



git

Tener listo Git

Trabajando con repositorios git

Haciendo committing al realizar cambios

Staging Area

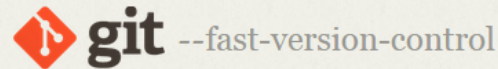
Volviendo atrás a lo que hemos hecho

Tener listo Git

- Git ya viene instalado en las últimas distribuciones de Mac y Linux por defecto
- En Windows y algunas distribuciones de Mac tenemos que descargarlo e instalarlo
- En Linux es sencillo, solo hay que utilizar el gestor de paquetes de tu distribución o el comando `sudo apt-get install git`
- En Windows es más complicado, ya que fue creado para Linux y muchas dependencias no están disponibles en Windows
- Existe un instalador disponible en la página de Git para facilitar esta tarea

Tener listo Git

<http://git-scm.com>



Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient staging areas, and **multiple workflows**.



Learn Git in your browser for free with **Try Git**.



About

The advantages of Git compared to other source control systems.



Documentation

Command reference pages, Pro Git book content, videos and other material.



Downloads

GUI clients and binary releases for all major platforms.



Community

Get involved! Mailing list, chat, development and more.



Pro Git by Scott Chacon is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).



Windows GUIs



Tarballs



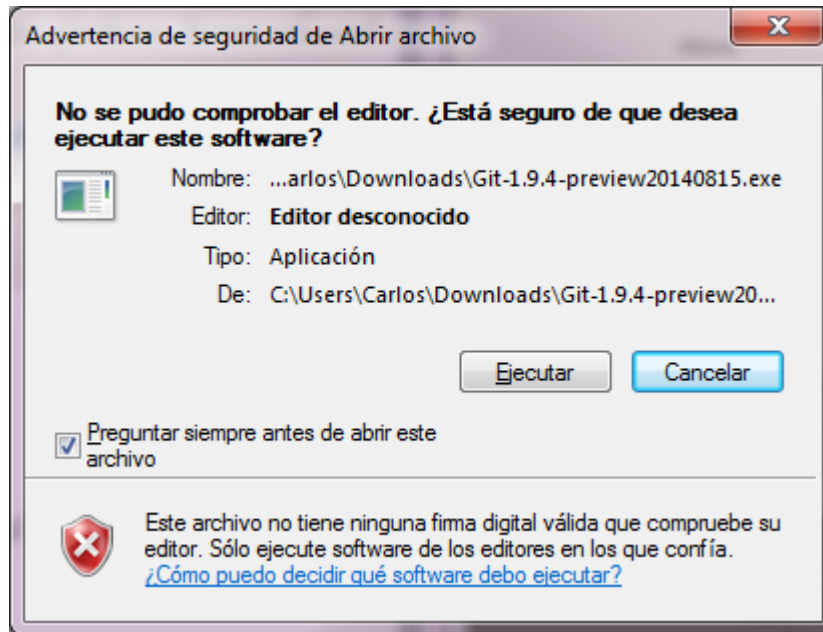
Mac Build



Source Code

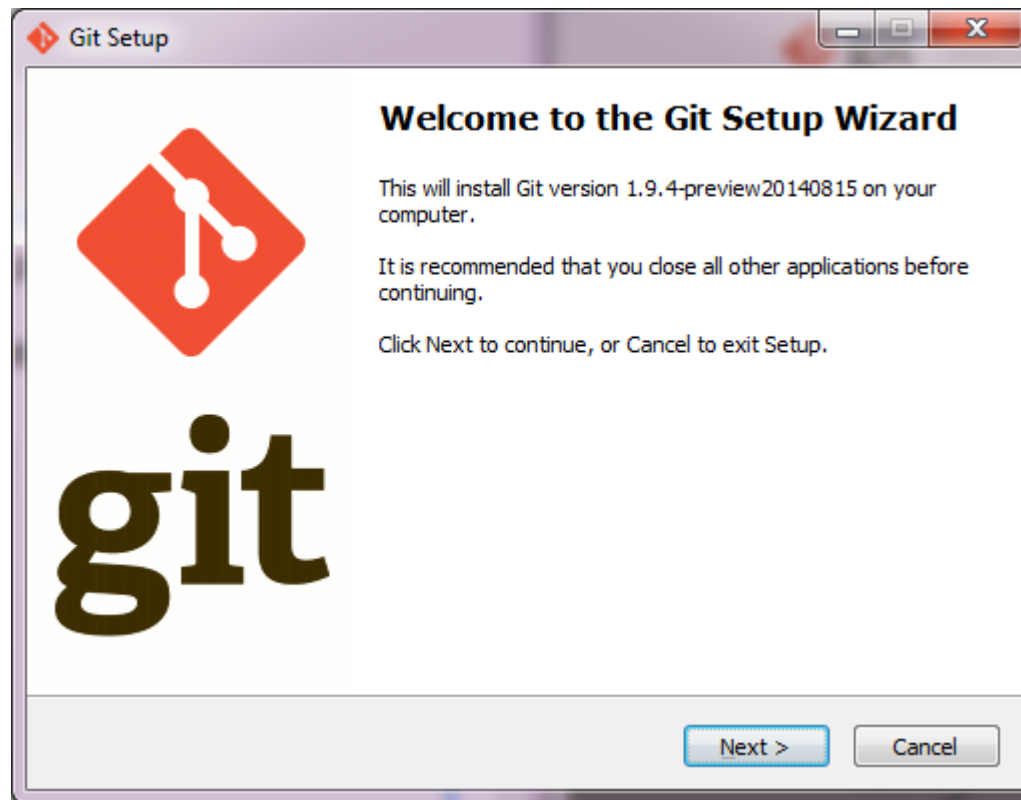
Tener listo Git

- Descargamos la última versión para windows y la instalamos.
- Ejecutamos el programa y aceptamos en la advertencia de seguridad para continuar



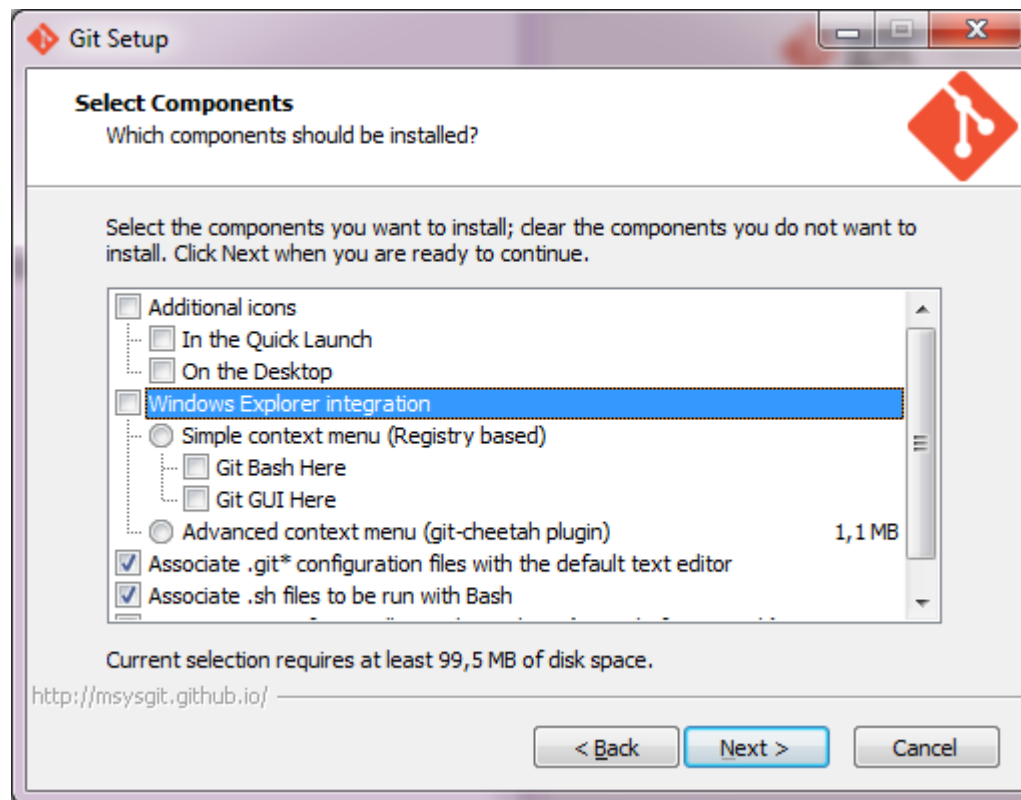
Tener listo Git

- Pulsamos siguiente , aceptamos la licencia y dejamos la ruta de instalación por defecto



