CIPFP Mislata

Centre Integrat Públic
Formació Professional Superior

Introducción a Git

Autor: Joan Puigcerver Ibáñez
Correo electrónico: j.puigcerveribanez@edu.gva.es
Curso: 2023/2024

1.¿Qué es Git?

Git es **un sistema de control de versiones libre y distribuido** diseñado para gestionar pequeños y grandes proyectos con rapidez y eficiencia. El objetivo principal de Git es controlar y gestionar los cambios realizados en una enorme cantidad de archivos de una manera fácil y eficiente.

Git fue diseñado en 2005 por Linux Torvalds, creador del kernel del sistema operativo Linux, y desde entonces se ha convertido en una herramienta fundamental e imprescindible en la gestión de código fuente en proyectos colaborativos.

Git está basado en **repositorios**, que se inicializan en un directorio concreto y contienen toda la información de los cambios realizados en todo el árbol de directorios y archivos a partir de ese directorio.

Los principales objetivos y características de Git son:

- **Control de versiones**: Git realiza un seguimiento de las modificaciones en los archivos a lo largo del tiempo, lo que permite a los desarrolladores ver y recuperar versiones anteriores de su código. Esta característica es esencial para trabajar en equipos y solucionar problemas o errores.
- **Distribuido**: Cada copia de un repositorio Git contiene todo el historial de cambios y puede operar de forma independiente. Esto facilita el trabajo sin conexión y la colaboración en equipos distribuidos.
- **Rama y fusión**: Git facilita la creación de ramas (*branching*) para desarrollar características o solucionar problemas sin afectar a la rama principal. Después, puedes fusionar (*merge*) las ramas de nuevo en la rama principal cuando estén a punto.

- **Gestión de conflictos**: Git ofrece herramientas para gestionar conflictos en caso de que dos o más personas hayan realizado cambios en la misma parte del código. Los desarrolladores pueden resolver estos conflictos manualmente.
- **Colaboración**: Git facilita la colaboración en proyectos de código abierto o en equipos, ya que permite a múltiples personas trabajar en el mismo proyecto de forma eficiente. Plataformas como GitHub, GitLab y Bitbucket se utilizan comúnmente para alojar repositorios Git online y colaborar en proyectos.
- **Código abierto y gratuito**: Git es de código abierto y gratuito, lo que significa que cualquiera puede utilizarlo sin coste y contribuir al desarrollo de la herramienta.

2.Instalación

Git puede ser descargado desde su página web: https://git-scm.com/ .
Sin embargo, puedes ser instalado fácilmente en Ubuntu o distribuciones basadas en Debian mediante:

```
sudo apt update
sudo apt install git
```

En Windows, una vez instalado tendrás acceso a la herramienta **Git Bash**, que es una terminal basada en el intérprete de Linux Bash, que te permitirá realizar los pedidos de Git en la consola.
Por defecto, Git utiliza el editor `ViM`, un editor de texto por terminal muy potente pero difícil y nada intuitivo para trabajar. Durante la instalación de Git, puede cambiarlo y establecer otro editor con el que está familiarizado.
Si desea cambiar el editor a posteriori, puede ejecutar el mandato:

```
git config --global core.editor <editor>
```

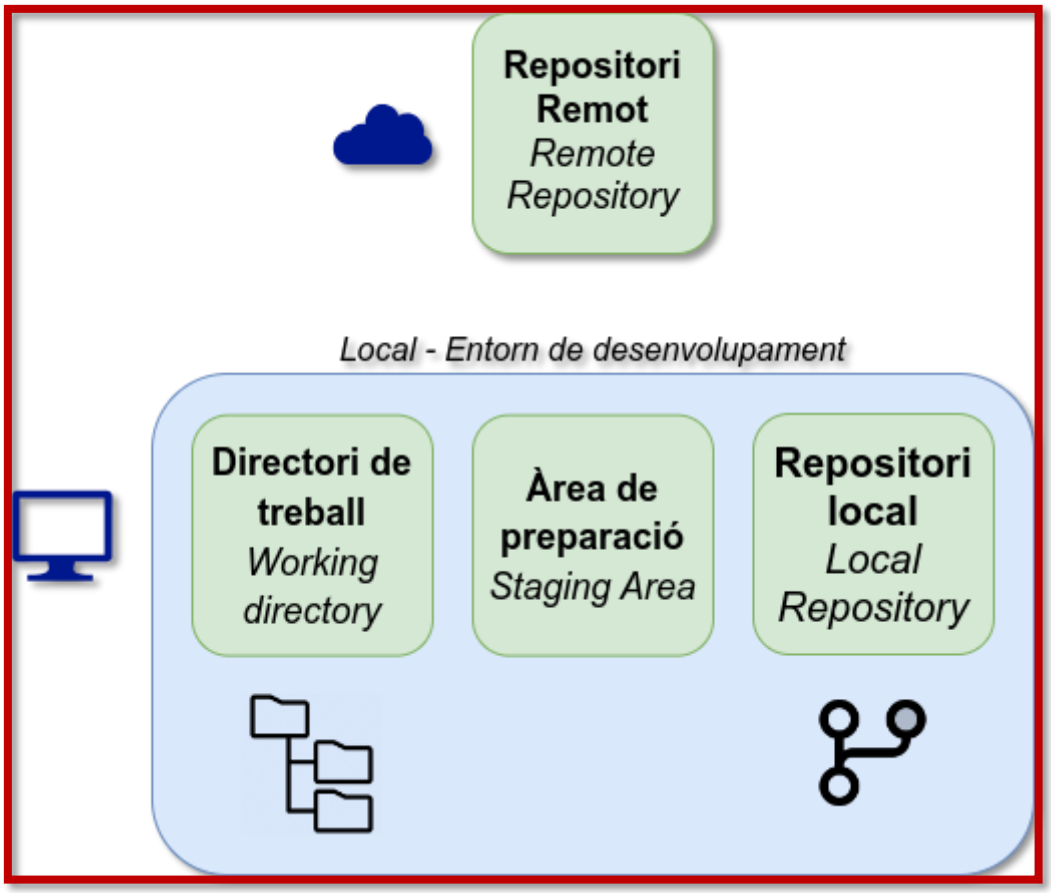
En Windows, el editor más sencillo de utilizar es `notepad`.

```
joapuiib@FP:~ $ git config --global core.editor notepad
```

3.Estructura de un repositorio de Git

En esta introducción, nos centraremos en cómo funcionan los repositorios de Git de una manera **local**, donde todavía no habremos conectado ningún repositorio **remoto**.

