

TRABAJO E.D: CONTROL DE VERSIONES GIT

REALIZADO POR:
ANTONIO MARTINEZ NAVARRO

CONTENIDO

1. INTRODUCCIÓN.....	3
2. CREAR UN REPOSITORIO REMOTO EN GITLAB.....	3
3. EMPEZAR A USAR GIT BASH DENTRO DE NUESTRO PROYECTO.....	4
4. SUBIR NUESTRO PROYECTO ACTUAL AL REPOSITORIO REMOTO. ..	8
5. CREAR UNA RAMA NUEVA PARA HACER MODIFICACIONES.....	9
6. MODIFICACIÓN DEL PROYECTO ORIGINAL.	9
7. POSIBLE CASO EN LA VIDA REAL	11
8. BIBLIOGRAFIA.....	12
9. ENLACE AL PROYECTO.....	12

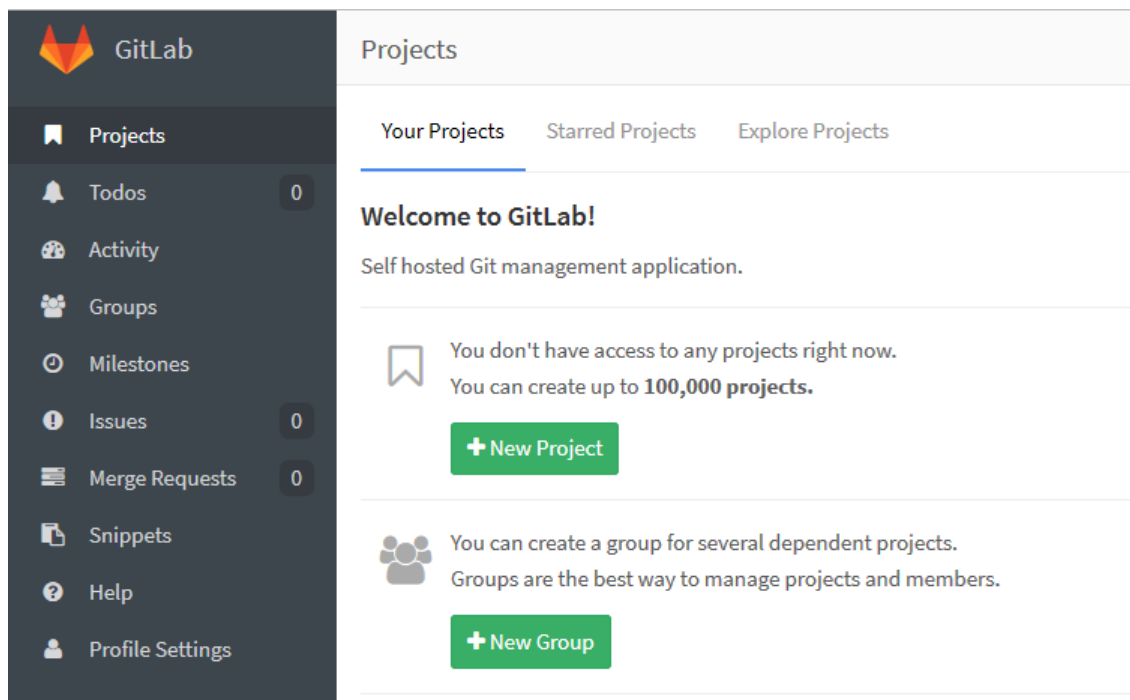
1. INTRODUCCIÓN

Esto es una demostración de cómo usar un software de Control de Versiones como GIT. Este software es muy útil a la hora de trabajar conjuntamente en un mismo proyecto.

Vamos a ver unos pasos para entender mejor cómo funciona GIT.

2. CREAR UN REPOSITORIO REMOTO EN GITLAB.

Lo primero que tenemos que hacer es entrar en la página de GITLAB y darse de alta para después crear un repositorio en el cual vamos a trabajar.



