

¿Qué es y cómo usar Git?

*UNA COLABORACIÓN DE ALCIDES LUQUEZ Y
AGUSTIN MARCO A BASES DE DATOS APLICADA*

Índice

- [¿Qué es git?](#)
- [Glosario](#)
- [1. Instalar git](#)
- [2. Crear el repositorio local](#)
- [3. Crear repositorio en la nube](#)
- [4. Clonar repositorio](#)
- [5. Publicar cambios](#)
- [6. Recibir cambios](#)
- [7. Conflictos y posibles soluciones](#)
- [8. Integrar git con SSMS](#)
- [9. Esconder archivos con git](#)
- [Bibliografía](#)

¿Qué es git?

Git es un sistema de control de versiones, lo que significa que nos ayuda rastrear y gestionar cambios en los archivos de nuestros proyectos, sirviendo como herramienta para coordinar el trabajo de varias personas.

Con Git vamos a poder guardar un historial de los cambios en el proyecto, pudiendo ver qué cambió, quién lo hizo y cuándo.

Glosario:

Repositorio: Es el lugar donde se almacenan todos los archivos y el historial de cambios del proyecto. Puede estar en un equipo (repositorio local) o en un servidor remoto. (Plataformas como GitHub, BitBucket, GitLab, etc.).

Commit: Cada commit guarda un estado específico del proyecto.

Branch: Es la separación del desarrollo en diferentes versiones dentro del repositorio, podemos tener, por ejemplo, una rama donde está la versión de proyecto en la que trabajamos nosotros, u otra donde esté una versión donde estamos desarrollando una nueva funcionalidad.

Merge: Es la acción de combinar dos ramas en una sola, siguiendo el ejemplo pasado, cuando terminamos la nueva funcionalidad podemos hacer un merge con la rama original.

Push y Pull: Acciones que sincronizan el repositorio local con el remoto. Push guarda tus cambios al servidor remoto y Pull actualiza tus archivos locales con la última versión que tenga el repositorio.

1. Instalar git

Descargamos el instalador de git desde el siguiente enlace [Git - Downloads \(git-scm.com\)](https://git-scm.com) y seguimos los pasos de instalación presentado en el video a continuación.

About

Documentation

Downloads

GUI Clients

Logos

Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Downloads



macOS



Windows



Linux/Unix



Older releases are available and the Git source repository is on GitHub.

GUI Clients

Git comes with built-in GUI tools (**git-gui**, **gitk**), but there are several third-party tools for users looking for a platform-specific experience.

[View GUI Clients →](#)

Logos

Various Git logos in PNG (bitmap) and EPS (vector) formats are available for use in online and print projects.

[View Logos →](#)

Git via Git

If you already have Git installed, you can get the latest development version via Git itself:

```
git clone https://github.com/git/git
```

You can also always browse the current contents of the git repository using the [web interface](#).



About this site

Patches, suggestions, and comments are welcome.

