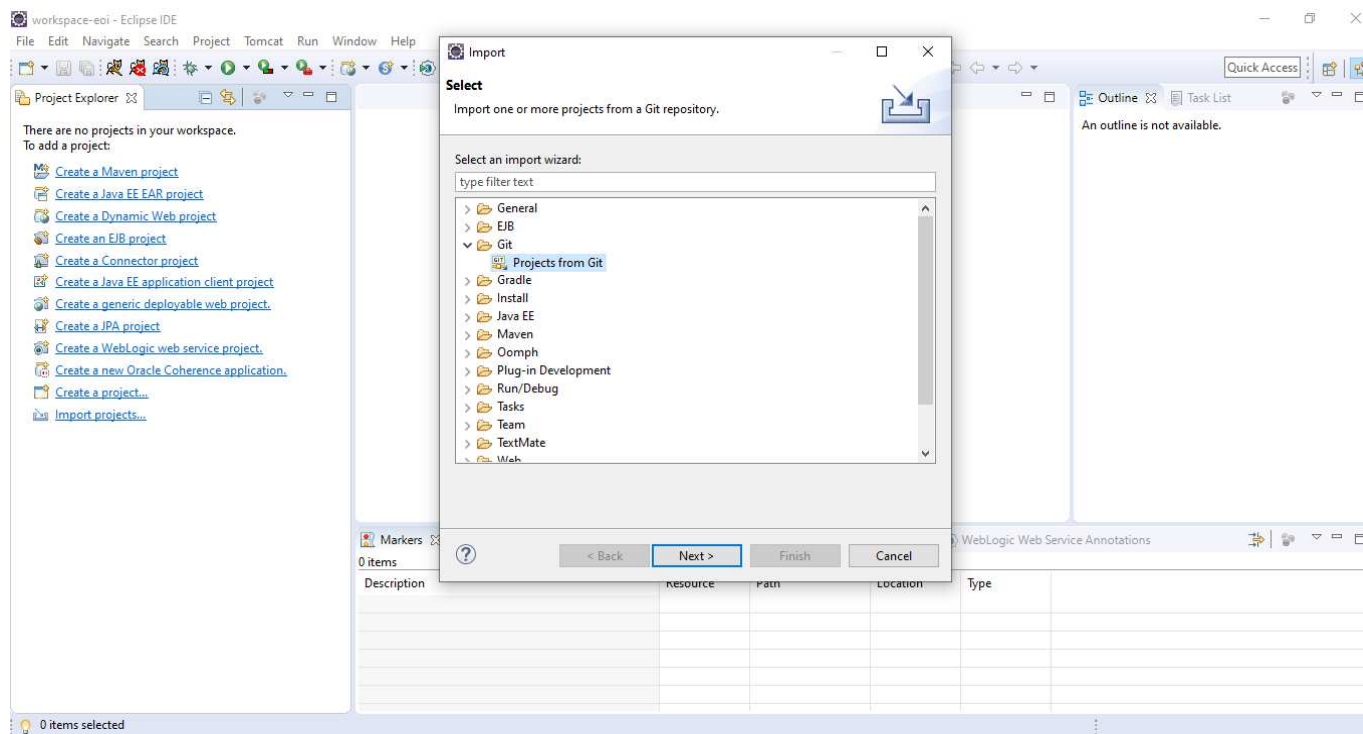
```
BECA-EOI#git --version  
git version 2.35.1.windows.2
```

Software necesario – Usaremos un terminal CMD de Windows

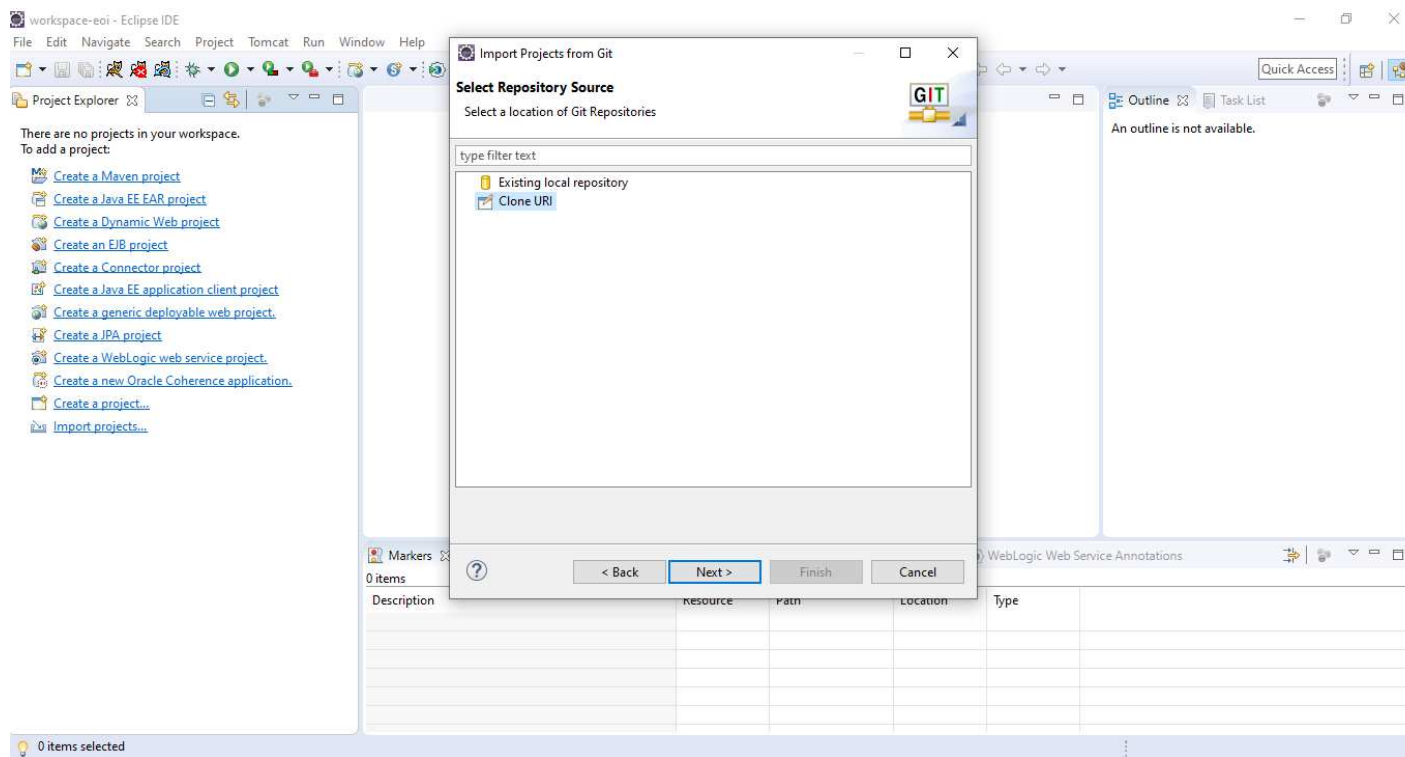
- Todos los comandos **Git** que veremos los ejecutaremos desde un **CMD** de **Windows**



Software necesario – Veremos también el uso de Git desde nuestro IDE



Software necesario – Veremos también el uso de Git desde nuestro IDE



Software necesario – Configuración de usuario y correo

Para poder usar nuestro cliente Git tendremos que configurar primeramente nuestro nombre de usuario y correo, los cuales los utilizará Git para identificarnos en cada operación que hagamos : **git config --global --edit** (Indicad vuestros datos como indica la sección **[user]**)

```
C:\EOI\EOI-RAMON\2022\Marzo2022\GitOk\032022>git config --global --edit
hint: Waiting for your editor to close the file...

C:\Users\rmesegue\gitconfig - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
1 [user]
2   email = aquirammon@gmail.com
3   name = R.J.M.I
4 [http]
5   sslVerify = false
6 [difftool "sourcetree"]
7   cmd = '' \'$LOCAL\' \'$REMOTE\'
8 [mergetool "sourcetree"]
9   cmd = ''
10  trustExitCode = true
11 [filesystem "Oracle Corporation|1.8.0_172|1484618545"]
12   timestampResolution = 24001 microseconds
13   minRacyThreshold = 0 nanoseconds
14 [filesystem "Oracle Corporation|11.0.8|1484618545"]
15   timestampResolution = 2005 microseconds
16   minRacyThreshold = 0 nanoseconds
17 [filesystem "Oracle Corporation|11.0.9|1484618545"]
18   timestampResolution = 1000 microseconds
19   minRacyThreshold = 0 nanoseconds
20 [filter "lfs"]
21   process = git-lfs filter-process
22   required = true
23   clean = git-lfs clean -- %f
24   smudge = git-lfs smudge -- %f
```



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO

EOI Escuela de
organización
industrial



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



01

Conceptos Básicos



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Escuela de
organización
industrial



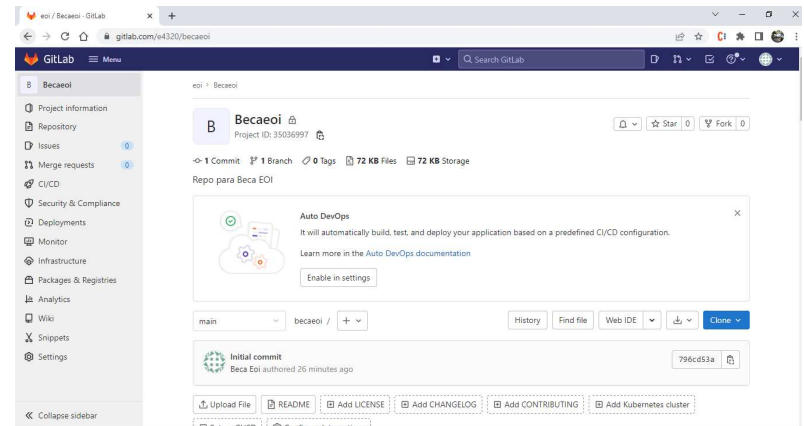
Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Sistemas de control de versiones

¿Qué es un control de versiones ?