Ante todo, debemos conocer la estructura de un repositorio de Git.



En la figura anterior podemos observar lo que se conoce como **Entorno de desarrollo** o ***Development Enviorment***. Esta parte es la parte **local**, presente en tu dispositivo en el que te dispondrás a desarrollar tu aplicación.

Por otro lado, está el **Repositorio Remoto**, normalmente alojado en un servidor accesible por todos los desarrolladores.

Dentro del Entorno *de desarrollo* encontramos los siguientes componentes:

- **Directorio de trabajo** o ***Working directory*** : Directorio o carpeta del sistema donde se almacena *localmente* los contenidos del repositorio.
- **Área de preparación** o ***Staging Area***: Área que se utiliza para indicar qué cambios quieren ser confirmados.
- **Repositorio local** o **Local repository**: Repositorio almacenado *localmente* donde se quedan registradas todas las versiones y cambios realizados en los archivos del repositorio.

En este material nos centraremos en ver cómo se trabaja en Git localmente.

4.Inicialización de un repositorio de Git

Info

Este material describe la utilización de los pedidos de Git desde la **terminal**.
En sistemas *Windows*, puede realizarlo con la herramienta **Git Bash**.

Antes de empezar, he creado un directorio llamado **git_tutorial** en mi carpeta de usuario mediante la herramienta `mkdir`. Después, me he situado dentro de esta carpeta con el comando `cd` y he consultado el contenido con el comando `ls`. En este último comando he incluido la opción `-a`o `--all`, que permite mostrar los archivos o directorios ocultos.

```
joapuiib@FP:~ $ mkdir git_tutorial
joapuiib@FP:~ $ cd git_tutorial
joapuiib@FP:~/git_tutorial $ ls -a
.  ..
```

Nota

En la terminal, puede observar que cada línea comienza con una información, justo antes del cursor, donde puedes escribir texto.
Esta información se llama **prompt** e incluye información del usuario actual, la máquina a la que está conectada y la carpeta actual.
En este caso, tiene el formato `<user>@<host>:<dir> (<branch>) $`.

- `user`hace referencia al usuario actual.
- `host`hace referencia al nombre de la máquina o dispositivo.
- `dir`hace referencia al directorio en el que nos encontramos. El símbolo `~`representa su carpeta de usuario.
    - Puedes ejecutar el pedido `echo ~`para ver en qué directorio se corresponde. (Normalmente `/home/<user>`a sistemas Linux y `C:/Users/<user>`en sistemas Windows.)
- `(branch)`. En Git Bash, además, se muestra la rama actual en la que estamos situados en el repositorio de Git.
- `$`hace referencia al final del prompt. A partir de ahí, el usuario puede escribir el comando que desea ejecutar.

Esta carpeta creada no es más que un directorio normal y corriente que por el momento no tiene ningún contenido. En este momento todavía no existe ningún repositorio de Git.
El mandato `git status` permite ver el estado actual de un repositorio de Git. Podemos comprobar cómo de momento, todavía no existe ninguna.

```
joapuiib@FP:~/git_tutorial $ git status
fatal: not a git repository (or any of the parent directories): .git
```

Por tanto, lo primero que haremos es inicializar un nuevo repositorio de Git. Esta tarea se realiza mediante el orden `git init`:

```
joapuiib@FP:~/git_tutorial $ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositoris, which will suppress this warning, call:
hint:
hint: git config --global init.dafaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint: git branch -m <name>
Initialized empty Git repository in /home/joapuiib/git_tutorial/.git/
joapuiib@FP:~/git_tutorial (master) $ git branch -m main
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main

No commits yet

Nothing to commit (create/copy files and use "git add" to track)
joapuiib@FP:~/git_tutorial (main) $ ls -a
.  ..  .git/
```

Puede observar que en el momento de la inicialización, la herramienta Git crea una rama principal llamada `master`, pero nos indica cómo podemos configurar la herramienta para que de forma predeterminada utilice otro nombre y también nos indica cómo le podemos cambiar el nombre a la rama ya creada.

Siempre se ha utilizado la nomenclatura `master`, pero para evitar la nomenclatura **master/slave**, que tiene connotaciones racistas, últimamente se ha decidido utilizar el nombre `main`. En este caso, se ha renombrado la rama principal a `main`mediante `git branch -m main`.

Una vez creado el repositorio, podemos ver con `git status` el estado del mismo. En este caso, está vacío: " *No commits yet* ".

Finalmente, se ha ejecutado de nuevo el comando `ls -a` para ver los contenidos del directorio actual. Se puede observar que se ha creado el directorio `.git/`, directorio oculto que contiene el **Repositorio local** .

5.Añadir nuevas cosas

Añadiremos el primer archivo a nuestro repositorio. El archivo lo llamaremos **README.md** y añadiremos el siguiente contenido:

```
# Tutorial de Git
Estem aprenent a utilitzar Git!
```

Nota

El archivo **README.md** es un archivo que cualquier repositorio de Git debería tener. El nombre *README* que se puede traducir como *Léame*, es un fichero utilizado para dar información sobre el contenido del repositorio y su utilización.

Este archivo normalmente está escrito en formato *Markdown*, un lenguaje de marcas que permite escribir en un formato de texto plano, que después será traducido a HTML.

Live editor de *Markdown*: https://markdown-it.github.io/

Consejo

Para editar un archivo, puede utilizar cualquier editor de texto o incluso, realizarlo mediante pedidos en la terminal.