Git is a member of Software Freedom Conservancy



Búsqueda



2. Crear el repositorio local

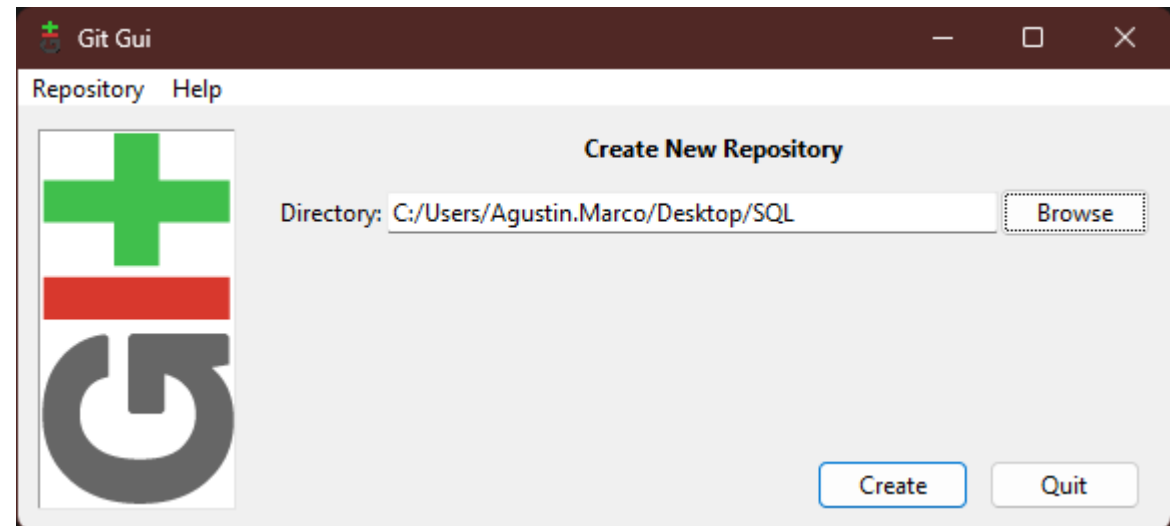
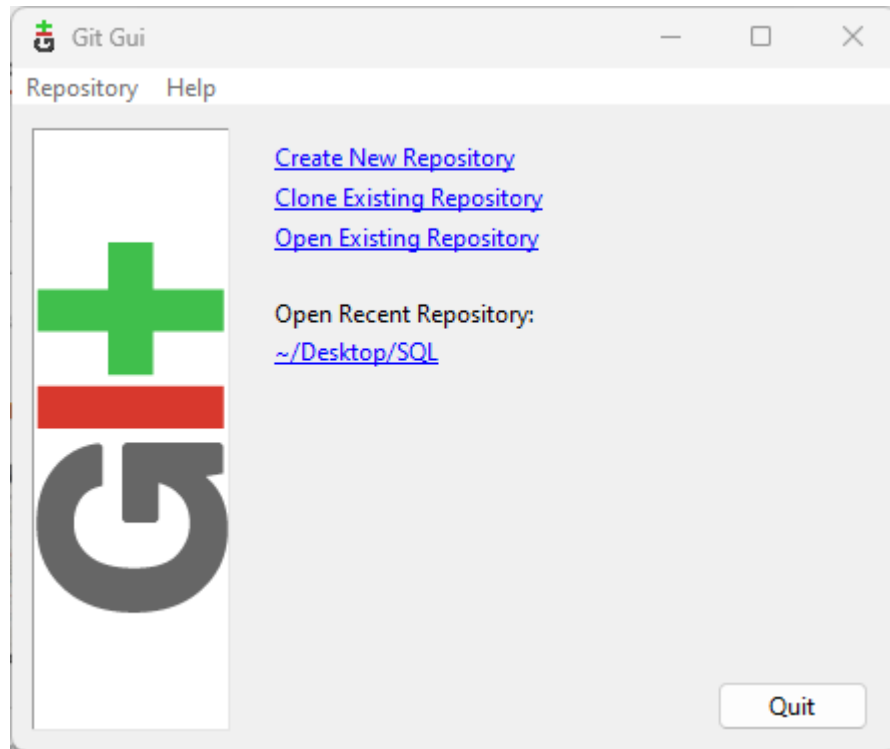
Para que git pueda comenzar a seguir tus archivos hay que primero inicializar un repositorio.

Así que, en la carpeta raíz de nuestro proyecto, Clic derecho -> Abrir en Terminal, y escribimos el comando:

```
git init
```

Al finalizar git debería crear una carpeta oculta '.git', puede borrar esa carpeta en caso de que ocurra alguna falla y quiera reiniciar el estado del repositorio.

Alternativamente, Git ofrece una interfaz gráfica. Iniciando el programa 'Git GUI', seleccionando la opción de 'Create New Repository' y seleccionar la carpeta raíz del proyecto.



3. Crear repositorio en la nube

Para que varios usuarios colaboren en un mismo proyecto, necesitamos de un repositorio central al cual todos puedan acceder.

En esta presentación vamos a utilizar GitHub [github.com] para armar un repositorio central en la nube. Pero existen otras opciones para alojar nuestros repositorios en la nube o inclusive en un servidor *on-premise*.


Para crear un repositorio en GitHub ingresamos a github.com/new

Le asignamos nombre, descripción (opcional) y lo creamos.

Para simplificar el proceso no vamos a indicarle que agregue un README ni una licencia.

Required fields are marked with an asterisk (*).

Owner *

 sad-ko

Repository name *

Azure-Git-Tutorial

✓ Azure-Git-Tutorial is available.

Great repository names are short and memorable. Need inspiration? How about **cuddly-meme** ?

Description (optional)

Repositorio de prueba para demostrar como integrar Git con Azure Data Studio



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

Ahora tenemos que vincular nuestro repositorio local con el de la nube.