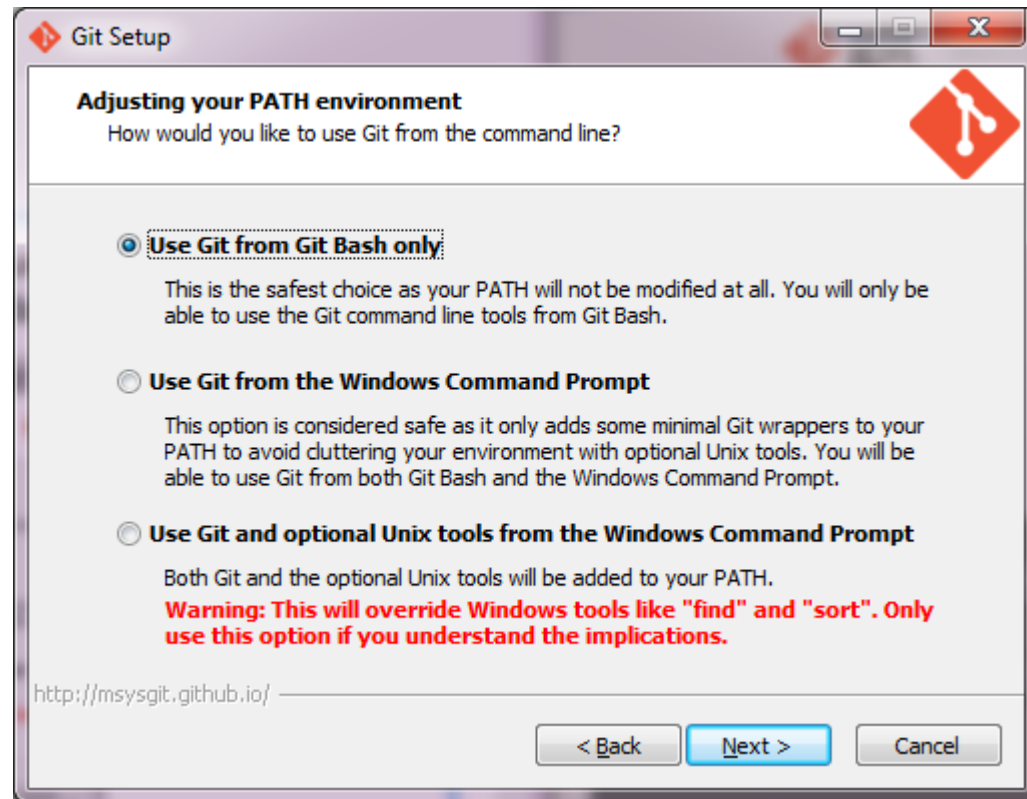
Tener listo Git

- Quitamos la opción de integrarlo con el explorador de windows



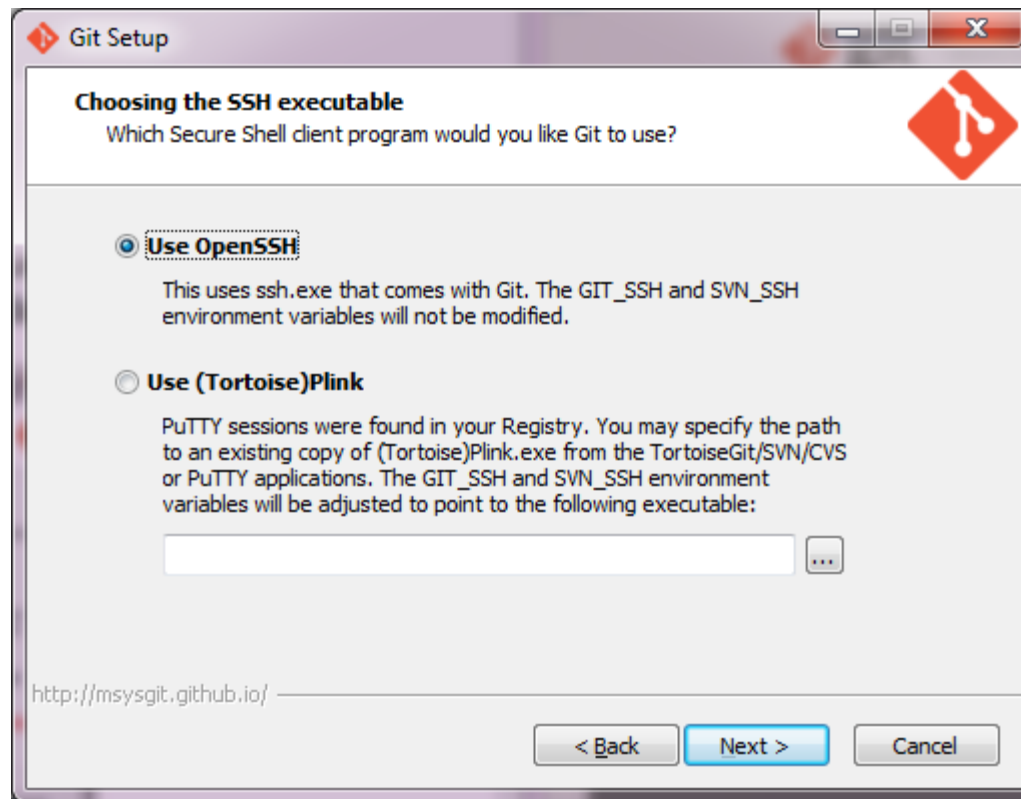
Tener listo Git

- Aquí decidimos desde donde queremos que se acepten los comandos, dejamos marcada la primera



