

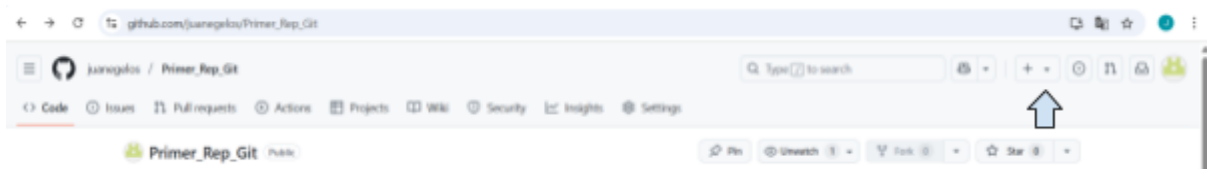
1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas) :

a) ¿Qué es GitHub?

Es una plataforma donde se pueden alojar y gestionar todos los archivos de un proyecto, utilizando el sistema de control de versiones Git. Estos archivos se organizan en repositorios que los contienen, junto a su historial de cambios, dando la posibilidad de que otras personas puedan participar del proyecto, fusionen sus cambios y revisen el código de otros.

b) ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en GitHub, primero debes abrir tu cuenta en la plataforma y luego desde el botón (+) crear un nuevo repositorio. Se le debe dar un nombre, poner una descripción (opcionalmente), establecer si tiene carácter “público o privado”, y si lo inicializas con un archivo Readme.



Create a new repository



A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *  juanegelos /

Great repository names are short and memorable. Need inspiration? How about [upgraded-octo-goggles](#) ?

Description (optional)

- ☒  **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.](#)

c) ¿Cómo crear una rama en Git?

Para crear una rama, sobre un repositorio local que ya tiene al menos un archivo que se ha incorporado, usamos la sentencia **git branch xxxxxx** para crear una nueva rama (xxxx simboliza el nombre que le asignamos a la rama que estamos creando)

d) ¿Cómo cambiar a una rama en Git?

Para cambiar a una rama en un repositorio, usamos la sentencia **git checkout xxxxx**, donde xxxxx es el nombre de la rama hacia la cual nos queremos desplazar.

e) ¿Cómo fusionar ramas en Git?

Para fusionar ramas se usa el comando **git merge xxxxxx**, donde xxxxx es el nombre de la rama que queremos fusionar sobre la que estamos trabajando. Ej: tenemos la rama “master” y la rama “nueva_rama”, si queremos fusionar esta última, primero me posiciono sobre la rama “master” y luego ejecuto el comando **git merge nueva_rama**.

f) ¿Cómo crear un commit en Git?

Para crear un commit debemos usar el comando **git commit -m “mensaje”**. Cuando hacemos alguna modificación sobre un archivo que ya está incorporado al seguimiento de git, primero preparamos el archivo para confirmar usando el comando **git add <nombre-archivo>** o simplemente **git add .**, luego de esto ejecutamos el comando **git commit -m “mensaje”**, y en el mensaje ponemos una breve descripción de lo que realizamos. Esto nos permitirá, de ser necesario, volver a un punto anterior del desarrollo del proyecto.

g) ¿Cómo enviar un commit a GitHub?

Para enviar un commit a GitHub, previamente debemos haberlo hecho en Git, y además debemos haber vinculado nuestro repositorio local con nuestro repositorio remoto

(GitHub). Para vincular el repositorio local con el remoto se ejecuta el comando **git remote add origin xxxxx**, donde xxxx es la ruta donde se encuentra el repositorio remoto. Ej: git remote add origin https://github.com/juanegelos/Primer_Rep_Git.git

Posteriormente debemos ejecutar el comando **git push -u origin master** para enviar el commit al repositorio remoto. Este comando se utiliza solo para el primer envío, posteriormente podemos usar solo **git push**.

h) ¿Qué es un repositorio remoto?

Es un repositorio alojado en una plataforma, donde estos repositorios pueden ser visualizados y compartirlo con otros usuarios.

i) ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git, debemos ejecutar el comando **git clone url**, donde url es la dirección específica donde se encuentra el repositorio remoto que deseamos agregar.

j) ¿Cómo empujar cambios a un repositorio remoto?