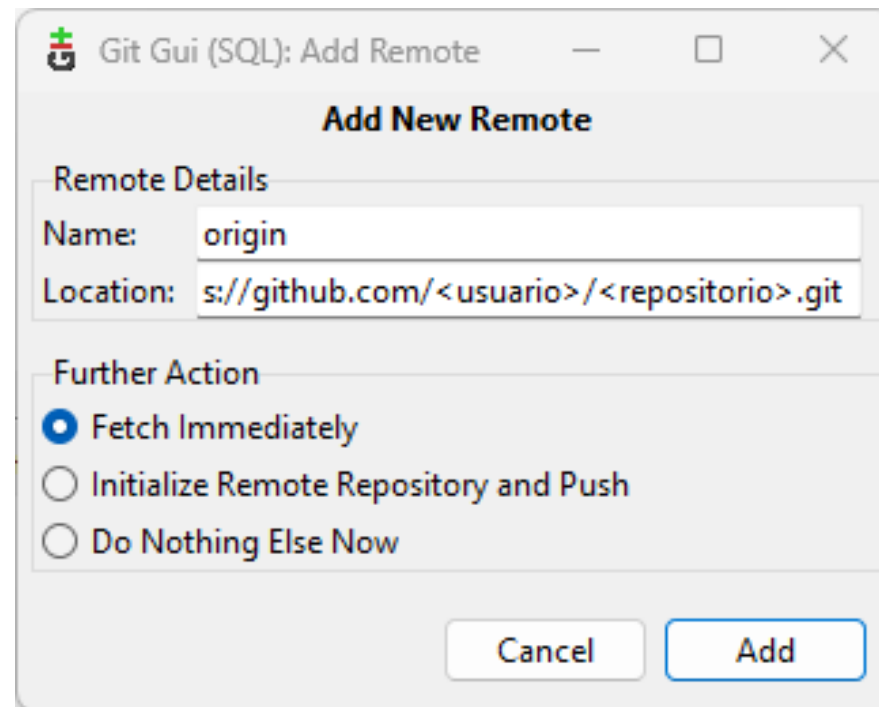
En una terminal ubicada en la carpeta de nuestro proyecto (con git ya inicializado) escribimos el comando:

```
git remote add origin https://github.com/<usuario>/<repositorio>.git
```

Con '**git remote show origin**' podemos verificar si se vincularon exitosamente.

```
PS C:\Users\Agustin.Marco\Projects\Azure Git Tutorial> git remote show origin
* remote origin
  Fetch URL: https://github.com/sad-ko/Azure-Git-Tutorial.git
  Push URL: https://github.com/sad-ko/Azure-Git-Tutorial.git
  HEAD branch: (unknown)
```

Alternativamente, en Git GUI. Podemos ir al menú Remote -> Add (Ctrl+A) y completar la ventana como se muestra a continuación.



4. Clonar repositorio

Lo mostrado hasta ahora serviría para el caso en el que no existirá un repositorio global previamente creado. Si alguno de nuestros compañeros ya se tomó las molestias de realizar el trabajo previo ¿Cómo podemos usar ese repositorio global?

Para poder realizar cambios a un repositorio global existente necesitamos cumplir con dos requisitos:

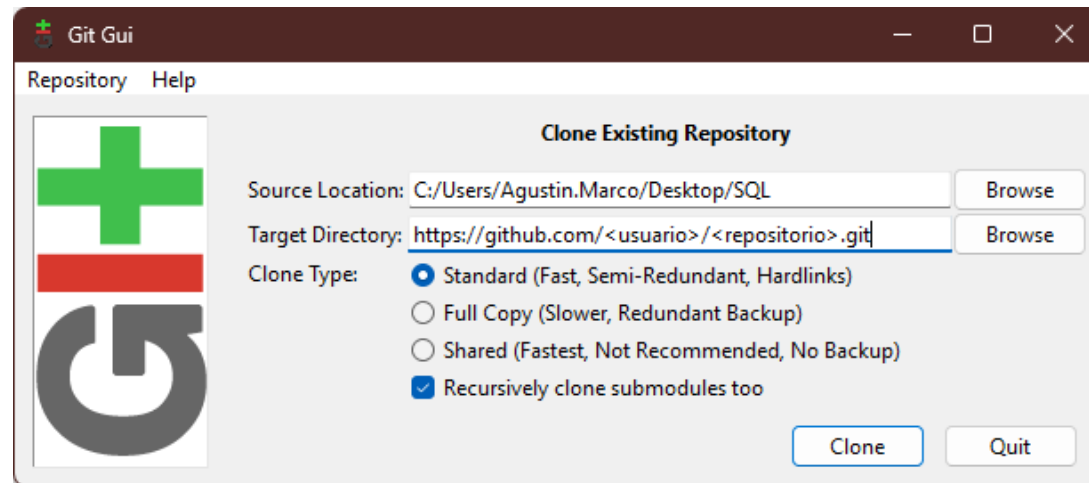
1. Clonar el repositorio global.
2. Tener permisos para publicar en ese repositorio global.

