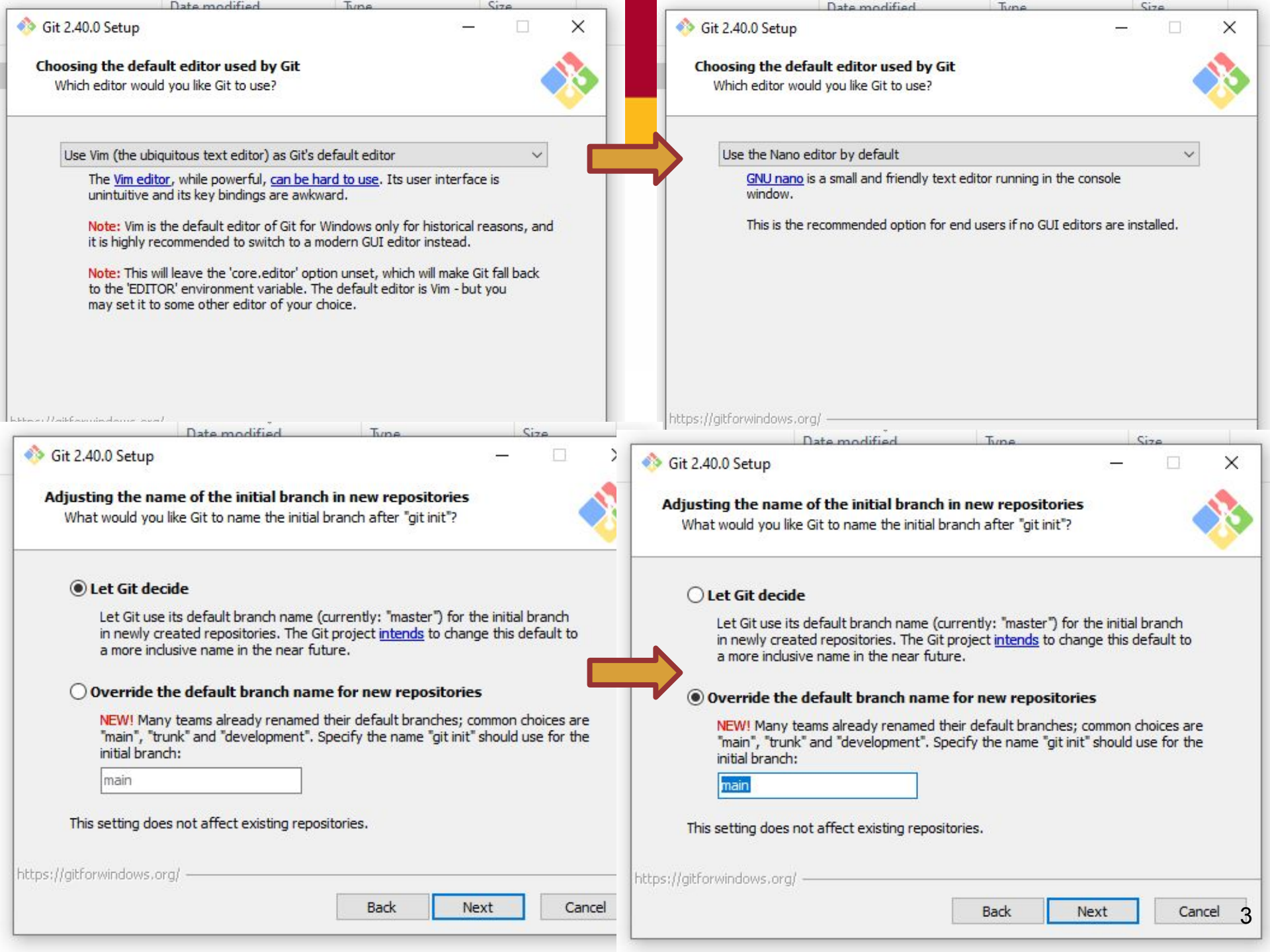
Guía rápida de Git en modo gráfico

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.
https://gitforwindows.org/
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:
main
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:
main
This setting does not affect existing repositories.
https://gitforwindows.org/
Back
Next
Cancel

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`"

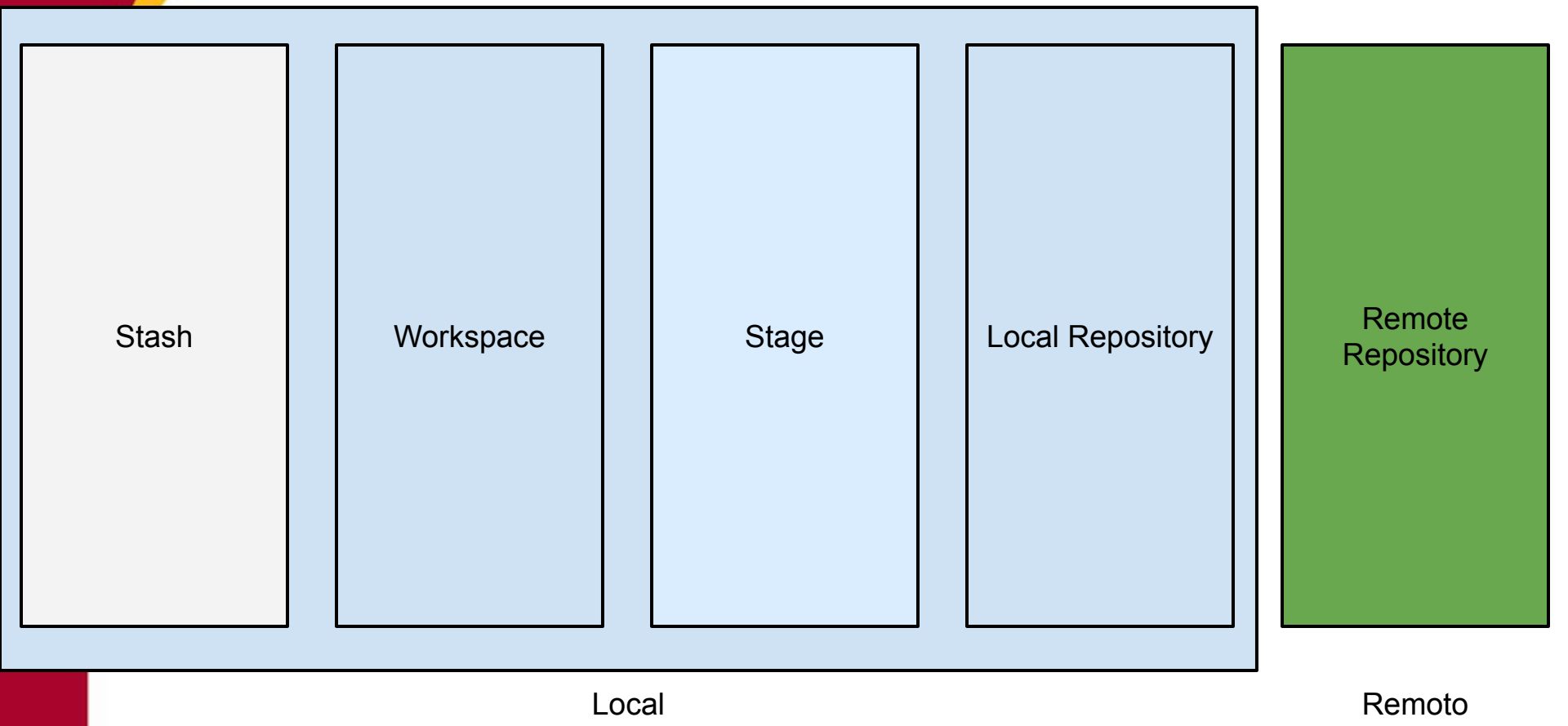
Contenido

- 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

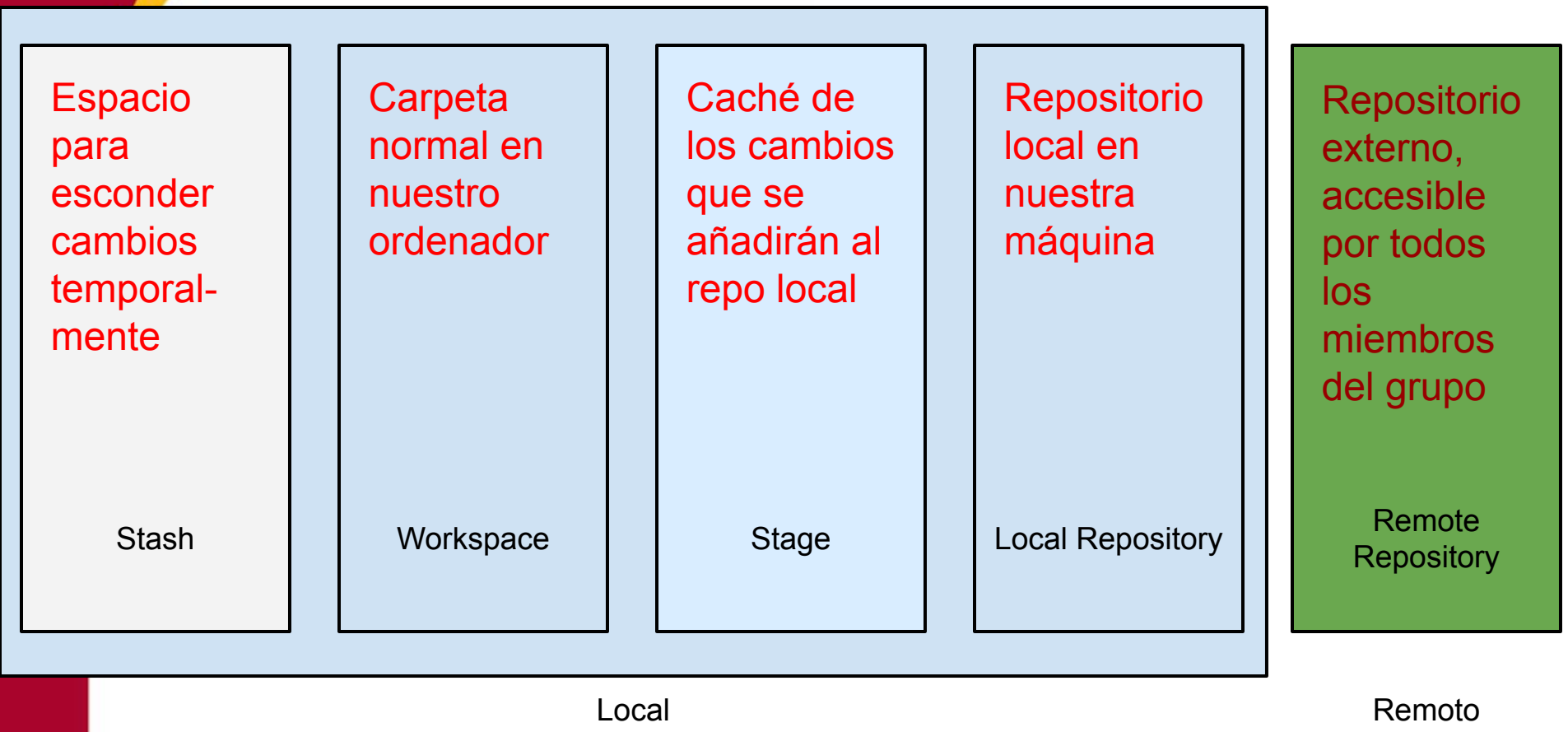
- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- Push
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout

Áreas


Stash
Workspace
Stage
Local Repository
Remote Repository
Local
Remoto

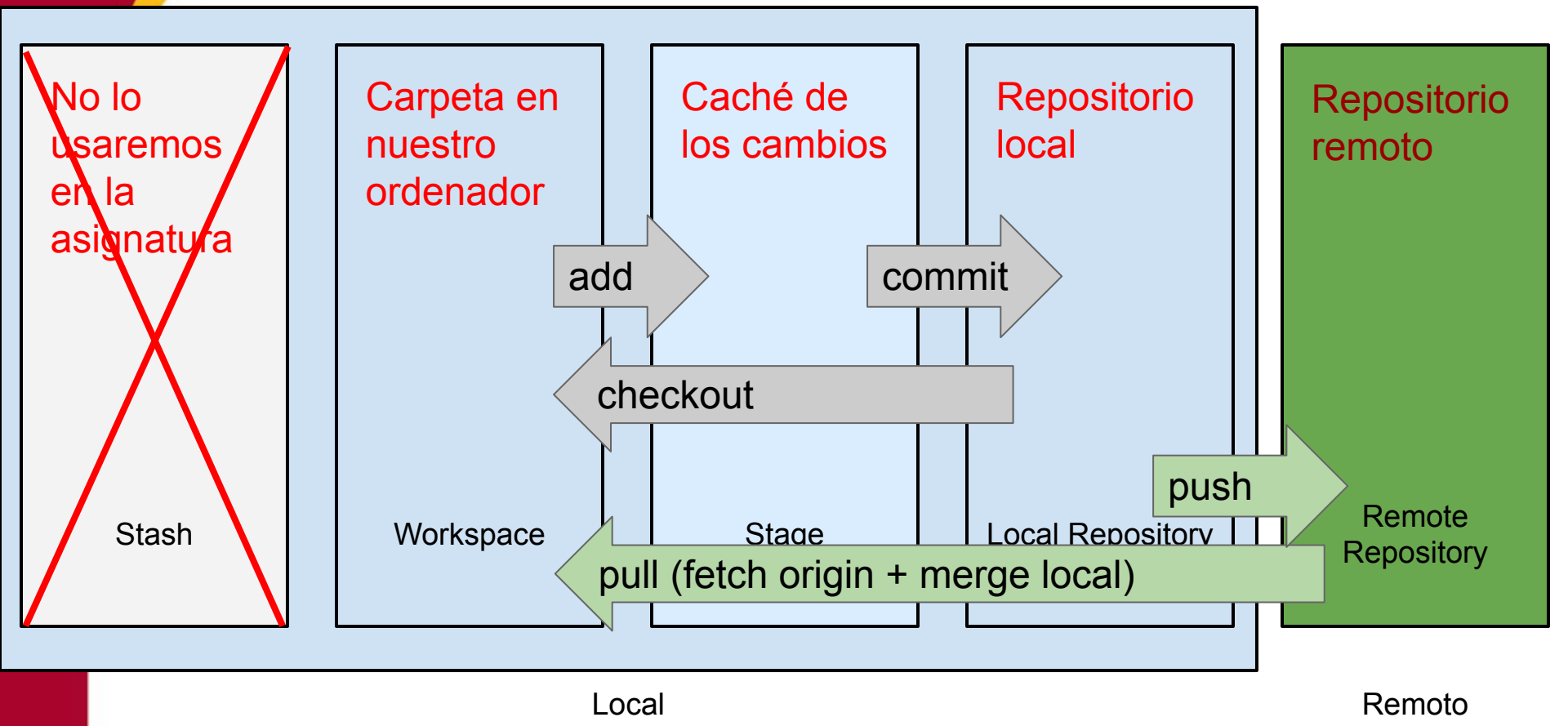
Áreas de git

Descripción



Áreas de git

Comandos

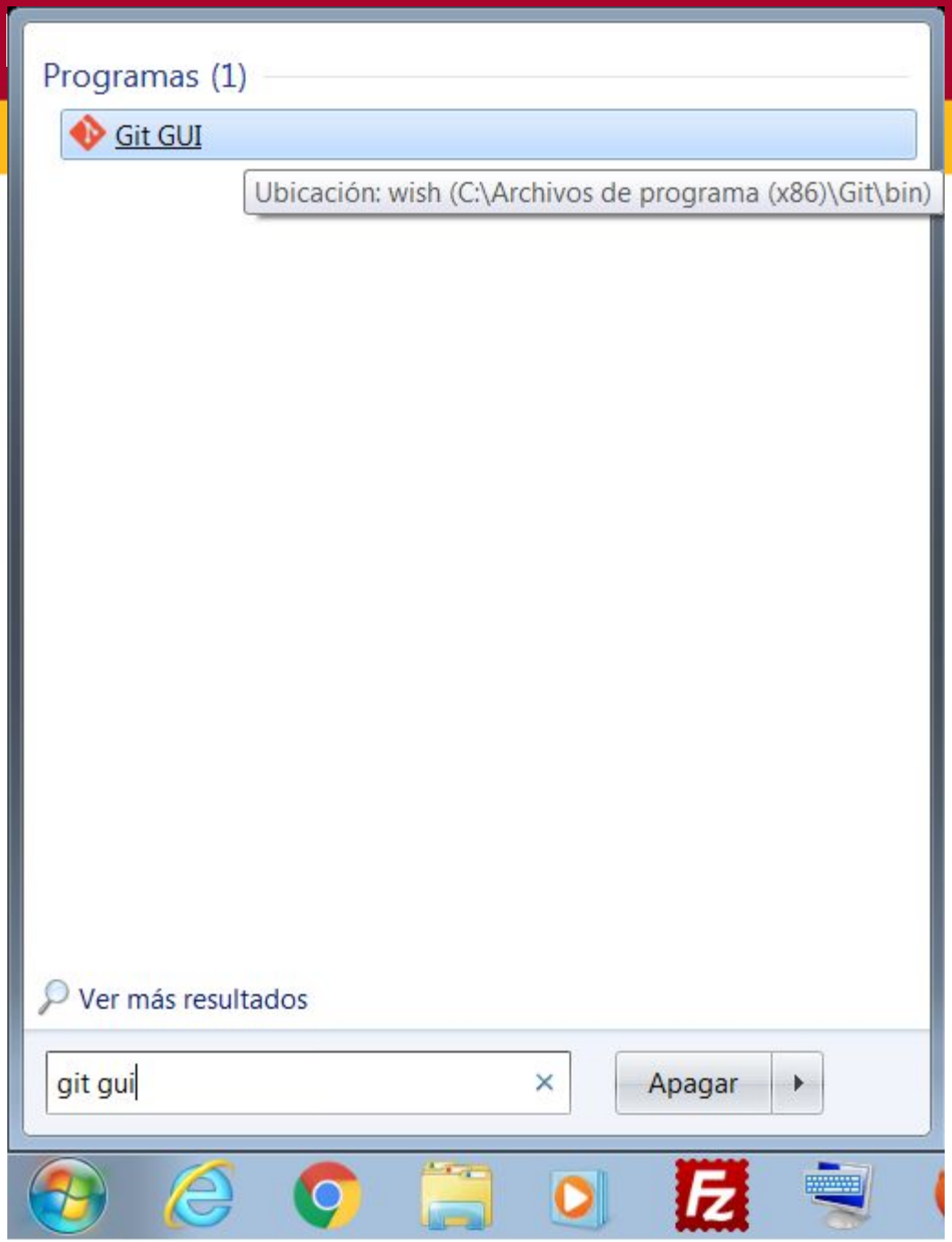


Contenido

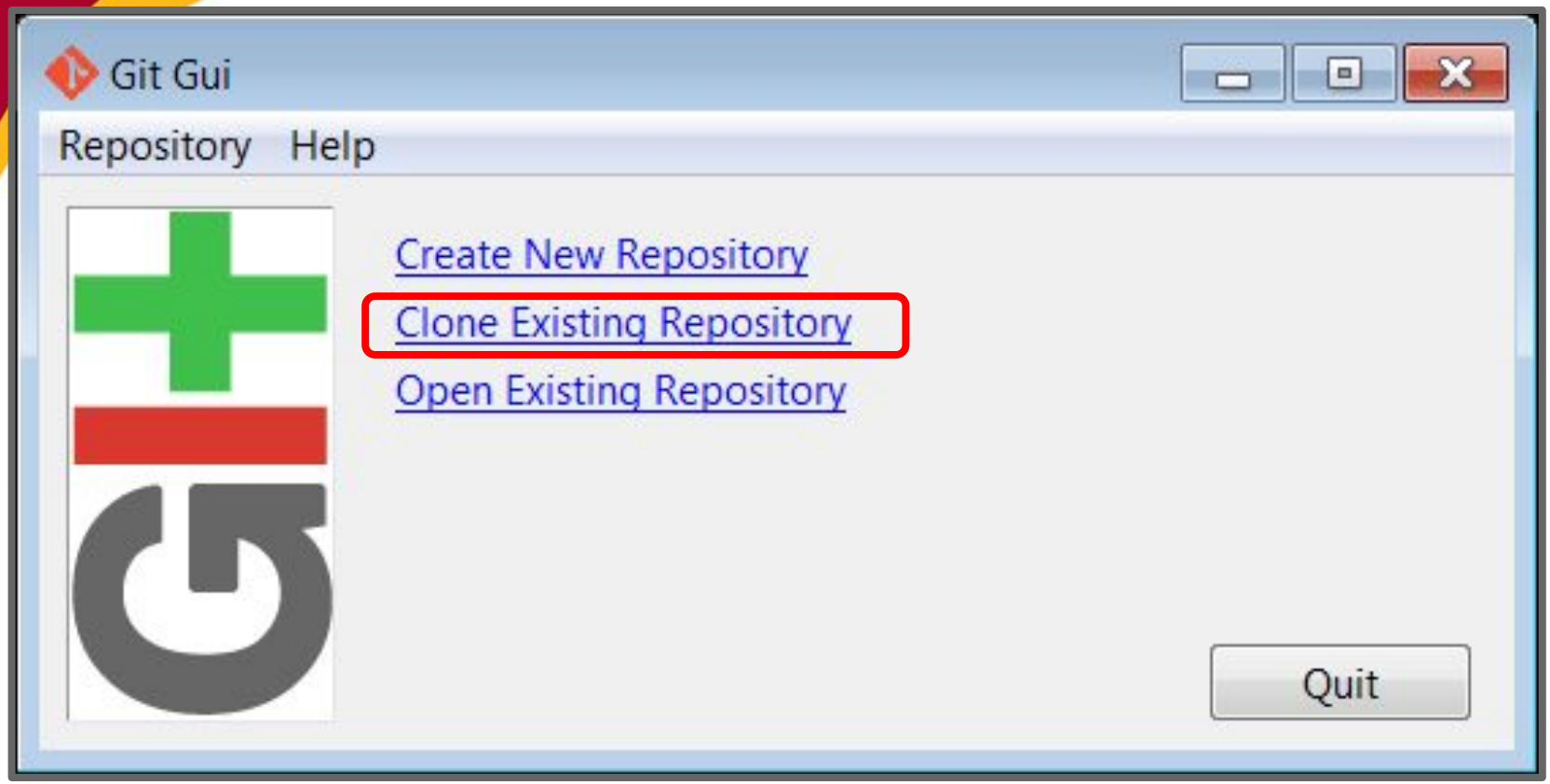
- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- Push
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout