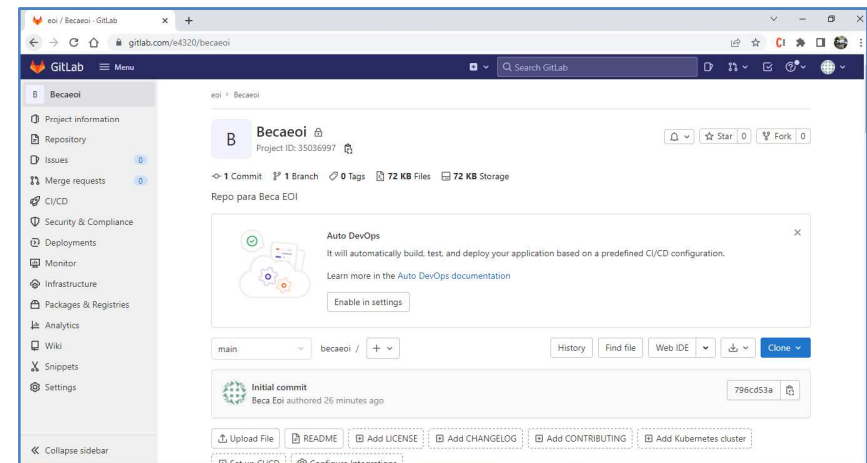
- Podemos pensar en un sistema de control de versiones o VCS como algo similar a una base de datos. Esta herramienta nos permitirá grabar una instantánea de todo nuestro proyecto en un momento determinado.
- Posteriormente, podremos en el momento en que sea necesario comparar la última foto instantánea de nuestro proyecto con cualquier otra “versión” que hayamos grabado con anterioridad, comprobando así exactamente que diferencia una versión de otra.



Sistemas de control de versiones

Ramas principales

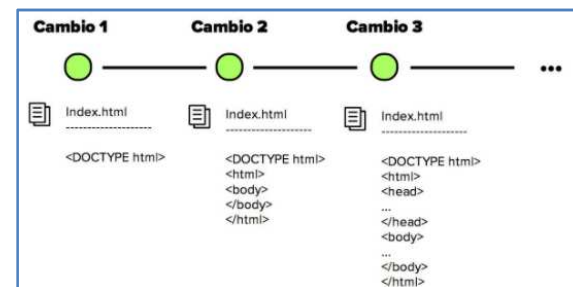
- La rama principal **main** o **master** es la rama principal, la rama que tiene que contener la versión de Software que vayamos a usar en producción
- La rama **develop** se suele usar para los desarrollos que se están realizando, a partir del código que se encuentra en la rama principal. Una vez se ha terminado de trabajar con esta rama se deberán integrar los cambios en la rama principal (**merge**)
- Pueden existir más ramas con más nombre, depende del proyecto y de las especificaciones del mismo e integraciones con sistemas DevOps
- Nosotros usaremos las ramas **main** y otras que llevarán las iniciales de los nombres de cada uno, estas ramas con las iniciales serán creadas a partir de la rama **main**



¿Porqué necesito un control de versiones ?

Colaboración

- Un sistema de control de versiones aporta numerosas ventajas , enumeremos las más evidentes:
- Sin un sistema de control de versiones , probablemente estés trabajando junto con otros compañeros en una carpeta compartida en red donde no es posible trabajar de manera colaborativa con un flujo de trabajo aceptable.
- Esta dinámica de trabajo es muy propensa a cometer errores, además es una mera cuestión de tiempo que alguien sobre escriba el trabajo de su compañero.
- Con un SCV cualquier miembro del equipo puede trabajar con total libertad en cualquier fichero . Posteriormente podremos integrar todos estos cambios de manera simple y ordenada en una sola versión.



Todo esto además de confuso y poco eficiente, hace cada vez más difícil responder a una cuestión esencial:

¿Qué ha cambiado exactamente entre cada una de las versiones almacenadas ?

Almacenar versiones (Correctamente)

Almacenar una versión de tu proyecto después de hacer cambios en él es un hábito más que saludable, sin embargo puede convertirse en una tarea tediosa y confusa rápidamente.

Pronto surgirán cuestiones sobre qué cambios deben almacenarse o cómo han de renombrarse las versiones sucesivas . Esto con el tiempo se degradará en una lista de nombre de fichero parecida a esta:

- Proyecto_acme
- Proyecto_Acme_v1
- Proyecto_Acme_revxx99B_sin_parametrizar
- proyecto Acme_revxx99B_sin_parametrizar_andres_ocutbre_2017_v4_o v_5_o unamezcla deyanimeacuerdo_3A)

Recuperar versiones anteriores

Ser capaz de recuperar efectivamente versiones anteriores de un fichero o incluso de todo el proyecto, implica que no puedan cometerse errores irreversibles, de tal forma que cualquier fragmento de código erróneo que introduzcamos puede revertirse con unos cuantos clicks.

git revert

git reset

Comprender los cambios

Cada vez que subimos una nueva versión introducimos comentarios que ayudan a comprender los cambios que se han implementado. Además, si se trata código o un fichero en texto, podemos ver exactamente el cambio introducido.

git log

02

Software de
control de
versiones



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Escuela de
organización
industrial



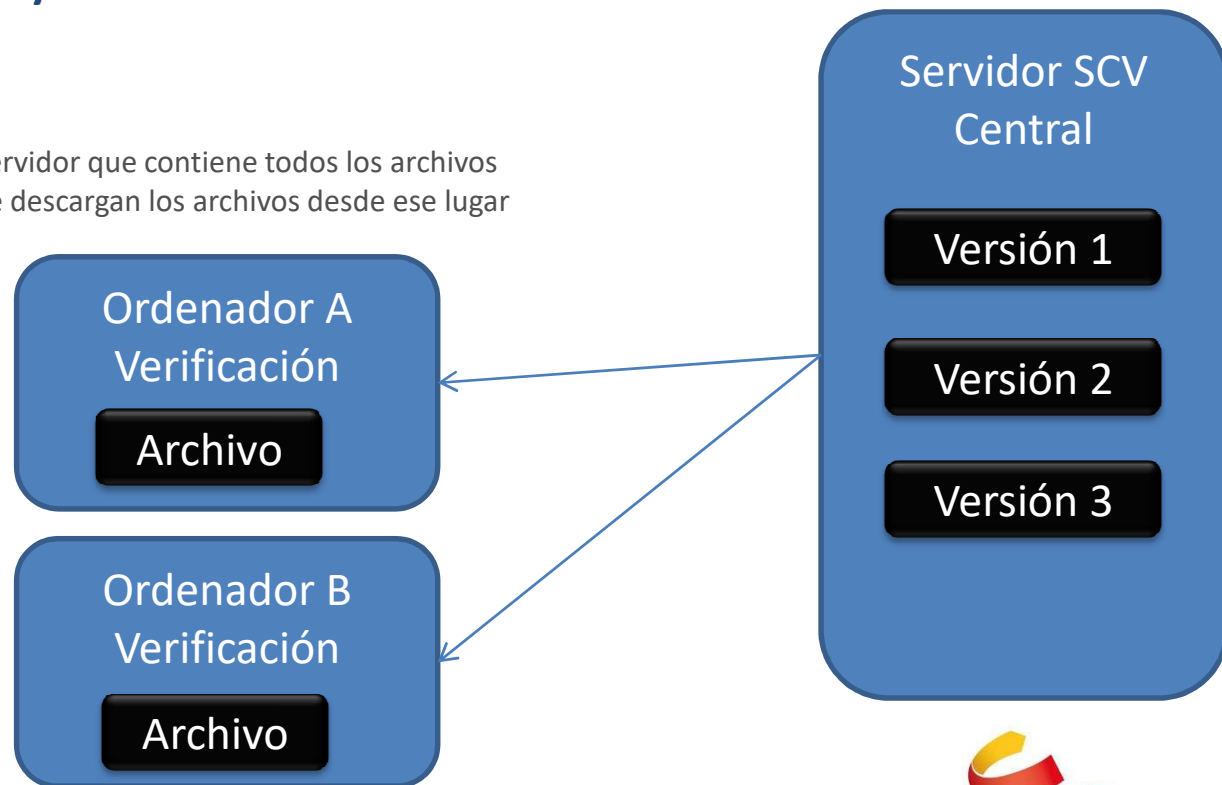
Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Sistemas centralizados y sistemas distribuidos

