

Herramienta GIT

<<FW ABC>>

Número Versión	Acción, C,M,D,A*	Fecha Acción•	Resumen Cambios	Responsables de la acción	Distribuido a
1.0	C	06/10/2017	Creación base de plantilla	Wilson Medina	-
1.1	M	29/01/2018	Creación de contenido	Juan Camilo Alvarez Uribe	

\*: C =Creación, M =Modificación, D =Distribución, A = Aprobación (Excluyentes).

•: El contenido de la columna Fecha Acción (AAAA-MM-DD) es requerido y se diligencia de la siguiente manera: Escribir para las acciones de Creación y Modificación del documento la fecha de iniciación de la acción, para Distribución la fecha de envío del documento.

## Contenido

I.	OBJETIVO DEL DOCUMENTO .....	2
1.1	General .....	2
1.2	Específicos .....	2
II.	INSTALACION .....	3
III.	GENERALIDADES .....	8
3.1	Entendiendo GIT .....	8
3.2	Configurar opciones globales de Git .....	9
3.3	Creación de Repositorio Local .....	9
3.4	Guardando cambios en el repositorio .....	11
3.5	Ramificación y fusión.....	17
IV.	Implementacion con gitHUB .....	24
4.1	Creación de repositorio en GitHub .....	24
V.	SENTENCIAS DE GIT.....	28
VI.	ERRORES DOCUMENTADOS.....	¡Error! Marcador no definido.
VIII.	ANEXOS.....	¡Error! Marcador no definido.
	<<todas las plantillas de apoyo y colaboración>> .....	¡Error! Marcador no definido.



Herramienta GIT

<<FW ABC>>

## I. OBJETIVO DEL DOCUMENTO

### ■ General

Brindar la documentación base para el conocimiento e implementación de la herramienta de control de versionamiento GIT.

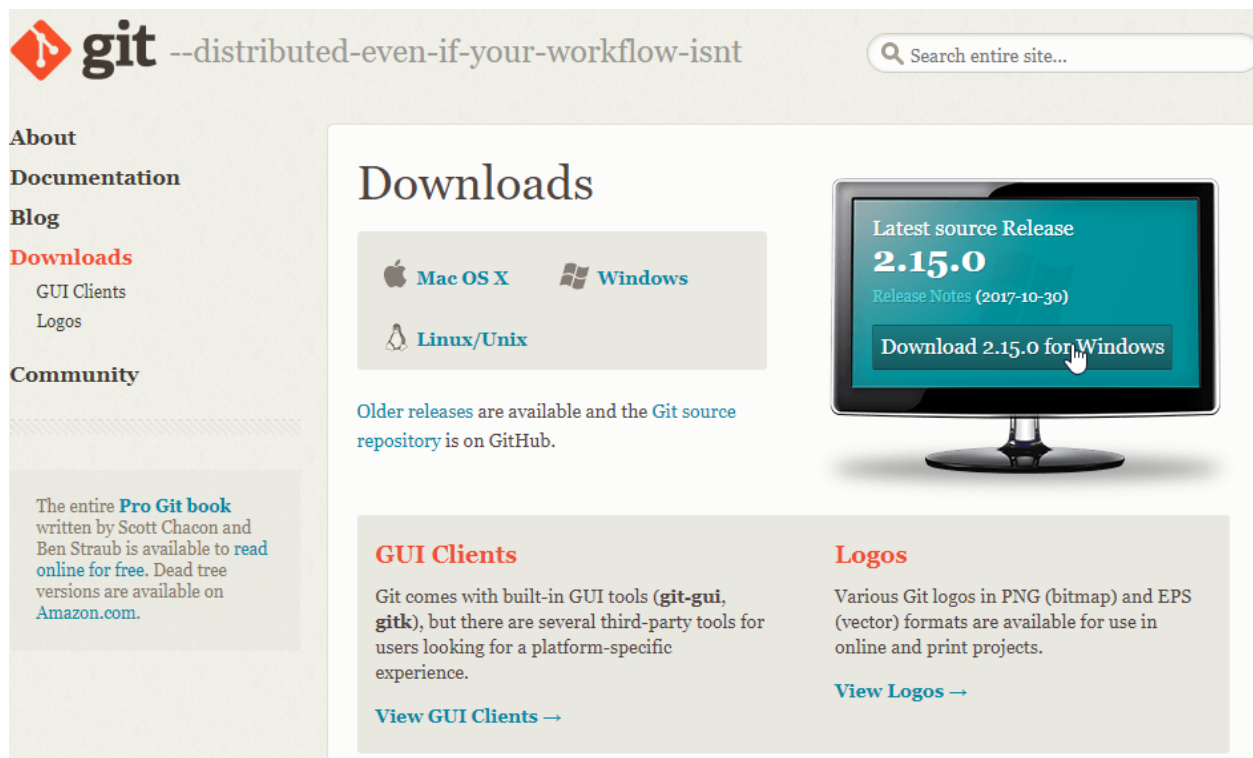
### ■ Específicos

- Dar a conocer la existencia de una herramienta de versionamiento muy potente

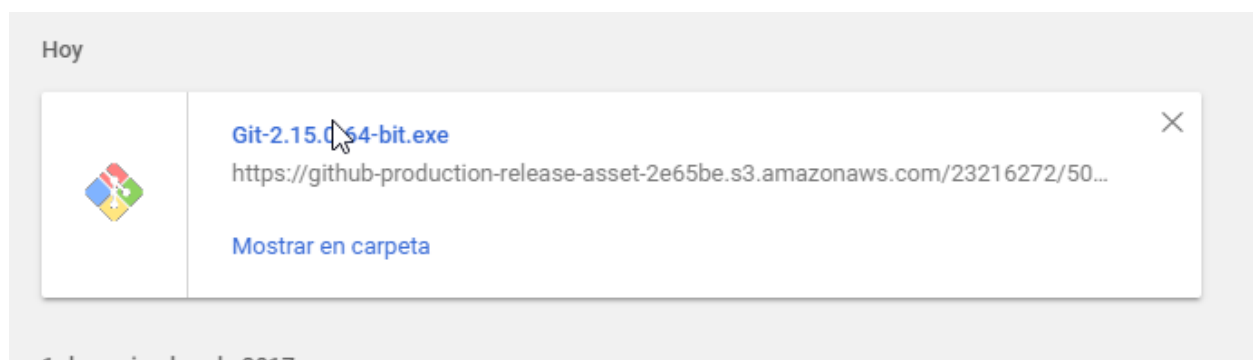


## II. INSTALACION

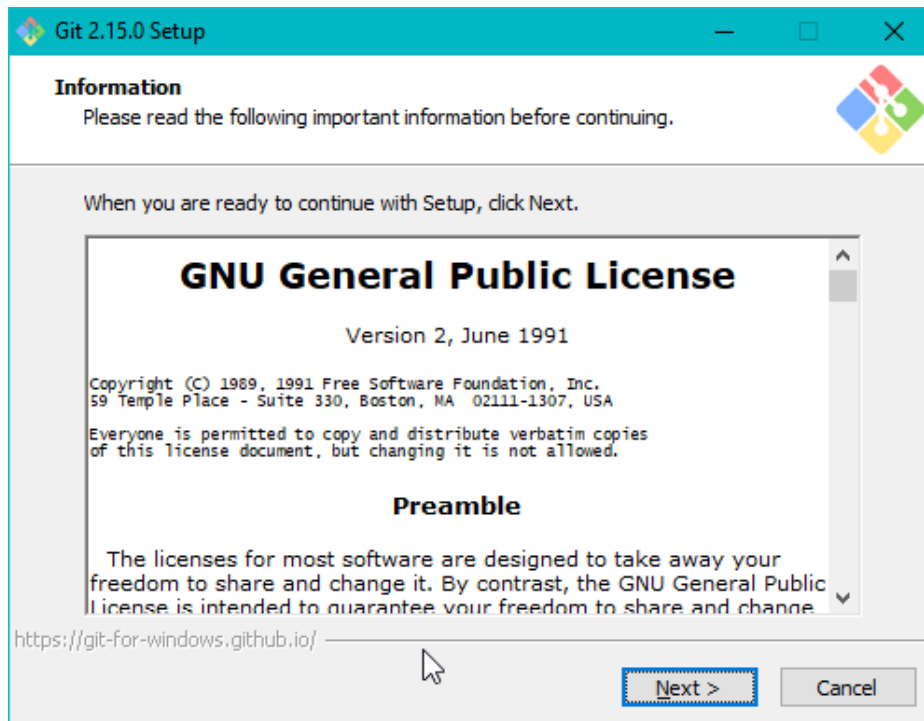
- Ir a <https://git-scm.com/downloads> y dar clic en Download 2.XX.X for Windows



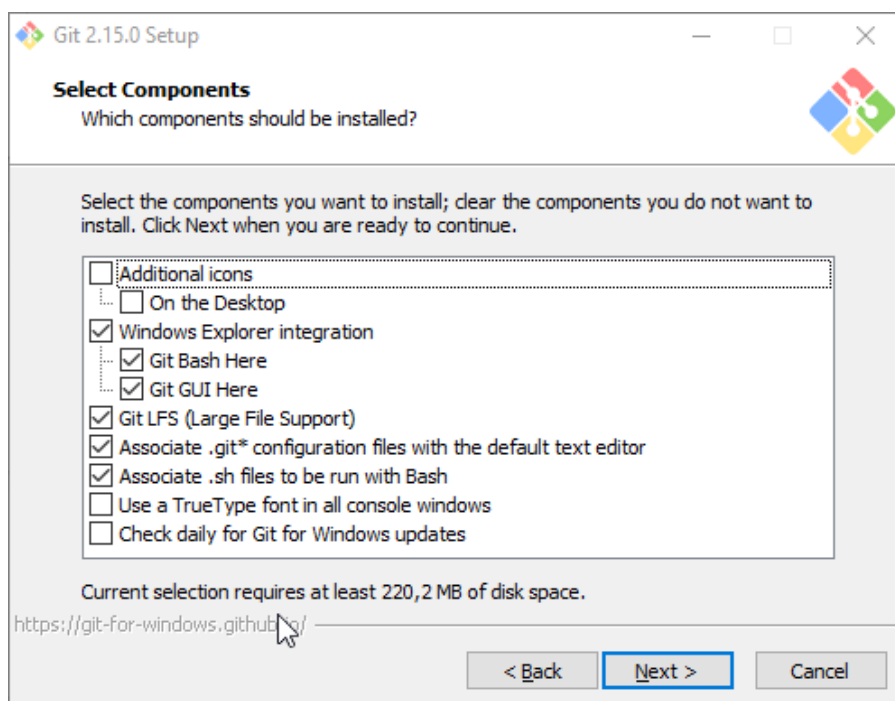
- Cuando la descarga finalice la ejecutamos



