

## Práctico 2: Git y GitHub

### Objetivo:

#### Actividades

- 1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

- ¿Qué es GitHub?  
Es una web o plataforma similar a una nube creada por Git donde almacena y comparte proyecto de desarrollo.
- ¿Cómo crear un repositorio en GitHub?  
ingresar a: github.com, hacer login, clic en signo + ó Create new (arriba ala derecha) seleccionar Repositorio, completar los datos y crear.
- ¿Cómo crear una rama en Git?  
Abrir la consola de Bash dentro de la carpeta del proyecto o Terminal en VS Code, escribir el siguiente comando: `git branch nombre-de-la-rama`
- ¿Cómo cambiar a una rama en Git?  
Verificar en que rama se encuentra actualmente con el siguiente comando: `git branch`, luego escribir el siguiente comando para hacer el cambio: `git checkout nombre-de-la-rama`
- ¿Cómo fusionar ramas en Git?  
Escribir en la consola de Bash o terminal de VS Code el siguiente comando: `git merge nombre-de-la-rama`
- ¿Cómo crear un commit en Git?  
Para actualizar los cambios hechos de manera local en el repositorio remoto escribir el siguiente código: `git commit -m "Agregá una referencia del cambio o mensaje entre comillas dobles."`
- ¿Cómo enviar un commit a GitHub?
  1. Posicionado en el archivo que se desea agregar a GitHub abrir la terminal o Bash y ejecutar el siguiente comando: `git add .` -> Para agregar el archivo Stage
  2. Escribe y ejecuta el siguiente comando: `git commit -m "mensaje."` -> Para preparar o confirmar los cambios.
  3. Escribe y ejecuta el comando: `git push` -> Para enviar los cambios a GitHub.
- ¿Qué es un repositorio remoto?  
Es un lugar disponible online, en la nube ó red donde se almacena una copia del proyecto.
- ¿Cómo agregar un repositorio remoto a Git?  
Escribe y ejecuta el comando: `git add remote` seguido del nombre que le deseas dar y luego de la url, por ejemplo, `git add remote nuevo_repo_remoto url`
- ¿Cómo empujar cambios a un repositorio remoto?  
Escribir y ejecutar el siguiente comando: `git push` seguido del nombre del repositorio remoto y si lo existiera nombre de la rama, por ejemplo, `git push nuevo_repo_remoto`

- ¿Cómo tirar de cambios de un repositorio remoto?

Para obtener los cambios de un repositorio remoto en Git, puedes escribir y ejecutar comando: `git pull`

- ¿Qué es un fork de repositorio?

Es una copia creada en mi repositorio de un repositorio publico subido por otro autor, sin que las modificaciones de estas afecten al del autor principal.

- ¿Cómo crear un fork de un repositorio?

En GitHub buscar el repositorio que deseo clonar, seleccionarlo y dentro del mismo aparecerá un menú de opciones, seleccionar Fork, completar los datos y crear la copia de este en mi repositorio.

¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

Teniendo las dos ramas a fusionar, seleccionamos la que deseamos fusionar con la principal, hacer clic en "Compare & pull request" completamos lo datos y haz clic en "Create pull request".

- ¿Cómo aceptar una solicitud de extracción?

Haz clic en la pestaña de "Your pull request" desplazarse en el menú hasta la opción Review requests, se selecciona el que se desea aceptar, revisa los cambios y, si son correctos, haz clic en "Merge pull request" confirmar el merge.

- ¿Qué es un etiqueta en Git?

Es un marcador de puntos importantes específicos del historial, generalmente usado para identificar versiones.

- ¿Cómo crear una etiqueta en Git?

Para crear etiquetas anotadas se usa el siguiente comando: `git tag -a nombre_etiqueta -m "Comentario de la etiqueta"`, por ejemplo, `git tag -a v1.0.0 -m "Versión 1.0.0"`

- ¿Cómo enviar una etiqueta a GitHub?

Una vez creada, se utiliza el comando: `git push origin [etiqueta]`, ejemplo: `git push origin v1.0.0`

- ¿Qué es un historial de Git?