Como clonar un repositorio

Esta es la parte más sencilla, bastaría con ejecutar la siguiente línea de comando dentro de una carpeta vacía.

```
git clone https://github.com/<usuario>/<repositorio>.git
```

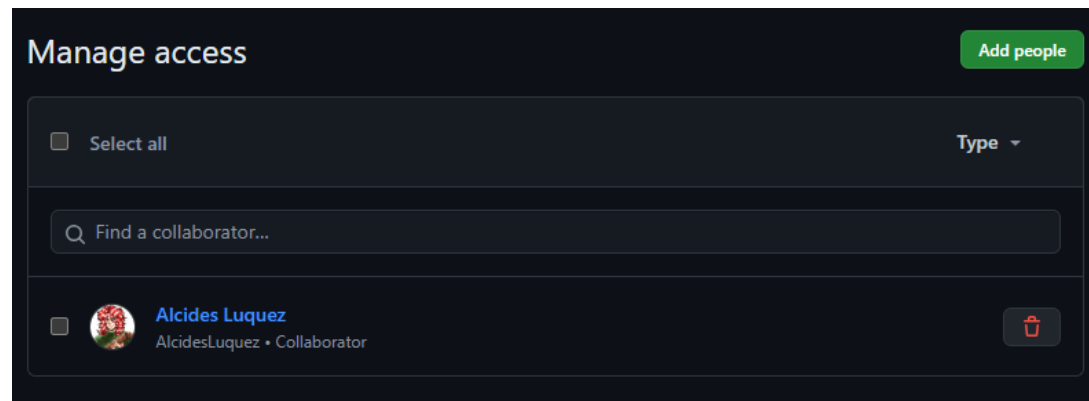
O en Git GUI, mediante la opción 'Clone Existing Repository' y completando la ventana como a continuación.



Como darle permisos a un usuario

Para que varias personas puedan publicar modificaciones a un repositorio global en GitHub, primero necesitan ser colaboradores en el repositorio. Para lograr eso el dueño del repositorio se los tiene que conceder de la siguiente forma:

1. Dentro del repositorio en GitHub, vamos a la pestaña 'Settings' -> 'Collaborators'.
2. Hacemos clic en 'Add people' ingresamos el nombre de usuario y enviamos la invitación.
3. El usuario invitado debe aceptar la invitación para poder comenzar a hacer cambios en el repositorio.



5. Publicar cambios

Git mantiene un registro de los últimos cambios realizados a todos los archivos del proyecto, para poder verlos utilizamos el comando:

git status

En este ejemplo, podemos ver que tenemos dos scripts SQL que todavía no fueron agregados al repositorio global.

```
PS C:\Users\Agustin.Marco\Projects\Azure Git Tutorial> git status
On branch main

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\Agustin.Marco\Projects\Azure Git Tutorial> |
```


Para poder agregar esos archivos al repositorio global, primero tenemos que realizar un ***commit***.

Para poder publicar estos archivos hay que ejecutar los siguientes comandos:

- **git add <X>** -- Reemplazando <X> por el archivo a añadir al *commit* o usando un punto para indicarle a git que queremos añadir todos los archivos modificados (**git add .**)
- **git commit -m "Aca explicamos brevemente que cambios hicimos"**
- **git push origin -u main** -- 'origin' sería el repositorio en la nube que vinculamos en el paso anterior, '-u main' le indica a git que cree esa branch si no existe (ver 6.)

main

1 Branch

0 Tags

Go to file

t

Add file

<> Code

Agustin Marco and Agustin Marco

Creamos tabla de Etiquetas y SP para comprobar su di... 0d6a469 · 16 minutes ago

1 Commits

sp_ComprobarDisponibilidad.sql

Creamos tabla de Etiquetas y SP para comprobar su disponi...

