

# Guía rápida de Git en modo gráfico

Hipólito Guzmán Miranda  
Universidad de Sevilla

# Instalación

En windows:

- Descargar el instalador: <https://git-scm.com/download/>
- IMPORTANTE: Hay **dos** momentos de la instalación en el que **no** debéis marcar la opción por defecto:
  - Editor por defecto: poner Nano (es sencillo: Ctrl+O es guardar, Ctrl+X es salir) u otro que conozcáis
  - Rama inicial por defecto: main en vez de master

Git 2.40.0 Setup

### Choosing the default editor used by Git

Which editor would you like Git to use?

Use Vim (the ubiquitous text editor) as Git's default editor

The [Vim editor](#), while powerful, [can be hard to use](#). Its user interface is unintuitive and its key bindings are awkward.

**Note:** Vim is the default editor of Git for Windows only for historical reasons, and it is highly recommended to switch to a modern GUI editor instead.

**Note:** This will leave the 'core.editor' option unset, which will make Git fall back to the 'EDITOR' environment variable. The default editor is Vim - but you may set it to some other editor of your choice.

Git 2.40.0 Setup

### Choosing the default editor used by Git

Which editor would you like Git to use?

Use the Nano editor by default

[GNU nano](#) is a small and friendly text editor running in the console window.

This is the recommended option for end users if no GUI editors are installed.



Git 2.40.0 Setup

### Adjusting the name of the initial branch in new repositories

What would you like Git to name the initial branch after "git init"?

**Let Git decide**

Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project [intends](#) to change this default to a more inclusive name in the near future.

**Override the default branch name for new repositories**

**NEW!** Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:

This setting does not affect existing repositories.

<https://gitforwindows.org/>

Back Next Cancel

Git 2.40.0 Setup

### Adjusting the name of the initial branch in new repositories

What would you like Git to name the initial branch after "git init"?

**Let Git decide**

Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project [intends](#) to change this default to a more inclusive name in the near future.

**Override the default branch name for new repositories**

**NEW!** Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:

This setting does not affect existing repositories.

<https://gitforwindows.org/>

Back Next Cancel 3



# Instalación

En Linux:

Se instala desde los repositorios de paquetes, por ejemplo:

- debian/ubuntu: `apt install git git-gui`
- centos/redhat: `yum install git git-gui`

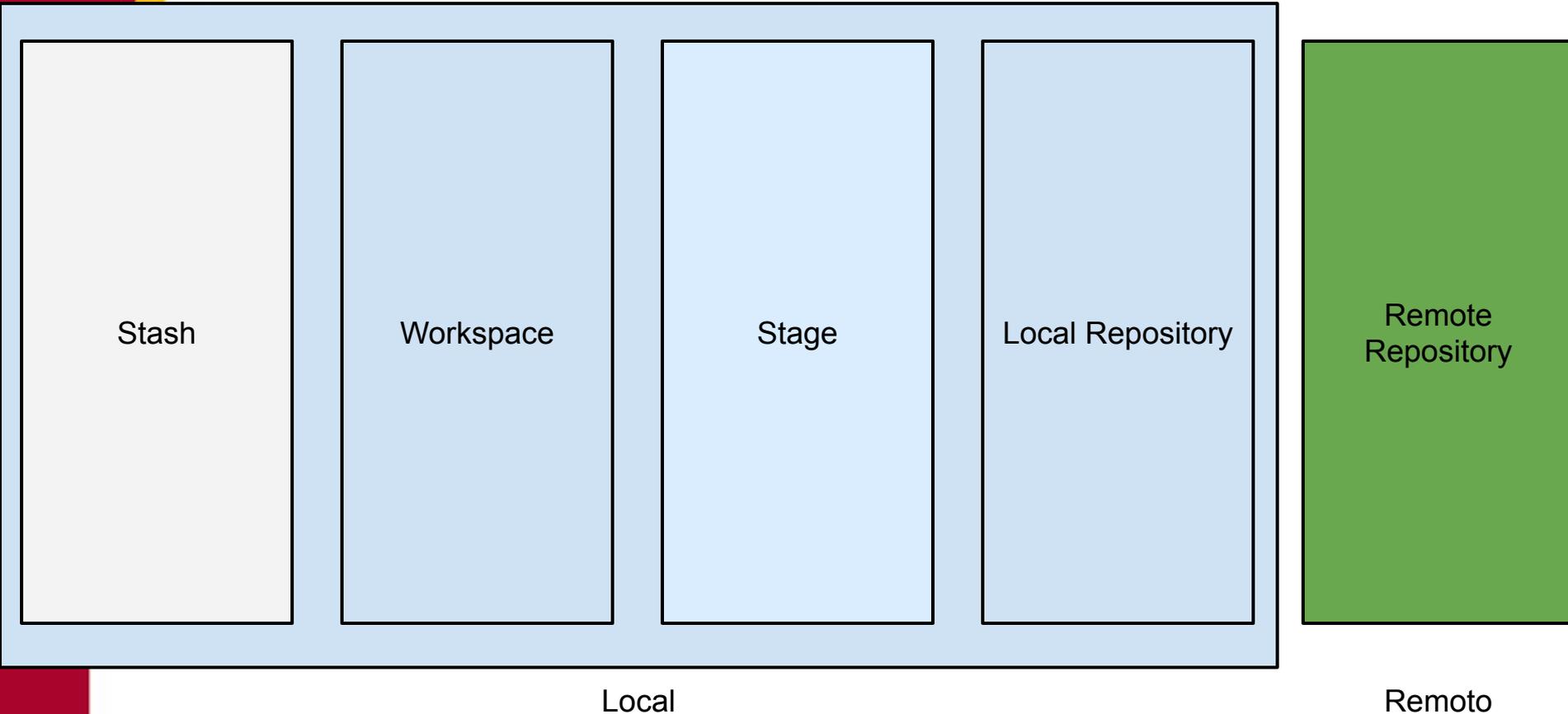
# Contenido

- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- Push
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout

# Contenido

- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- Push
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout

## Áreas



## Descripción

Espacio para esconder cambios temporalmente

Stash

Carpeta normal en nuestro ordenador

Workspace

Caché de los cambios que se añadirán al repo local

Stage

Repositorio local en nuestra máquina

Local Repository

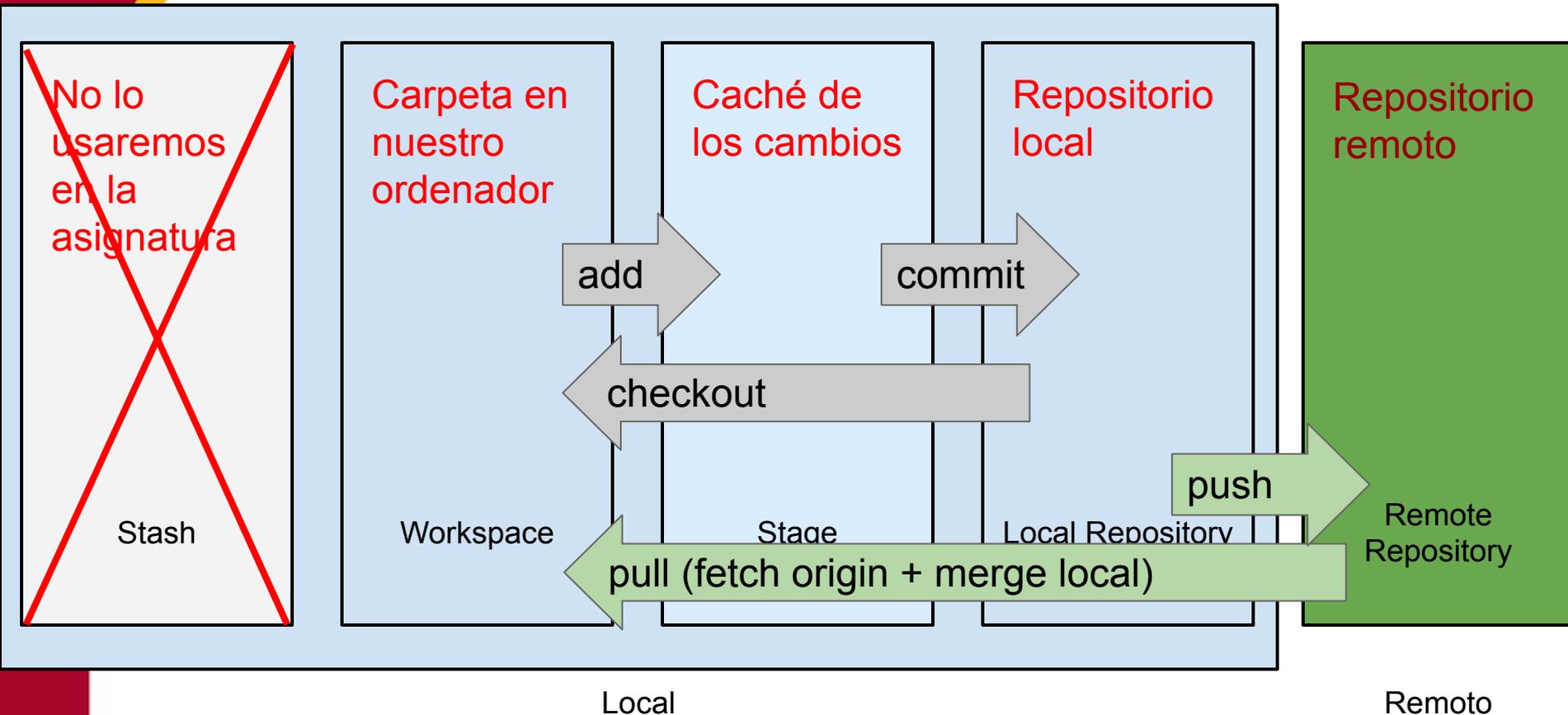
Repositorio externo, accesible por todos los miembros del grupo

Remote Repository

Local

Remoto

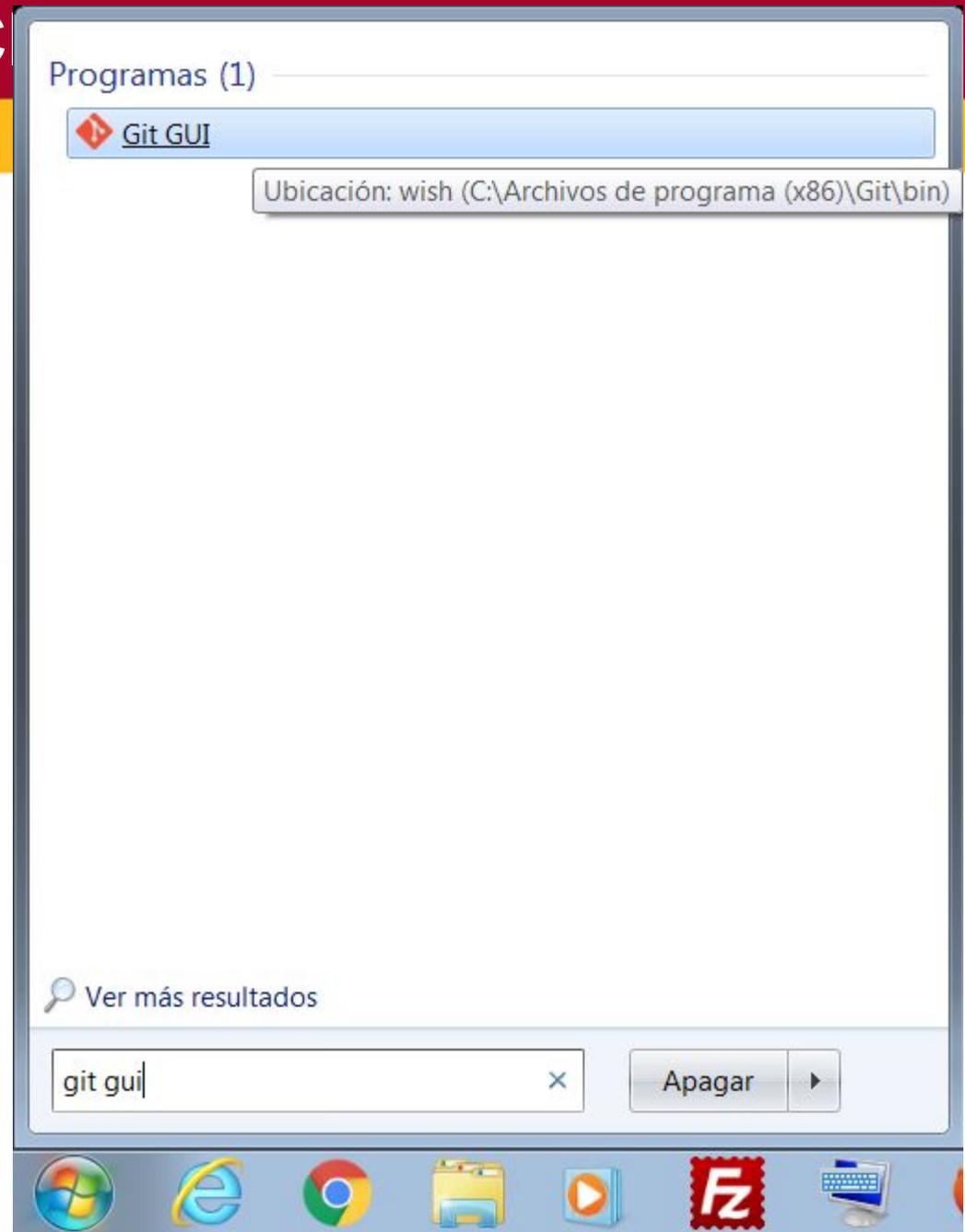
## Comandos



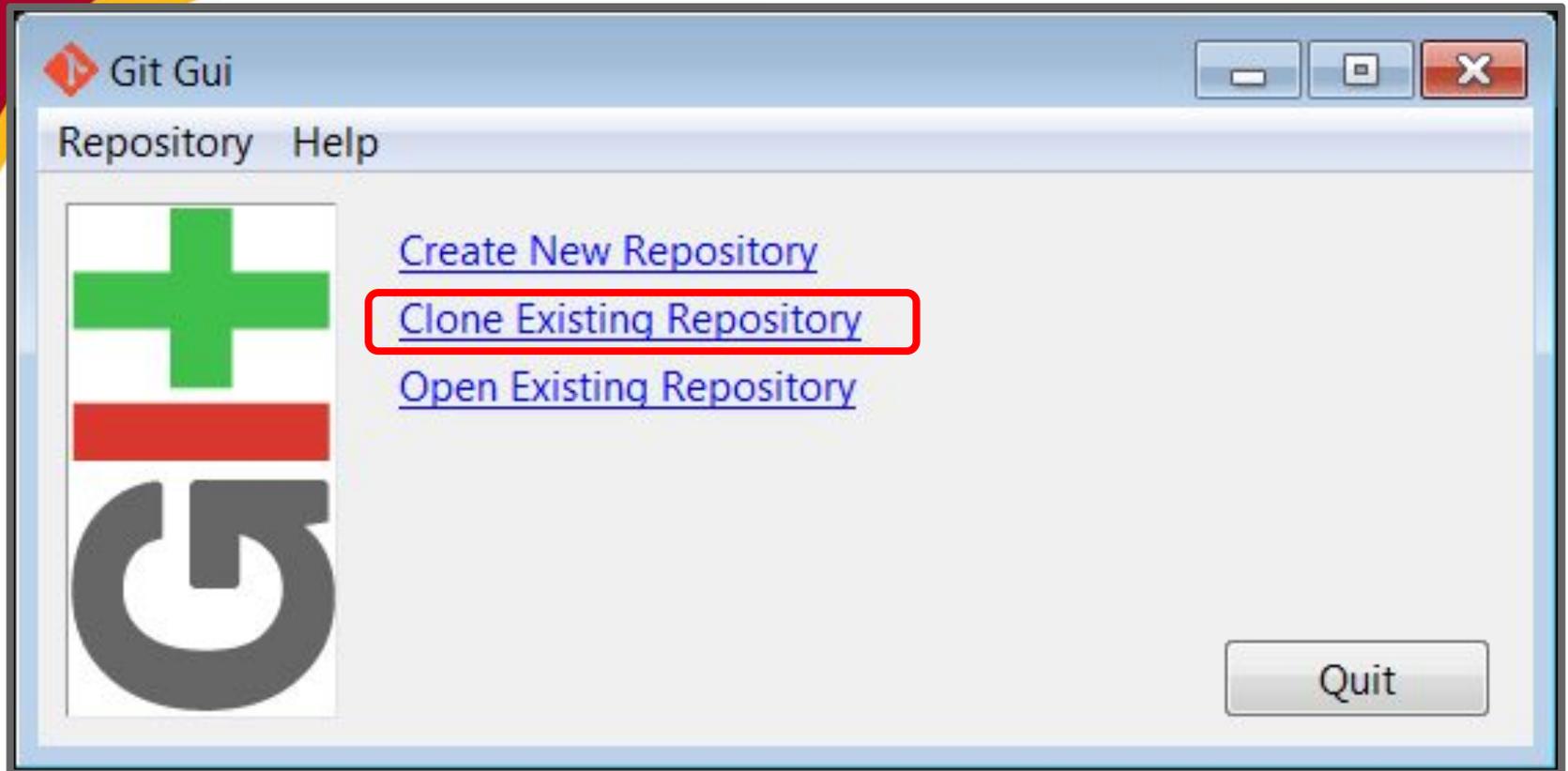
# Contenido

- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- Push
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout

# Abrimos Git GUI



# Clonar el repo



Navigation: pigiern > pi01 > pi01

**pi01** Project ID: 34392079

🔔 Star 0 Fork 0

1 Commit 1 Branch 0 Tags 61 KB Files 61 KB Storage

Repositorio para el grupo 01

**Auto DevOps**

It will automatically build, test, and deploy your application based on a predefined CI/CD configuration.

Learn more in the [Auto DevOps documentation](#)

[Enable in settings](#)

main pi01 / + [History](#) [Find file](#) [Web IDE](#) [Download](#) [Clone](#)

**Initial commit**

Hipólito Guzmán-Miranda authored 9 minutes ago

[Upload File](#) [README](#) [Add LICENSE](#) [Add CHANGELOG](#) [Add](#)

[Set up CI/CD](#) [Configure Integrations](#)

**Clone with SSH**

git@gitlab.com:pigiern/pi\_2022/

**Clone with HTTPS**

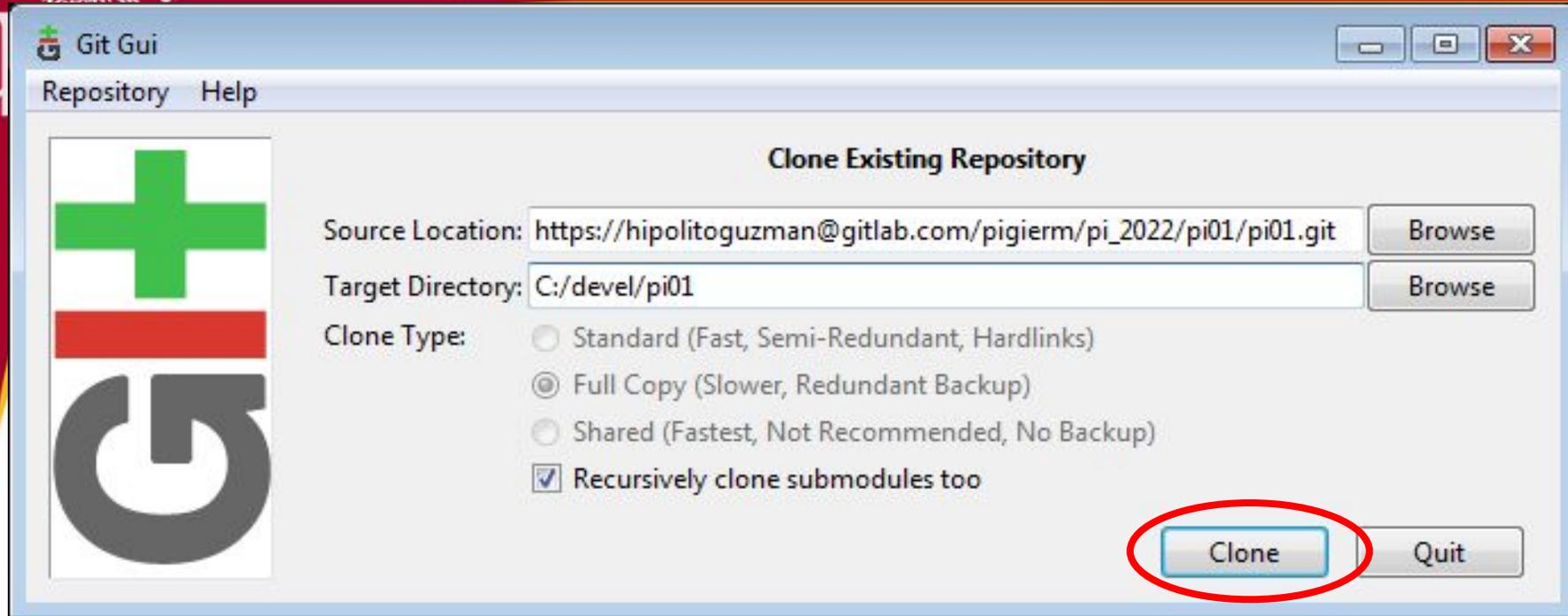
[m/pigiern/pi\\_2022/pi01/pi01.git](https://gitlab.com/pigiern/pi_2022/pi01/pi01.git)