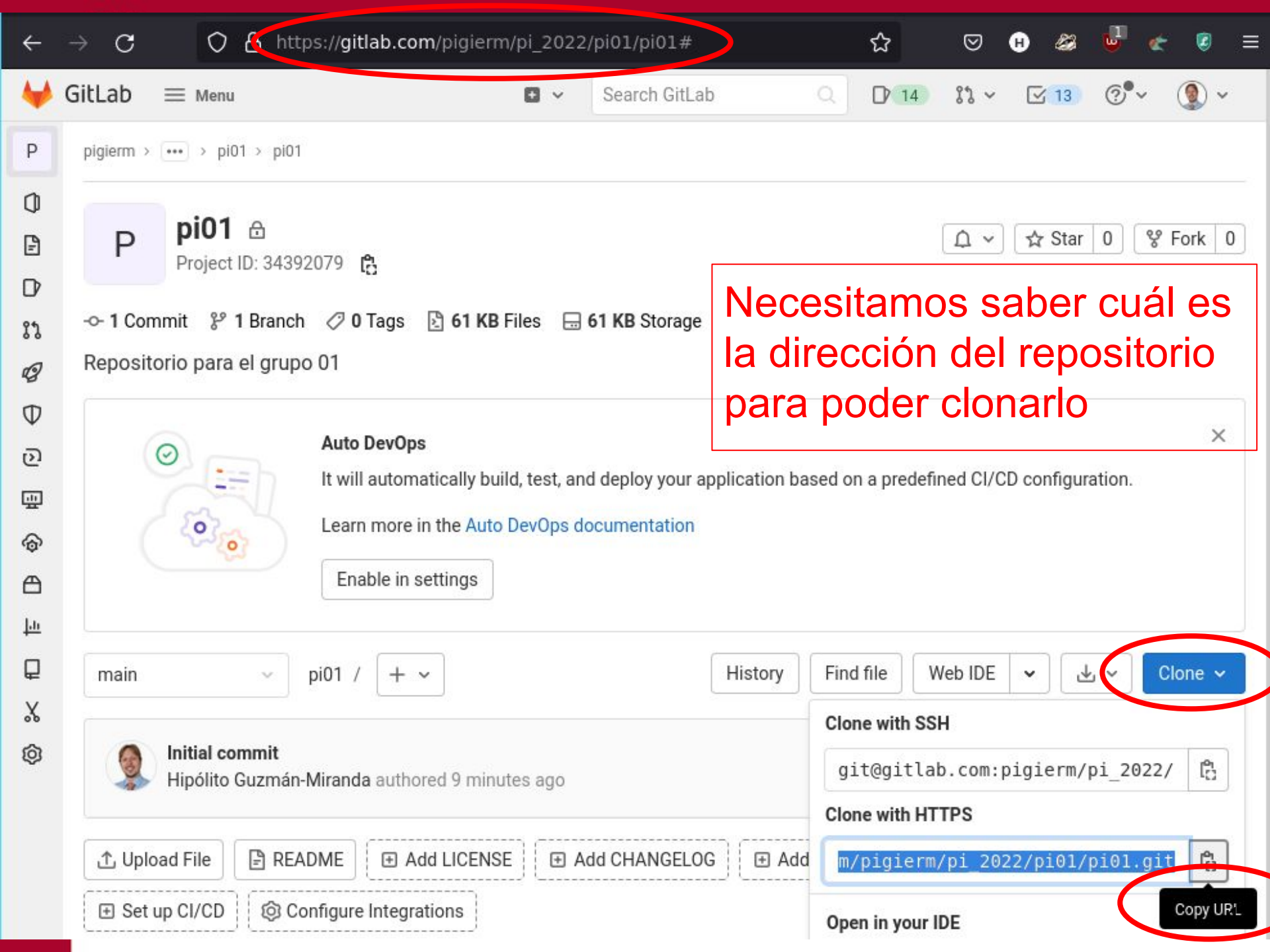
C

Abrimos
Git GUI

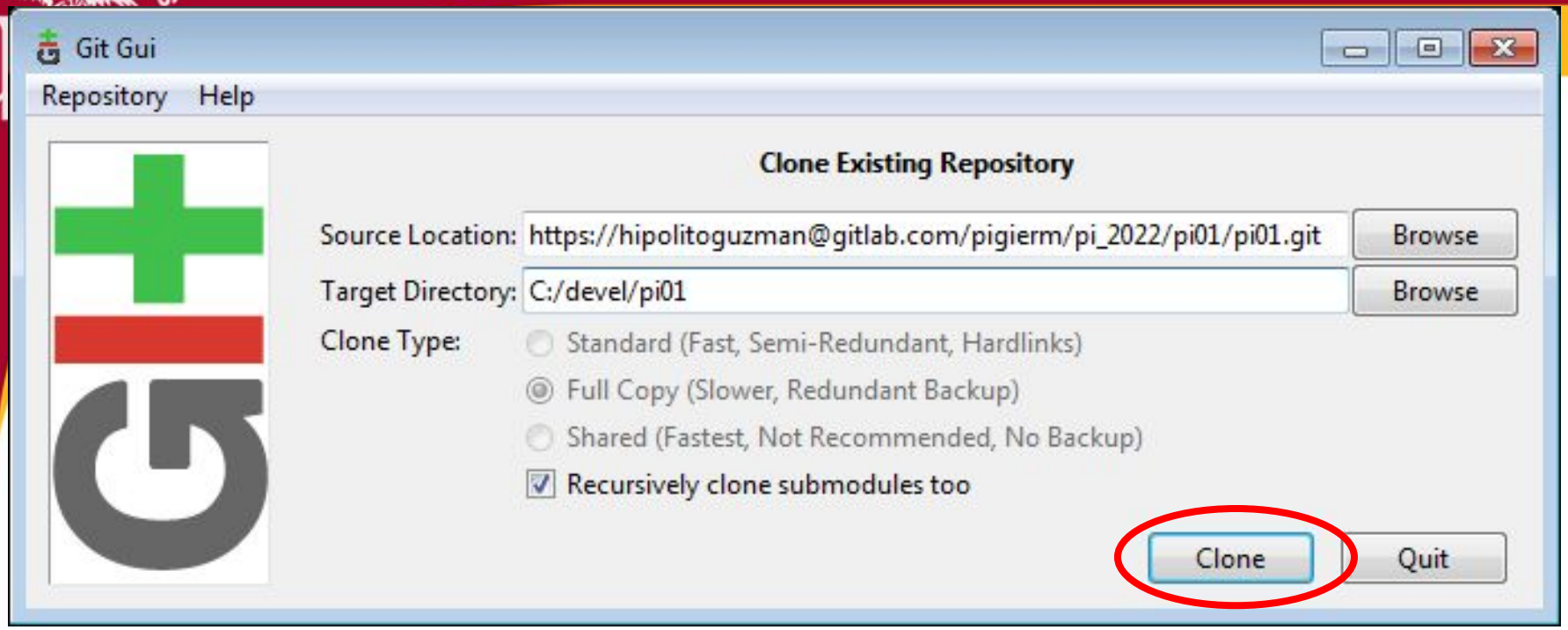


Clonar el repo



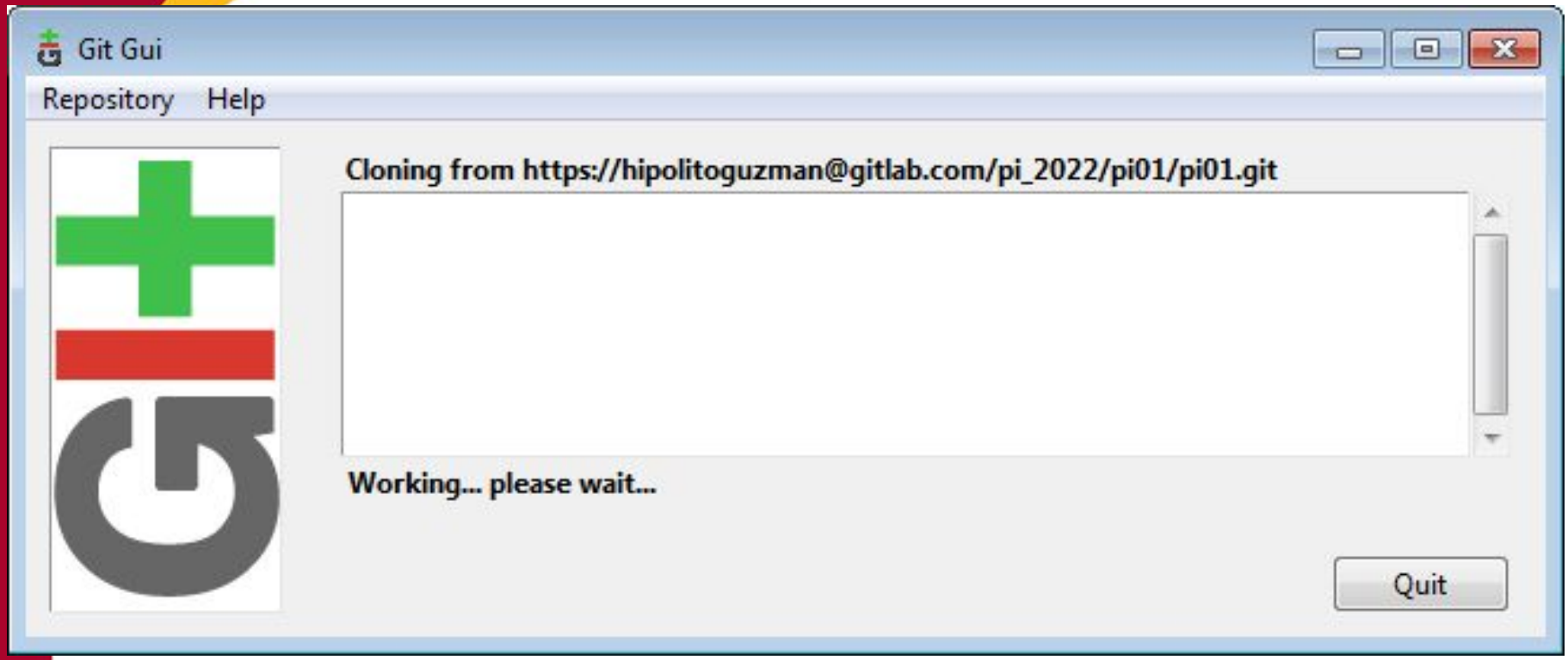

https://gitlab.com/pigierm/pi_2022/pi01/pi01#
GitLab
Menu
Search GitLab
pigierm › ••• › pi01 › pi01
P
pi01
Project ID: 34392079
Star 0
Fork 0
1 Commit
1 Branch
0 Tags
61 KB Files
61 KB Storage
Repositorio para el grupo 01
Necesitamos saber cuál es la dirección del repositorio para poder clonarlo
Auto 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
Clone
Clone with SSH
git@gitlab.com:pigierm/pi_2022/
Clone with HTTPS
m/pigierm/pi_2022/pi01/pi01.git
Copy URL
Open in your IDE
Initial commit
Hipólito Guzmán-Miranda authored 9 minutes ago
Upload File
README
Add LICENSE
Add CHANGELOG
Add
Set up CI/CD
Configure Integrations

Clonar el repo

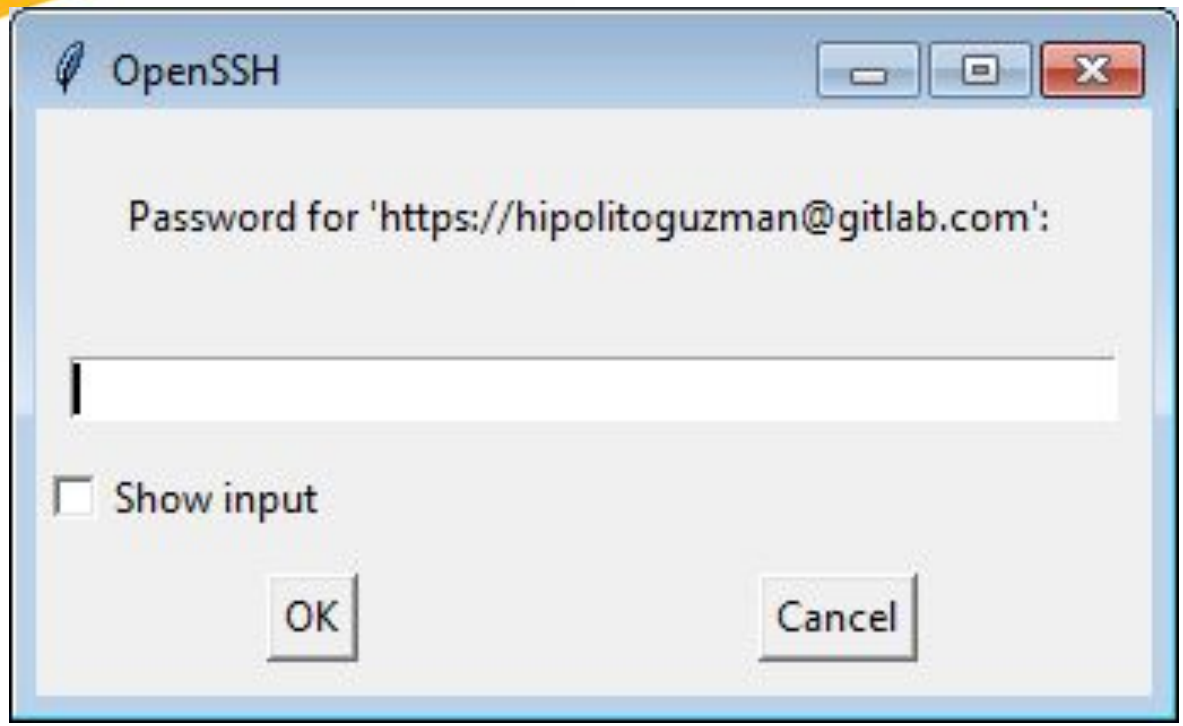


- Podemos pegar directamente el enlace copiado de gitlab en “Source location”
- El target directory será creado al clonar el repositorio (no debe existir de antes!)

Clonar el repo



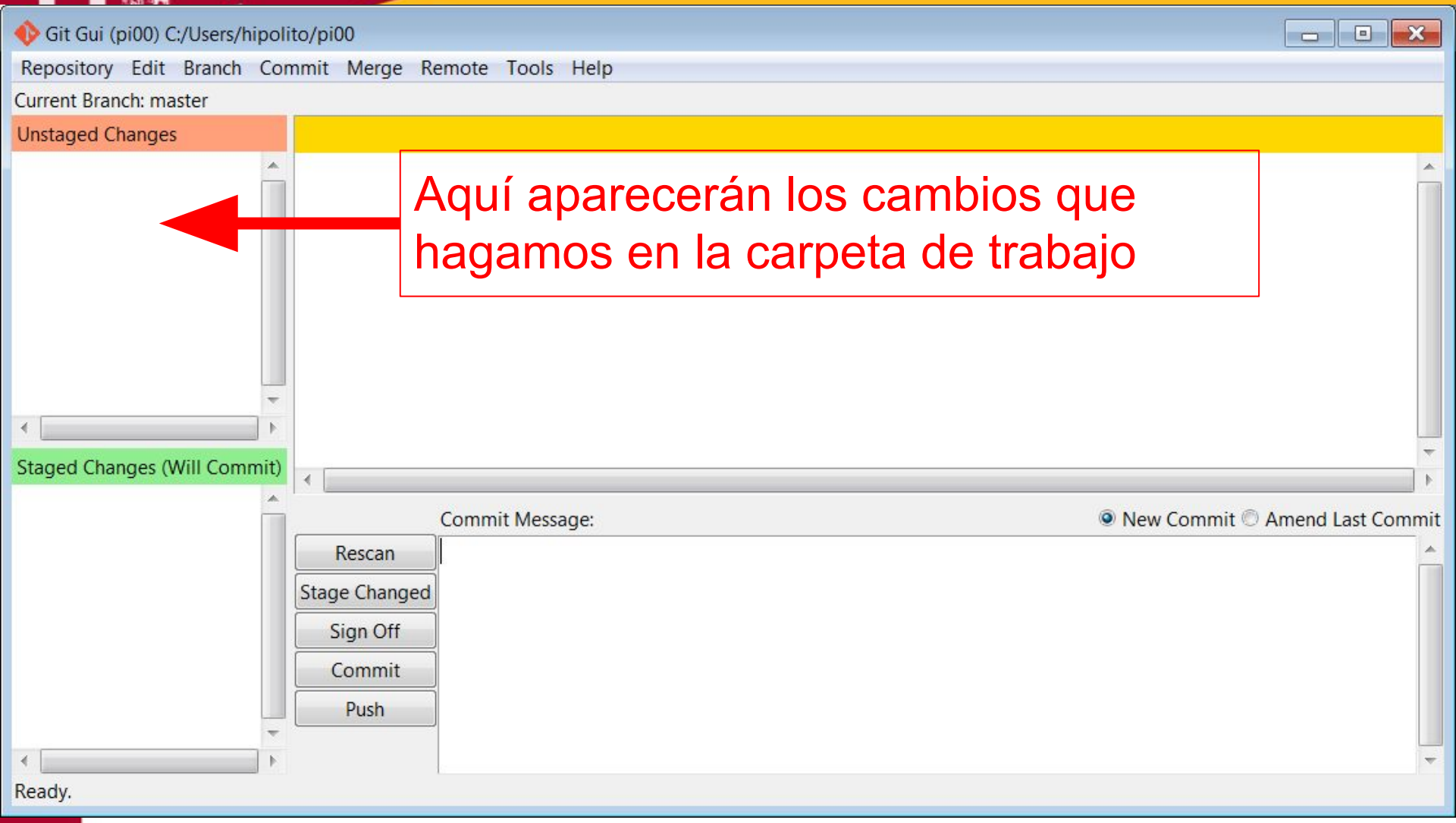
Clonar el repo

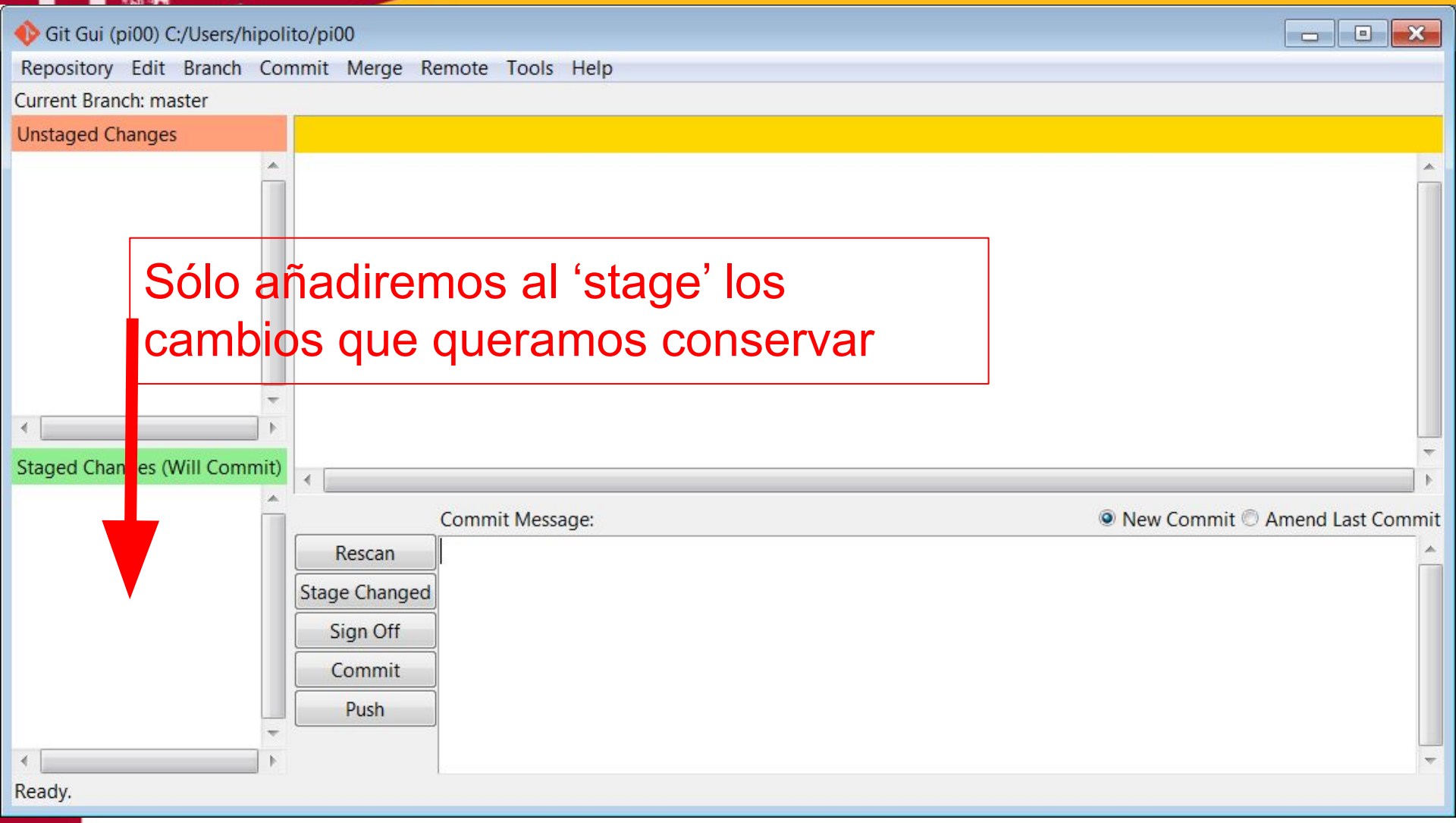


Siempre que hagamos una operación con el repositorio remoto nos pedirá el password (al clonar puede que nos lo pida más de una vez)

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


Git Gui (pi00) C:/Users/hipolito/pi00
Repository Edit Branch Commit Merge Remote Tools Help
Current Branch: master
Unstaged Changes
Staged Changes (Will Commit)
Aquí aparecerán los cambios que hagamos en la carpeta de trabajo
Commit Message:
New Commit
Amend Last Commit
Rescan
Stage Changed
Sign Off
Commit
Push
Ready.