Pinchamos en New Project.

New Project

Project path

Want to house several dependent projects under the same namespace? [Create a group](#)

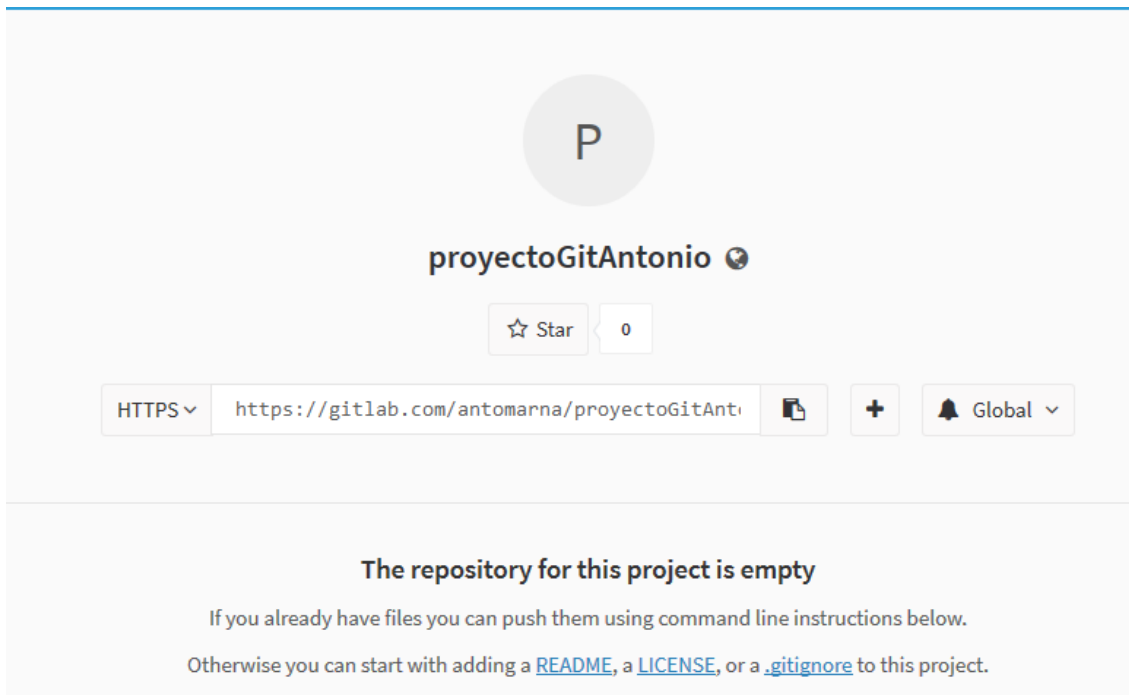
Import project from

Description (optional)

Visibility Level (?)

- Private**
Project access must be granted explicitly to each user.
- Internal**
The project can be cloned by any logged in user.
- Public**
The project can be cloned without any authentication.

En el cuadro de texto ponemos el nombre a nuestro repositorio. Pulsamos en **Create Project** y nos aparece esta pantalla:



Ya tenemos el repositorio de trabajo creado y la URL que utilizaremos desde GIT Bash para subir cambios.

3. EMPEZAR A USAR GIT BASH DENTRO DE NUESTRO PROYECTO.

Abrimos GIT Bash y nos vamos dentro del directorio **src** del proyecto.

```
MINGW64:/c/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio
$ cd src/
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src
$
```

Ahora ejecutamos el comando **git init**.

```
MINGW64:/c/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src
$ git init
Initialized empty Git repository in C:/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src/.git/
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$
```

Automáticamente se crea la rama **master**.

Ahora podemos ejecutar **git status** para ver el estado de nuestro proyecto.

```
MINGW64:/c/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        modelo/
        proyectoGit/

nothing added to commit but untracked files present (use "git add" to track)
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$
```

Como podemos ver hay dos directorios en rojo, eso significa que no se han añadido a la rama en la que estamos actualmente (master). Para añadirlos ejecutamos **git add** .