Es un registro de todos los cambios realizados en un proyecto.

¿Cómo ver el historial de Git?

Para ver el historial de archivos en Git, se puede usar el comando `git log`.

- ¿Cómo buscar en el historial de Git?

Para buscar en el historial podemos digitar el comando `git log` seguido del parámetro o filtro a buscar, por ejemplo, `git log --grep="texto a buscar"` -> Buscar por mensaje de commit

¿Cómo borrar el historial de Git?

De manera local basta con eliminar la carpeta `.git` en el directorio donde se inicio o con el comando: `rm -rf .git`

- ¿Qué es un repositorio privado en GitHub?

Es un espacio donde se puede guardar código, archivos y revisiones de forma segura, donde solo el propietario tiene acceso o aquellos usuarios que tengan concedido acceso explícitamente.

- ¿Cómo crear un repositorio privado en GitHub?

Ingresa a: [github.com](https://github.com), hacer login, clic en signo + ó Create new (arriba a la derecha) seleccionar Repositorio, completar los datos de título y descripción, marca la opción "Private"

Dentro del perfil

- ¿Cómo invitar a alguien a un repositorio privado en GitHub?

Situado en el repositorio que se quiere compartir, hacer clic en configuración "setting", luego en la barra izquierda seleccionar Colaboradores "Collaborators", en el apartado Manage Access hacer clic en el botón Agregar Persona "Add people", ingresar el nombre de usuario o correo de la persona y envía la invitación.

- ¿Qué es un repositorio público en GitHub?

Es un espacio donde se puede guardar código, archivos y revisiones de forma segura, donde cualquier persona con acceso a internet puede acceder sin necesidad de permisos especiales.

- ¿Cómo crear un repositorio público en GitHub?

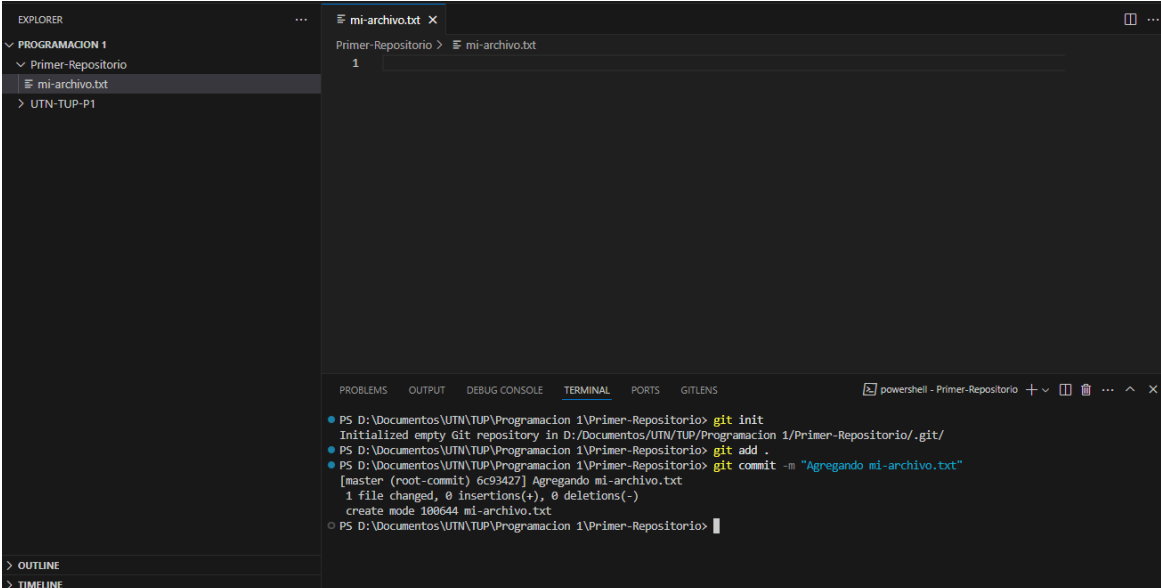
Una vez registrados/logueados en GitHub, hacer clic en el botón "New repository", completar los datos (Nombre y descripción) seleccionar "public" y crear en el botón "Create repository".