Para empujar cambios al repositorio remoto debemos ejecutar el comando **git push -u origin master**, si es la primera vez que subimos los cambios, o directamente **git push**.

k) ¿Cómo tirar de cambios de un repositorio remoto?

Hay dos formas para tirar o descargar e integrar cambios de un repositorio remoto a un repositorio local. La primera es utilizando el comando **git pull origin master** si la rama local no tiene configurada una rama remota de seguimiento, caso contrario, podemos utilizar simplemente **git pull**. Git se encargará de bajar e intentará fusionar los cambios con nuestra rama local actual.

La segunda forma es hacer la descarga utilizando el comando **git fetch origin master**, lo que creará una rama remota llamada origin/master en nuestro repositorio local, dejando nuestra rama master sin cambios, y posteriormente ejecutaremos el comando **git merge origin/master**, lo que intentará fusionar los cambios a nuestra rama master local. Si hay conflictos, deberemos resolverlos manualmente.

l) ¿Qué es un fork de repositorio?

Un fork es una copia del repositorio de otro usuario, en nuestra cuenta de GitHub. Es como si fuera una rama del proyecto principal, donde nosotros podremos efectuar todas las modificaciones que se nos ocurra, sin alterar el proyecto original. Para que nuestras modificaciones se incluyan en el proyecto principal, debemos hacer un **pull request** y el propietario del proyecto decidirá si incorpora o no esos cambios.

m) ¿Cómo crear un fork de un repositorio?

Primero debemos buscar el repositorio sobre el que queremos hacer un fork, por ejemplo <https://github.com/usuario-original/nombre-del-repositorio>. Luego vamos al botón fork y hacemos click. Github comenzará el proceso de creación y nos preguntará donde queremos crearlo. Ahí seleccionamos nuestra cuenta de GitHub y listo.

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

Entramos en la cuenta donde tenemos el repositorio que forkeamos y que ya hemos modificado. Luego presionamos el botón **“Contribute”**, se despliega una ventana y

presionamos el botón **Open pull request**. Se abre una nueva ventana, agregamos un título y una descripción de lo modificado y se presiona el botón **Create Pull request**

o) ¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción debes ser propietario del repositorio o tener permiso de escritura (mantenedores o colaboradores). Además, previamente se deberá revisar la pull request en cuanto a título, descripción, que los archivos modificados sean correctos, necesarios y no introduzcan problemas, si hay conflictos, etc. Una vez verificado todo, se puede optar por la forma en que se va a fusionar con el repositorio original. Se puede elegir **create merge commit** que crea un commit de fusión combinando el historial del colaborador con el original, conservando el historial completo de ambas ramas.

O **Squash and merge** esta opción combina todos los commits de la rama del colaborador, en un único commit en la rama original.

O **Rebase and merge** donde los commits del colaborador se aplican sobre la punta de la rama original.

Elegimos una de estas opciones y confirmamos.

p) ¿Qué es un etiqueta en Git?

Una etiqueta o tag es una referencia estática a un punto específico en la historia del repositorio. Se utiliza para marcar puntos importantes.

q) ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta usamos el comando **git tag xxxxx**, donde xxxx es el nombre de la etiqueta. Estas son etiquetas livianas porque no tienen ninguna otra información. También podemos crear etiquetas anotadas con el comando **git tag -a xxxxx -m "mensaje"**, donde xxxx es el nombre de la etiqueta y además podemos poner un mensaje.

r) ¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta a GitHub usamos el comando **git push origin xxxxx**, donde xxxx es el nombre de la etiqueta a enviar. Si queremos enviar todas las etiquetas al repositorio remoto, usamos el comando **git push -tags**.

s) ¿Qué es un historial de Git?

El historial de git es el registro cronológico de todos los cambios que se han producido en el repositorio.

t) ¿Cómo ver el historial de Git?