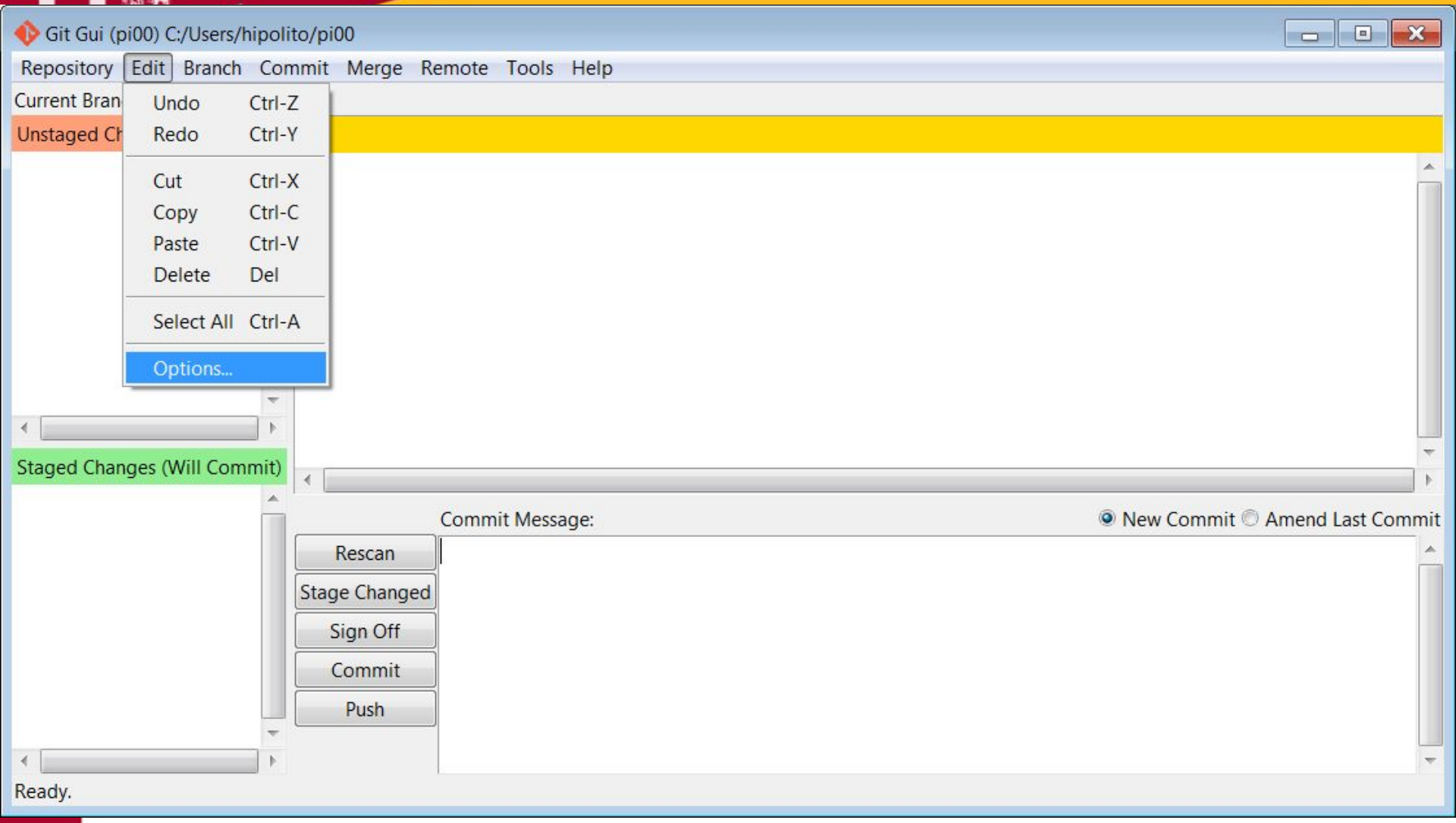


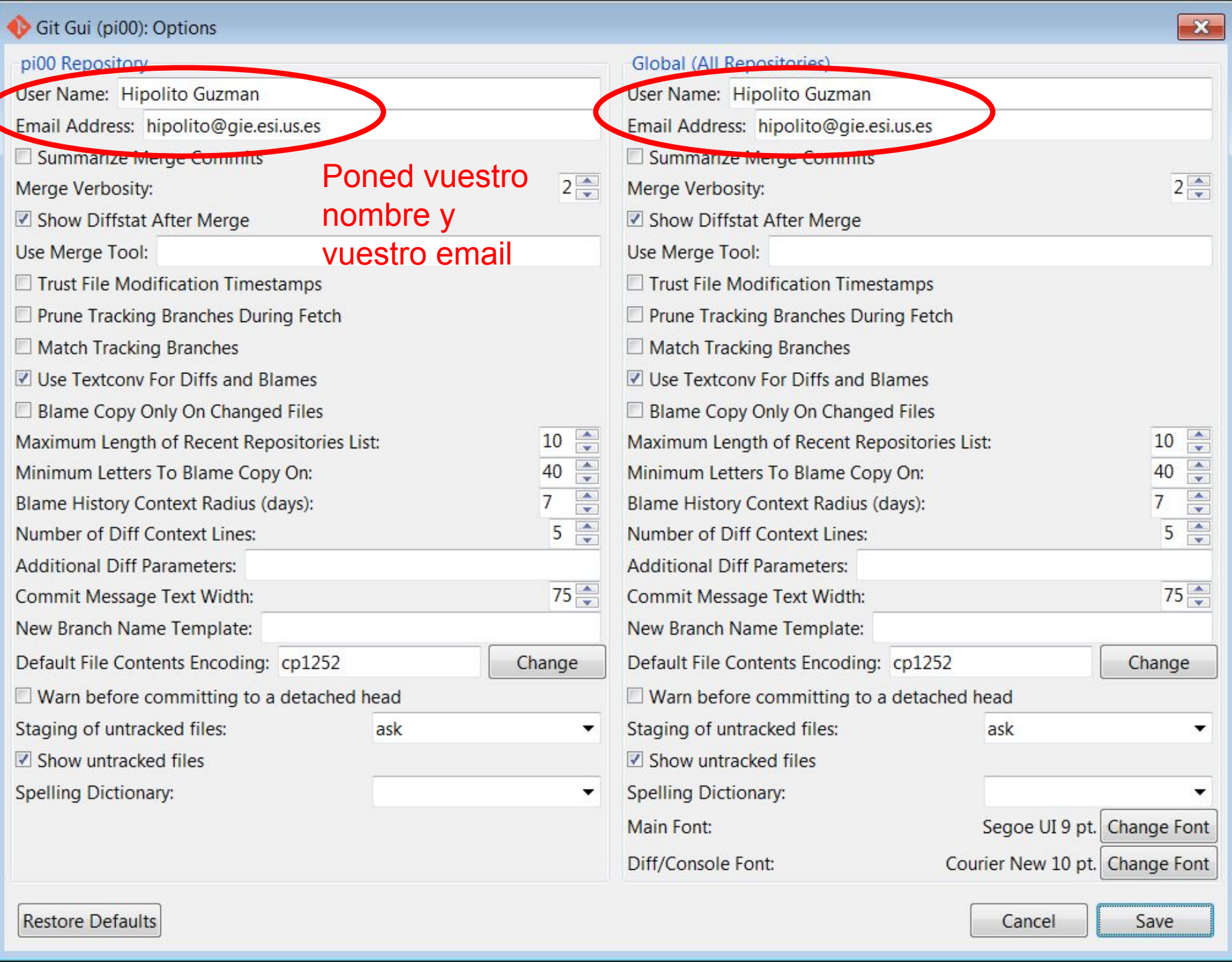
Sólo añadiremos al ‘stage’ los cambios que queramos conservar

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

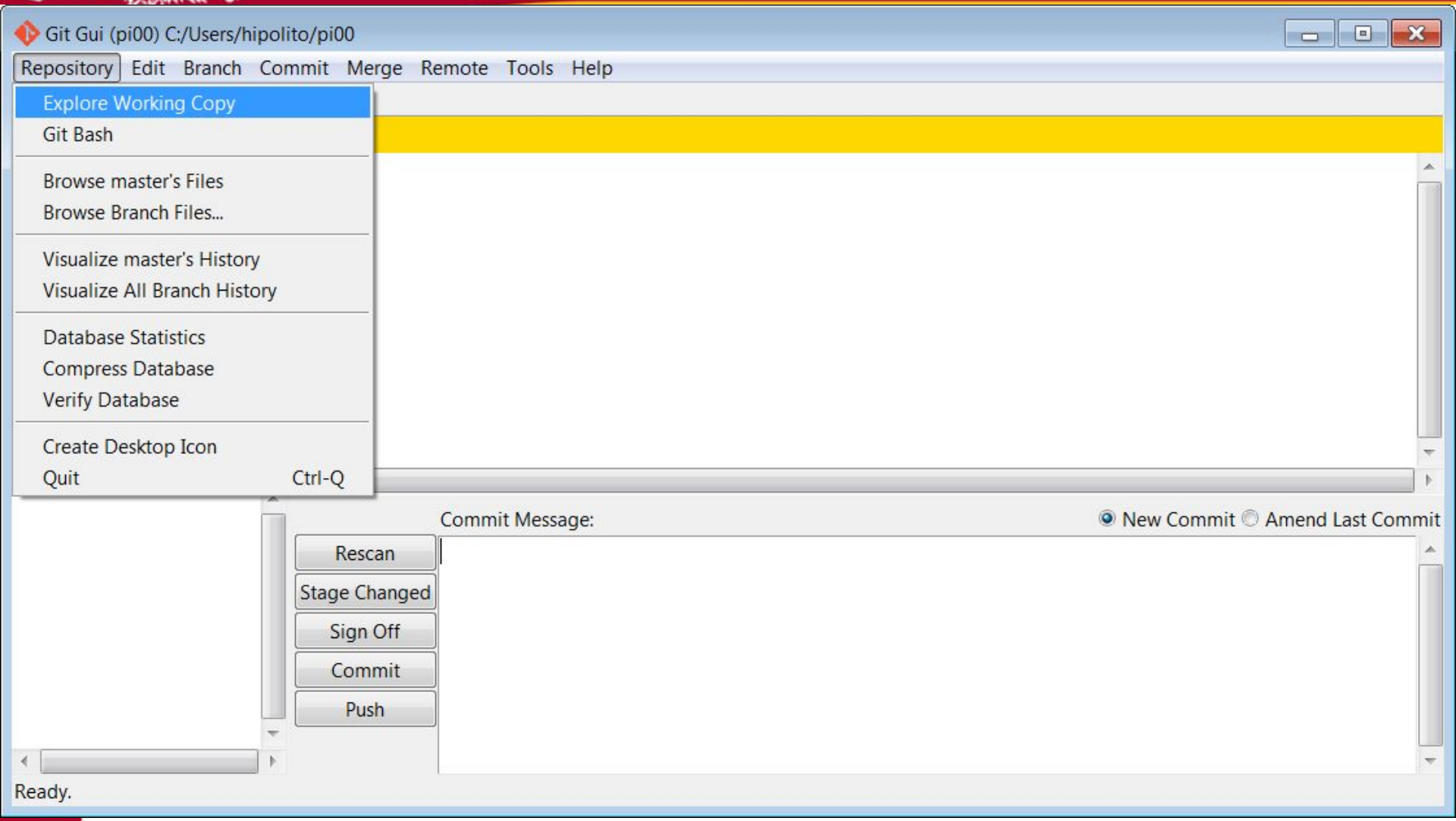



Git Gui (pi00): Options
pi00 Repository
User Name: Hipolito Guzman
Email Address: hipolito@gie.esi.us.es
Summarize Merge Commits
Merge Verbosity: 2
Show Diffstat After Merge
Use Merge Tool:
Trust File Modification Timestamps
Prune Tracking Branches During Fetch
Match Tracking Branches
Use Textconv For 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
Change
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
Change
Warn before committing to a detached head
Staging of untracked files: ask
Show untracked files
Spelling Dictionary:
Main Font: Segoe UI 9 pt.
Change Font
Diff/Console Font: Courier New 10 pt.
Change Font
Restore Defaults
Cancel
Save
Poned vuestro nombre y vuestro email

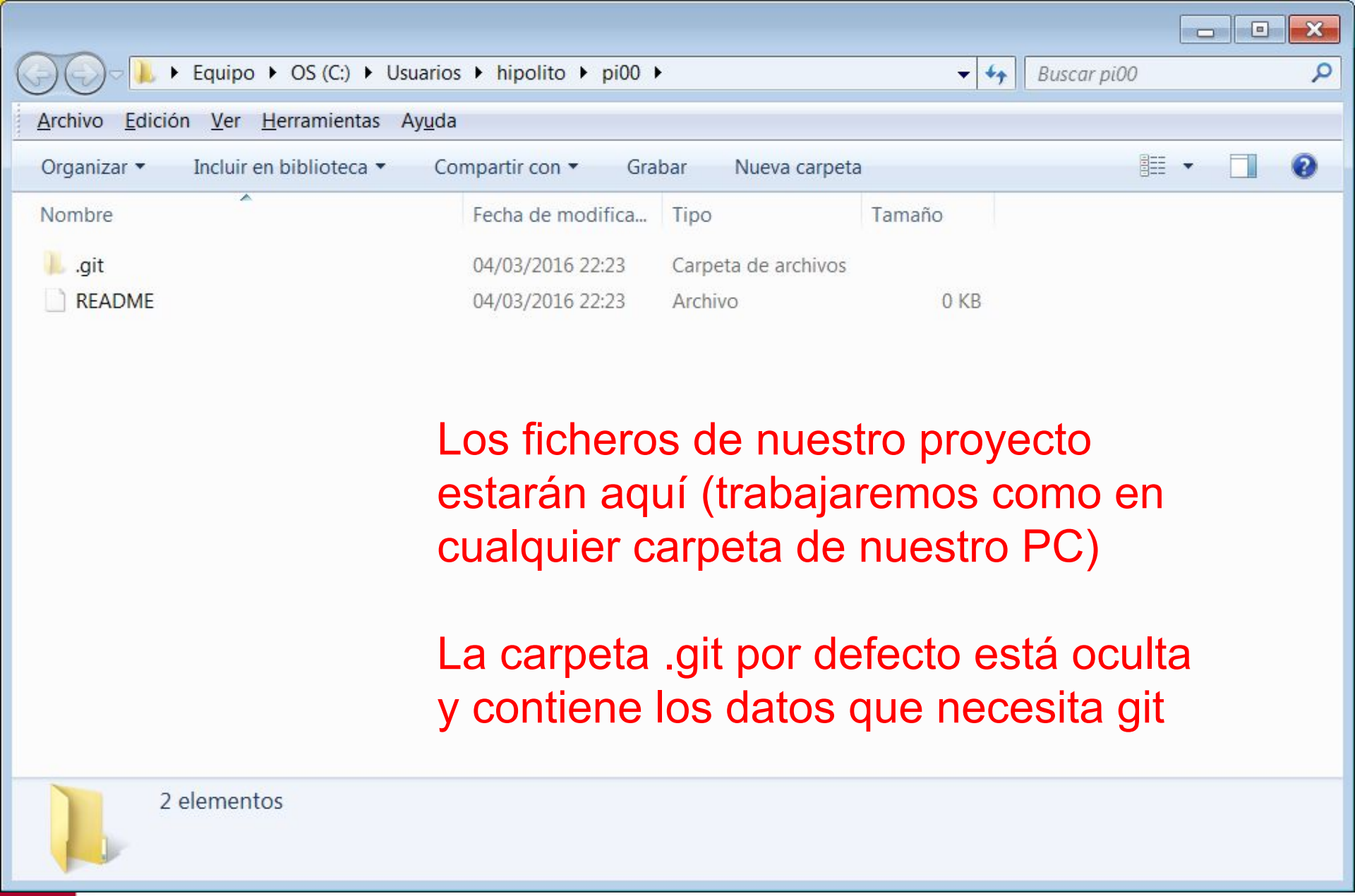
Contenido

- 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

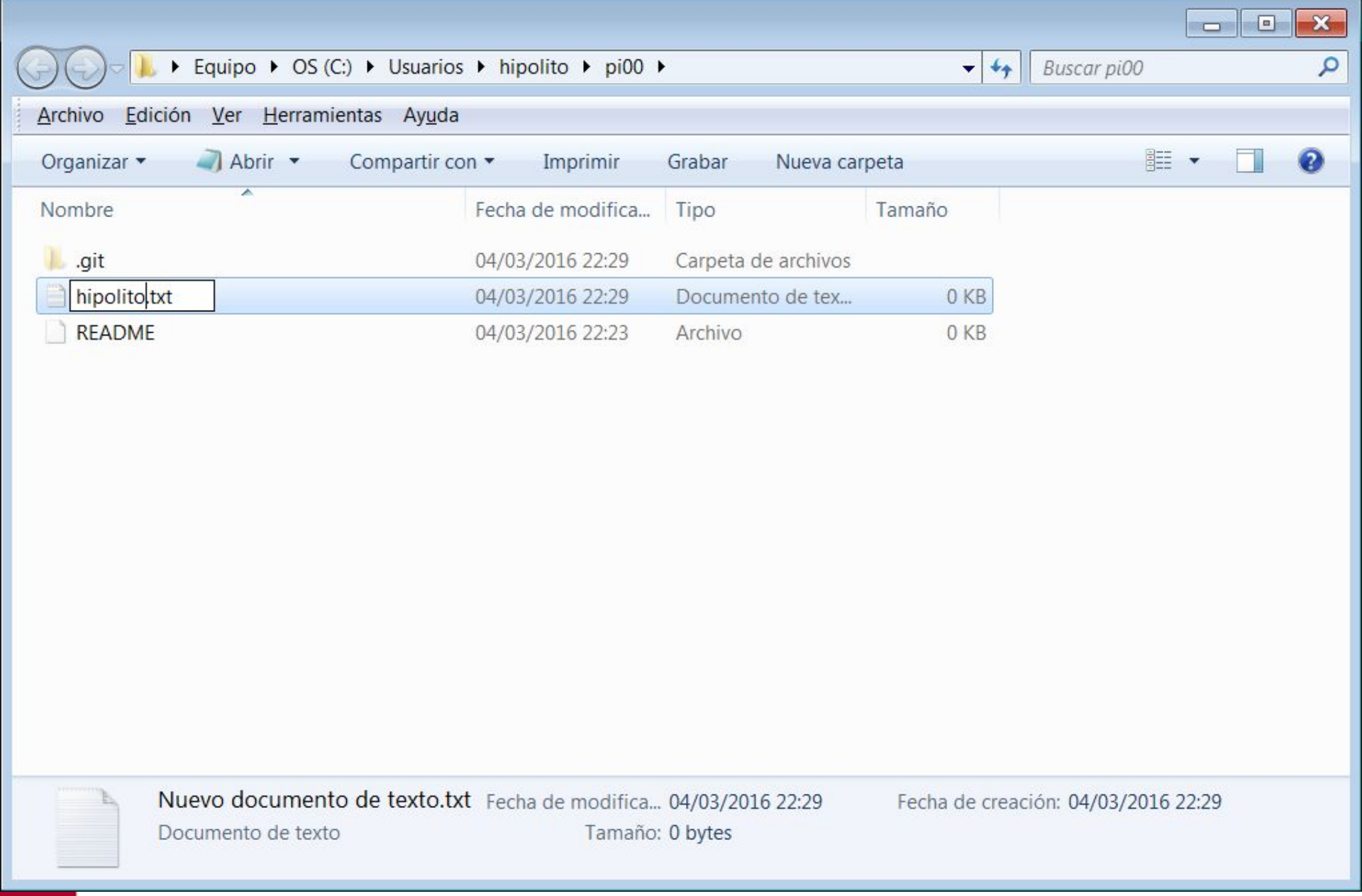
Añadir ficheros al stage



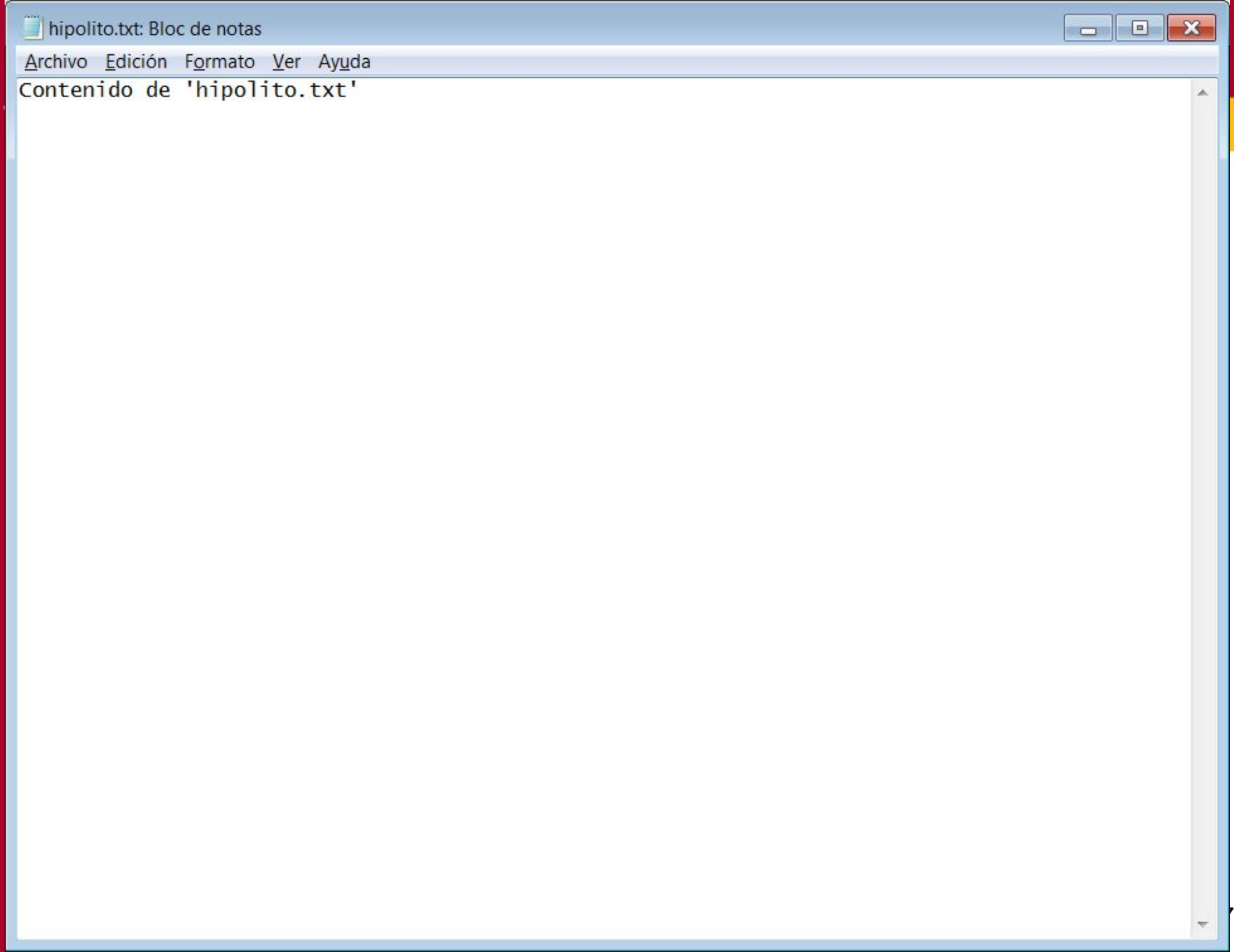
Abrimos la carpeta del repo



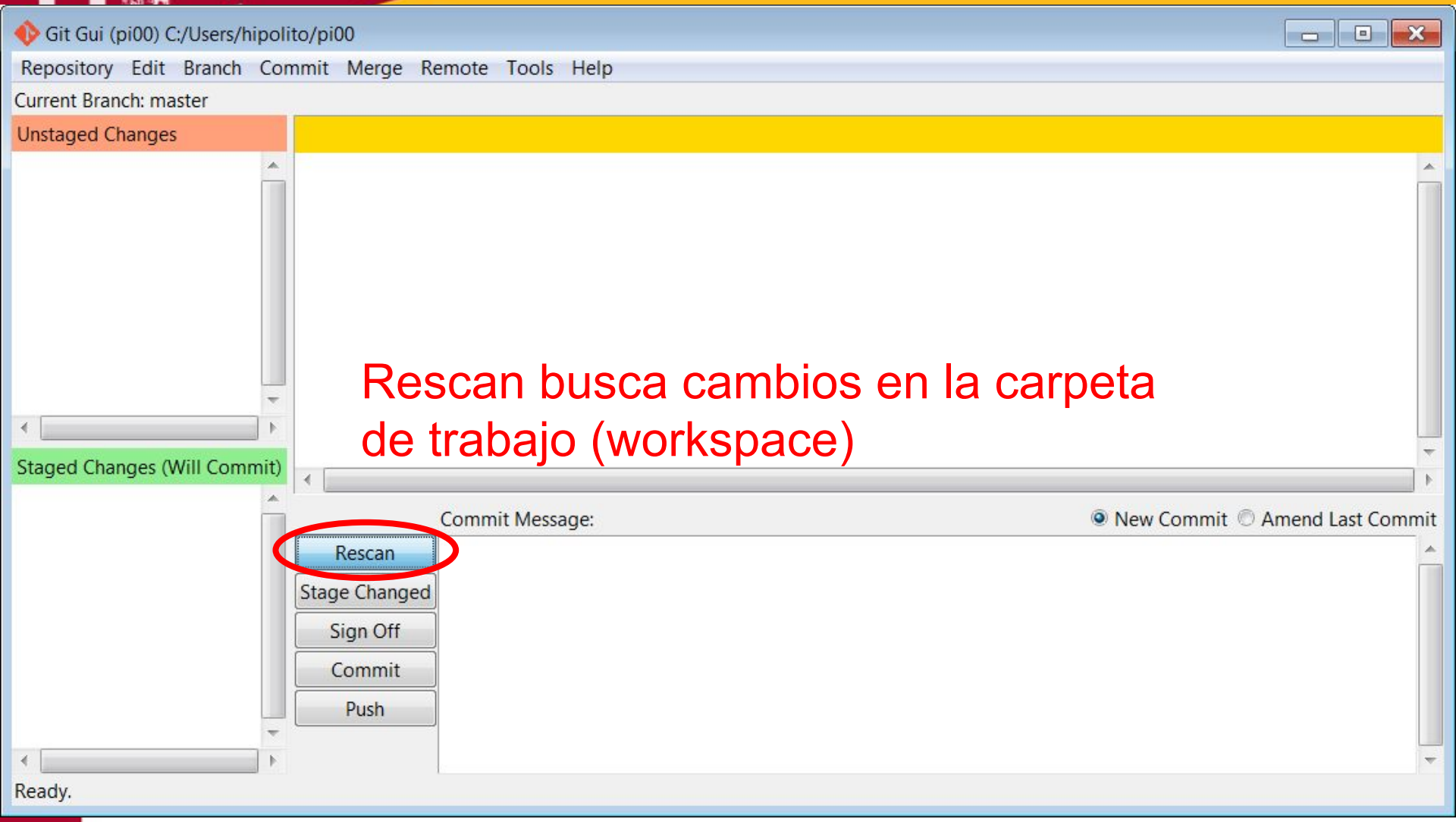
Workspace es una carpeta más de nuestro PC