- ¿Cómo compartir un repositorio público en GitHub?

Situados en el repositorio que se desea compartir, hacer clic en botón <> Code, copiar el link y compartirlo.

2) Realizar la siguiente actividad:

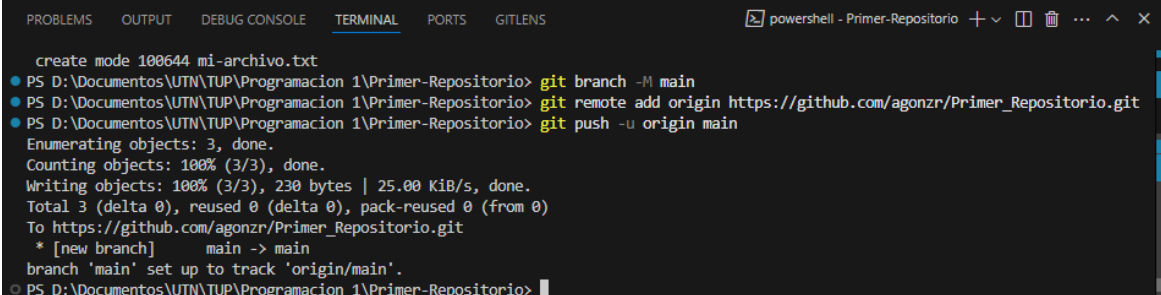
- Crear un repositorio.
  - Dale un nombre al repositorio.
  - Elije el repositorio sea público.
  - Inicializa el repositorio con un archivo.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows a project named 'PROGRAMACION 1' with a subfolder 'Primer-Repositorio' containing a file 'mi-archivo.txt'. The main editor shows the content of 'mi-archivo.txt' with the number '1'. The bottom terminal pane shows the following commands and output:

```
PS D:\Documentos\UTN\TUP\Programacion 1\Primer-Repositorio> git init
Initialized empty Git repository in D:\Documentos\UTN\TUP\Programacion 1\Primer-Repositorio\.git/
PS D:\Documentos\UTN\TUP\Programacion 1\Primer-Repositorio> git add .
PS D:\Documentos\UTN\TUP\Programacion 1\Primer-Repositorio> git commit -m "Agregando mi-archivo.txt"
[master (root-commit) 6c93427] Agregando mi-archivo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 mi-archivo.txt
PS D:\Documentos\UTN\TUP\Programacion 1\Primer-Repositorio>
```

- Agregando un Archivo
  - Crea un archivo simple, por ejemplo, "mi-archivo.txt".
  - Realiza los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"` en la línea de comandos.
  - Sube los cambios al repositorio en GitHub con `git push origin main` (o el nombre de la rama correspondiente).



The screenshot shows the Visual Studio Code terminal with the following commands and output:

```
create mode 100644 mi-archivo.txt
PS D:\Documentos\UTN\TUP\Programacion 1\Primer-Repositorio> git branch -M main
PS D:\Documentos\UTN\TUP\Programacion 1\Primer-Repositorio> git remote add origin https://github.com/agonzr/Primer_Repositorio.git
PS D:\Documentos\UTN\TUP\Programacion 1\Primer-Repositorio> git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 230 bytes | 25.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/agonzr/Primer_Repositorio.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS D:\Documentos\UTN\TUP\Programacion 1\Primer-Repositorio>
```

- Creando Branchs
  - Crear una Branch
  - Realizar cambios o agregar un archivo
  - Subir la Branch