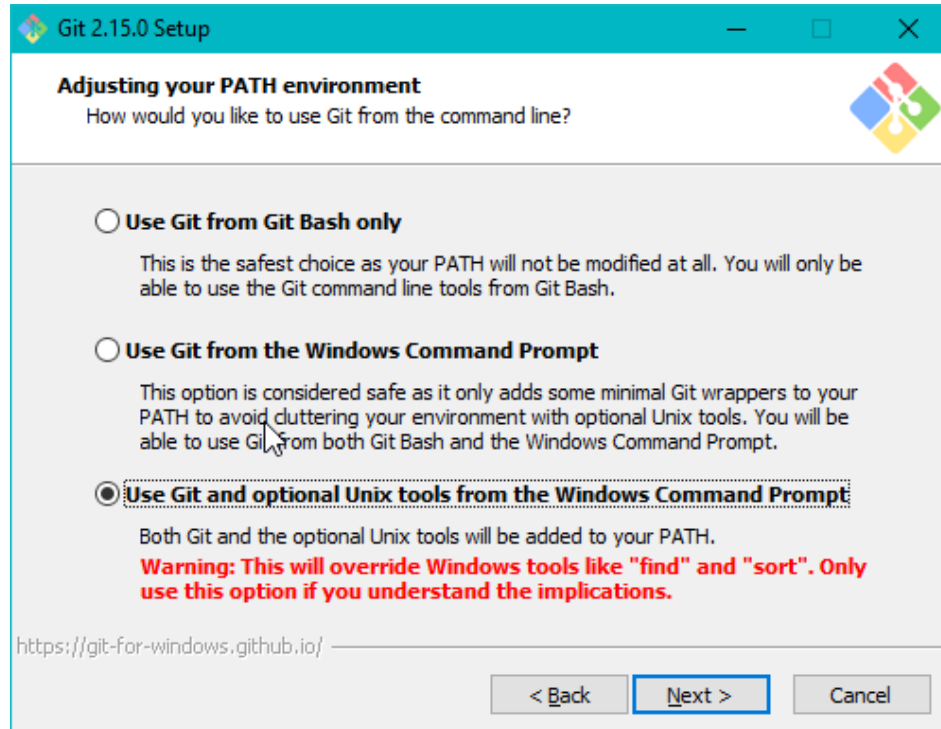
- Dar clic en Next



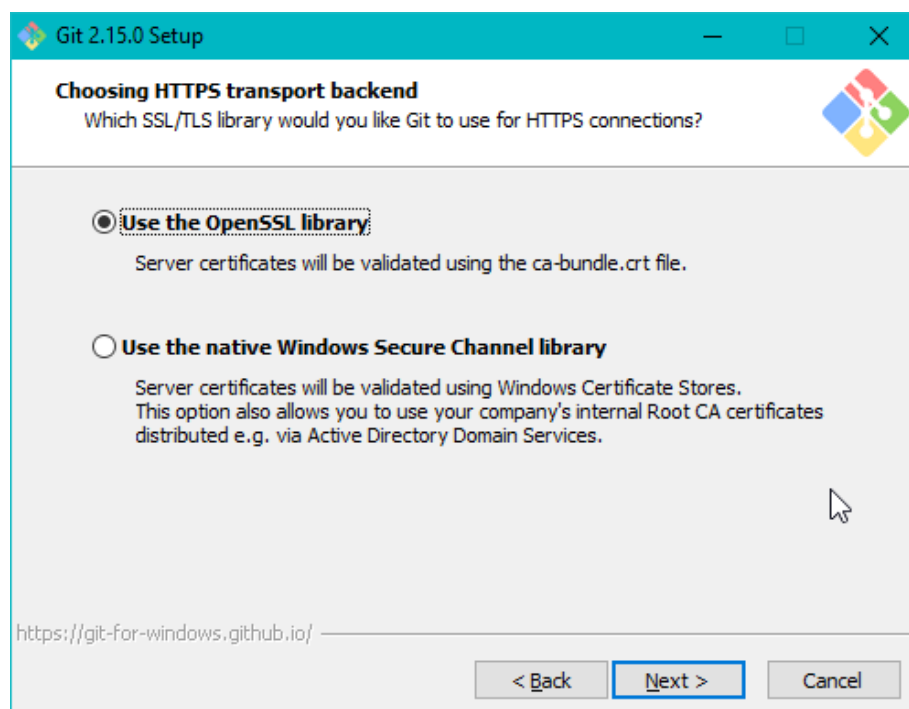
- Dar clic en Next



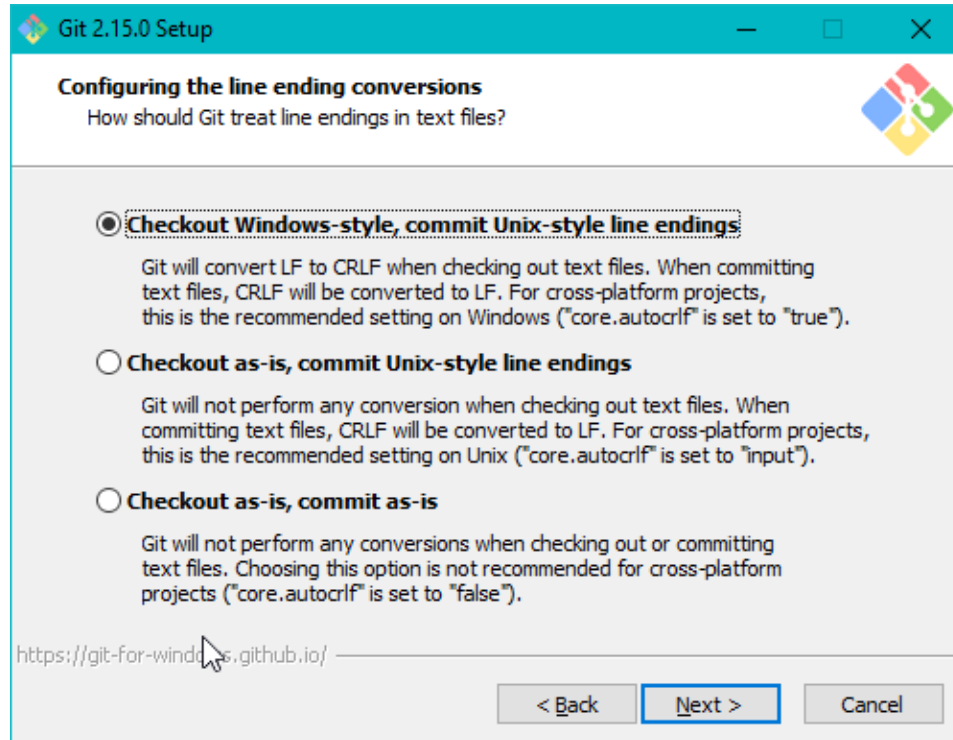
- Seleccionar Use Git and optional Unix tools from the Windows Command Prompt y dar clic en Next



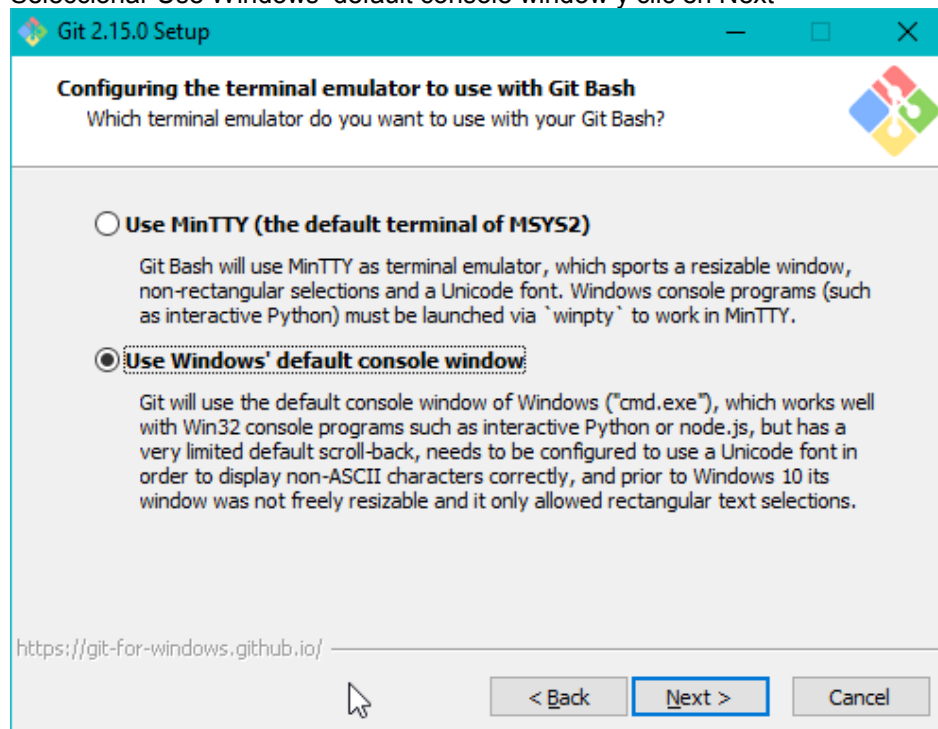
- Seleccionar Use the OpenSSL library y dar clic en next



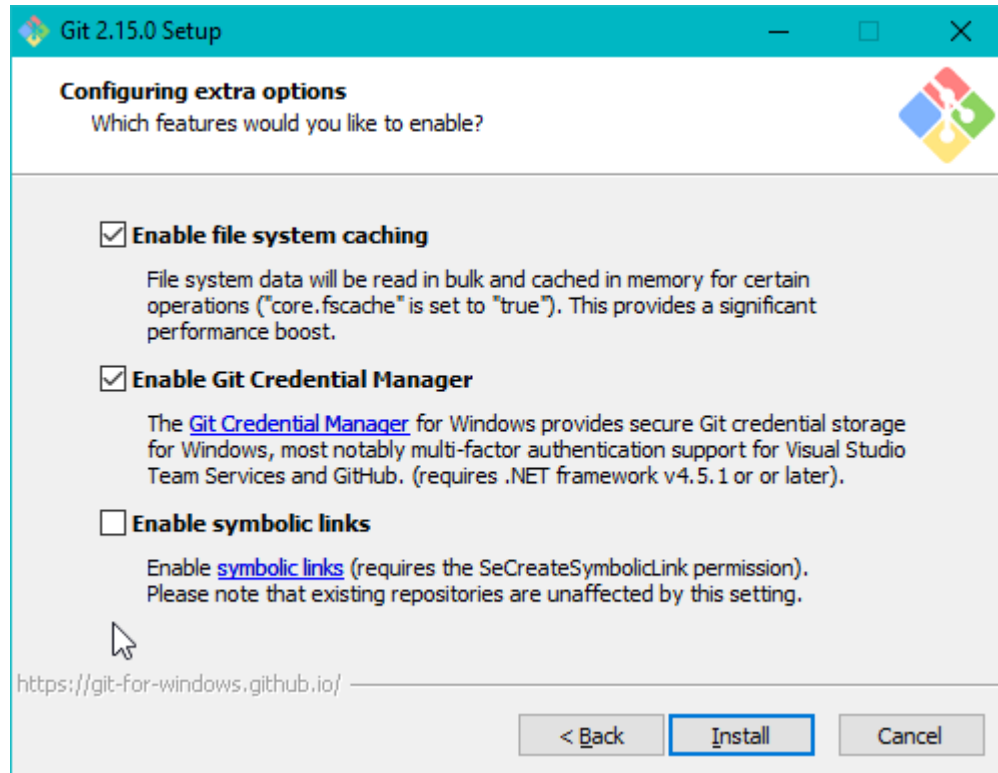
- Seleccionar Checkout Windows-style, commit Unix-style line endings y dar clic en Next



- Seleccionar Use Windows' default console window y clic en Next



- Dar clic en Install

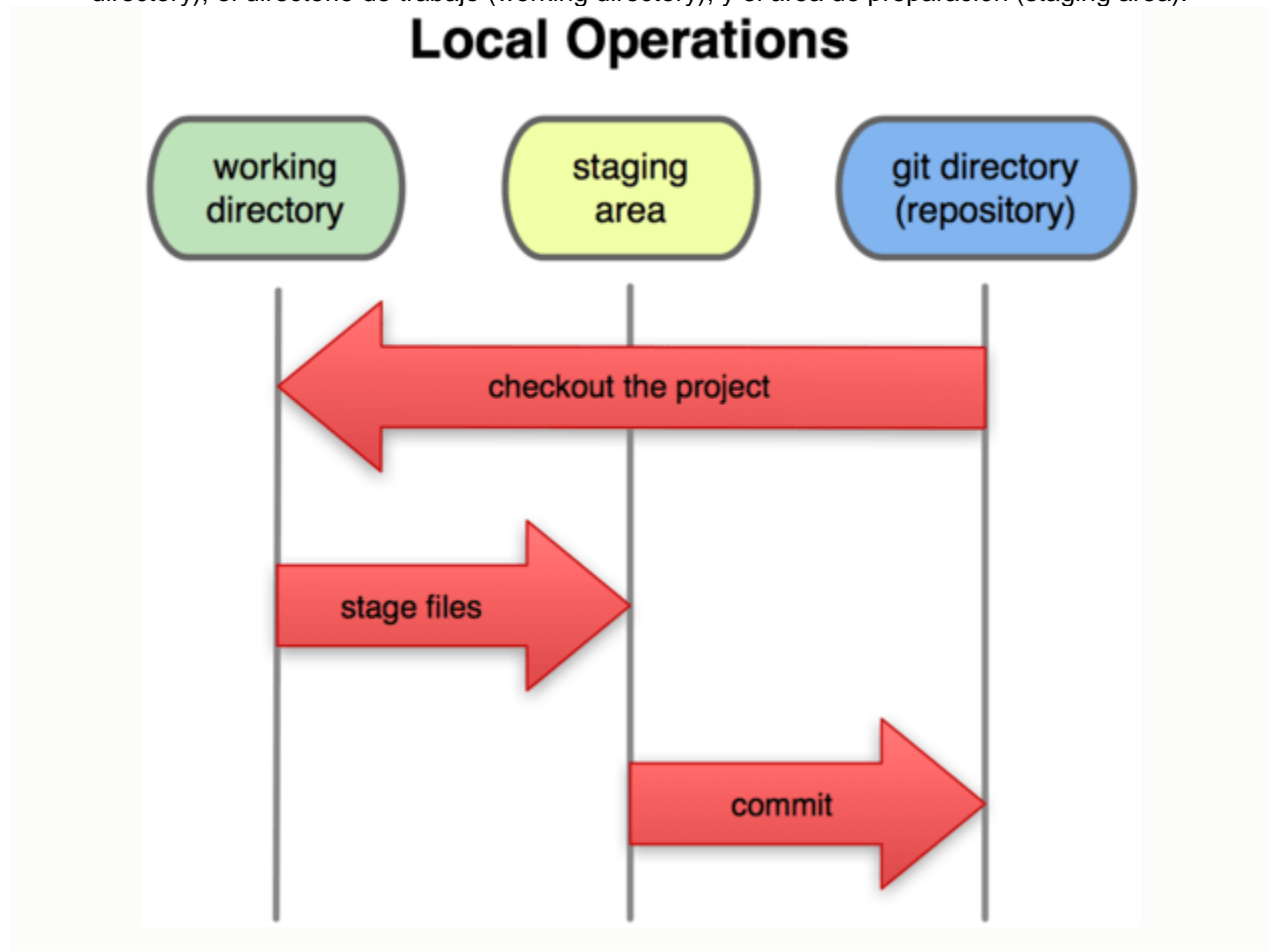


### III. GENERALIDADES

#### 3.1 Entendiendo GIT

Git tiene tres estados principales en los que se pueden encontrar tus archivos: confirmado (committed), modificado (modified), y preparado (staged). Confirmado significa que los datos están almacenados de manera segura en tu base de datos local. Modificado significa que has modificado el archivo pero todavía no lo has confirmado a tu base de datos. Preparado significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

Esto nos lleva a las tres secciones principales de un proyecto de Git: el directorio de Git (Git directory), el directorio de trabajo (working directory), y el área de preparación (staging area).