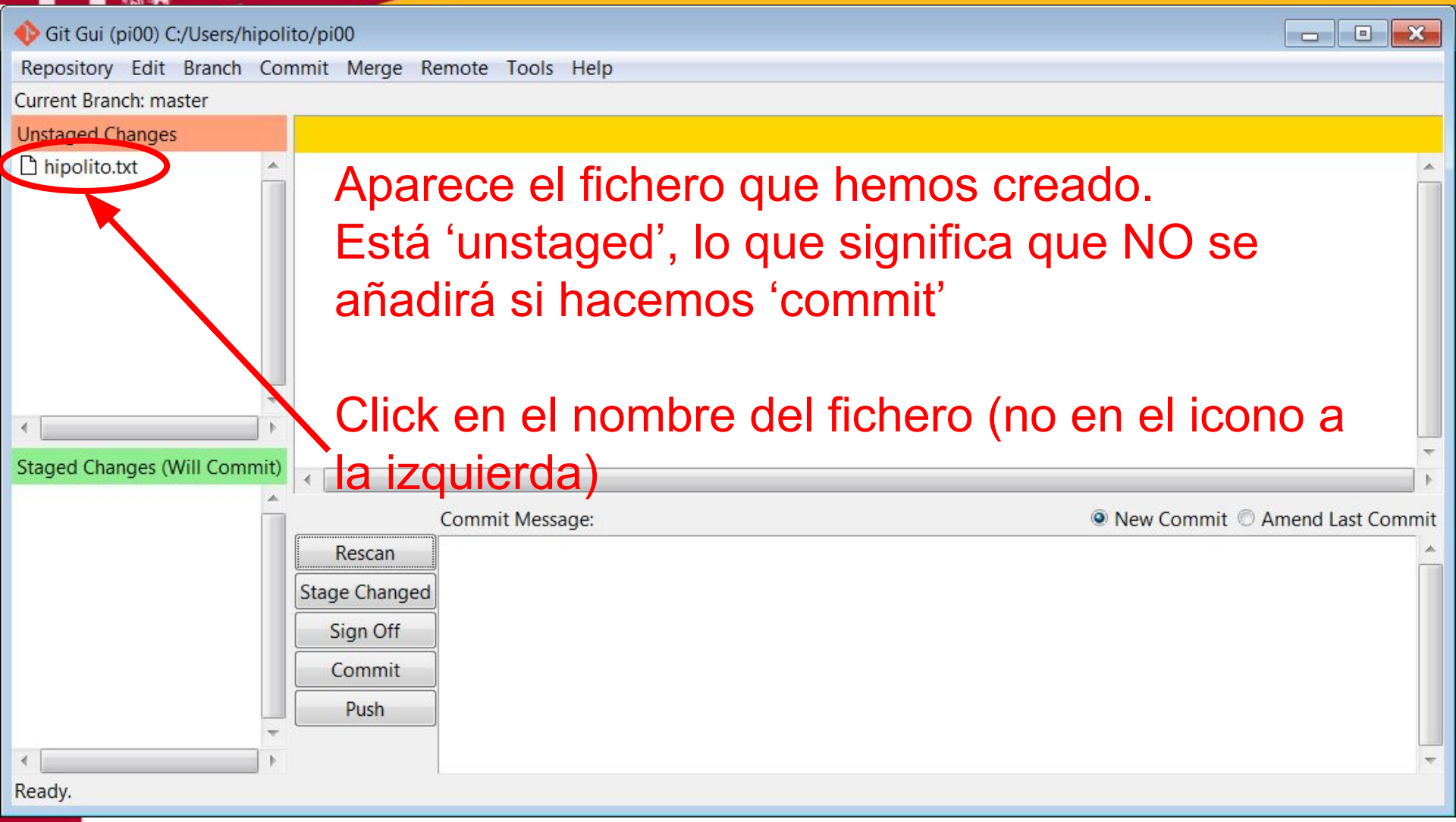
Creamos un fichero con nuestro nombre


hipolito.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
Contenido de 'hipolito.txt'

Añadir ficheros al stage



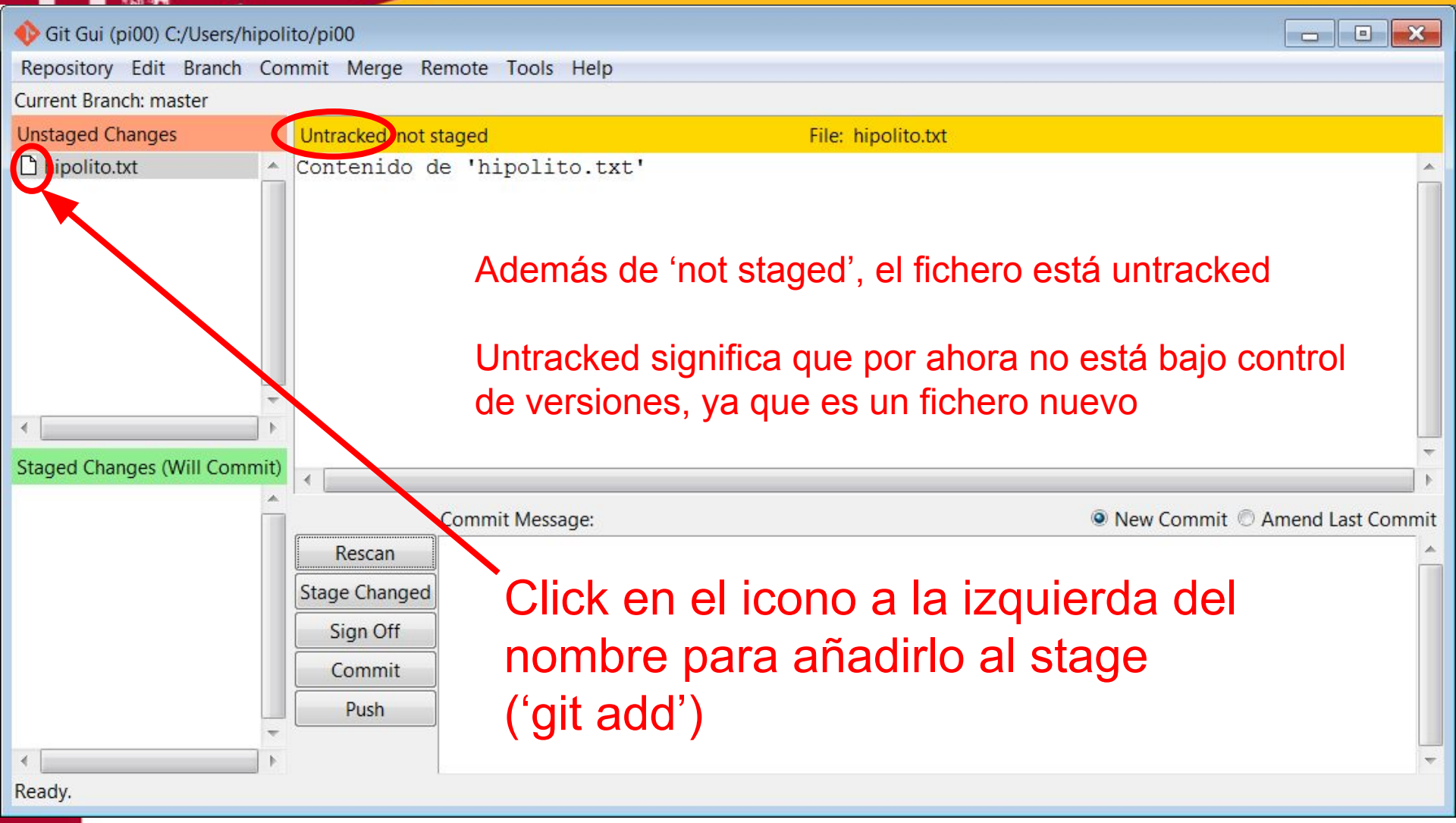
Añadir ficheros al stage



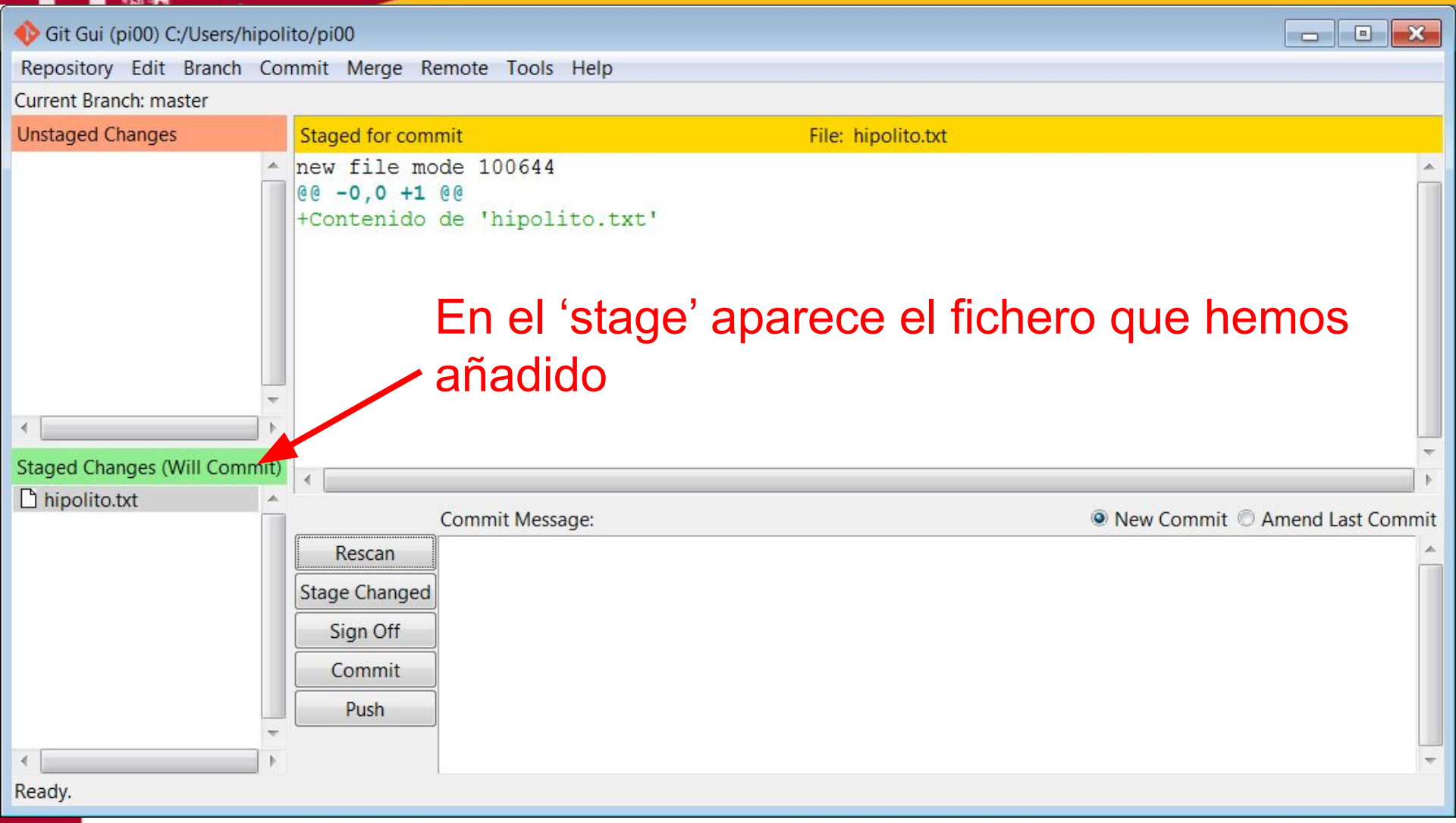
Aparece el fichero que hemos creado.
Está 'unstaged', lo que significa que NO se añadirá si hacemos 'commit'

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

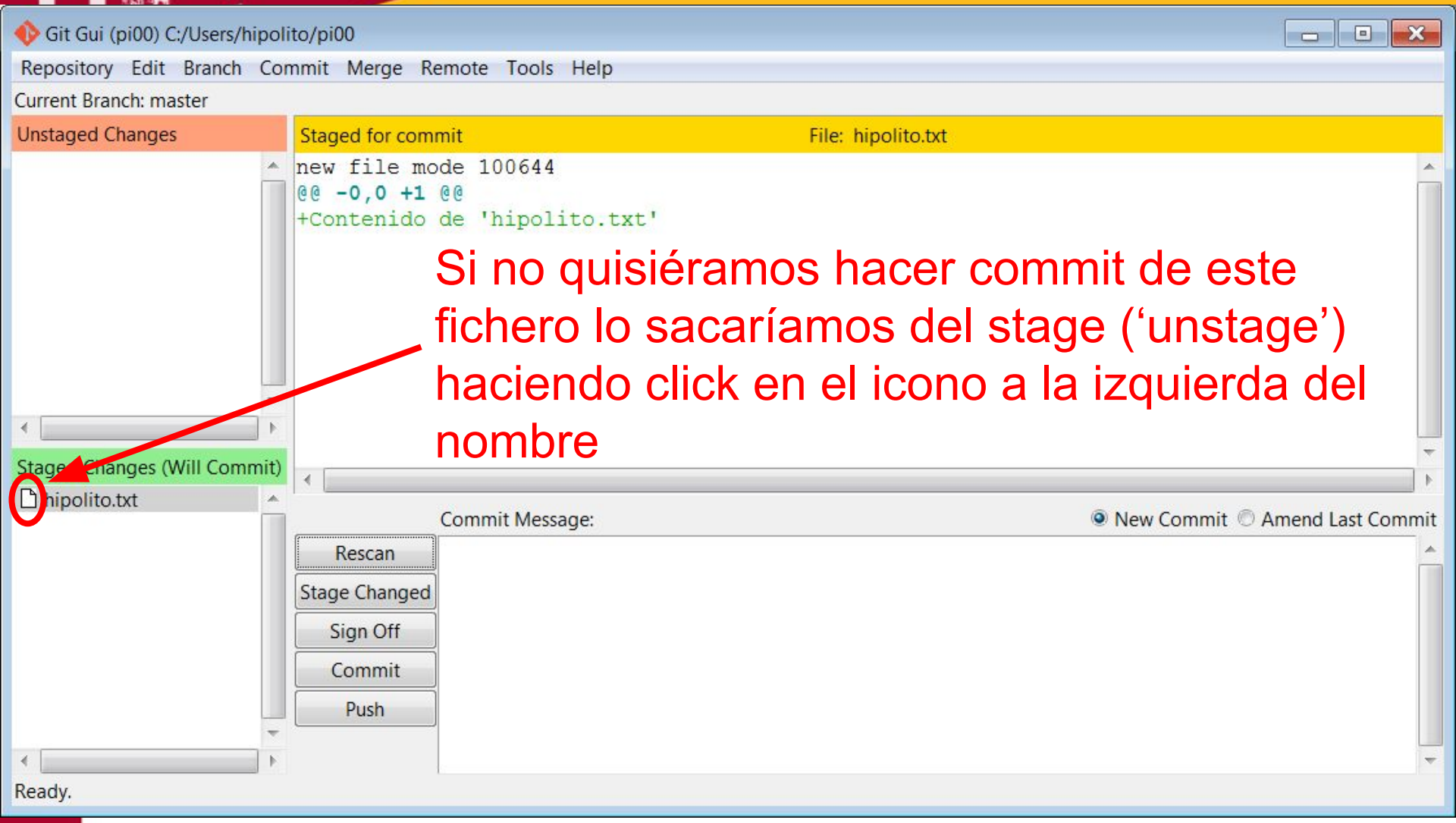
Añadir ficheros al stage



Añadir ficheros al stage



Añadir ficheros al stage

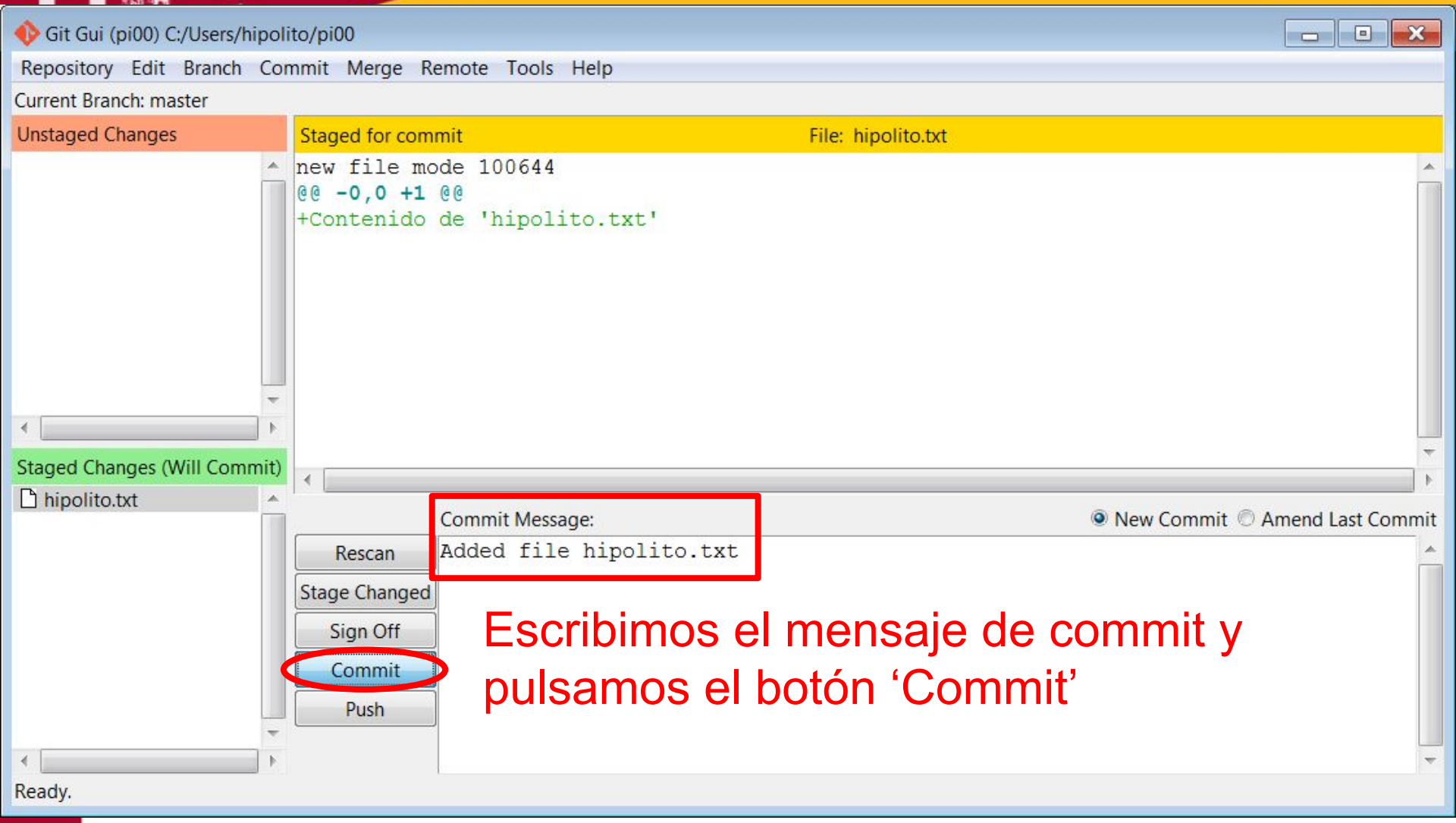


Si no quisiéramos hacer commit de este fichero lo sacaríamos del stage ('unstage') haciendo click en el icono a la izquierda del nombre

Contenido

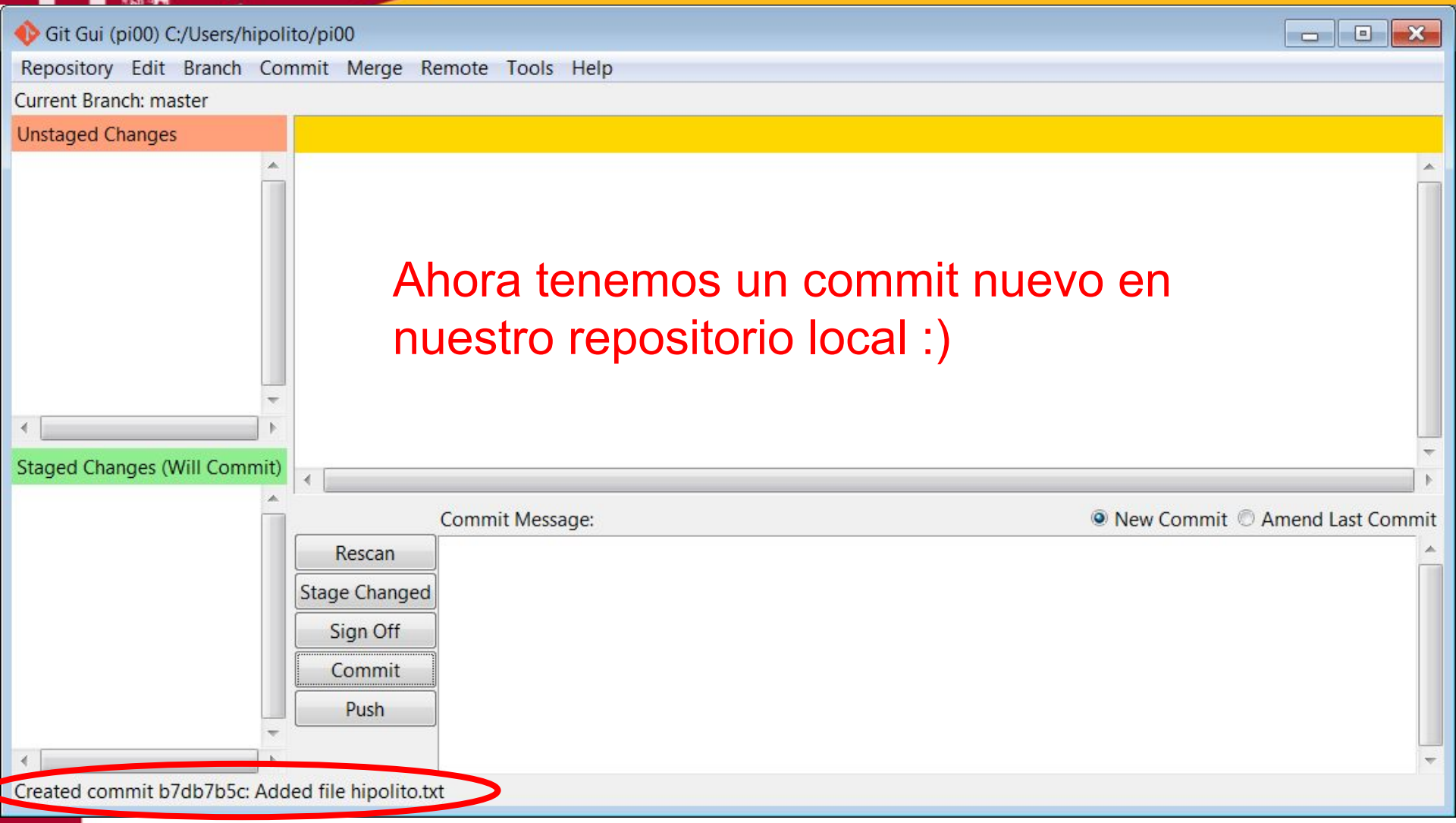
- Áreas de git
- Clonar el repositorio
- Configurar git
- Añadir ficheros al stage
- Creando commits
- Push
- Pull = Fetch from origin + Merge local
- Resolución de conflictos
- Revertir errores usando checkout