16 minutes ago

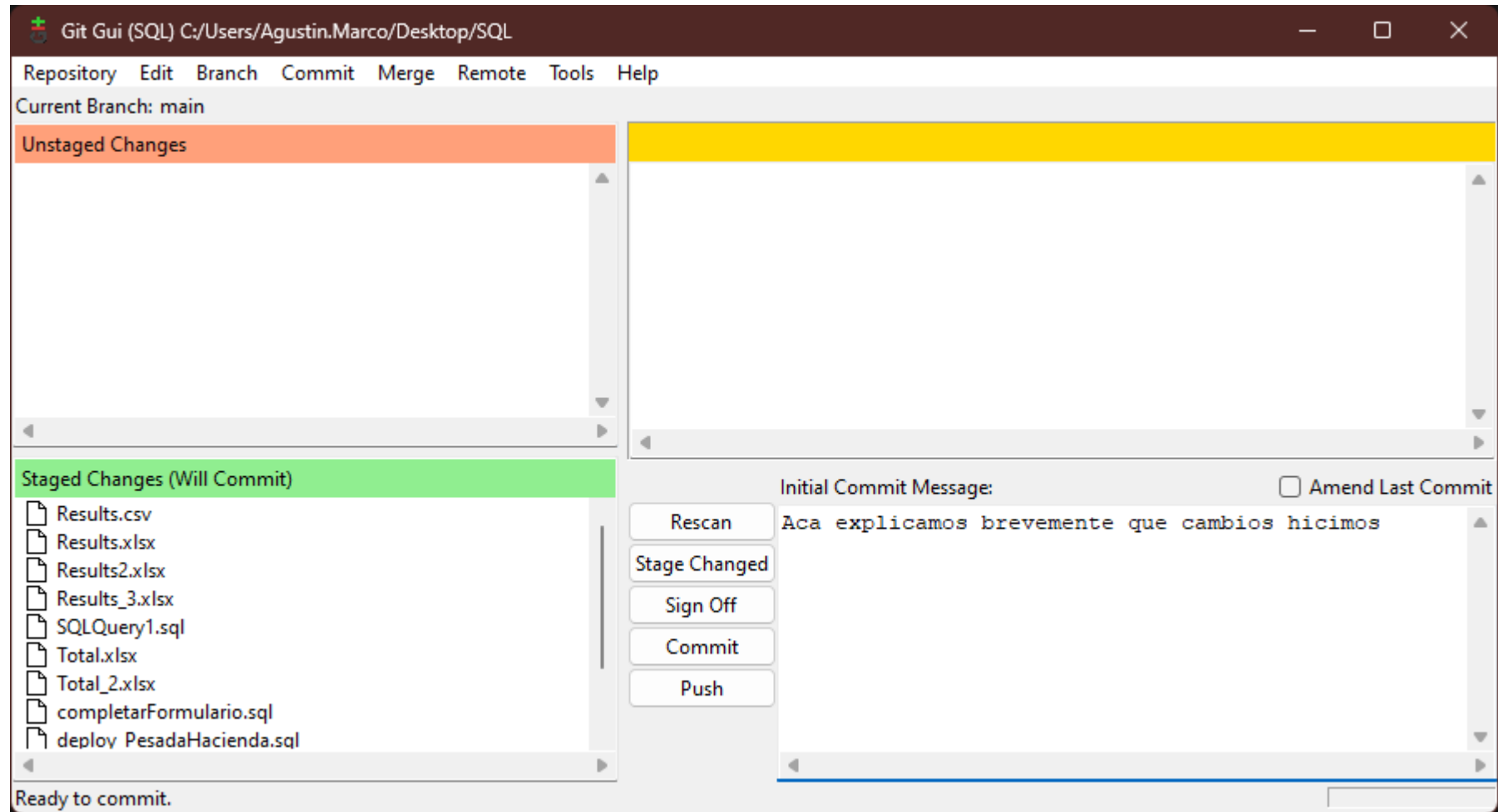
t_FaenaEtiquetas.sql

Creamos tabla de Etiquetas y SP para comprobar su disponi...

16 minutes ago

Si todo salió como debía al entrar al repositorio en GitHub deberíamos poder ver nuestro *commit* recién publicado.

Alternativamente, en Git GUI. Podemos ir al menú Commit -> Stage Changed Files to Commit (Ctrl+I), agregar un mensaje al commit y presionar los botones 'Commit' y luego 'Push' para publicar los cambios al repositorio global.



6. Recibir cambios

Ya sabemos cómo subir cambios al repositorio global.