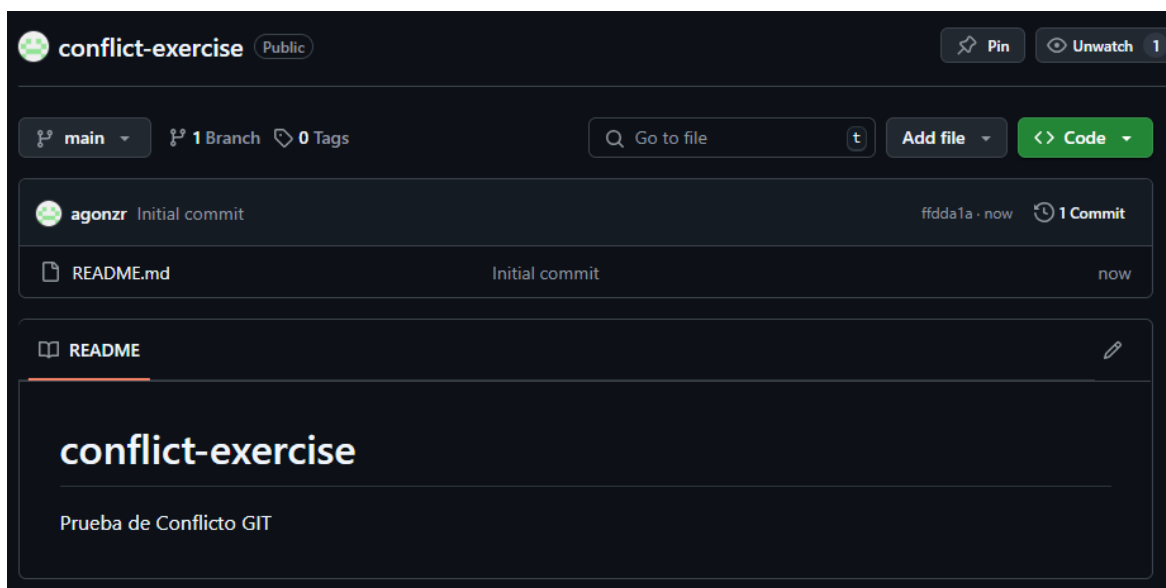
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS D:\Documentos\UTN\TUP\Programacion 1\Primer-Repositorio> git push origin update
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 300 bytes | 150.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/agonzr/Primer_Repositorio.git
   6c93427..1a54f66  update -> update
PS D:\Documentos\UTN\TUP\Programacion 1\Primer-Repositorio> |
```

3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

- Ve a GitHub e inicia sesión en tu cuenta.
- Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
- Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
- Opcionalmente, añade una descripción.
- Marca la opción "Initialize this repository with a README".
- Haz clic en "Create repository".



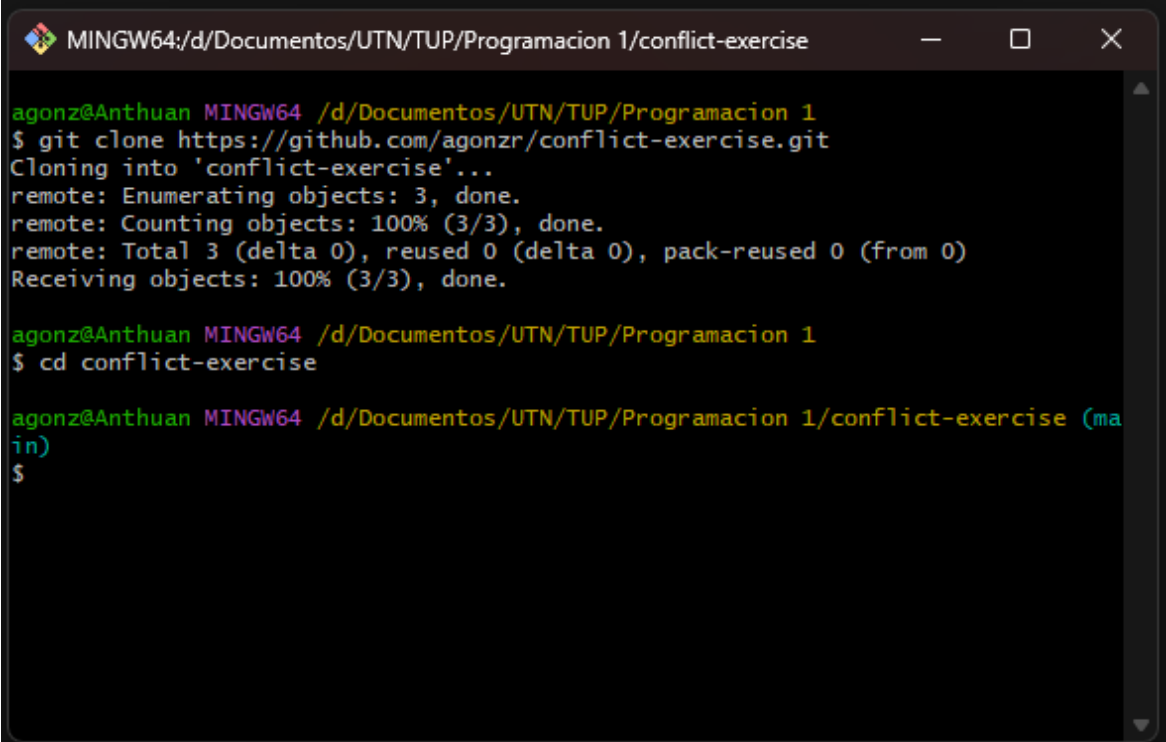
Paso 2: Clonar el repositorio a tu máquina local

- Copia la URL del repositorio (usualmente algo como `https://github.com/tuusuario/conflict-exercise.git`).
- Abre la terminal o línea de comandos en tu máquina.
- Clona el repositorio usando el comando:

```
git clone https://github.com/tuusuario/conflict-exercise.git
```

- Entra en el directorio del repositorio:

```
cd conflict-exercise
```



```
MINGW64:/d/Documentos/UTN/TUP/Programacion 1/conflict-exercise
agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1
$ git clone https://github.com/agonzr/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1
$ cd conflict-exercise

agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (main)
$
```

Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:

```
git checkout -b feature-branch
```

- Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo:

Este es un cambio en la feature branch.

- Guarda los cambios y haz un commit:

```
git add README.md
```

```
git commit -m "Added a line in feature-branch"
```

```
agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (feature-branch)
$ git add README.md

agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch ea4db4a] Added a line in feature-branch
1 file changed, 1 insertion(+)

agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (feature-branch)
$ |
```

Paso 4: Volver a la rama principal y editar el mismo archivo

- Cambia de vuelta a la rama principal (main):

```
git checkout main
```

- Edita el archivo README.md de nuevo, añadiendo una línea diferente:

Este es un cambio en la main branch.

- Guarda los cambios y haz un commit:

```
git add README.md
```

```
git commit -m "Added a line in main branch"
```

```
agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (main)
$ git add README.md

agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 045297d] Added a line in main branch
1 file changed, 1 insertion(+)

agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (main)
$ |
```

Paso 5: Hacer un merge y generar un conflicto

- Intenta hacer un merge de la feature-branch en la rama main:

```
git merge feature-branch
```

- Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md.

Paso 6: Resolver el conflicto

- Abre el archivo README.md en tu editor de texto. Verás algo similar a esto:

```
<<<<<<< HEAD
```

Este es un cambio en la main branch.

```
=====
```

Este es un cambio en la feature branch.

```
>>>>>>> feature-branch
```

- Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
- Edita el archivo para resolver el conflicto y guarda los cambios (Se debe borrar lo marcado en verde en el archivo donde estes solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).
- Añade el archivo resuelto y completa el merge:

```
git add README.md
```

```
git commit -m "Resolved merge conflict"
```

```
agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (main)
$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (main|MERGING)
$ git add README.md

agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main ca9fd55] Resolved merge conflict

agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (main)
$
```



#### Paso 7: Subir los cambios a GitHub

- Sube los cambios de la rama main al repositorio remoto en GitHub:

`git push origin main`

```
agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 755 bytes | 68.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/agonzr/conflict-exercise.git
   ffdda1a..ca9fd55  main -> main

agonz@Anthuan MINGW64 /d/Documentos/UTN/TUP/Programacion 1/conflict-exercise (main)
$
```

- También sube la feature-branch si deseas:

`git push origin feature-branch`

#### Paso 8: Verificar en GitHub

- Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.
- Puedes revisar el historial de commits para ver el conflicto y su resolución.