**Open in your IDE**

[Copy URL](#)

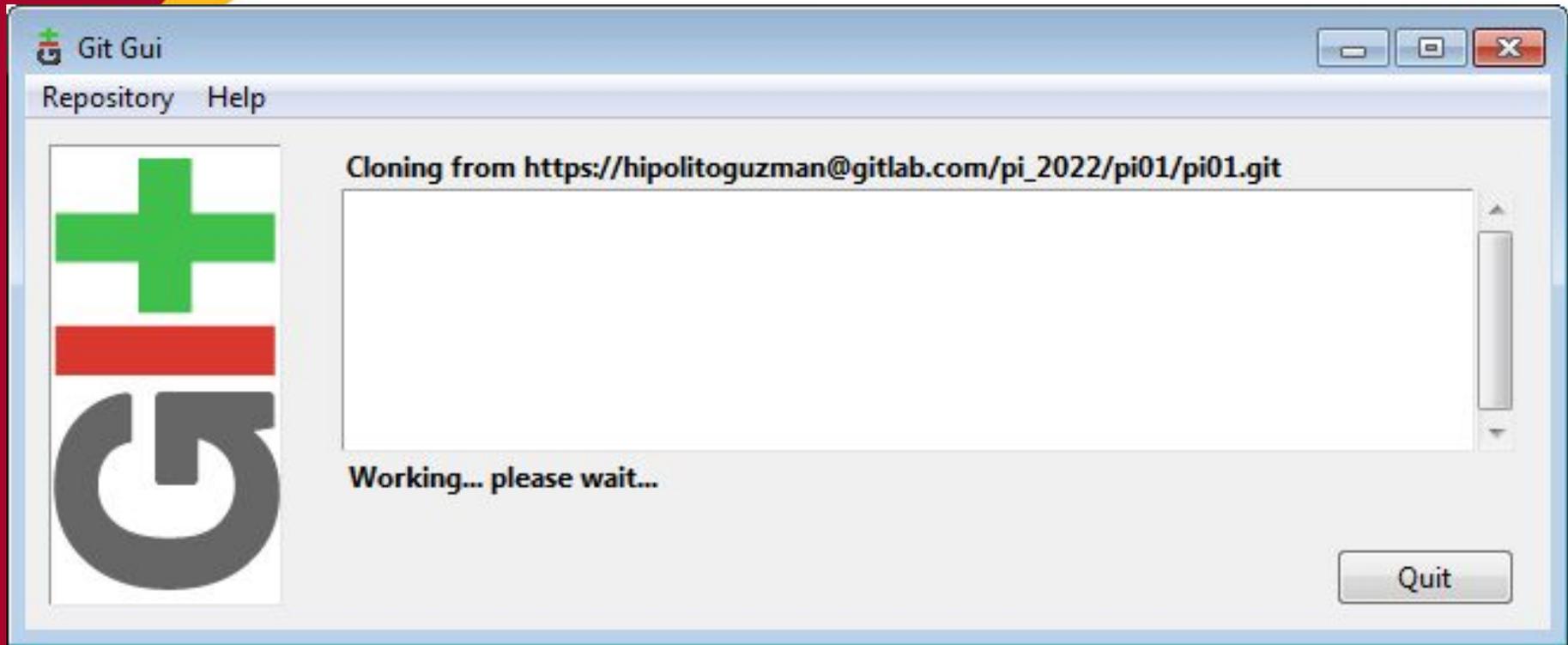
Necesitamos saber cuál es la dirección del repositorio para poder clonarlo

# Clonar el repo

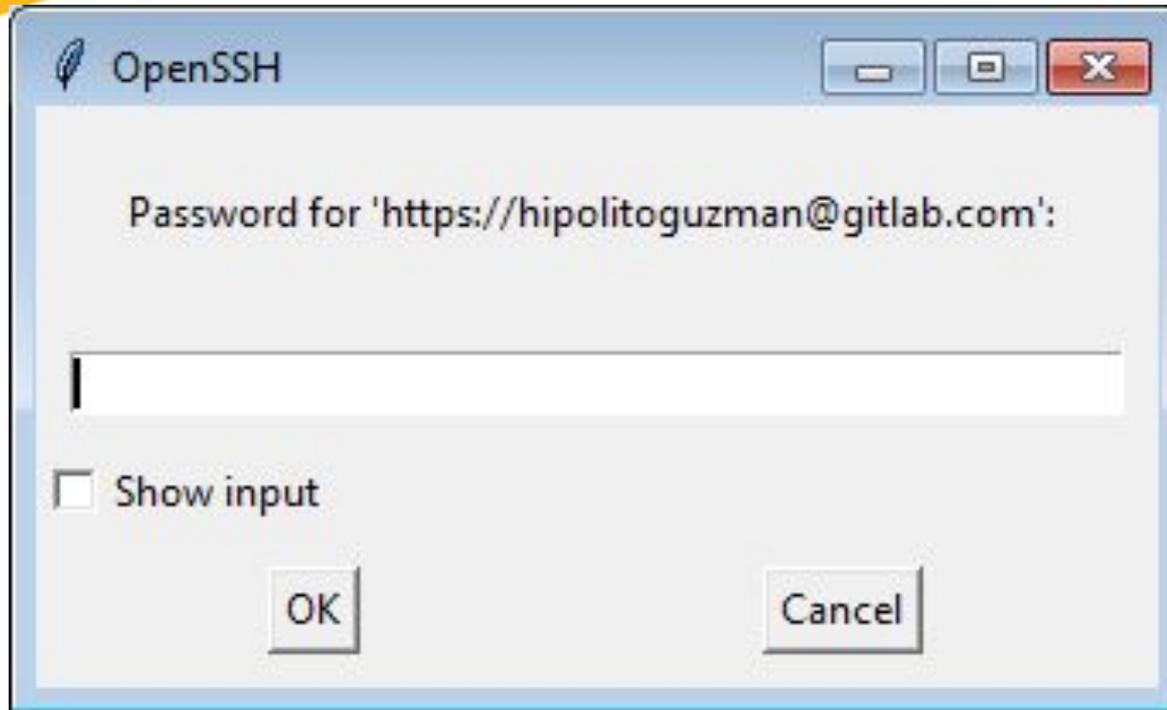


- Podemos pegar directamente el enlace copiado de gitlab en “Source location”
- El target directory será creado al clonar el repositorio (no debe existir de antes!)

# Clonar el repo



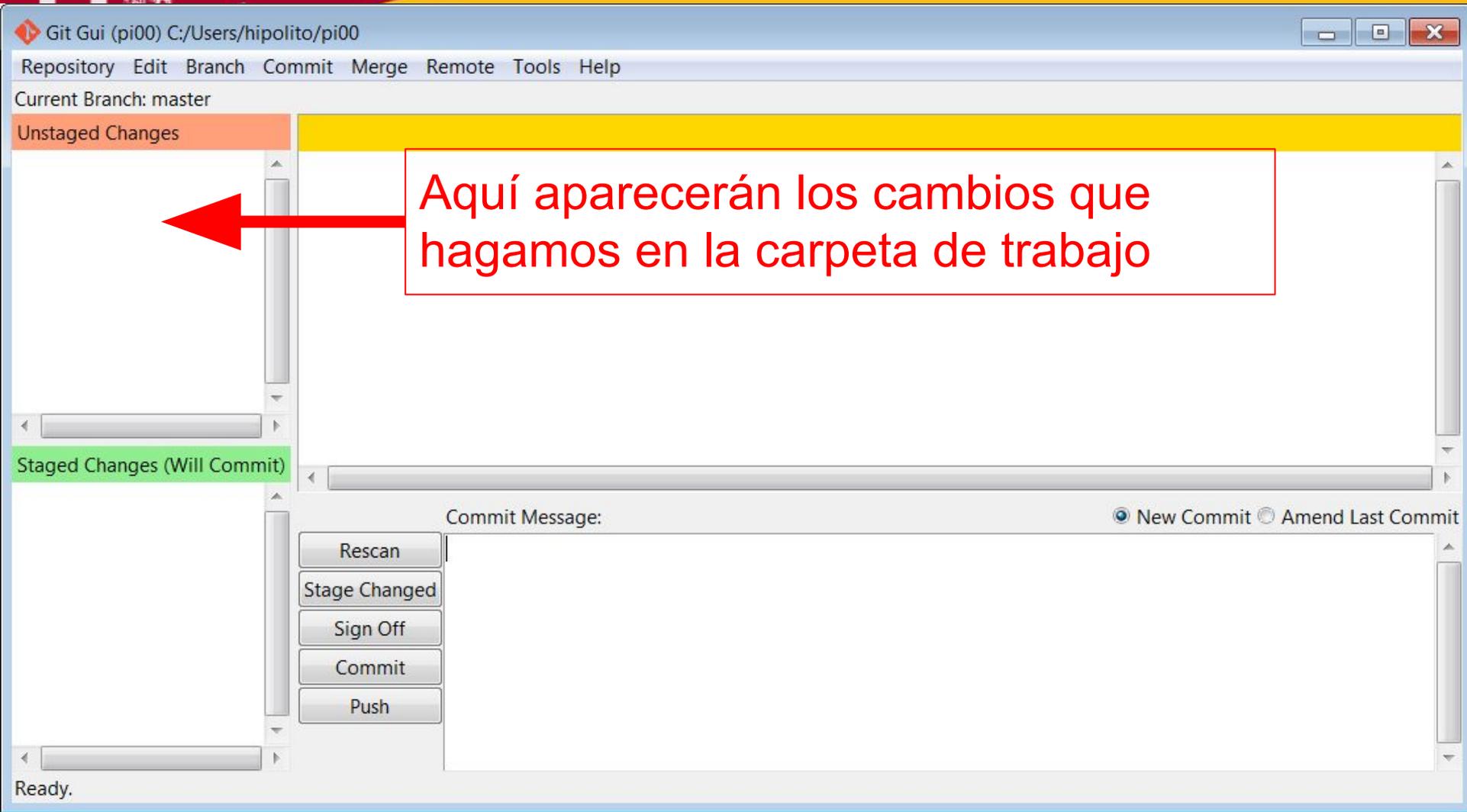
# Clonar el repo



Siempre que hagamos una operación con el repositorio remoto nos pedirá el password (al clonar puede que nos lo pida más de una vez)

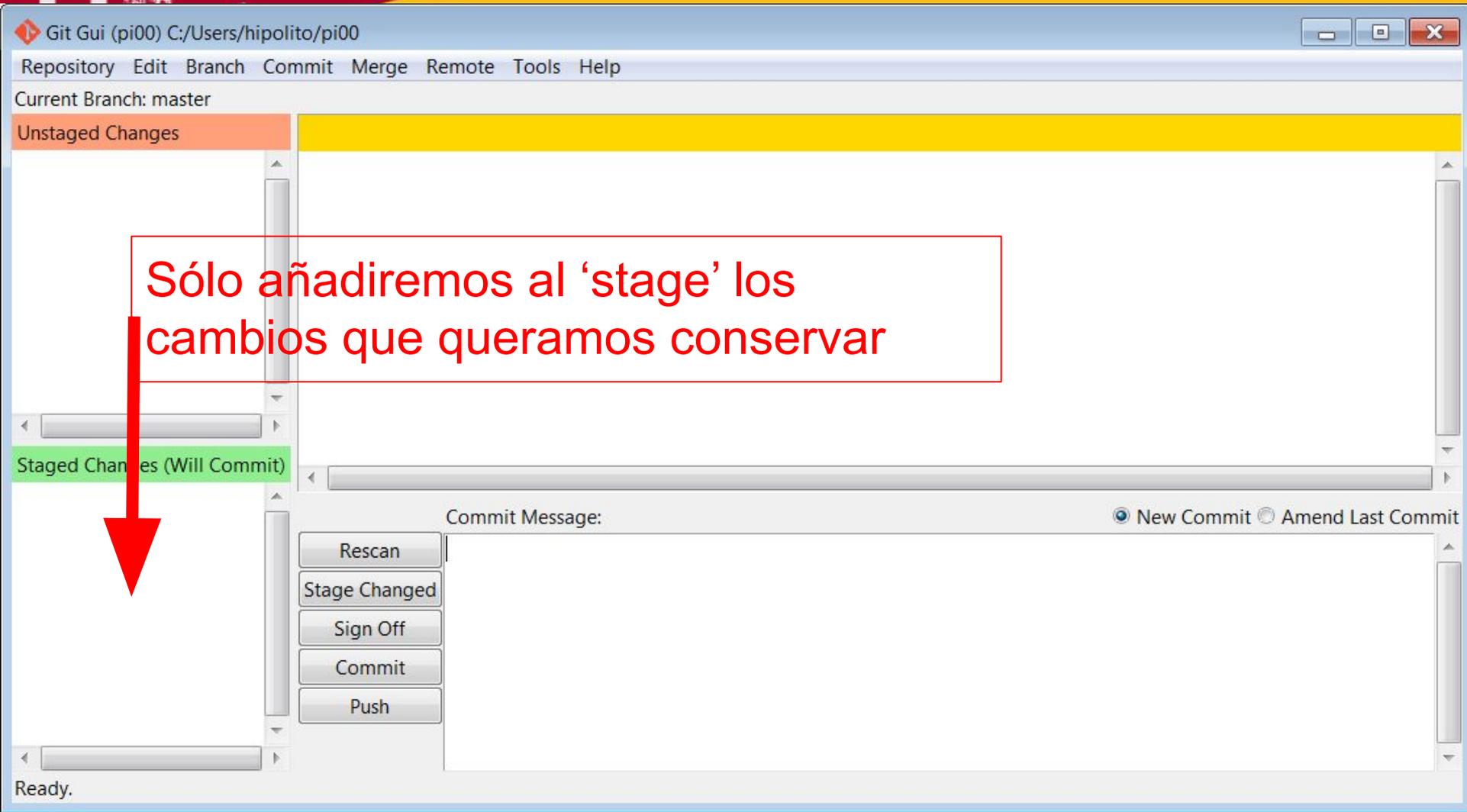
# Un comentario sobre la rama principal

- Hasta hace unos años, a la rama principal de un repositorio git se le llamaba **master**
- Por temas de inclusión en el mundo anglosajón, hoy día a la rama principal de los repositorios nuevos se prefiere llamarla **main**
- Las capturas de las siguientes transparencias están hechas sobre un repositorio antiguo cuya rama principal se llama **master**. Vosotros trabajaréis en la rama **main** y salvo ese cambio de nombre, el resto es exactamente igual



The screenshot shows the Git GUI application window. The title bar reads "Git Gui (pi00) C:/Users/hipolito/pi00". The menu bar includes "Repository", "Edit", "Branch", "Commit", "Merge", "Remote", "Tools", and "Help". Below the menu bar, it says "Current Branch: master". The main area is divided into two sections: "Unstaged Changes" (highlighted in orange) and "Staged Changes (Will Commit)" (highlighted in green). A red arrow points from a text box to the "Unstaged Changes" section. The "Commit Message:" field is visible, along with buttons for "Rescan", "Stage Changed", "Sign Off", "Commit", and "Push". The status bar at the bottom left says "Ready.".

Aquí aparecerán los cambios que hagamos en la carpeta de trabajo

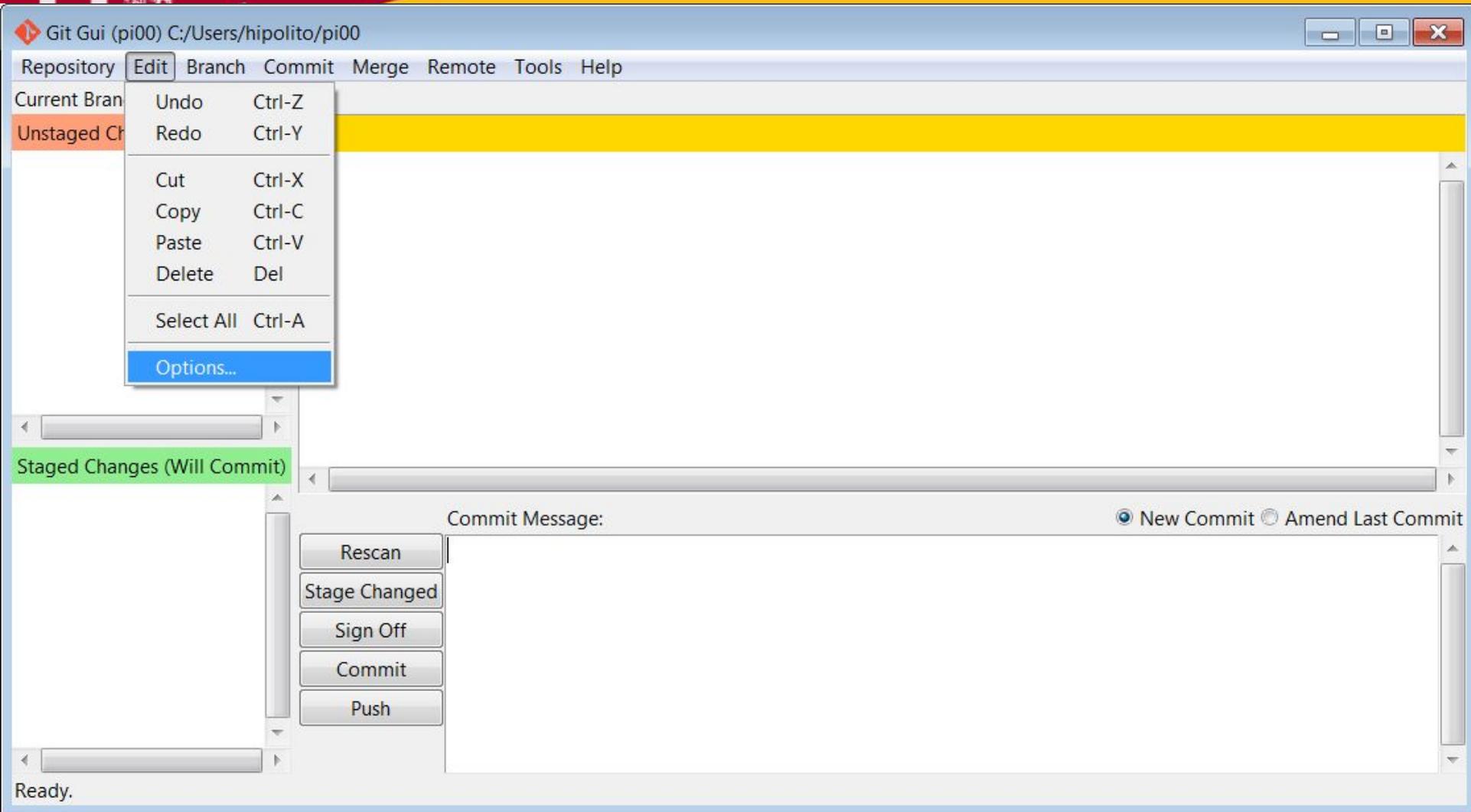


The screenshot shows the Git GUI application window. The title bar reads "Git Gui (pi00) C:/Users/hipolito/pi00". The menu bar includes "Repository", "Edit", "Branch", "Commit", "Merge", "Remote", "Tools", and "Help". The status bar at the top indicates "Current Branch: master". The main interface is divided into two main sections: "Unstaged Changes" (highlighted in yellow) and "Staged Changes (Will Commit)" (highlighted in green). A red box with the text "Sólo añadiremos al 'stage' los cambios que queremos conservar" is positioned over the "Unstaged Changes" area, with a red arrow pointing down towards the "Staged Changes" area. Below the "Staged Changes" section, there is a "Commit Message:" field with radio buttons for "New Commit" (selected) and "Amend Last Commit". A vertical toolbar on the left contains buttons for "Rescan", "Stage Changed", "Sign Off", "Commit", and "Push". The status bar at the bottom left says "Ready."

# Contenido

- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- Push
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout

# Configuramos git



pi00 Repository

User Name: Hipolito Guzman

Email Address: hipolito@gie.esi.us.es

Summarize Merge Commits

Merge Verbosity: 2

Show Diffstat After Merge

Use Merge Tool:

Trust File Modification Timestamps

Prune Tracking Branches During Fetch

Match Tracking Branches

Use Textconv For Diffs and Blames

Blame Copy Only On Changed Files

Maximum Length of Recent Repositories List: 10

Minimum Letters To Blame Copy On: 40

Blame History Context Radius (days): 7

Number of Diff Context Lines: 5

Additional Diff Parameters:

Commit Message Text Width: 75

New Branch Name Template:

Default File Contents Encoding: cp1252

Warn before committing to a detached head

Staging of untracked files: ask

Show untracked files

Spelling Dictionary:

Global (All Repositories)

User Name: Hipolito Guzman

Email Address: hipolito@gie.esi.us.es

Summarize Merge Commits

Merge Verbosity: 2

Show Diffstat After Merge

Use Merge Tool:

Trust File Modification Timestamps

Prune Tracking Branches During Fetch

Match Tracking Branches

Use Textconv For Diffs and Blames

Blame Copy Only On Changed Files

Maximum Length of Recent Repositories List: 10

Minimum Letters To Blame Copy On: 40

Blame History Context Radius (days): 7

Number of Diff Context Lines: 5

Additional Diff Parameters:

Commit Message Text Width: 75

New Branch Name Template:

Default File Contents Encoding: cp1252

Warn before committing to a detached head

Staging of untracked files: ask

Show untracked files

Spelling Dictionary:

Main Font: Segoe UI 9 pt.