SCV Centralizados

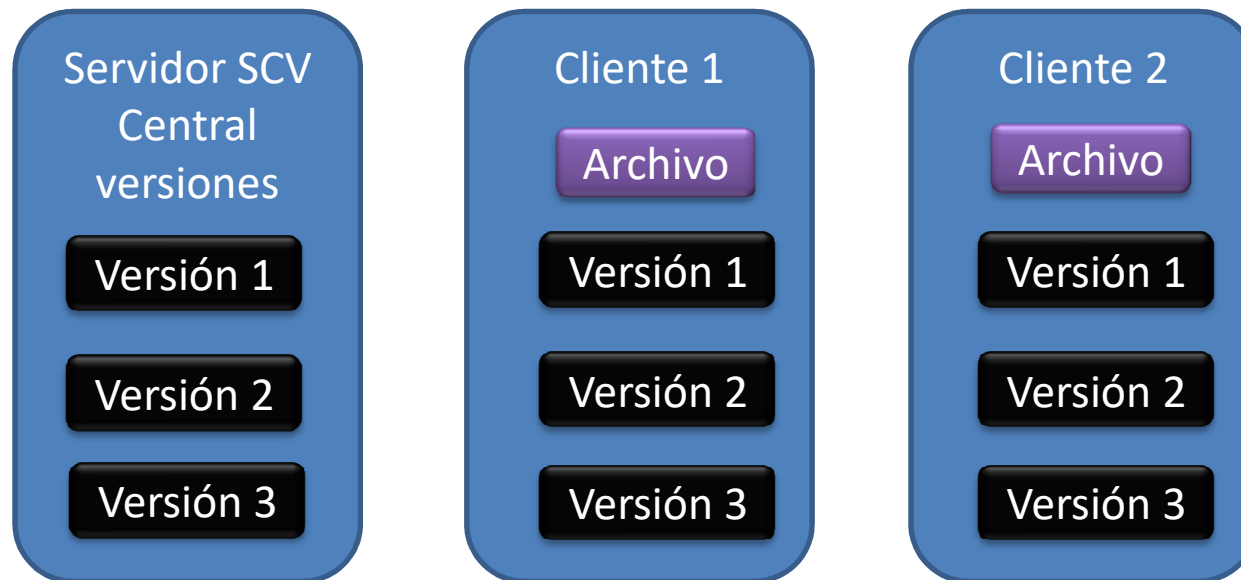
Estos sistemas tienen un único servidor que contiene todos los archivos versionados, y varios clientes que descargan los archivos desde ese lugar central. CVS, Subversion.



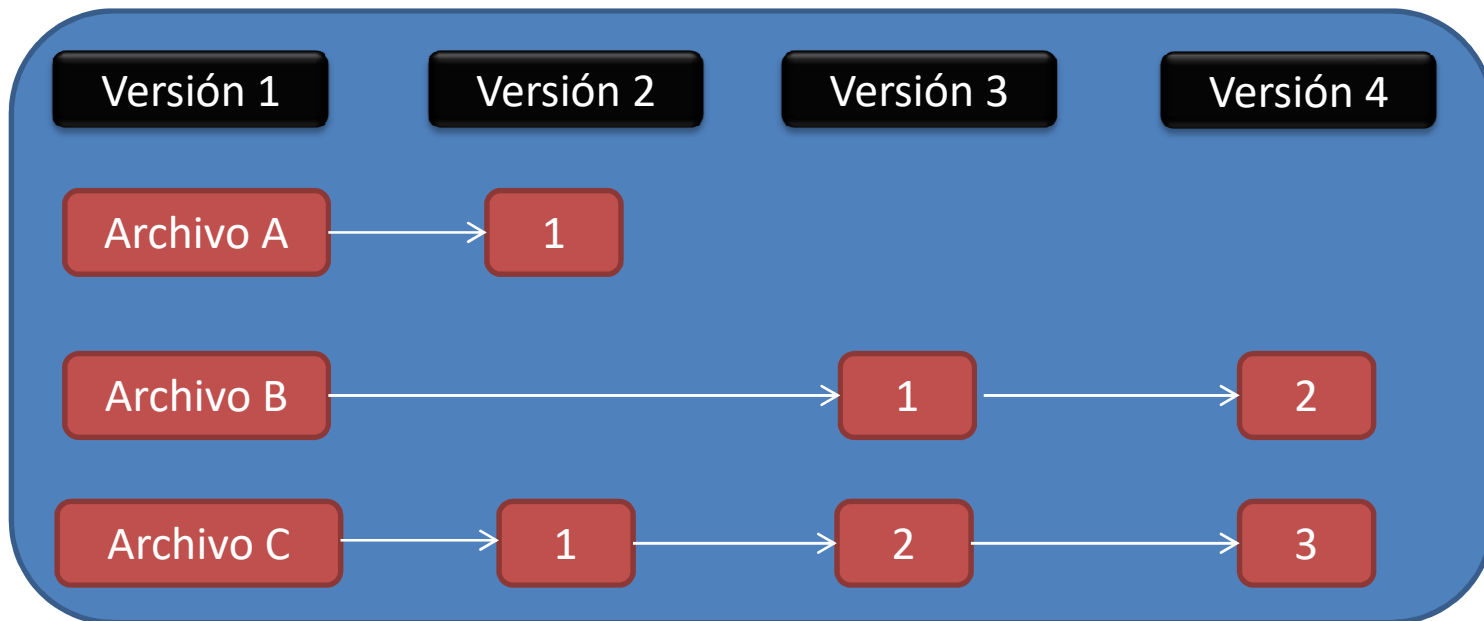
Sistemas centralizados y sistemas distribuidos

SCV Distribuidos

En un SCV, los clientes no sólo descargan la última instantánea de los archivos: replican completamente el repositorio, así son independientes del servidor. (Ejemplos : GIT, Mercurial)

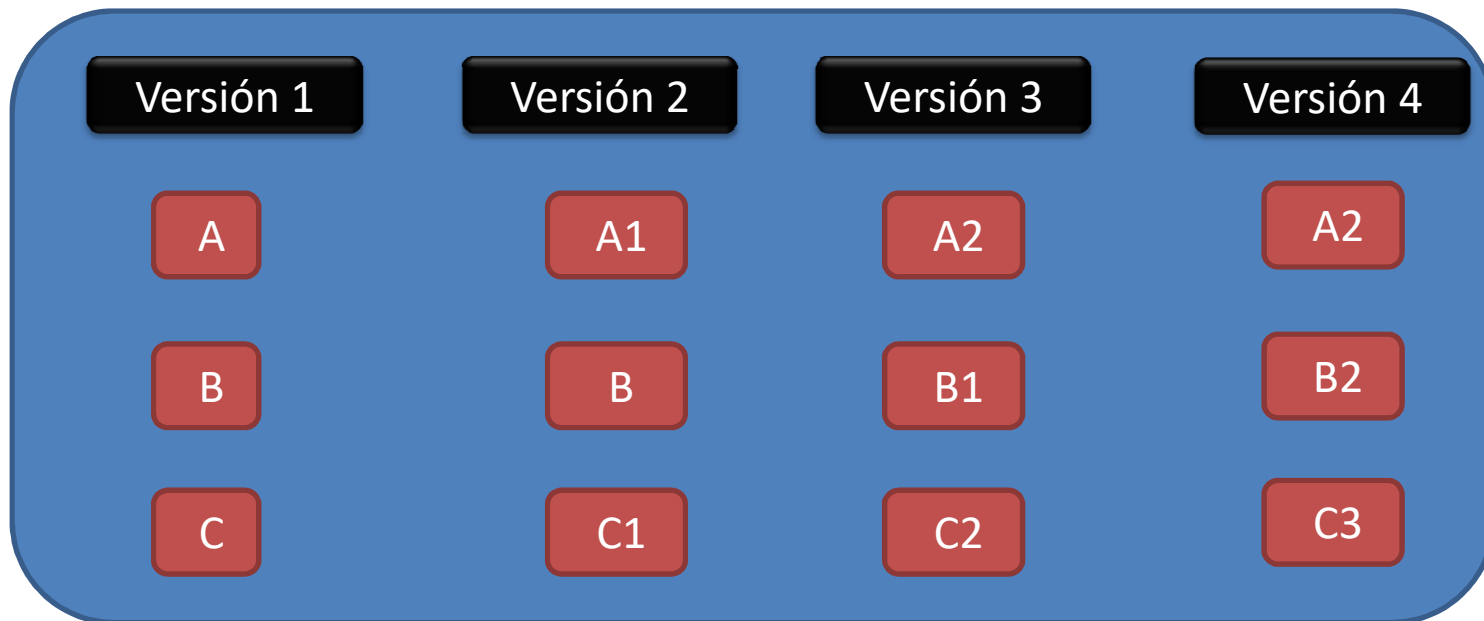


Versionado en subversión



Estos sistemas (CVS, Subversion, Perforce, Bazaar, etc.) modelan la información que almacenan como un conjunto de archivos y las modificaciones hechas sobre cada uno de ellos a lo largo del tiempo

Versionado en GIT



Cada versión es un «back-up» de todo el contenido del repositorio en un momento concreto.