Editores de texto:

- En *Windows*:
  - Notepad o bloc de notas. Viene instalado por defecto.
  - Notepad++.
- En *Linux*:
  - Cada distribución utiliza un editor de texto distinto.
  - En *Ubuntu* está Gedit. Puede lanzarse desde la terminal mediante `gedit <file>`.
- *Multiplataforma*:
  - Visual Estudio Code

Editores en la terminal:

- `nano`. Uso básico.
- `vim`. Uso avanzado (`:w` para guardar y `:q` salir)

Desde la terminal: Mediante redirecciones se pueden modificar los archivos.

- `>`: Crea o sobrescribe un archivo con la **salida estándar** de un pedido.
- `>>`: Crea y añade al final de un archivo la **salida estándar** de un pedido.

```
echo "# Tutorial de Git" > README.md
echo "Estem aprenent a utilitzar Git!" >> README.md
```

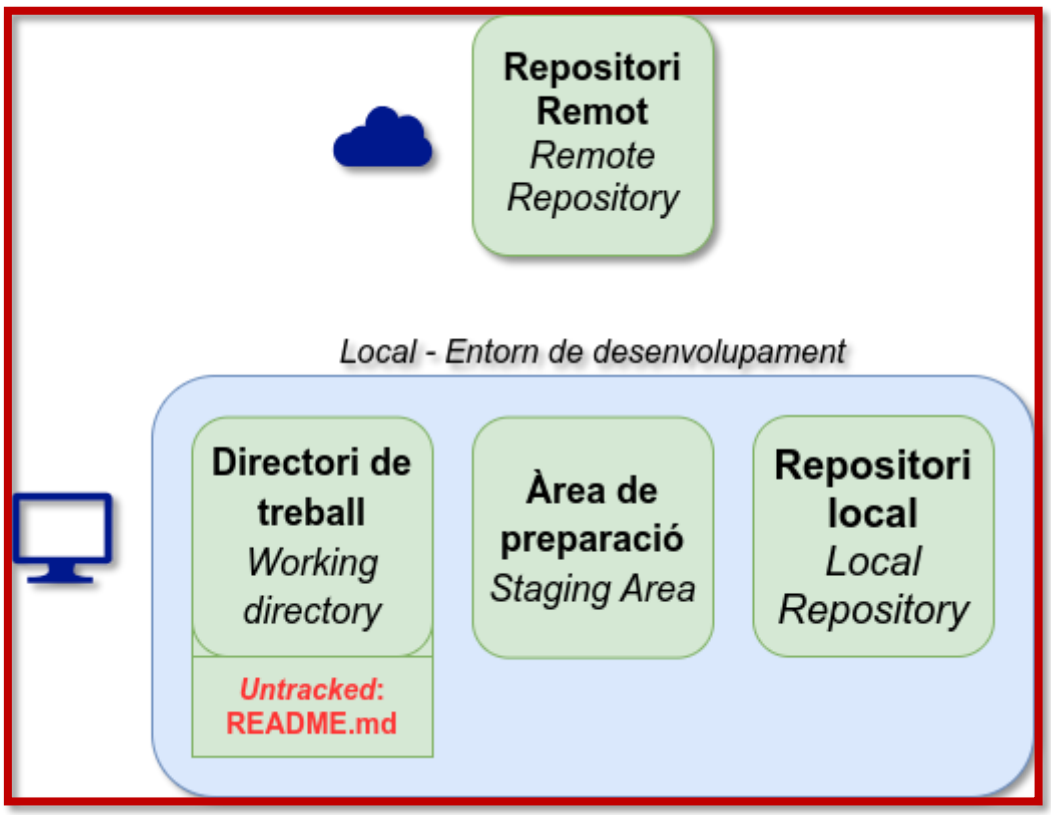
Yo he optado por la opción de la terminal para crear y añadir el texto al archivo **README.md**. Podemos consultar el contenido del archivo con el pedido `cat`.

```
joapuiib@FP:~/git_tutorial (main) $ ls
joapuiib@FP:~/git_tutorial (main) $ echo "# Tutorial de Git" > README.md
joapuiib@FP:~/git_tutorial (main) $ echo "Estem aprenent a utilitzar
Git!" >> README.md
joapuiib@FP:~/git_tutorial (main) $ ls
README.md
joapuiib@FP:~/git_tutorial (main) $ cat README.md
Tutorial de Git
Estem aprenent a utilitzar Git!
```

Este archivo que hemos creado se ha añadido al *Directorio de Trabajo*. Vamos a ver el estado del repositorio con `git status`.