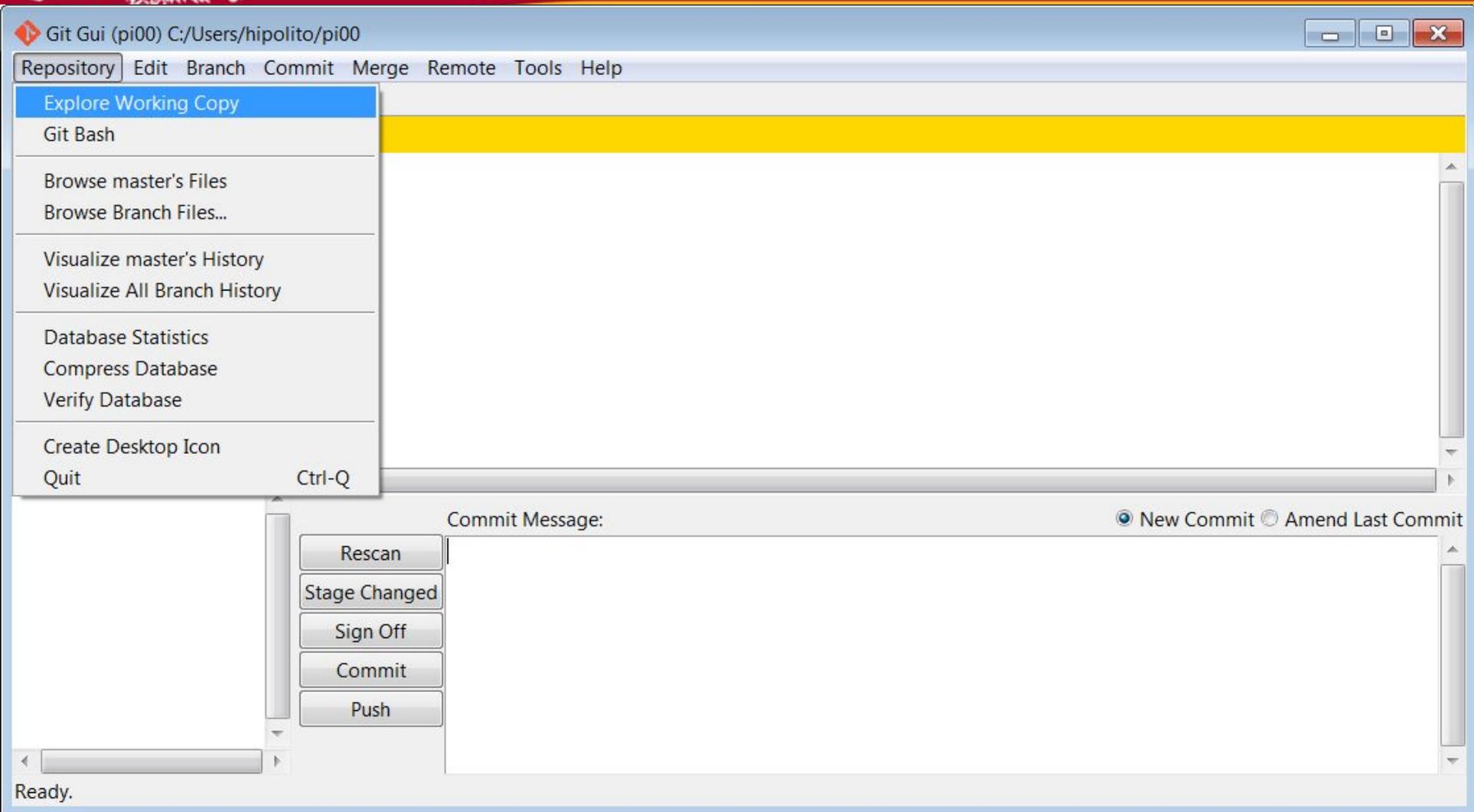
Diff/Console Font: Courier New 10 pt.

Poned vuestro nombre y vuestro email

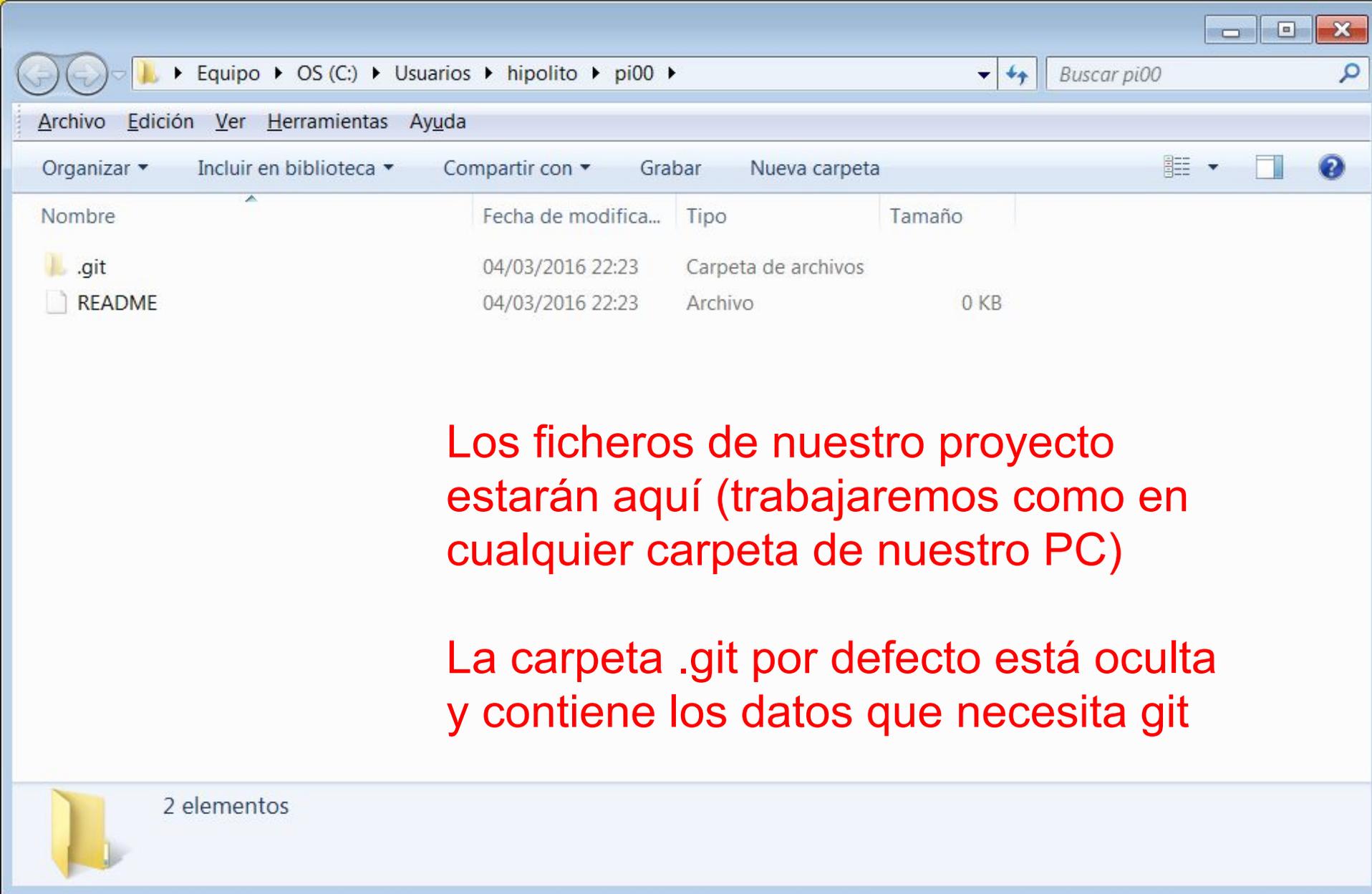
# Contenido

- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- Push
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout

# Añadir ficheros al stage



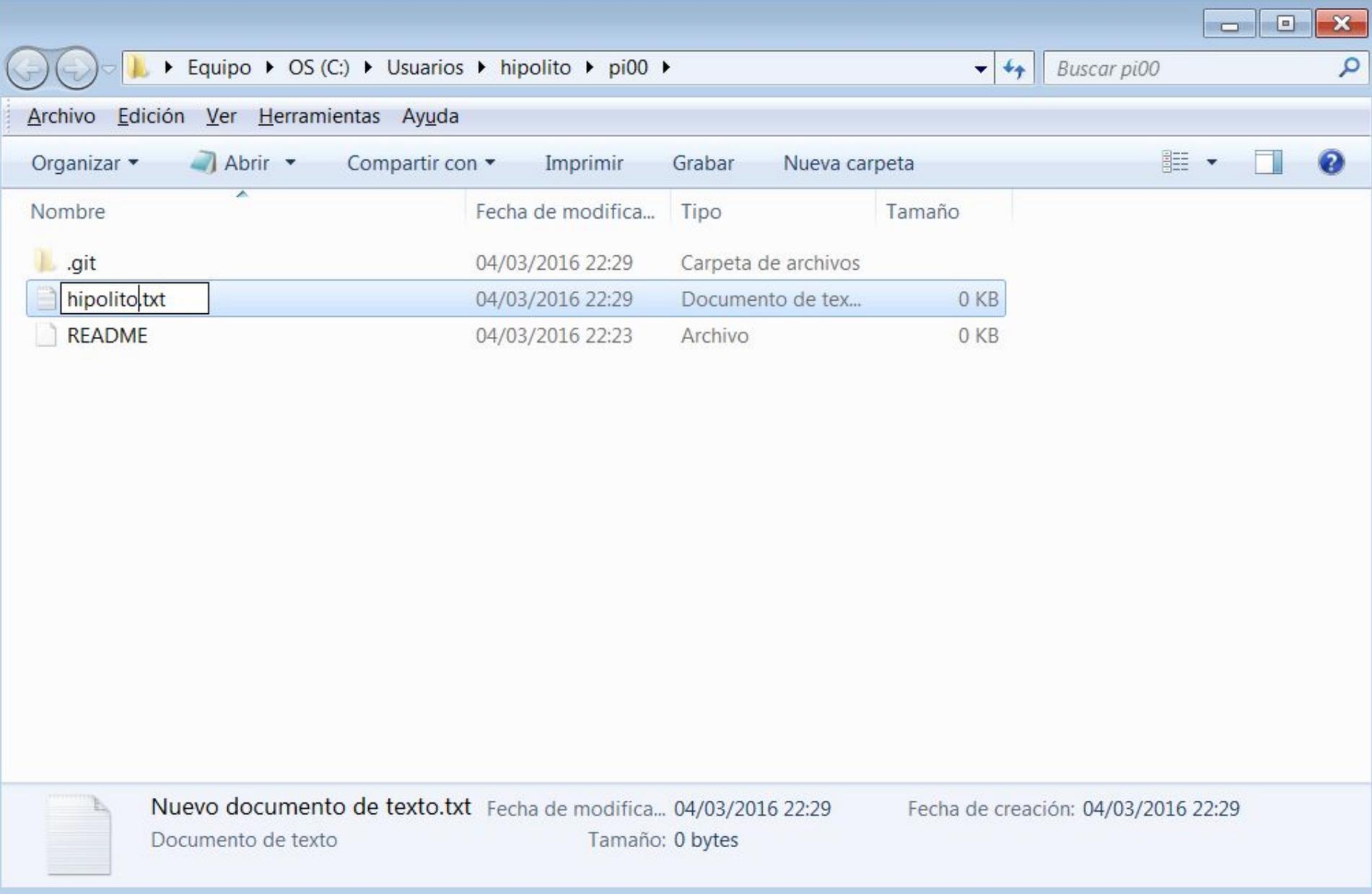
Abrimos la carpeta del repo



Los ficheros de nuestro proyecto estarán aquí (trabajaremos como en cualquier carpeta de nuestro PC)

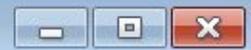
La carpeta .git por defecto está oculta y contiene los datos que necesita git

Workspace es una carpeta más de nuestro PC



Creamos un fichero con nuestro nombre

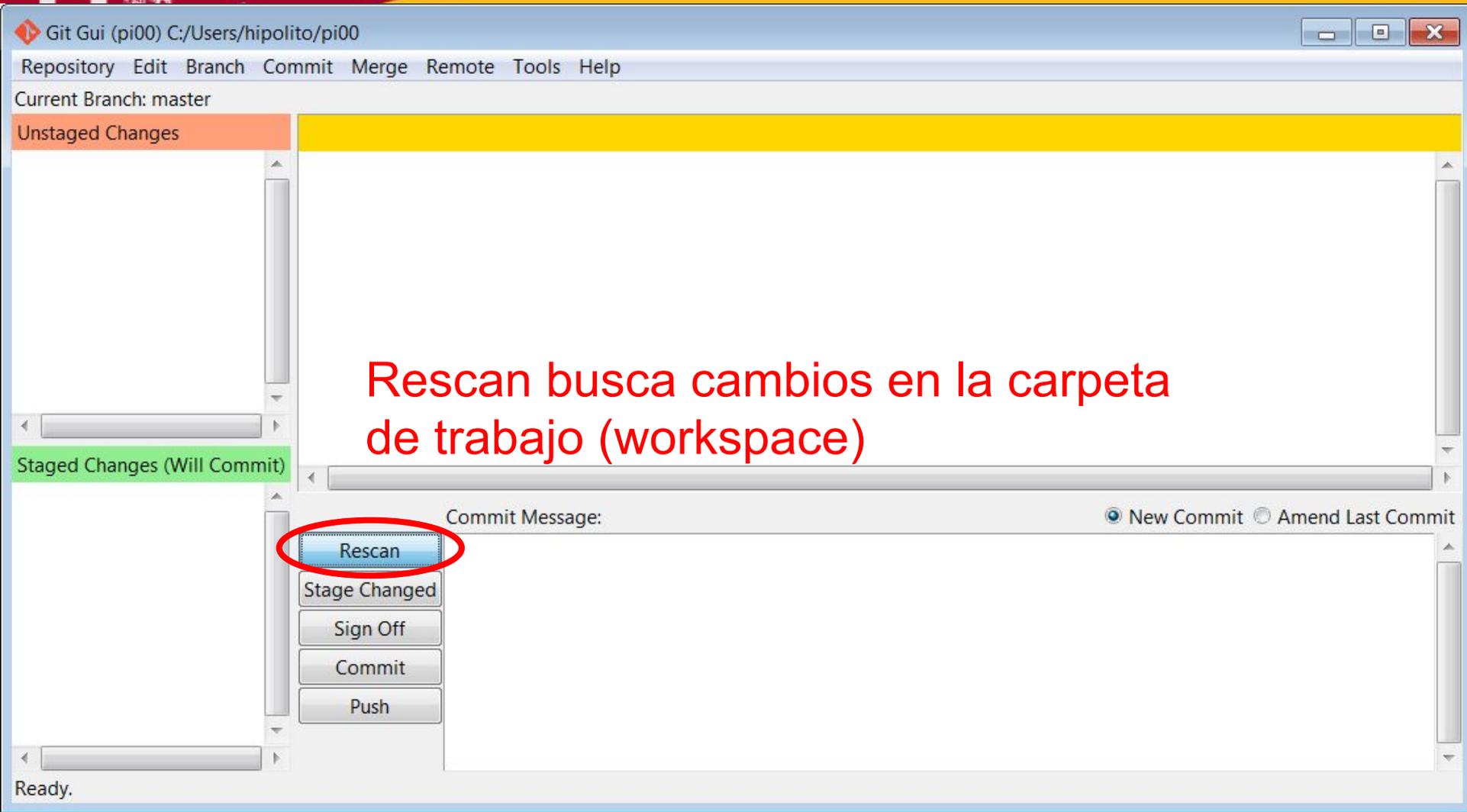
hipolito.txt: Bloc de notas



Archivo Edición Formato Ver Ayuda

Contenido de 'hipolito.txt'

# Añadir ficheros al stage



Rescan busca cambios en la carpeta de trabajo (workspace)

# Añadir ficheros al stage

Git Gui (pi00) C:/Users/hipolito/pi00

Repository Edit Branch Commit Merge Remote Tools Help

Current Branch: master

Unstaged Changes

hipolito.txt

Staged Changes (Will Commit)

Commit Message:  New Commit  Amend Last Commit

Rescan  
Stage Changed  
Sign Off  
Commit  
Push

Ready.

Aparece el fichero que hemos creado. Está 'unstaged', lo que significa que NO se añadirá si hacemos 'commit'

Click en el nombre del fichero (no en el icono a la izquierda)

# Añadir ficheros al stage

Git Gui (pi00) C:/Users/hipolito/pi00

Repository Edit Branch Commit Merge Remote Tools Help

Current Branch: master

Unstaged Changes **Untracked not staged** File: hipolito.txt

hipolito.txt Contenido de 'hipolito.txt'

Staged Changes (Will Commit)

Commit Message:  New Commit  Amend Last Commit

Rescan  
Stage Changed  
Sign Off  
Commit  
Push

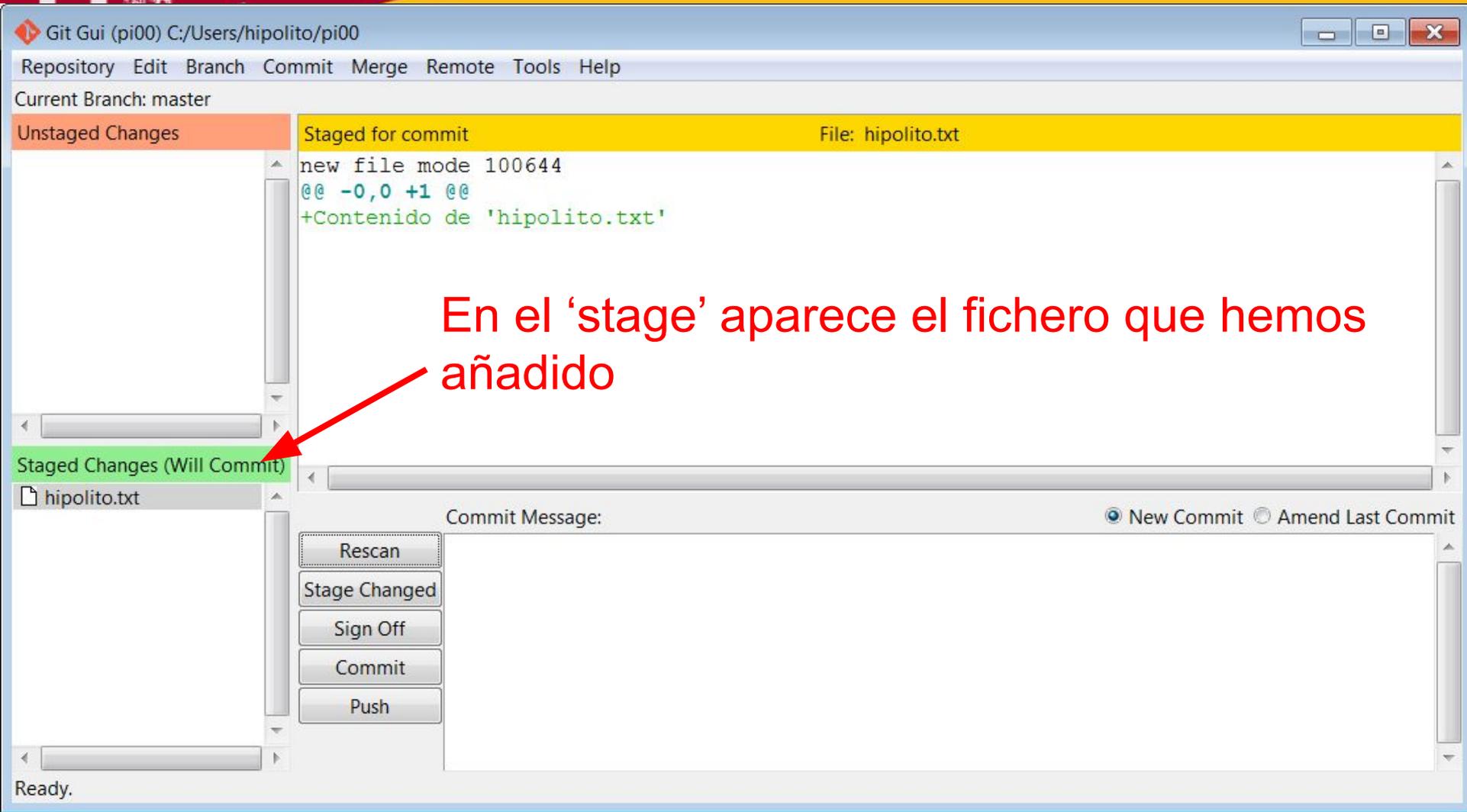
Ready.