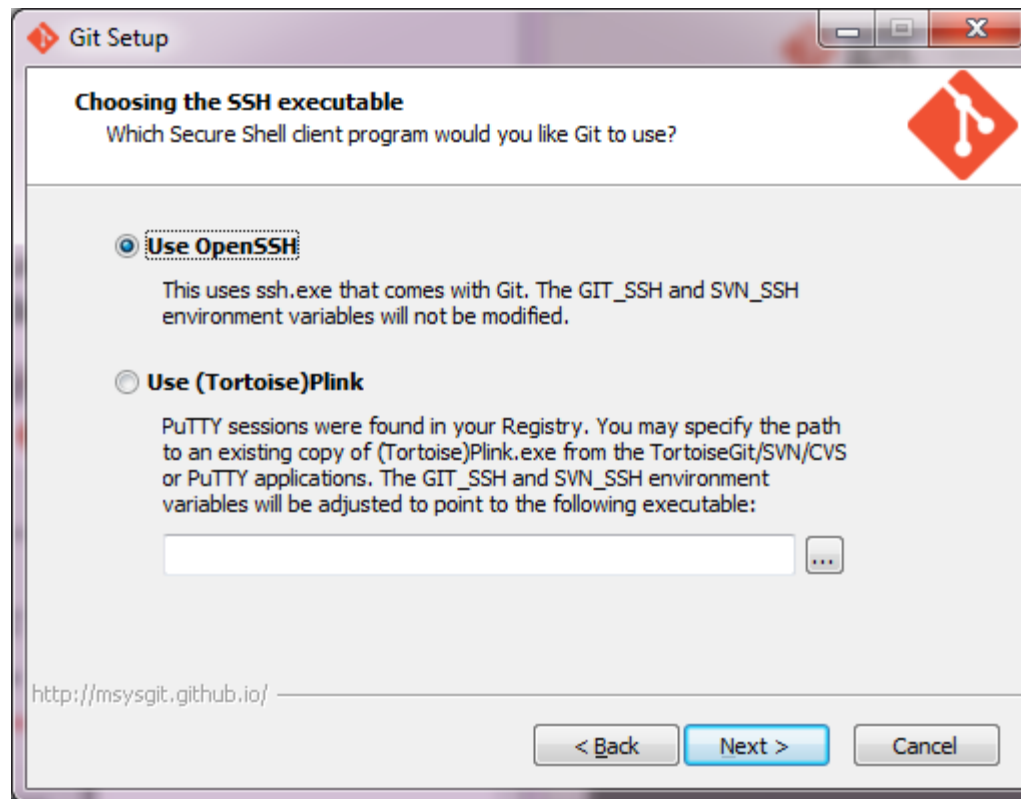
Tener listo Git

- Dejamos marcada la opción de utilizar OpenSSH



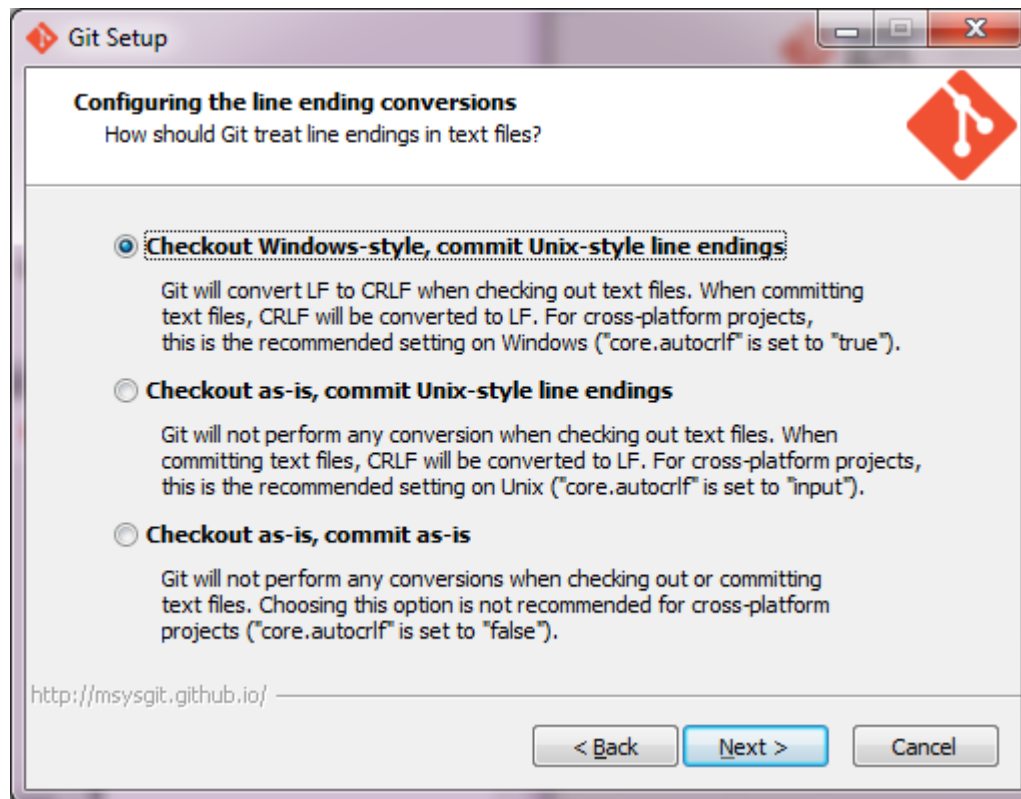
Tener listo Git

- Dejamos marcada la opción de utilizar OpenSSH



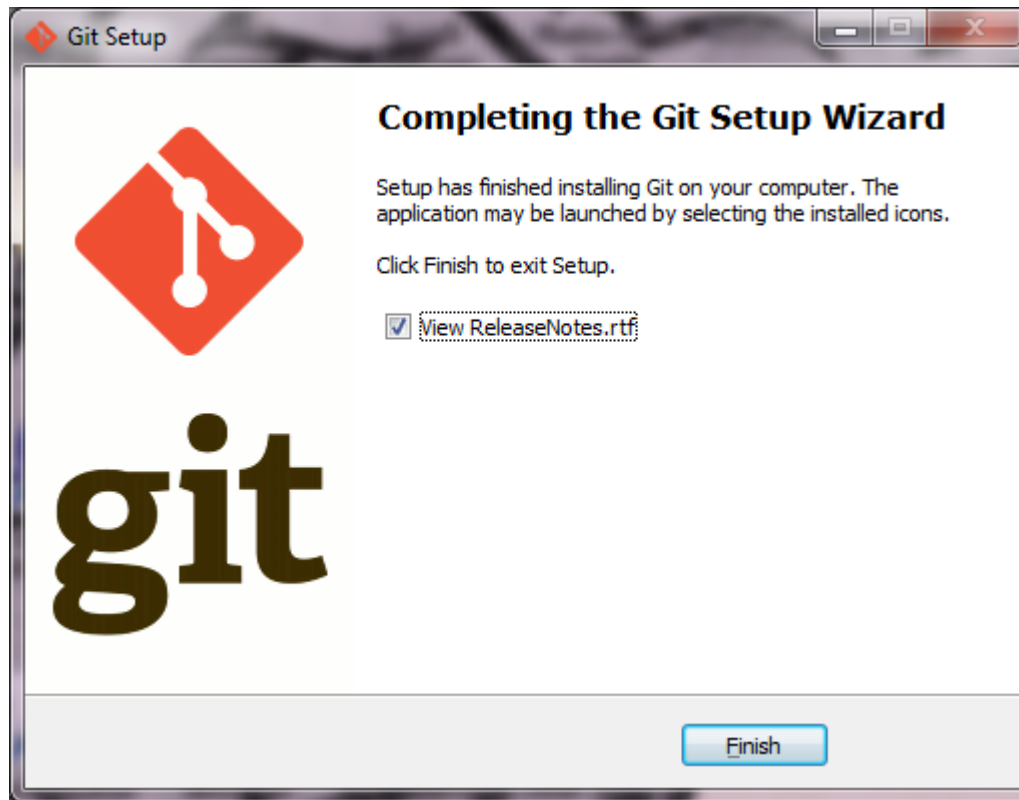
Tener listo Git

- Dejamos marcada la opción la primera opción más segura



Tener listo Git

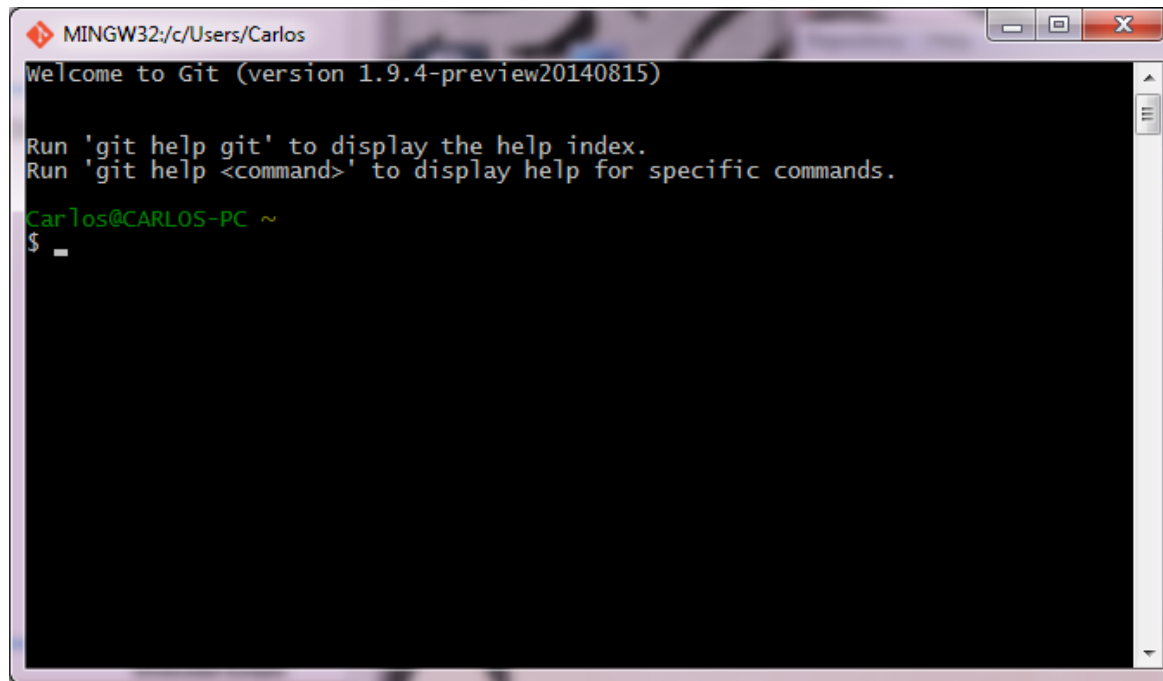
- Ya se ha finalizado la instalación



Tener listo Git

Git Bash

- Git bash para modo comandos



```
MINGW32:/c/Users/Carlos
Welcome to Git (version 1.9.4-preview20140815)

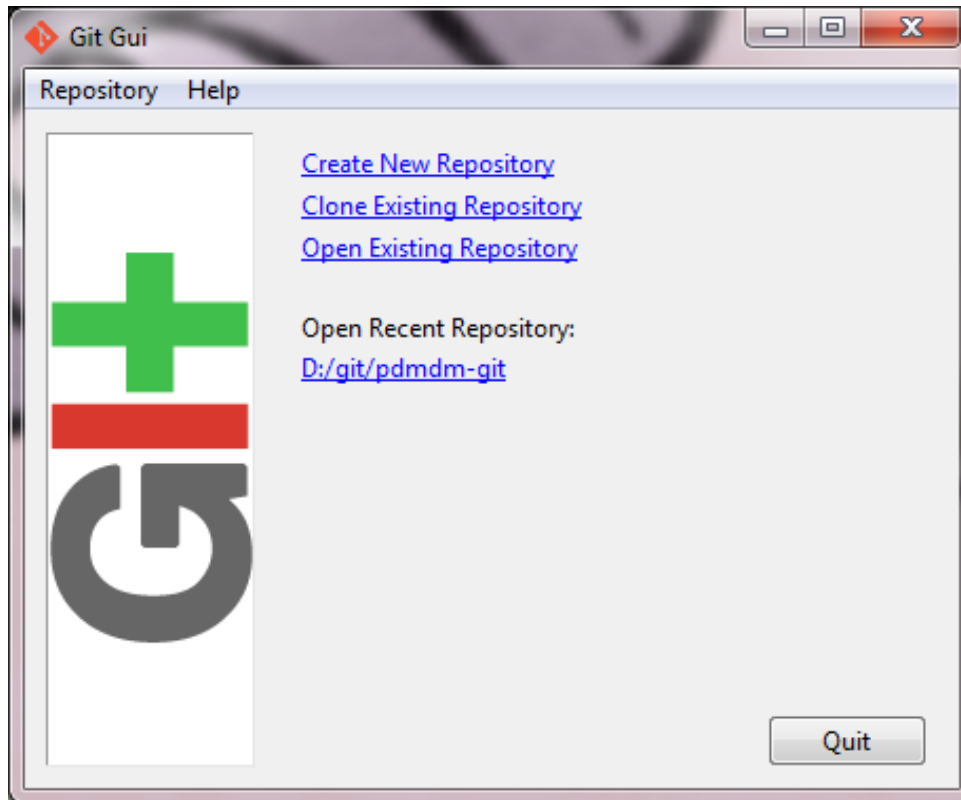
Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

Carlos@CARLOS-PC ~
$
```

Tener listo Git

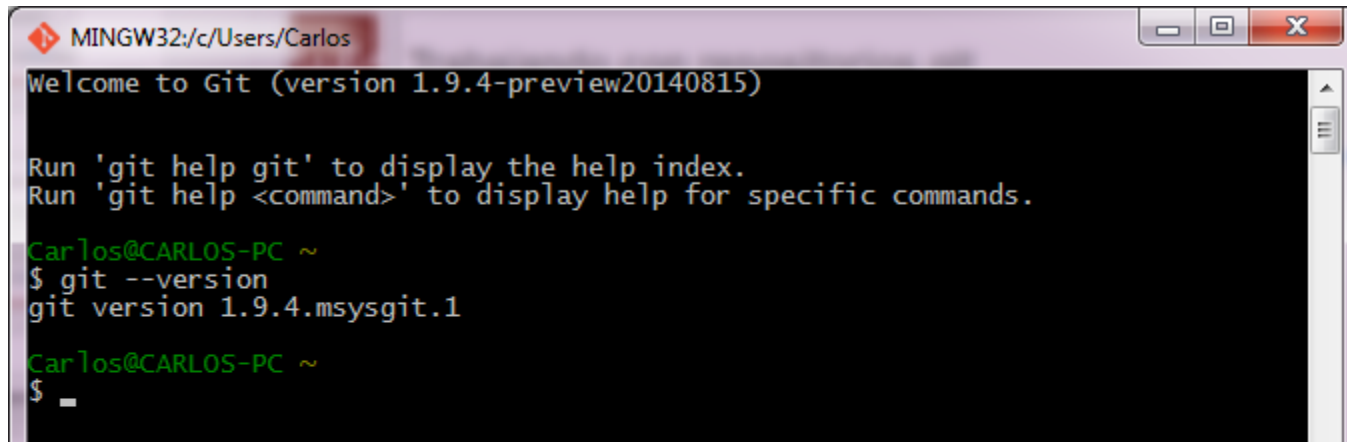
Git Gui

- Git en modo gráfico



Trabajando con repositorios git

- Para comprobar que tenemos instalado git, ejecutamos `git --version`



A screenshot of a terminal window titled 'MINGW32:/c/Users/Carlos'. The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The terminal output is as follows:

```
Welcome to Git (version 1.9.4-preview20140815)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

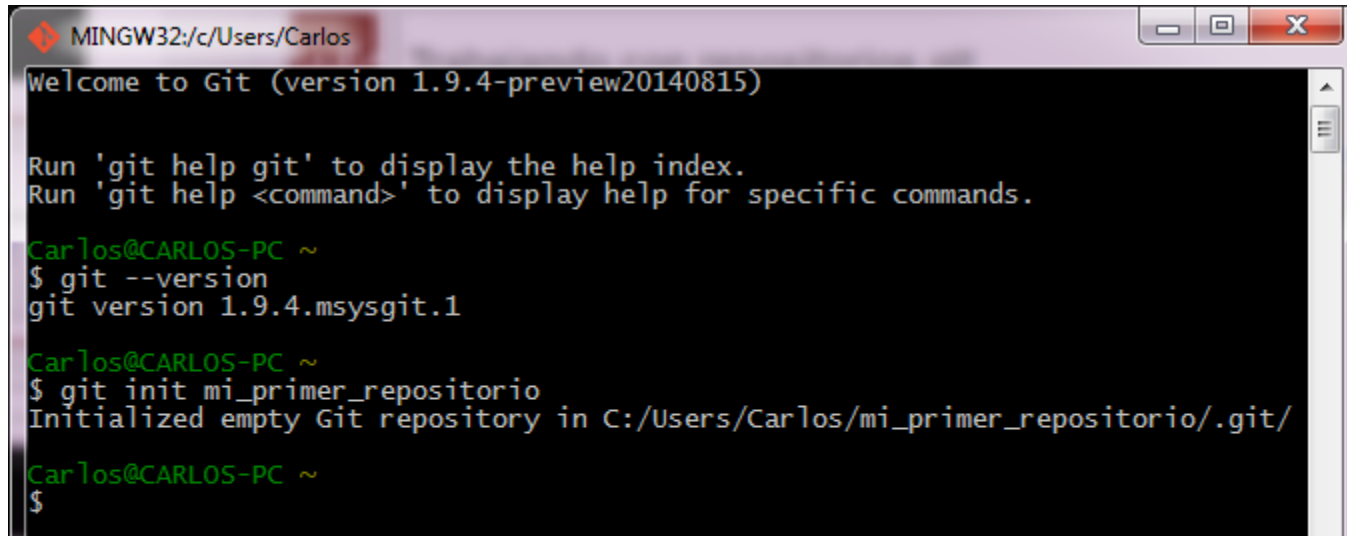
Carlos@CARLOS-PC ~
$ git --version
git version 1.9.4.msysgit.1

Carlos@CARLOS-PC ~
$ _
```

Trabajando con repositorios git

crear repositorios

- Para crear un repositorio ejecutamos el comando `git init [nombre_del_proyecto]`



```
MINGW32/c/Users/Carlos
Welcome to Git (version 1.9.4-preview20140815)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

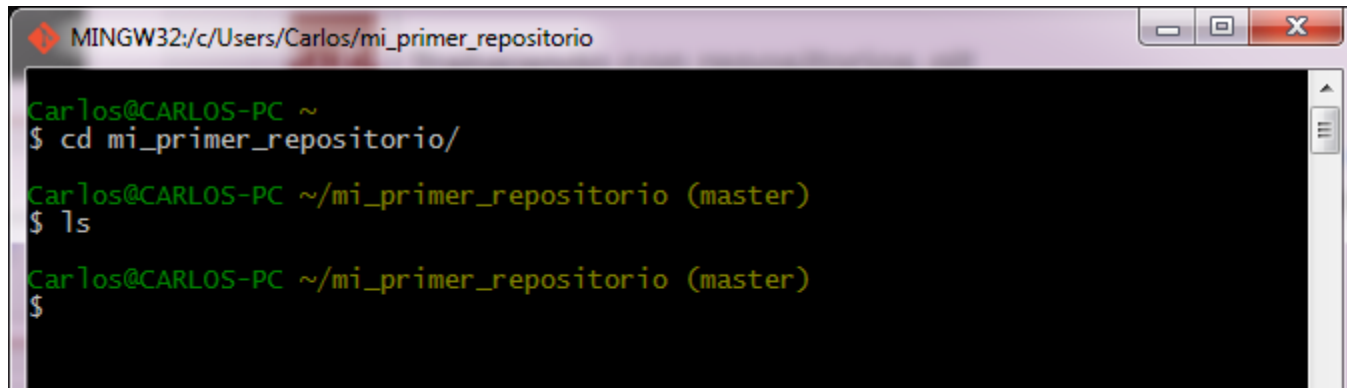
Carlos@CARLOS-PC ~
$ git --version
git version 1.9.4.msysgit.1

Carlos@CARLOS-PC ~
$ git init mi_primer_repositorio
Initialized empty Git repository in C:/Users/Carlos/mi_primer_repositorio/.git/

Carlos@CARLOS-PC ~
$
```