El flujo de trabajo básico en Git es algo así:

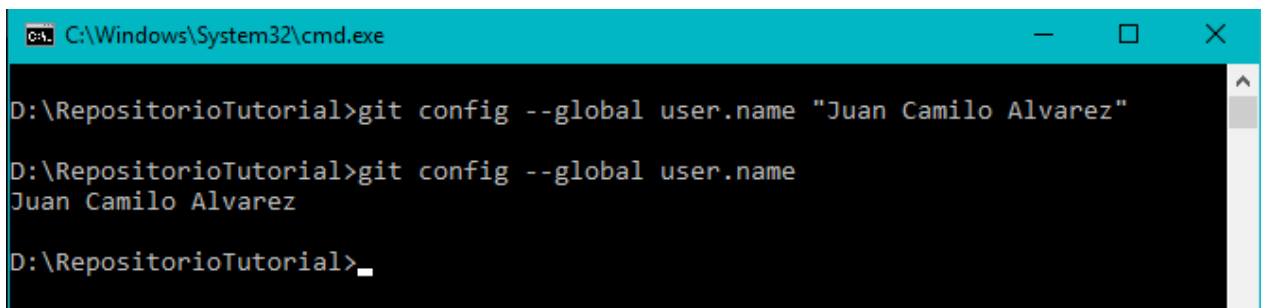
- Modificas una serie de archivos en tu directorio de trabajo.
- Preparas los archivos, añadiéndolos a tu área de preparación.
- Confirmas los cambios, lo que toma los archivos tal y como están en el área de preparación, y almacena esas instantáneas de manera permanente en tu directorio de Git.



Si una versión concreta de un archivo está en el directorio de Git, se considera confirmada (committed). Si ha sufrido cambios desde que se obtuvo del repositorio, pero ha sido añadida al área de preparación, está preparada (staged). Y si ha sufrido cambios desde que se obtuvo del repositorio, pero no se ha preparado, está modificada (modified)

### 3.2 Configurar opciones globales de Git

- Cambio de nombre ejecutar el comando `git config --global user.name "Nombre"`. Para ver el cambio realizado ejecutamos el mismo comando pero sin los caracteres finales que indican el nombre



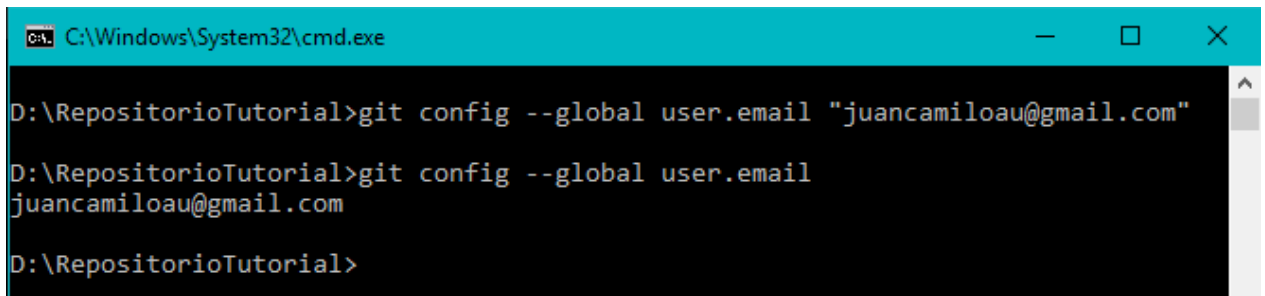
```
C:\Windows\System32\cmd.exe

D:\RepositorioTutorial>git config --global user.name "Juan Camilo Alvarez"

D:\RepositorioTutorial>git config --global user.name
Juan Camilo Alvarez

D:\RepositorioTutorial>
```

- Cambiando el correo, ejecutar `git config --global user.email Correo@correo.com`. Para ver el cambio realizado ejecutamos el mismo comando pero sin los caracteres finales que indican el correo



```
C:\Windows\System32\cmd.exe

D:\RepositorioTutorial>git config --global user.email "juancamiloau@gmail.com"

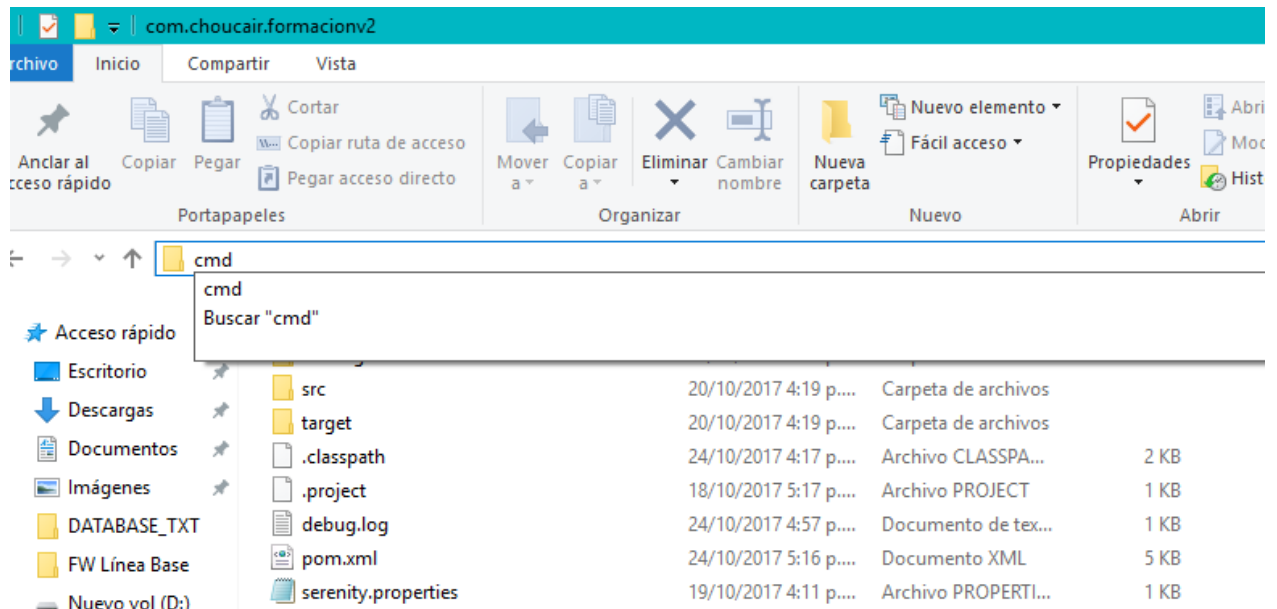
D:\RepositorioTutorial>git config --global user.email
juancamiloau@gmail.com

D:\RepositorioTutorial>
```

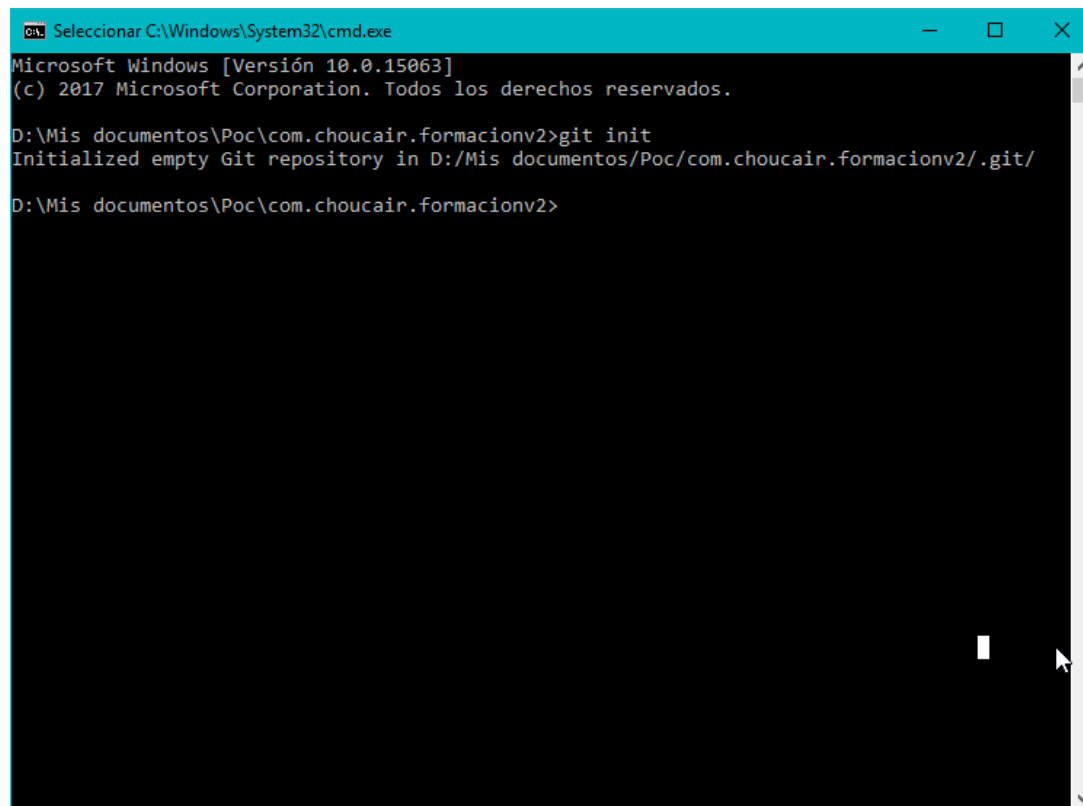
### 3.3 Creación de Repositorio Local

- En la ruta del proyecto en el explorador de archivos escribir CMD





- Escribir el comando Git init.



### 3.4 Guardando cambios en el repositorio

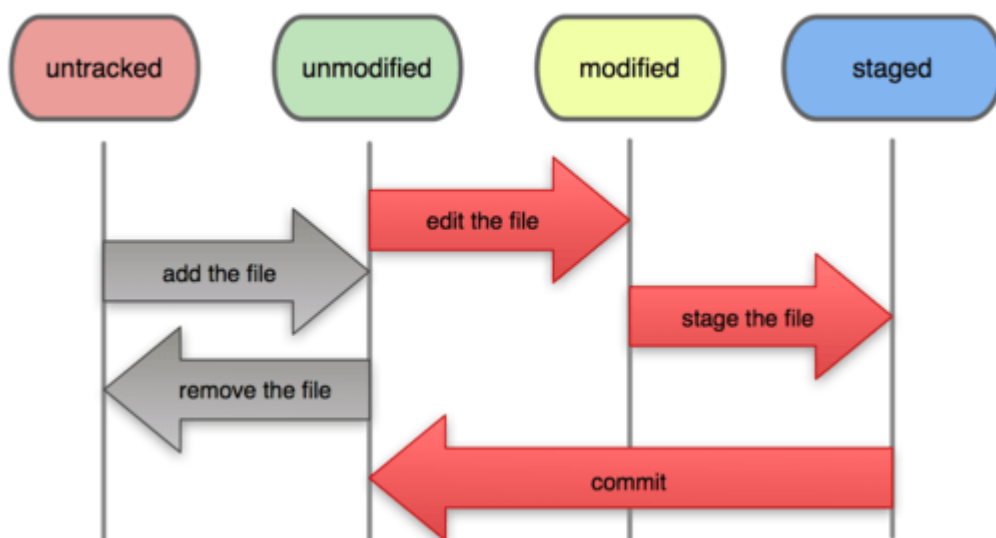
Debemos agregar el proyecto al estado seguimiento (Tracked) para después poder documentar el cambio.

Tienes un repositorio Git completo, y una copia de trabajo de los archivos de ese proyecto. Necesitas hacer algunos cambios, y confirmar instantáneas de esos cambios a tu repositorio cada vez que el proyecto alcance un estado que desees grabar.

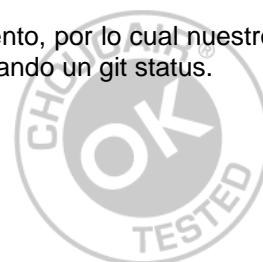
Recuerda que cada archivo de tu directorio de trabajo puede estar en uno de estos dos estados: bajo seguimiento (tracked), o sin seguimiento (untracked). Los archivos bajo seguimiento son aquellos que existían en la última instantánea; pueden estar sin modificaciones, modificados, o preparados. Los archivos sin seguimiento son todos los demás —cualquier archivo de tu directorio que no estuviese en tu última instantánea ni está en tu área de preparación—. La primera vez que clonas un repositorio, todos tus archivos estarán bajo seguimiento y sin modificaciones, ya que los acabas de copiar y no has modificado nada.

A medida que editas archivos, Git los ve como modificados, porque los has cambiado desde tu última confirmación. Preparas estos archivos modificados y luego confirmas todos los cambios que hayas preparado, y el ciclo se repite.