Creando commits



Escribimos el mensaje de commit y pulsamos el botón ‘Commit’

Creando commits



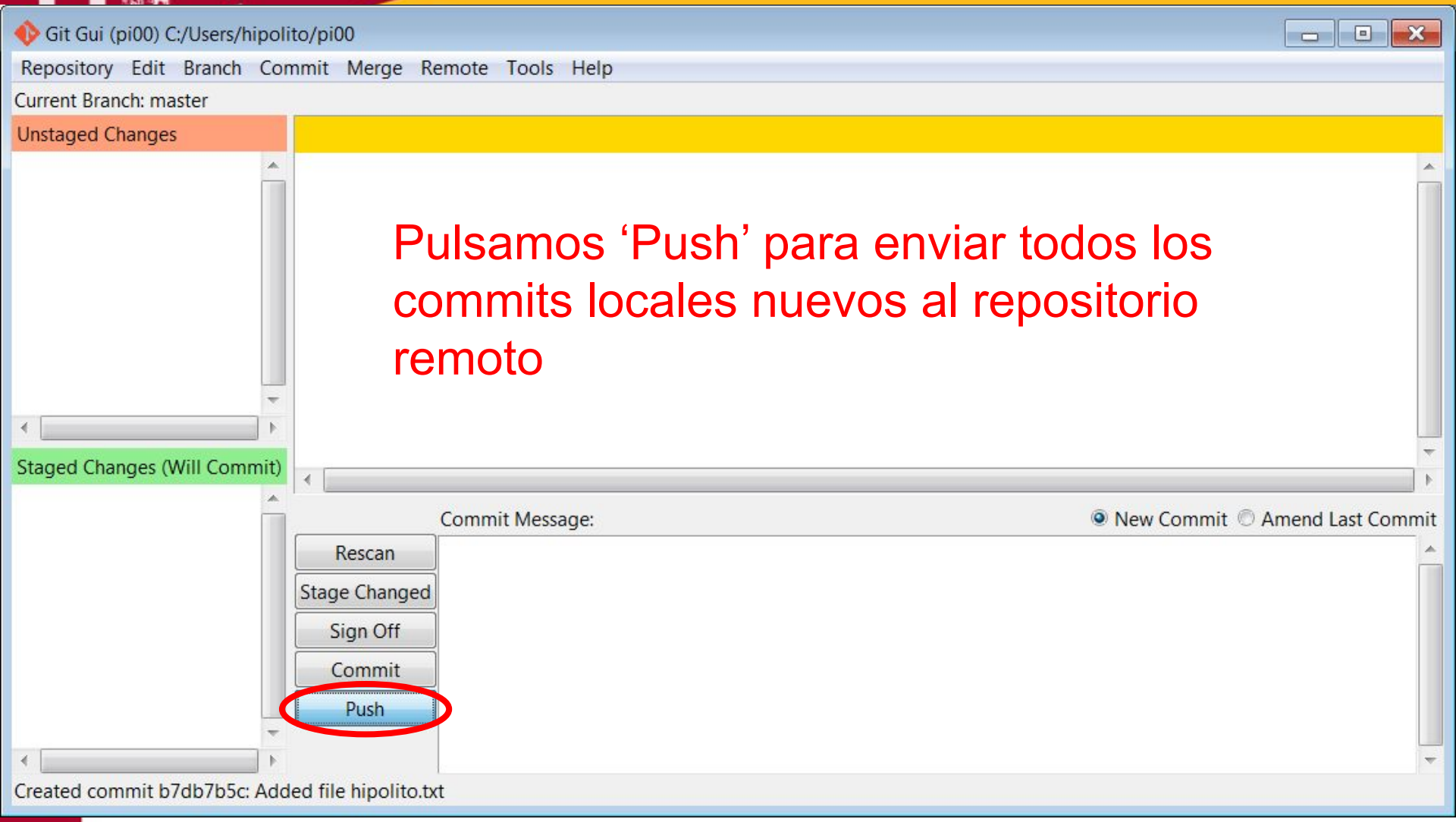
Formato correcto de un mensaje de commit

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

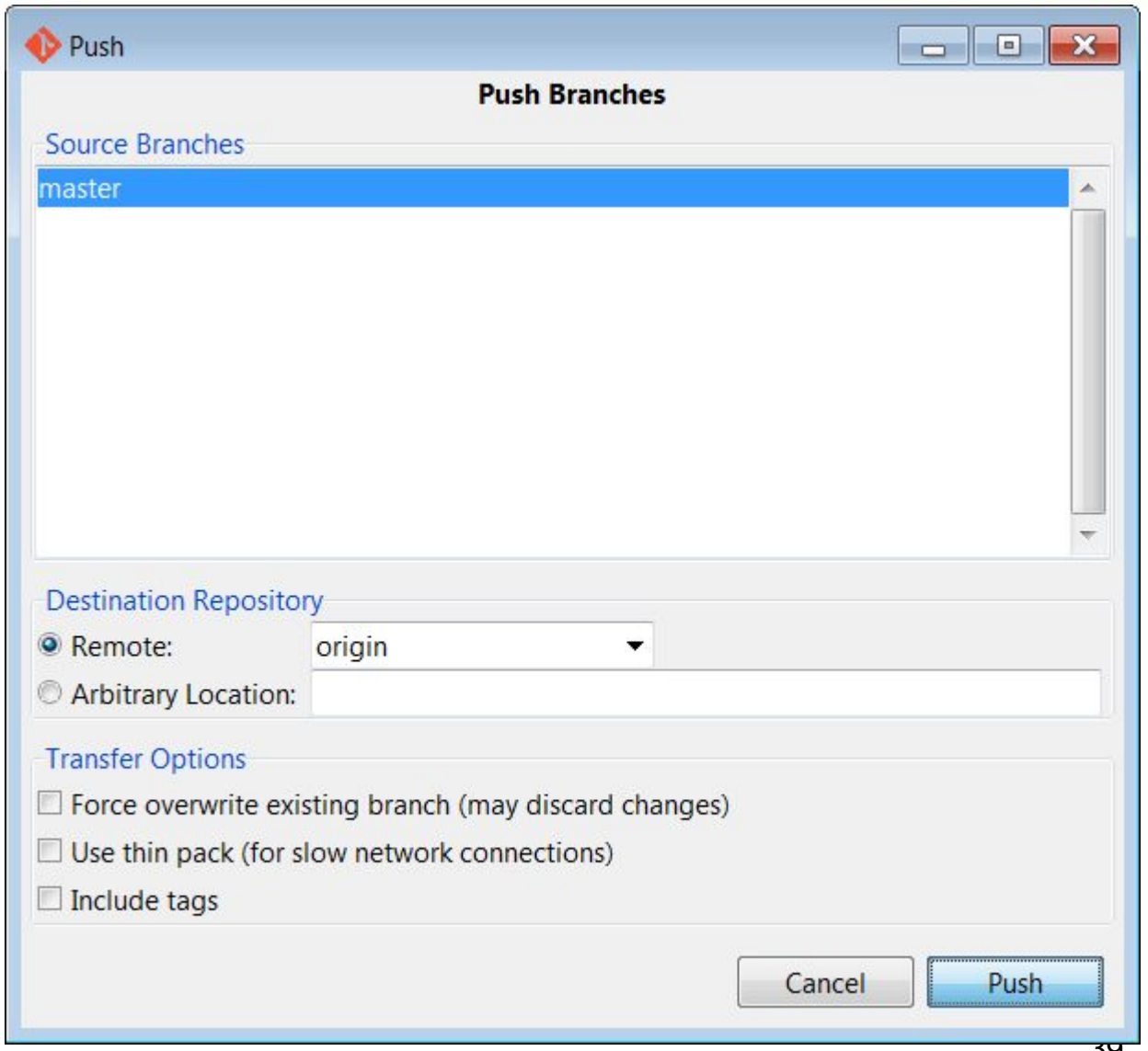
Contenido

- 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

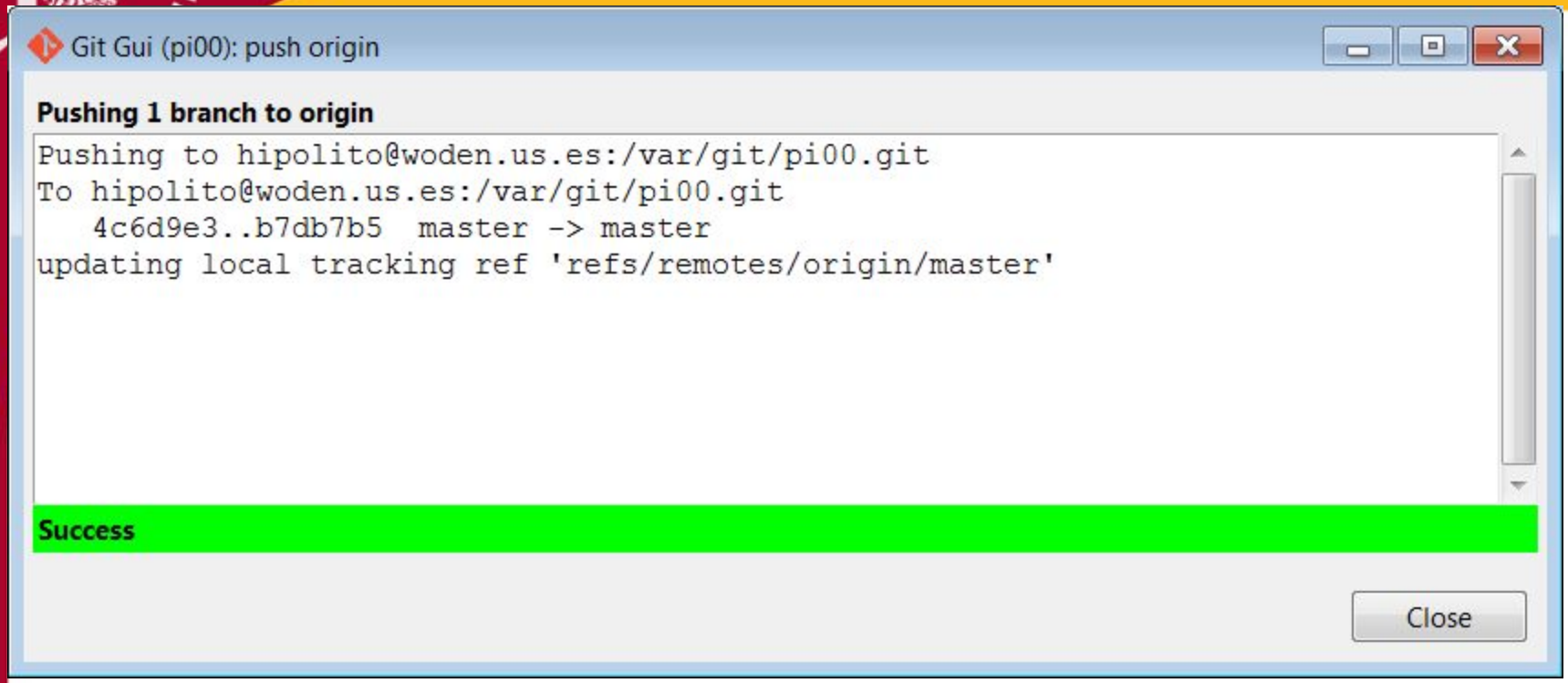

Git Gui (pi00) C:/Users/hipolito/pi00
Repository Edit Branch Commit Merge Remote Tools Help
Current Branch: master
Unstaged Changes
Staged Changes (Will Commit)
Pulsamos 'Push' para enviar todos los commits locales nuevos al repositorio remoto
Commit Message:
New Commit
Amend Last Commit
Rescan
Stage Changed
Sign Off
Commit
Push
Created commit b7db7b5c: Added file hipolito.txt

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)

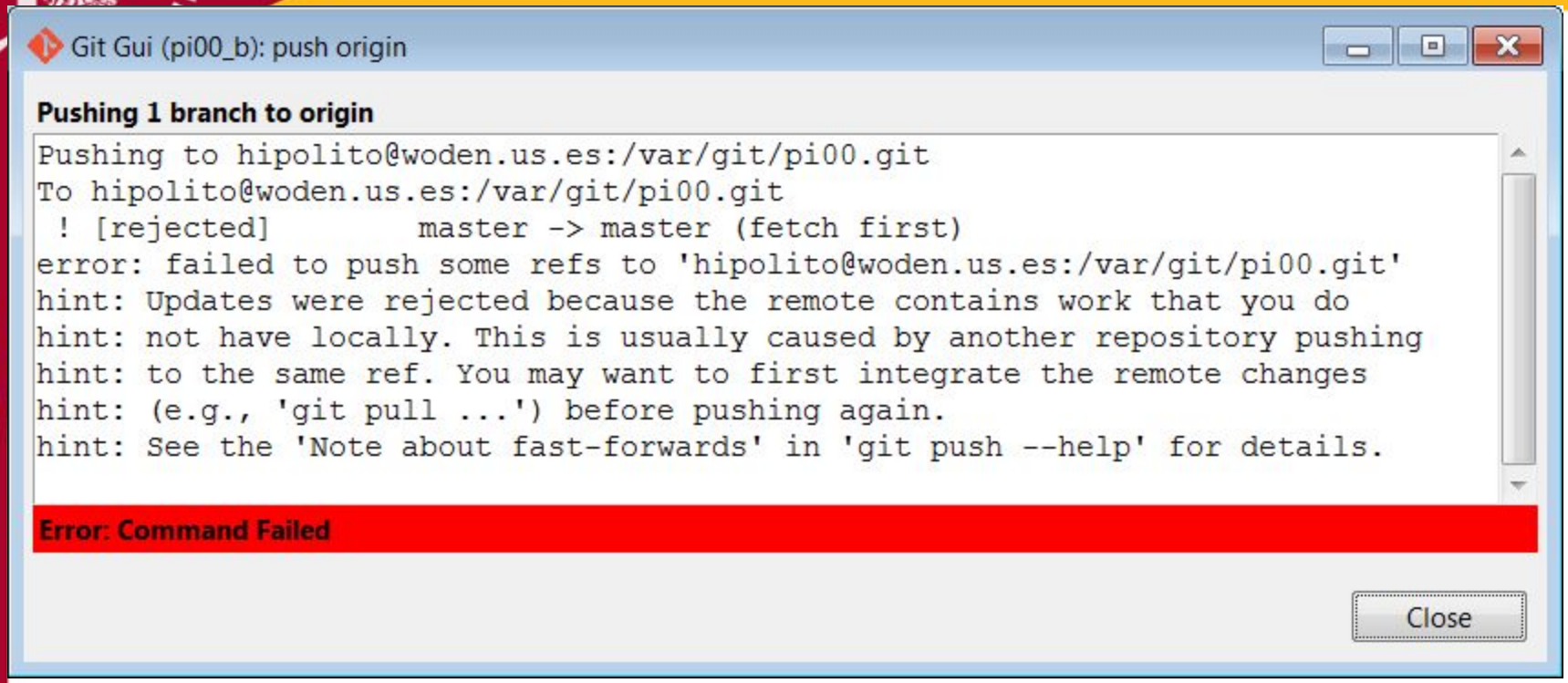


Push



Normalmente todo va bien...

Push

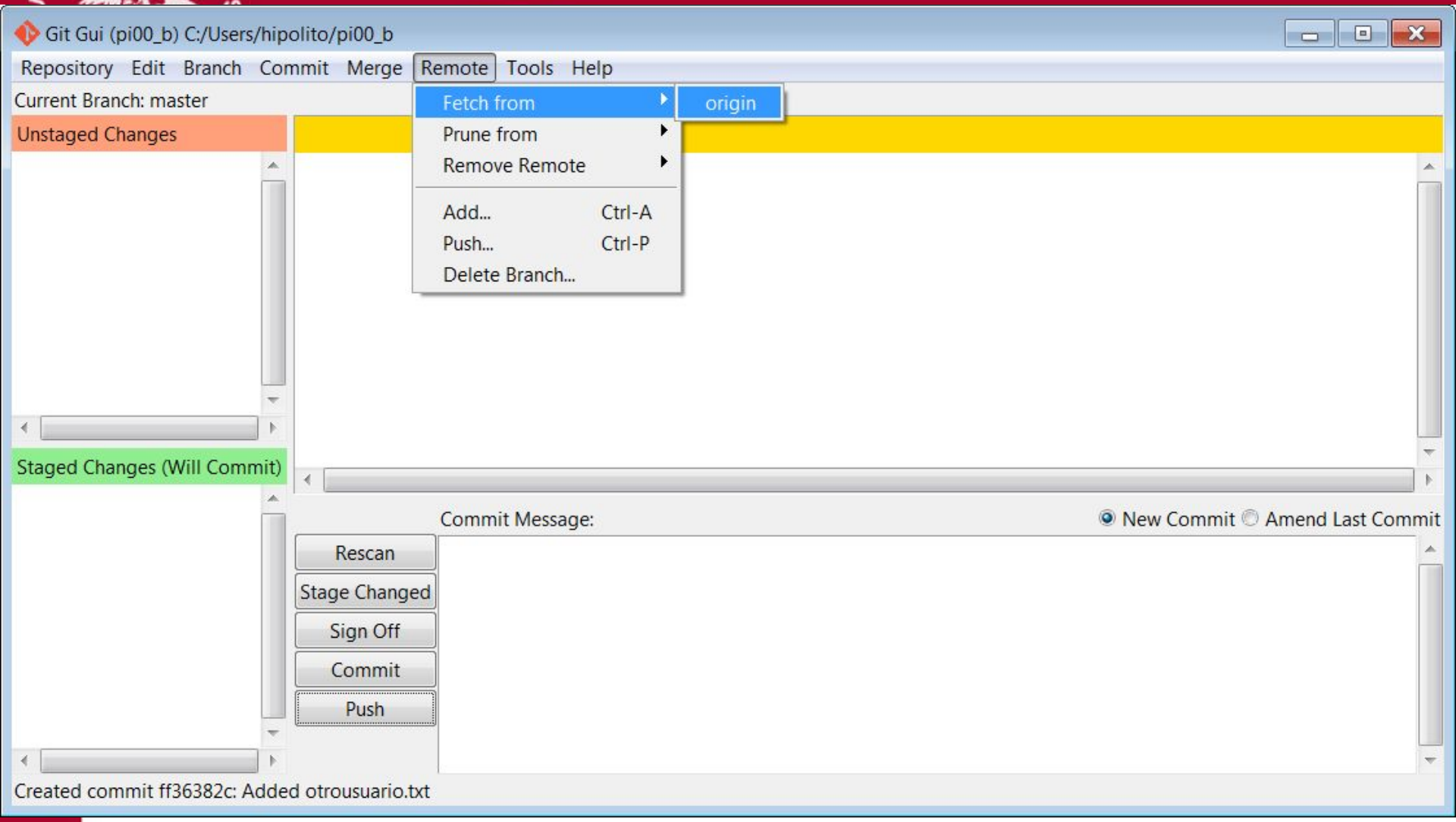


Si nos da este error, es porque otro compañero ha hecho push antes que nosotros: debemos actualizar nuestro repo local (fetch + merge) antes de hacer push