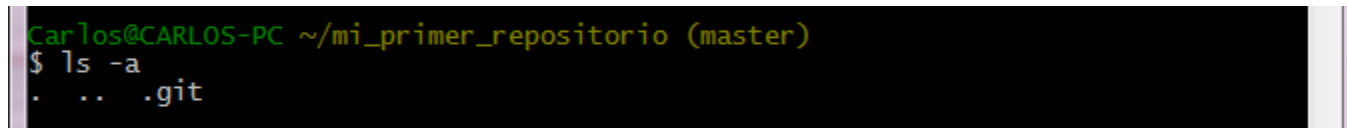

Trabajando con repositorios git

- Tenemos que utilizar los comandos POSIX de linux para movernos en git bash

A screenshot of a Windows terminal window titled 'MINGW32:/c/Users/Carlos/mi_primer_repositorio'. The terminal shows a series of commands and their outputs. The prompt is 'Carlos@CARLOS-PC ~'. The first command is '\$ cd mi_primer_repositorio/'. The second command is '\$ ls', which outputs 'Carlos@CARLOS-PC ~/mi_primer_repositorio (master)'. The third command is '\$', which outputs 'Carlos@CARLOS-PC ~/mi_primer_repositorio (master)'.

```
MINGW32:/c/Users/Carlos/mi_primer_repositorio
Carlos@CARLOS-PC ~
$ cd mi_primer_repositorio/
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ ls
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$
```

- Si mostramos el contenido parece vacío, esto es porque los archivos y carpetas están ocultos

A screenshot of a terminal window showing the command '\$ ls -a' and its output. The prompt is 'Carlos@CARLOS-PC ~/mi_primer_repositorio (master)'. The command '\$ ls -a' outputs '. .. .git'.

```
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ ls -a
. .. .git
```

Trabajando con repositorios git

- Dentro de la carpeta oculta .git se guardarán todos los archivos del repositorio de tu proyecto
- Creamos un proyecto ficticio llamado mi_app



```
MINGW32:/c/Users/Carlos/mi_primer_repositorio/mi_app
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ ls -a
. .. .git

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ mkdir mi_app

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ cd mi_app/

Carlos@CARLOS-PC ~/mi_primer_repositorio/mi_app (master)
$ touch file1

Carlos@CARLOS-PC ~/mi_primer_repositorio/mi_app (master)
$ touch file2

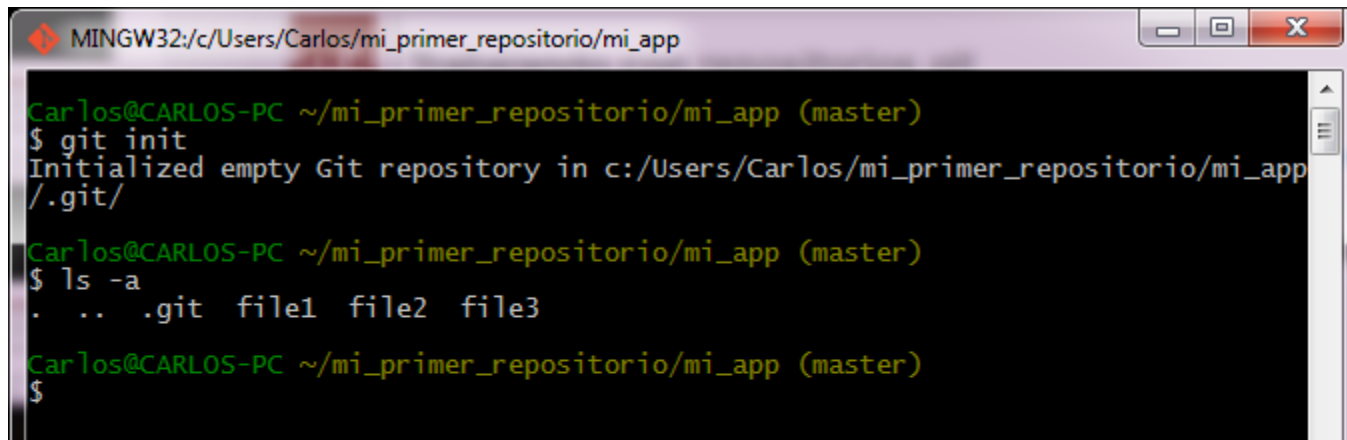
Carlos@CARLOS-PC ~/mi_primer_repositorio/mi_app (master)
$ touch file3

Carlos@CARLOS-PC ~/mi_primer_repositorio/mi_app (master)
$ ls
file1 file2 file3

Carlos@CARLOS-PC ~/mi_primer_repositorio/mi_app (master)
$
```

Trabajando con repositorios git

- Si ejecutamos git init, estaremos creando un nuevo repositorio para el directorio en el que nos encontramos



```
MINGW32:/c/Users/Carlos/mi_primer_repositorio/mi_app
Carlos@CARLOS-PC ~/mi_primer_repositorio/mi_app (master)
$ git init
Initialized empty Git repository in c:/Users/Carlos/mi_primer_repositorio/mi_app/.git/

Carlos@CARLOS-PC ~/mi_primer_repositorio/mi_app (master)
$ ls -a
.  ..  .git  file1  file2  file3

Carlos@CARLOS-PC ~/mi_primer_repositorio/mi_app (master)
$
```

- El nombre del repositorio no es importante, podemos cambiar el nombre de la carpeta sin miedo mientras se conserve la carpeta .git
- Para borrar un repositorio, basta con borrar la carpeta .git

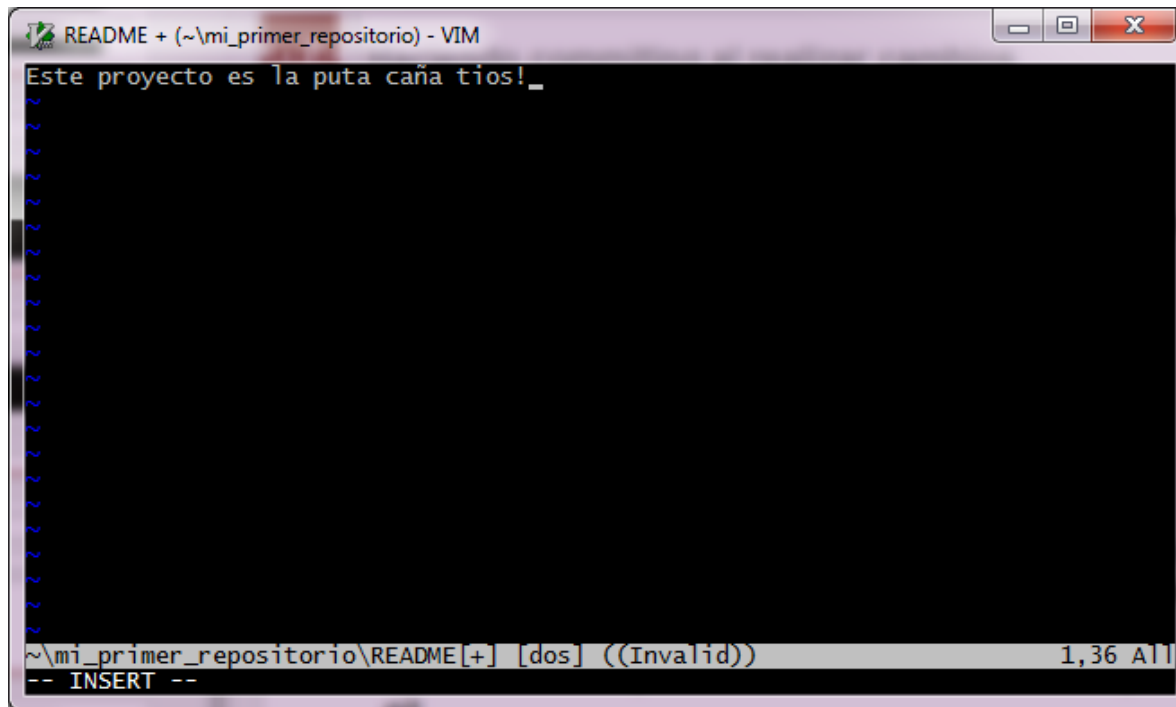
Trabajando con repositorios git

ejercicio

- Instalar Git en tu ordenador
- Abre git bash crea un repositorio en el directorio en que te encuentras
- Borra el repositorio que acabas de crear
- Crea un nuevo repositorio en la carpeta llamado como tu quieras

Haciendo committing al realizar cambios

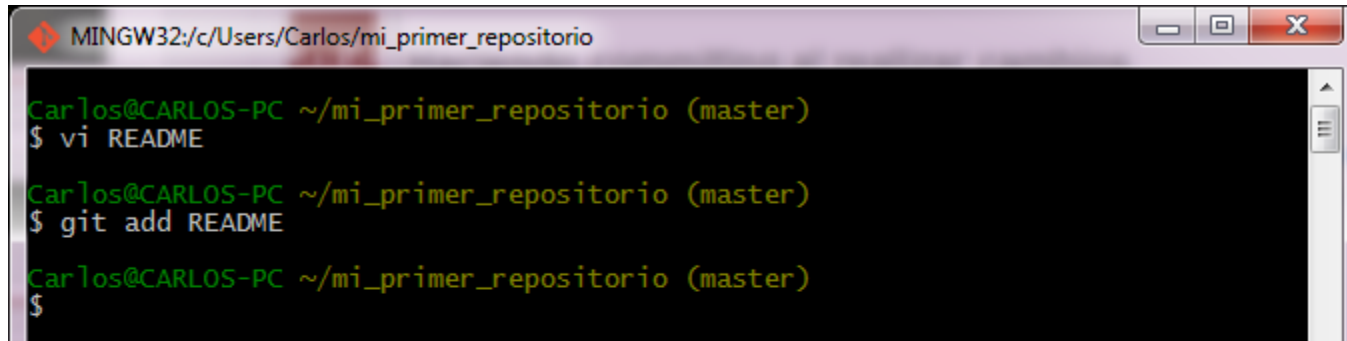
- Vamos a crear un archivo Readme que vamos a editar mediante vi con un texto



The screenshot shows a VIM editor window titled "README + (~\mi_primer_repositorio) - VIM". The main text area is black with white text. The first line of the file is "Este proyecto es la puta caña tios!_". Below this line, there are several lines of blue wavy lines, indicating that the text is wrapped. At the bottom of the window, there is a status bar that reads: "~\mi_primer_repositorio\README[+] [dos] ((Invalid)) 1,36 All -- INSERT --".