Beneficios de GIT

Velocidad

- Casi todas las operaciones que realiza GIT las ejecuta en el ordenador local. No necesita información de otro ordenador, por lo que la velocidad es mucho mayor que en sistemas centralizados.
- Por ejemplo, para recuperar una versión anterior del proyecto no es necesario acudir al servidor porque nuestro ordenador local ya tiene una copia de todas las versiones, por lo que el acceso es casi instantáneo.
- También podemos hacer «commit» de los cambios incluso cuando estamos «offline». Y cuando estemos conectados simplemente subir los cambios al servidor.

Beneficios de GIT

Integridad

- GIT realiza una verificación de la calidad de los datos al guardarlos, denominado «checksum» que es una forma de proteger la integridad de los datos.
- Es imposible cambiar los contenidos de los archivos sin que GIT lo sepa.
- No se puede perder información ni guardar archivos corrompidos sin que GIT lo detecte y lo impida.
- GIT realiza el «checksum» utilizando un «hash» generado en SHA-1 y es la forma que tiene de identificar los archivos (no los identifica por su nombre).

Beneficios de GIT

Tranquilidad

- Las acciones en GIT siempre son modificables, es difícil hacer algo que provoque la pérdida de datos o que sea inmodificable.
- Después de hacer un «commit» es muy difícil perder datos puesto que se crean «snapshots» que se copian a todos los ordenadores que utilizan el proyecto.
- Aunque el proyecto se trabaje sólo en un equipo se pueden utilizar sistemas como Gitlab o Github o Bitbucket para almacenar una copia del repositorio.

03

Comandos Básicos de GIT



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Escuela de
organización
industrial



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



El Workflow Básico en Control de versiones

El Repositorio

Antes de perdernos en comandos de Git, será conveniente comprender cual es el flujo de trabajo básico. La punto de partida de todo esto es el Repositorio.

Pensad en un repositorio como una especie de Base de datos donde se almacenan todas la versiones y metadatos de tu proyecto. En GIT ese repositorio es una simple carpeta cuyo nombre es :

.git

A la hora de obtener ese repositorio pueden ocurrir dos cosas:

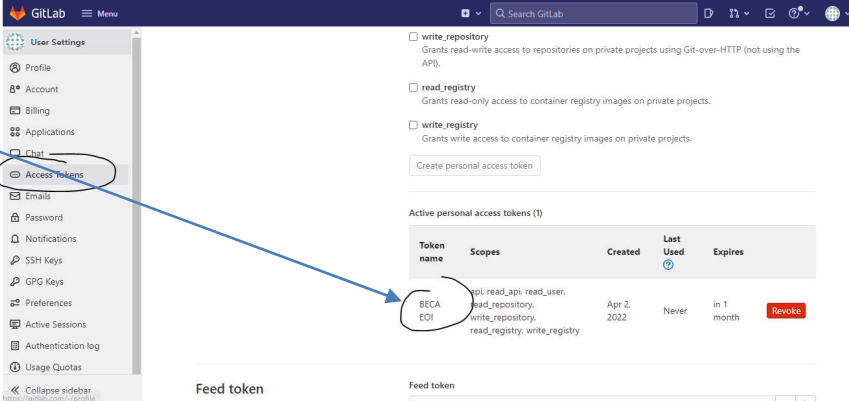
- Tenemos un proyecto que aún no está bajo control de versiones. En ese caso inicializaremos nuestro repositorio local para ese proyecto.
- Tenemos un proyecto en un servidor remoto, que deberemos clonar y descargar para contar con nuestro repositorio local. Para ello necesitaremos una URL a la que conectar, en nuestro caso es :
 - <https://gitlab.com/e4320/becaeoi.git>

El Workflow Básico en Control de versiones

El Repositorio

- Credenciales para acceder al repositorio :
 - Usaremos todos el mismo Access Token para acceder al repositorio Git

`glpat-sXTm9Un4JT6ea8hT99j3`



Token name	Scopes	Created	Last Used	Expires
BECA EOI	api.read_api.read_user, read_repository, write_repository, read_registry, write_registry	Apr 2, 2022	Never	in 1 month

- <https://glpat-sXTm9Un4JT6ea8hT99j3:glpat-sXTm9Un4JT6ea8hT99j3@gitlab.com/e4320/becaeoi.git>

El Workflow Básico en Control de versiones

Staging Area

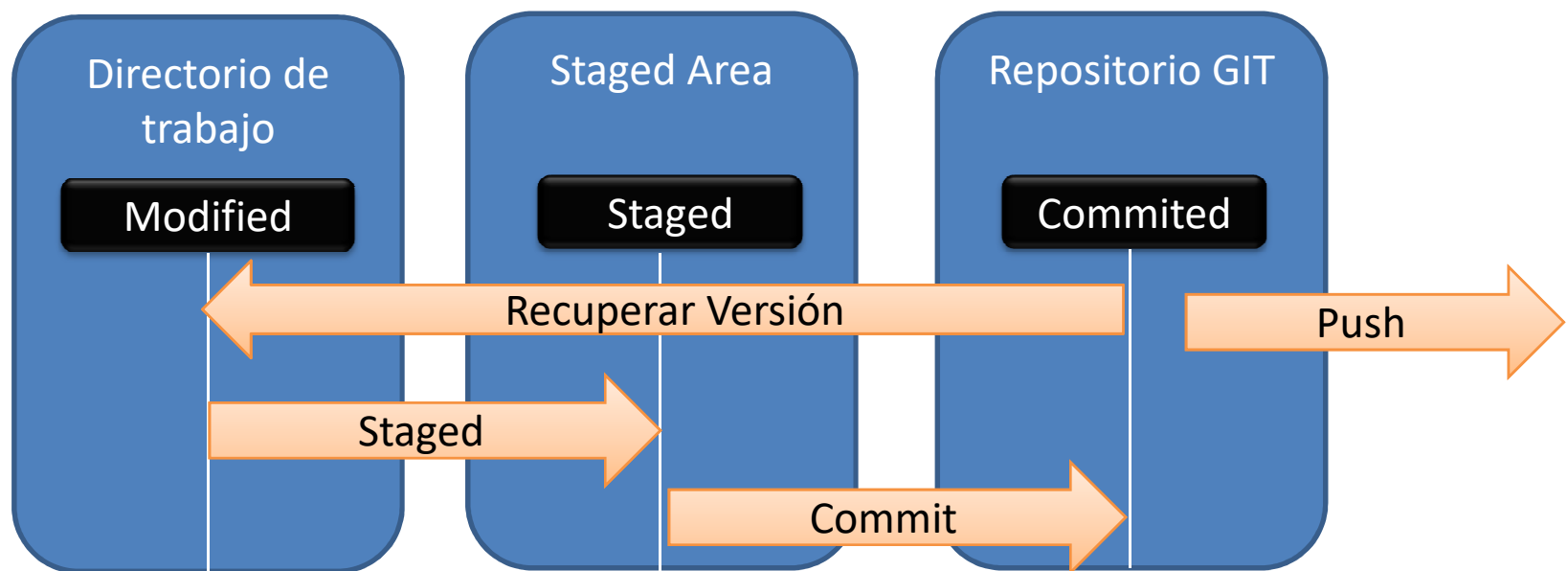
El mero hecho de modificar un archivo no implica necesariamente que esa modificación deba subir como nueva versión. Debemos indicar a GIT explícitamente que cambios queremos que formen parte de la próxima versión. Para ello añadiremos los ficheros a la llamada **Staging Area**.

Una vez hecho esto es el momento de realizar el commit, como buena práctica incluiremos siempre un breve comentario descriptivo del cambio, y lo subiremos al repositorio.

Commit

Podemos definir el commit como un conjunto de cambios específicos cada uno de estos conjuntos de cambios conformarán una versión diferente de nuestro proyecto. Por este motivo, cada commit representa una foto instantánea de todo nuestro proyecto en un determinado momento.