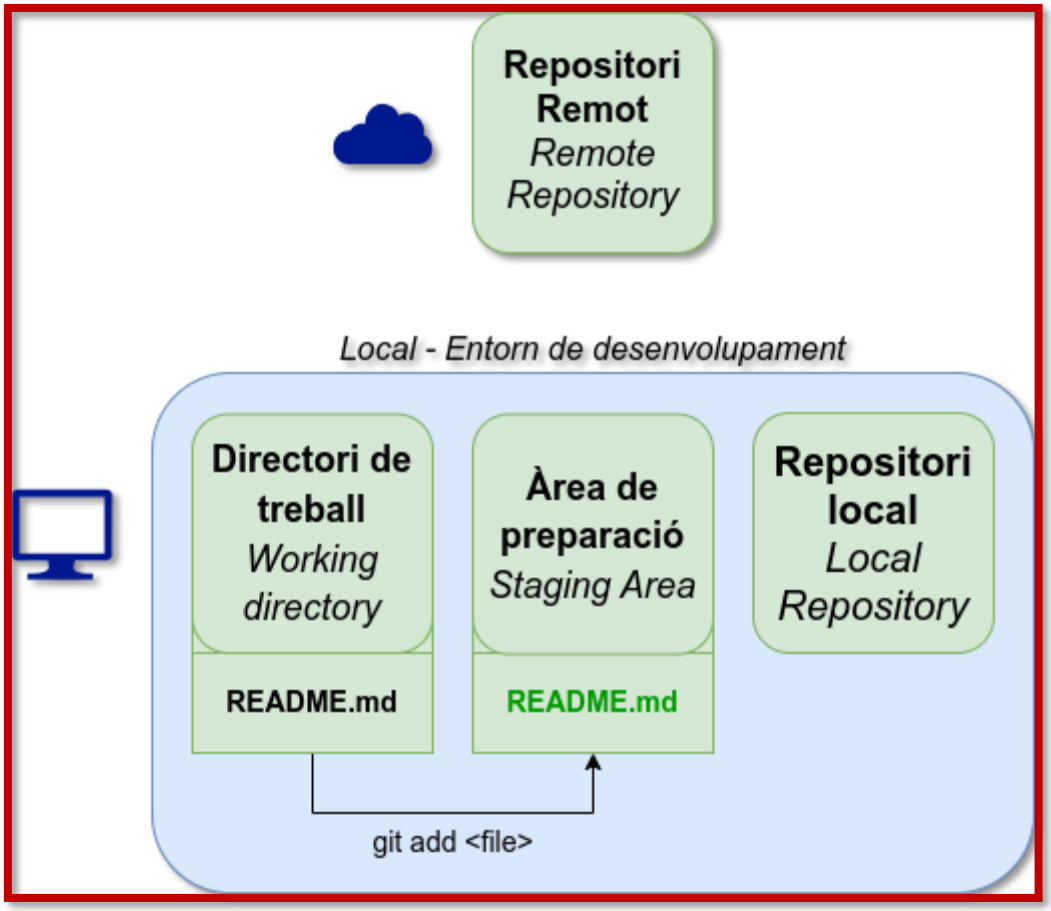
```
joapuiib@FP:~/git_tutorial (main) $ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to
track)
```

Si leemos la información que nos da este pedido, podemos observar que Git reconoce que se ha creado este archivo, pero que todavía no lo tiene en cuenta en nuestro repositorio. Está en el estado **untracked**.

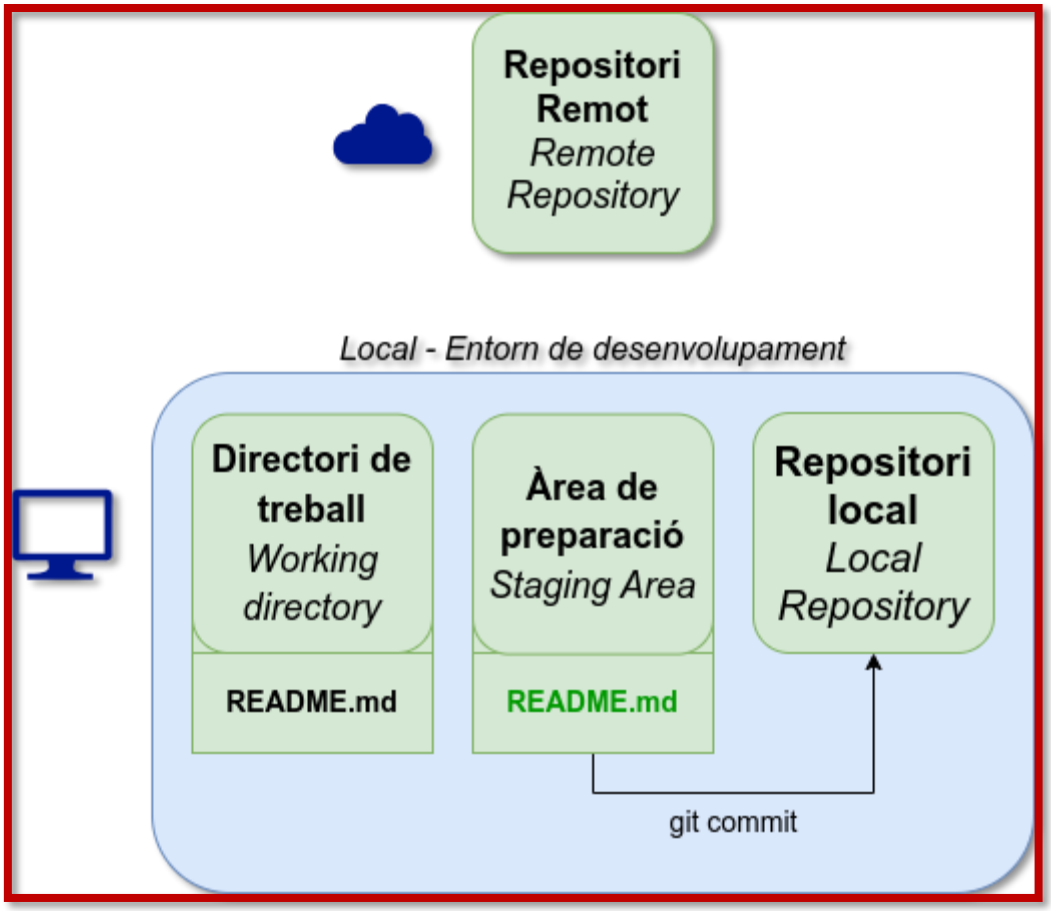


El siguiente paso para añadir los cambios a nuestro repositorio es añadir los cambios al *Área de Preparación* mediante el comando `git add`. Este pedido permite especificar qué cambios se desea añadir.

```
joapuiib@FP:~/git_tutorial (main) $ git add README.md
joapuiib@FP:~/git_tutorial (main) $ git status
On branch master

No commits yet

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



El *Área de Preparación* sirve para recopilar todos los cambios que queremos hacer efectivos en el repositorio de Git. En nuestro caso, sólo queremos añadir el archivo README.md que hemos creado.
Una vez añadidos todos los cambios, ya podemos **confirmar** todos los cambios, mediante el comando `git commit`.



Info

Veremos más adelante qué información se guarda en un `commit`, pero de momento, debe saber que todos `commits`deben contener un mensaje.
Si ejecutas directamente el pedido `git commit`, se abrirá el editor de texto que haya configurado en la instalación de Git. En el editor permite escribir un mensaje. Después de guardar y cerrar el editor, se creará el nuevo `commit`.
Una forma más fácil de especificar el mensaje de uno `commit`es mediante la opción `-m`, que permite especificar el mensaje directamente en la terminal.

