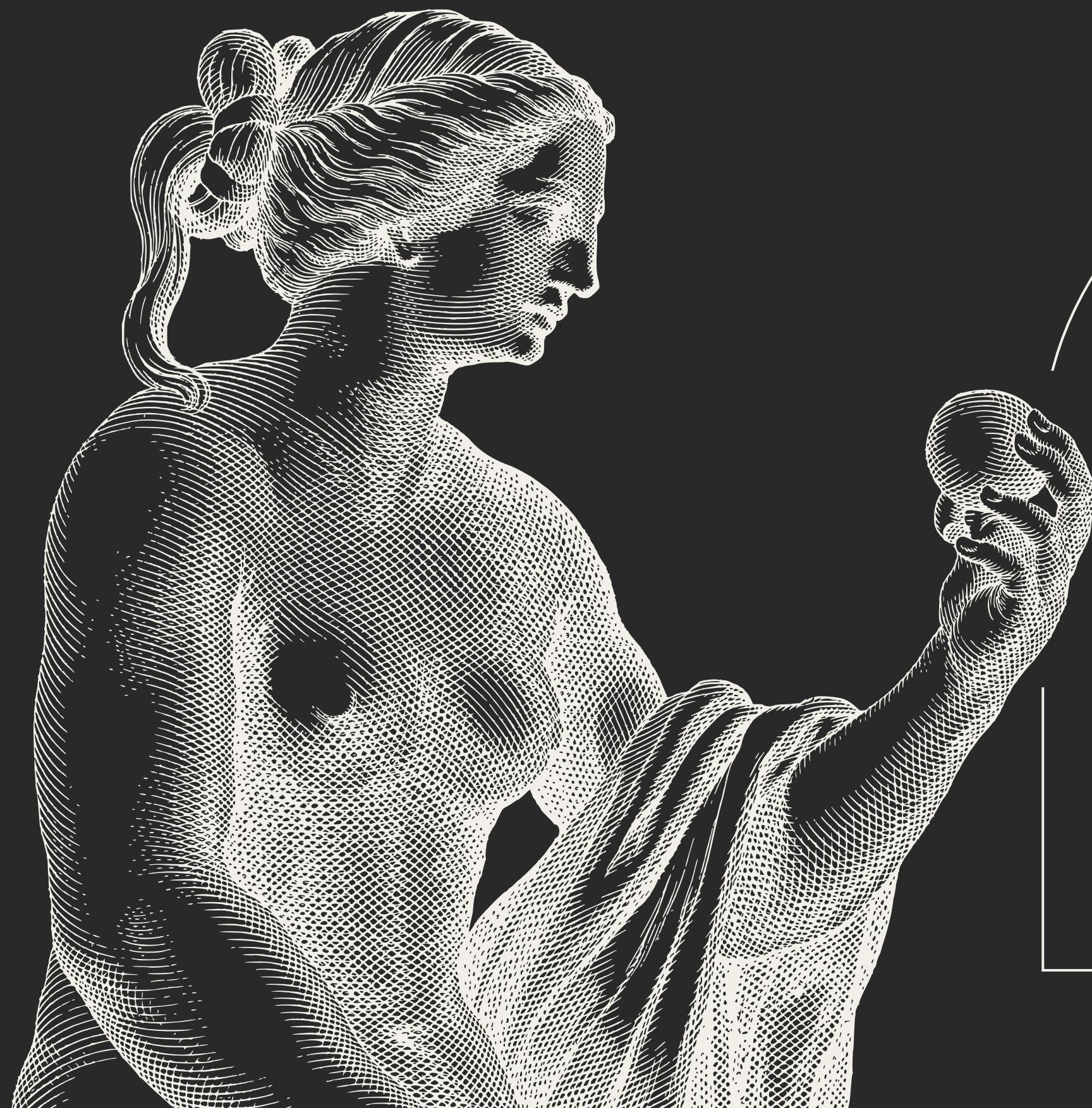


02 OCTUBRE 2024

ALEJANDRO MAX

SEMANA 5

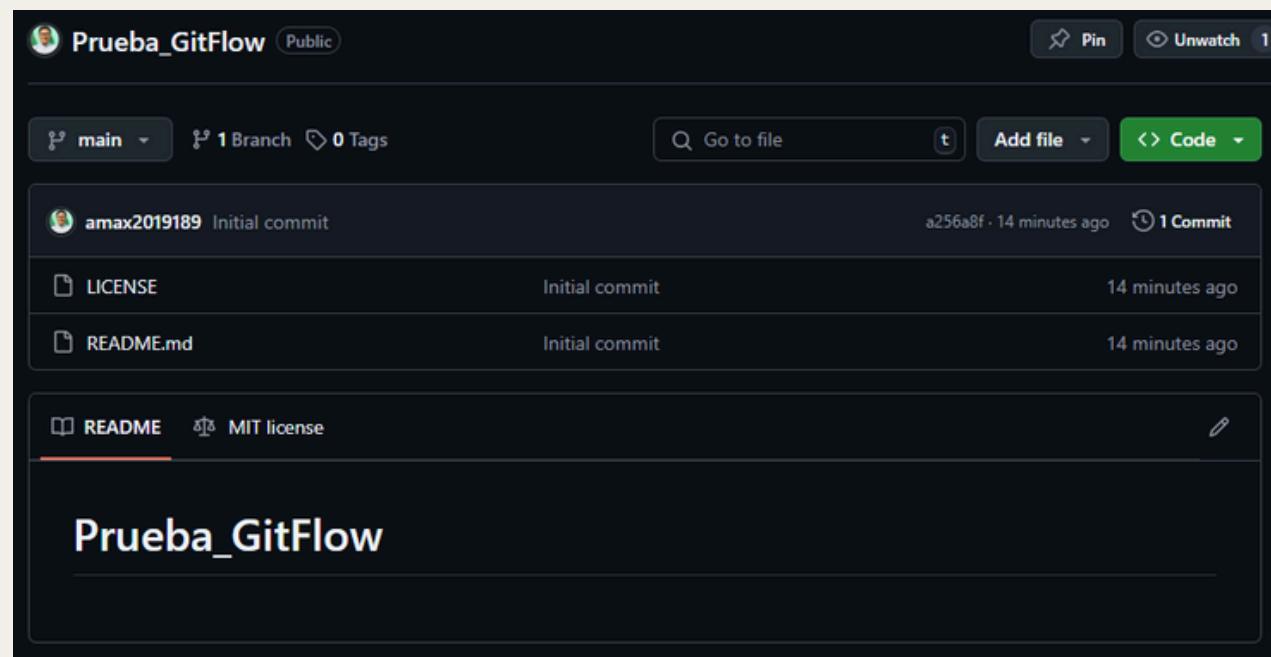


MANUAL

giffow

Trabajamos con GIT FLOW

1. Comenzaremos creando un nuevo repositorio y clonándolo desde consola como se explica en el apartado “Como utilizar comandos en Git”.