Haciendo committing al realizar cambios

- Ahora que hemos creado un nuevo fichero, para hacer commit, primero tenemos que añadirlo al repositorio mediante el comando `git add`

A screenshot of a terminal window titled 'MINGW32:/c/Users/Carlos/mi_primer_repositorio'. The terminal shows a series of commands and their outputs. The prompt is 'Carlos@CARLOS-PC ~/mi_primer_repositorio (master)'. The first command is '\$ vi README', followed by '\$ git add README'. The prompt then changes to '\$' on a new line.

```
MINGW32:/c/Users/Carlos/mi_primer_repositorio

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ vi README

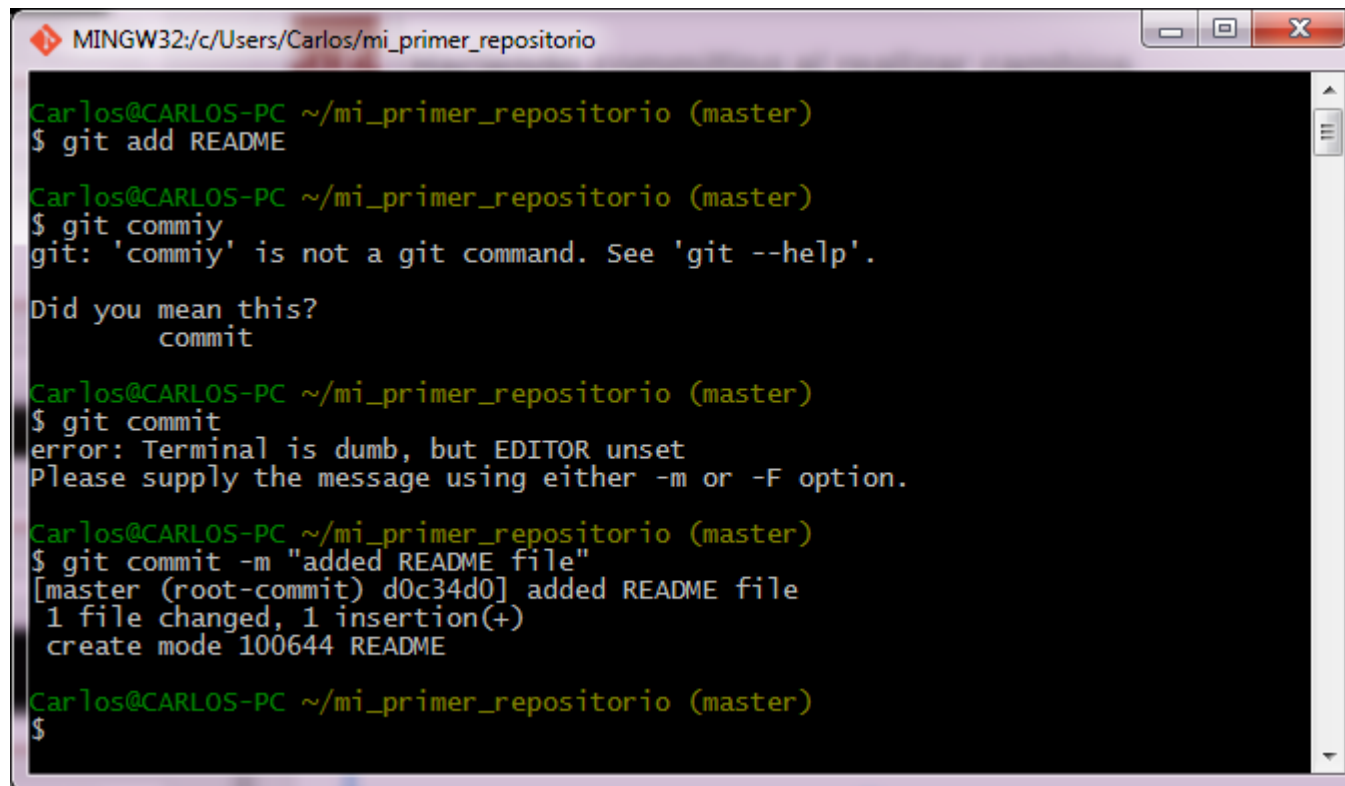
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git add README

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$
```

- A continuación ya podemos hacer commit mediante el comando `git commit`, se abrirá un editor donde indicaremos de forma corta y clara los cambios realizados
- Si no hiciéramos commit el archivo no se compartiría

Haciendo committing al realizar cambios

- Si estamos en windows y no hay un editor fijado, indicaremos el mensaje mediante la opción `-m`



```
MINGW32:/c/Users/Carlos/mi_primer_repositorio

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git add README

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git commiy
git: 'commiy' is not a git command. See 'git --help'.

Did you mean this?
    commit

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git commit
error: Terminal is dumb, but EDITOR unset
Please supply the message using either -m or -F option.

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git commit -m "added README file"
[master (root-commit) d0c34d0] added README file
1 file changed, 1 insertion(+)
create mode 100644 README

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$
```

Haciendo committing al realizar cambios

- Para fijar el nombre de nuestro usuario y correo en todos nuestro proyectos utilizaremos el siguiente comando

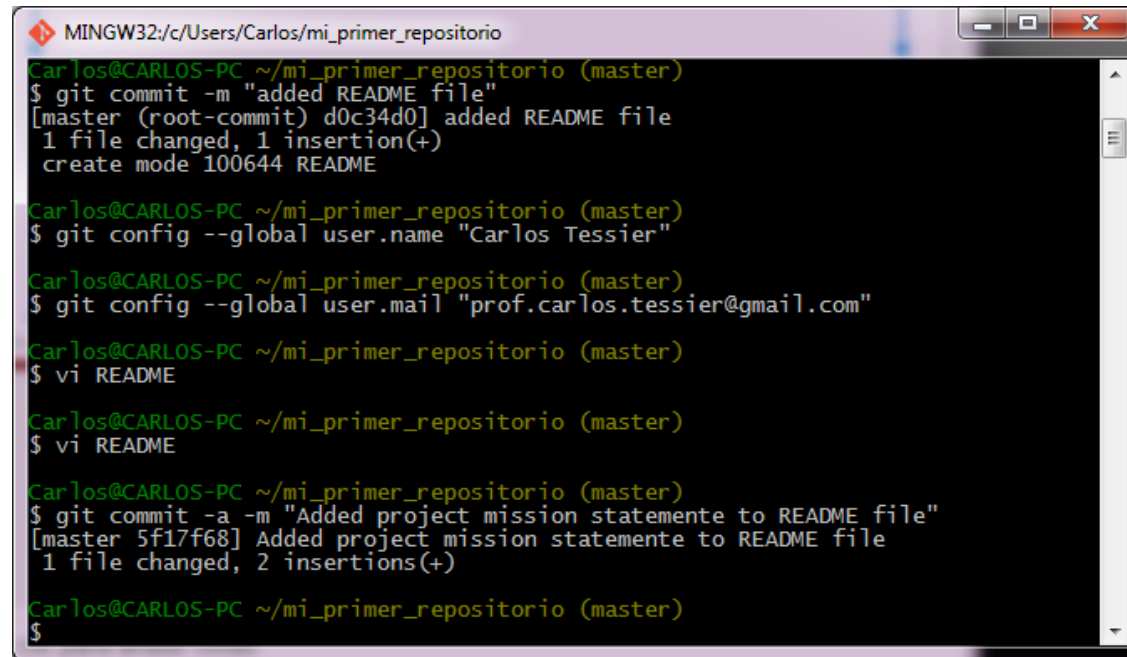
```
git config --global user.name "mi nombre"  
git config --global user.email  
"mi_mail@gmail.com"
```

- Para fijar el editor utilizaremos

```
git config --global core.editor nano
```


Haciendo committing al realizar cambios

- La respuesta es si, cuanto más commit realizemos, más detallado será el historial de nuestro proyecto
- Otra opción interesante es `-a`, esta opción dice a git que haga un commit a todos los ficheros modificados



```
MINGW32/c/Users/Carlos/mi_primer_repositorio
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git commit -m "added README file"
[master (root-commit) d0c34d0] added README file
1 file changed, 1 insertion(+)
create mode 100644 README

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git config --global user.name "Carlos Tessier"

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git config --global user.mail "prof.carlos.tessier@gmail.com"

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ vi README

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ vi README

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git commit -a -m "Added project mission statemente to README file"
[master 5f17f68] Added project mission statemente to README file
1 file changed, 2 insertions(+)

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$
```

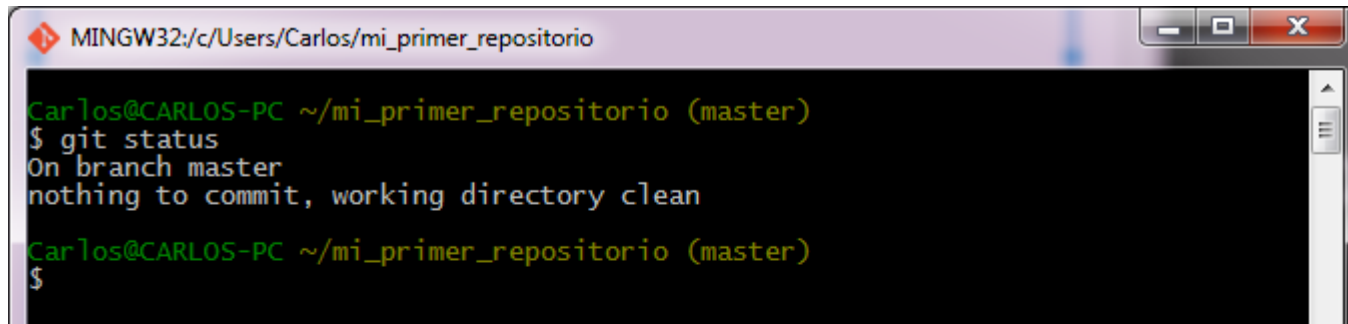
Haciendo committing al realizar cambios

Ejercicio

- Crea en tu repositorio un nuevo fichero llamado README.txt
- Realiza un commit a todos los ficheros modificados, no olvides añadir una descripción que tenga significado
- Realiza algún cambio en el fichero README y realiza otro commit

Staging Area

- El comando `git status` muestra en que rama estamos trabajando y si hay algún cambio para realizar commit

A screenshot of a terminal window titled "MINGW32:/c/Users/Carlos/mi_primer_repositorio". The terminal shows the command `git status` being executed. The output is: `On branch master`, `nothing to commit, working directory clean`. The prompt `Carlos@CARLOS-PC ~/mi_primer_repositorio (master)` is visible at the top and bottom of the terminal output.

```
MINGW32:/c/Users/Carlos/mi_primer_repositorio
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git status
On branch master
nothing to commit, working directory clean
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$
```

- Si el fichero ya ha sido añadido al repositorio y no ha sido modificado desde el último commit no se ejecutarán acciones relevantes

Staging Area

- Si añadimos nuevos ficheros, el comando git status nos mostrará los ficheros nuevos que no está siendo seguidos por el repositorio

```
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ touch file1 file2 file3

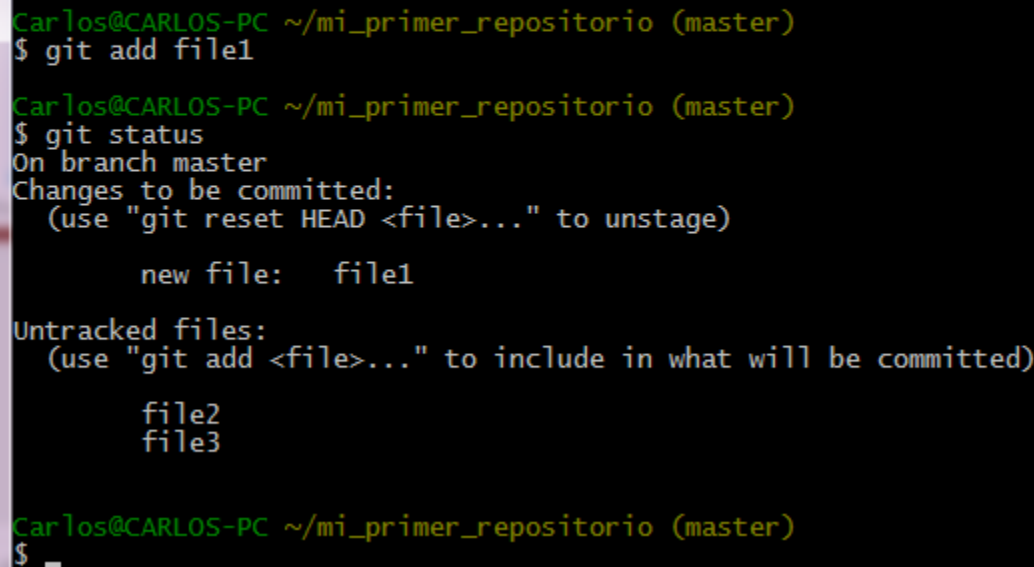
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        file1
        file2
        file3

nothing added to commit but untracked files present (use "git add" to track)
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ _
```