Estados de los archivos en GIT



04

Comandos GIT



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Escuela de
organización
industrial



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Comandos básicos : Clonar repositorio **git clone**

Clonar repositorio remoto en local : **git clone**

C:\Windows\System32\cmd.exe

```
BECA-EOI#git clone https://glpat-sXTm9Un4JT6ea8hT99j3:glpat-sXTm9Un4JT6ea8hT99j3@gitlab.com/e4320/becaeoi.git
Cloning into 'becaeoi'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 2.78 KiB | 91.00 KiB/s, done.
BECA-EOI#
```



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Escuela de
organización
industrial



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Comandos básicos : Ver estado **git status** y ramas **git branch**

Status

```
BECA-EOI#git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Listar ramas (Repositorio Local y Repositorio Remoto origin/*)

```
BECA-EOI#git branch -l
* main

BECA-EOI#git branch -l -r
origin/HEAD -> origin/main
origin/main
```



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Escuela de
organización
industrial



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Comandos básicos: Crear una rama a partir de otra **git checkout -b <rama-nueva>**

Crear rama en repositorio local a partir de otra (Ejemplo creando rama a partir de la rama **main**)

Primero comprobar siempre en que rama nos encontramos, la rama se creará a partir de la rama donde nos encontremos e inmediatamente cambiará la nueva rama creada (Crear una rama con las iniciales de vuestro nombre , no usar el de la transparencia)

```
BECA-EOI#git checkout -b RJMI
Switched to a new branch 'RJMI'

BECA-EOI#git branch -l
* RJMI
  main

BECA-EOI#git status
On branch RJMI
nothing to commit, working tree clean
```



MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO

EOI Escuela de
organización
industrial

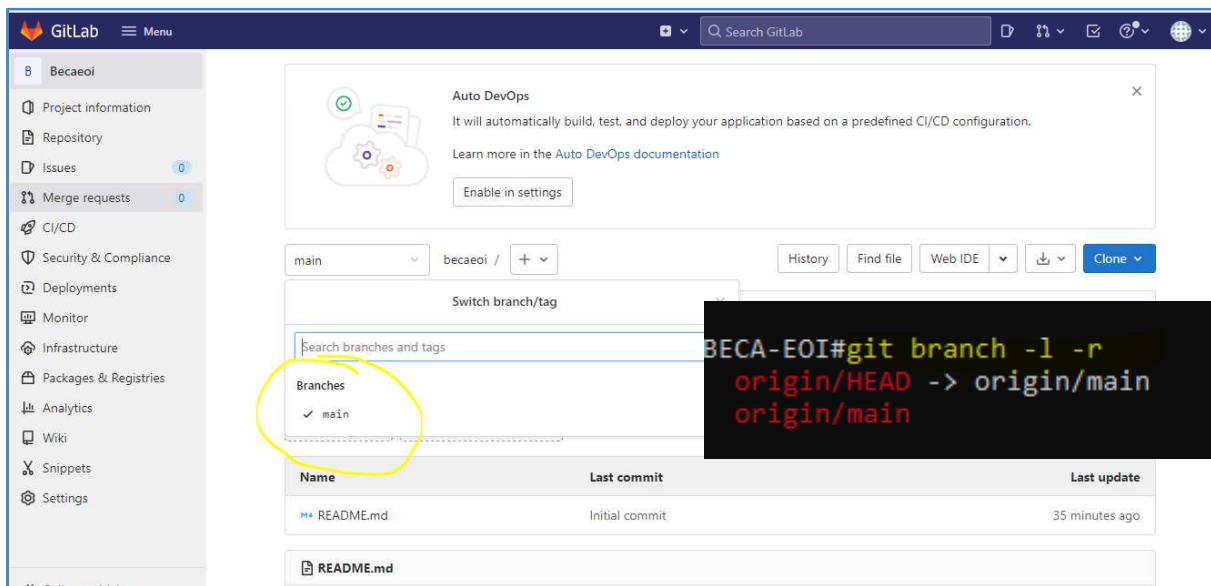


Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Comandos básicos

La rama que hemos creado sólo existe en nuestro repositorio Local. La dos ramas estarán totalmente relacionadas para que podamos gestionar los cambios correctamente desde nuestra rama Local a nuestra rama Remota



Comandos básicos : Sincronizar/Subir una rama nueva local a remoto

git push -u origin <rama>

Para sincronizar (subir) los cambios de nuestra rama local a la rama remota usaremos este comando :

git push -u origin RJMI (Recordad , usad el nombre de vuestra rama)

```
BECA-EOI#git branch -l -r
origin/HEAD -> origin/main
origin/main

BECA-EOI#git push -u origin RJMI
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: To create a merge request for RJMI, visit:
remote:   https://gitlab.com/e4320/becaeoi/-/merge_requests/new?merge_request%5Bsource_branch%5D=RJMI
remote:
To https://gitlab.com/e4320/becaeoi.git
 * [new branch]      RJMI -> RJMI
Branch 'RJMI' set up to track remote branch 'RJMI' from 'origin'.

BECA-EOI#git branch -l -r
origin/HEAD -> origin/main
origin/RJMI
origin/main
```

Comandos básicos : Cambiar a otra rama **git checkout <rama>** y borrar rama local **git branch -D <rama>**

Eliminar rama del repositorio local (No podemos estar situados en la rama que vamos a eliminar, antes tendremos que situarnos en otra rama)

```
BECA-EOI#git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

BECA-EOI#git branch -l
  RJMI
* main

BECA-EOI#git branch -D RJMI
Deleted branch RJMI (was 796cd53).

BECA-EOI#git branch -l
* main
```

Comandos básicos : Borrar rama remota **git push origin -d <rama>**

Eliminar rama del repositorio remoto

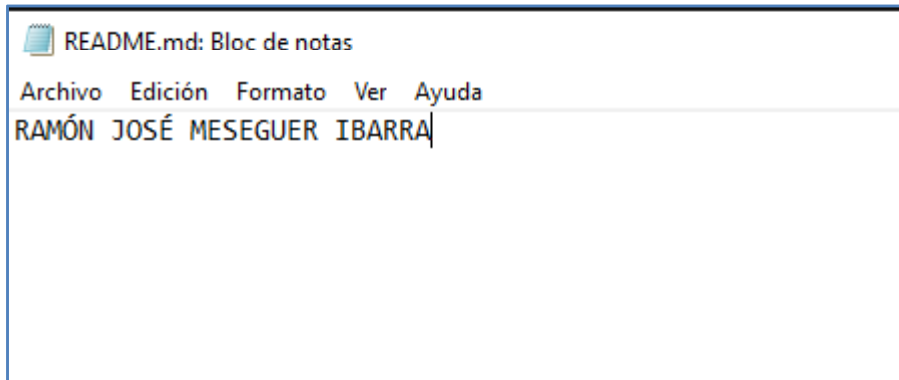
```
BECA-EOI#git branch -l -r
origin/HEAD -> origin/main
origin/RJMI
origin/main

BECA-EOI#git push origin -d RJMI
To https://gitlab.com/e4320/becaeoi.git
- [deleted]          RJMI

BECA-EOI#git branch -l -r
origin/HEAD -> origin/main
origin/main
```

Comandos básicos : descargar cambios, añadir cambios y subirlos al repositorio remote
git pull, git add . , git commit –m <comentario>, git push

Modificar el fichero README.md, eliminar todo el contenido y dejar vuestro nombre (Podéis usar cualquier editor, notepad,)



Comandos básicos : descargar cambios, añadir cambios y subirlos al repositorio remote
git pull, git add . , git commit -m <comentario>, git push

Una vez modificado veremos el estado de los cambios con **git status**

```
BECA-EOI#git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

Nos está indicando que el fichero ha cambiado

Comandos básicos : descargar cambios, añadir cambios y subirlos al repositorio remote
git pull, git add . , git commit –m <comentario>, git push

Para sincronizar los cambios del repositorio local con el repositorio remoto tenemos que hacer 4 cosas :

- 1) Descargar primeramente los cambios del repositorio remoto a nuestro repositorio local : **git pull**
- 2) Indicar que ficheros pasan al staged área : **git add <fichero o directorio>** (Preparados para subir al repositorio remoto)
- 3) Indicar comentario de subida para poder identificarlo: **git commit –m <comentario>**
- 4) Subir cambios al repositorio remoto : **git push**

Comandos básicos : descargar cambios, añadir cambios y subirlos al repositorio remoto
git pull, git add . , git commit -m <comentario>, git push

Para sincronizar los cambios del repositorio local con el repositorio remoto tenemos que hacer 4 cosas :

```
BECA-EOI#git status
On branch RJMI
Your branch is up to date with 'origin/RJMI'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

BECA-EOI#git pull
Already up to date.

BECA-EOI#git add .

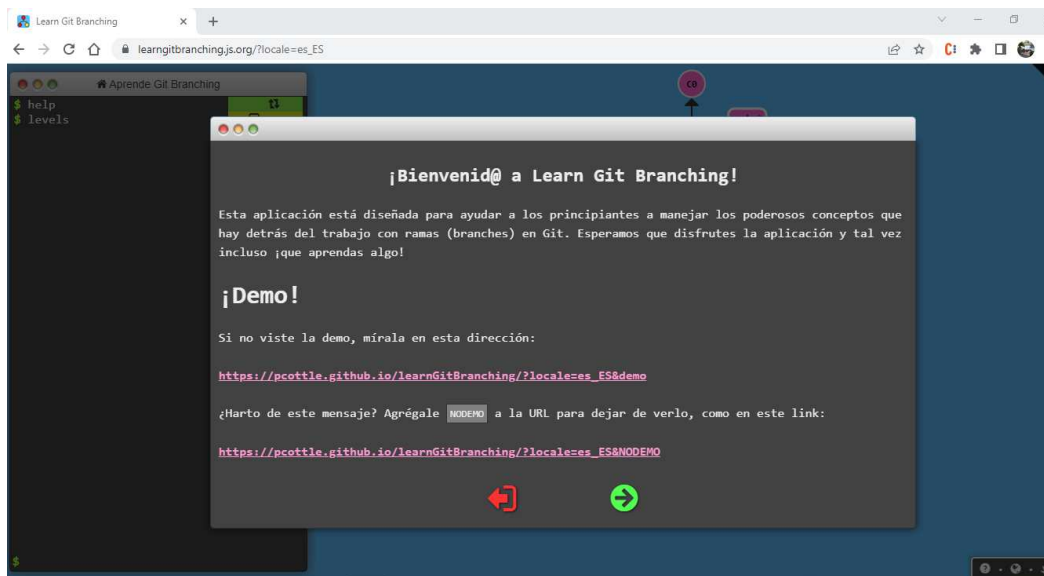
BECA-EOI#git commit -m "Modificado fichero Readme"
[RJMI 8029824] Modificado fichero Readme
1 file changed, 1 insertion(+), 92 deletions(-)
rewrite README.md (100%)

BECA-EOI#git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (1/1), done.
Writing objects: 100% (3/3), 278 bytes | 34.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: To create a merge request for RJMI, visit:
remote:   https://gitlab.com/e4320/becaeoi/-/merge_requests/new?merge_request%5Bsource_branch%5D=RJMI
remote:
To https://gitlab.com/e4320/becaeoi.git
796cd53..8029824 RJMI -> RJMI
```