```
git commit -m "<message>"
```

Advertencia

Configure previamente el editor de Git (véase el apartado Instalación), ya que el editor configurado por defecto es **ViM**, un editor potente pero complicado de utilizar.
Si por casualidad ha entrado en este editor, puede utilizar las siguientes combinaciones para salir:

- `:wq`: Para guardar y salir.
- `:q!`: Para salir sin guardar.

```
joapuiib@FP:~/git_tutorial (main) $ git commit -m "Added README.md"
*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.
```

La primera vez que haga uno `commit`seguramente se encuentra con el error anterior. Git necesita saber quién es la persona que ha realizado un `commit`, por lo tanto, debe indicarlo previamente mediante el pedido `git config`.
Esta configuración se puede aplicar globalmente mediante la opción `--global`, o a nivel de repositorio, sin esta opción.
Mediante estas órdenes, podemos configurar nuestro nombre y el correo electrónico, que será público y se almacenará en cada `commit` que hagamos.

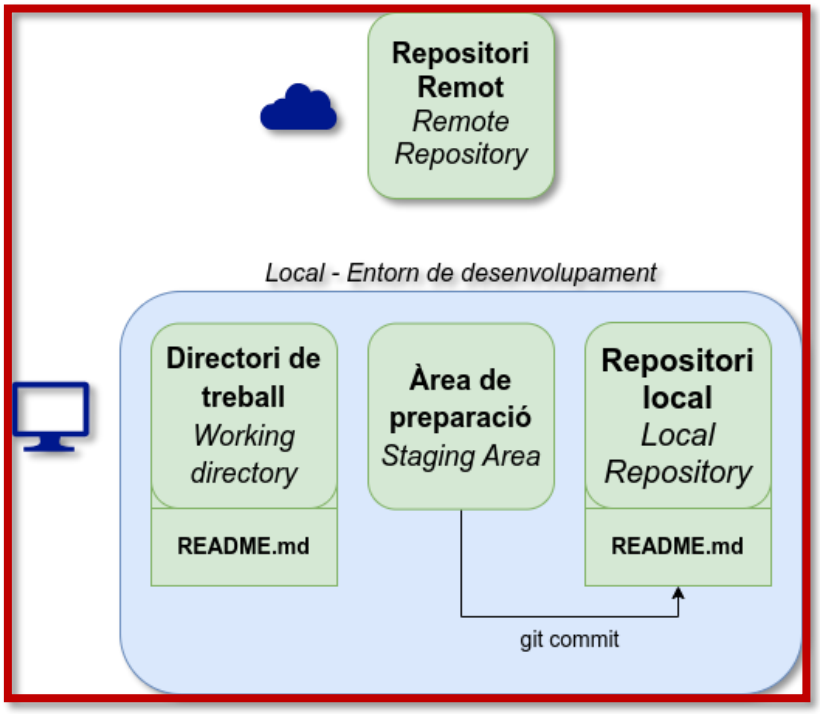
```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

```
joapuiib@FP:~/git_tutorial (main) $ git config --global user.email
"j.puigcerveribanez@edu.gva.es"
joapuiib@FP:~/git_tutorial (main) $ git config --global user.name "Joan
Puigcerver"
```

Una vez realizada esta configuración, ya podemos realizar nuestro primer `commit`.

```
joapuiib@FP:~/git_tutorial (main) $ git commit -m "Added README.md"
[main (root-commit) 8e70293] Added Readme.md
 1 file changed, 2 insertions(+)
 create mode 100644 README.md
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
nothing to commit, working tree clean
```

Podemos observar que, después de haber confirmado los cambios mediante `git commit`, el pedido `git status`nos indica que no existe ningún cambio que no esté registrado en nuestro repositorio. Por tanto, los cambios realizados en el archivo **README.md** ya forman parte del repositorio.



Podemos consultar nuestro recién creado `commit`con el comando `git log`.

```
joapuiib@FP:~/git_tutorial (main) $ git log
commit 8e702933d5dbec9ee71100a1599ae4491085e1aa (HEAD -> main)
Author: Joan Puigcerver <j.puigcerveribanez@edu.gva.es>
Date:   Fri Oct 13 16:06:59 2023 +0200

    Added Readme.md
```

Se puede observar el `commit`, que incluye el autor (nombre y correo electrónico), la fecha en la que se ha hecho, el mensaje especificado y un identificador, que es uno de los `hash`cambios realizados (Más información).

Sumario

Hasta este momento, hemos visto los siguientes pedidos de **Git**:

- `git init`: Inicializa el repositorio de Git.
- `git status`: Muestra el estado actual del repositorio de Git.
- `git add <file>`: Añade el/los archivo/s especificados en el *Staging Area*.
- `git commit -m <message>`: Confirma los cambios del *Staging Area* y crea uno nuevo `commit`.
- `git log`: Muestra los `commits`del repositorio.

6.Más cambios

Repetimos el proceso anterior para añadir nuevos cambios al archivo **README.md**. Modificamos el archivo para que el contenido sea el siguiente:

```
# Tutorial de Git
Estem aprenent a utilitzar Git!
Hem modificat un fitxer existent.
```

```
joapuiib@FP:~/git_tutorial (main) $ cat README.md
Tutorial de Git
Estem aprenent a utilitzar Git!
joapuiib@FP:~/git_tutorial (main) $ echo >> README.md
joapuiib@FP:~/git_tutorial (main) $ echo "Hem modificat un fitxer
existent." >> README.md
joapuiib@FP:~/git_tutorial (main) $ cat README.md
Tutorial de Git
Estem aprenent a utilitzar Git!

Hem modificat un fitxer existent.
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

Si vemos el estado del repositorio con `git status`, podemos observar que **Git** nos indica que el archivo **README.md** ha sido modificado y cómo poder añadir los cambios al repositorio o descartarlos.

Si queremos observar los cambios realizados, disponemos del orden `git diff <file>`. Este mandato muestra los cambios del *Directorio de Trabajo* del archivo especificado. Si no se especifica el archivo, se muestran todos los cambios.