Además de 'not staged', el fichero está untracked

Untracked significa que por ahora no está bajo control de versiones, ya que es un fichero nuevo

Click en el icono a la izquierda del nombre para añadirlo al stage ('git add')

# Añadir ficheros al stage



The screenshot shows the Git GUI interface for a repository at C:/Users/hipolito/pi00. The current branch is master. The interface is divided into several sections:

- Unstaged Changes:** This section is currently empty.
- Staged for commit:** This section is highlighted in yellow and shows the details of the file 'hipolito.txt' that has been staged. The content is:

```
new file mode 100644
@@ -0,0 +1 @@
+Contenido de 'hipolito.txt'
```
- Staged Changes (Will Commit):** This section is highlighted in green and shows the file 'hipolito.txt' as a single item to be committed.
- Commit Message:** A text area for entering a commit message, with radio buttons for 'New Commit' (selected) and 'Amend Last Commit'.
- Buttons:** A vertical stack of buttons: Rescan, Stage Changed, Sign Off, Commit, and Push.

A red arrow points from the text 'En el 'stage' aparece el fichero que hemos añadido' to the 'hipolito.txt' entry in the 'Staged Changes (Will Commit)' section.

En el 'stage' aparece el fichero que hemos añadido

# Añadir ficheros al stage

Git Gui (pi00) C:/Users/hipolito/pi00

Repository Edit Branch Commit Merge Remote Tools Help

Current Branch: master

Unstaged Changes

Staged for commit File: hipolito.txt

```
new file mode 100644
@@ -0,0 +1 @@
+Contenido de 'hipolito.txt'
```

Stage Changes (Will Commit)

- hipolito.txt

Commit Message:  New Commit  Amend Last Commit

Rescan  
Stage Changed  
Sign Off  
Commit  
Push

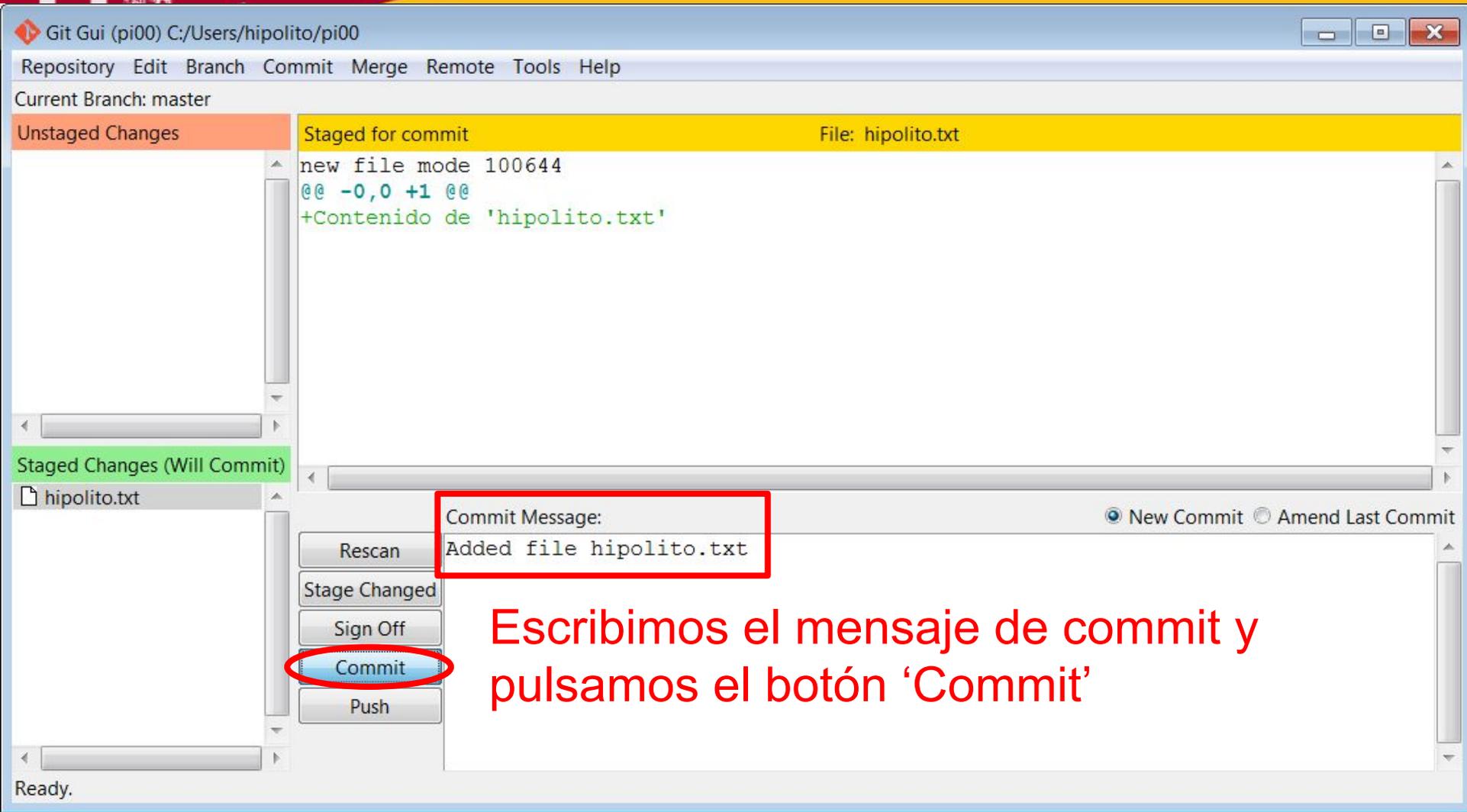
Ready.

Si no quisiéramos hacer commit de este fichero lo sacaríamos del stage ('unstage') haciendo click en el icono a la izquierda del nombre

# Contenido

- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- Push
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout

# Creando commits



Git Gui (pi00) C:/Users/hipolito/pi00

Repository Edit Branch Commit Merge Remote Tools Help

Current Branch: master

Unstaged Changes

Staged for commit File: hipolito.txt

```
new file mode 100644
@@ -0,0 +1 @@
+Contenido de 'hipolito.txt'
```

Staged Changes (Will Commit)

hipolito.txt

Commit Message: Added file hipolito.txt

New Commit  Amend Last Commit

Rescan

Stage Changed

Sign Off

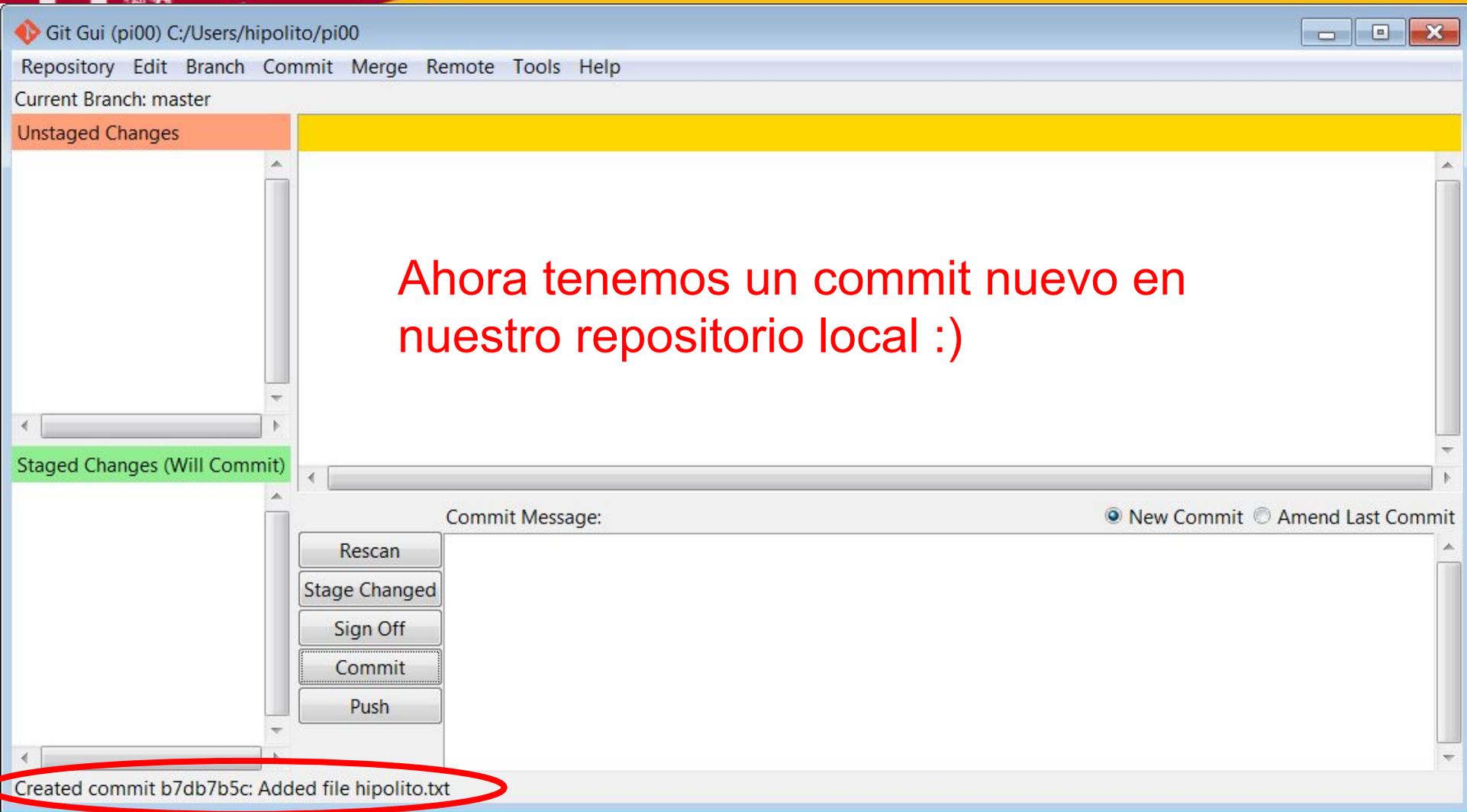
**Commit**

Push

Ready.

Escribimos el mensaje de commit y pulsamos el botón 'Commit'

# Creando commits



The screenshot shows the Git GUI application window. The title bar reads "Git Gui (pi00) C:/Users/hipolito/pi00". The menu bar includes "Repository", "Edit", "Branch", "Commit", "Merge", "Remote", "Tools", and "Help". The status bar indicates "Current Branch: master".

The main workspace is divided into two sections: "Unstaged Changes" (highlighted in yellow) and "Staged Changes (Will Commit)" (highlighted in green). The "Staged Changes" section is currently empty.

Below the "Staged Changes" section is the "Commit Message:" field, which is empty. To the right of this field are two radio buttons: "New Commit" (selected) and "Amend Last Commit".

On the left side of the "Commit Message:" field, there is a vertical stack of buttons: "Rescan", "Stage Changed", "Sign Off", "Commit", and "Push".

At the bottom of the window, a status bar displays the message "Created commit b7db7b5c: Added file hipolito.txt", which is circled in red.

Ahora tenemos un commit nuevo en nuestro repositorio local :)

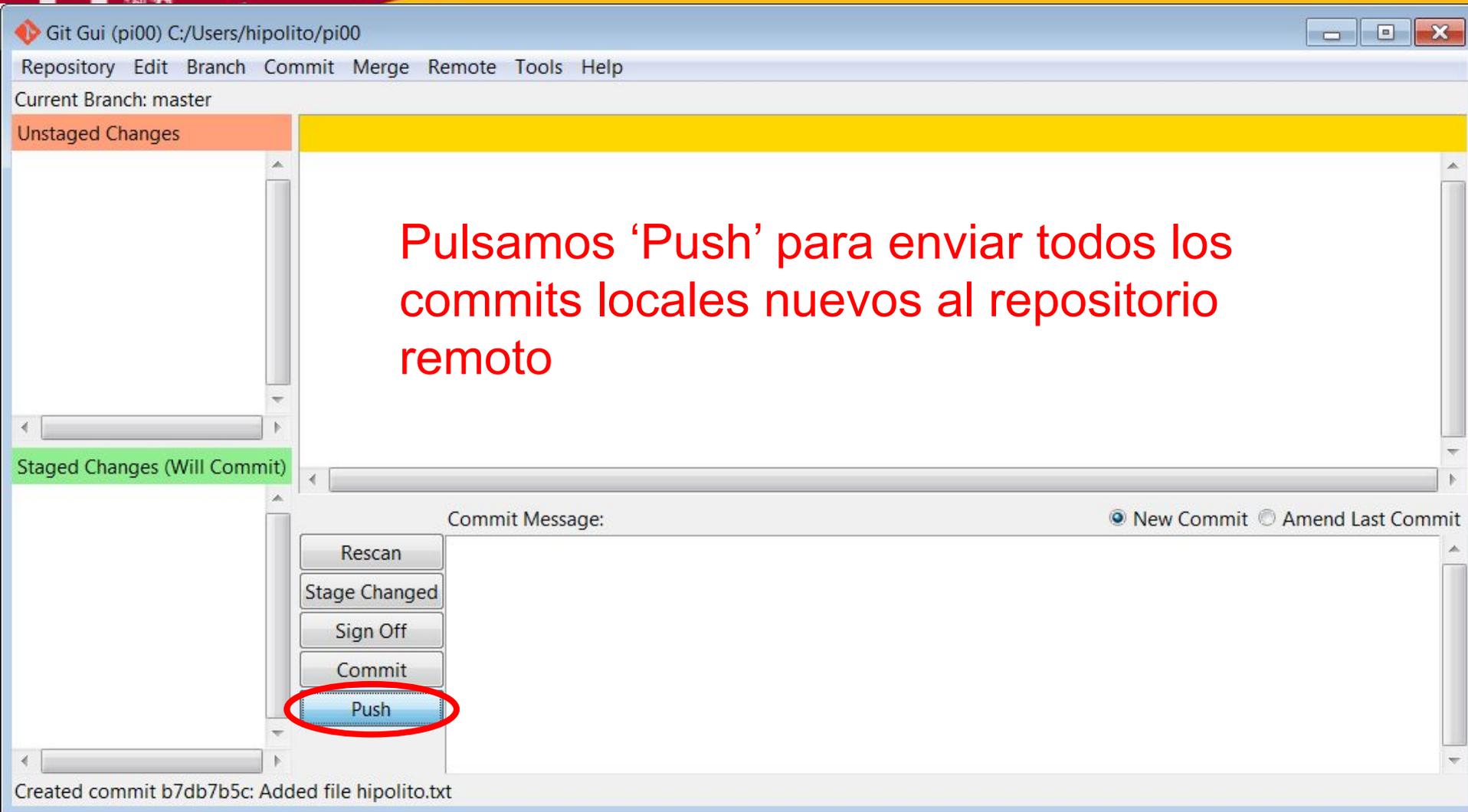
## Formato correcto de un mensaje de commit

Aunque es texto plano y podemos hacerlo como queramos, lo recomendado es:

- Primera línea:
  - Resumen conciso de los cambios
  - Máximo 50 caracteres
- Si añadimos una descripción más extensa:
  - Dejamos la segunda línea en blanco
  - Tercera línea y posteriores:
    - Descripción más detallada
    - máximo 72 caracteres

# Contenido

- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- **Push**
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout



The screenshot shows the Git GUI application window. The title bar reads "Git Gui (pi00) C:/Users/hipolito/pi00". The menu bar includes "Repository", "Edit", "Branch", "Commit", "Merge", "Remote", "Tools", and "Help". The status bar at the top indicates "Current Branch: master".

The interface is divided into several sections:

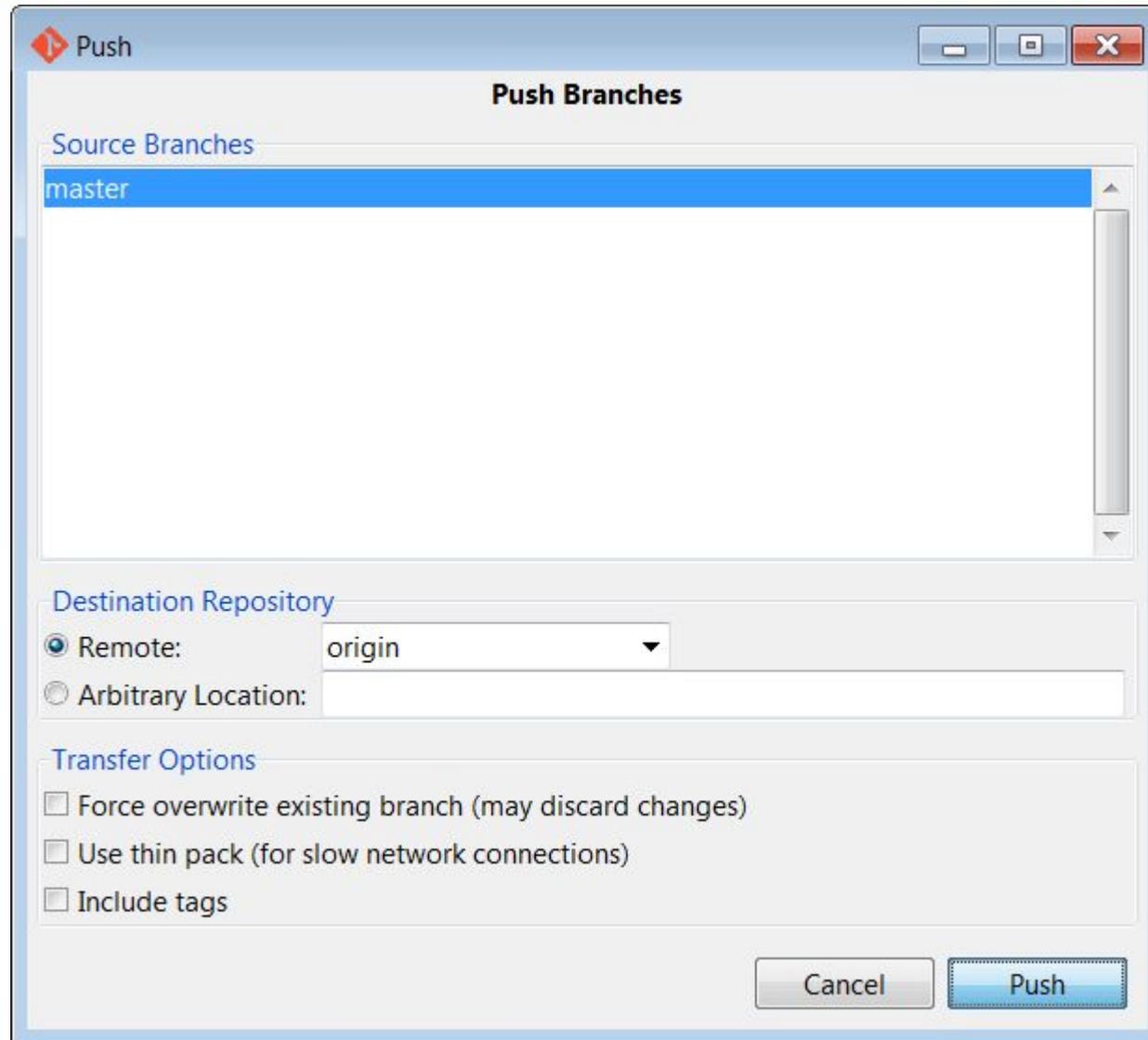
- Unstaged Changes:** A yellow bar at the top of the main area.
- Staged Changes (Will Commit):** A green bar below the unstaged changes.
- Commit Message:** A text area for entering the commit message, with radio buttons for "New Commit" (selected) and "Amend Last Commit".
- Action Buttons:** A vertical stack of buttons on the left side: "Rescan", "Stage Changed", "Sign Off", "Commit", and "Push". The "Push" button is highlighted with a red circle.

At the bottom of the window, a status bar displays: "Created commit b7db7b5c: Added file hipolito.txt".