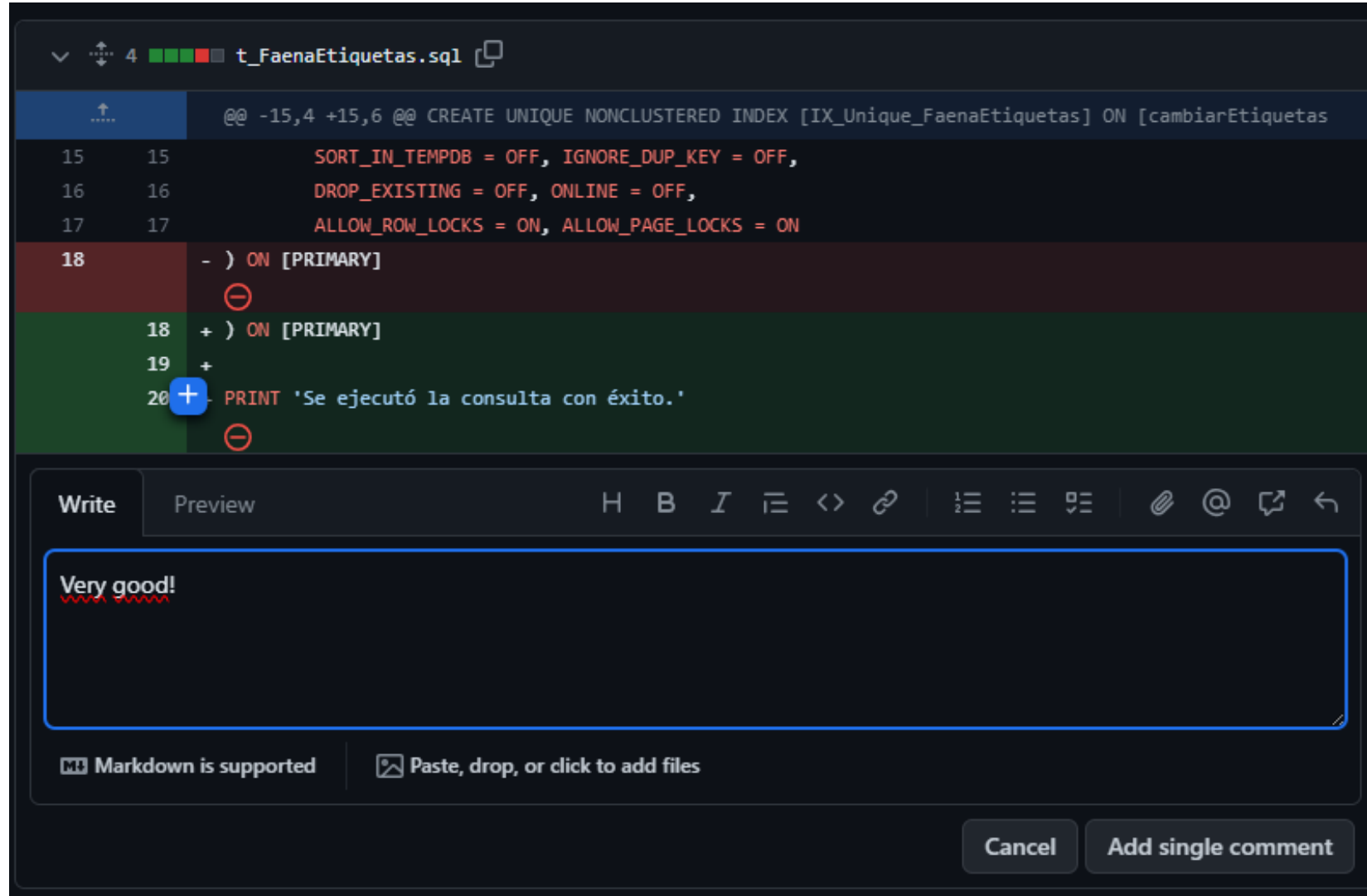
Pero ¿qué pasa cuando otro programador realiza esos cambios?

¿Como podemos actualizar nuestro repositorio local con esos ajustes?

Como podemos ver en la siguiente imagen, vemos que alguien más hizo una actualización al repositorio, podemos leer el comentario que dejaron explicando el propósito de este *commit* y haciendo clic en el comentario podemos ver puntualmente que líneas modificaron y/o agregaron a que archivos.



También GitHub nos permite agregar comentarios en alguna línea puntual para hacer algunas observaciones al respecto.



Si ejecutamos '**git status**' en nuestro repositorio local vamos a ver que git erróneamente nos dice que estamos al día con el repositorio original.

```
PS C:\Users\Agustin.Marco\Projects\Azure Git Tutorial> git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Para poder arreglar eso podemos forzar la actualización con '**git pull**' o bien podemos hacer '**git fetch origin**' para que solo nos actualice la metadata del repositorio.

Ahora si volvemos a ejecutar '**git status**', git nos va a informar que estamos atrasados por un *commit*.

Para poder poner a día nuestro repositorio local bastaría con ejecutar:

```
git pull
```

Y nos va a traer desde GitHub los archivos actualizados, también nos va a informar que archivos se actualizaron y cuantas líneas cambiaron.