Repaso de los principales comandos GIT

Vamos a revisar con un tutorial interactivo todos los comandos Git con los que trabajaremos, los que hemos visto y alguno más

https://learngitbranching.js.org/?locale=es_ES



06

Mergeos y conflictos



GOBIERNO
DE ESPAÑA

MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO



Escuela de
organización
industrial

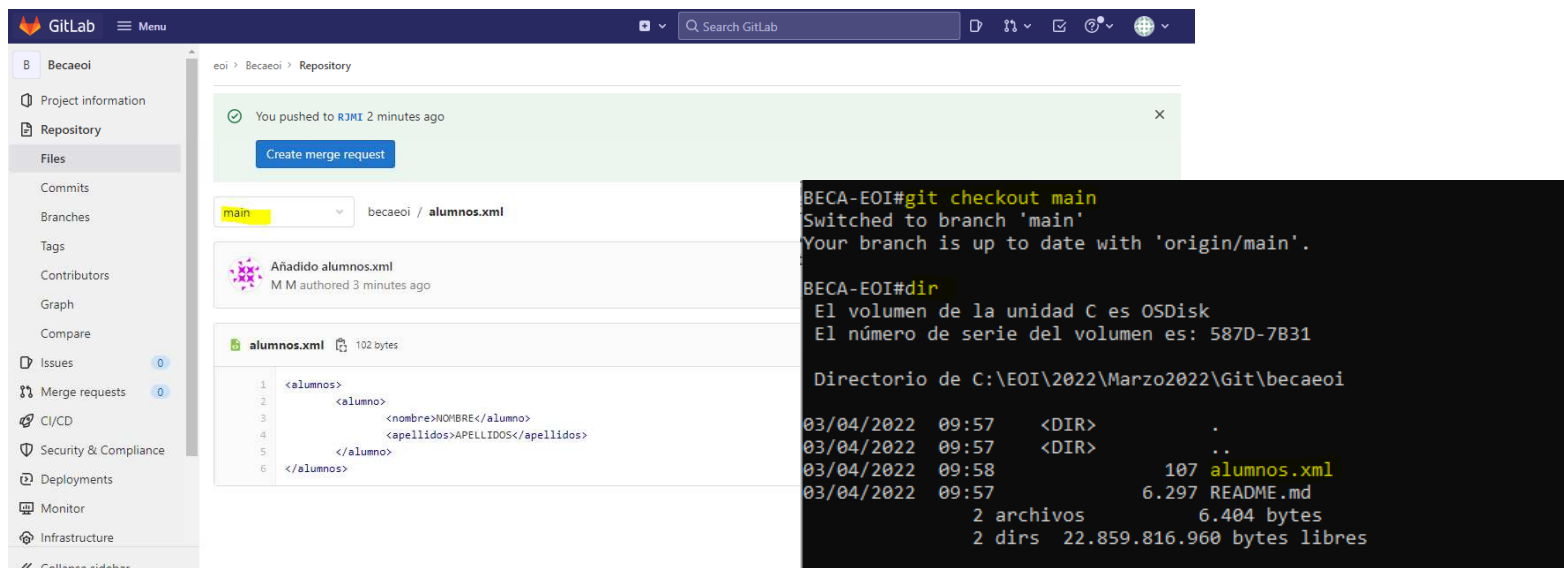


Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Merge y Conflictos

Tenemos este fichero **alumnos.xml** en nuestra rama local, porque la hemos clonado de la rama **main** en la que el fichero ya existía



The image shows a GitLab repository interface for 'Becaeoi' and a terminal window. The repository page displays a commit 'Añadido alumnos.xml' by 'M M' 3 minutes ago. The file 'alumnos.xml' is shown with its content:

```
1 <alumnos>
2   <alumno>
3     <nombre>NOMBRE</alumno>
4     <apellidos>APELLIDOS</apellidos>
5   </alumno>
6 </alumnos>
```

The terminal window shows the following commands and output:

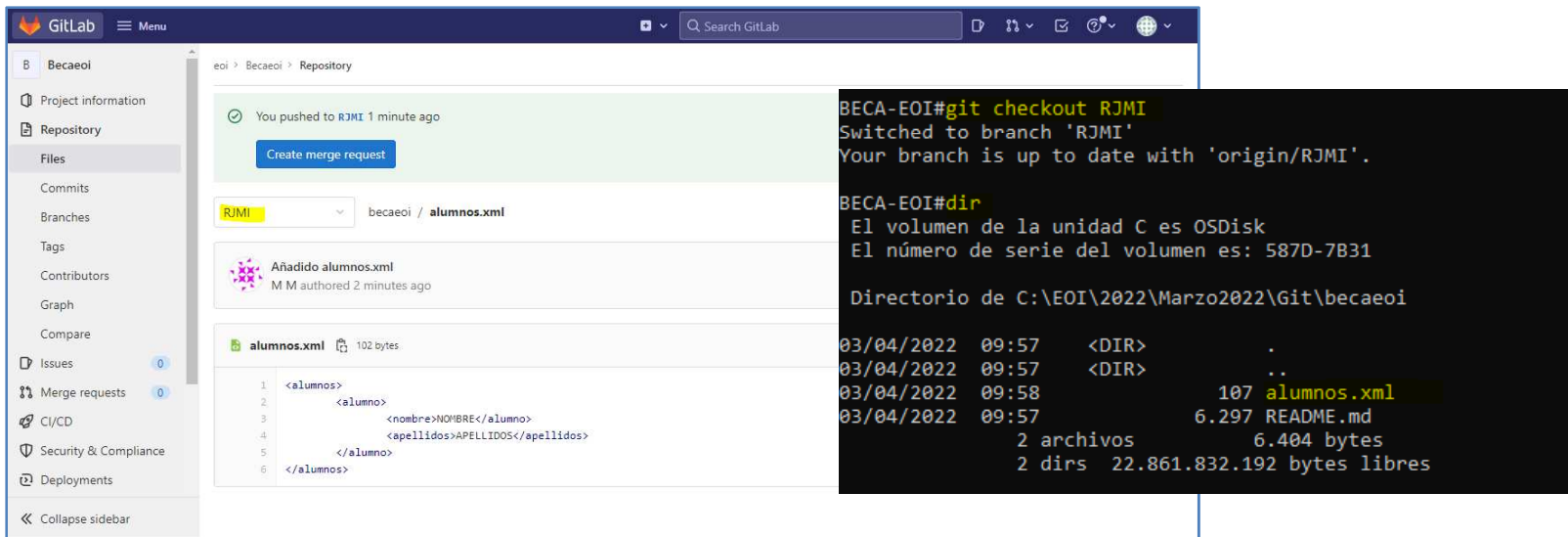
```
BECA-EOI#git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

BECA-EOI#dir
El volumen de la unidad C es OSDisk
El número de serie del volumen es: 587D-7B31

Directorio de C:\EOI\2022\Marzo2022\Git\becaeoi
03/04/2022 09:57 <DIR> .
03/04/2022 09:57 <DIR> ..
03/04/2022 09:58      107 alumnos.xml
03/04/2022 09:57    6.297 README.md
                2 archivos    6.404 bytes
                2 dirs    22.859.816.960 bytes libres
```

Merge y Conflictos

Tenemos este fichero **alumnos.xml** en nuestra rama local, porque la hemos clonado de la rama **main** en la que el fichero ya existía



The screenshot shows the GitLab web interface for the repository 'Becaeoi'. The left sidebar contains navigation links: Project information, Repository, Files, Commits, Branches, Tags, Contributors, Graph, Compare, Issues (0), Merge requests (0), CI/CD, Security & Compliance, and Deployments. The main content area shows a commit to the file 'alumnos.xml' by user 'M M' 2 minutes ago. The file content is an XML snippet:

```
1 <alumnos>
2   <alumno>
3     <nombre>NOMBRE</nombre>
4     <apellidos>APELLIDOS</apellidos>
5   </alumno>
6 </alumnos>
```