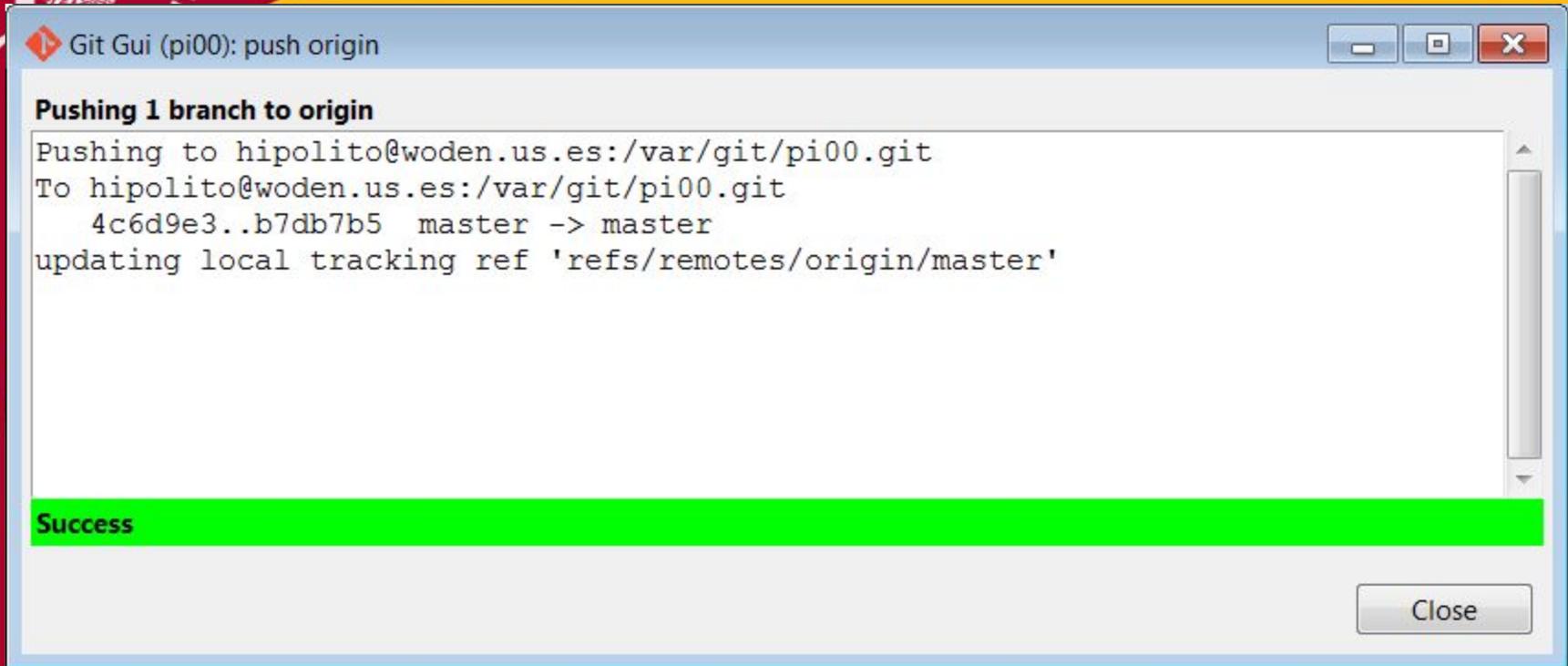
Overlaid on the main area is the text: **Pulsamos 'Push' para enviar todos los commits locales nuevos al repositorio remoto**

# Push

Por defecto  
'pusheamos' los  
cambios que  
hemos hecho en  
nuestra copia  
local de la rama  
principal (**main**  
en repos nuevos,  
**master** en repos  
antiguos)



# Push



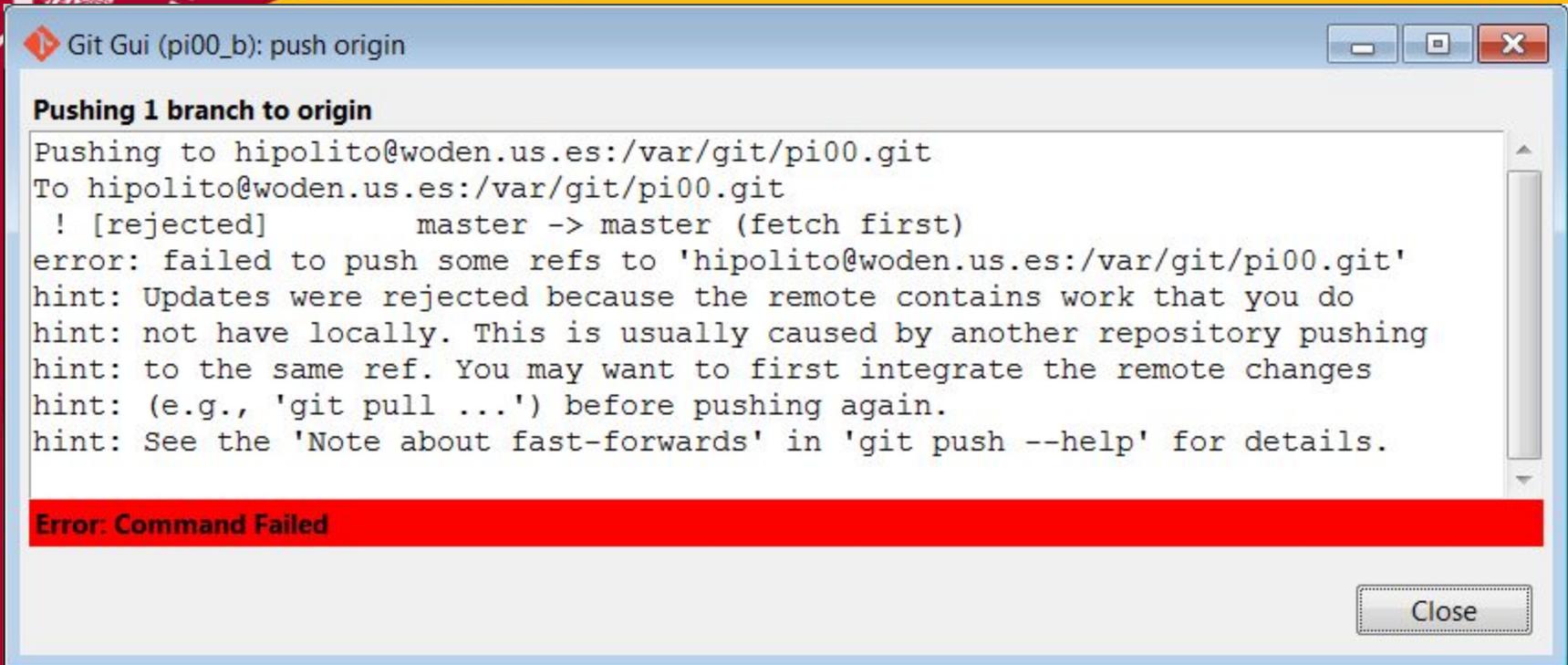
```
Git Gui (pi00): push origin

Pushing 1 branch to origin
Pushing to hipolito@woden.us.es:/var/git/pi00.git
To hipolito@woden.us.es:/var/git/pi00.git
    4c6d9e3..b7db7b5  master -> master
updating local tracking ref 'refs/remotes/origin/master'

Success
```

Close

Normalmente todo va bien...



```
Git Gui (pi00_b): push origin

Pushing 1 branch to origin
Pushing to hipolito@woden.us.es:/var/git/pi00.git
To hipolito@woden.us.es:/var/git/pi00.git
 ! [rejected]          master -> master (fetch first)
error: failed to push some refs to 'hipolito@woden.us.es:/var/git/pi00.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.

Error: Command Failed

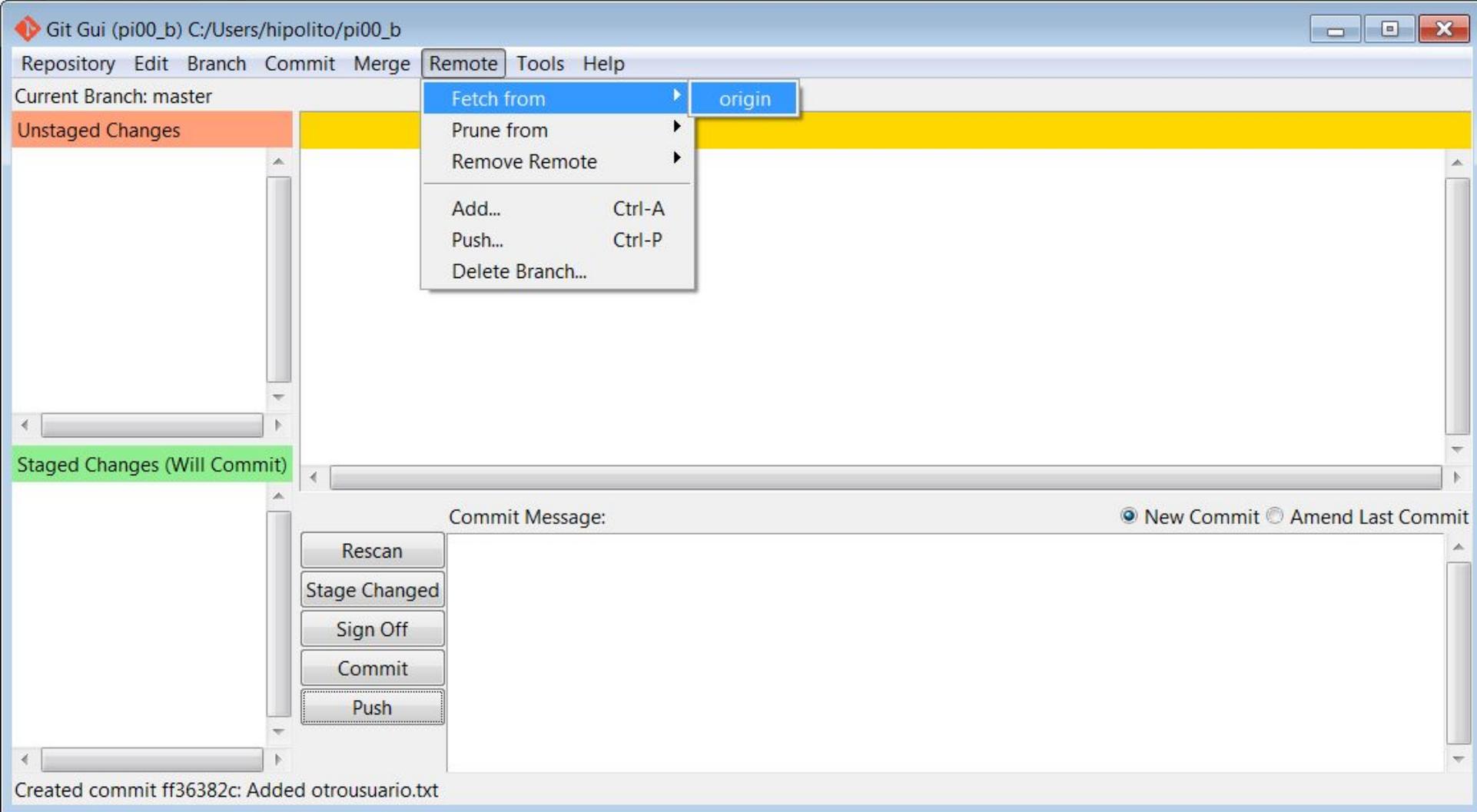
Close
```

Si nos da este error, es porque otro compañero ha hecho push antes que nosotros: debemos actualizar nuestro repo local (fetch + merge) antes de hacer push

# Contenido

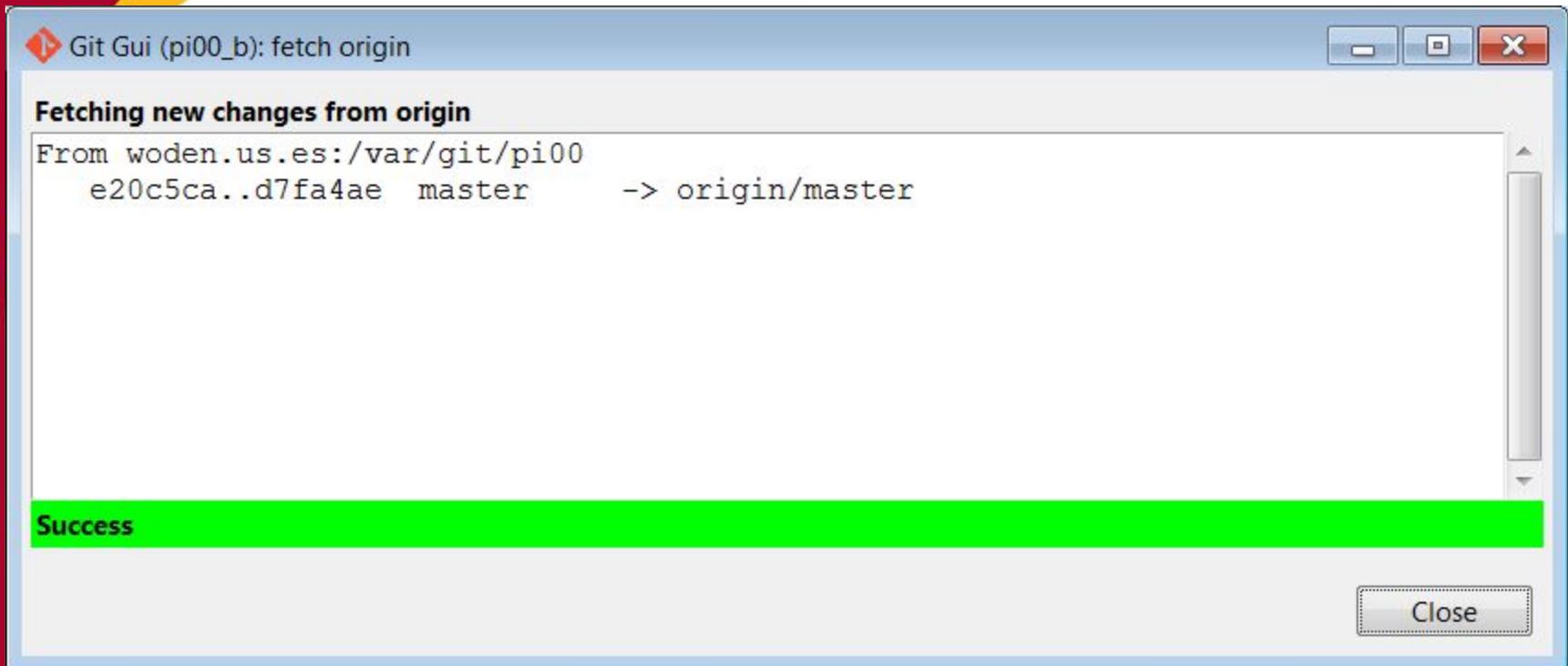
- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- Push
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout

# Pull: fetch + merge



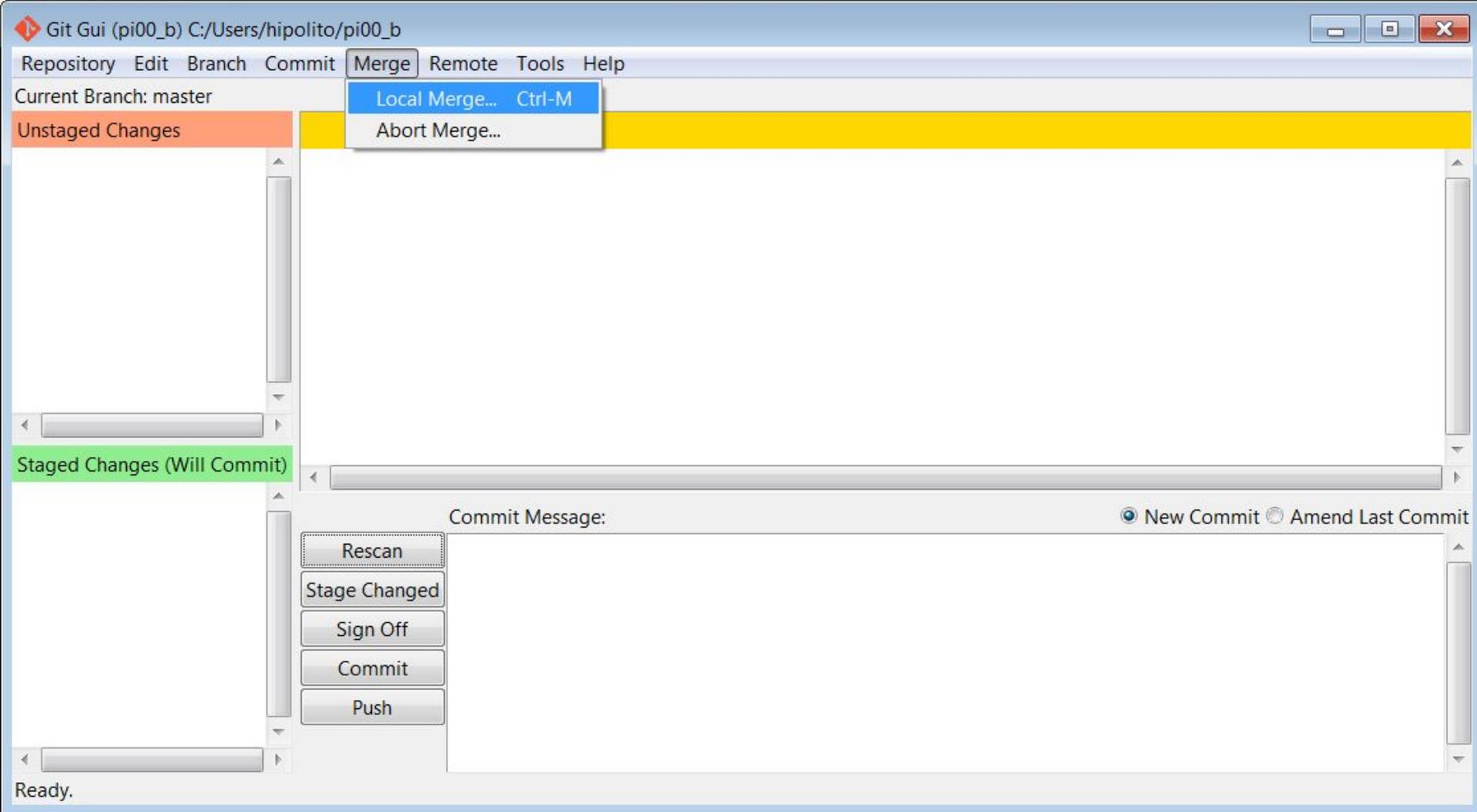
Fetch from origin descarga los cambios nuevos que haya en el repo remoto

# Pull: fetch + merge



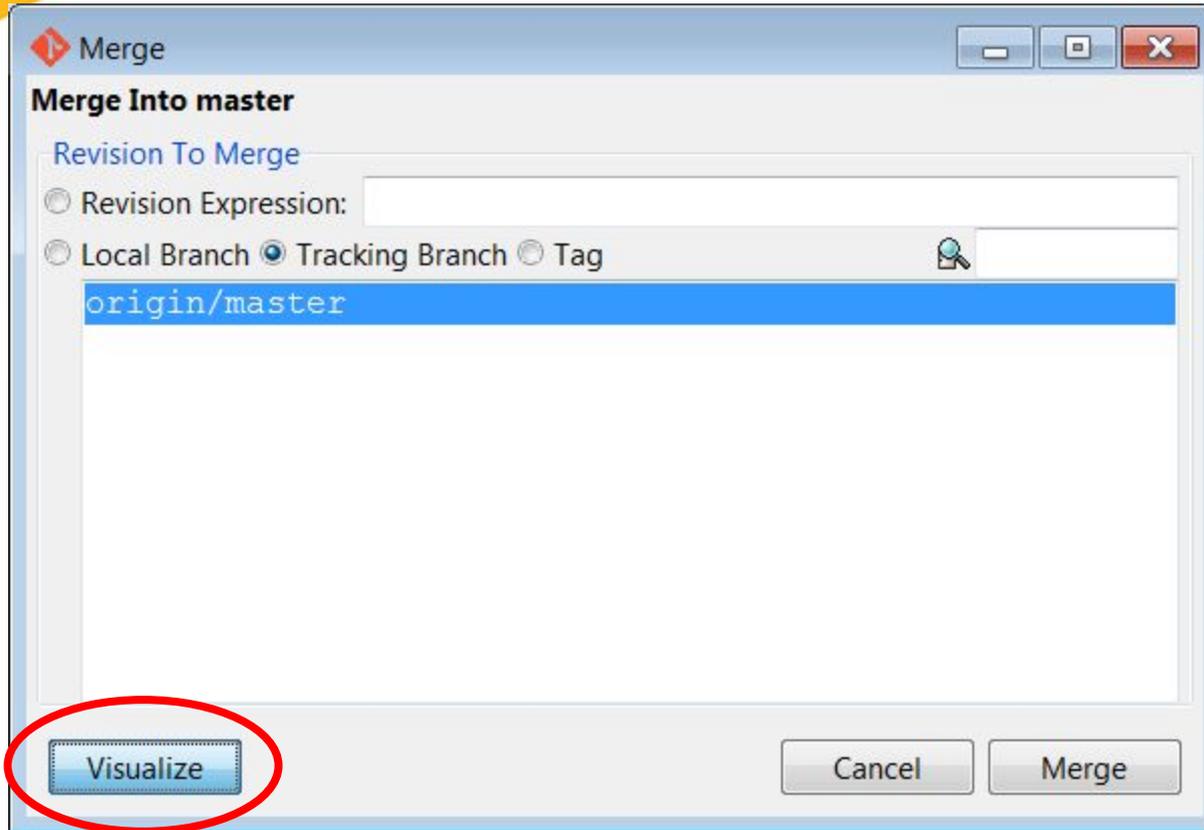
(Nota: si al hacer fetch este cuadro de diálogo no contiene texto, es que no hay cambios nuevos, por lo que no será necesario un merge)

# Pull: fetch + merge



Local merge fusiona los cambios nuevos en el repo local y el workspace

# Pull: fetch + merge



Podemos ver los cambios antes de fusionar

# Pull: fetch + merge

gitk pi00\_b

Archivo Editar Vista Ayuda

remotes/origin/master Added hipolito.txt  
Initial commit

Hipolito Guzman <hipolito@challenger> 2016-03-04 21:56:00  
root <root@woden.(none)> 2013-10-23 23:09:46