```
PS C:\Users\Agustin.Marco\Projects\Azure Git Tutorial> git pull
Updating c65debc..94711a3
Fast-forward
 t_FaenaEtiquetas.sql      | 4 +++-
 v_FaenaEtiquetasActivas.sql | 2 ++
2 files changed, 5 insertions(+), 1 deletion(-)
create mode 100644 v_FaenaEtiquetasActivas.sql
```


7. Conflictos y posibles soluciones

En ocasiones puede suceder que tenemos que actualizar nuestro repositorio local, pero al tener cambios no publicados nos tira el siguiente error:

```
PS C:\Users\Agustin.Marco\Projects\Azure Git Tutorial - copia> git pull
error: Your local changes to the following files would be overwritten by merge:
    t_FaenaEtiquetas.sql
Please commit your changes or stash them before you merge.
Aborting
Updating c65debc..94711a3
```

Si lo googleamos hay múltiples formas de solucionar esto (stash, reset, rebase), pero en nuestra experiencia pueden terminar causando aún más problemas y corriendo riesgos innecesarios de perder el trabajo realizado.

Una solución más sencilla que ofrecemos es publicar los últimos cambios a una nueva rama, y unificar ambas ramas antes de actualizar el repositorio local.

¿Qué es una rama en git?

Una rama en git (o branch, como se lo suele googlear) es básicamente otra línea temporal de cómo sería el proyecto si se hubiera hecho X cosa diferente. Se lo suele utilizar para experimentar con cambios en el programa o agregar nuevas funciones sin romper la build principal.



Para eso vamos a tener que primero saltar a esta nueva rama con:

```
git checkout -b nueva_rama
```

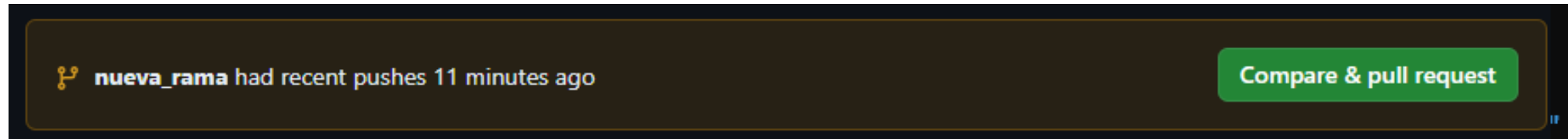
Luego publicamos el *commit* como siempre:

1. **git add .**
2. **git commit -m "Cambios experimentales"**
3. **git push -u origin nueva_rama** (o también podemos solamente escribir '**git pull**' dejando la rama a la que publicar implícita, debido a que ya se seleccionó la rama actual con el comando *checkout*)

Git nos advertirá de un error si se quiere realiza un git push a una nueva rama sin el parámetro `-u/--set-upstream`

```
PS C:\Users\Agustin.Marco\Projects\Azure Git Tutorial - copia> git checkout -b nueva_rama
Switched to a new branch 'nueva_rama'
PS C:\Users\Agustin.Marco\Projects\Azure Git Tutorial - copia> git add .
PS C:\Users\Agustin.Marco\Projects\Azure Git Tutorial - copia> git commit -m "Cambios experimentales"
[nueva_rama b651b13] Cambios experimentales
 1 file changed, 3 insertions(+), 1 deletion(-)
PS C:\Users\Agustin.Marco\Projects\Azure Git Tutorial - copia> git push origin nueva_rama
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 321 bytes | 321.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'nueva_rama' on GitHub by visiting:
remote:   https://github.com/sad-ko/Azure-Git-Tutorial/pull/new/nueva_rama
remote:
To https://github.com/sad-ko/Azure-Git-Tutorial.git
 * [new branch]      nueva_rama -> nueva_rama
PS C:\Users\Agustin.Marco\Projects\Azure Git Tutorial - copia> |
```

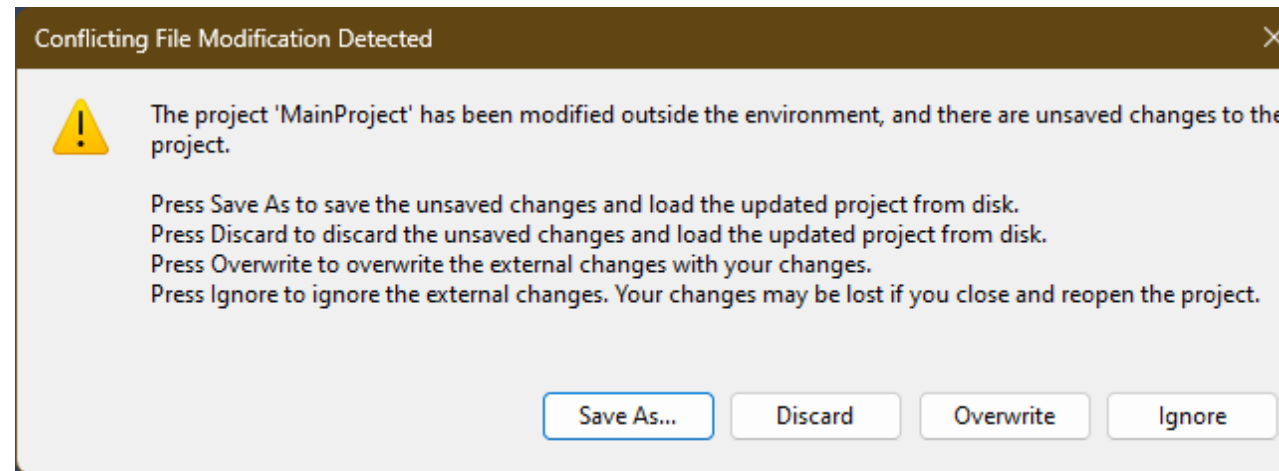
Si miramos en GitHub vamos a poder ver que se publicó el *commit* en una nueva rama dejando la rama principal intacta.



Ahora podemos crear una *pull request* y organizar con nuestros compañeros para validar los cambios e integrarlos a la rama principal.

8. Integrar git con SSMS

SSMS no incluye ningún tipo de integración con git, pero no hace falta, porque por eso aprendimos a utilizarlo desde la terminal.



Si actualizamos el repositorio local mientras tenemos SSMS abierto vamos a ver que nos advierte que se han realizado algunos cambios en los archivos del proyecto y tenemos que reiniciarlo. Descartemos la opción de guardar antes, porque eso pisaría los cambios recién bajados de GitHub.

9. Esconder archivos con git

Como pudimos ver hasta ahora, todo lo que se encuentre en nuestra carpeta del repositorio local va a ser subido al repositorio en la nube.

Pero ¿qué pasa si hay archivos que no queremos que se suba?

Como, por ejemplo, credenciales, archivos temporales o borradores.