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

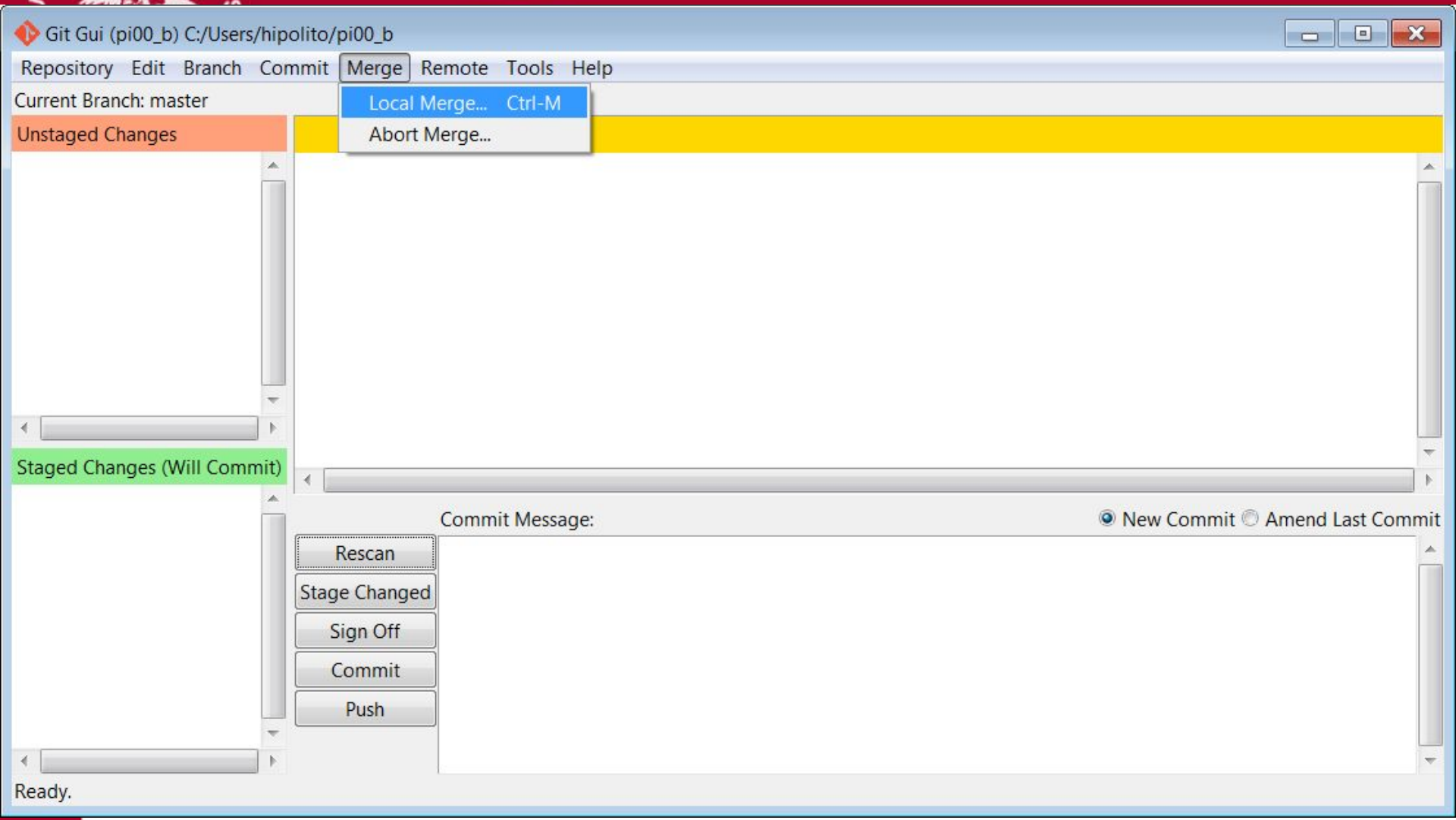


Fetch from origin descarga los cambios nuevos que haya en el repo remoto

Pull: fetch + merge

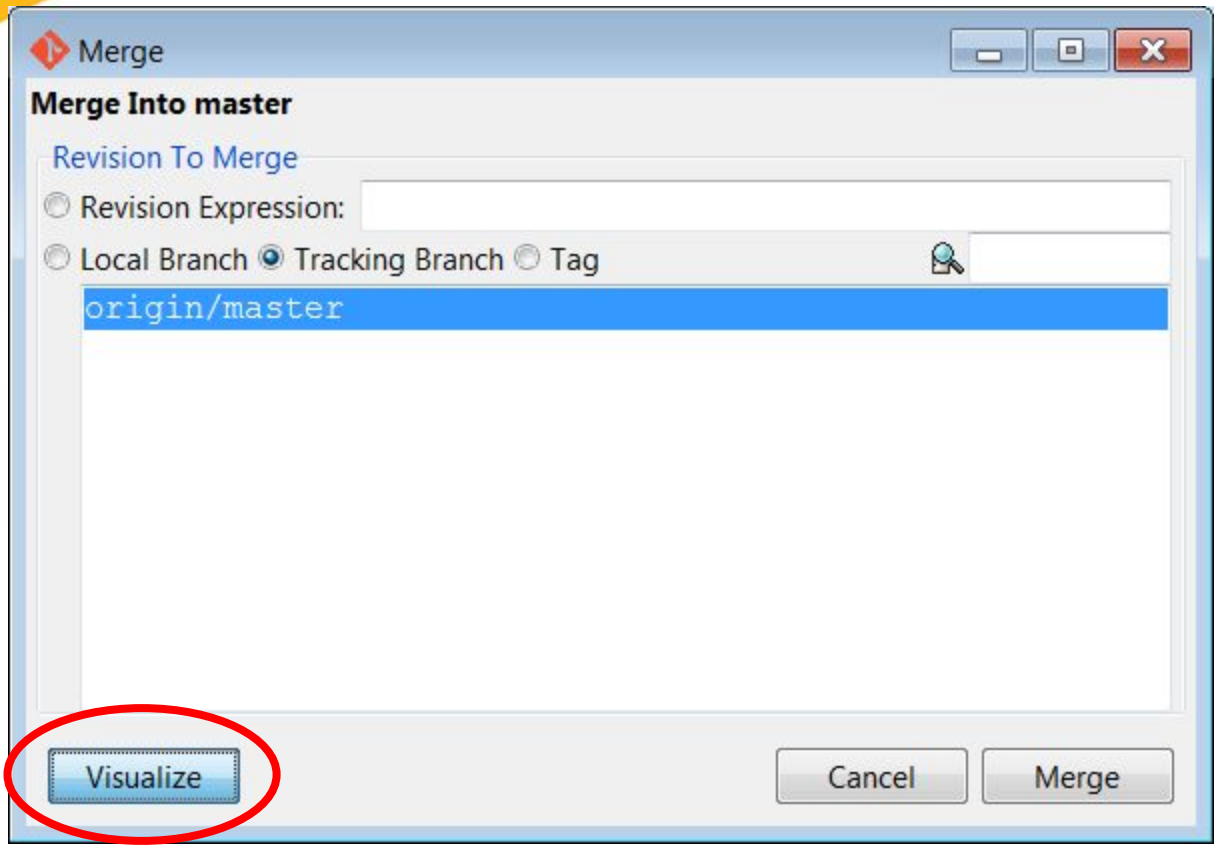
(Nota: si al hacer fetch este cuadro de diálogo no contiene texto, es que no hay cambios nuevos, por lo que no será necesario un merge)

Pull: fetch + merge



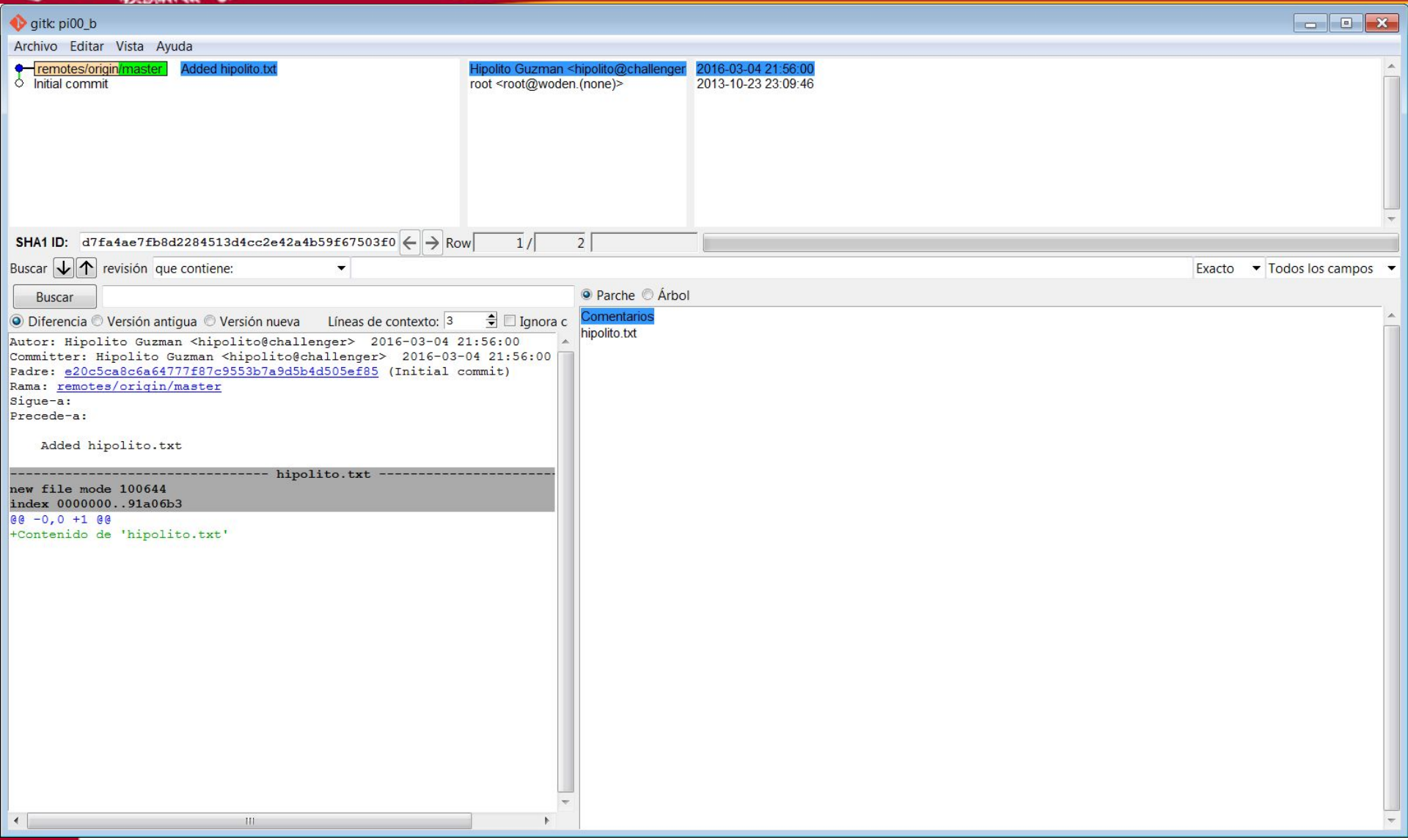
Local merge fusiona los cambios nuevos en el repo local y el workspace

Pull: fetch + merge

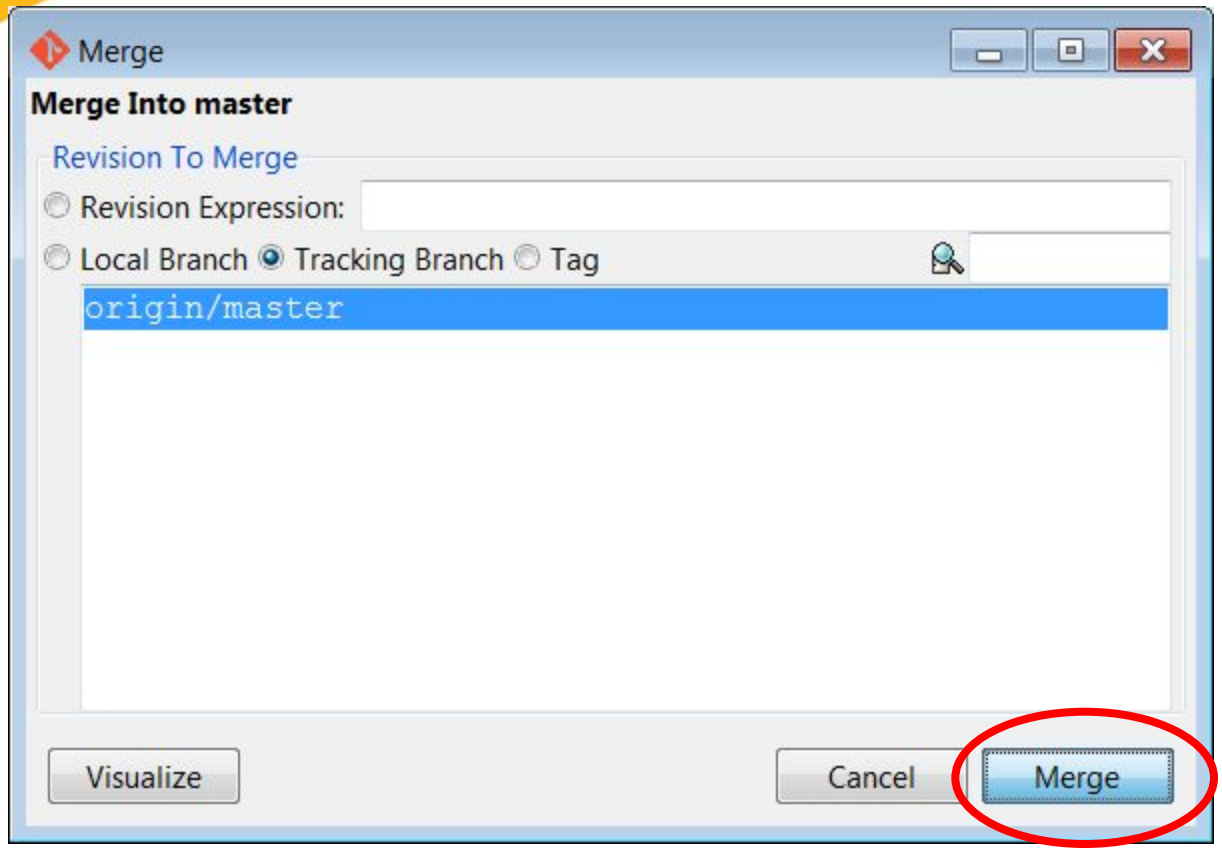


Podemos ver los cambios antes de fusionar

Pull: fetch + merge



Pull: fetch + merge

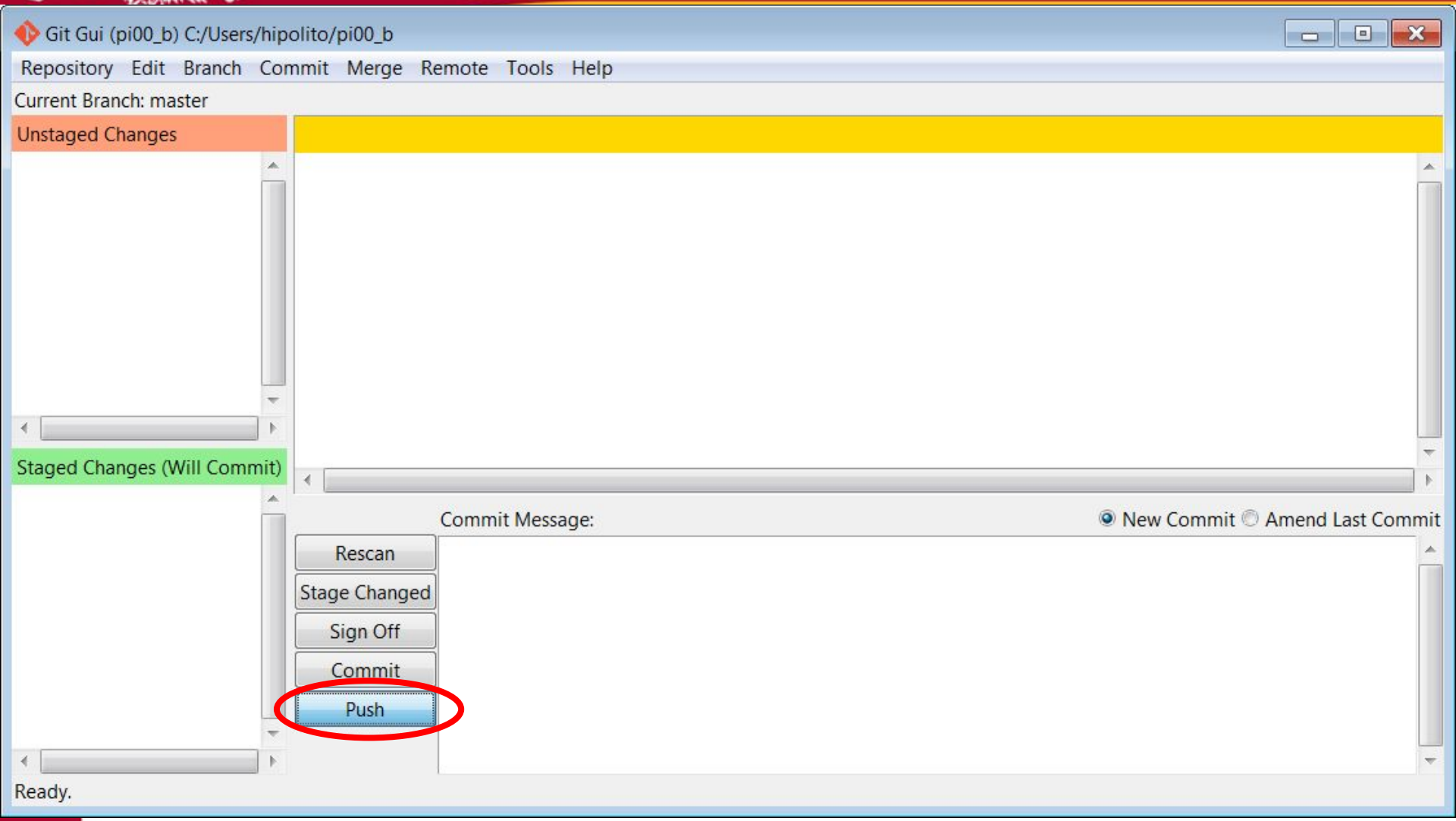


Pulsamos ‘Merge’

Pull: fetch + merge

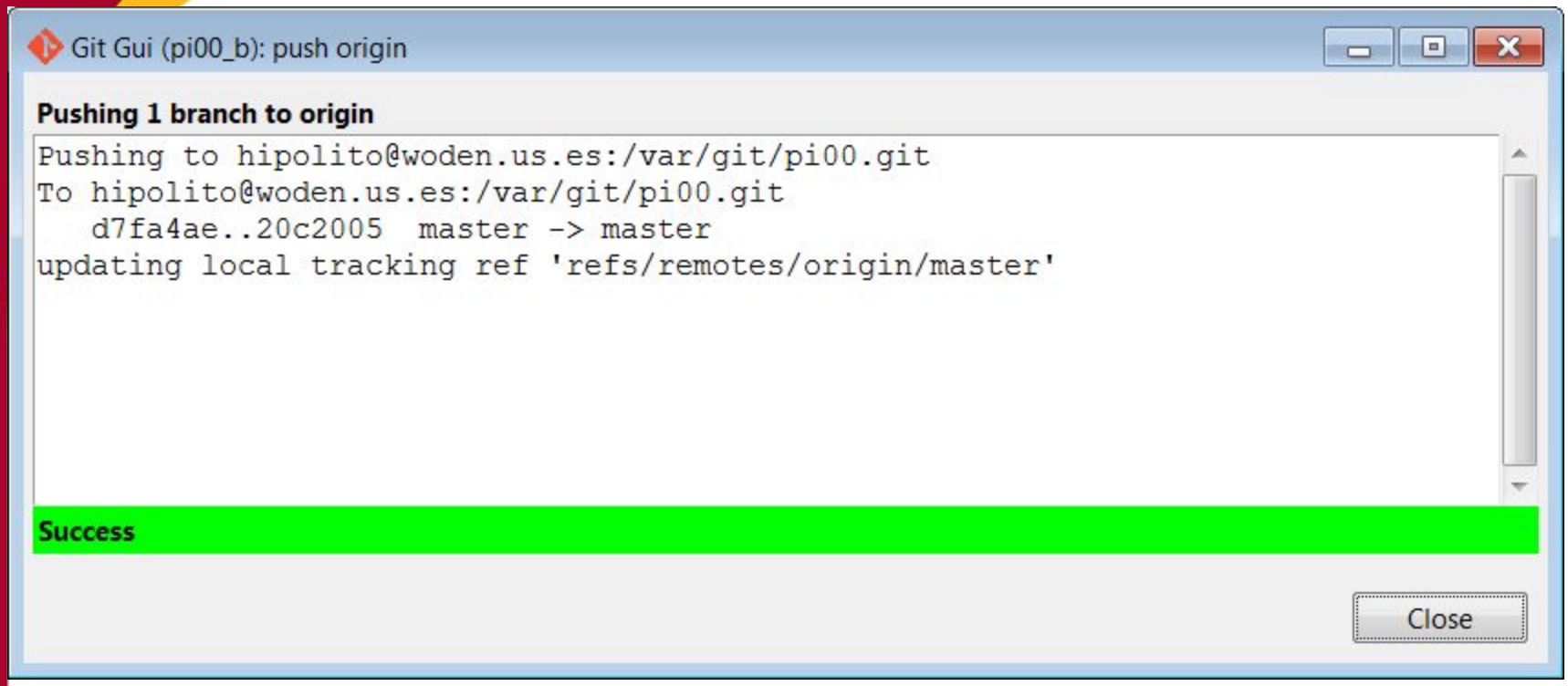
Merge completado con éxito

Push (2)



Tras el merge, ya podemos ‘pushear’ nuestros cambios

Push (2)



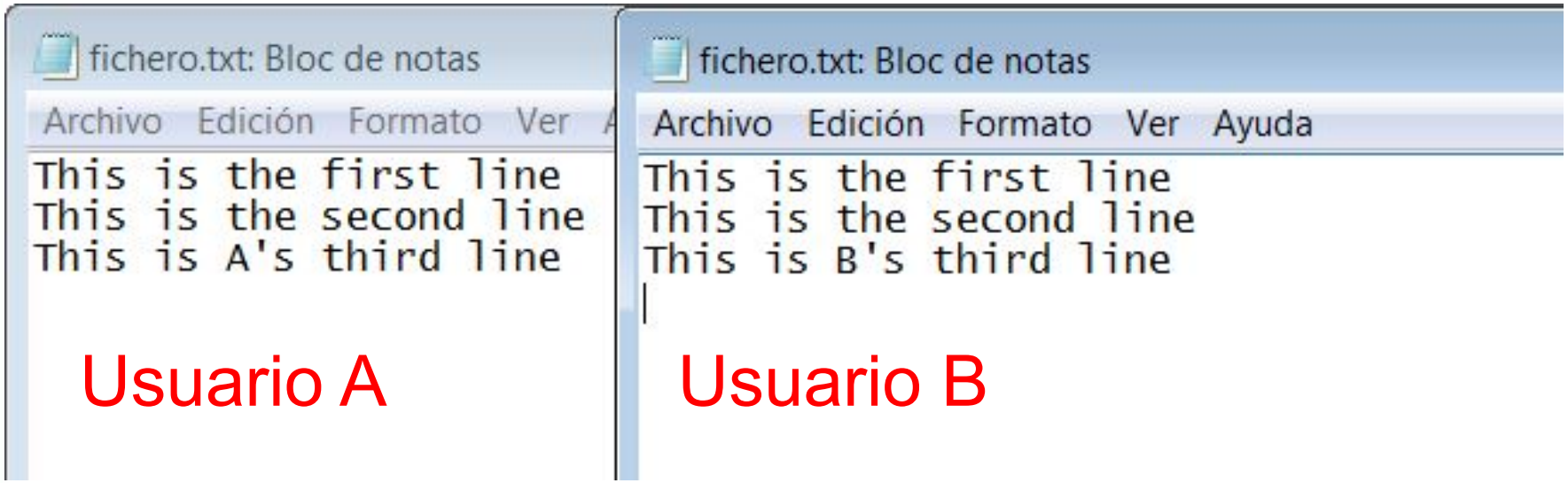
Y esta vez, el push termina correctamente