#### File Status Lifecycle



- En este momento nuestro proyecto está en el estado sin seguimiento, por lo cual nuestro repositorio git nos dice que no hay ningún cambio realizado ejecutando un git status.



```
C:\> Seleccionar C:\Windows\System32\cmd.exe

D:\Mis documentos\Poc\com.choucair.formacionv2>git status
On branch master

No commits yet

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

    .classpath
    .project
    .settings/
    debug.log
    pom.xml
    serenity.properties
    src/
    target/

nothing added to commit but untracked files present (use "git add" to track)
D:\Mis documentos\Poc\com.choucair.formacionv2>
```

- Para pasar un fichero al estado Staged debemos ejecutar el comando git add NombreArchivo.Extension y luego se observa con el comando git status que se ha agregado un archivo al estado Staged y nos muestra los que aún no han sido agregados

```
C:\> Seleccionar C:\Windows\System32\cmd.exe

D:\Mis documentos\Poc\com.choucair.formacionv2>git add pom.xml

D:\Mis documentos\Poc\com.choucair.formacionv2>git status
On branch master

No commits yet

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

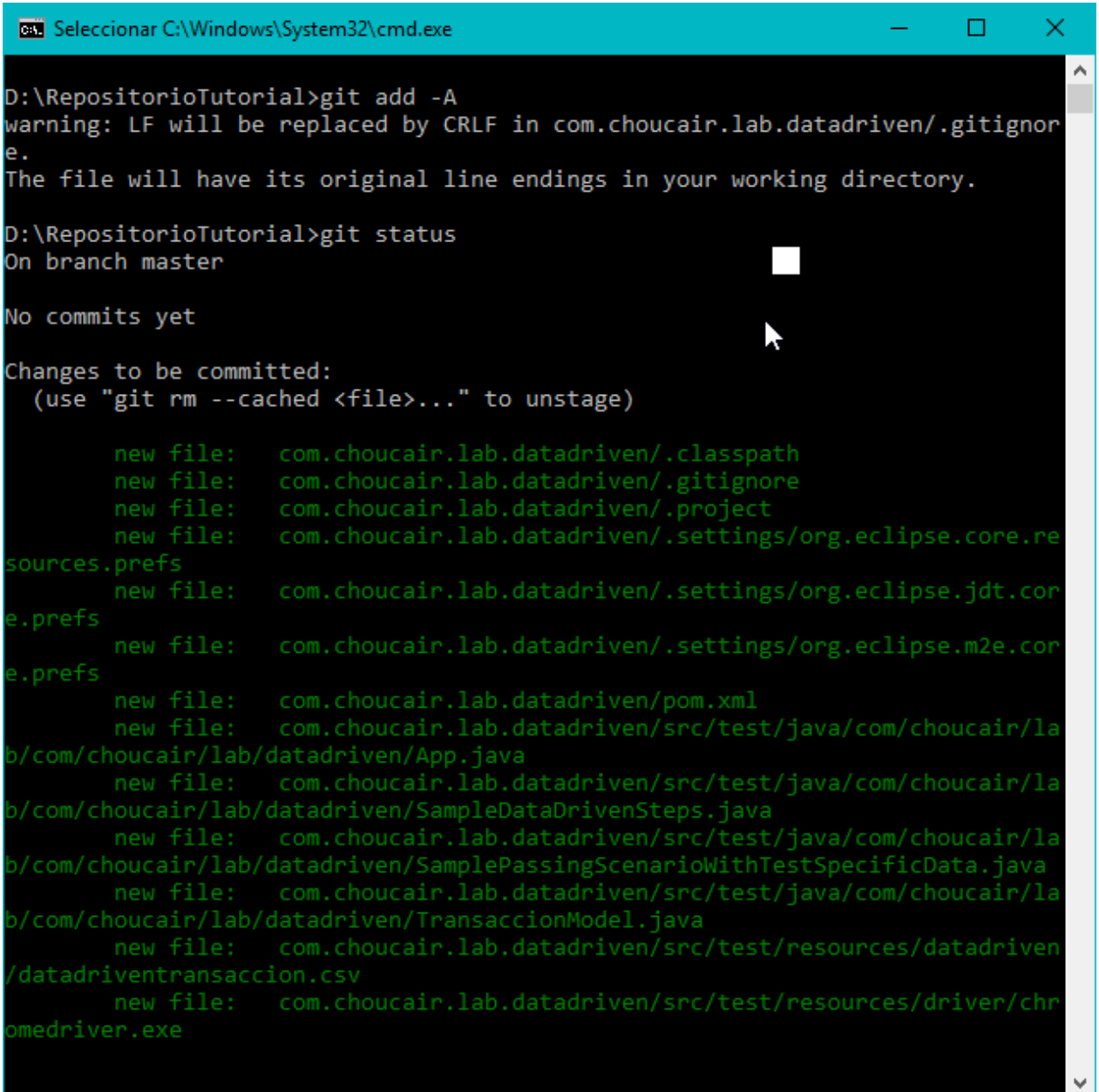
    new file:   pom.xml

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

    .classpath
    .project
    .settings/
    debug.log
    serenity.properties
    src/
    target/

D:\Mis documentos\Poc\com.choucair.formacionv2>
```

- Esta acción se puede realizar fichero x fichero o se puede ejecutar un comando git add -A (Donde la A significa all files), y se verifica con un git status.
- 



```
Seleccionar C:\Windows\System32\cmd.exe

D:\RepositorioTutorial>git add -A
warning: LF will be replaced by CRLF in com.choucair.lab.datadriven/.gitignore.
The file will have its original line endings in your working directory.

D:\RepositorioTutorial>git status
On branch master

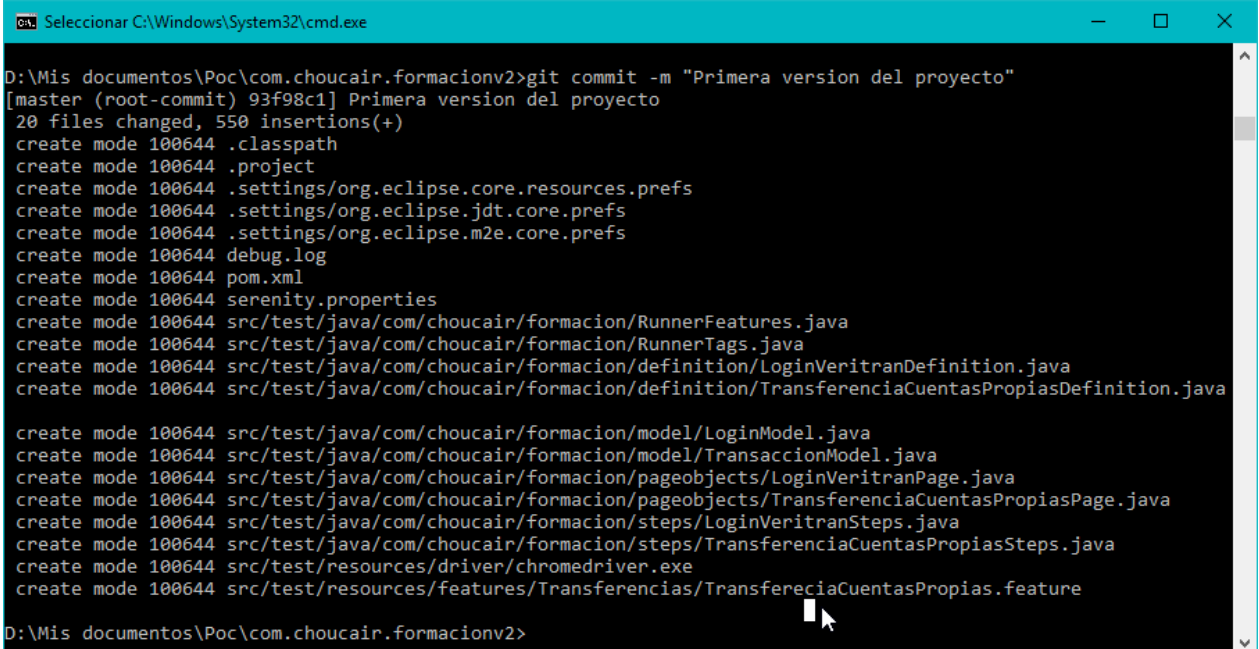
No commits yet

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

        new file:   com.choucair.lab.datadriven/.classpath
        new file:   com.choucair.lab.datadriven/.gitignore
        new file:   com.choucair.lab.datadriven/.project
        new file:   com.choucair.lab.datadriven/.settings/org.eclipse.core.re
sources.prefs
        new file:   com.choucair.lab.datadriven/.settings/org.eclipse.jdt.cor
e.prefs
        new file:   com.choucair.lab.datadriven/.settings/org.eclipse.m2e.cor
e.prefs
        new file:   com.choucair.lab.datadriven/pom.xml
        new file:   com.choucair.lab.datadriven/src/test/java/com/choucair/la
b/com/choucair/lab/datadriven/App.java
        new file:   com.choucair.lab.datadriven/src/test/java/com/choucair/la
b/com/choucair/lab/datadriven/SampleDataDrivenSteps.java
        new file:   com.choucair.lab.datadriven/src/test/java/com/choucair/la
b/com/choucair/lab/datadriven/SamplePassingScenarioWithTestSpecificData.java
        new file:   com.choucair.lab.datadriven/src/test/java/com/choucair/la
b/com/choucair/lab/datadriven/TransaccionModel.java
        new file:   com.choucair.lab.datadriven/src/test/resources/datadriven
/datadriventransaccion.csv
        new file:   com.choucair.lab.datadriven/src/test/resources/driver/chr
omedriver.exe
```

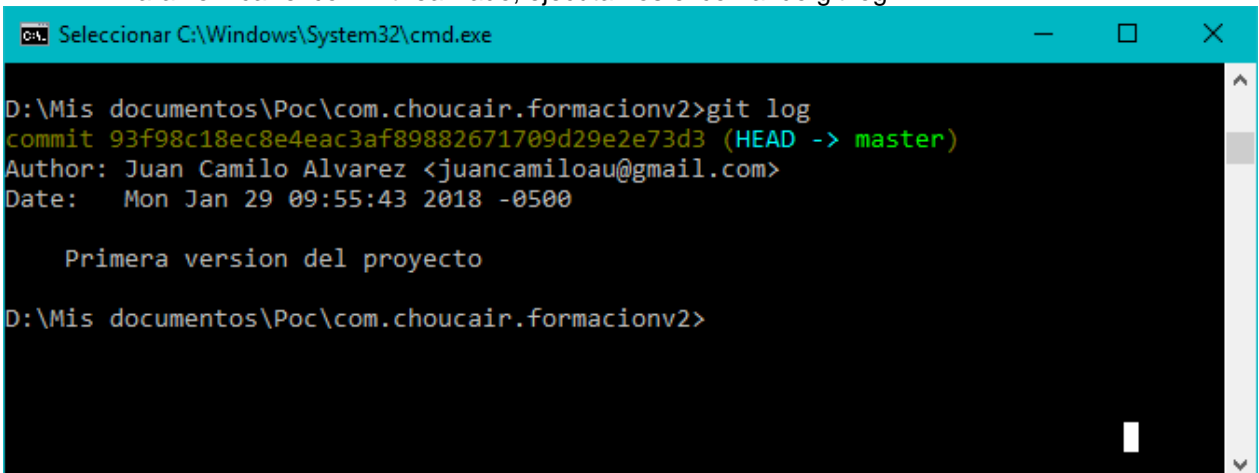
Nota: Git aún no reconoce las tildes, por ende omitir dichos acentos en los mensajes que se agreguen

- Para agregar todos los ficheros en estado Staged se debe ejecutar el comando git commit -m "Mensaje de commit o cambio"



```
C:\Windows\System32\cmd.exe
D:\Mis documentos\Poc\com.choucair.formacionv2>git commit -m "Primera version del proyecto"
[master (root-commit) 93f98c1] Primera version del proyecto
20 files changed, 550 insertions(+)
create mode 100644 .classpath
create mode 100644 .project
create mode 100644 .settings/org.eclipse.core.resources.prefs
create mode 100644 .settings/org.eclipse.jdt.core.prefs
create mode 100644 .settings/org.eclipse.m2e.core.prefs
create mode 100644 debug.log
create mode 100644 pom.xml
create mode 100644 serenity.properties
create mode 100644 src/test/java/com/choucair/formacion/RunnerFeatures.java
create mode 100644 src/test/java/com/choucair/formacion/RunnerTags.java
create mode 100644 src/test/java/com/choucair/formacion/definition/LoginVeritranDefinition.java
create mode 100644 src/test/java/com/choucair/formacion/definition/TransferenciaCuentasPropiasDefinition.java
create mode 100644 src/test/java/com/choucair/formacion/model/LoginModel.java
create mode 100644 src/test/java/com/choucair/formacion/model/TransaccionModel.java
create mode 100644 src/test/java/com/choucair/formacion/pageobjects/LoginVeritranPage.java
create mode 100644 src/test/java/com/choucair/formacion/pageobjects/TransferenciaCuentasPropiasPage.java
create mode 100644 src/test/java/com/choucair/formacion/steps/LoginVeritranSteps.java
create mode 100644 src/test/java/com/choucair/formacion/steps/TransferenciaCuentasPropiasSteps.java
create mode 100644 src/test/resources/driver/chromedriver.exe
create mode 100644 src/test/resources/features/Transferencias/TransferenciaCuentasPropias.feature
D:\Mis documentos\Poc\com.choucair.formacionv2>
```