Podemos comprobar el estado actual con **git status**.

```
MINGW64:/c/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   modelo/Alumno.java
    new file:   modelo/ConexionBD.java
    new file:   modelo/Curso.java
    new file:   proyectoGit/PanelAlumnoAlta.form
    new file:   proyectoGit/PanelAlumnoAlta.java
    new file:   proyectoGit/PanelAlumnoBaja.form
    new file:   proyectoGit/PanelAlumnoBaja.java
    new file:   proyectoGit/PanelAlumnoListado.form
    new file:   proyectoGit/PanelAlumnoListado.java
    new file:   proyectoGit/PanelCursoAlta.form
    new file:   proyectoGit/PanelCursoAlta.java
    new file:   proyectoGit/PanelCursoBaja.form
    new file:   proyectoGit/PanelCursoBaja.java
    new file:   proyectoGit/PanelCursoListado.form
    new file:   proyectoGit/PanelCursoListado.java
    new file:   proyectoGit/T12p03.java
    new file:   proyectoGit/T12p03_GUI.form
    new file:   proyectoGit/T12p03_GUI.java

ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$
```

Ahora se han añadido todos esos ficheros. Lo que debemos hacer a continuación es confirmar los cambios. Para ello ejecutamos **git commit -m "mensaje"**.

Comprobamos el estado y nos debe salir lo siguiente:

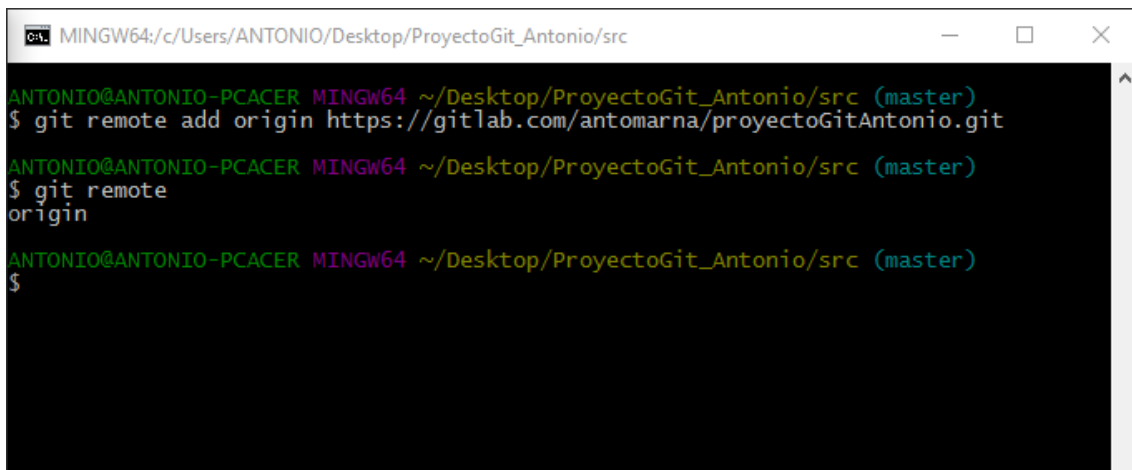
```
MINGW64:/c/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git status
On branch master
nothing to commit, working directory clean

ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$
```

4. SUBIR NUESTRO PROYECTO ACTUAL AL REPOSITORIO REMOTO.

Lo primero que tenemos que hacer es decirle la ruta en la que se encuentra el repositorio remoto. Para esto ejecutamos:

git remote add <nombre> <url>

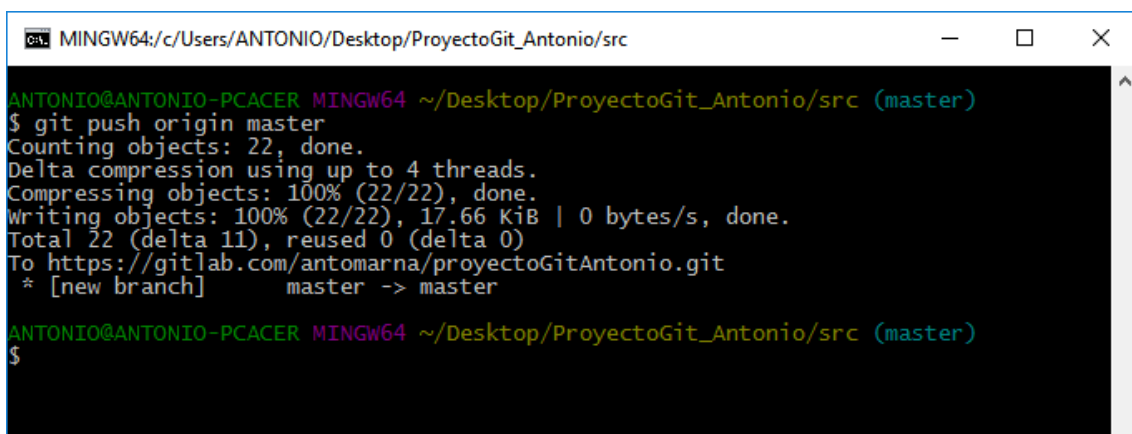


```
MINGW64:/c:/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git remote add origin https://gitlab.com/antomarna/proyectoGitAntonio.git
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git remote
origin
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$
```

Yo he utilizado de nombre **origin**. A partir de ahora utilizaré este nombre en vez de la url.

Ahora ya podemos subir el proyecto a nuestro repositorio remoto ejecutando el siguiente comando:

git push <nombre> <nombre-rama>



```
MINGW64:/c:/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git push origin master
Counting objects: 22, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (22/22), done.
Writing objects: 100% (22/22), 17.66 KiB | 0 bytes/s, done.
Total 22 (delta 11), reused 0 (delta 0)
To https://gitlab.com/antomarna/proyectoGitAntonio.git
 * [new branch]      master -> master
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$
```

Si entramos en la página de gitlab podemos ver lo que acabamos de subir.



Antonio Martinez pushed new branch master at Antonio Martinez / proyectoGitAntonio
ad777b50 · Primera confirmacion

5. CREAR UNA RAMA NUEVA PARA HACER MODIFICACIONES.

Si queremos hacer modificaciones a nuestro proyecto sin afectar al proyecto original podemos crear una nueva rama de trabajo, y al final, si queremos, fusionar el trabajo modificado al original (master).

Para crear una nueva rama tenemos que ejecutar el comando:

git branch <nombre-rama>

Para posicionarnos en esa rama ejecutamos el comando:

git checkout <nombre-rama>

```
MINGW64:/c:/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git branch
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git checkout developer
Switched to branch 'developer'
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (developer)
$ git branch
* developer
  master
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (developer)
$
```

Para ver las todas las ramas ejecutamos **git branch**. El asterisco nos indica en que rama nos encontramos actualmente.

Ahora ya podemos empezar a hacer nuestras modificaciones en el proyecto desde nuestro entorno de desarrollo.

6. MODIFICACIÓN DEL PROYECTO ORIGINAL.

En el proyecto utilizado para esta práctica quiero añadir un nuevo atributo a los alumnos, que será si es repetidor o no. Para ello añado un check-box en el JPanel del alta de alumno y realizo las modificaciones necesarias en las clases involucradas.

Una vez realizados los cambios, volvemos al bash y ejecutamos **git status** para ver el estado del proyecto en la rama actual.

```
MINGW64:/c:/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (developer)
$ git status
On branch developer
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    proyectoGit/T12p03.java

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

        modified:   modelo/Alumno.java
        modified:   modelo/ConexionBD.java
        modified:   proyectoGit/PanelAlumnoAlta.form
        modified:   proyectoGit/PanelAlumnoAlta.java
        modified:   proyectoGit/PanelAlumnoListado.java
```

Nos muestra todos los ficheros que han sido modificados y están por confirmar.