2. Después de haber clonado he ingresado a la carpeta inicializaremos Git Flow con el siguiente comando.

“git flow init”

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (main)
$ git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Prueba_GitFlow/.git/hooks]
```

3. Al momento de inicializar con esta estructura, se nos crearan dos ramas por defecto las cuales son las que podemos observar a continuación.

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (main)
$ git branch
  develop
* main
```

4. A continuación nos cambiaremos de rama con el siguiente comando.

“git checkout <nombre de la rama>”

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (main)
$ git checkout develop
Switched to branch 'develop'

alejo@Alejandro MINGW64 /c/Prueba_GitFlow (develop)
$ |
```

5. Después crearemos una rama feature para realizar los cambios que deseemos. Con el siguiente comando realizaremos la rama.

“git flow feature start <nombre de la rama>”

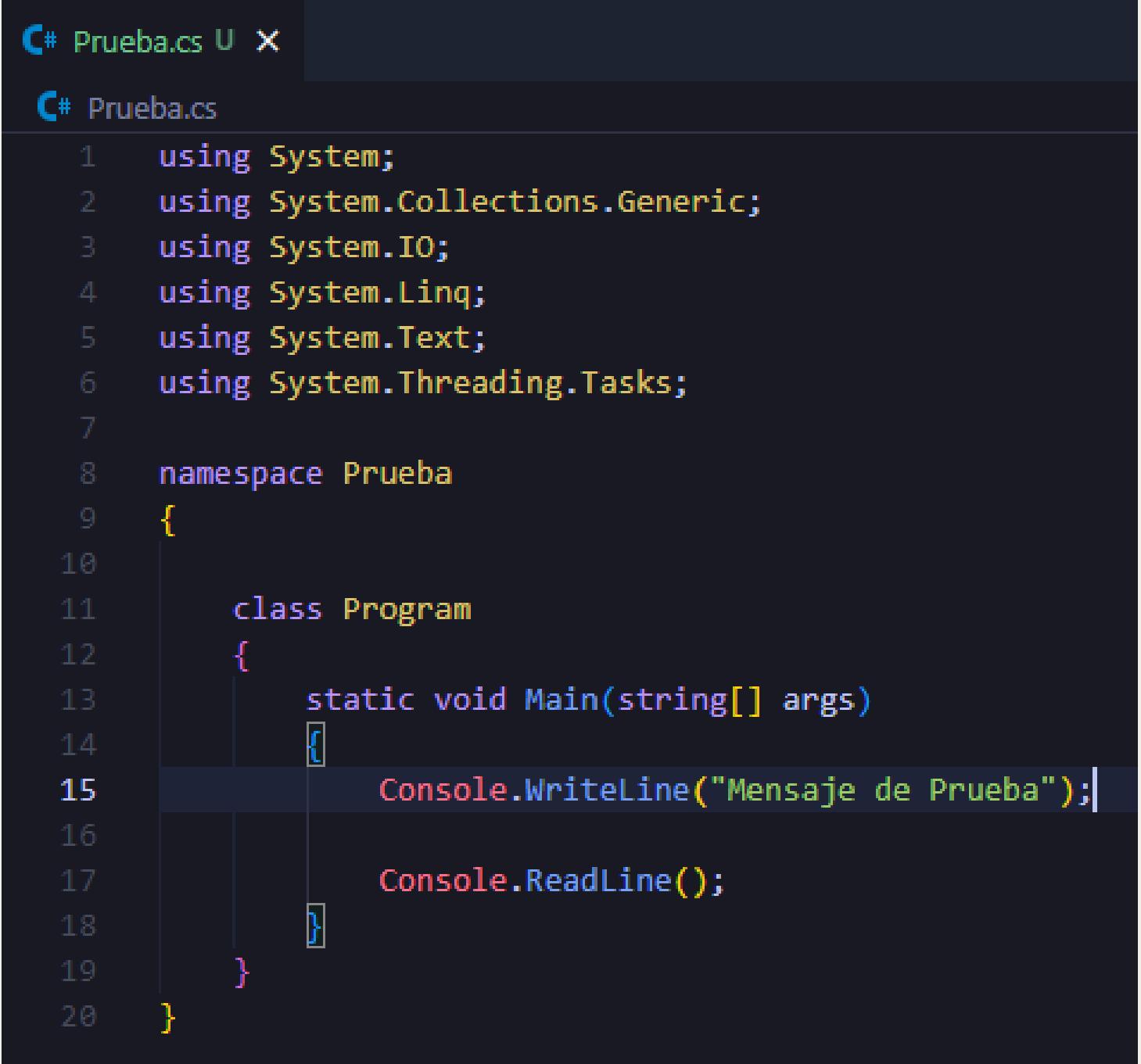
```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (develop)
$ git flow feature start Prueba
Switched to a new branch 'feature/Prueba'
```

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (feature/Prueba)
$ |
```

6. Ahora agregaremos un archivo para demostrar como funcionan las fusiones de ramas.

“code .” - Comando para abrir nuestro editor de código por defecto

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (feature/Prueba)
$ code .
```



```
C# Prueba.cs U X
C# Prueba.cs
 1  using System;
 2  using System.Collections.Generic;
 3  using System.IO;
 4  using System.Linq;
 5  using System.Text;
 6  using System.Threading.Tasks;
 7
 8  namespace Prueba
 9  {
10
11      class Program
12      {
13          static void Main(string[] args)
14          {
15              Console.WriteLine("Mensaje de Prueba");
16
17              Console.ReadLine();
18          }
19      }
20  }
```

7. Agregamos todos los archivos nuevos o modificados con los comandos correspondientes.

“git status” - Observar cambios o estado

“git add .” Guardar todos los cambios realizados a uno o varios archivos

“git commit -m ‘mensaje’ ” - Crear un identificador de commits

“git flow feature finish <nombre de la rama>” - Realiza la fusión de la rama

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (feature/Prueba)
$ git status
On branch feature/Prueba
Untracked files:
  (use "git add <file> ..." to include in what will be committed)
    Prueba.cs

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

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (feature/Prueba)
$ git add .
```

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (feature/Prueba)
$ git commit -m 'Cambios Practica'
[feature/Prueba 0fd9e25] Cambios Practica
  1 file changed, 20 insertions(+)
  create mode 100644 Prueba.cs
```

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (feature/Prueba)
$ git flow feature finish Prueba
Switched to branch 'develop'
Updating a256a8f..0fd9e25
Fast-forward
  Prueba.cs | 20 ++++++=====
  1 file changed, 20 insertions(+)
  create mode 100644 Prueba.cs
Deleted branch feature/Prueba (was 0fd9e25).
```

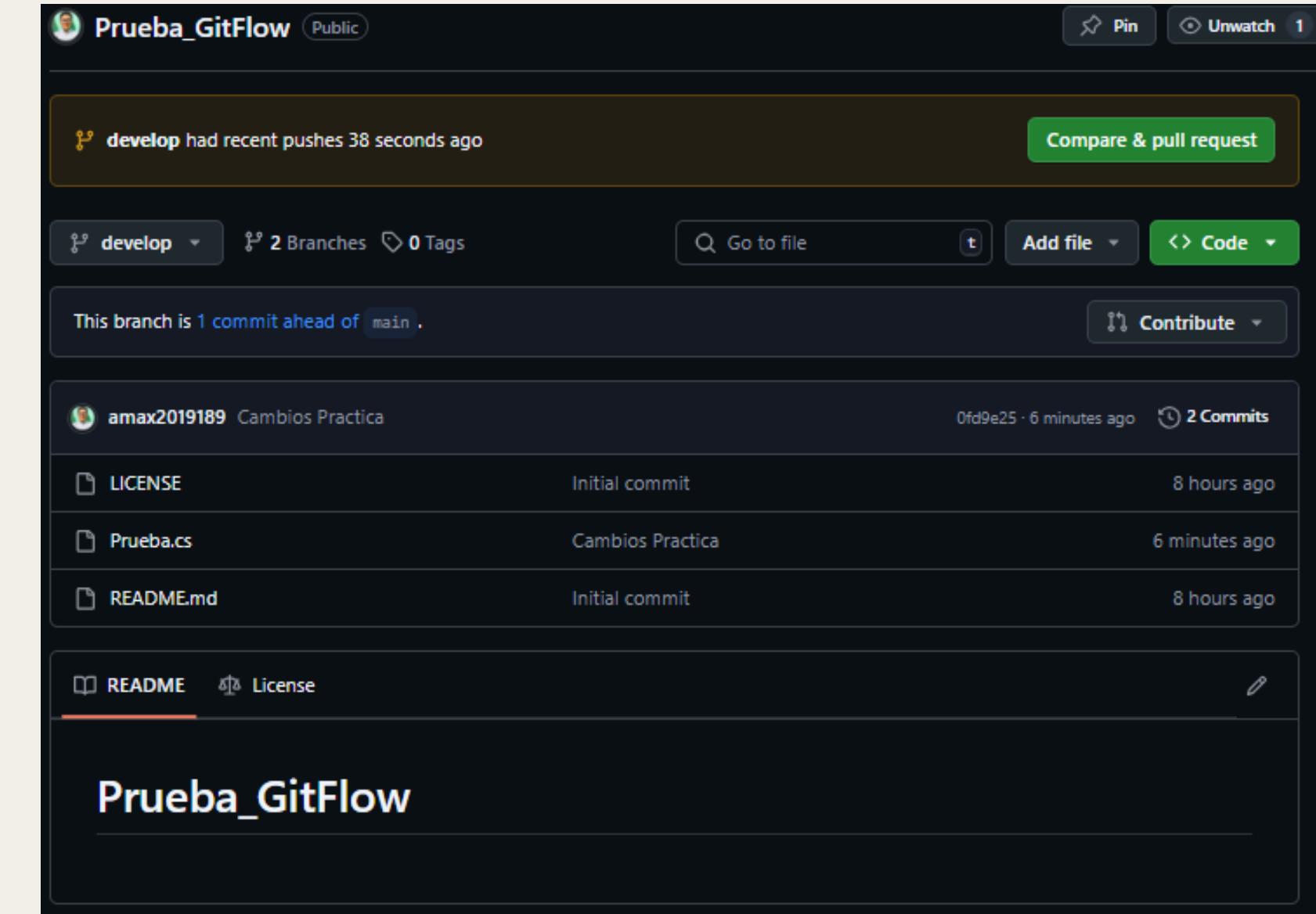
Summary of actions:

- The feature branch 'feature/Prueba' was merged into 'develop'
- Feature branch 'feature/Prueba' has been locally deleted
- You are now on branch 'develop'

8. Usando el siguiente comando subiremos nuestros cambios a nuestro repositorio en GitHub.

“git push origin <nombre de la rama>”

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (develop)
$ git push origin develop
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 496 bytes | 496.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:     https://github.com/amax2019189/Prueba_GitFlow/pull/new/develop
remote:
To https://github.com/amax2019189/Prueba_GitFlow.git
 * [new branch]      develop -> develop
```



9. Para crear una rama Release es muy similar por lo que utilizaremos los siguientes comandos.

“git flow release start <versión>”

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (develop)
$ git flow release start v1.0.0
Switched to a new branch 'release/v1.0.0'

Summary of actions:
- A new branch 'release/v1.0.0' was created, based on 'develop'
- You are now on branch 'release/v1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish 'v1.0.0'

alejo@Alejandro MINGW64 /c/Prueba_GitFlow (release/v1.0.0)
```

10. Con los siguientes comandos abriremos el editor de código y modificaremos el código.

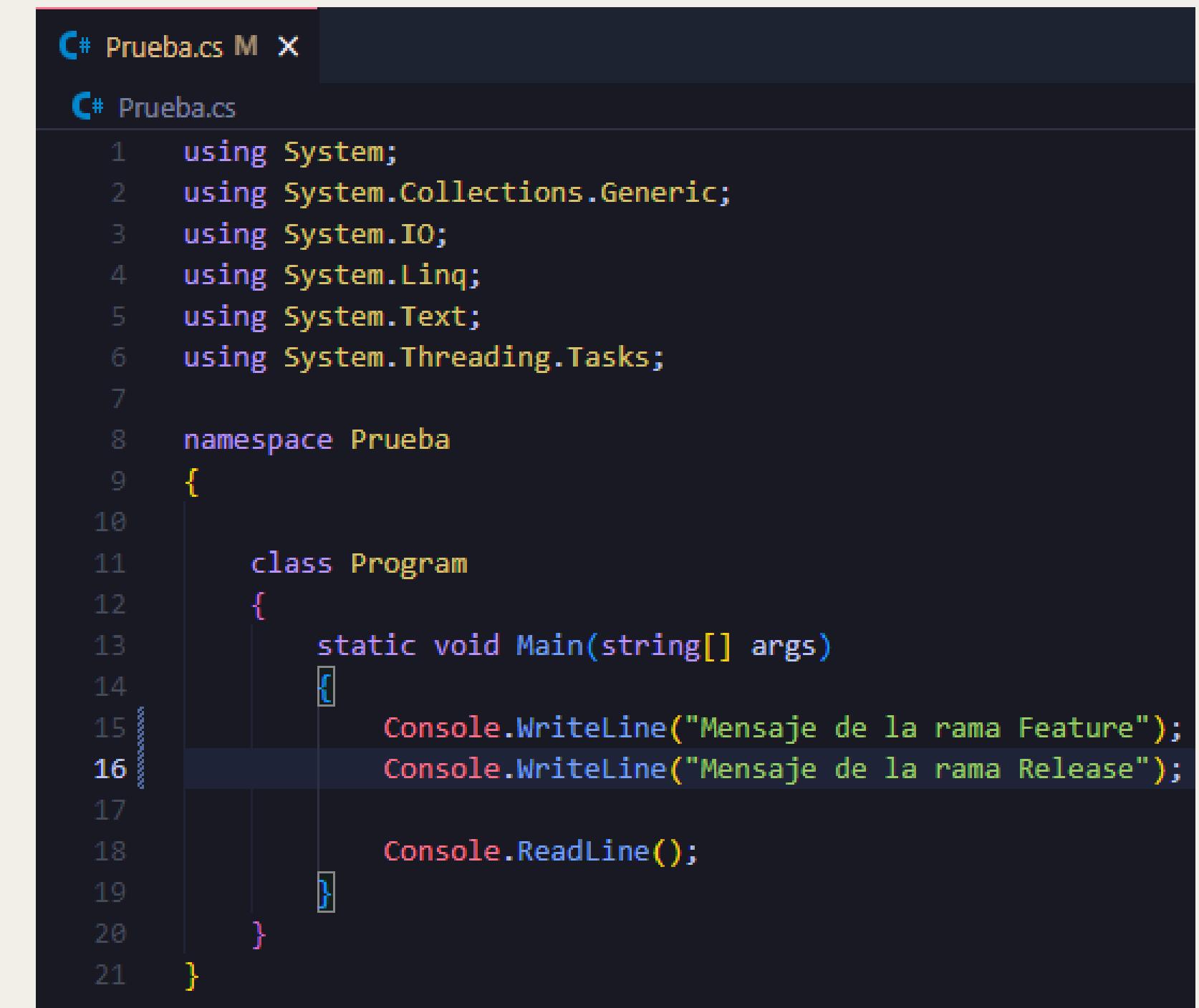
“git code .”

“git status”

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (release/v1.0.0)
$ code .
```

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (release/v1.0.0)
$ git status
On branch release/v1.0.0
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:   Prueba.cs

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



The screenshot shows a code editor window with a dark theme. The title bar says 'C# Prueba.cs M X'. The code itself is:

```
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace Prueba
9  {
10
11     class Program
12     {
13         static void Main(string[] args)
14         {
15             Console.WriteLine("Mensaje de la rama Feature");
16             Console.WriteLine("Mensaje de la rama Release");
17
18             Console.ReadLine();
19         }
20     }
21 }
```

11. Agregamos todos los archivos nuevos o modificados con los comandos correspondientes.

“git add .” Guardar todos los cambios realizados a uno o varios archivos

“git commit -m ‘mensaje’ ” - Crear un identificador de commits

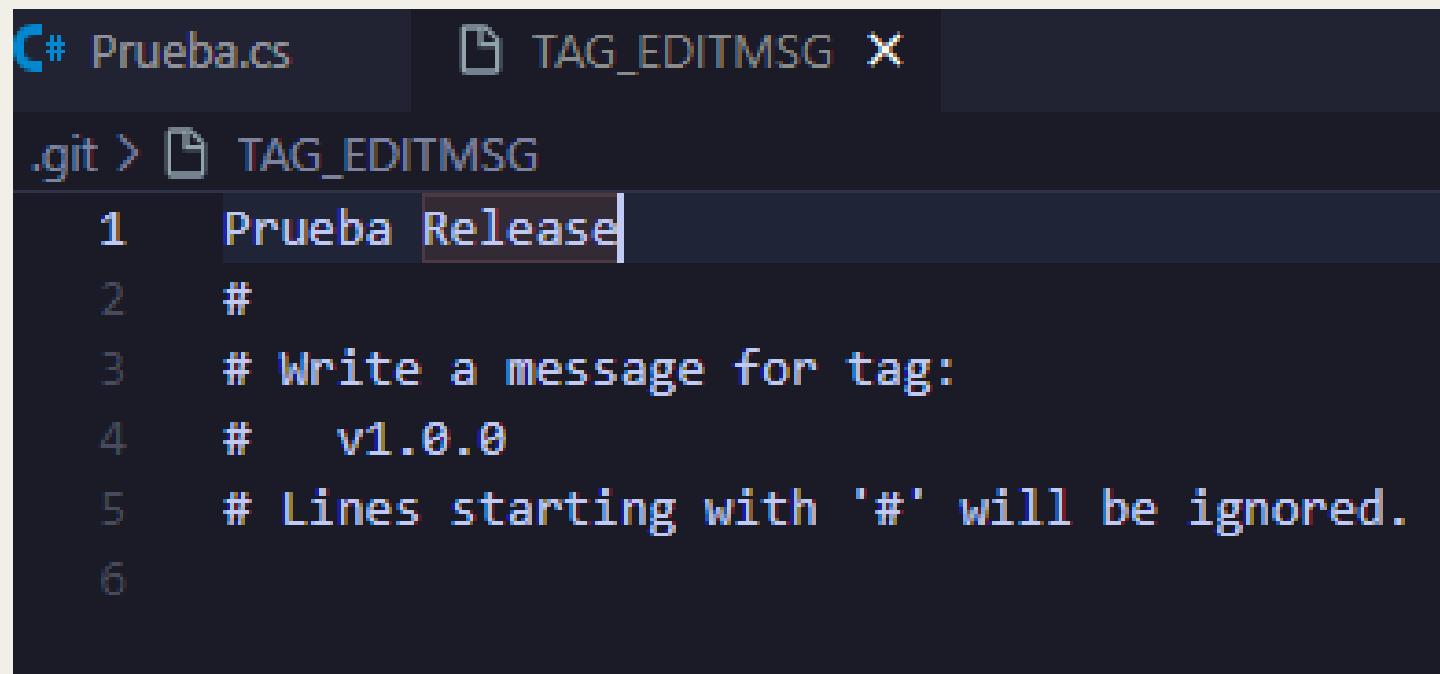
“git flow release finish <nombre de la rama>” - Realiza la fusión de la rama

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (release/v1.0.0)
$ git add .
```

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (release/v1.0.0)
$ git commit -m 'Cambios Release'
[release/v1.0.0 519e583] Cambios Release
 1 file changed, 2 insertions(+), 1 deletion(-)
```

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (release/v1.0.0)
$ git flow release finish v1.0.0
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
Merge made by the 'ort' strategy.
  Prueba.cs | 21 ++++++=====
  1 file changed, 21 insertions(+)
  create mode 100644 Prueba.cs
```

12. Al momento de usar el comando “git flow release finish <nombre de la rama>”, se nos abrirá nuestro editor de código por defecto y abrirá un documento el cual podemos agregar un comentario o solo podemos cerrarlo, al momento de agregar un comentario hay que guardar los cambios y cerrar el documento y en automático se finalizara todo.



```
C# Prueba.cs TAG_EDITMSG x  
.git > TAG_EDITMSG  
1 Prueba Release  
2 #  
3 # Write a message for tag:  
4 # v1.0.0  
5 # Lines starting with '#' will be ignored.  
6
```

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (release/v1.0.0)  
$ git flow release finish v1.0.0  
Switched to branch 'main'  
Your branch is up to date with 'origin/main'.  
Merge made by the 'ort' strategy.  
  Prueba.cs | 21 ++++++  
  1 file changed, 21 insertions(+)  
  create mode 100644 Prueba.cs  
Already on 'main'  
Your branch is ahead of 'origin/main' by 3 commits.  
  (use "git push" to publish your local commits)  
Switched to branch 'develop'  
Merge made by the 'ort' strategy.  
  Prueba.cs | 3 +-  
  1 file changed, 2 insertions(+), 1 deletion(-)  
Deleted branch release/v1.0.0 (was 519e583).  
  
Summary of actions:  
- Release branch 'release/v1.0.0' has been merged into 'main'  
- The release was tagged 'v1.0.0'  
- Release tag 'v1.0.0' has been back-merged into 'develop'  
- Release branch 'release/v1.0.0' has been locally deleted  
- You are now on branch 'develop'
```

13. Usando el siguiente comando subiremos nuestros cambios a nuestro repositorio en GitHub.

“git push origin <nombre de la rama>”

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (develop)
$ git push origin develop
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 690 bytes | 690.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/amax2019189/Prueba_GitFlow.git
  0fd9e25..6b5dd58  develop -> develop
```

The screenshot shows a GitHub repository page for 'Prueba_GitFlow'. At the top, it says 'develop had recent pushes 2 minutes ago' with a 'Compare & pull request' button. Below that, it shows 'develop' selected, '2 Branches', and '0 Tags'. There are buttons for 'Go to file', 'Add file', and 'Code'. A message states 'This branch is 1 commit ahead of main.' On the right, there's a 'Contribute' button. The commit history lists:

- amax2019189 Merge tag 'v1.0.0' into develop · 6b5dd58 · 3 minutes ago · 5 Commits
- LICENSE Initial commit · 8 hours ago
- Prueba.cs Cambios Release · 7 minutes ago
- README.md Initial commit · 8 hours ago

At the bottom, there are tabs for 'README' and 'License', with 'README' being the active tab. The content area displays the text 'Prueba_GitFlow'.

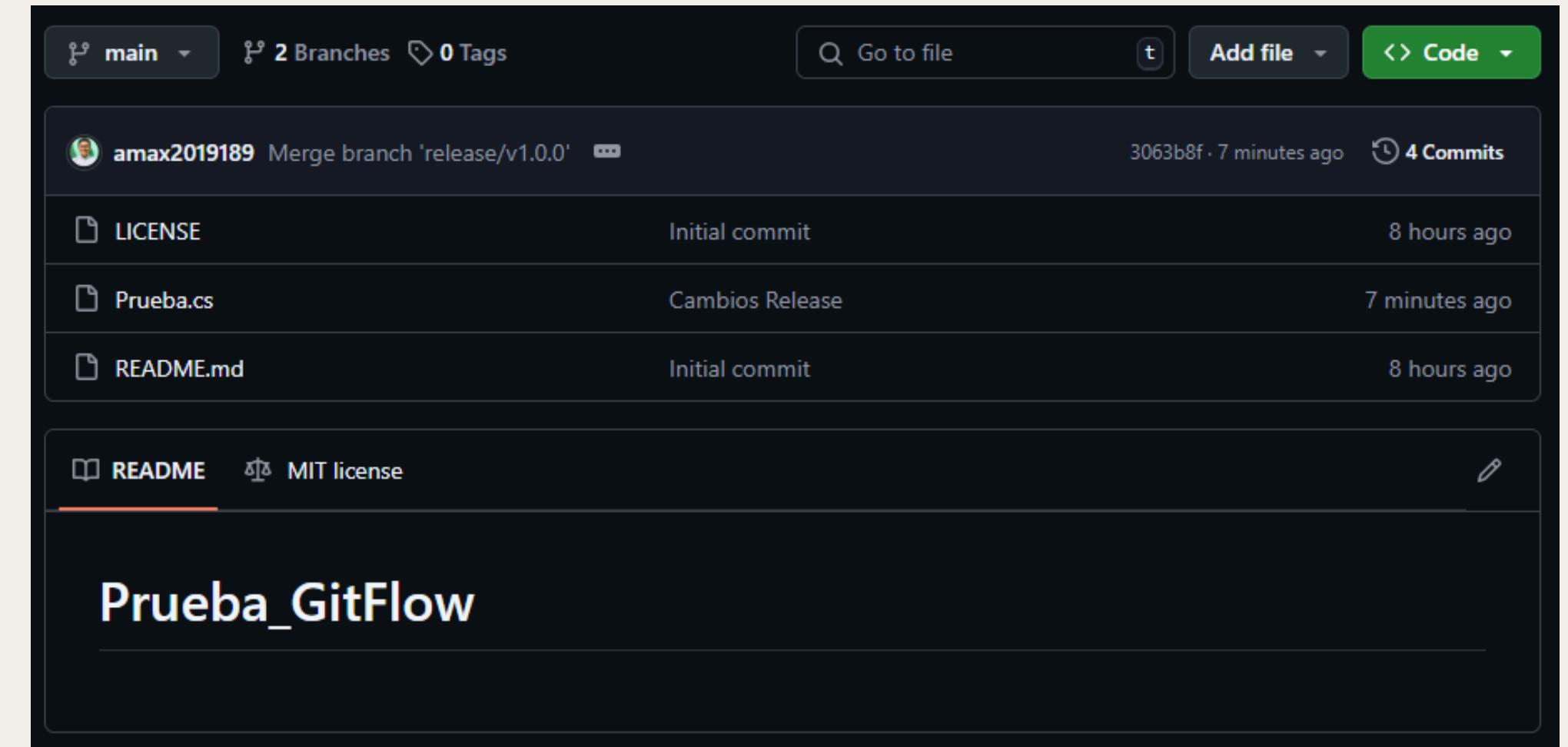
14. Nos cambiamos de rama para poder finalizar el proyecto y poder fusionar la rama develop. Utilizando los siguientes comandos podremos realizar lo ya mencionado.