- Para verificar el commit realizado, ejecutamos el comando git log

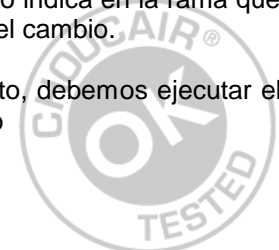


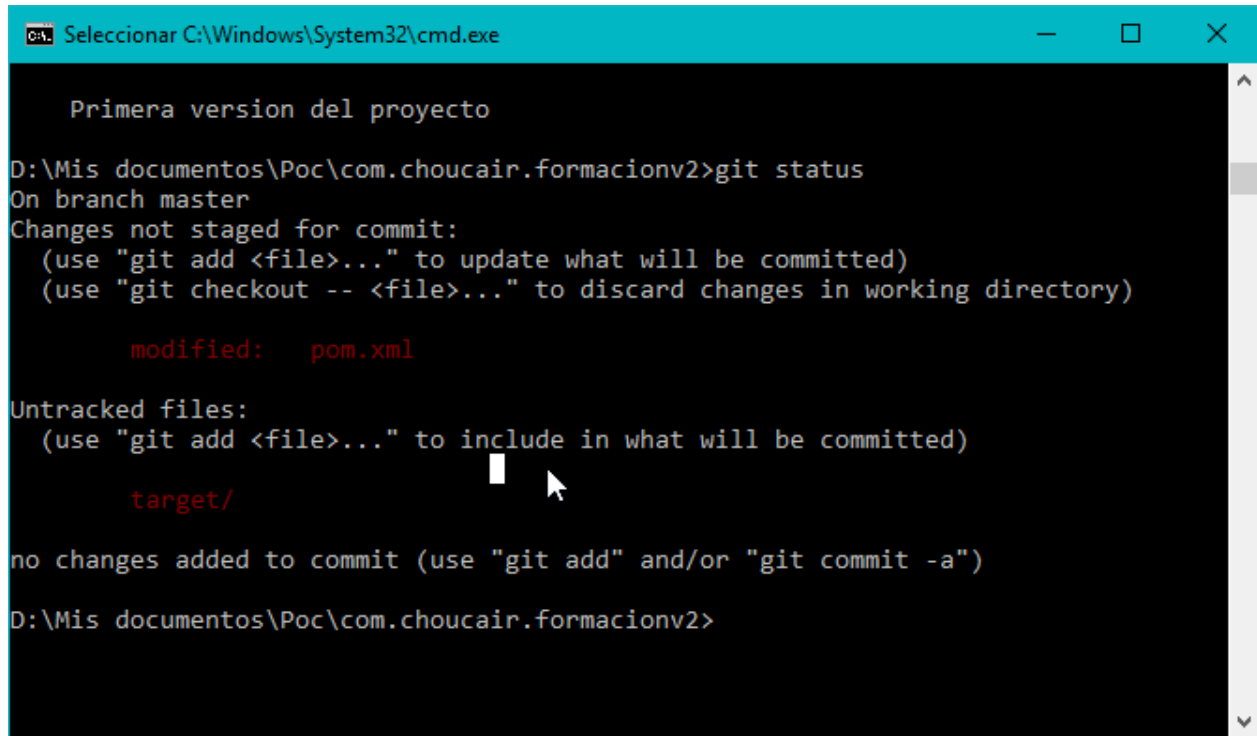
```
C:\Windows\System32\cmd.exe
D:\Mis documentos\Poc\com.choucair.formacionv2>git log
commit 93f98c18ec8e4eac3af89882671709d29e2e73d3 (HEAD -> master)
Author: Juan Camilo Alvarez <juancamiloau@gmail.com>
Date: Mon Jan 29 09:55:43 2018 -0500

    Primera version del proyecto
D:\Mis documentos\Poc\com.choucair.formacionv2>
```

Git le asigna a cada commit un código hash SHA-1, este código se utiliza para identificar cada commit y luego hacer regresiones o retornar los cambios anteriores, adicional a esto indica en la rama que estamos parados (HEAD -> master), agrega el autor y la fecha específica del cambio.

- Cuando se realiza un cambio en alguna clase de nuestro proyecto, debemos ejecutar el comando git status para verificar que git haya detectado el cambio





```
Seleccionar C:\Windows\System32\cmd.exe

Primera version del proyecto

D:\Mis documentos\Poc\com.choucair.formacionv2>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)

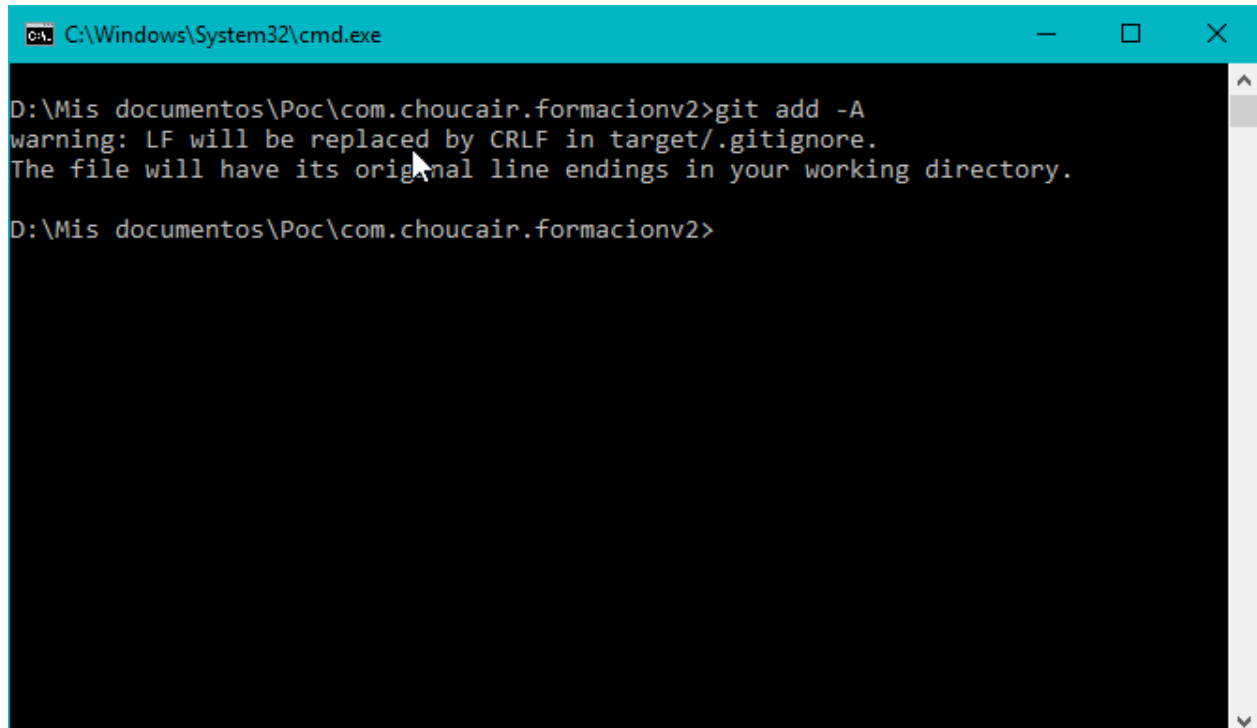
        modified:   pom.xml

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

        target/

no changes added to commit (use "git add" and/or "git commit -a")
D:\Mis documentos\Poc\com.choucair.formacionv2>
```

- Agregamos el cambio a estado Staged con git add -A para poder guardar el cambio en nuestro versionamiento

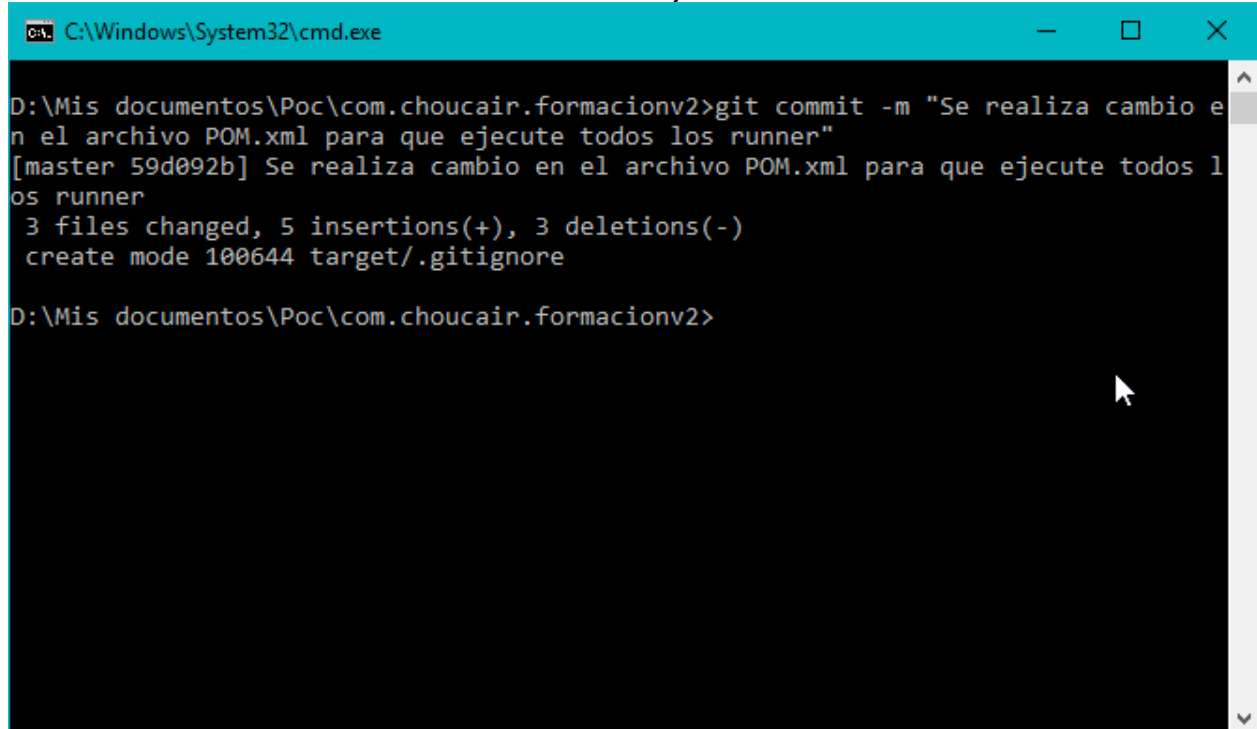


```
C:\Windows\System32\cmd.exe

D:\Mis documentos\Poc\com.choucair.formacionv2>git add -A
warning: LF will be replaced by CRLF in target/.gitignore.
The file will have its original line endings in your working directory.

D:\Mis documentos\Poc\com.choucair.formacionv2>
```

- Se realiza el commit con nuestro mensaje

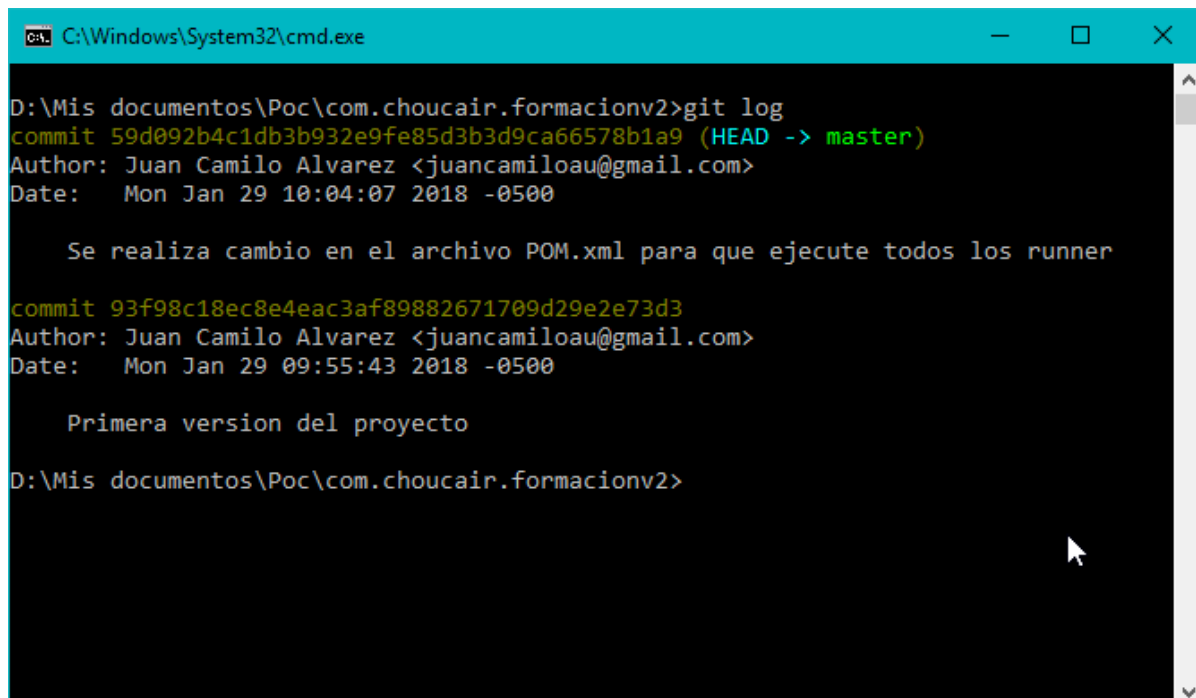


```
C:\Windows\System32\cmd.exe

D:\Mis documentos\Poc\com.choucair.formacionv2>git commit -m "Se realiza cambio en el archivo POM.xml para que ejecute todos los runner"
[master 59d092b] Se realiza cambio en el archivo POM.xml para que ejecute todos los runner
 3 files changed, 5 insertions(+), 3 deletions(-)
 create mode 100644 target/.gitignore

D:\Mis documentos\Poc\com.choucair.formacionv2>
```