SHA1 ID: d7fa4ae7fb8d2284513d4cc2e42a4b59f67503f0 Row 1 / 2

Buscar revisión que contiene: Exacto Todos los campos

Buscar

Diferencia  Versión antigua  Versión nueva Líneas de contexto: 3  Ignora c

Parche  Árbol

Comentarios  
hipolito.txt

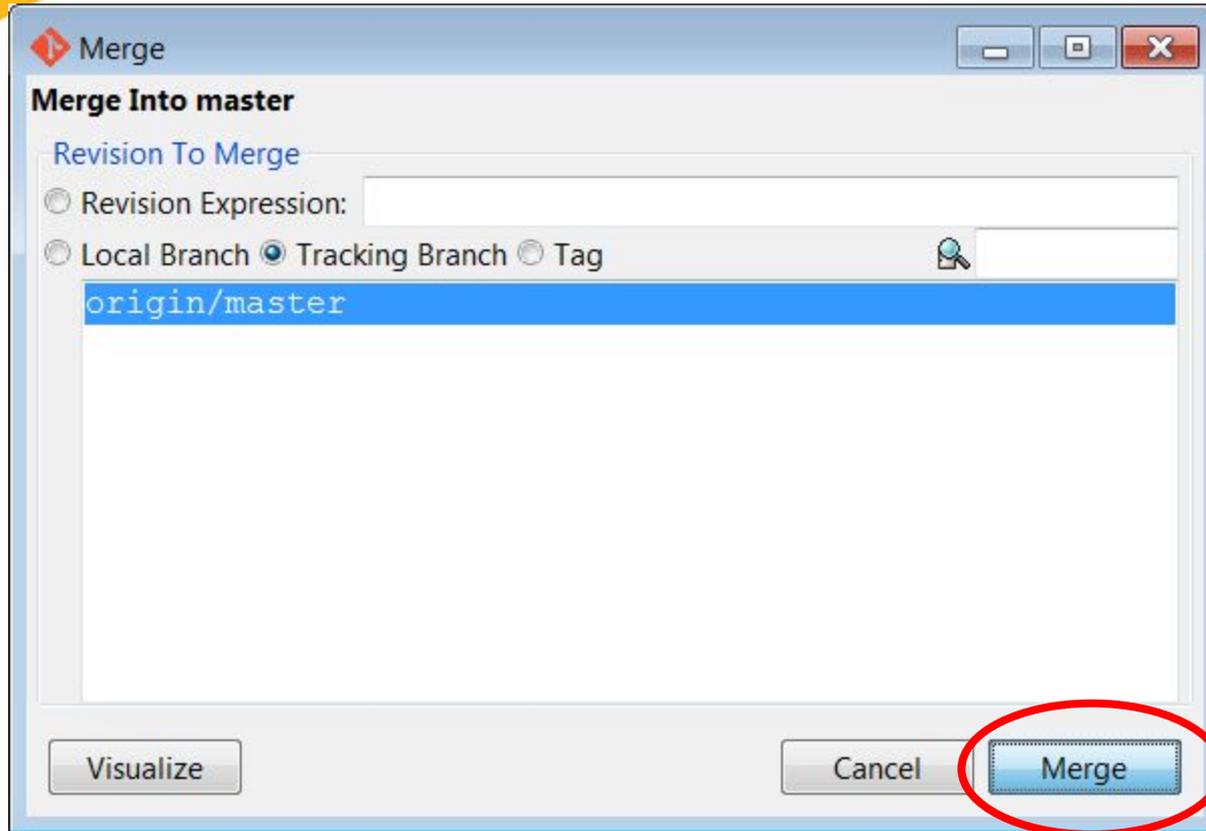
```

Autor: Hipolito Guzman <hipolito@challenger> 2016-03-04 21:56:00
Committer: Hipolito Guzman <hipolito@challenger> 2016-03-04 21:56:00
Padre: e20c5ca8c6a64777f87c9553b7a9d5b4d505ef85 (Initial commit)
Rama: remotes/origin/master
Sigue-a:
Precede-a:

Added hipolito.txt

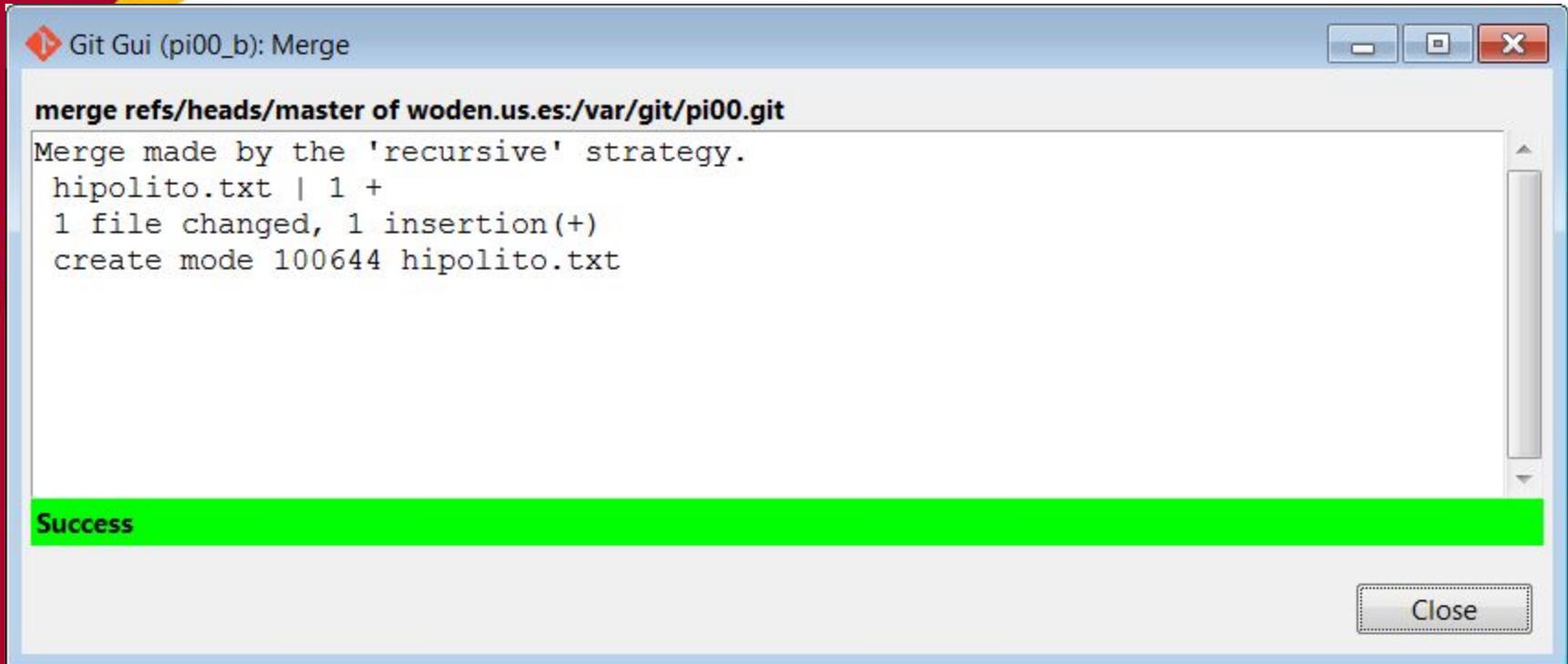
----- hipolito.txt -----
new file mode 100644
index 0000000..91a06b3
@@ -0,0 +1 @@
+Contenido de 'hipolito.txt'
    
```

# Pull: fetch + merge



Pulsamos 'Merge'

# Pull: fetch + merge



Git Gui (pi00\_b): Merge

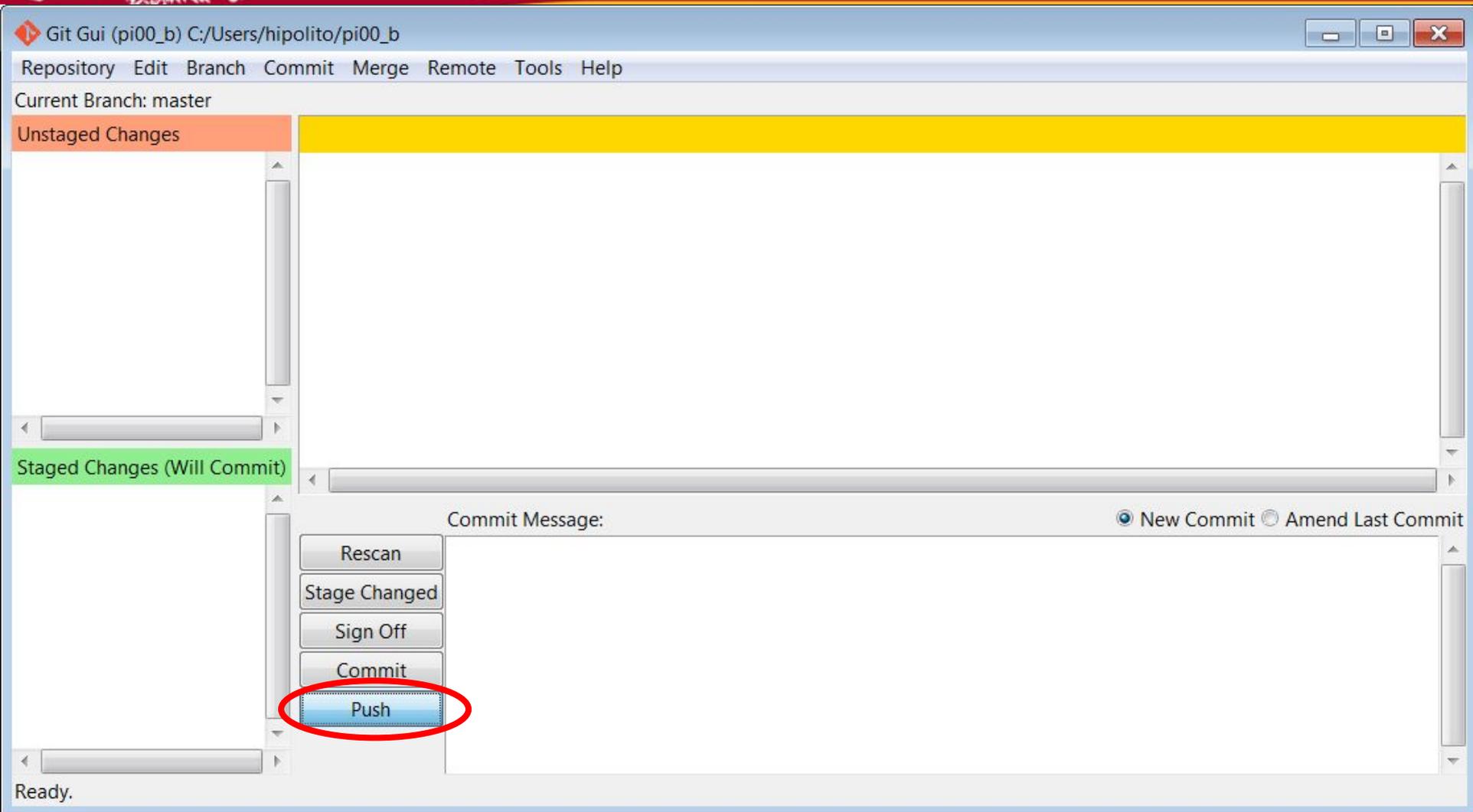
```
merge refs/heads/master of woden.us.es:/var/git/pi00.git  
Merge made by the 'recursive' strategy.  
hipolito.txt | 1 +  
1 file changed, 1 insertion(+)  
create mode 100644 hipolito.txt
```

Success

Close

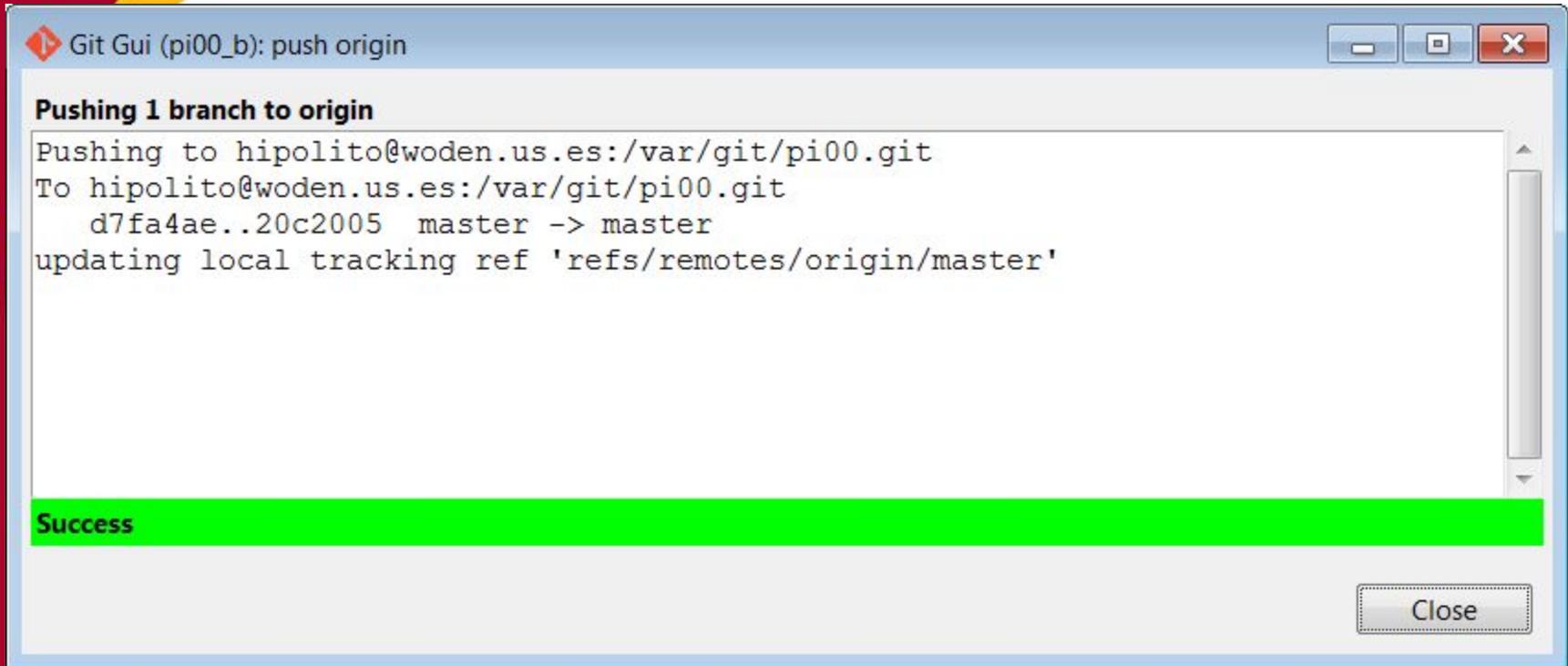
Merge completado con éxito

# Push (2)



Tras el merge, ya podemos 'pushear' nuestros cambios

# Push (2)



Git Gui (pi00\_b): push origin

**Pushing 1 branch to origin**

```
Pushing to hipolito@woden.us.es:/var/git/pi00.git  
To hipolito@woden.us.es:/var/git/pi00.git  
    d7fa4ae..20c2005  master -> master  
updating local tracking ref 'refs/remotes/origin/master'
```

**Success**

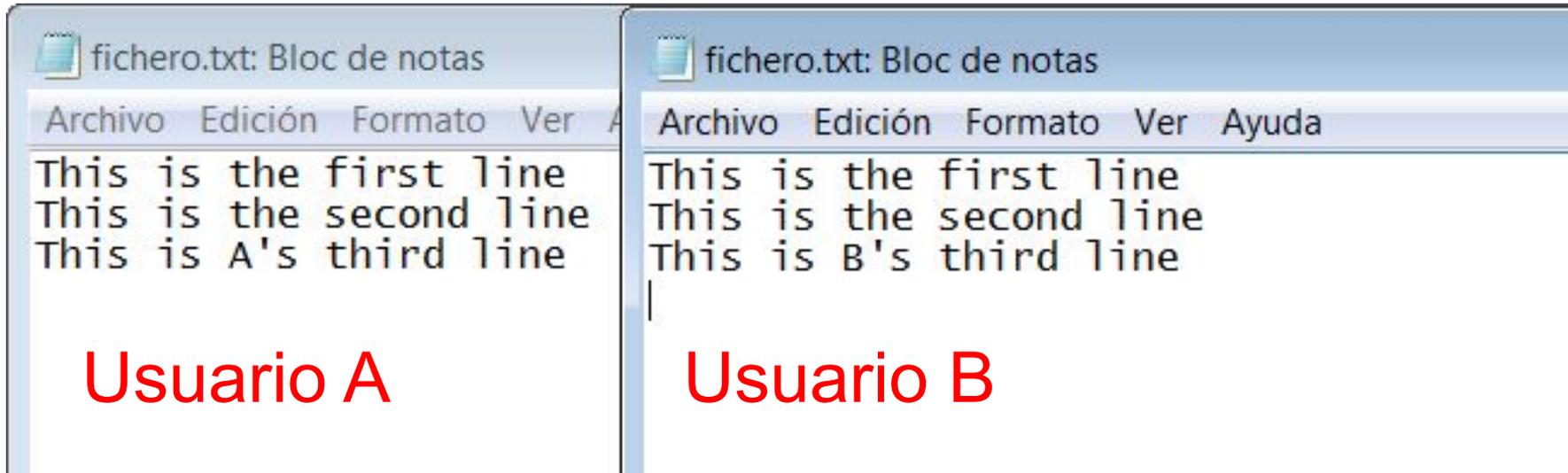
Close

Y esta vez, el push termina correctamente

# Contenido

- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- Push
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout

¿Qué ocurre si dos personas modifican la misma zona del mismo fichero?



The image shows two side-by-side Notepad windows, both titled 'fichero.txt: Bloc de notas'. The left window, labeled 'Usuario A', has a menu bar with 'Archivo', 'Edición', 'Formato', and 'Ver'. Its text content is: 'This is the first line', 'This is the second line', and 'This is A's third line'. The right window, labeled 'Usuario B', has a menu bar with 'Archivo', 'Edición', 'Formato', 'Ver', and 'Ayuda'. Its text content is: 'This is the first line', 'This is the second line', and 'This is B's third line'. The cursor in the right window is positioned at the start of the third line.

**Usuario A**

**Usuario B**

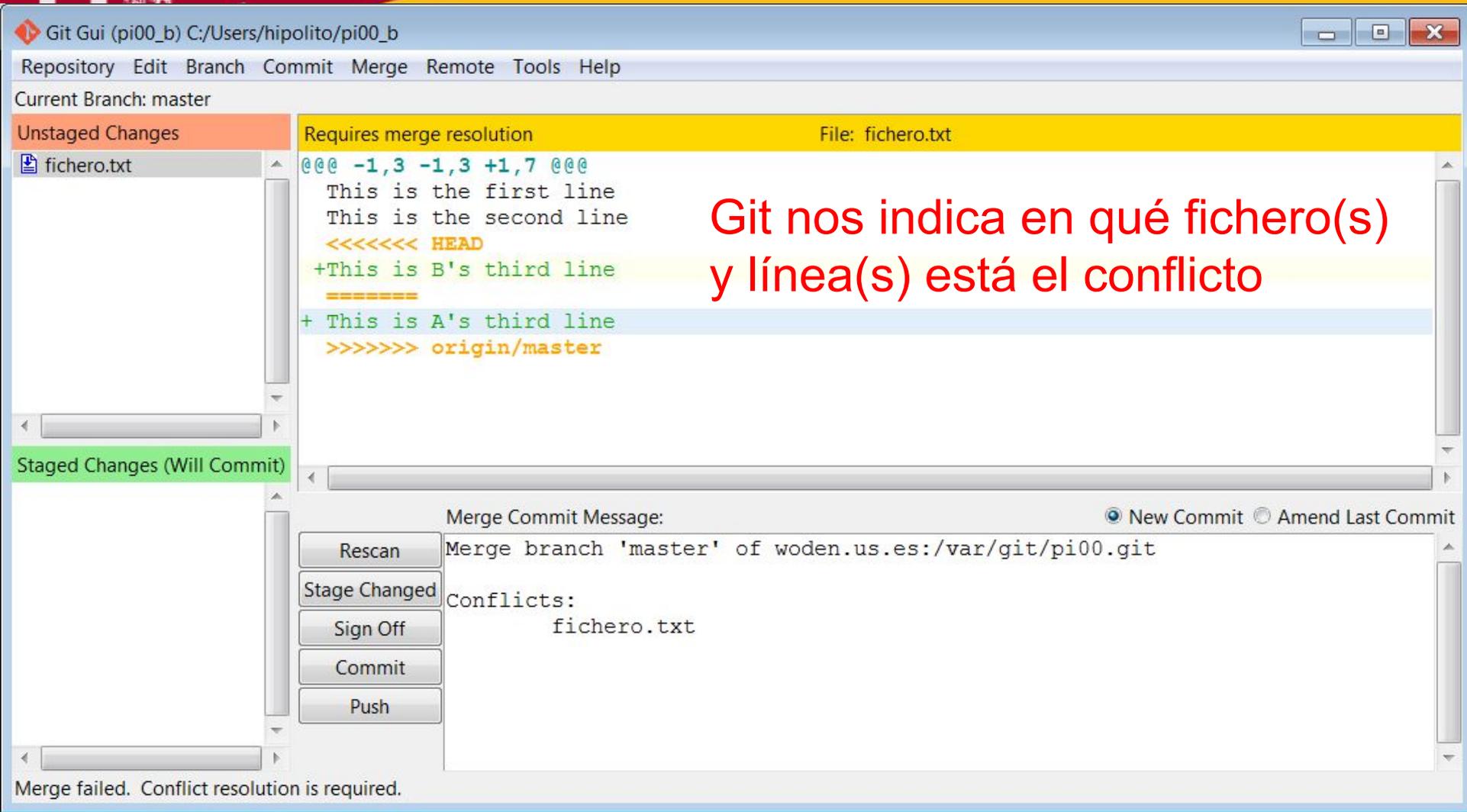
A 'pushea' primero y no ve nada raro  
B tiene que hacer fetch + merge, pero al hacer merge:



```
Git Gui (pi00_b): Merge
merge refs/heads/master of woden.us.es:/var/git/pi00.git
Auto-merging fichero.txt
CONFLICT (content): Merge conflict in fichero.txt
Automatic merge failed; fix conflicts and then commit the result.
```