“git checkout <nombre de la rama>”

“git push origin <nombre de la rama>”

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (develop)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)
```

```
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (main)
$ git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/amax2019189/Prueba_GitFlow.git
 a256a8f..3063b8f  main -> main
```





Comandos
BÁSICOS GIT

- git init
- git clone <url>
- git add .
- git commit -m “Texto”
- git push origin <nombre rama>
- git push -u origin main
- git pull
- git fetch
- git log
- git remote add origin <url>

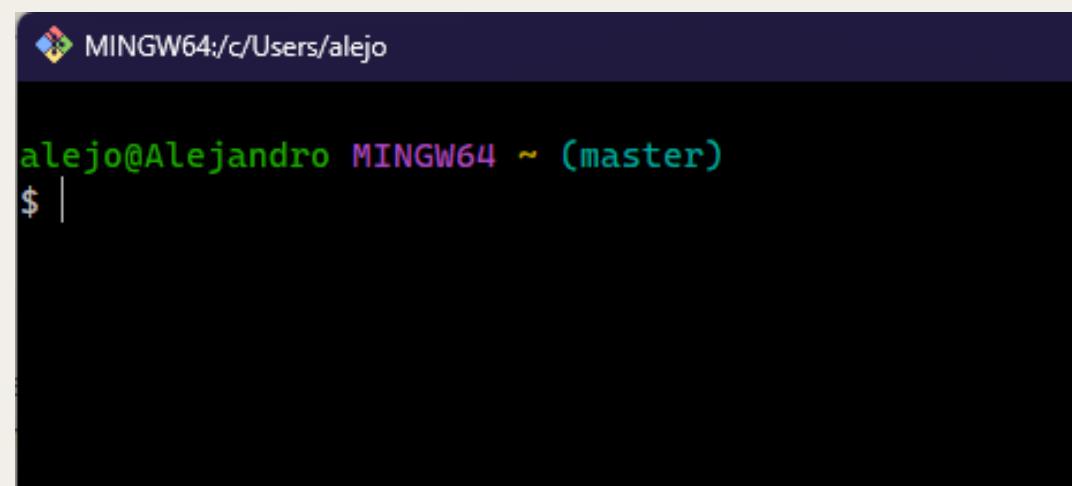
Estos son comandos básicos los cuales tenemos que tener presentes para cuando trabajamos con Git, en cualquiera de sus consolas, como lo puede ser Git Bash, Windows PowerShell, etc.

Aquí te dejo un link donde podemos encontrar más sobre los comandos que podemos utilizar en Git:

1. <https://gist.github.com/dasdo/9ff71c5c0efa037441b6>
2. <https://www.atlassian.com/es/git/glossary#commands>
3. <https://www.hostinger.es/tutoriales/comandos-de-git>

Como crear un **REPOSITORIO LOCAL**

1. Debemos iniciar abriendo la consola de nuestra preferencia, en este caso Git Bash.



```
MINGW64:/c/Users/alejo
alejo@Alejandro MINGW64 ~ (master)
$ |
```

2. Usando el comando “cd ..” llegaremos al disco local de nuestro dispositivo.



```
MINGW64:/c
alejo@Alejandro MINGW64 ~ (master)
$ cd ..

alejo@Alejandro MINGW64 /c/Users
$ cd ..

alejo@Alejandro MINGW64 /c
$ |
```

3. A continuación debemos crear una carpeta con el nombre que nosotros queramos.



4. Usando el comando “cd Prueba_GIT” para ingresar a la carpeta.

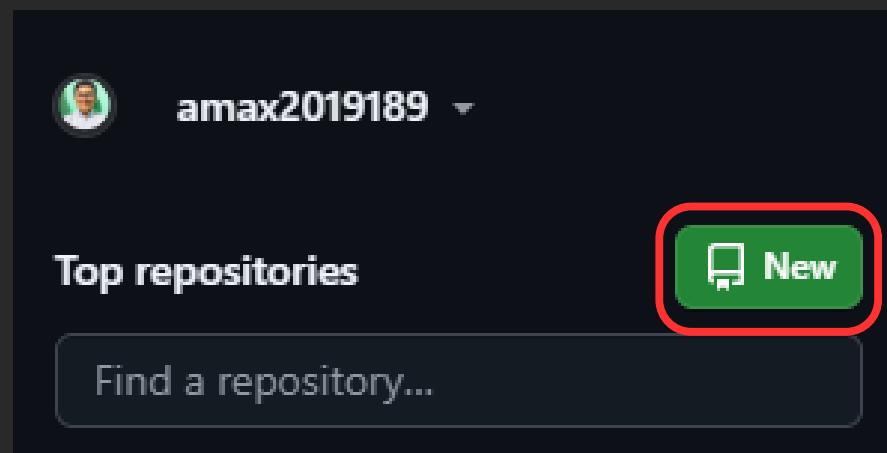
```
alejo@Alejandro MINGW64 /c  
$ cd Prueba_GIT  
  
alejo@Alejandro MINGW64 /c/Prueba_GIT  
$
```

5. Usando el comando “git init” iniciamos el repositorio local. Ya estamos listos para agregar todos los archivos con los que trabajaremos.

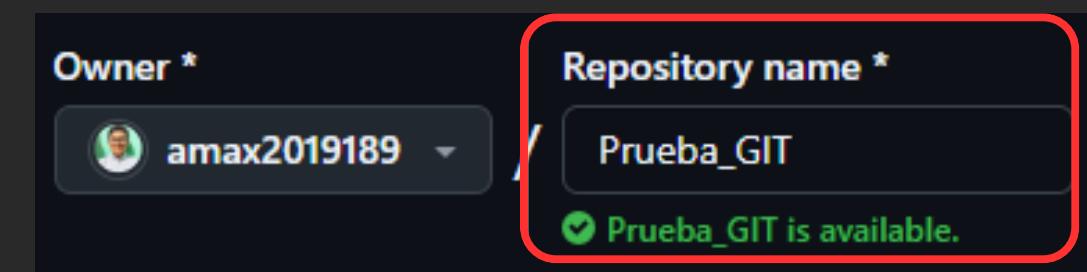
```
alejo@Alejandro MINGW64 /c/Prueba_GIT  
$ git init  
Initialized empty Git repository in C:/Prueba_GIT/.git/  
  