- Verificar el cambio realizado con el comando git log



```
C:\Windows\System32\cmd.exe

D:\Mis documentos\Poc\com.choucair.formacionv2>git log
commit 59d092b4c1db3b932e9fe85d3b3d9ca66578b1a9 (HEAD -> master)
Author: Juan Camilo Alvarez <juancamiloau@gmail.com>
Date:   Mon Jan 29 10:04:07 2018 -0500

    Se realiza cambio en el archivo POM.xml para que ejecute todos los runner

commit 93f98c18ec8e4eac3af89882671709d29e2e73d3
Author: Juan Camilo Alvarez <juancamiloau@gmail.com>
Date:   Mon Jan 29 09:55:43 2018 -0500

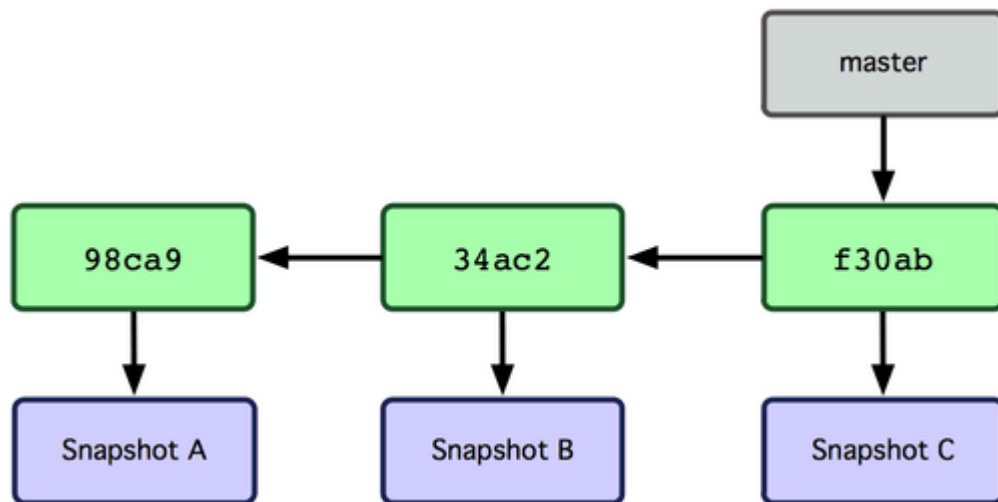
    Primera version del proyecto

D:\Mis documentos\Poc\com.choucair.formacionv2>
```

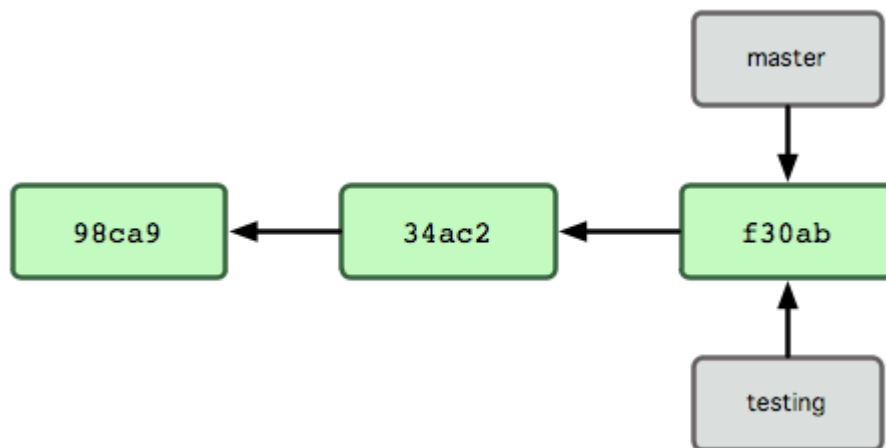


### 3.5 Ramificación y fusión

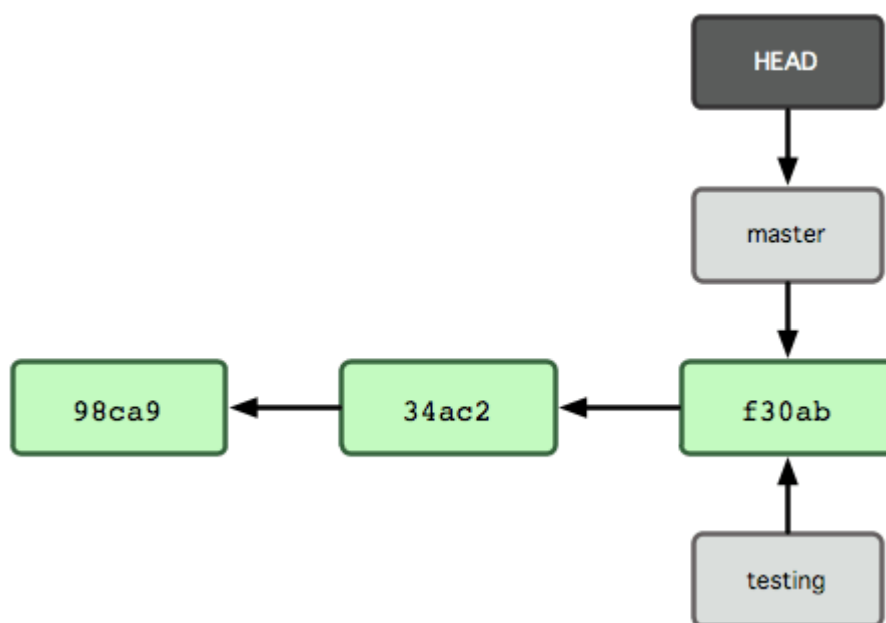
Una rama Git es simplemente un apuntador móvil apuntando a una de esas confirmaciones. La rama por defecto de Git es la rama **master**. Con la primera confirmación de cambios que realicemos, se creará esta rama principal master apuntando a dicha confirmación. En cada confirmación de cambios que realicemos, la rama irá avanzando automáticamente. Y la rama master apuntará siempre a la última confirmación realizada.



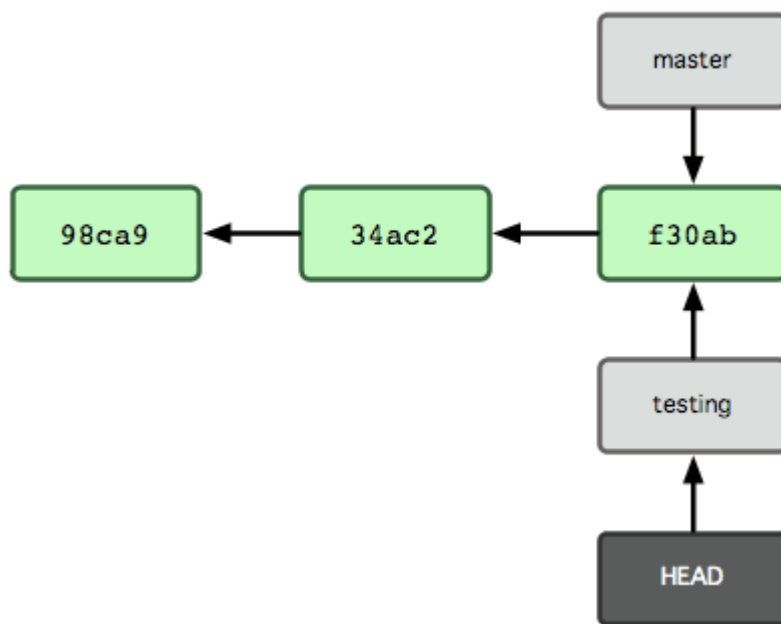
¿Qué sucede cuando creas una nueva rama? Simplemente se crea un nuevo apuntador para que lo puedas mover libremente. Por ejemplo, si quieres crear una nueva rama denominada "testing". Usarás el comando `git branch testing`



Y, ¿cómo sabe Git en qué rama estás en este momento? Pues..., mediante un apuntador especial denominado HEAD, es simplemente el apuntador a la rama local en la que tú estés en ese momento. En este caso, en la rama master. Puesto que el comando git branch solamente crea una nueva rama, y no salta a dicha rama.



Para saltar de una rama a otra, tienes que utilizar el comando git checkout testing. Esto mueve el apuntador HEAD a la rama testing



Herramienta GIT

&lt;&lt;FW ABC&gt;&gt;

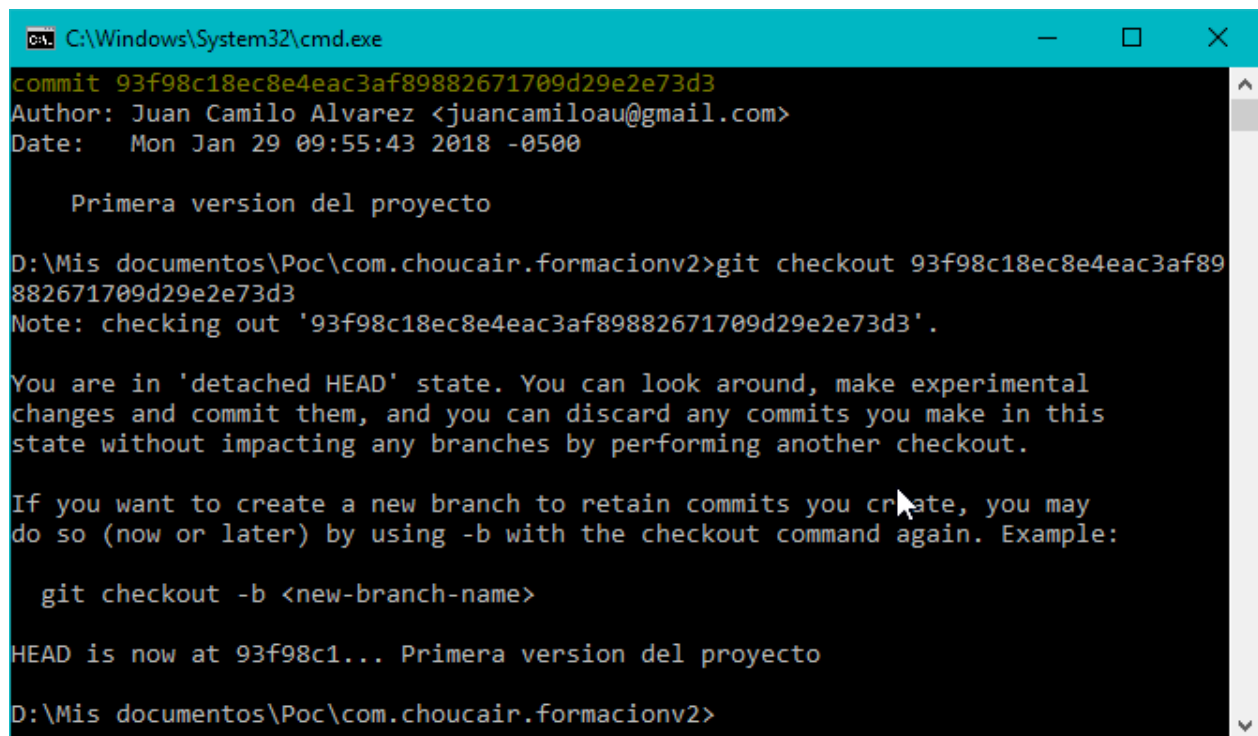
El comando checkout no solo te permite brincar a otras ramas, también te permite ubicar el HEAD en algún commit utilizando el respectivo código hash SHA-1 y poder generar un branch desde ese commit. Es importante recordar que Git revierte la carpeta de trabajo exactamente al estado en que estaba en la confirmación (commit) apuntada por la rama que activamos (checkout) en cada momento. Git añade, quita y modifica archivos automáticamente. Para asegurarte que tu copia de trabajo es exactamente tal y como era la rama en la última confirmación de cambios realizada sobre ella.