**Error: Command Failed**

Close



Git Gui (pi00\_b) C:/Users/hipolito/pi00\_b

Repository Edit Branch Commit Merge Remote Tools Help

Current Branch: master

Unstaged Changes

fichero.txt

Requires merge resolution File: fichero.txt

```
@@@ -1,3 -1,3 +1,7 @@@
This is the first line
This is the second line
<<<<<< HEAD
+This is B's third line
=====
+ This is A's third line
>>>>>> origin/master
```

Git nos indica en qué fichero(s) y línea(s) está el conflicto

Staged Changes (Will Commit)

Merge Commit Message:  New Commit  Amend Last Commit

Rescan

Stage Changed

Sign Off

Commit

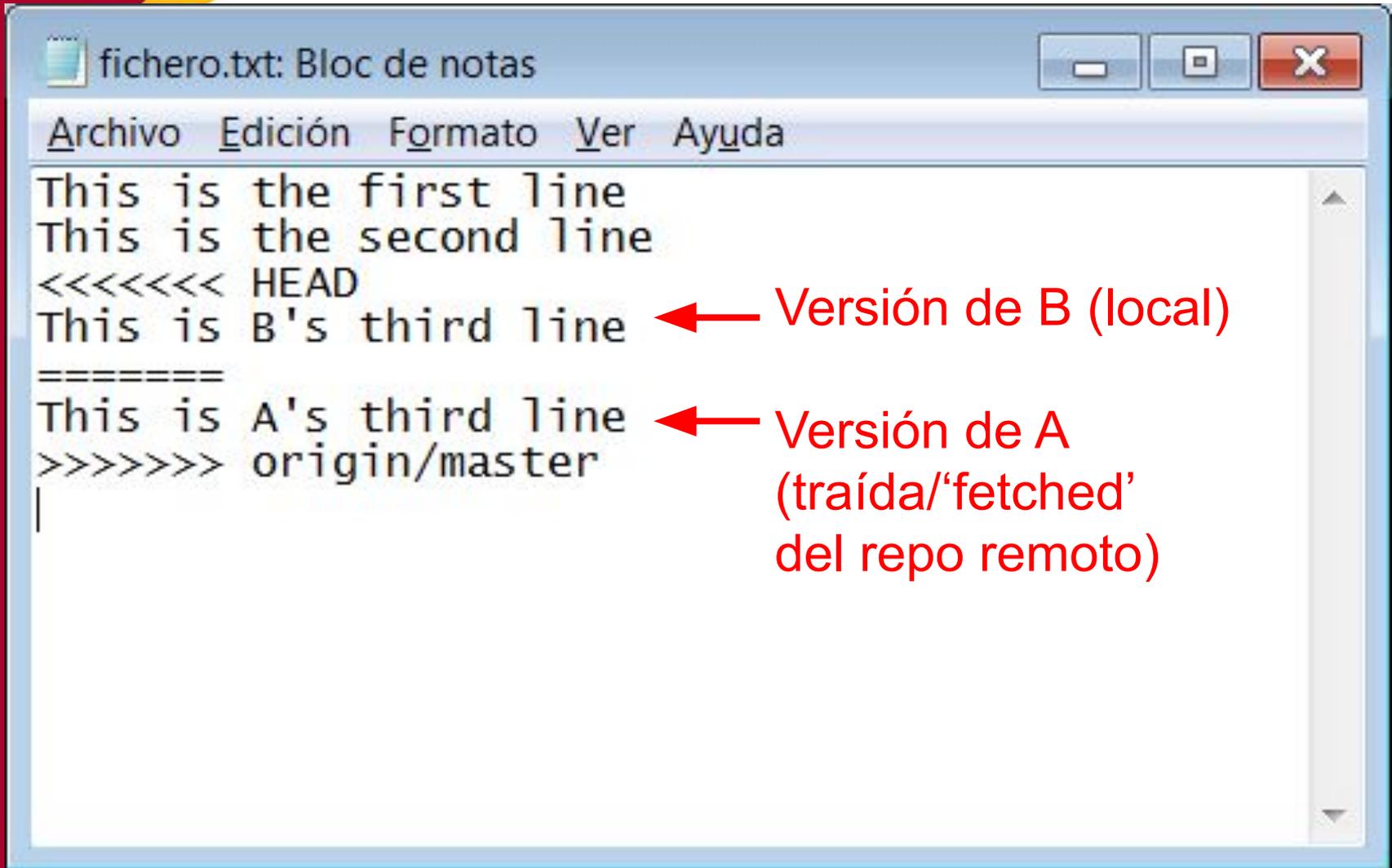
Push

Merge branch 'master' of woden.us.es:/var/git/pi00.git

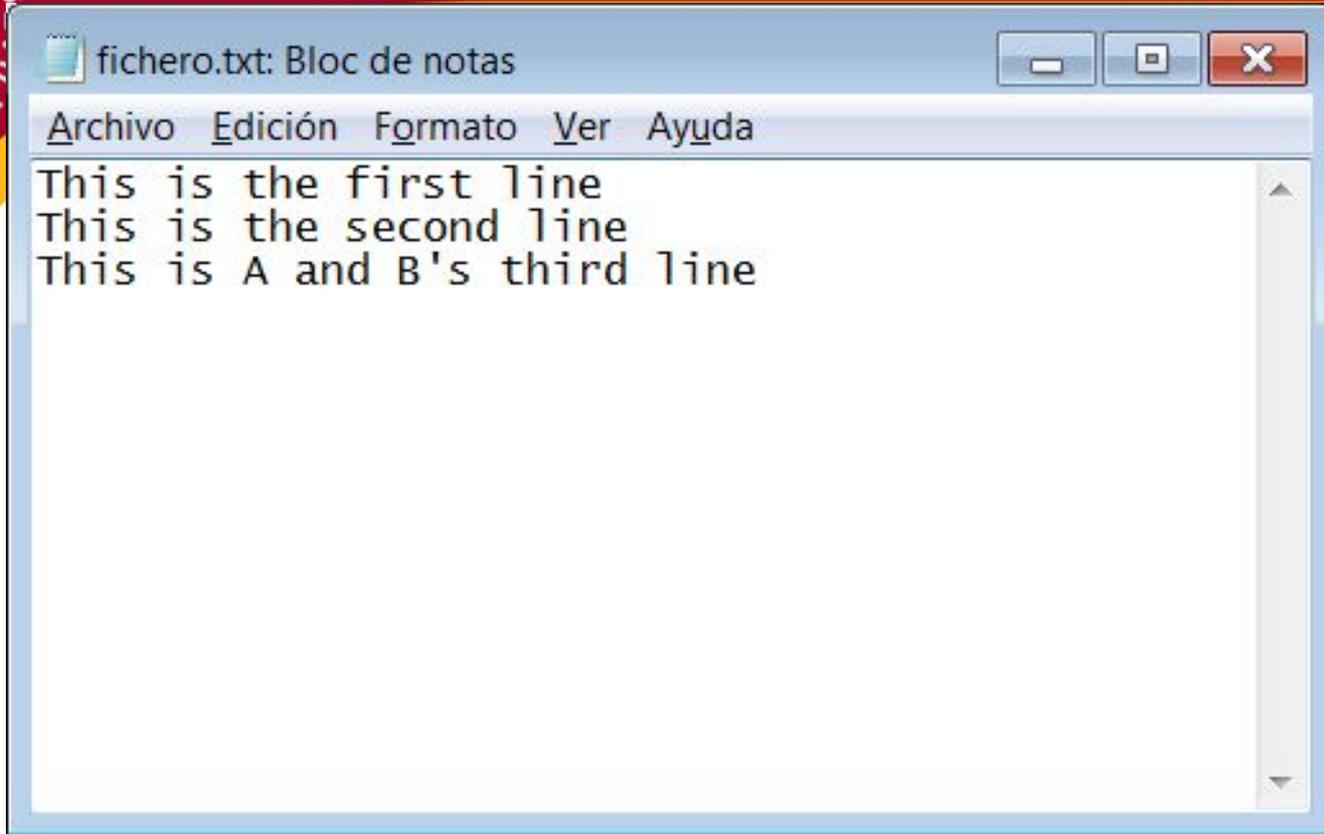
Conflicts:

fichero.txt

Merge failed. Conflict resolution is required.

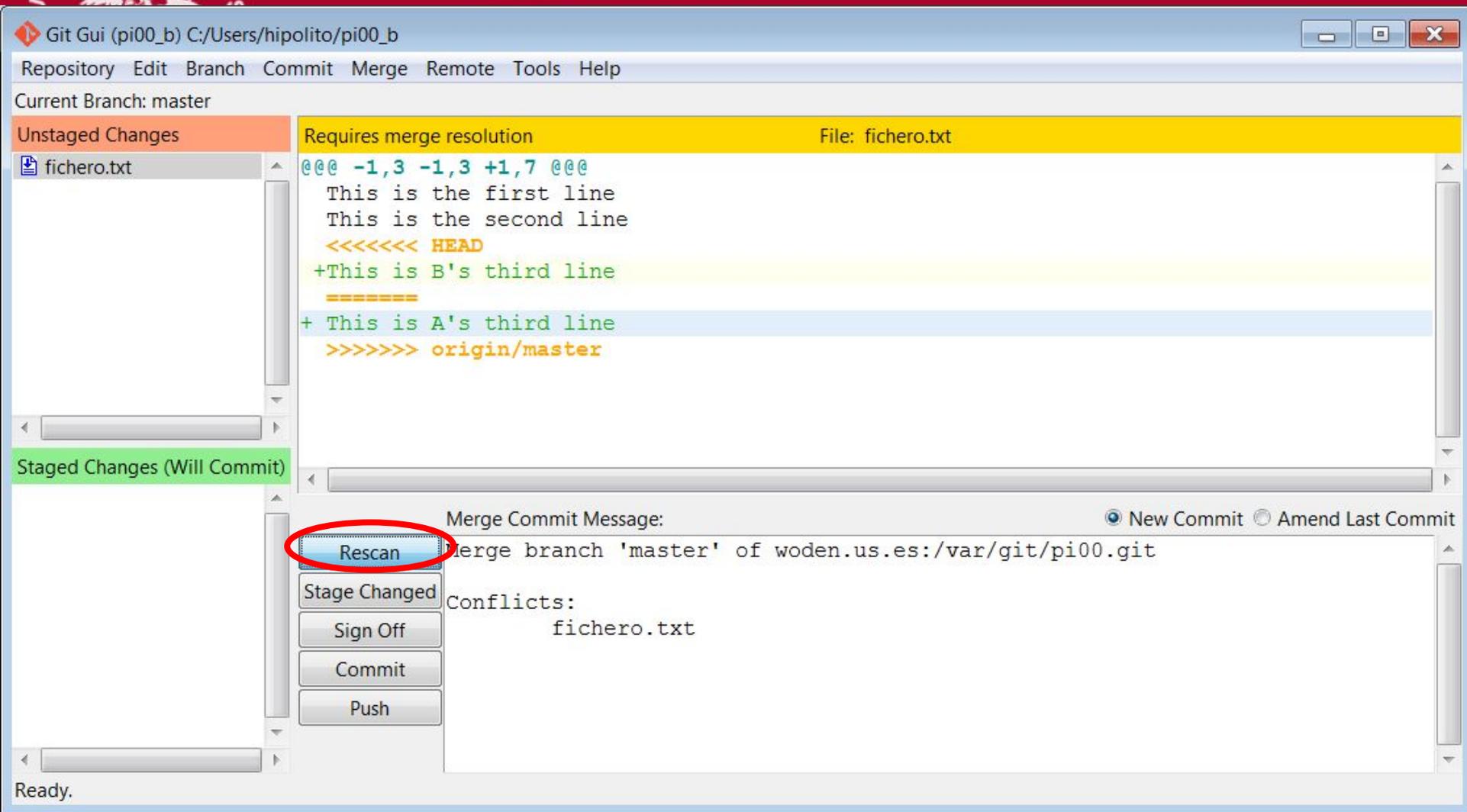


```
fichero.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
This is the first line
This is the second line
<<<<<<< HEAD
This is B's third line
=====  
This is A's third line
>>>>>>> origin/master
|
```



Solucionamos el conflicto:

- Decidiendo qué versión es la correcta
- Eliminando los caracteres de control (<<, >>, ==)



Pulsamos 'Rescan' para buscar cambios en el workspace

Git Gui (pi00\_b) C:/Users/hipolito/pi00\_b

Repository Edit Branch Commit Merge Remote Tools Help

Current Branch: master

Unstaged Changes

Requires merge resolution File: fichero.txt

```
@@@ -1,3 -1,3 +1,3 @@@
  This is the first line
  This is the second line
- This is B's third line
-This is A's third line
++This is A and B's third line
```

Staged Changes (Will Commit)

Merge Commit Message:  New Commit  Amend Last Commit

Merge branch 'master' of woden.us.es:/var/git/pi00.git

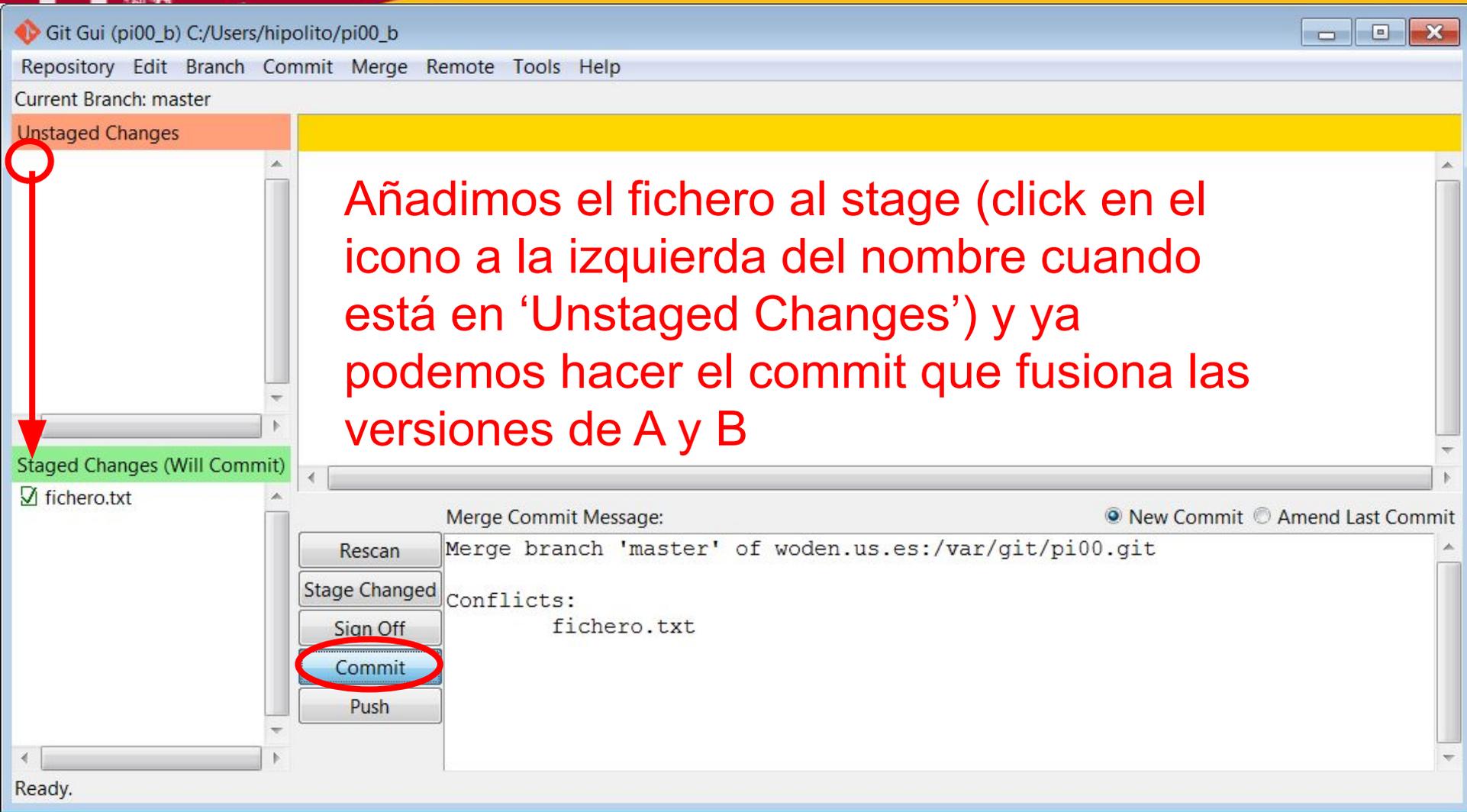
Conflicts:

fichero.txt

Rescan  
Stage Changed  
Sign Off  
Commit  
Push

Ready.

Vemos la versión corregida del fichero



Git Gui (pi00\_b) C:/Users/hipolito/pi00\_b

Repository Edit Branch Commit Merge Remote Tools Help

Current Branch: master

Unstaged Changes

Staged Changes (Will Commit)

fichero.txt

Rescan

Stage Changed

Sign Off

**Commit**

Push

Merge Commit Message:  New Commit  Amend Last Commit

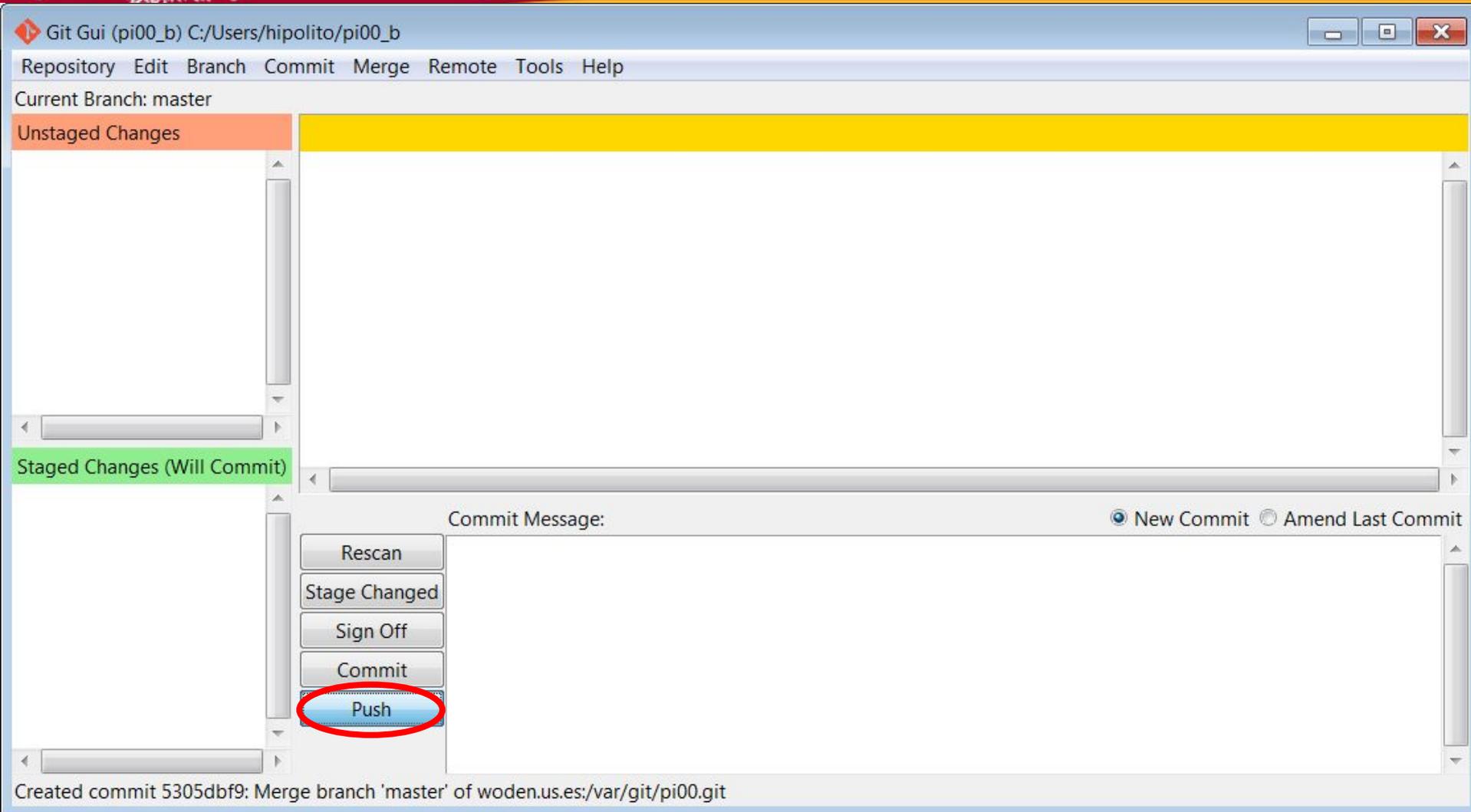
Merge branch 'master' of woden.us.es:/var/git/pi00.git

Conflicts:

fichero.txt

Ready.