alejo@Alejandro MINGW64 /c/Prueba_GIT (master)  
$ |
```

Como crear un REPOSITORIO EN GITHUB

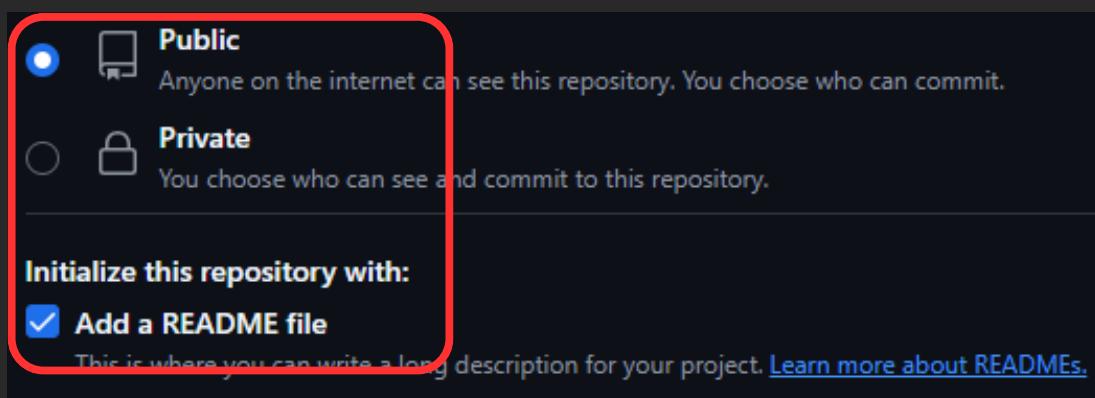
1. Debemos ingresar a la pagina de GitHub y en la pantalla inicial presionar el siguiente botón.



2. Después debemos darle un nombre a nuestro repositorio.



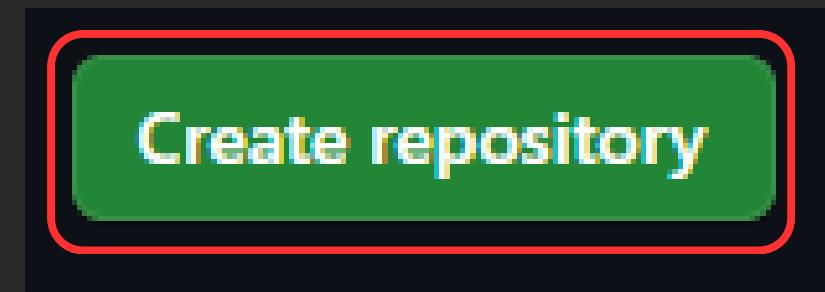
3. A continuación verificamos si el repositorio queremos que sea publico o privado y seleccionamos la opción a nuestra elección.



4. Luego de los pasos anteriores agregamos la licencia de nuestra preferencia.

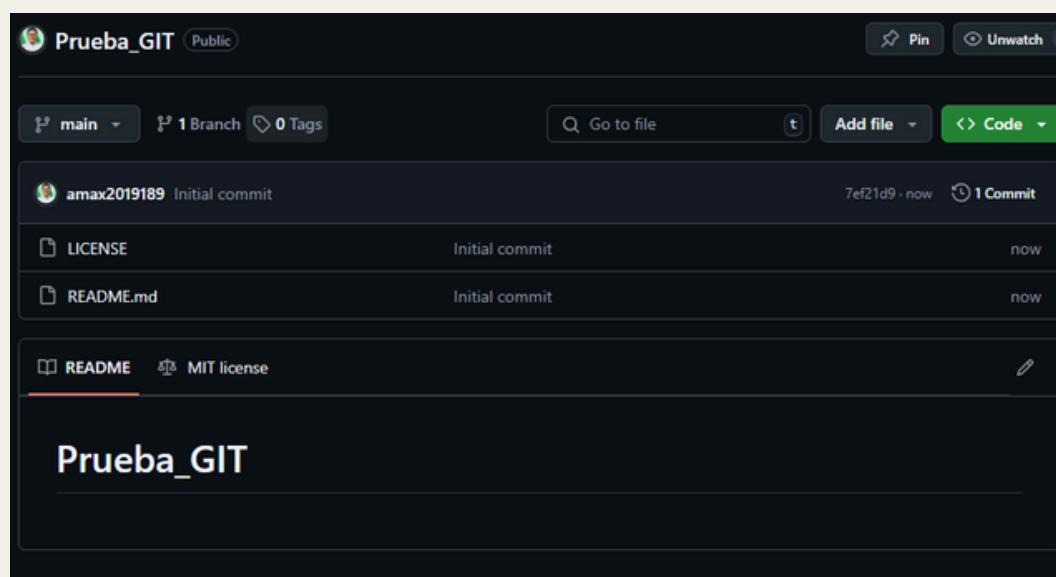


5. Para finalizar bajamos y damos clic en el botón (Create repository).

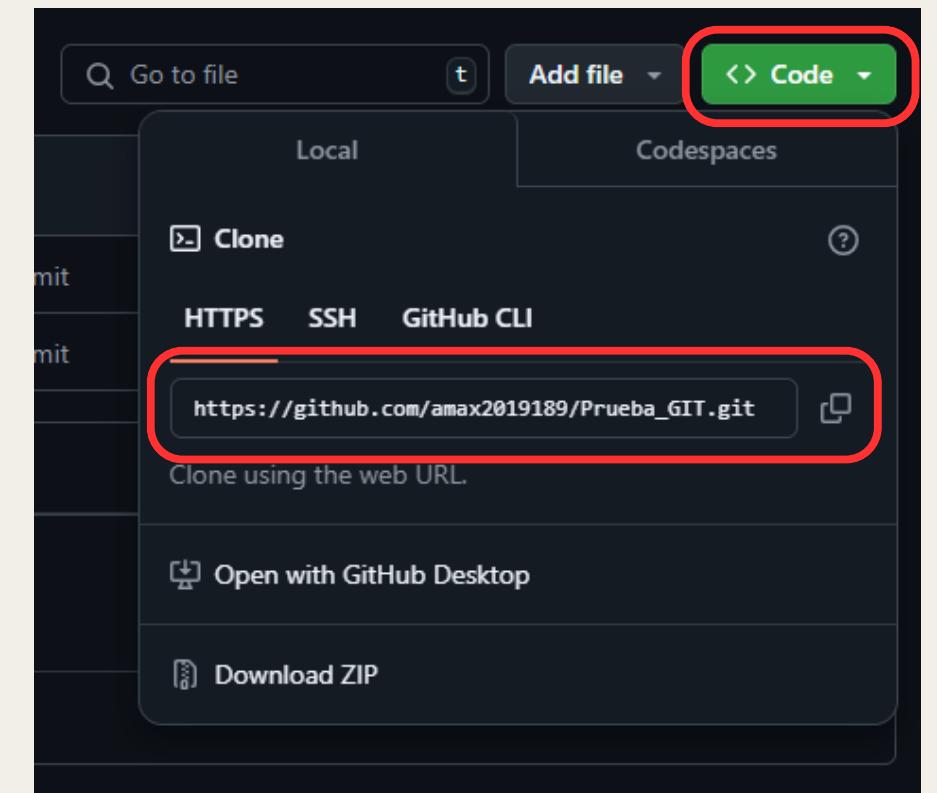


Como conectar un **REPOSITORIO LOCAL**

1. Como primer paso debemos crear un repositorio en GitHub de preferencia con el mismo nombre del proyecto.



2. Obtenemos la URL del repositorio en el botón de (<> Code) y la copiamos ya que nos servirá mas adelante.



3. Continuaremos creando e inicializando el repositorio local explicado en los incisos anteriores. He ingresamos a la carpeta desde la consola.