Lo que tenemos que hacer a continuación es añadir los cambios con el comando **git add .** y confirmar los cambios con **git commit -m "mensaje"**.

Ahora lo que podemos hacer es, o bien subir el proyecto desde la rama actual (developer), o fusionar esta rama y subir el original (master).

Para fusionar la rama **developer** con la rama **master**, nos situamos en la rama master (**git checkout master**) y ejecutamos el siguiente comando:

git merge --no-ff <nombre de la rama a fusionar>

En mi caso el nombre de la rama a fusionar es **developer**.

```
MINGW64:/c:/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git merge --no-ff developer
Removing proyectoGit/T12p03.java
Merge made by the 'recursive' strategy.
 modelo/Alumno.java          | 16 ++++-
 modelo/ConexionBD.java     |  3 +-
 proyectoGit/PanelAlumnoAlta.form | 25 ++++++-
 proyectoGit/PanelAlumnoAlta.java | 28 ++++++-
 proyectoGit/PanelAlumnoListado.java |  5 +-
 proyectoGit/T12p03.java     | 120 -----
 6 files changed, 66 insertions(+), 131 deletions(-)
 delete mode 100644 proyectoGit/T12p03.java

ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$
```

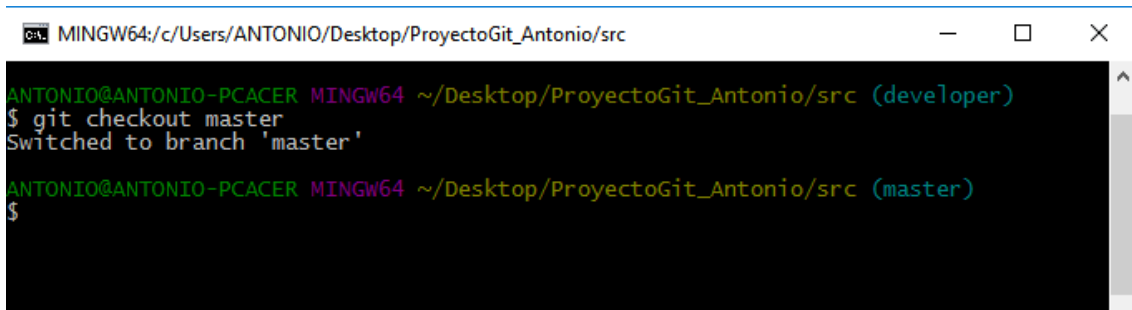
Para subir el proyecto actual ejecutamos **git push origin master**

Si queremos ver el registro de nuestra actividad ejecutamos el comando **git log**.

7. POSIBLE CASO EN LA VIDA REAL

Supongamos que estamos trabajando en una rama para hacer algún cambio en nuestro proyecto original y nos llaman para decirnos que tenemos que arreglar un problema en el original.

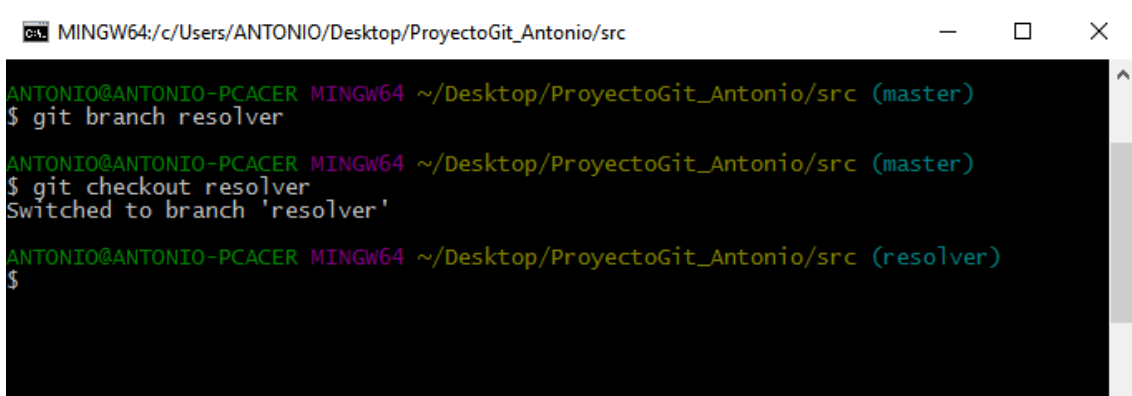
Lo primero que debemos hacer es volver a la rama de producción original (master).



```
MINGW64:/c/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (developer)
$ git checkout master
Switched to branch 'master'

ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$
```