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

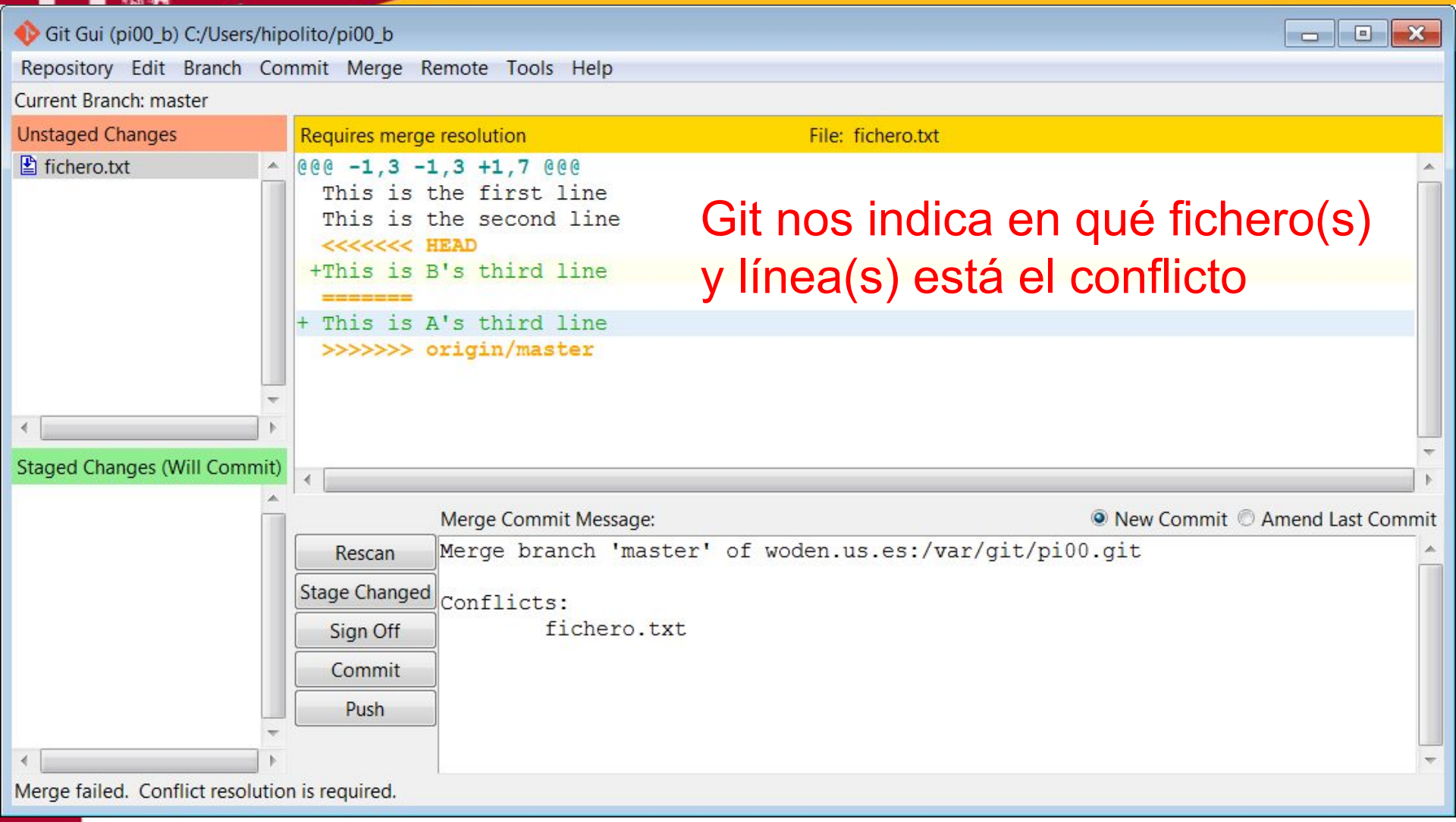
Resolución de Conflictos

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

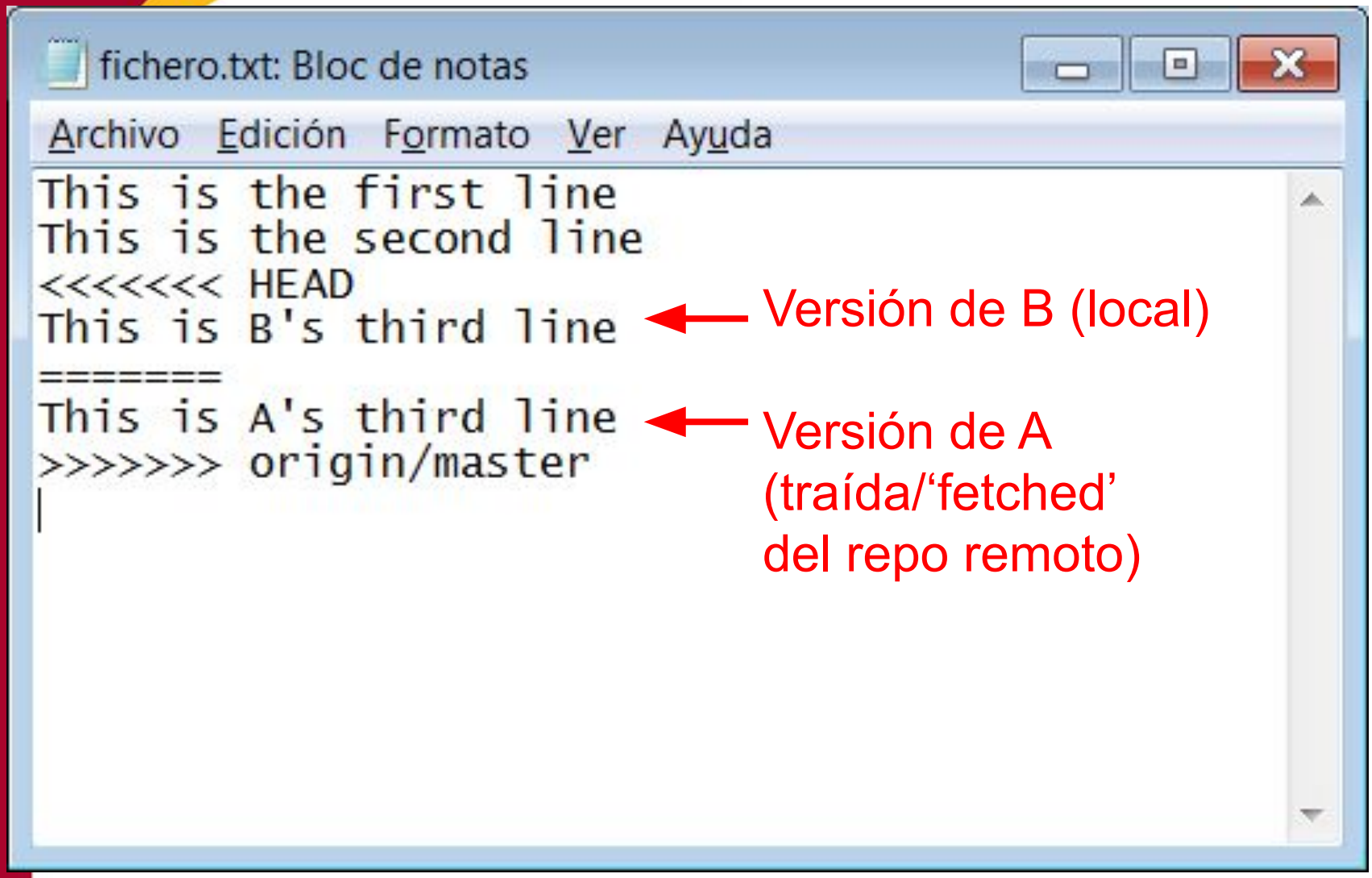


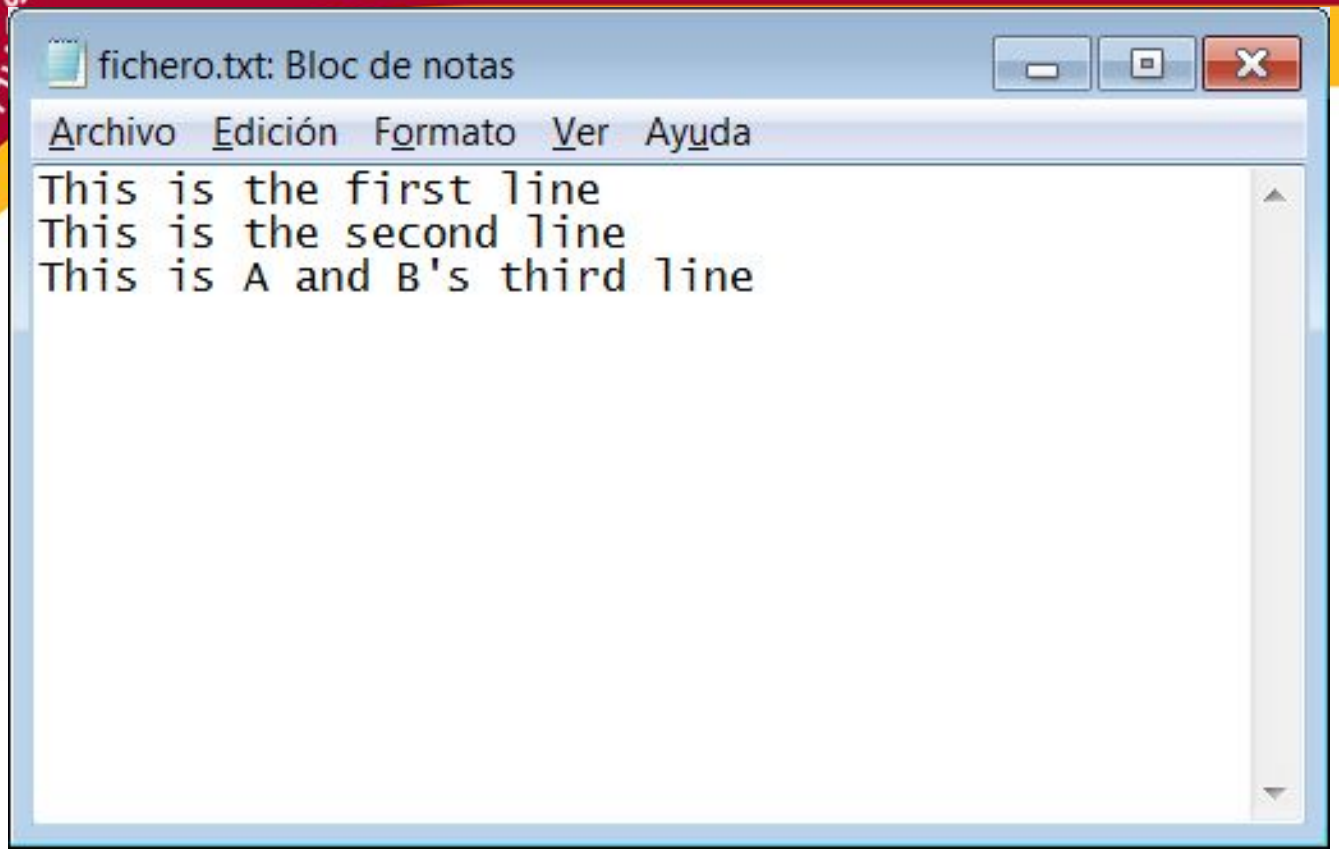
A ‘pushea’ primero y no ve nada raro

B tiene que hacer fetch + merge, pero al hacer merge:



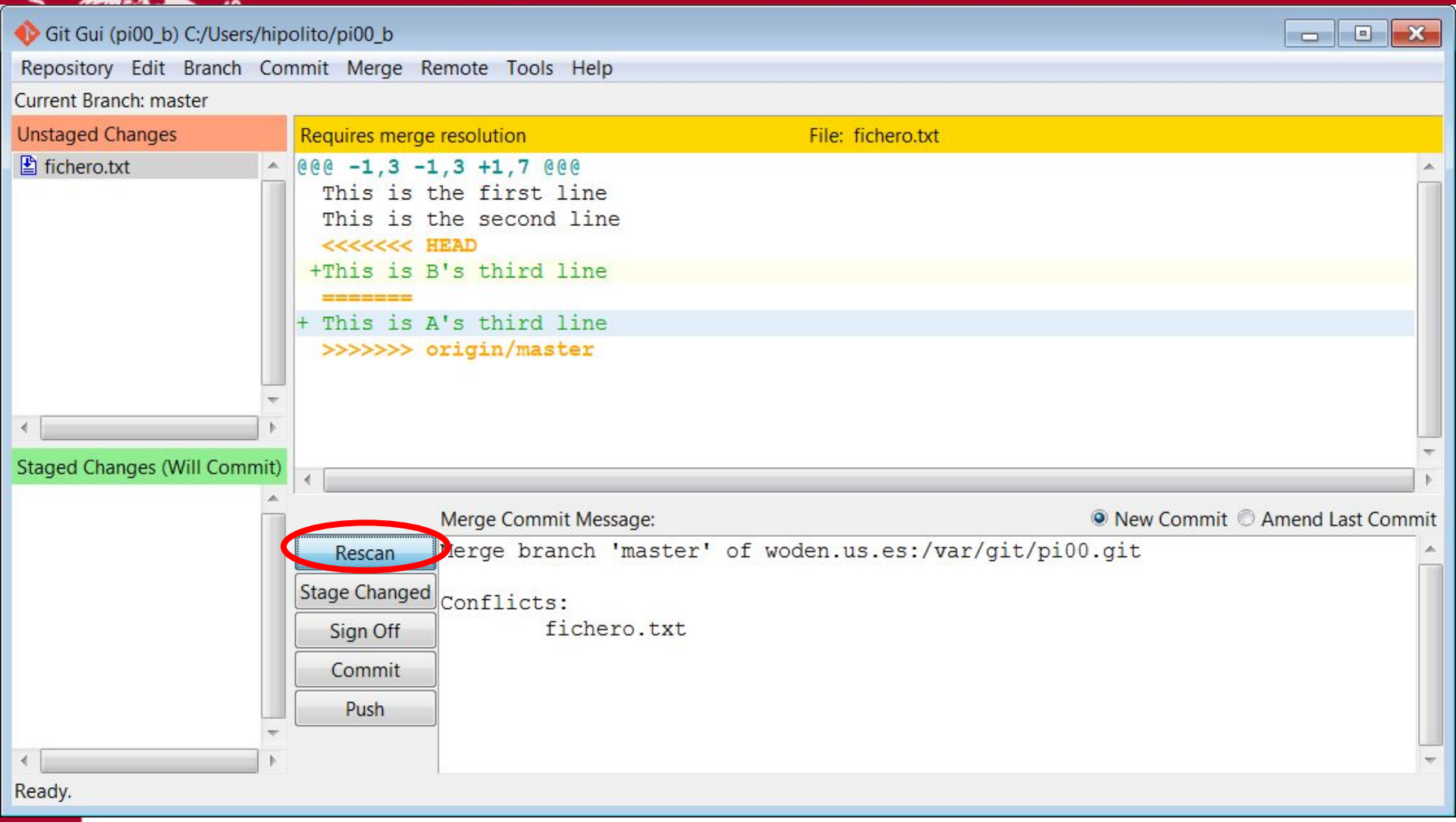
Git nos indica en qué fichero(s) y línea(s) está el conflicto


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
Versión de B (local)
Versión de A
(traída/'fetched'
del repo remoto)

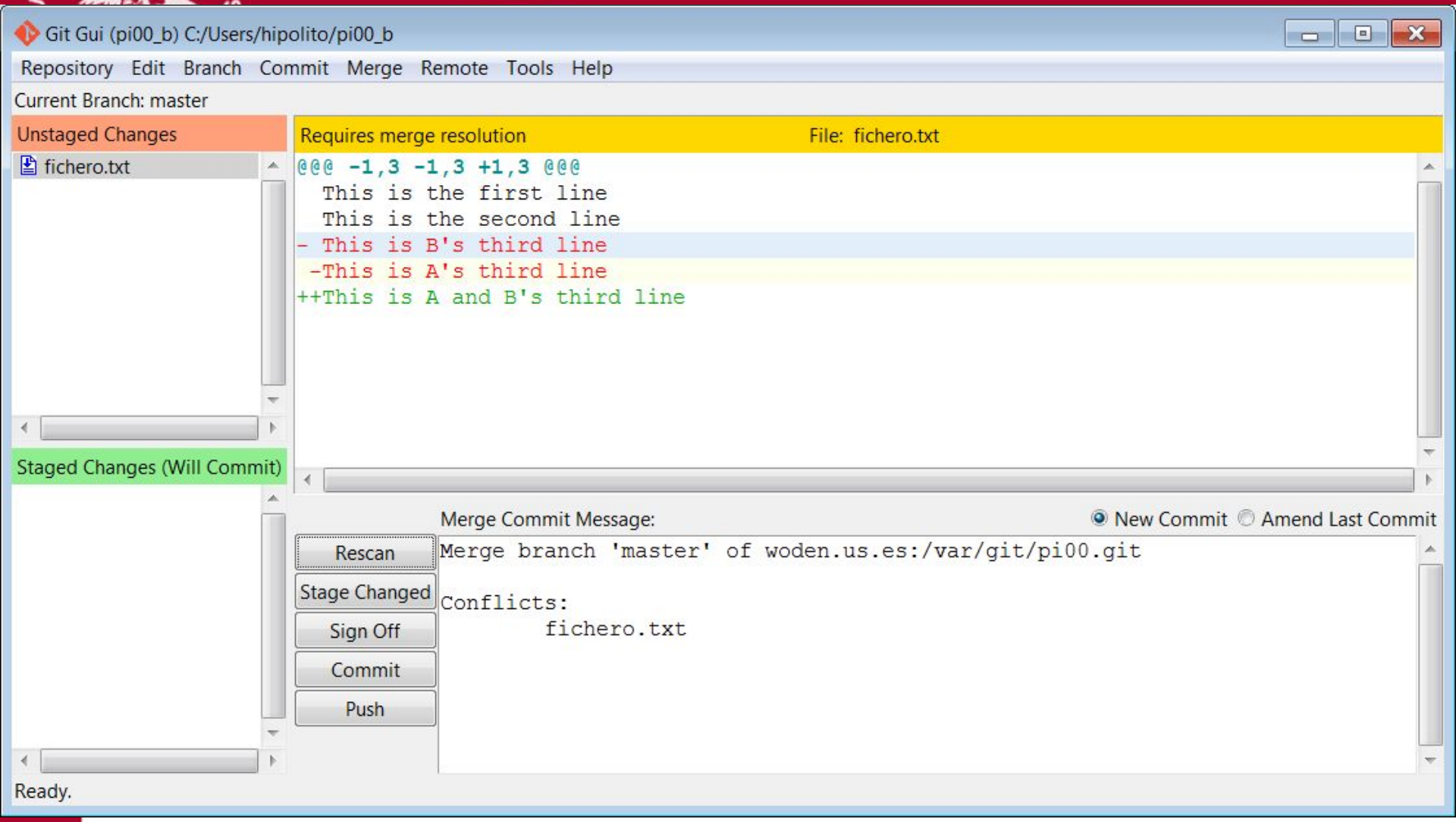


Solucionamos el conflicto:

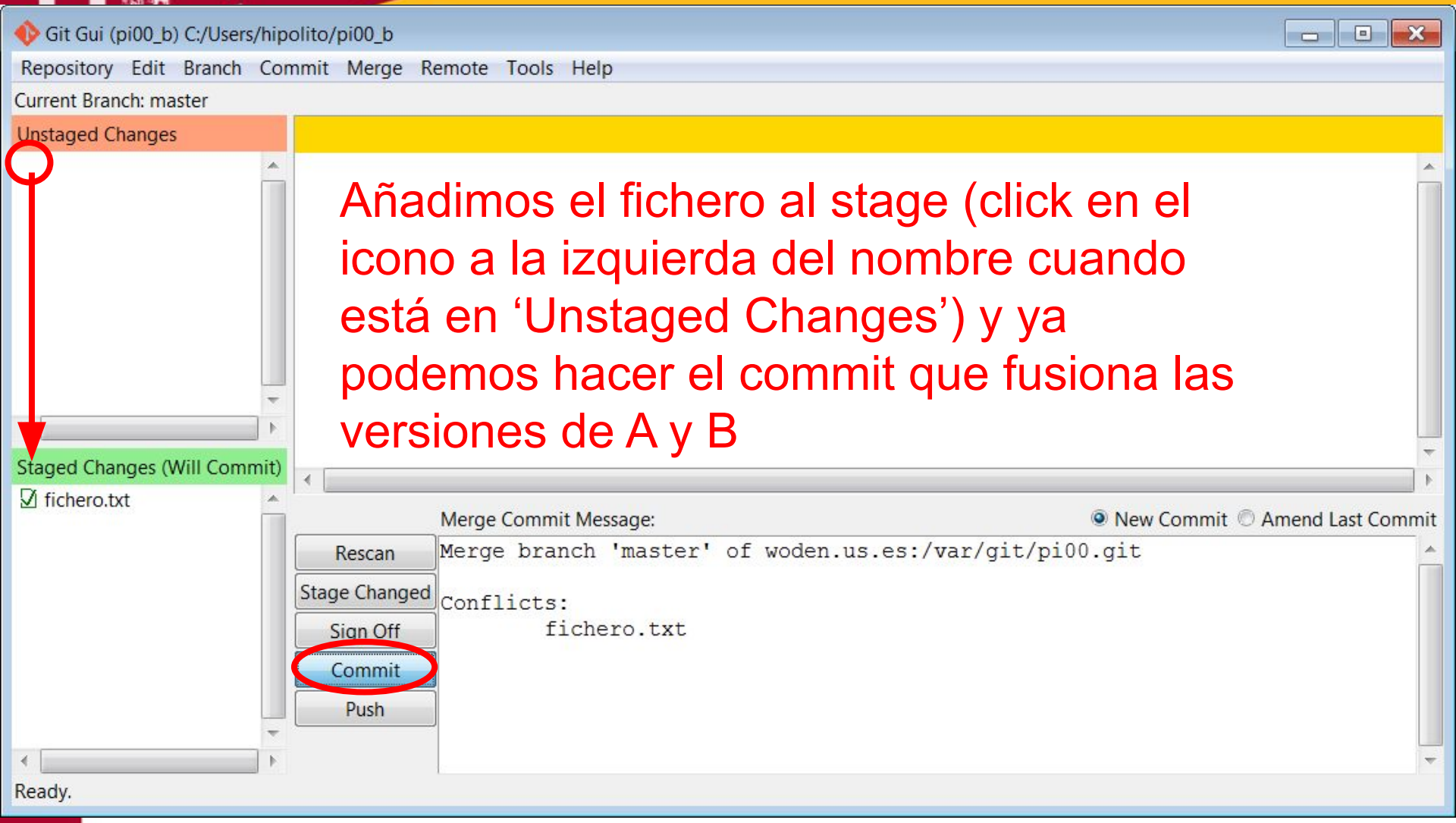
- Decidiendo qué versión es la correcta
- Eliminando los caracteres de control (<<, >>, ==)



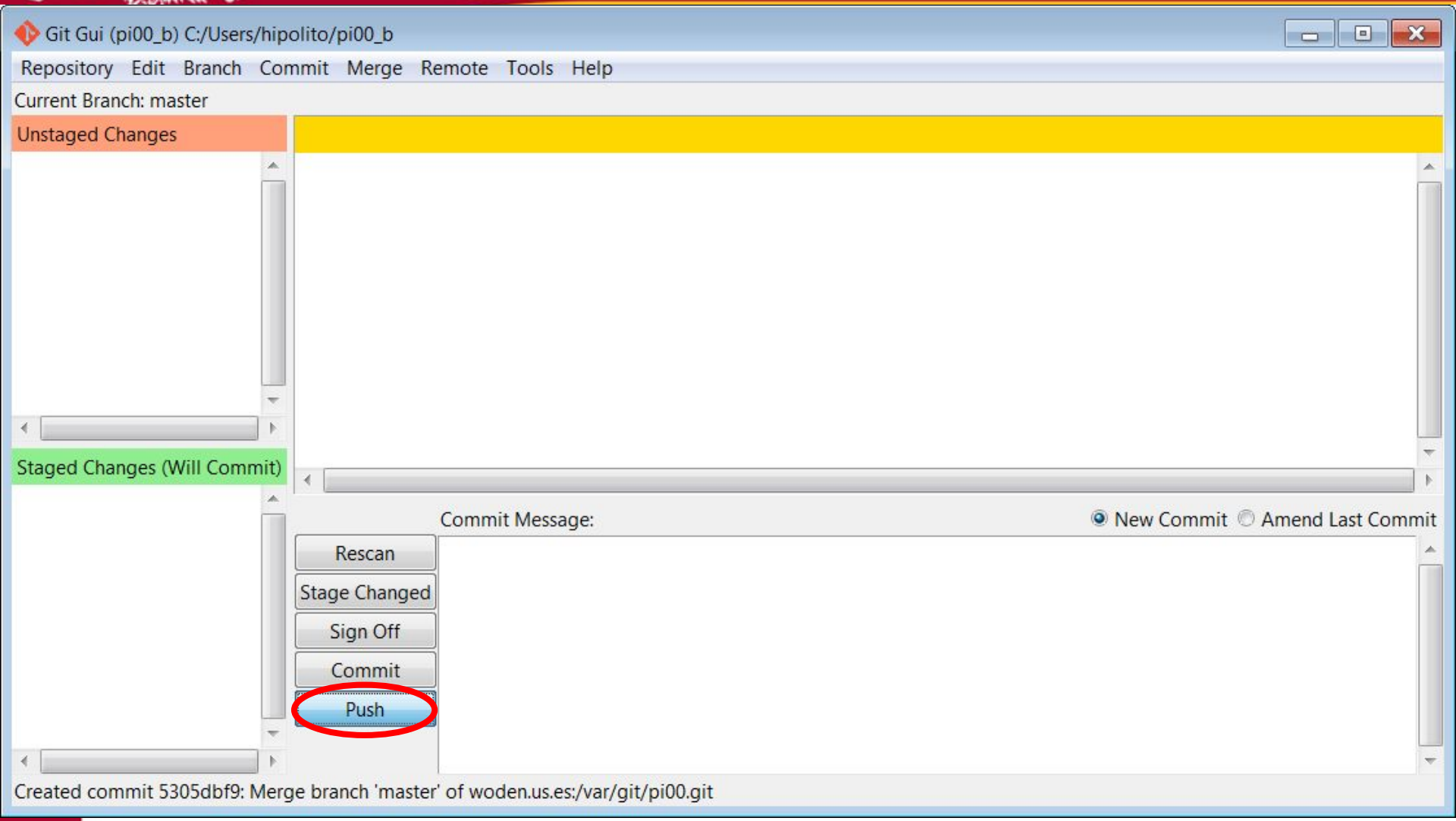
Pulsamos ‘Rescan’ para buscar cambios en el workspace