Para ver el historial existen diferentes formas:

Desde una terminal, podemos usar el comando **git log** (muestra todo el historial con todos sus datos, en varias líneas); **git log --oneline** (muestra el historial con todos los datos en una sola línea); **git log --all** (muestra el historial de todas las ramas); **git log --author="nombre"** (muestra los commits de un determinado autor); **git log --since="fecha" / git log --until="fecha"** (muestra los commits en un rango de fechas determinadas); **git log <archivo>** (muestra los commits de un determinado archivo); **git log -p** (muestra las diferencias intriducidas por cada commit).

Desde un editor de código como VSC (Visual Studio Code), agregando la extensión "source control" , ahí podemos ver todas las modificaciones que realizamos.

u) ¿Cómo buscar en el historial de Git?

Se busca de distintas formas y combinando distintas formas, por ejemplo:

- a) Por autor con el comando **git log --author="nombre del autor"**;
- b) Por fecha: **git log --since="fecha"**, **git log --until="fecha"**, **git log --after="fecha"**, **git log --before="fecha"**, o mostrando la fecha corta **git log --date=short**
- c) Por mensaje del commit: **git log --grep="palabra clave"**
- d) Por contenido del cambio: **git log -S"función específica"** (busca commits que introdujeron o eliminaron líneas que coinciden con la frase); **git log -G"Patron"** (busca commits que modificaron líneas que coinciden con la expresión regular).
- e) Por archivo: **git log <ruta y nombre del archivo>**
- f) Combinando: **git log --author="nombre del autor" --since="fecha" --grep="mensaje"**

v) ¿Cómo borrar el historial de Git?

Una forma es borrando el directorio .git que se creó dentro de la carpeta principal de nuestro proyecto. Esto elimina todo el historial completo.

Otra forma es borrando commits específicos del historial. para esto usamos el comando **git rebase -i <hash del commit anterior>**, esto eliminará los commits posteriores a este. Si queremos eliminar desde el inicio podemos hacer **git rebase -i --root**, se abre un editor de texto con el listado de los commits y los que queremos eliminar debemos cambiar la palabra pick por drop. Luego guardamos y cerramos el editor.

w) ¿Qué es un repositorio privado en GitHub?

Es un repositorio al cual puede acceder solo la persona que lo ha creado o las personas que él ha explícitamente concedido acceso.

x) ¿Cómo crear un repositorio privado en GitHub?

Al momento de crear un repositorio, optamos por la forma privado.

y) ¿Cómo invitar a alguien a un repositorio privado en GitHub?

Estando dentro del repositorio privado, buscamos la opción **"Add collaborators to this repository"** y presionamos el botón **"invite collaborators"**. Una vez dentro de **"Collaborators and teams"** presionamos el botón **"Add people"** y buscamos la persona que queremos invitar.

z) ¿Qué es un repositorio público en GitHub?

Es un repositorio que está visible y accesible para cualquier persona en internet.

aa) ¿Cómo crear un repositorio público en GitHub?

Al momento de crear un repositorio, optamos por la forma público.

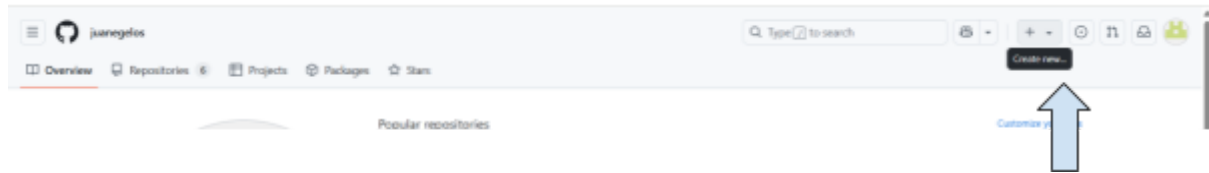
bb) ¿Cómo compartir un repositorio público en GitHub?

Compartiendo la URL del repositorio.

2) Realizar la siguiente actividad:

- Crear un repositorio.

Voy a mi cuenta de GitHub y presiono el boton + New repository



Le coloco de nombre Repo-tarea2, le hago un pequeño comentario y le doy carácter público y lo inicializo con un archivo Redme. Para finalizar presiono el botón “Create repository”

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner *

juanegelos ▾

Repository name *

Repo-tarea2

✓ Repo-tarea2 is available.

Great repository names are short and memorable. Need inspiration? How about [animated-broccoli](#)?