Lo siguiente es crear una nueva rama para solucionar el problema.

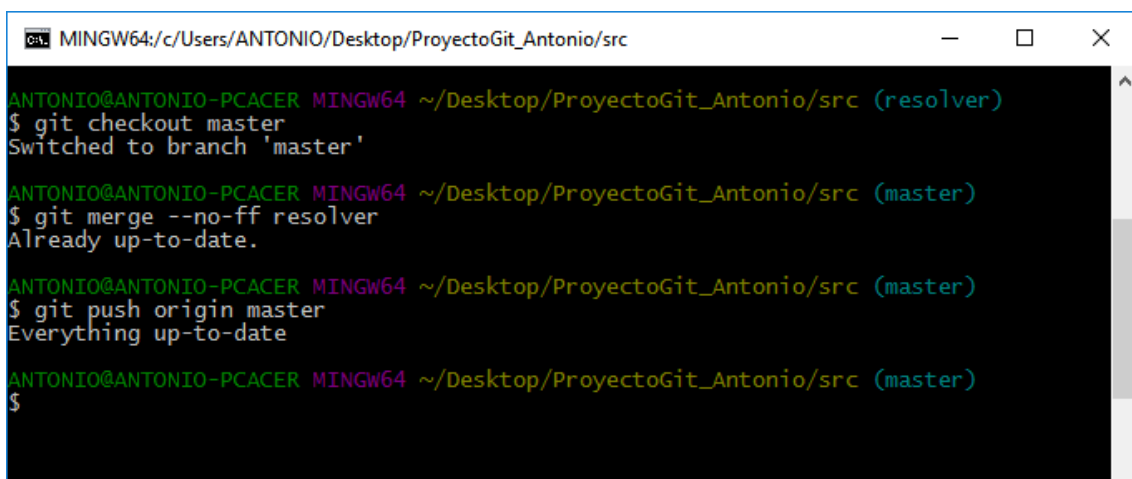


```
MINGW64:/c/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git branch resolver

ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git checkout resolver
Switched to branch 'resolver'

ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (resolver)
$
```

Hacer todos los cambios para resolver el problema y tras esto fusionar esta rama con el comando **git merge** y subirlo a la rama de producción.



```
MINGW64:/c/Users/ANTONIO/Desktop/ProyectoGit_Antonio/src
ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (resolver)
$ git checkout master
Switched to branch 'master'

ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git merge --no-ff resolver
Already up-to-date.

ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$ git push origin master
Everything up-to-date

ANTONIO@ANTONIO-PCACER MINGW64 ~/Desktop/ProyectoGit_Antonio/src (master)
$
```

Una vez hecho esto podemos volver a nuestra rama de desarrollo y continuar con nuestro trabajo.

8. BIBLIOGRAFIA.

- Apuntes de Cubichilazaro:
<http://www.piradoiv.com/blog/aprende-usar-git-con-este-ejemplo-real>
<http://www.piradoiv.com/blog/usando-git-con-tus-companeros-de-proyecto>
- Documentación de la página oficial de GIT
<https://git-scm.com/book/es/v1/Fundamentos-de-Git>

9. ENLACE AL PROYECTO

- <https://gitlab.com/antomarna/proyectoGitAntonio.git>