Añadimos el fichero al stage (click en el icono a la izquierda del nombre cuando está en 'Unstaged Changes') y ya podemos hacer el commit que fusiona las versiones de A y B

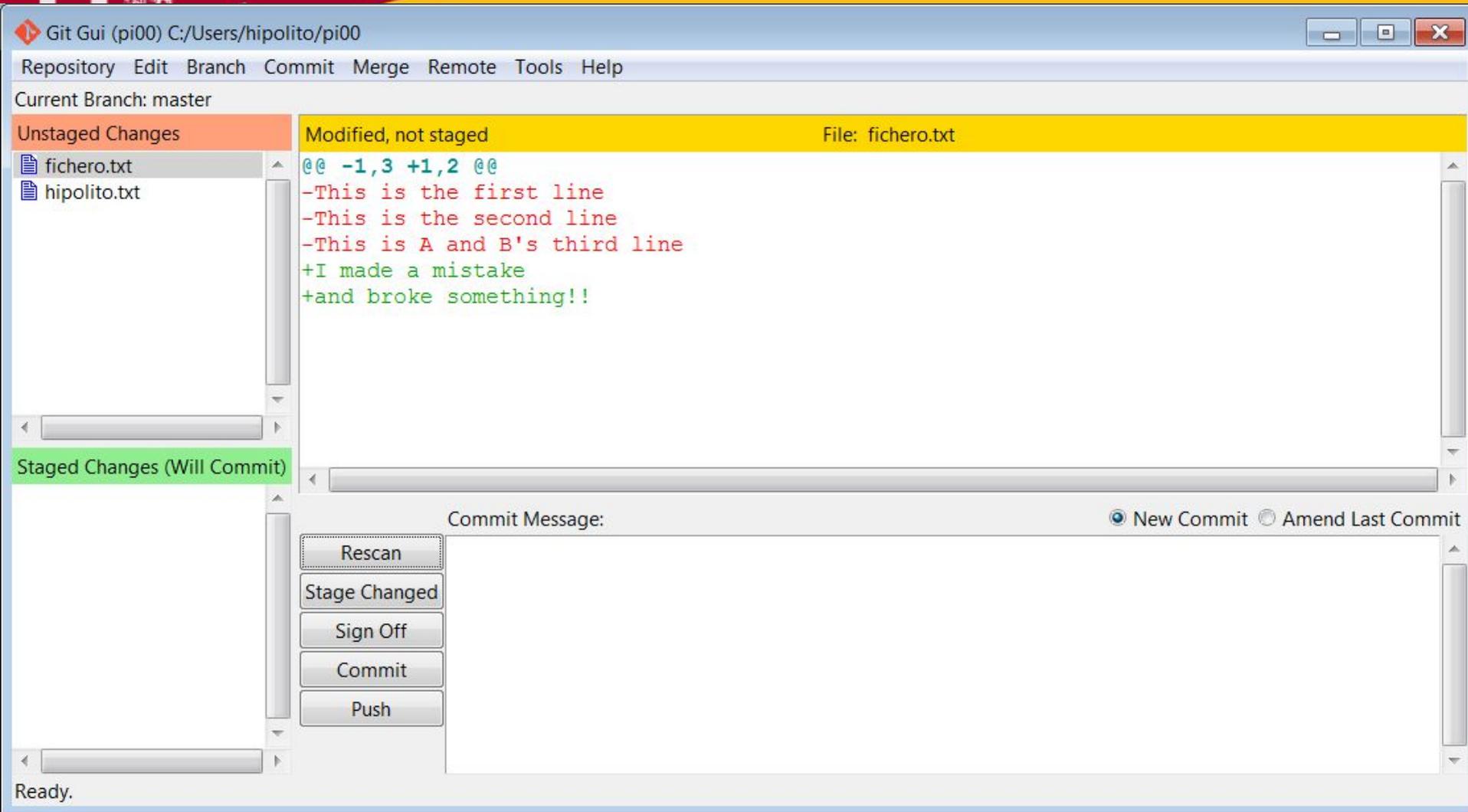


Y ya podemos empujar este nuevo commit al repositorio remoto

# Contenido

- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- Push
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout

# Revertir errores usando checkout



The screenshot shows the Git GUI application window. The title bar reads "Git Gui (pi00) C:/Users/hipolito/pi00". The menu bar includes "Repository", "Edit", "Branch", "Commit", "Merge", "Remote", "Tools", and "Help". The status bar at the top indicates "Current Branch: master".

The main workspace is divided into two sections: "Unstaged Changes" and "Staged Changes (Will Commit)".

**Unstaged Changes:** A list on the left shows "fichero.txt" and "hipolito.txt". The "fichero.txt" file is selected, and its content is displayed in the main area. The status is "Modified, not staged" and the file name is "File: fichero.txt". The content shows a diff:

```
@@ -1,3 +1,2 @@  
-This is the first line  
-This is the second line  
-This is A and B's third line  
+I made a mistake  
+and broke something!!
```

**Staged Changes (Will Commit):** This section is currently empty.

**Commit Message:** A text area for entering a commit message. To the right of this area are two radio buttons: "New Commit" (selected) and "Amend Last Commit".

**Buttons:** A vertical stack of buttons is located on the left side of the commit message area: "Rescan", "Stage Changed", "Sign Off", "Commit", and "Push".

The status bar at the bottom left of the window reads "Ready."

# Revertir errores usando checkout



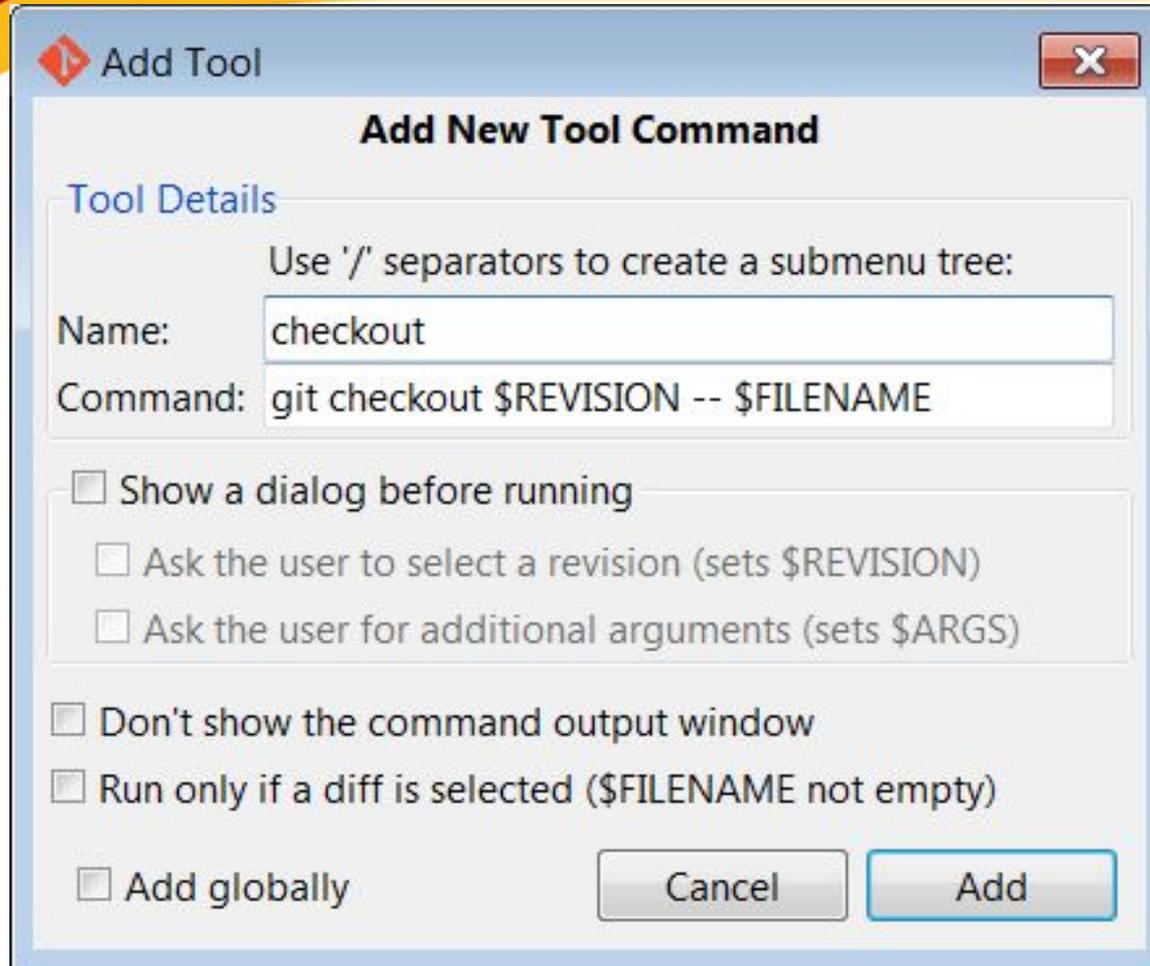
The screenshot shows the Git GUI interface for a repository located at C:/Users/hipolito/pi00. The current branch is master. The interface is divided into several sections:

- Unstaged Changes:** A list on the left shows 'fichero.txt' and 'hipolito.txt'. The main area displays a diff for 'fichero.txt' with the following content:

```
@@ -1,3 +1,2 @@  
-This is the first line  
-This is the second line  
-This is A and B's third line  
+I made a mistake  
+and broke something!!
```
- Staged Changes (Will Commit):** This section is currently empty.
- Commit Message:** A text area for entering a commit message, with radio buttons for 'New Commit' (selected) and 'Amend Last Commit'.
- Buttons:** A vertical stack of buttons on the left includes 'Rescan', 'Stage Changed', 'Sign Off', 'Commit', and 'Push'.

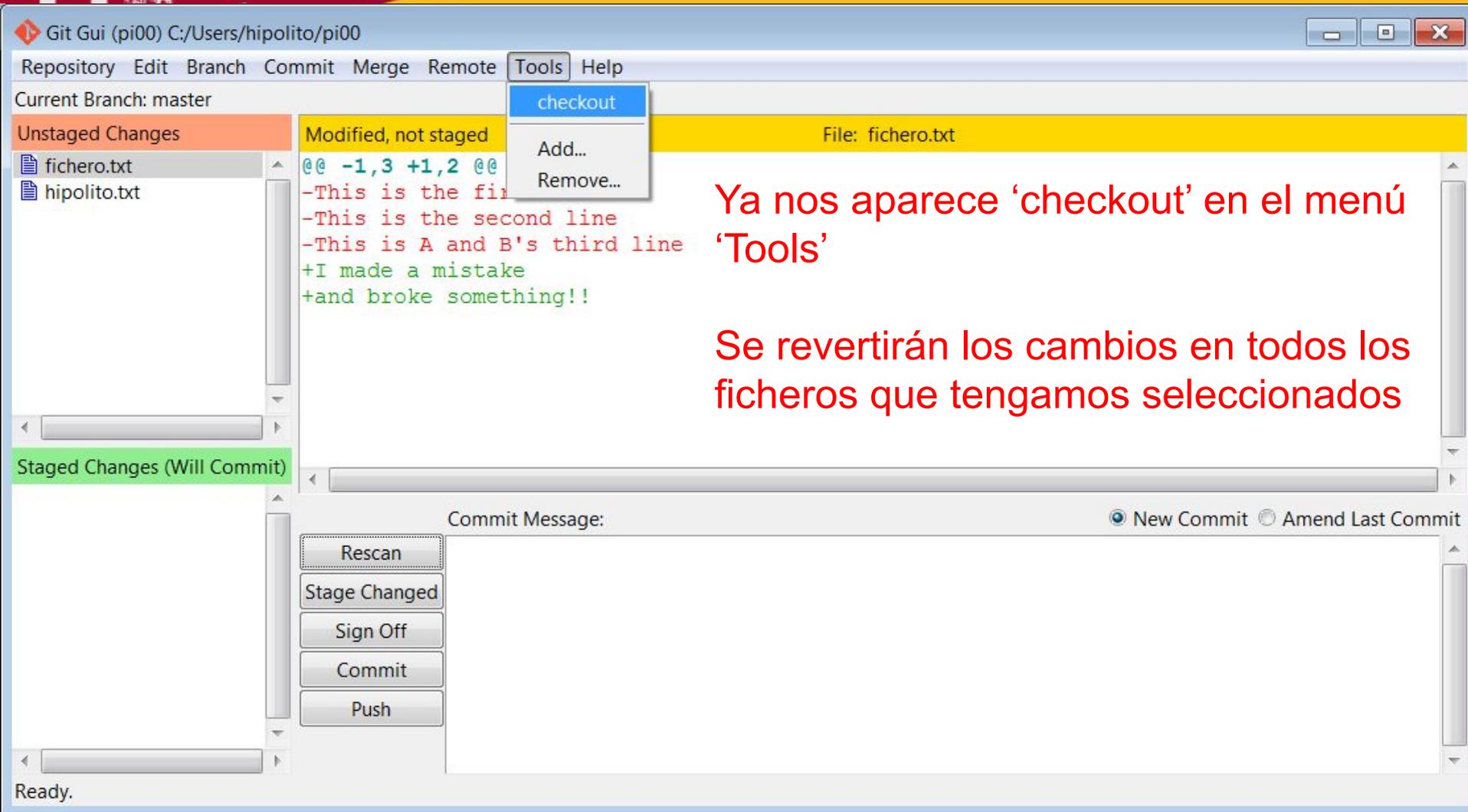
The status bar at the bottom left indicates 'Ready.'.

# Revertir errores usando checkout



(Sólo tenemos que añadir el comando si no lo tenemos de antes)

# Revertir errores usando checkout



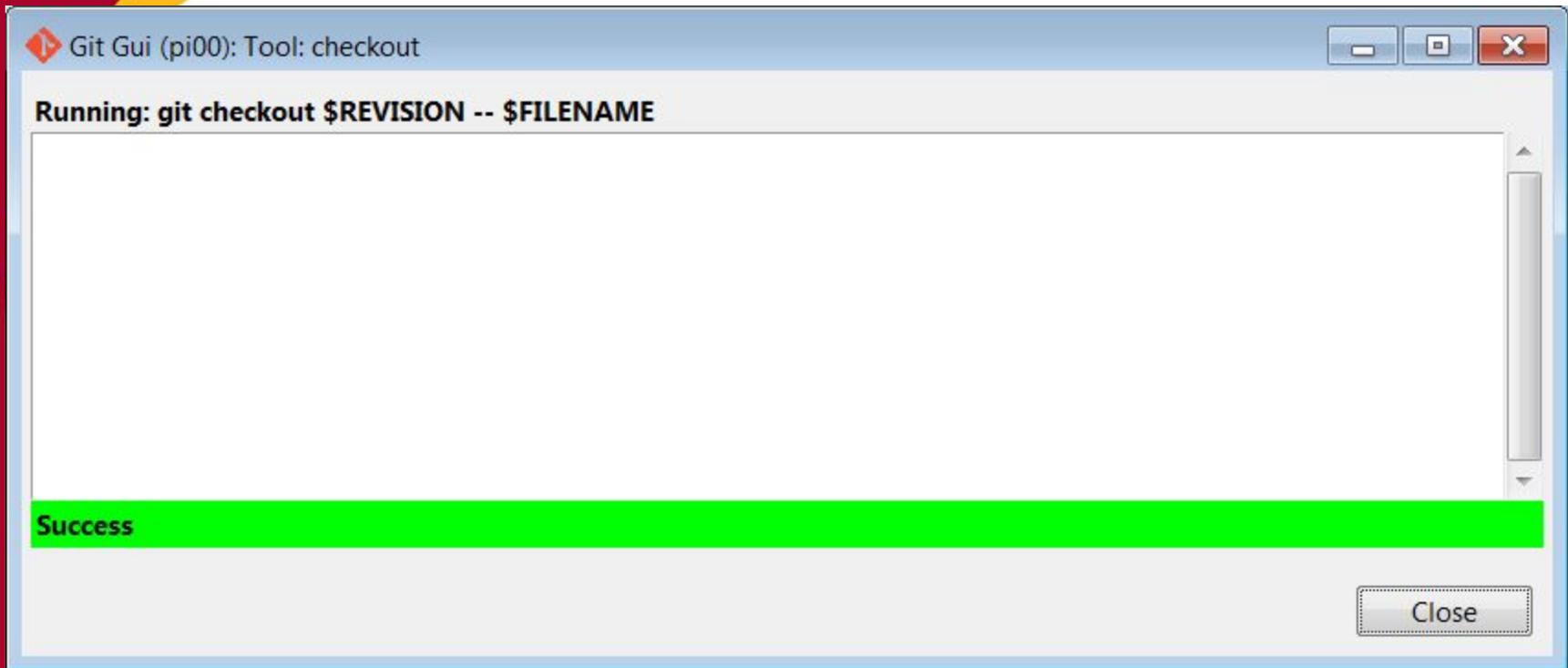
The screenshot shows the Git GUI interface. The 'Tools' menu is open, and the 'checkout' option is highlighted. The main window displays the file 'fichero.txt' with the following content:

```
@@ -1,3 +1,2 @@  
-This is the first line  
-This is the second line  
-This is A and B's third line  
+I made a mistake  
+and broke something!!
```

The interface also shows the 'Staged Changes (Will Commit)' section, which is currently empty. The 'Commit Message' field is visible, along with buttons for 'Rescan', 'Stage Changed', 'Sign Off', 'Commit', and 'Push'. The status bar at the bottom indicates 'Ready.'.

Ya nos aparece 'checkout' en el menú 'Tools'

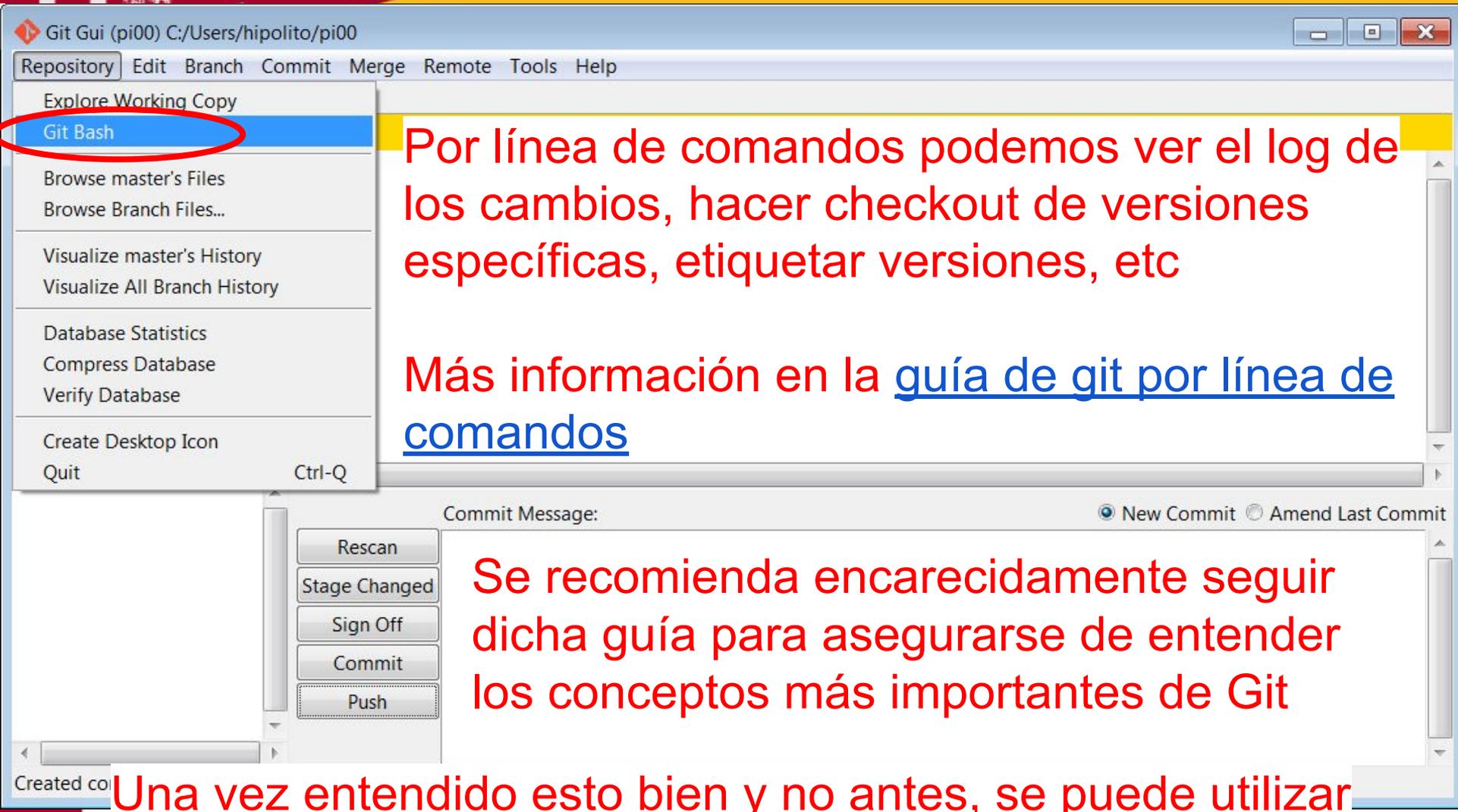
Se revertirán los cambios en todos los ficheros que tengamos seleccionados



Tras esto, los ficheros que hayamos 'checked out' estarán igual que la versión del repo

**Para profundizar**

# Para profundizar



**Valor... y al repo!**