```
joapuiib@FP:~/git_tutorial (main) $ git diff
diff --git a/README.md b/README.md
index 6d747b3..ff524e4 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,4 @@
 # Tutorial de Git
 Estem aprenent a utilitzar Git!
+
+Hem modificat un fitxer existent.
```

Esta herramienta muestra un `diff`, una diferencia entre el estado anterior del archivo y los cambios realizados. En este caso nos indica que se han añadido dos líneas al final del archivo.
Probamos a descartar los cambios del *Directorio de Trabajo* con la herramienta `git restore`.

```
joapuiib@FP:~/git_tutorial (main) $ cat README.md
Tutorial de Git
Estem aprenent a utilitzar Git!

Hem modificat un fitxer existent.
joapuiib@FP:~/git_tutorial (main) $ git restore README.md
joapuiib@FP:~/git_tutorial (main) $ cat README.md
Tutorial de Git
Estem aprenent a utilitzar Git!
joapuiib@FP:~/git_tutorial (main) $ git diff
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
nothing to commit, working tree clean
```

Observamos que después de descartar los cambios, el archivo **README.md** está en el último estado conocido por Git y los cambios que hemos hecho han desaparecido.

Peligro
¡Los cambios del *Directorio de Trabajo* que han sido descartados mediante `git restore` no pueden ser recuperados!
Volvemos a incluir los cambios en el archivo **README.md**, y en este caso, vamos a añadirlos al *Staging Area*.

```
joapuiib@FP:~/git_tutorial (main) $ echo >> README.md
joapuiib@FP:~/git_tutorial (main) $ echo "Hem modificat un fitxer existent." >> README.md
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
joapuiib@FP:~/git_tutorial (main) $ git add README.md
joapuiib@FP:~/git_tutorial (main) $ git diff
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```

Mediante `git add` hemos añadido los cambios en el *Staging Area* y después de hacerlo, podemos observar que el pedido `git diff` no muestra nada. Esto se debe a que `git diff`muestra los cambios del *Directorio de Trabajo*. Si queremos ver los cambios del *Staging Area* , debemos añadir la opción `--staged`.

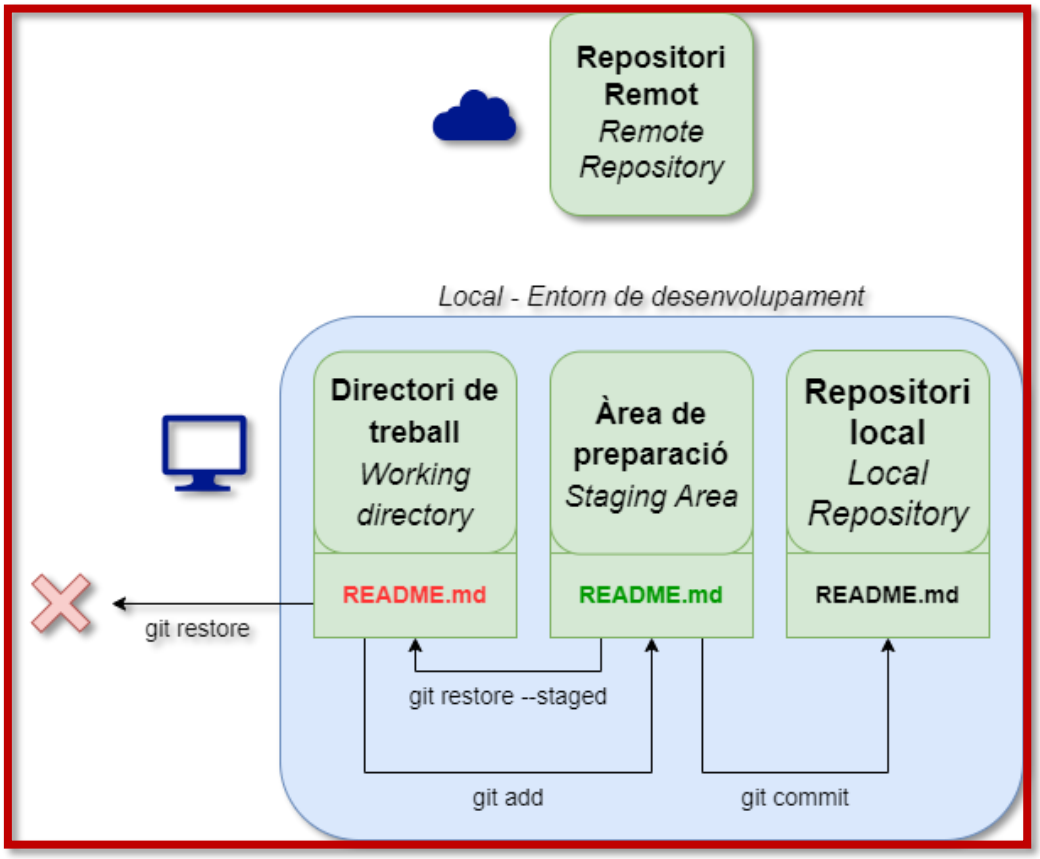
```
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
joapuiib@FP:~/git_tutorial (main) $ git diff
joapuiib@FP:~/git_tutorial (main) $ git diff --staged
diff --git a/README.md b/README.md
index 6d747b3..ff524e4 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,4 @@
 # Tutorial de Git
 Estem aprenent a utilitzar Git!
+
+Hem modificat un fitxer existent.
```

Si lee la información de `git status`, puede observar que nos indica que podemos quitar cambios del *Staging Area* con el orden `git restore --staged`. Si hacemos este pedido, los cambios todavía siguen presentes en el *Working Directory* .

```
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
joapuiib@FP:~/git_tutorial (main) $ git restore --staged README.md
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

Por último, volvemos a añadir los cambios y crear uno nuevo `commit` con estos cambios.