Veamos un ejemplo:

El código en el segundo commit

```
<includes>
  <include>**/Runner*.java</include>
</includes>
```

Ejecutó el comando git checkout + el código hash SHA-1 del commit de mi primera versión



```
C:\Windows\System32\cmd.exe

commit 93f98c18ec8e4eac3af89882671709d29e2e73d3
Author: Juan Camilo Alvarez <juancamiloau@gmail.com>
Date: Mon Jan 29 09:55:43 2018 -0500

Primera version del proyecto

D:\Mis documentos\Poc\com.choucair.formacionv2>git checkout 93f98c18ec8e4eac3af89882671709d29e2e73d3
Note: checking out '93f98c18ec8e4eac3af89882671709d29e2e73d3'.

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:

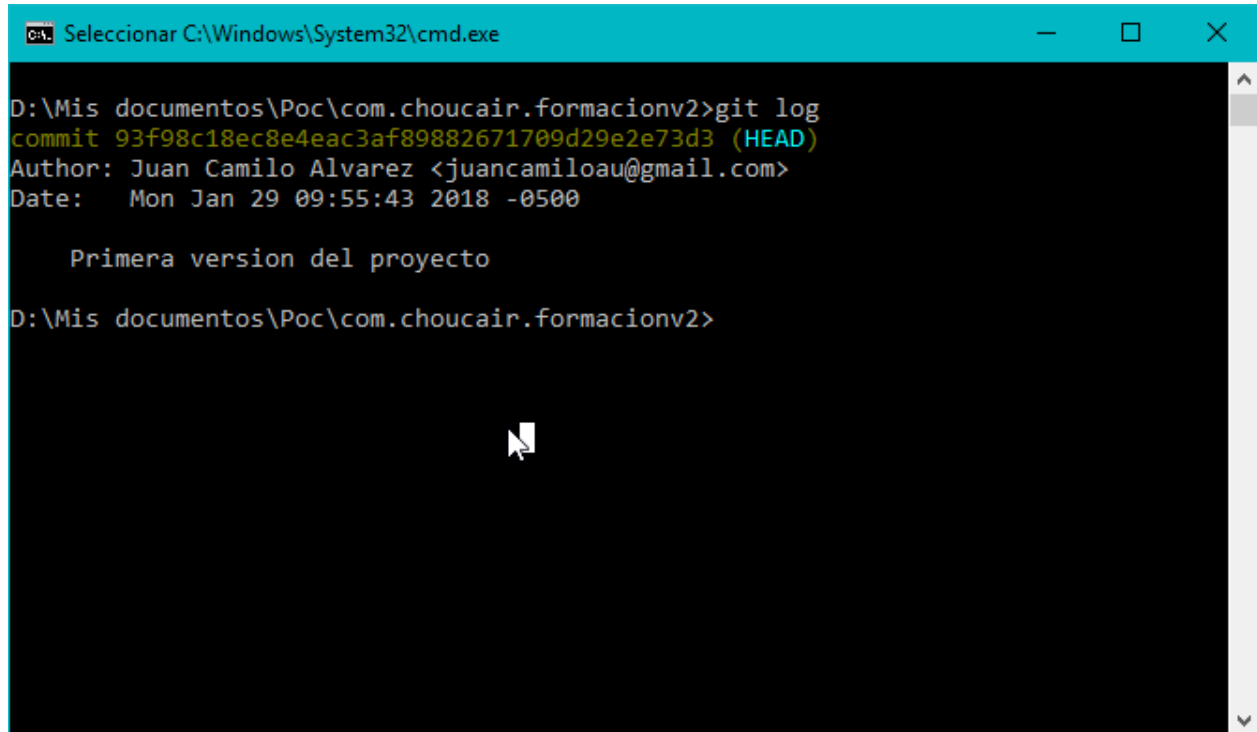
  git checkout -b <new-branch-name>

HEAD is now at 93f98c1... Primera version del proyecto
D:\Mis documentos\Poc\com.choucair.formacionv2>
```

Esto me actualiza automáticamente la línea de código que ejecutaba otros archivos java

```
<includes>
  <include>**/definitions/**/*Test.java</include>
</includes>
```

Al verificar el historial de commits con git log, validamos que ya solo nos aparece un commit, el de la primera versión en el cual estamos situados.



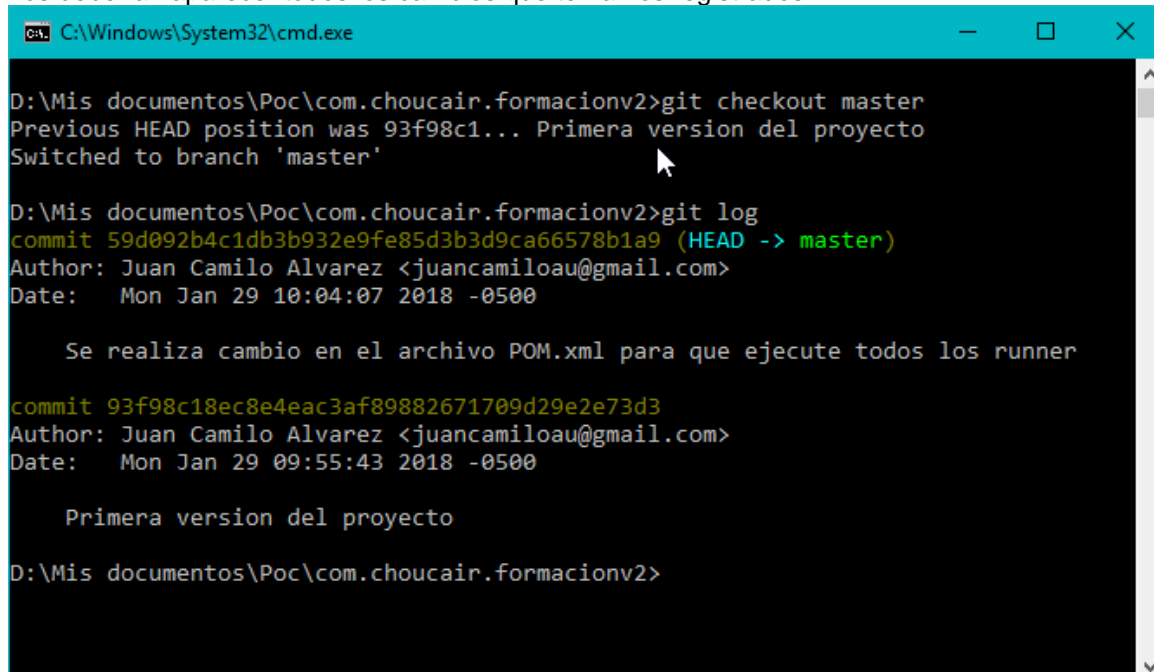
```
C:\Windows\System32\cmd.exe

D:\Mis documentos\Poc\com.choucair.formacionv2>git log
commit 93f98c18ec8e4eac3af89882671709d29e2e73d3 (HEAD)
Author: Juan Camilo Alvarez <juancamiloau@gmail.com>
Date: Mon Jan 29 09:55:43 2018 -0500

    Primera version del proyecto

D:\Mis documentos\Poc\com.choucair.formacionv2>
```

Para a nuestro último commit, escribimos git checkout master y verificamos nuestro historial de cambios ya nos deberían aparecer todos los cambios que teníamos registrados



```
C:\Windows\System32\cmd.exe

D:\Mis documentos\Poc\com.choucair.formacionv2>git checkout master
Previous HEAD position was 93f98c1... Primera version del proyecto
Switched to branch 'master'

D:\Mis documentos\Poc\com.choucair.formacionv2>git log
commit 59d092b4c1db3b932e9fe85d3b3d9ca66578b1a9 (HEAD -> master)
Author: Juan Camilo Alvarez <juancamiloau@gmail.com>
Date: Mon Jan 29 10:04:07 2018 -0500

    Se realiza cambio en el archivo POM.xml para que ejecute todos los runner

commit 93f98c18ec8e4eac3af89882671709d29e2e73d3
Author: Juan Camilo Alvarez <juancamiloau@gmail.com>
Date: Mon Jan 29 09:55:43 2018 -0500

    Primera version del proyecto

D:\Mis documentos\Poc\com.choucair.formacionv2>
```



- Creamos una nueva rama en nuestro head

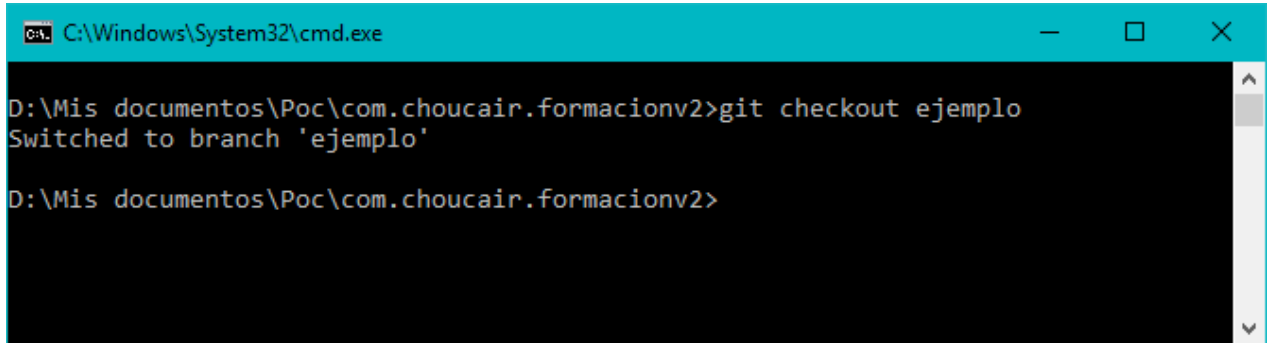
```
Seleccionar C:\Windows\System32\cmd.exe

D:\Mis documentos\Poc\com.choucair.formacionv2>git branch Ejemplo

D:\Mis documentos\Poc\com.choucair.formacionv2>
```



- Nos situamos en la rama que acabamos de crear con el comando git checkout



```
C:\Windows\System32\cmd.exe

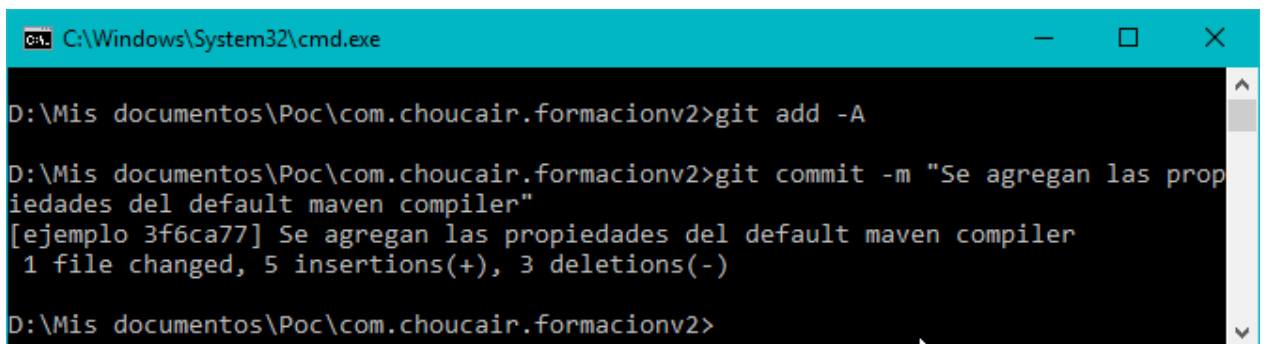
D:\Mis documentos\Poc\com.choucair.formacionv2>git checkout ejemplo
Switched to branch 'ejemplo'

D:\Mis documentos\Poc\com.choucair.formacionv2>
```

- Realizamos una modificación en nuestro código (Agregue las líneas <maven.compiler>

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <serenity.version>1.5.7</serenity.version> <!-- https://bintray.com
  <serenity.maven.version>1.5.7</serenity.maven.version>
  <serenity.cucumber.version>1.1.36</serenity.cucumber.version> <!--
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
</properties>
```

- Cambiamos el estado a modificado con git add -A y luego realizamos el commit



```
C:\Windows\System32\cmd.exe

D:\Mis documentos\Poc\com.choucair.formacionv2>git add -A

D:\Mis documentos\Poc\com.choucair.formacionv2>git commit -m "Se agregan las prop
iedades del default maven compiler"
[ejemplo 3f6ca77] Se agregan las propiedades del default maven compiler
1 file changed, 5 insertions(+), 3 deletions(-)

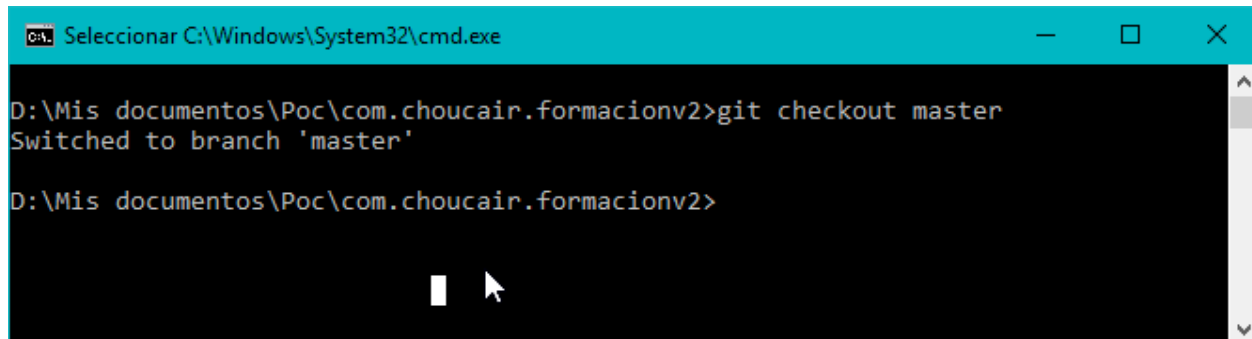
D:\Mis documentos\Poc\com.choucair.formacionv2>
```



## Herramienta GIT

&lt;&lt;FW ABC&gt;&gt;

- Nos situamos en la rama que va a absorber la otra rama, es decir, master va a añadir los cambios de la rama comentario.

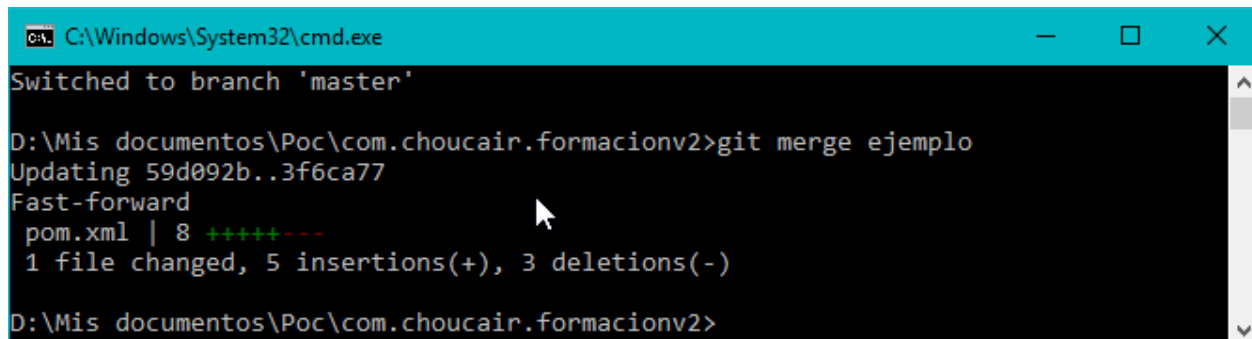