```
alejo@Alejandro MINGW64 /c/Prueba_GIT
$ git init
Initialized empty Git repository in C:/Prueba_GIT/.git/
alejo@Alejandro MINGW64 /c/Prueba_GIT (master)
$ |
```

4. Usando el comando propuesto más adelante conectaremos nuestro repositorio local con el repositorio en GitHub.

“git remote add origin <URL>”

```
alejo@Alejandro MINGW64 /c/Prueba_GIT (master)
$ git remote add origin https://github.com/amax2019189/Prueba_GIT.git
```

5. A continuación usando el comando propuesto más adelante para verificar que nuestra conexión se haya realizado exitosamente.

```
alejo@Alejandro MINGW64 /c/Prueba_GIT (master)
$ git remote -v
origin  https://github.com/amax2019189/Prueba_GIT.git (fetch)
origin  https://github.com/amax2019189/Prueba_GIT.git (push)
```

6. Al inicializar el repositorio local se nos crea con la rama (Master) debemos renombrarla con el siguiente comando.

“git branch -M main”

```
alejo@Alejandro MINGW64 /c/Prueba_GIT (master)
$ git branch -M main

alejo@Alejandro MINGW64 /c/Prueba_GIT (main)
$
```

7. Ahora que ya la rama tiene el mismo nombre del repositorio en GitHub podemos usar el comando dado a continuación para subir los cambios realizados.

```
alejo@Alejandro MINGW64 /c/Prueba_GIT (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 238 bytes | 238.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/amax2019189/Prueba_GIT.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Como agregar ARCHIVOS EN GITHUB

1. Cuando tenemos que agregar un documento, lo agregamos en el repositorio local.

📁 .git	2/10/2024 21:24	Carpeta de archivos
📄 Prueba	2/10/2024 20:46	Documento de te... 1 KB
📄 Prueba 1	2/10/2024 21:18	Documento de te... 1 KB
📄 Prueba 2	2/10/2024 21:18	Documento de te... 1 KB

2. En este momento tenemos que usar el siguiente comando para guardar los cambios.
“git status” - Para ver el estado de los cambios
“git add .” - Para guardar los cambios

```
alejo@Alejandro MINGW64 /c/Prueba_GIT (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file> ..." to include in what will be committed)
    Prueba 1.txt
    Prueba 2.txt

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

```
alejo@Alejandro MINGW64 /c/Prueba_GIT (main)
$ git add .
```

```
alejo@Alejandro MINGW64 /c/Prueba_GIT (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file> ..." to unstage)
    new file:   Prueba 1.txt
    new file:   Prueba 2.txt
```

3. Para continuar debemos usar el siguiente comando para crear un punto de referencia el cual quedara registrado.

“git commit -m ‘Mensaje’ ”

```
alejo@Alejandro MINGW64 /c/Prueba_GIT (main)
$ git commit -m 'Primeros cambios'
[main 6e8953f] Primeros cambios
 2 files changed, 2 insertions(+)
 create mode 100644 Prueba 1.txt
 create mode 100644 Prueba 2.txt
```

4. Ahora, si realizamos un nuevo cambio se registrara y necesitaremos realizar los mismos pasos anteriores para conservar los cambios. Cuando deseemos ver todos los commits realizados utilizaremos el siguiente comando el cual mostrara todos los commits realizados.

“git log”

```
alejo@Alejandro MINGW64 /c/Prueba_GIT (main)
$ git log
commit 6e8953fe73bef6a52ada2ab8a83823c1ab2def00 (HEAD -> main)
Author: amax2019189 <amax-2019189@kinal.edu.gt>
Date:   Wed Oct 2 21:31:49 2024 -0600

  Primeros cambios

commit 9141fbff075cae899b228e50cd37a7dbc07bf79e (origin/main)
Author: amax2019189 <amax-2019189@kinal.edu.gt>
Date:   Wed Oct 2 20:46:32 2024 -0600