```
joapuiib@FP:~/git_tutorial (main) $ git add REDME.md
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
joapuiib@FP:~/git_tutorial (main) $ git commit -m "Changed README.md"
[main c9fc6c8] Changed README.md
 1 file changed, 2 insertions(+)
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
nothing to commit, working tree clean
```

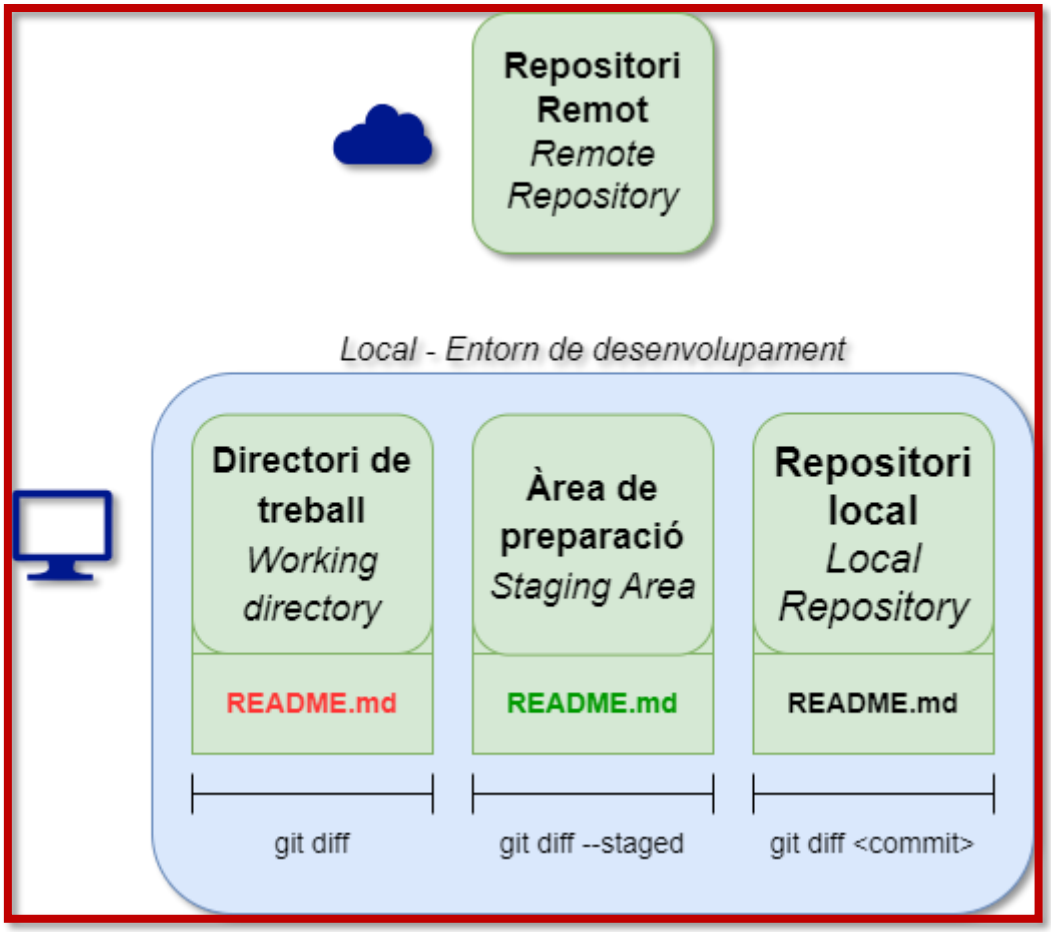


Podemos consultar nuestro recién creado `commit`con el comando `git log`.

```
joapuiib@FP:~/git_tutorial (main) $ git log
commit c9fc6c856c2d52744b85a6f8d92feac496e60bd6 (HEAD -> main)
Author: Joan Puigcerver <j.puigcerveribanez@edu.gva.es>
Date:   Mon Oct 16 11:43:20 2023 +0200

    Changed README.md

commit 8e702933d5dbec9ee71100a1599ae4491085e1aa
Author: Joan Puigcerver <j.puigcerveribanez@edu.gva.es>
Date:   Fri Oct 13 16:06:59 2023 +0200

    Added Readme.md
```

Además, podemos utilizar la herramienta `git show` para mostrar la información y los cambios (`diff`) de cada `commit`.

```
joapuiib@FP:~/git_tutorial (main) $ git show 8e70293
commit 8e702933d5dbec9ee71100a1599ae4491085e1aa
Author: Joan Puigcerver <j.puigcerveribanez@edu.gva.es>
Date:   Fri Oct 13 16:06:59 2023 +0200

    Added Readme.md

diff --git a/README.md b/README.md
new file mode 100644
index 0000000..6d747b3
--- /dev/null
+++ b/README.md
@@ -0,0 +1,2 @@
+# Tutorial de Git
+Estem aprenent a utilitzar Git!

joapuiib@FP:~/git_tutorial (main) $ git show c9fc6c8
commit c9fc6c856c2d52744b85a6f8d92feac496e60bd6 (HEAD -> main)
Author: Joan Puigcerver <j.puigcerveribanez@edu.gva.es>
Date:   Mon Oct 16 11:43:20 2023 +0200

    Changed README.md

diff --git a/README.md b/README.md
index 6d747b3..ff524e4 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,4 @@
# Tutorial de Git
 Estem aprenent a utilitzar Git!
+
+Hem modificat un fitxer existent.
```



7.Commit

Uno `commit` Git es un punto fijo en la historia de un repositorio, que representa un conjunto de cambios en los ficheros de tu proyecto. Cada `commit`almacena la siguiente información:

- **Un identificador (hash)**: Identificador único (una cadena hexadecimal) que le diferencia de todos los demás commits en el repositorio.
- **Autor y correo electrónico**: Información de quien hizo el commit, que suele ser el nombre y el correo electrónico del autor. Esta información se configura mediante `git config`.
- **Fecha y hora**: La fecha y hora en que se creó el commit.
- **Mensaje**: Un mensaje que proporciona una descripción significativa de los cambios realizados en este commit.
- **Referencia al commit anterior**: Cada commit hace referencia al commit anterior en el historial, creando una línea de tiempo o historia de los cambios del proyecto.
- **Cambios en los archivos**: Los cambios específicos en los archivos del proyecto que se realizaron en este commit.

Toda esta información se puede consultar con el pedido `git show`