Staging Area

- Si añadimos un fichero al repositorio pero no hacemos commit, este fichero pasará al staging área para que estén listos cuando ejecutes el comando `git commit`

A terminal window with a black background and green text. The prompt is 'Carlos@CARLOS-PC ~/mi_primer_repositorio (master)'. The first command is '\$ git add file1'. The second command is '\$ git status'. The output shows 'On branch master', 'Changes to be committed:', '(use "git reset HEAD <file>..." to unstage)', 'new file: file1', 'Untracked files:', '(use "git add <file>..." to include in what will be committed)', 'file2', and 'file3'. The third command is '\$' followed by a cursor.

```
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git add file1

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   file1

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

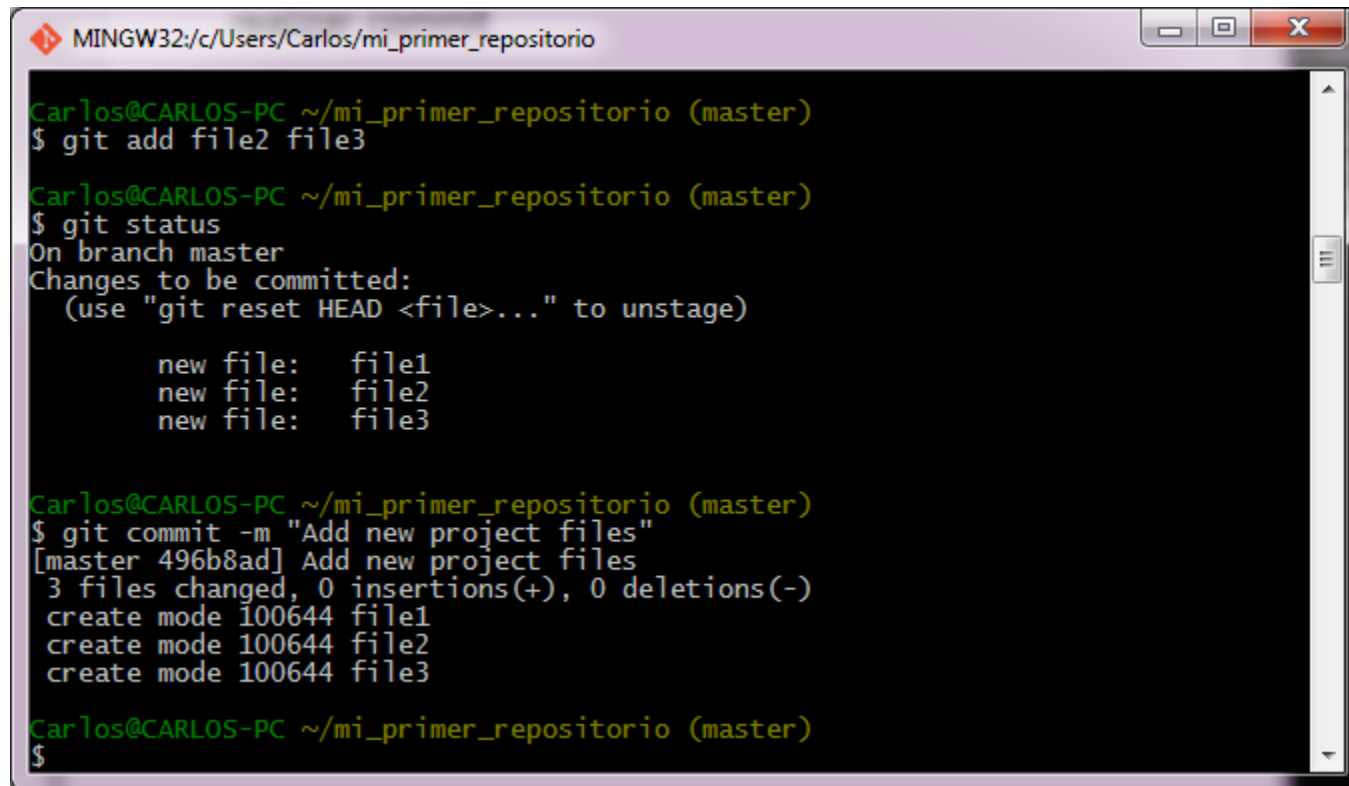
        file2
        file3

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$
```

- Es como si fuera un paquete listo para ser enviado por correo

Staging Area

- Podemos añadir el resto de ficheros al staging area y realizar commit



```
MINGW32/c/Users/Carlos/mi_primer_repositorio

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git add file2 file3

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   file1
        new file:   file2
        new file:   file3

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git commit -m "Add new project files"
[master 496b8ad] Add new project files
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 file1
create mode 100644 file2
create mode 100644 file3

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$
```

Staging Area

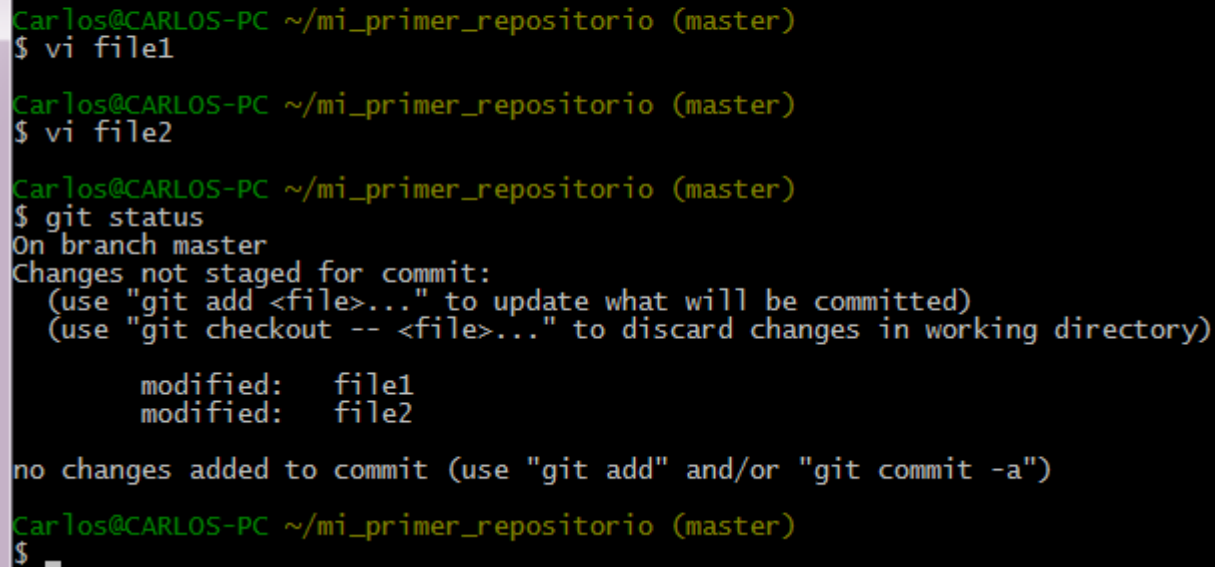
- Si volvemos a ejecutar git status, no mostrará nada.

```
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git status
On branch master
nothing to commit, working directory clean

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$
```


Staging Area

- Esta vez vamos a modificar un par de ficheros y ejecutar de nuevo git status

A terminal window with a black background and green text. The prompt is 'Carlos@CARLOS-PC ~/mi_primer_repositorio (master)'. The user enters '\$ vi file1' and '\$ vi file2'. Then they enter '\$ git status'. The output shows 'On branch master', 'Changes not staged for commit:', and two lines of 'modified: file1' and 'modified: file2'. It also includes instructions on how to stage changes. The prompt returns to '\$ _' at the end.

```
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ vi file1

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ vi file2

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git status
On branch master
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:   file1
        modified:   file2

no changes added to commit (use "git add" and/or "git commit -a")
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ _
```

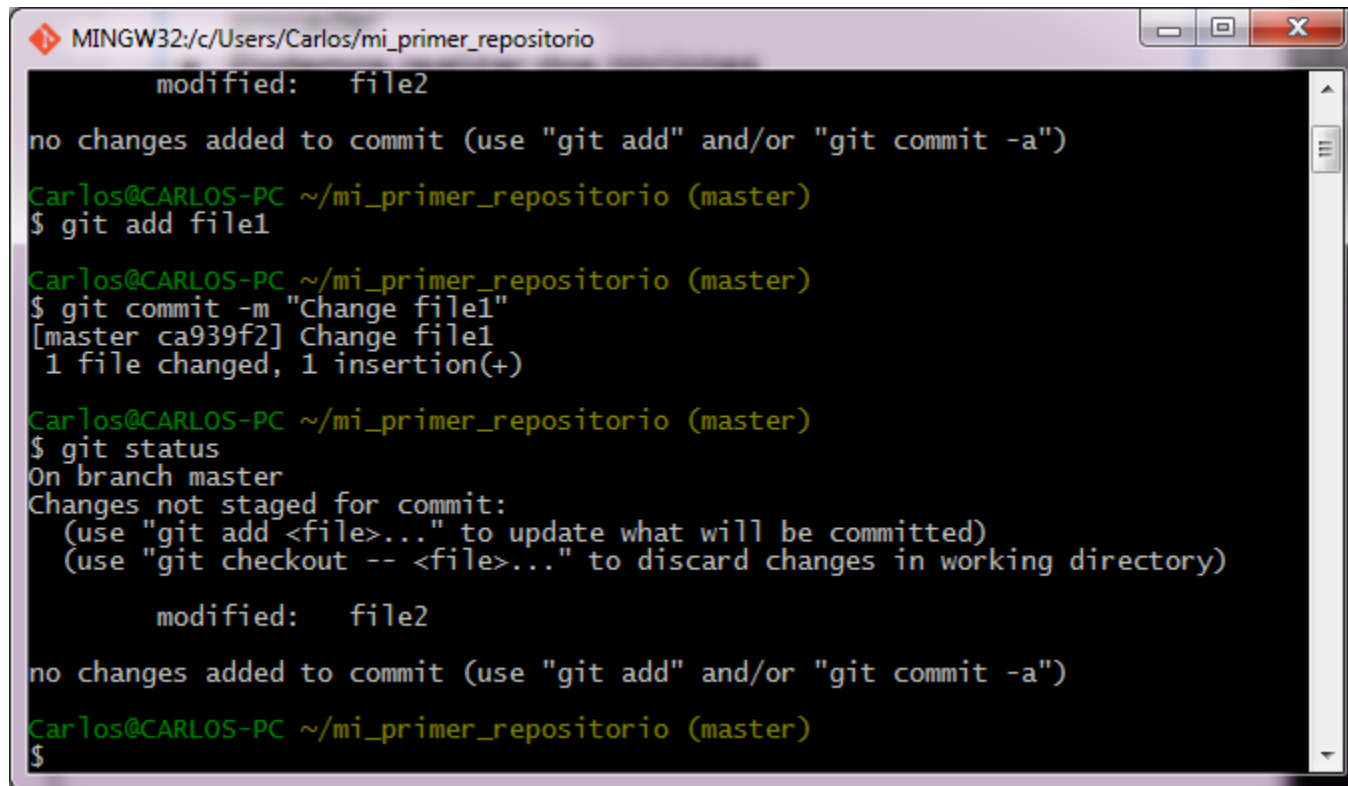
- En este nuevo listado “changes not staged for commit”

Staging Area

- Si nos fijamos el listado mismo nos indica como proceder
- Podemos realizar dos opciones
 - Utilizar `git add` para añadir los ficheros al staging area para que estén preparados para el commit
 - Utilizar `git commit -a` para realizar un commit a todos los ficheros modificados
- ¿Qué ocurrirá si solo añadimos el primer fichero y no hacemos commit a todos los ficheros?