  Primer Cambio
```

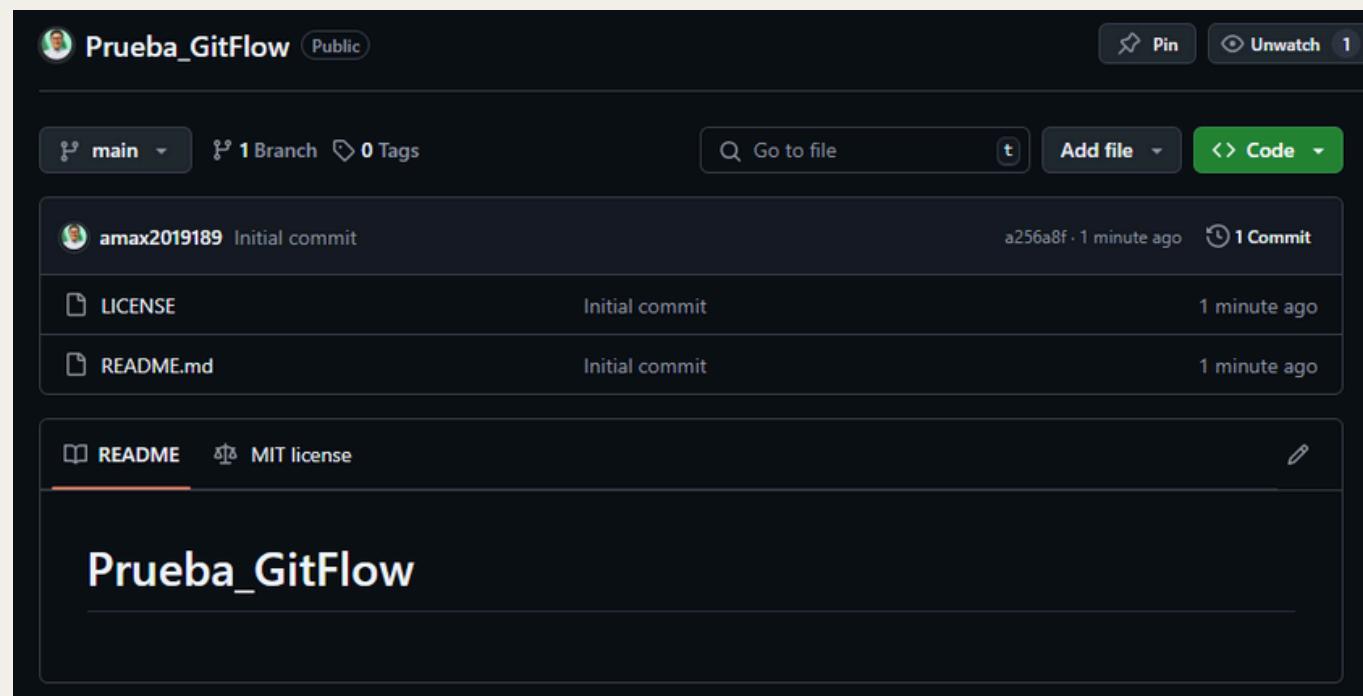
5. Para enviar todos los cambios al repositorio en GitHub usaremos el siguiente comando.

“git push origin <rama>”

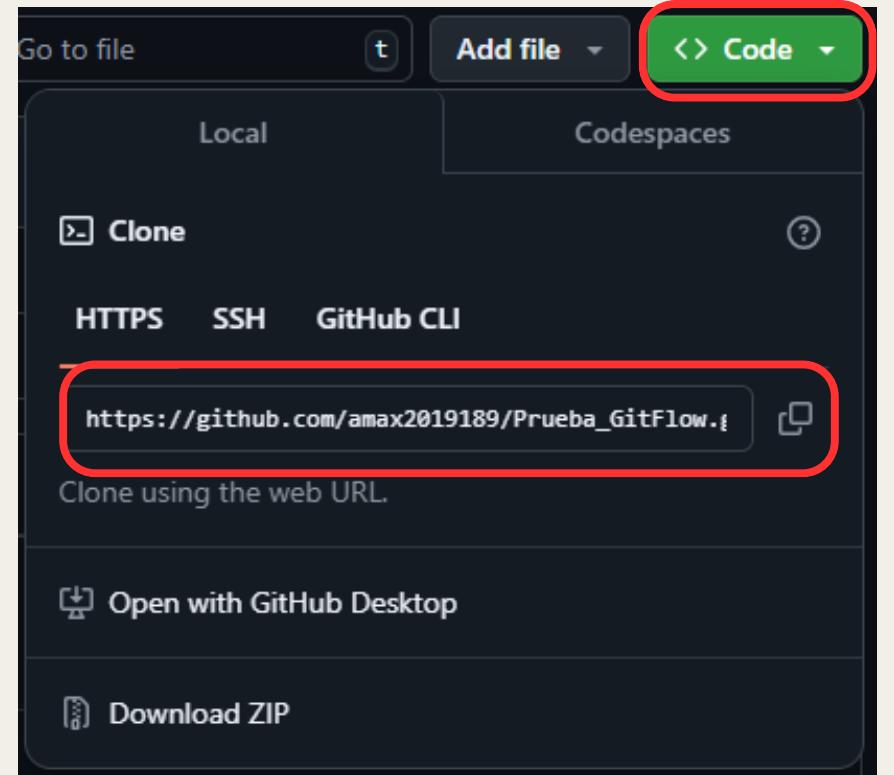
```
alejo@Alejandro MINGW64 /c/Prueba_GIT (main)
$ git push origin main
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 353 bytes | 353.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/amax2019189/Prueba_GIT.git
  9141fbf..6e8953f  main -> main
```

Como clonar un **REPOSITORIO**

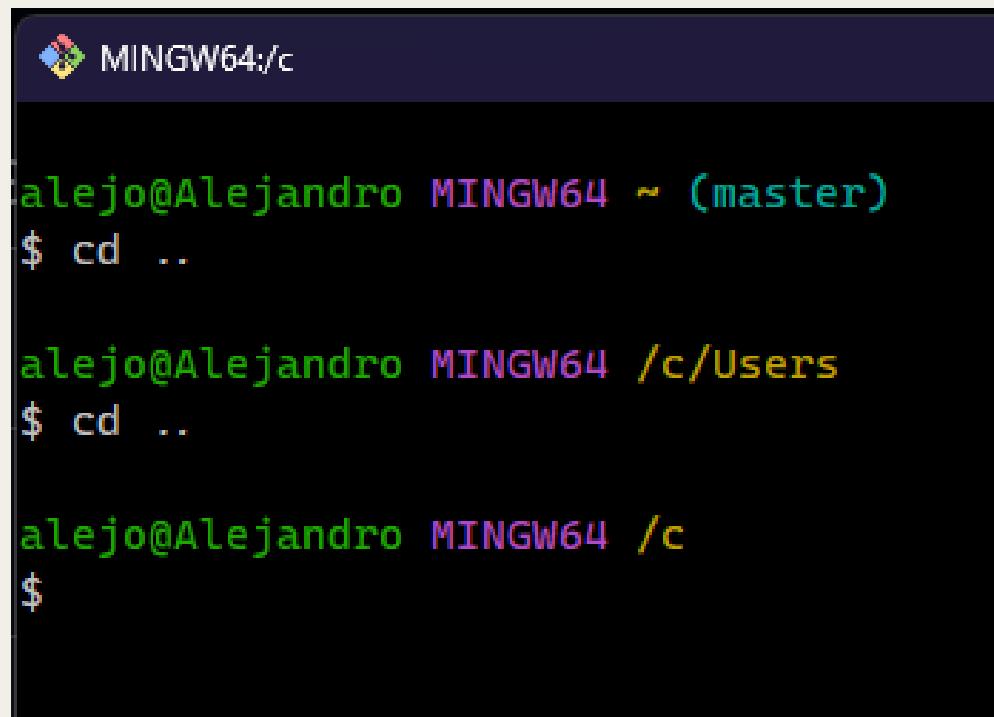
1. Iniciaremos creando nuestro repositorio en GitHub con las configuraciones de nuestra preferencia.



2. Obtenemos la URL del repositorio en el botón de (<> Code) y la copiamos ya que nos servirá mas adelante.



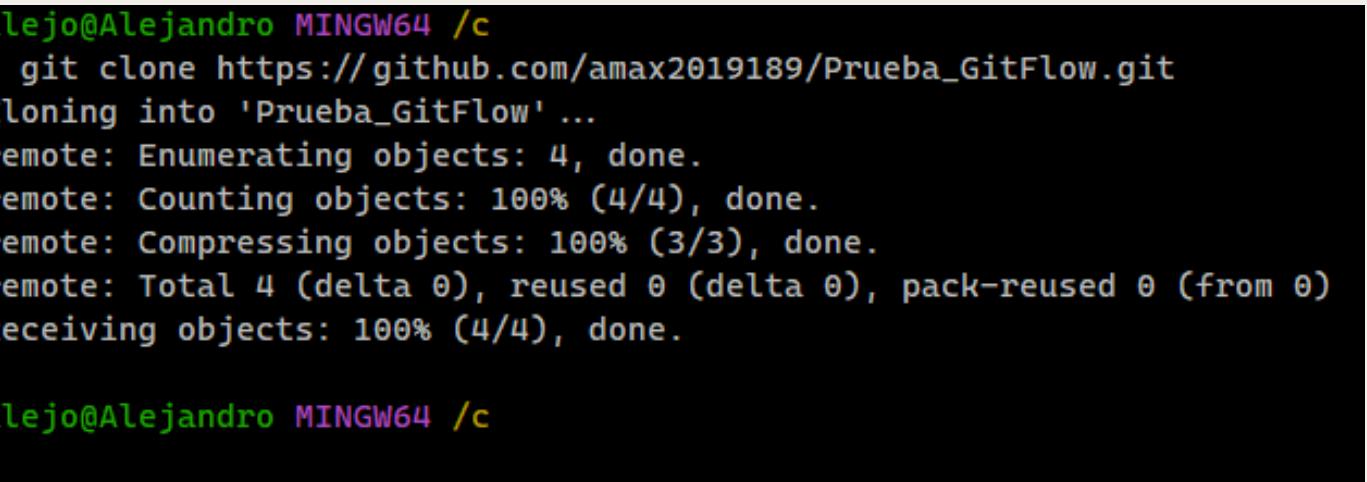
3. Abriremos nuestra consola de nuestra preferencia y nos dirigiremos al disco local C.



```
MINGW64:/c  
alejo@Alejandro MINGW64 ~ (master)  
$ cd ..  
  
alejo@Alejandro MINGW64 /c/Users  
$ cd ..  
  
alejo@Alejandro MINGW64 /c
```

4. Usando el siguiente comando clonaremos el repositorio en el disco.

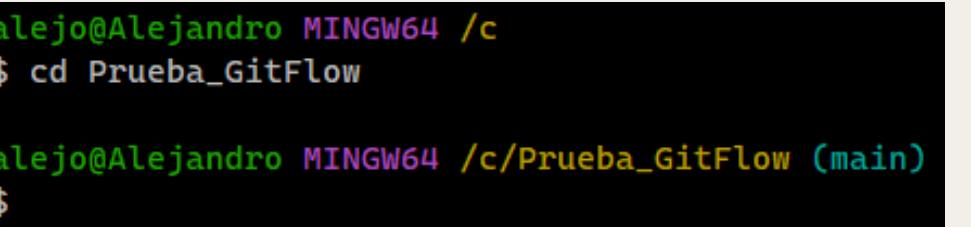
“git clone <URL>”



```
alejo@Alejandro MINGW64 /c  
$ git clone https://github.com/amax2019189/Prueba_GitFlow.git  
Cloning into 'Prueba_GitFlow' ...  
remote: Enumerating objects: 4, done.  
remote: Counting objects: 100% (4/4), done.  
remote: Compressing objects: 100% (3/3), done.  
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)  
Receiving objects: 100% (4/4), done.  
  
alejo@Alejandro MINGW64 /c
```

5. Verificamos si lo clono y podemos entrar a la carpeta con el siguiente comando.

“cd <nombre de la carpeta>”



```
alejo@Alejandro MINGW64 /c  
$ cd Prueba_GitFlow  
  
alejo@Alejandro MINGW64 /c/Prueba_GitFlow (main)
```

Muchas
GRACIAS