```
joapuiib@FP:~/git_tutorial (main) $ git show 8e70293
commit 8e702933d5dbec9ee71100a1599ae4491085e1aa
Author: Joan Puigcerver <j.puigcerveribanez@edu.gva.es>
Date:   Fri Oct 13 16:06:59 2023 +0200

    Added Readme.md

diff --git a/README.md b/README.md
new file mode 100644
index 0000000..6d747b3
--- /dev/null
+++ b/README.md
@@ -0,0 +1,2 @@
+# Tutorial de Git
+Estem aprenent a utilitzar Git!

joapuiib@FP:~/git_tutorial (main) $ git show c9fc6c8
commit c9fc6c856c2d52744b85a6f8d92feac496e60bd6 (HEAD -> main)
Author: Joan Puigcerver <j.puigcerveribanez@edu.gva.es>
Date:   Mon Oct 16 11:43:20 2023 +0200

    Changed README.md

diff --git a/README.md b/README.md
index 6d747b3..ff524e4 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,4 @@
# Tutorial de Git
 Estem aprenent a utilitzar Git!
+
+Hem modificat un fitxer existent.
```

La historia del repositorio se puede consultar con el orden `git log`. Si consulta las ejecuciones anteriores de este pedido, verá que se proporciona mucha información y es difícil ver una visión general de la historia a simple vista. Para ver de una forma más visual la historia, podemos definirnos un formato personalizado a este pedido. Para no tener que especificarlo cada vez, podemos crearnos los siguientes **alias**.

```
[alias]
        lg = log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)'
        lg2 = log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold cyan)%aD%C(reset) %C(bold green)(%ar)%C(reset)%C(bold yellow)%d%C(reset)%n'' %C(white)%s%C(reset) %C(dim white)- %an%C(reset)'
        lga = !"git lg --all"
```

Los **alias** se configuran en el archivo **~/.gitconfig** , situado en la carpeta de usuario.

Nota

La carpeta de usuario es la carpeta que almacena toda la información respecto al usuario, como los documentos o el escritorio.

- En sistemas **Linux** está en `/home/<user>`.
- En sistemas **Windows** está en `C:\Users\<user>`.

El archivo **.gitconfig**, además, almacena todas las configuraciones que realizamos en Git mediante `git config --global`. Podrá observar que el nombre y el correo especificados anteriormente están almacenados en el archivo:

```
[user]
        name = Joan Puigcerver
        email = j.puigcerveribanez@edu.gva.es

[core]
        editor = notepad

[alias]
        lg = log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)'
        lg2 = log --graph --abbrev-commit --decorate --format=format:'%C(bold blue)%h%C(reset) - %C(bold cyan)%aD%C(reset) %C(bold green)(%ar)%C(reset)%C(bold yellow)%d%C(reset)%n'' %C(white)%s%C(reset) %C(dim white)- %an%C(reset)'
        lga = !"git lg --all"
```

Una vez configurados los alias, podemos utilizar el pedido `git lg`, que ejecutará `git log`con el formato especificado. En este formato podemos ver cómo un commit se crea a partir de otro commit.

```
joapuiib@FP:~/git_tutorial (main) $ git lg
* c9fc6c8 - (1 day ago) Changed README.md - Joan Puigcerver (HEAD -> main)
* 8e70293 - (4 days ago) Added Readme.md - Joan Puigcerver
```

8.Estados de un archivo en Git

Los archivos de un repositorio de Git pueden estar en diferentes estados. Podemos consultar el estado de cada archivo con el pedido `git status`.

- **Untracked**: Estado inicial de un archivo que aún no ha sido añadido al repositorio. Git no conoce su existencia.

```
joapuiib@FP:~/git_tutorial (main) $ touch new_file.txt
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        new_file.txt

nothing added to commit but untracked files present (use "git add" to track)
```

- **Modified**: Cuando un archivo conocido por Git ha sido modificado, es decir, ha cambiado respecto al último commit.

```
joapuiib@FP:~/git_tutorial (main) $ echo "Modified" >> README.md
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

- **Staged**: Antes de hacer uno `commit`debes añadir los cambios al *Área de Preparación (Staging Area)*. Cuando añades un archivo, con el pedido `git add`, pasa a ese estado.

```
joapuiib@FP:~/git_tutorial (main) $ git add new_file.txt
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   new_file.txt
```

- **Commited**: El archivo existe en el repositorio de Git y no ha sido cambiado.

```
joapuiib@FP:~/git_tutorial (main) $ git restore README.md # Esborrem les modificacions
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
nothing to commit, working tree clean
```

- **Ignored**: Git puede ser configurado para que ignore ciertos archivos del *Directorio de Trabajo*. Los archivos ignorados se especifican en el archivo **.gitignore** .

```
joapuiib@FP:~/git_tutorial (main) $ touch new_file.txt
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        new_file.txt

nothing added to commit but untracked files present (use "git add" to track)
joapuiib@FP:~/git_tutorial (main) $ echo new_file.txt >> .gitignore
joapuiib@FP:~/git_tutorial (main) $ cat .gitignore
joapuiib@FP:~/git_tutorial (main) $ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
```

9.Recursos adicionales

- Curso de Git desde cero por MoureDev: https://www.youtube.com/watch?v=3GymExBkKjE&ab_channel=MoureDevbyBraisMoure
- https://github.com/UnseenWizzard/git_training

10. Bibliografía

- https://github.com/UnseenWizzard/git_training
- https://www.theserverside.com/feature/Why-GitHub-renamed-its-master-branch-to-main
- https://stackoverflow.com/questions/35430584/how-is-the-git-hash-calculated
- https://en.wikipedia.org/wiki/Diff