Description (optional)

Trabajo de la tecnicatura en programación - Unidad 02 Trabajo Colaborativo



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:



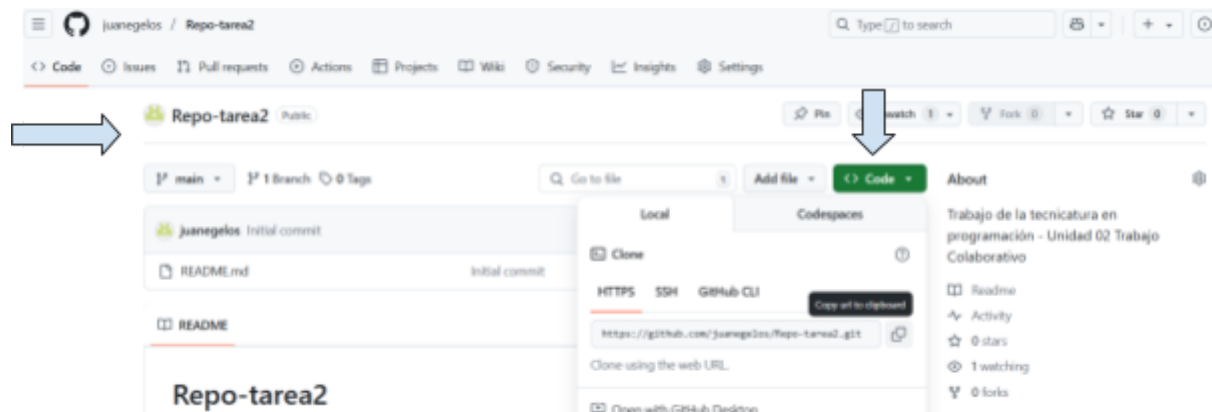
Add a README file

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

Add .gitignore

.gitignore template: None ▾

Una vez creado, copio la URL para poder clonarlo en mi PC. Esto se hace estando dentro del repo que creamos recién, presionamos el botón verde “<> Code” que despliega una mini ventana donde aparecerá la dirección https donde se aloja nuestro repositorio.



Luego abro una terminal en mi computadora, busco el directorio donde quiero copiar el repositorio y ejecuto el comando `git clone url`, donde url es la dirección que copiamos en el paso anterior. En mi caso, <https://github.com/juanegelos/Repo-tarea2.git>

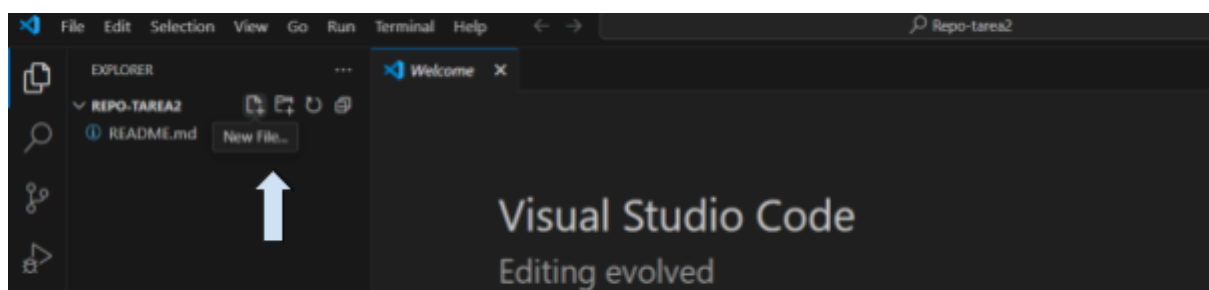
```

Windows PowerShell
PS C:\Users\juana\downloads> cd tecnica
PS C:\Users\juana\downloads\tecnica> cd "A1 - PROGRAMACION I"
PS C:\Users\juana\downloads\tecnica\A1 - PROGRAMACION I> git clone https://github.com/juanegelos/Repo-tarea2.git
Cloning into 'Repo-tarea2'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
PS C:\Users\juana\downloads\tecnica\A1 - PROGRAMACION I>
  
```

• Agregando un Archivo

o Crea un archivo simple, por ejemplo, "mi-archivo.txt".