Vemos la versión corregida del fichero



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

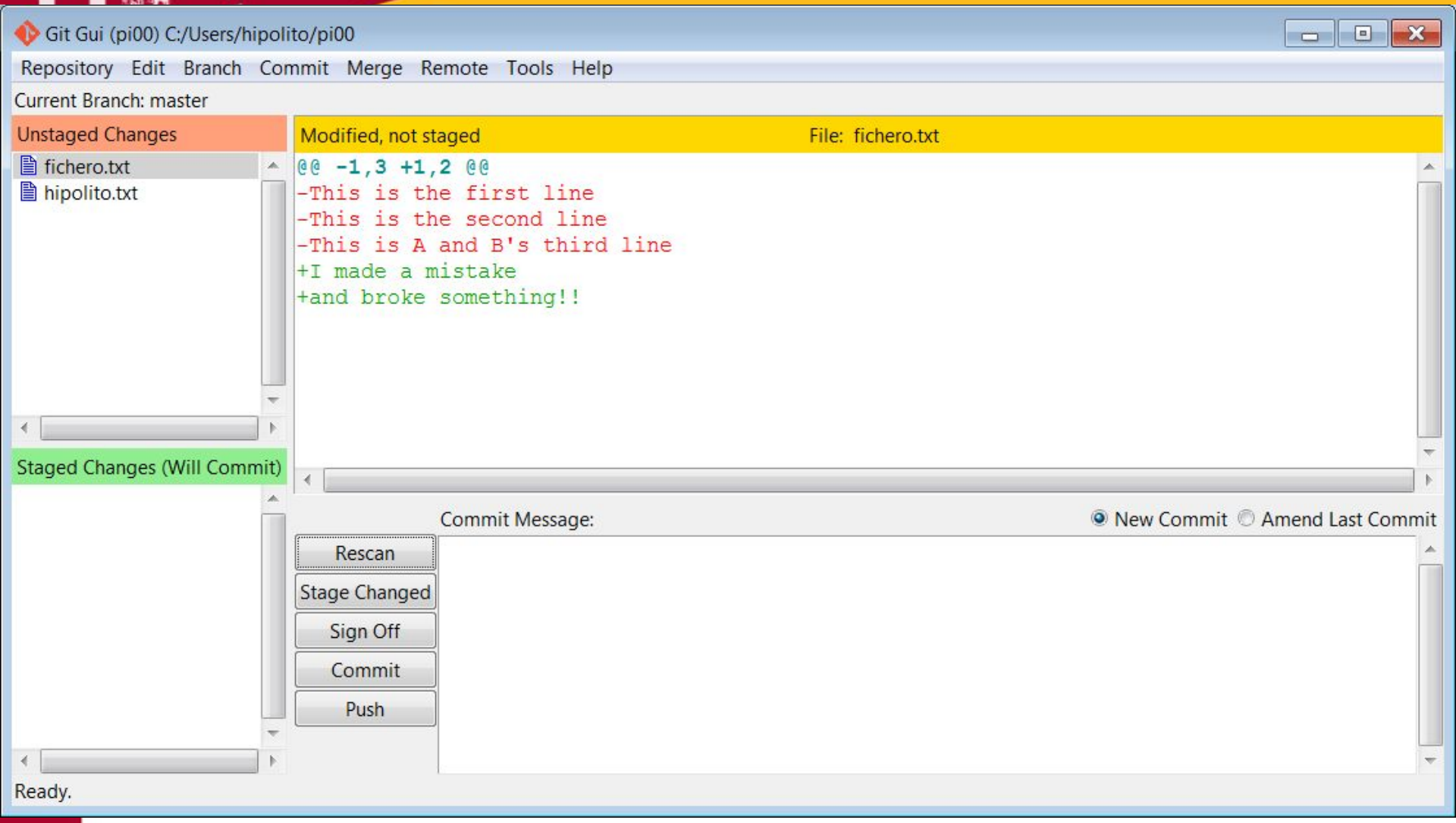


Y ya podemos empujar este nuevo commit al repositorio remoto

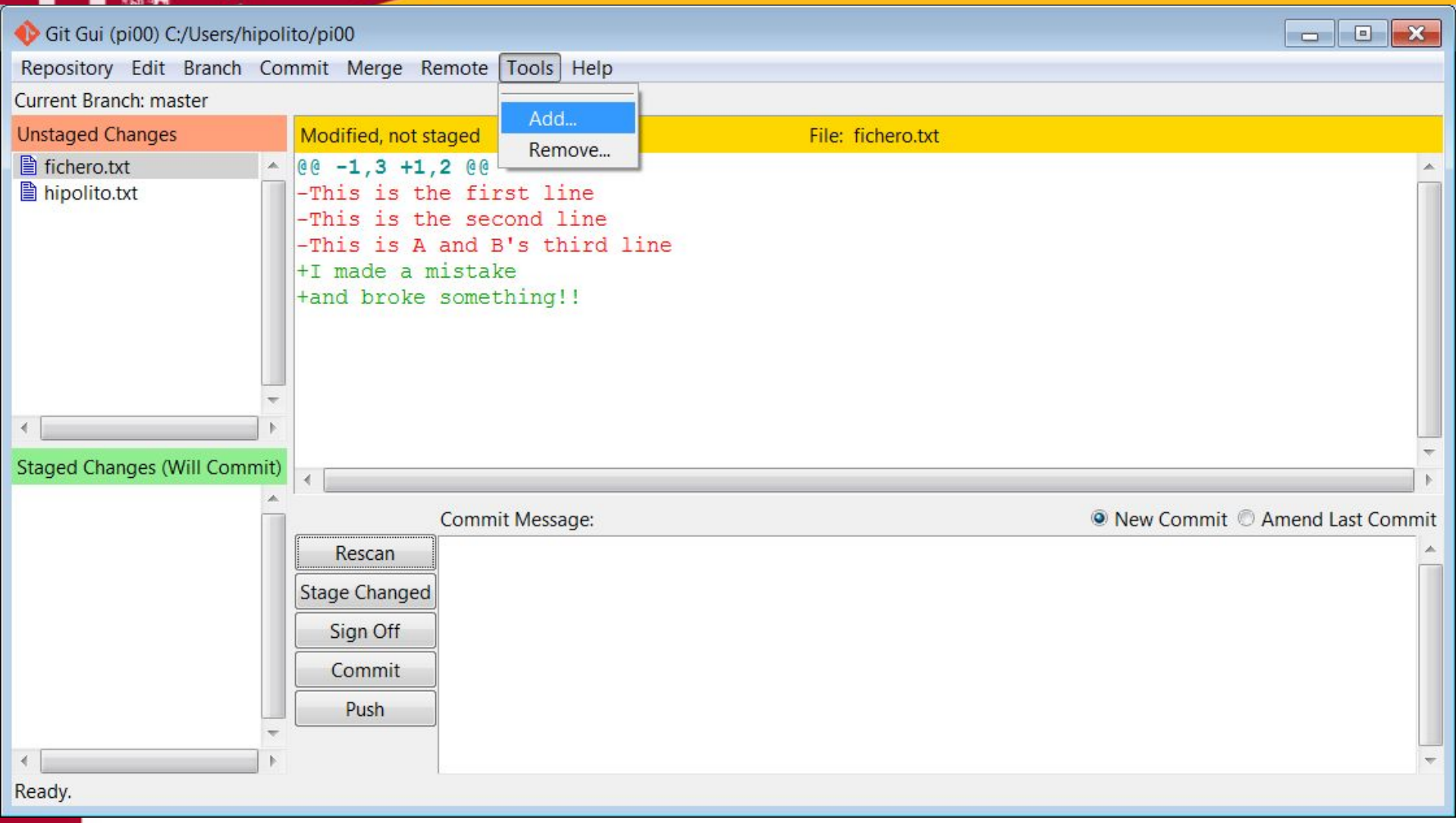
Contenido

- 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

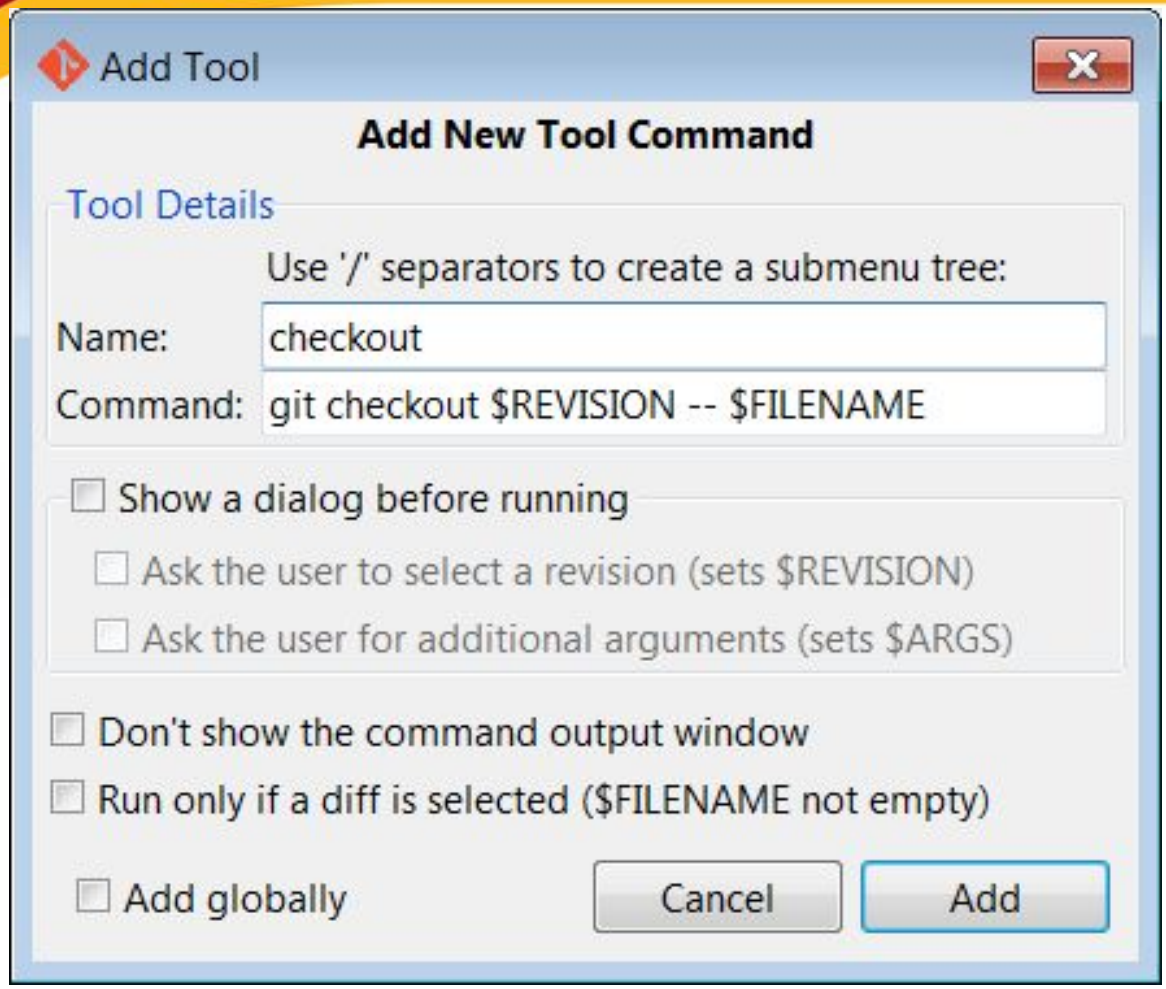
Revertir errores usando checkout



Revertir errores usando checkout

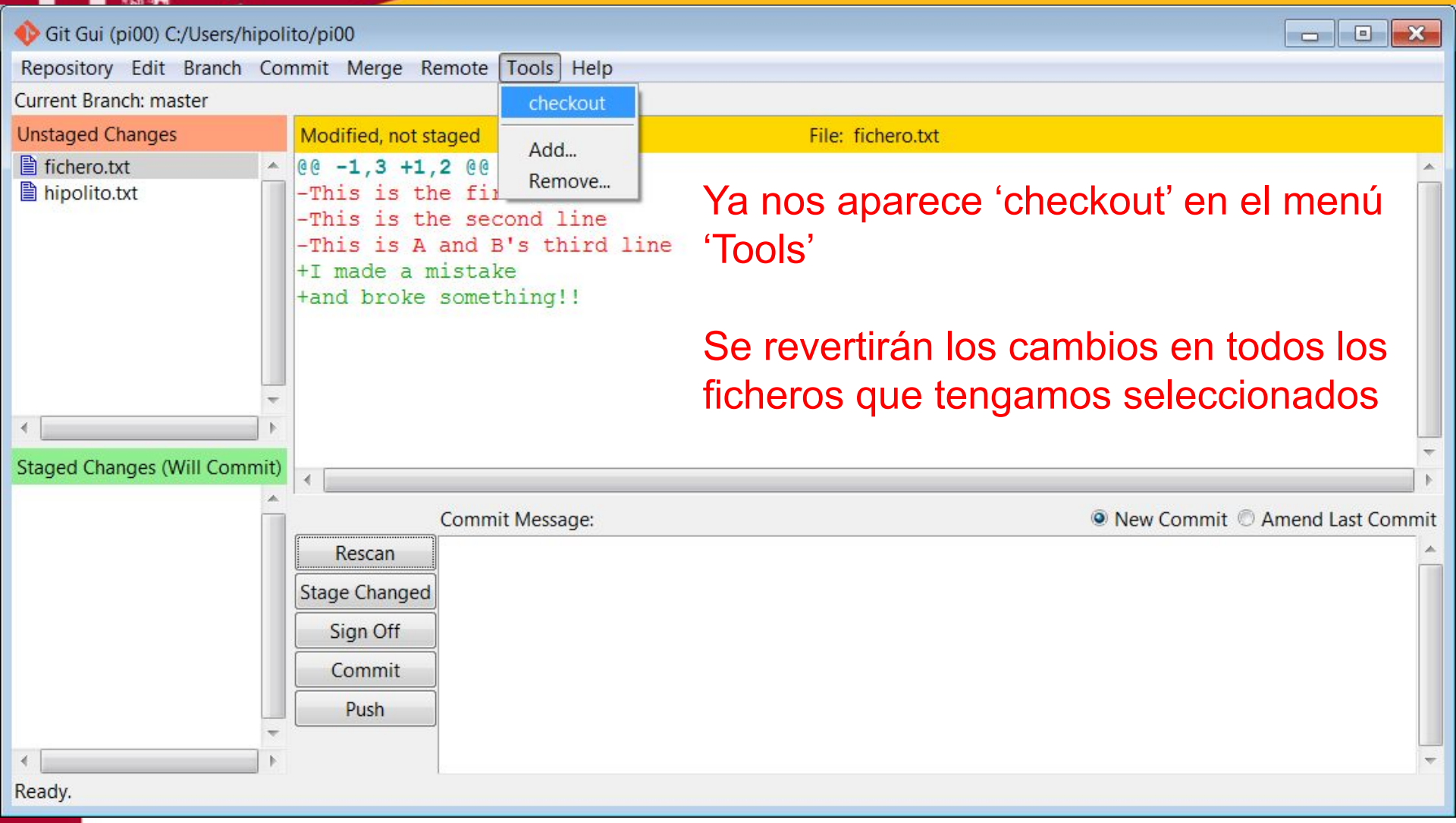


Revertir errores usando checkout



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

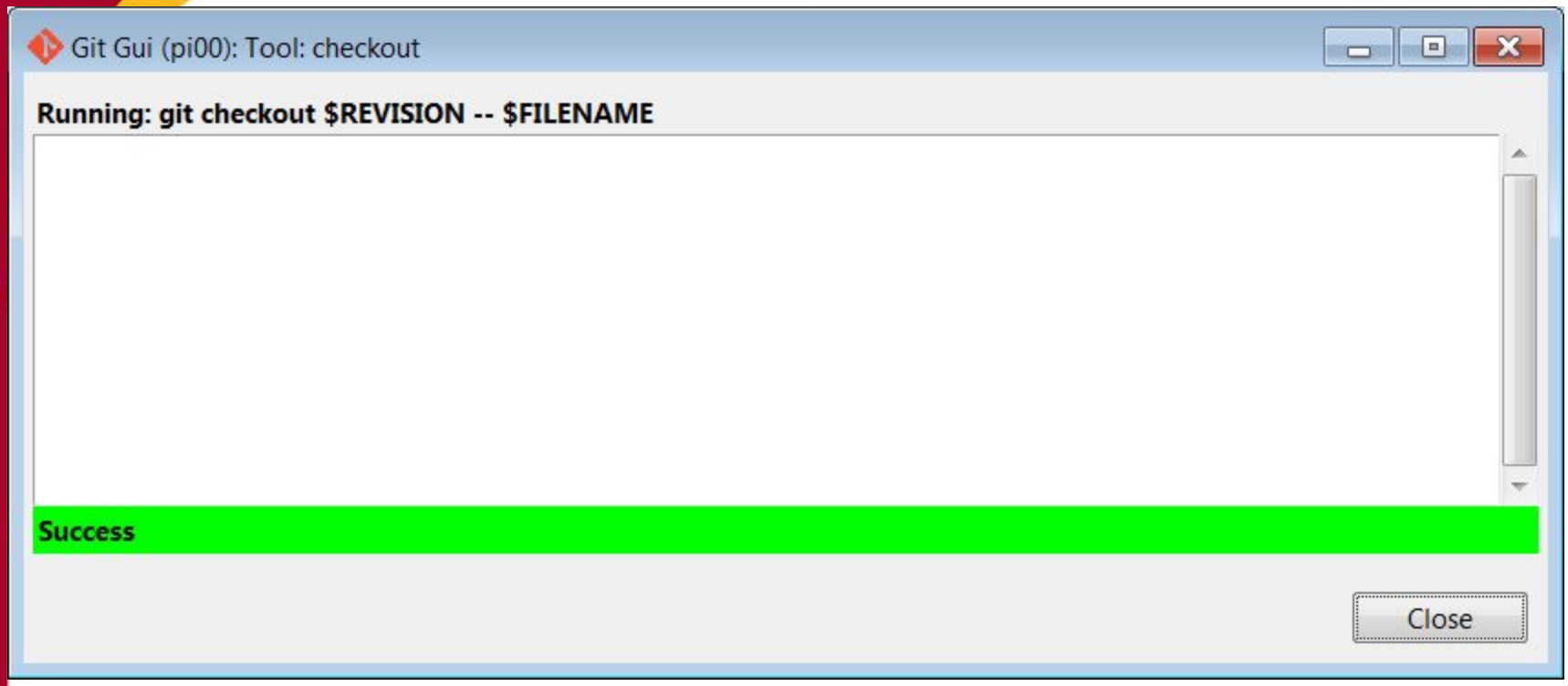
Revertir errores usando checkout



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

Se revertirán los cambios en todos los ficheros que tengamos seleccionados

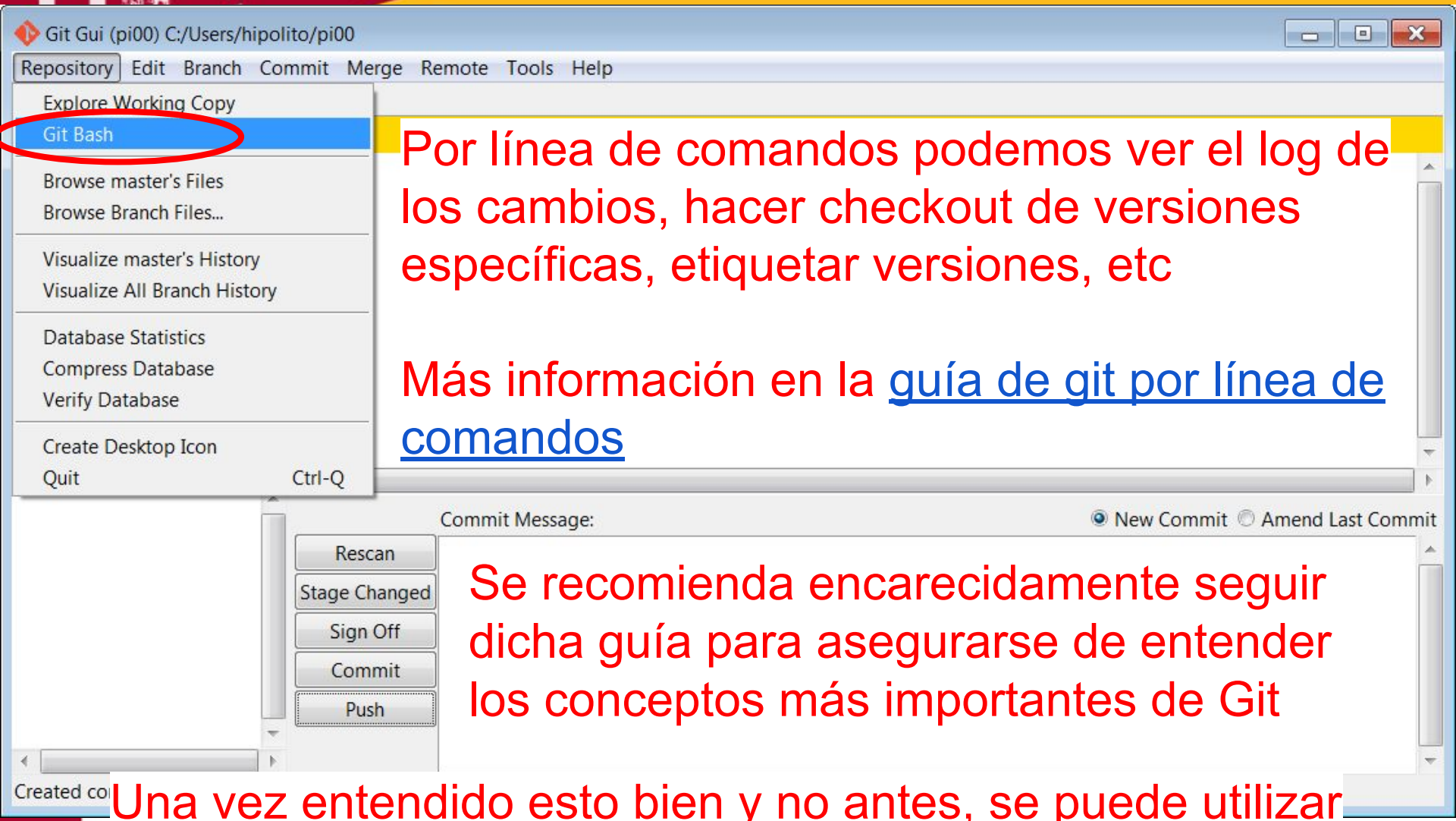
Revertir errores usando checkout



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

Para profundizar

Para profundizar



Por línea de comandos podemos ver el log de los cambios, hacer checkout de versiones específicas, etiquetar versiones, etc

Más información en la guía de git por línea de comandos

Se recomienda encarecidamente seguir dicha guía para asegurarse de entender los conceptos más importantes de Git

Una vez entendido esto bien y no antes, se puede utilizar alguna de las GUIs que aparecen en: https://git-scm.com/downloads/guis

Valor… y al repo!