Overlaid on the right is a terminal window showing the following commands and output:

```
BECA-EOI#git checkout RJMI
Switched to branch 'RJMI'
Your branch is up to date with 'origin/RJMI'.


BECA-EOI#dir
El volumen de la unidad C es OSDisk
El número de serie del volumen es: 587D-7B31

Directorio de C:\EOI\2022\Marzo2022\Git\becaeoi

03/04/2022 09:57 <DIR> .
03/04/2022 09:57 <DIR> ..
03/04/2022 09:58          107 alumnos.xml
03/04/2022 09:57        6.297 README.md
                2 archivos        6.404 bytes
                2 dirs  22.861.832.192 bytes libres
```

Merge y Conflictos

Ahora lo modificaremos en nuestra rama para que incluya sólo un elemento alumno con nuestro nombre

 alumnos: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<alumnos>
  <alumno>
    <nombre>Ramón José</nombre>
    <apellidos>Meseguer Ibarra</apellidos>
  </alumno>
</alumnos>
```

Merge y Conflictos

Después de modificarlo , subimos los cambios al repositorio remoto

```
BECA-EOI#git status
On branch RJMI
Your branch is up to date with 'origin/RJMI'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   alumnos.xml

no changes added to commit (use "git add" and/or "git commit -a")

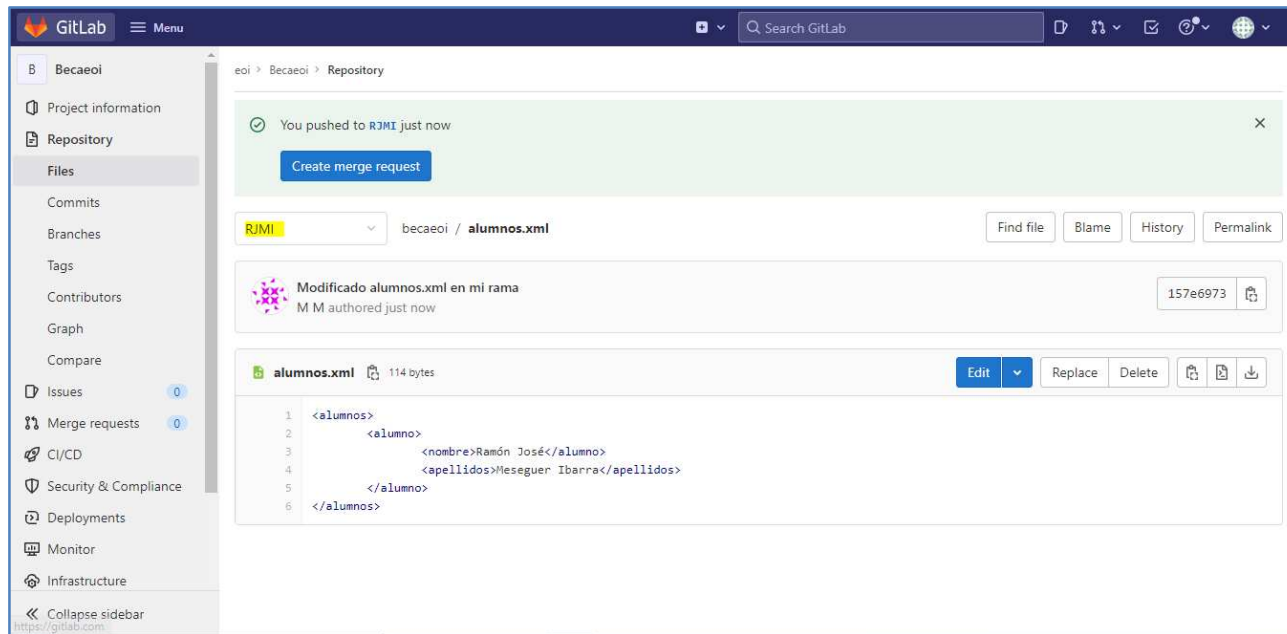
BECA-EOI#git add .

BECA-EOI#git commit -m "Modificado alumnos.xml en mi rama"
[RJMI 157e697] Modificado alumnos.xml en mi rama
 1 file changed, 2 insertions(+), 2 deletions(-)

BECA-EOI#git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 371 bytes | 92.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: To create a merge request for RJMI, visit:
remote:   https://gitlab.com/e4320/becaeoi/-/merge_requests/new?merge_request%5Bsource_branch%5D=RJMI
remote:
To https://gitlab.com/e4320/becaeoi.git
1e974db..157e697 RJMI -> RJMI
```

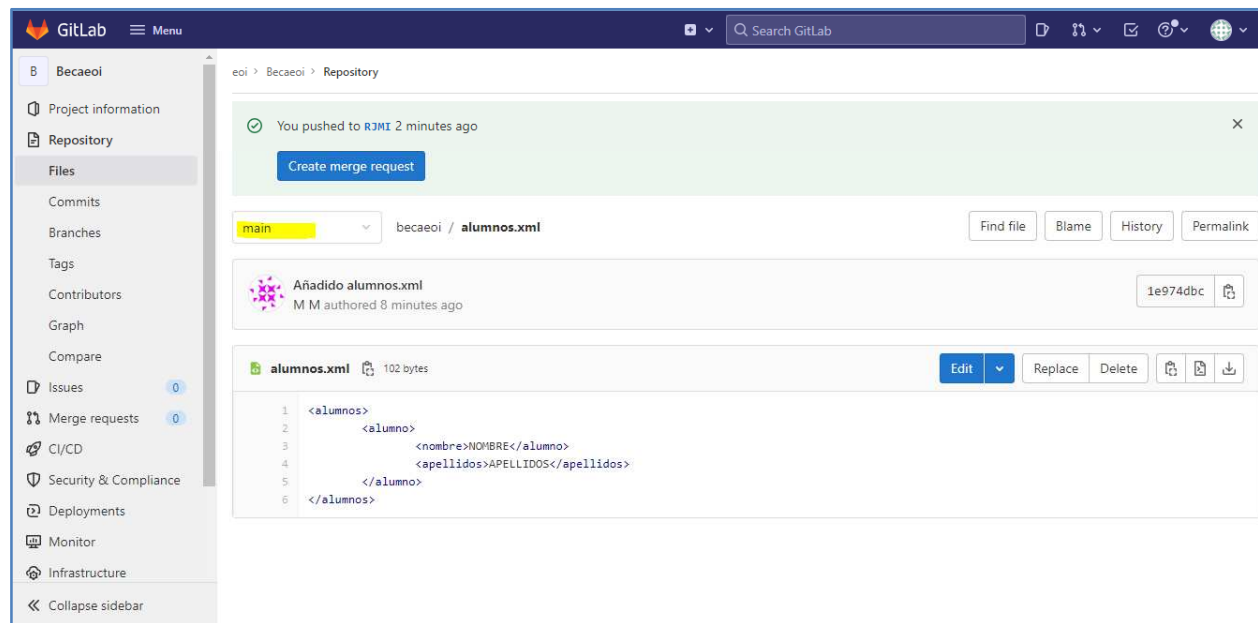
Merge y Conflictos

Después de modificarlo , subimos los cambios al repositorio remoto



Merge y Conflictos

Comprobamos que la rama **main** no tiene los cambios, sólo nuestra rama



Merge y Conflictos

Esto mismo lo podemos comprobar por comandos dentro del **cmd**

```
BECA-EOI#git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

BECA-EOI#more alumnos.xml
<alumnos>
  <alumno>
    <nombre>NOMBRE</nombre>
    <apellidos>APELLIDOS</apellidos>
  </alumno>
</alumnos>

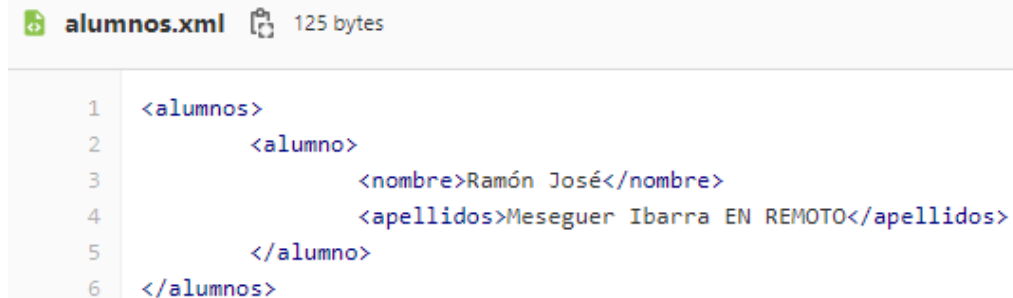
BECA-EOI#git checkout RJMI
Switched to branch 'RJMI'
Your branch is up to date with 'origin/RJMI'.