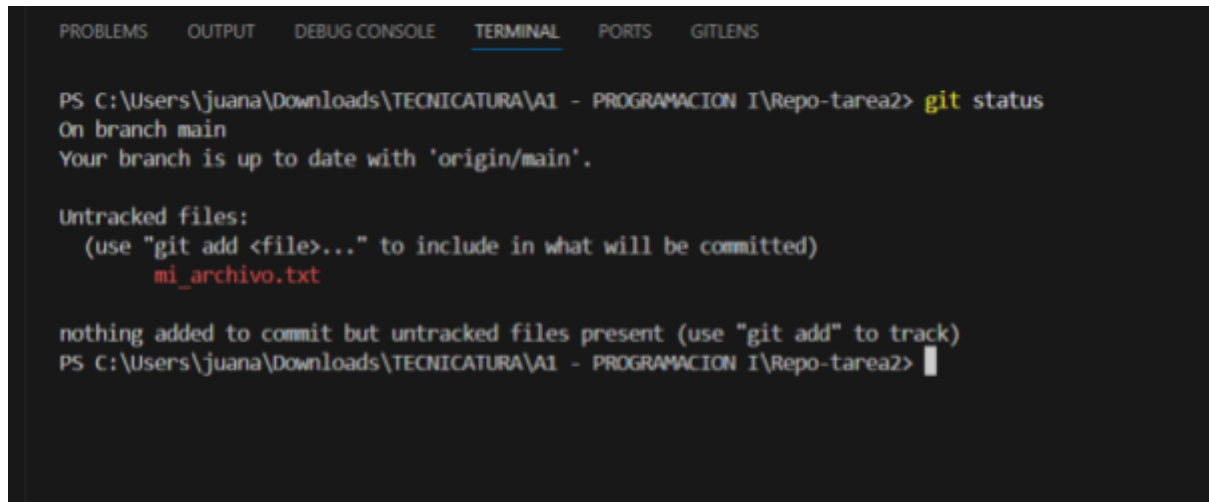
Voy a VSC abro la carpeta /Repo-tarea2 (Open Folder) y creo un archivo de texto "mi-archivo.txt" (New file), lo modifico y guardo los cambios ("ctrl + s").



o Realiza los comandos `git add .` y `git commit -m "Agregando mi-archivo.txt"` en la línea de comandos.

Desde la terminal de VSC o cualquier otra terminal ejecuto los comandos `git add .` para preparar el archivo modificado y que sea rastreable, y posteriormente el comando `git commit -m "Creacion archivo txt de prueba"`. Voy a ir ejecutando previo a cada paso el comando `git status` para ir viendo los distintos estados del archivo.

Ejecuto **git status** y obtengo esto:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

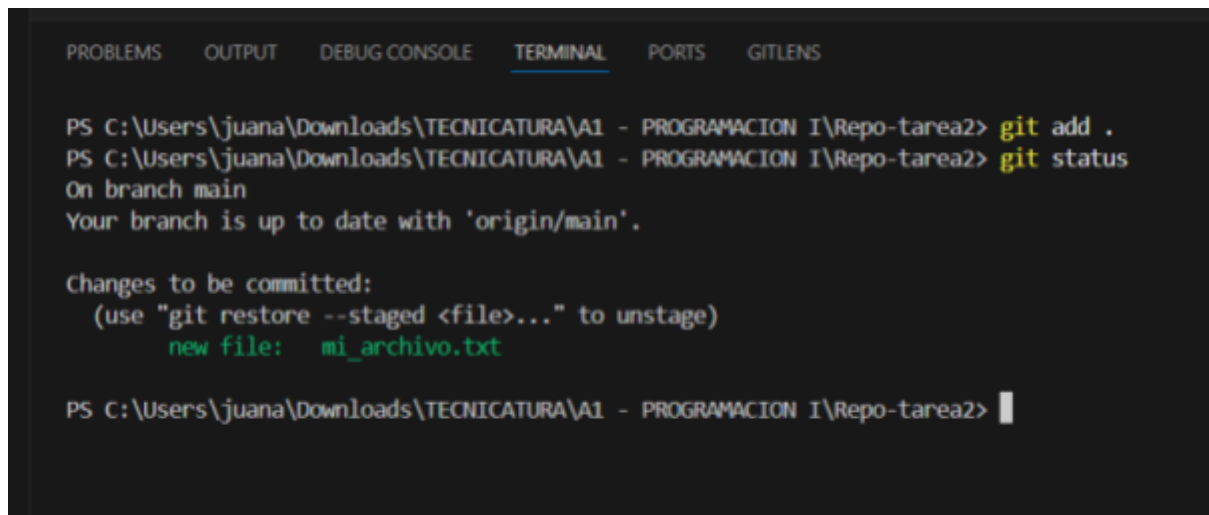
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> git status
On branch main
Your branch is up to date with 'origin/main'.