Staging Area

- El segundo fichero seguirá en el not staged area mientras que el primero ha desaparecido

A screenshot of a terminal window titled 'MINGW32:/c/Users/Carlos/mi_primer_repositorio'. The terminal shows a sequence of Git commands and their outputs. It starts with 'modified: file2' and a message 'no changes added to commit'. Then, 'git add file1' is run. Next, 'git commit -m "Change file1"' is executed, resulting in a commit hash 'ca939f2' and the message 'Change file1', with '1 file changed, 1 insertion(+)' noted. Finally, 'git status' is run, showing 'On branch master' and 'Changes not staged for commit:'. It lists '(use "git add <file>..." to update what will be committed)' and '(use "git checkout -- <file>..." to discard changes in working directory)'. It then shows 'modified: file2' and the same 'no changes added to commit' message. The prompt returns to '\$'.

Staging Area

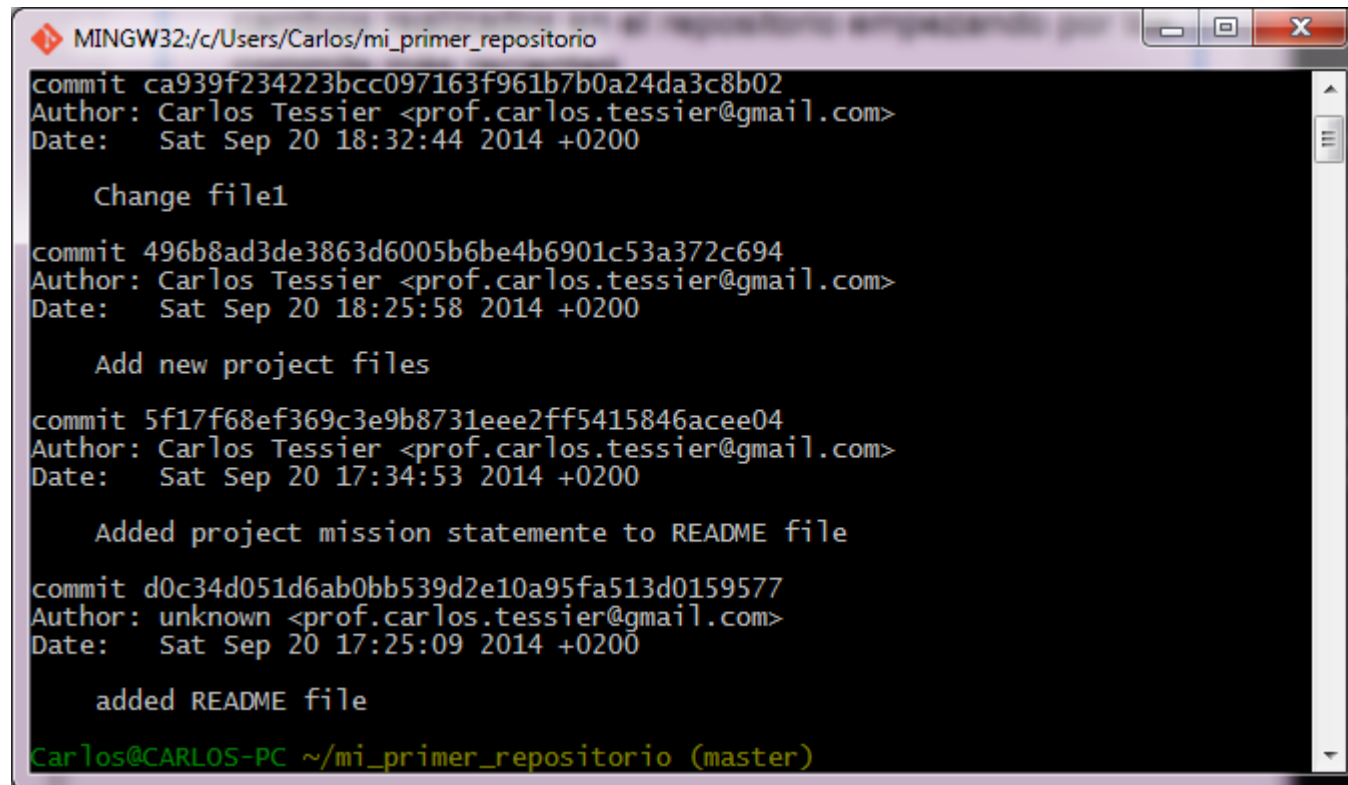
Ejercicio

- Modifica de nuevo el fichero README
- Imprime el estado del repositorio git
- Añade el fichero README al staging area
- Imprime de nuevo el estado del repositorio git
- Crea un nuevo fichero llamado index.html
- Añade el fichero index.html al repositorio
- Realiza un commit a todos los cambios realizados en el staging area
- Comprueba el estado del repositorio git una vez más para asegurarte que está limpio

Volviendo atrás a lo que hemos hecho

log del historial

- El comando `git log` muestra el historial de todos los cambios realizados en el repositorio empezando por los commits más recientes



```
MINGW32:/c/Users/Carlos/mi_primer_repositorio
commit ca939f234223bcc097163f961b7b0a24da3c8b02
Author: Carlos Tessier <prof.carlos.tessier@gmail.com>
Date: Sat Sep 20 18:32:44 2014 +0200

    Change file1

commit 496b8ad3de3863d6005b6be4b6901c53a372c694
Author: Carlos Tessier <prof.carlos.tessier@gmail.com>
Date: Sat Sep 20 18:25:58 2014 +0200

    Add new project files

commit 5f17f68ef369c3e9b8731eee2ff5415846acee04
Author: Carlos Tessier <prof.carlos.tessier@gmail.com>
Date: Sat Sep 20 17:34:53 2014 +0200

    Added project mission statemente to README file

commit d0c34d051d6ab0bb539d2e10a95fa513d0159577
Author: unknown <prof.carlos.tessier@gmail.com>
Date: Sat Sep 20 17:25:09 2014 +0200

    added README file

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
```

Volviendo atrás a lo que hemos hecho

log del historial

- Cada entrada log tiene mucha información
 - Un identificador único del commit llamado hash para asegurarse que no cambie en proyectos grandes
 - El autor y correo del usuario que lo realizó
 - La fecha en la que se realizó el commit
 - El mensaje que se guardó al realizar el commit
- Es importante que los mensajes tengan suficiente información para saber que se realizó en cada commit

Volviendo atrás a lo que hemos hecho

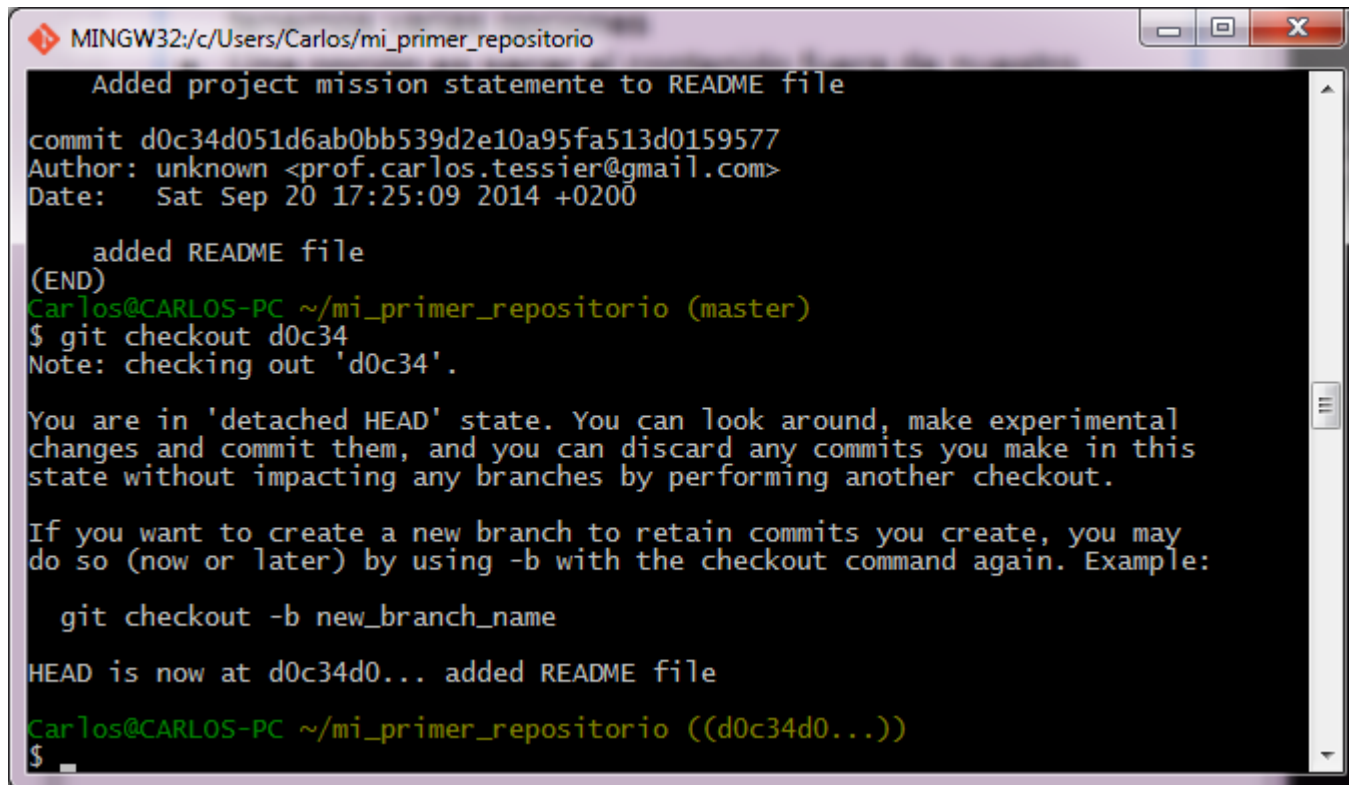
Ver los detalles de un proyecto anterior

- Si queremos ver los detalles de un proyecto anterior tenemos varias opciones
- Una opción es sacar el contenido fuera de nuestro repositorio, como si se tratase de un libro en una biblioteca, mediante el comando `git checkout [identificador]`
- Normalmente con los 5 primeros dígitos del identificador es suficiente

Volviendo atrás a lo que hemos hecho

Ver los detalles de un proyecto anterior

- Mostrará un mensaje avisándonos que estamos en una versión anterior y que si haces cambios pueden ocurrir efectos indeseados