BECA-EOI#more alumnos.xml
<alumnos>
  <alumno>
    <nombre>Ramón Jos</nombre>
    <apellidos>Meseguer Ibarra</apellidos>
  </alumno>
</alumnos>
```

Merge y Conflictos

Ahora modificaremos en el repositorio remoto el fichero y sin descargar los cambios lo modificaremos en el repositorio local e intentaremos subir los cambios, vamos a ver qué es lo que sucede.

Cambios realizados en remoto




alumnos.xml 125 bytes

```
1 <alumnos>
2   <alumno>
3     <nombre>Ramón José</nombre>
4     <apellidos>Meseguer Ibarra EN REMOTO</apellidos>
5   </alumno>
6 </alumnos>
```

Merge y Conflictos

Cambios realizados en local

 alumnos: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<alumnos>
  <alumno>
    <nombre>Ramón José</nombre>
    <apellidos>Meseguer Ibarra CAMBIOS EN LOCAL</apellidos>
  </alumno>
</alumnos>
```

Merge y Conflictos

E intentamos subir los cambios

```
BECA-EOI#git status
On branch RJMI
Your branch is up to date with 'origin/RJMI'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   alumnos.xml

no changes added to commit (use "git add" and/or "git commit -a")

BECA-EOI#git add .

BECA-EOI#git commit -m "Modificado alumnos.xml con CAMBIOS EN LOCAL"
[RJMI 493e765] Modificado alumnos.xml con CAMBIOS EN LOCAL
1 file changed, 1 insertion(+), 1 deletion(-)

BECA-EOI#git push
To https://gitlab.com/e4320/becaeoi.git
! [rejected]        RJMI -> RJMI (fetch first)
error: failed to push some refs to 'https://gitlab.com/e4320/becaeoi.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Merge y Conflictos

¿Qué está pasando? Nos indica que hay cambios en repositorio remoto que no tenemos descargados al repositorio local, tenemos que hacer **git pull** para descargar los cambios del repositorio remoto al repositorio local

```
BECA-EOI#git push
To https://gitlab.com/e4320/becaeoi.git
! [rejected]        RJMI -> RJMI (fetch first)
error: failed to push some refs to 'https://gitlab.com/e4320/becaeoi.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

BECA-EOI#git pull
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), 602 bytes | 10.00 KiB/s, done.
From https://gitlab.com/e4320/becaeoi
 157e697..f62b990  RJMI      -> origin/RJMI
Auto-merging alumnos.xml
CONFLICT (content): Merge conflict in alumnos.xml
Automatic merge failed; fix conflicts and then commit the result.
```

Merge y Conflictos

Ahora nos está indicando que tenemos conflictos para resolver y que debemos abrir el fichero y resolverlo

alumnos: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<alumnos>
```

```
  <alumno>
```

```
    <nombre>Ramón José</nombre>
```

```
<<<<<< HEAD
```

```
  <apellidos>Meseguer Ibarra CAMBIOS EN LOCAL</apellidos>
```

```
=====
```

```
  <apellidos>Meseguer Ibarra EN REMOTO</apellidos>
```

```
>>>>>> 5fffd78ee3734202c370dc507fa23d2d33cce524
```

```
  </alumno>
```


```
</alumnos>
```

Cambios hechos
en Local

Cambios hechos
en Remoto

Merge y Conflictos

Tenemos que modificar el fichero y dejarlo cómo queremos que esté , sin incluir <<<<< , ==== y >>>>>>

 alumnos: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<alumnos>
  <alumno>
    <nombre>Ramón José</nombre>
    <apellidos>Meseguer Ibarra CAMBIOS EN LOCAL</apellidos>
  </alumno>
</alumnos>
```

Merge y Conflictos

Y volvemos a realizar todo el proceso para la subida

```
BECA-EOI#git status
On branch RJMI
Your branch and 'origin/RJMI' have diverged,
and have 1 and 2 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   alumnos.xml

no changes added to commit (use "git add" and/or "git commit -a")

BECA-EOI#git add .

BECA-EOI#git commit -m "Resuelto conflicto alumnos.xml"
[RJMI 6525f3a] Resuelto conflicto alumnos.xml

BECA-EOI#git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 654 bytes | 327.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote:
remote: To create a merge request for RJMI, visit:
remote:   https://gitlab.com/e4320/becaeoi/-/merge_requests/new?merge_request%5Bsource_branch%5D=RJMI
remote:
To https://gitlab.com/e4320/becaeoi.git
   f62b990..6525f3a  RJMI -> RJMI
```



MINISTERIO
DE INDUSTRIA, COMERCIO
Y TURISMO

EOI Escuela de
organización
industrial



Unión Europea
Fondo Social Europeo
Iniciativa de Empleo Juvenil
El FSE invierte en tu futuro



Merge y Conflictos

Vamos a Mergear lo que hay en nuestra rama con la rama **main**

```
BECA-EOI#git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

BECA-EOI#git merge RJMI
Updating 1e974db..6525f3a
Fast-forward
 alumnos.xml | 6 +++--
 1 file changed, 3 insertions(+), 3 deletions(-)

BECA-EOI#git status
On branch main
Your branch is ahead of 'origin/main' by 5 commits.
  (use "git push" to publish your local commits)

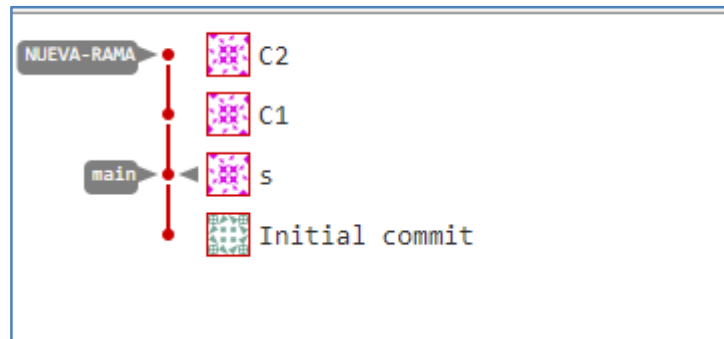
nothing to commit, working tree clean

BECA-EOI#git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://gitlab.com/e4320/becaeoi.git
 1e974db..6525f3a main -> main

BECA-EOI#more alumnos.xml
<alumnos>
  <alumno>
    <nombre>Ramón José</alumno>
    <apellidos>Meseguer Ibarra CAMBIOS EN LOCAL</apellidos>
  </alumno>
</alumnos>
```

Merge y Conflictos

Se puede usar otro tipo de merge llamado **rebase** que nos incluye todos los commits en la rama que le indiquemos



Merge y Conflictos

Aplicamos **rebase** sobre **main** de la rama **NUEVA-RAMA**, y vemos como se aplican los commits de **NUEVA-RAMA** a la rama **main**

```
BECA-EOI#git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

BECA-EOI#git rebase NUEVA-RAMA
Successfully rebased and updated refs/heads/main.

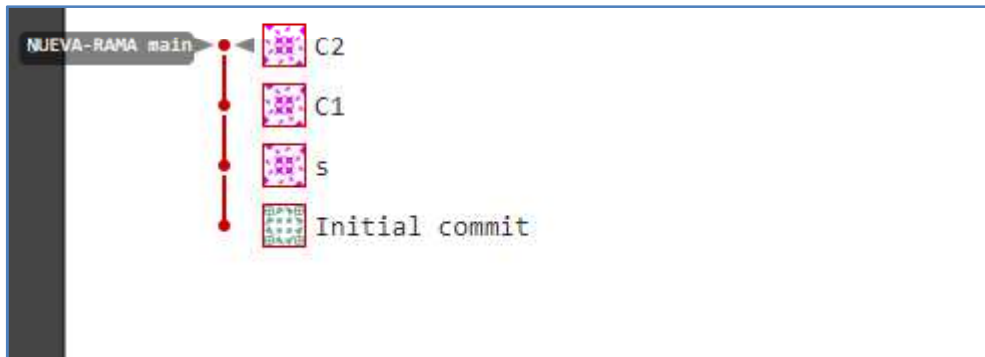
BECA-EOI#git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

BECA-EOI#git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://gitlab.com/e4320/becaeoi.git
90b6a4d..1a79df0  main -> main
```

Merge y Conflictos

Aplicamos **rebase** sobre **main** de la rama **NUEVA-RAMA**, y vemos como se aplican los commits de **NUEVA-RAMA** a la rama **main**



Ejercicio

Para automatizar la información sobre los alumnos de la Beca , vamos a modificar el fichero alumnos.xml y así probaremos entre todos lo aprendido.

Cada alumno, en su rama, debe cumplimentar las etiquetas con su información personal y el resultado debe ser un único fichero que contenga la información personal de cada uno de los alumnos.

Una vez tengamos los cambios realizados en nuestra rama Local deberemos integrarlos (mergearlos) con la rama **main**

El resultado final tiene que ser , que en la rama main el fichero alumnos.xml tenga todos los datos de los alumnos.

Se abrirá un BrainStorming para que los alumnos propongan cómo organizar el trabajo para conseguir el resultado indicado en el menor tiempo posible

```
<alumnos>
  <alumno>
    <nombre>Nombre ....</nombre>
    <apellidos>Apellidos....</apellido>
  </alumno>
  ....
</alumnos>
```