```
C:\Windows\System32\cmd.exe
D:\Mis documentos\Poc\com.choucair.formacionv2>git checkout master
Switched to branch 'master'
D:\Mis documentos\Poc\com.choucair.formacionv2>
```

- Verificamos el código y no debe de tener los cambios que agregamos en la rama comentario

```
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<serenity.version>1.5.7</serenity.version> <!-- https://bintray.com/serenity/maven/serenity-core/ -->
<serenity.maven.version>1.5.7</serenity.maven.version>
<serenity.cucumber.version>1.1.36</serenity.cucumber.version> <!-- https://bintray.com/serenity/maven/serenity-cu
</properties>
```

- Ahora ejecutamos el comando git merge + nombre de la rama para fusionar las ramas



```
C:\Windows\System32\cmd.exe
Switched to branch 'master'
D:\Mis documentos\Poc\com.choucair.formacionv2>git merge ejemplo
Updating 59d092b..3f6ca77
Fast-forward
 pom.xml | 8 +++++---
 1 file changed, 5 insertions(+), 3 deletions(-)
D:\Mis documentos\Poc\com.choucair.formacionv2>
```

- Al momento de verificar nuestro código nos damos cuenta que fusiono las dos ramas, ya que apareció la línea de comentario que estaba en la rama Comentario



- Para eliminar un branch que ya no vamos a utilizar, se usa el comando git branch -D + nombre de la rama

```
C:\Windows\System32\cmd.exe

D:\Mis documentos\Poc\com.choucair.formacionv2>git branch -D ejemplo
Deleted branch ejemplo (was 3f6ca77).

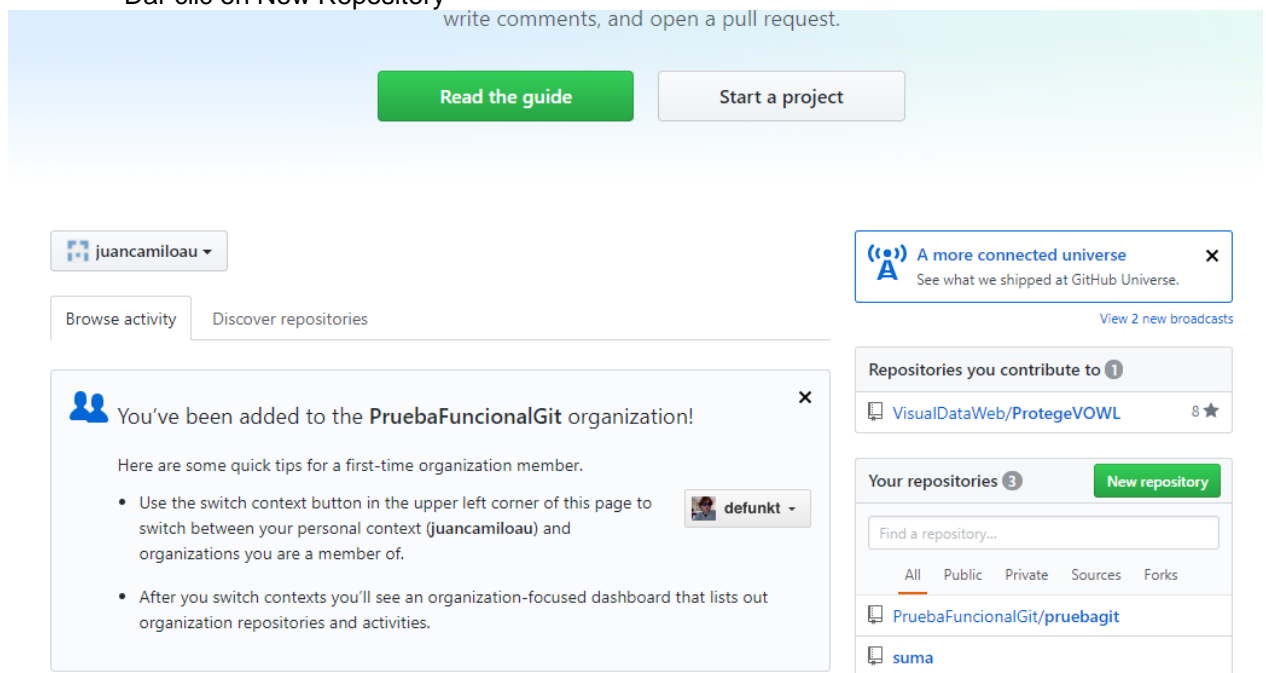
D:\Mis documentos\Poc\com.choucair.formacionv2>
```

#### IV. IMPLEMENTACION CON GITHUB

**Nota:** Esta implementación es solo una guía en caso de tener una implementación de un servidor de git, ya que las pruebas se realizarán con github que es un entorno público de código abierto.

##### Creación de repositorio en GitHub

- Registrarse en <http://www.github.com>
- Dar clic en New Repository






- Colocamos el nombre del repositorio y damos clic en Create Repository

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 juancamilou ▾

Repository name

/ tutorialgit ✓

Great repository names are short and memorable. Need inspiration? How about **didactic-winner**.

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾



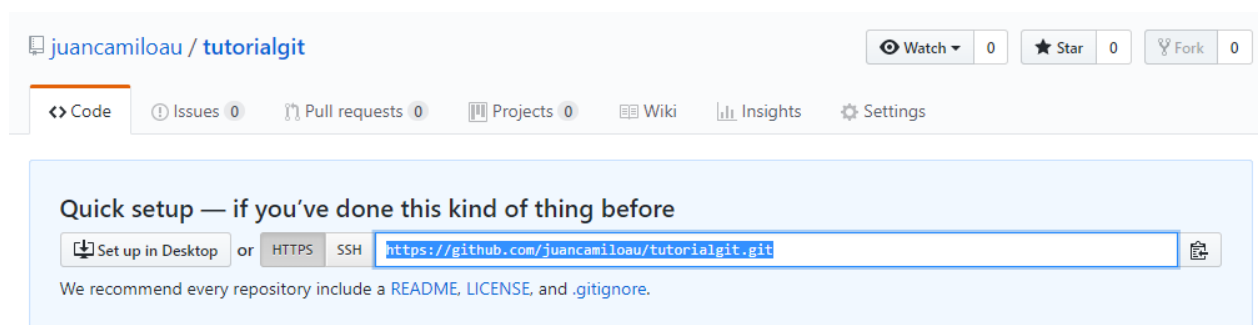
**Create repository**



- Debemos vincular nuestro proyecto local con nuestro proyecto de Github



- Para ello debemos copiar el link de nuestro repositorio de Github



## Herramienta GIT

<<FW ABC>>

- Ejecutar el comando git remote add origin + el link del proyecto github

```
C:\Windows\System32\cmd.exe

D:\RepositorioTutorial>git remote add origin https://github.com/juancamiloau/tutorialgit.git

D:\RepositorioTutorial>
```

- Después de vincular los repositorios debemos sincronizar el repositorio local con el de github, para eso utilizamos git push origin master

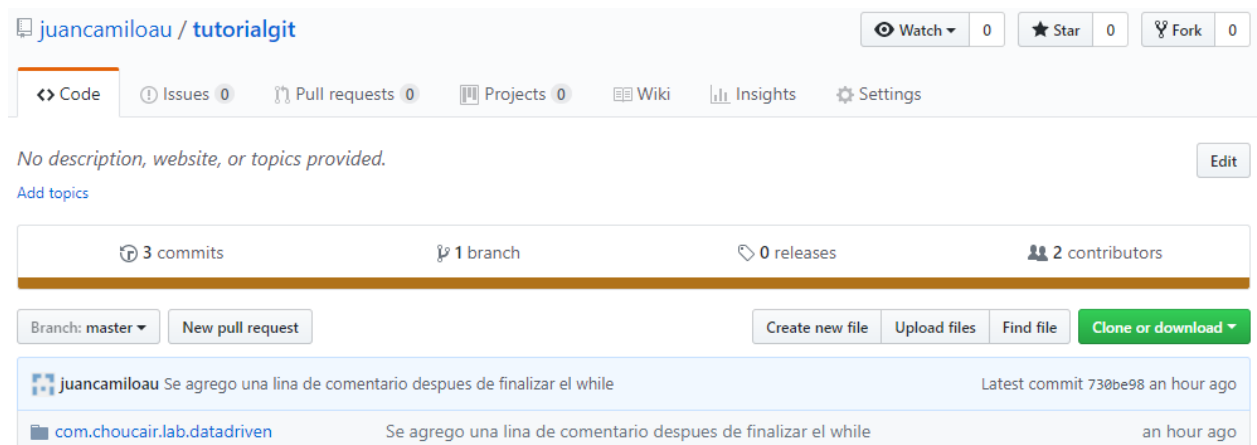
```
C:\Windows\System32\cmd.exe

D:\RepositorioTutorial>git push origin master
Counting objects: 58, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (31/31), done.
Writing objects: 100% (58/58), 4.03 MiB | 1.25 MiB/s, done.
Total 58 (delta 6), reused 0 (delta 0)
remote: Resolving deltas: 100% (6/6), done.
To https://github.com/juancamiloau/tutorialgit.git
 * [new branch]      master -> master

D:\RepositorioTutorial>
```

Nota: Para la primera vez, se debe agregar usuario y contraseña

- Verificamos en github y ya tenemos nuestro repositorio local en la nube



## V. SENTENCIAS DE GIT

- **git config:** Uno de los comandos más usados en git es git config, que puede ser usado para establecer una configuración específica de usuario, como sería el caso del email, un algoritmo preferido para diff, nombre de usuario y tipo de formato, etc... Por ejemplo, el siguiente comando se usa para establecer un email:  
`git config --global user.email sam@google.com`
- **git init:** Este comando se usa para crear un nuevo repertorio GIT
- **git add:** Este comando puede ser usado para agregar archivos al index. Por ejemplo, el siguiente comando agrega un nombre de archivo temp.txt en el directorio local del index:  
`git add temp.txt`  
`git add -A` (Agrega todos los archivos)
- **git clone:** Este comando se usa con el propósito de revisar repertorios. Si el repertorio está en un servidor remoto se tiene que usar el siguiente comando:  
`git clone https://github.com/juancamiloau/tutorialgit.git`

Pero si estás por crear una copia local funcional del repertorio, usa el comando:  
`git clone /path/to/repository`

- **git commit:** El comando commit es usado para cambiar a la cabecera. Ten en cuenta que cualquier cambio comprometido no afectara al repertorio remoto. Usa el comando:  
`git commit -m "Message to go with the commit here"`
- **git status:** Este comando muestra la lista de los archivos que se han cambiado junto con los archivos que están por ser añadidos o comprometidos.
- **git push:** Este es uno de los comandos más básicos. Un simple push envía los cambios que se han hecho en la rama principal de los repertorios remotos que están asociados con el directorio que está trabajando. Por ejemplo:  
`git push origin master`
- **git checkout:** El comando checkout se puede usar para crear ramas o cambiar entre ellas. Por ejemplo, el siguiente comando crea una nueva y se cambia a ella:  
`git checkout -b <branch-name>`  
Para cambiar de una rama a otra solo usa:  
`git checkout <branch-name>`
- **git remote:** El comando git se usa para conectar a un repositorio remoto. El siguiente comando muestra los repositorios remotos que están configurados actualmente:  
`git remote -v`



Este comando te permite conectar al usuario con el repositorio local a un servidor remoto:

```
git remote add origin https://github.com/juancamiloau/tutorialgit.git
```

- **git branch:** Este comando se usa para listar, crear o borrar ramas.  
Para borrar la rama:  

```
git branch -d <branch-name>
```
- **git pull:** Para poder fusionar todos los cambios que se han hecho en el repositorio local trabajando
- **git merge:** Este comando se usa para fusionar una rama con otra rama activa:  

```
git merge <branch-name>
```
- **git log:** Ejecutar este comando muestra una lista de commits en una rama junto con todos los detalles
- **git reset:** Para resetear el index y el directorio que está trabajando al último estado comprometido se usa este comando:  

```
git reset --hard HEAD
```
- **git rm:** Este comando se puede usar para remover archivos del index y del directorio que está trabajando:  

```
git rm filename.txt
```
- **git fetch:** Este comando le permite al usuario buscar todos los objetos de un repositorio remoto que actualmente no reside en el directorio local que está trabajando. Por ejemplo:  

```
git fetch origin
```