```
PS C:\Users\Agustin.Marco\Documents\SQL Server Management Studio\GitTest> git status
On branch ssms_branch
Your branch is up to date with 'origin/ssms_branch'.

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:   .vs/GitTest/v15/.ssms_suo

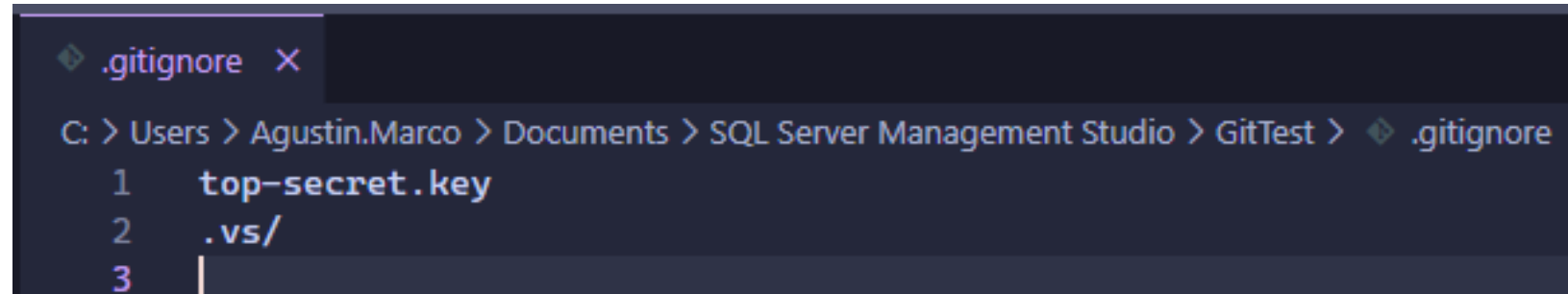
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        top-secret.key

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

Para eso git nos ofrece el archivo '.gitignore'

'`.gitignore`' es un simple archivo de texto donde se lista todos los archivos y/o carpetas que queremos dejar de seguir con git.

Basta con solo crear el archivo en la carpeta raíz del repositorio (o en la subcarpeta deseada) y escribir una lista de los archivos/carpetas que deseamos dejar de seguir con git.

A screenshot of a code editor window with a dark theme. The title bar at the top shows a file named ".gitignore" with a close button. The breadcrumb navigation at the top of the editor area shows the path: "C: > Users > Agustin.Marco > Documents > SQL Server Management Studio > GitTest > .gitignore". The main text area contains three lines of text: "1 top-secret.key", "2 .vs/", and "3" followed by a vertical cursor bar.

```
.gitignore
1 top-secret.key
2 .vs/
3
```


Bibliografía

- [git-scm -- Git Tutorial](#)
- [github -- Resolver un conflicto de fusión con la línea de comando](#)
- [sqlservercentral -- Getting Your DBA Teams Scripts into Git](#)

Fin