A screenshot of a terminal window titled 'MINGW32:/c/Users/Carlos/mi_primer_repositorio'. The terminal shows the output of a 'git checkout' command. It starts with 'Added project mission statemente to README file', followed by commit details: 'commit d0c34d051d6ab0bb539d2e10a95fa513d0159577', 'Author: unknown <prof.carlos.tessier@gmail.com>', and 'Date: Sat Sep 20 17:25:09 2014 +0200'. Then it says 'added README file' and '(END)'. The prompt changes to 'Carlos@CARLOS-PC ~/mi_primer_repositorio (master)'. The user enters '\$ git checkout d0c34'. The output continues with 'Note: checking out \'d0c34\'.' followed by a detailed explanation of the 'detached HEAD' state. It says 'You are in \'detached HEAD\' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.' and 'If you want to create a new branch to retain commits you create, you may do so (now or later) by using -b with the checkout command again. Example:'. An example command is shown: 'git checkout -b new_branch_name'. The terminal then shows 'HEAD is now at d0c34d0... added README file'. The prompt changes to 'Carlos@CARLOS-PC ~/mi_primer_repositorio ((d0c34d0...))' and the user enters '\$' followed by a cursor. The terminal window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

```
MINGW32:/c/Users/Carlos/mi_primer_repositorio
Added project mission statemente to README file
commit d0c34d051d6ab0bb539d2e10a95fa513d0159577
Author: unknown <prof.carlos.tessier@gmail.com>
Date: Sat Sep 20 17:25:09 2014 +0200

    added README file
(END)
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git checkout d0c34
Note: checking out 'd0c34'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

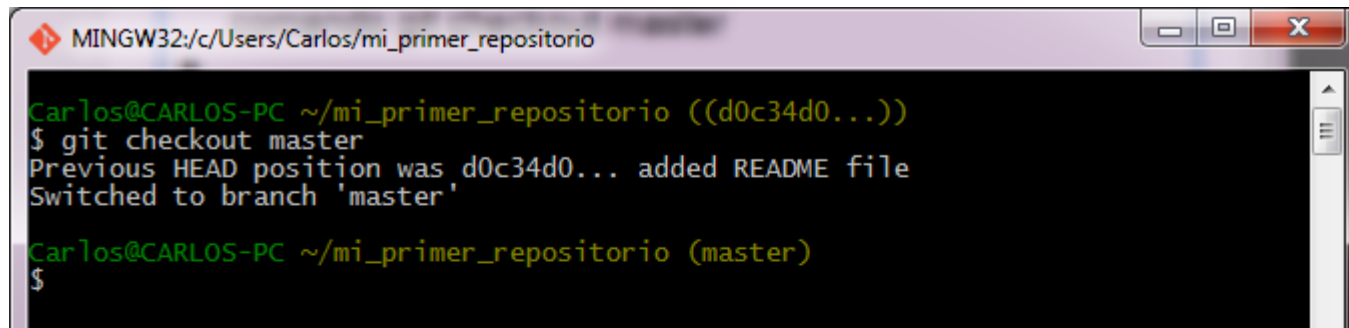
    git checkout -b new_branch_name

HEAD is now at d0c34d0... added README file
Carlos@CARLOS-PC ~/mi_primer_repositorio ((d0c34d0...))
$
```


Volviendo atrás a lo que hemos hecho

volver a la versión más reciente

- Para volver a la versión más actual de la rama maestra utilizaremos el comando `git checkout master`

A screenshot of a terminal window titled 'MINGW32:/c/Users/Carlos/mi_primer_repositorio'. The terminal shows the following text: 'Carlos@CARLOS-PC ~/mi_primer_repositorio ((d0c34d0...))', '\$ git checkout master', 'Previous HEAD position was d0c34d0... added README file', 'Switched to branch \'master\'', 'Carlos@CARLOS-PC ~/mi_primer_repositorio (master)', and '\$'. The terminal has a black background with green and white text. The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

```
MINGW32:/c/Users/Carlos/mi_primer_repositorio

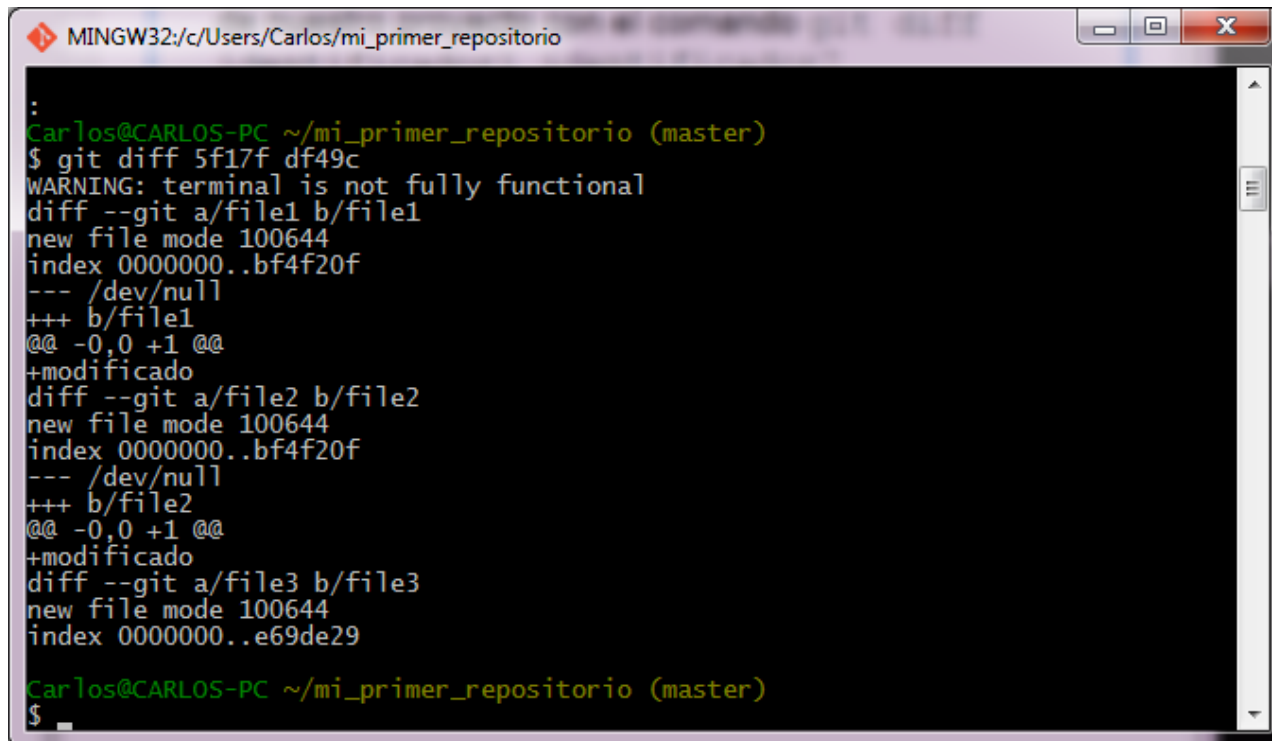
Carlos@CARLOS-PC ~/mi_primer_repositorio ((d0c34d0...))
$ git checkout master
Previous HEAD position was d0c34d0... added README file
Switched to branch 'master'

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$
```

Volviendo atrás a lo que hemos hecho

git diff

- Otra opción obtener las diferencias entre dos versiones de nuestro proyecto con el comando `git diff`
`identificador1 identificador2`



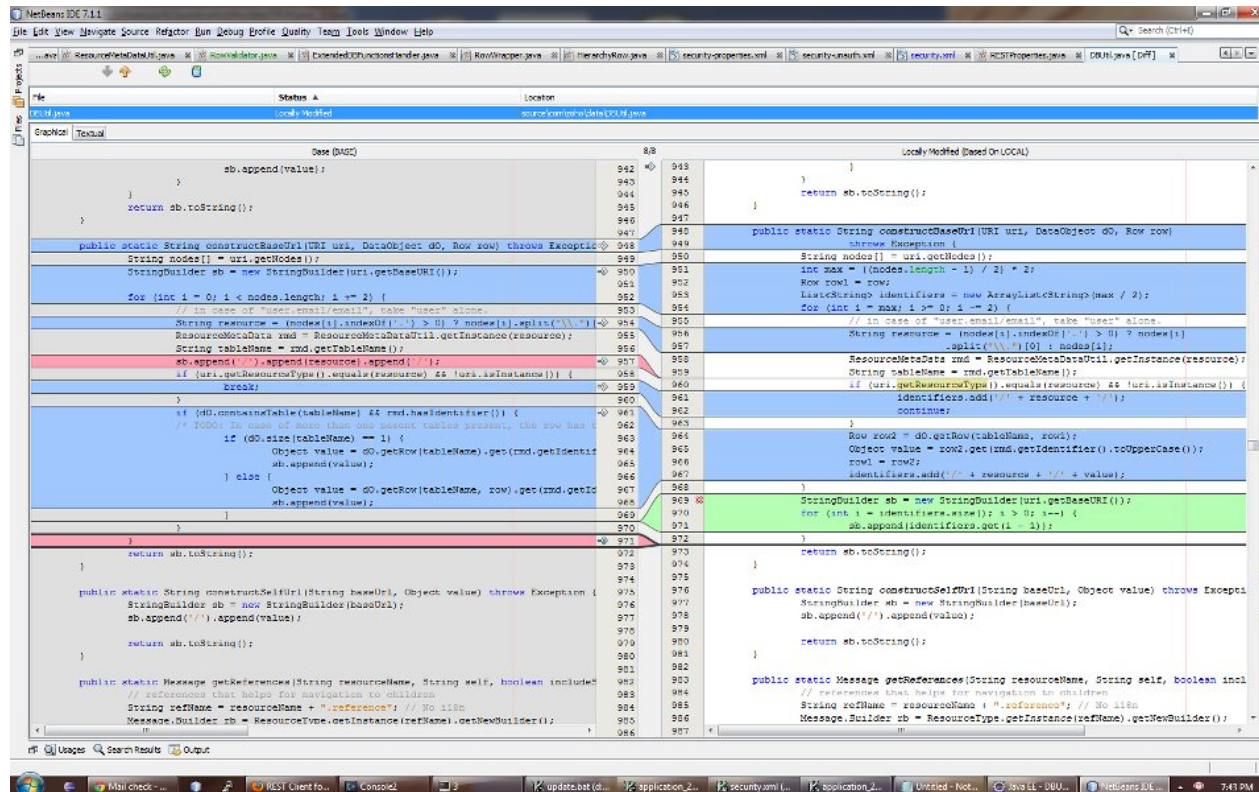
```
MINGW32:/c/Users/Carlos/mi_primer_repositorio
:
Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$ git diff 5f17f df49c
WARNING: terminal is not fully functional
diff --git a/file1 b/file1
new file mode 100644
index 0000000..bf4f20f
--- /dev/null
+++ b/file1
@@ -0,0 +1 @@
+modificado
diff --git a/file2 b/file2
new file mode 100644
index 0000000..bf4f20f
--- /dev/null
+++ b/file2
@@ -0,0 +1 @@
+modificado
diff --git a/file3 b/file3
new file mode 100644
index 0000000..e69de29

Carlos@CARLOS-PC ~/mi_primer_repositorio (master)
$
```

Volviendo atrás a lo que hemos hecho

diff en herramientas gráficas

- Existen herramientas gráficas que muestran la diferencia entre versiones de forma más amigable



Volviendo atrás a lo que hemos hecho

Ejercicio

- Crea un nuevo repositorio en una nueva carpeta
- Crea un fichero llamado index.html y
- Añádelo al staging area y realiza un commit
- Modifica el fichero index.html
- Visualiza el estado actual del repositorio
- Crea un nuevo fichero about.html y añádelo al staging area
- Añade al staging area los cambios realizados en index.html
- Realiza un commit a todos los cambios
- Muestra el historial del repositorio
- HEAD~1 es un identificador especial que se refiere al commit justo anterior, vete a la versión justo anterior
- Muestra el log para ver que estamos mirando en un commit anterior
- Vuelve al commit más reciente de la rama principal (HEAD)

Webgrafía

- <http://git-scm.com/>