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

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> █
```

Previo a ejecutar el comando **git add .**, me dice que tengo archivos que no están siendo trackeados o seguidos por git (Untracked files:)

Ahora ejecuto **git add .** y posteriormente **git status**, y obtengo esto:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> git add .
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> 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:   mi_archivo.txt

PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> █
```

Ahora me dice que tengo cambios para ser confirmados (Changes to be committed:) y me muestra los cambios: new file: mi_archivo.txt

Ahora ejecuto el comando **git commit -m "Creacion archivo txt de prueba"** y posteriormente **git status** de nuevo.


```
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> git commit -m "Creacion archivo txt de prueba"
[main 5405a07] Creacion archivo txt de prueba
1 file changed, 1 insertion(+)
create mode 100644 mi_archivo.txt
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> █
```

En la primera acción me indica que 1 archivo cambió con 1 inserción (**1 file changed, 1 insertion(+)**) y en la segunda acción me dice que mi rama está 1 confirmación por delante de origin/main que es el repositorio remoto (**Your branch is ahead of 'origin/main' by 1 commit.**), y me sugiere que empuje los cambios hacia el remoto (**use "git push" to publish your local commits**).

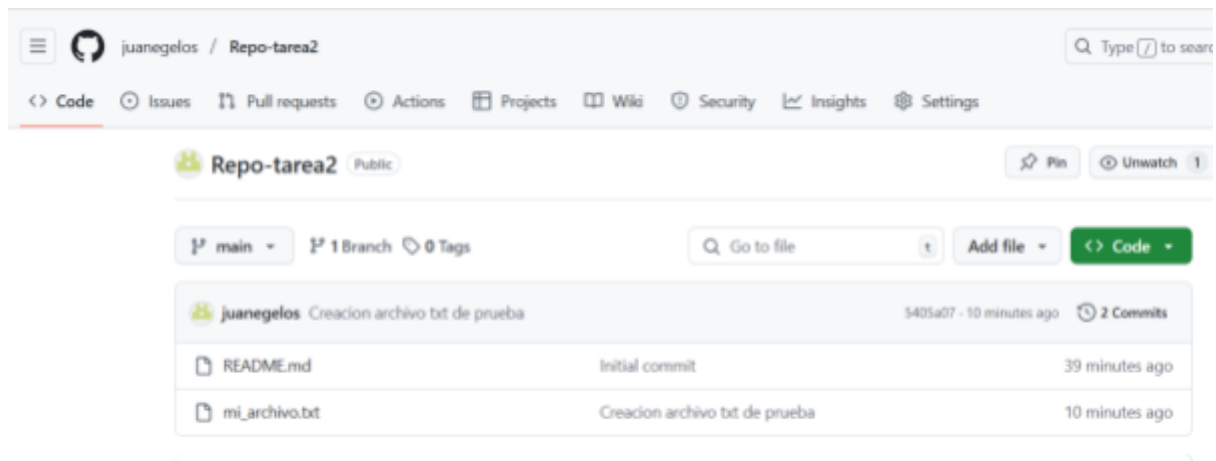
o Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).

Ejecuto el comando **git push** para subir los cambios realizados en el repositorio local, hacia el repositorio remoto.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 376 bytes | 376.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/juanegelos/Repo-tarea2.git
a389586..5405a07  main -> main
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> █
```

Ahora verifico que en mi repositorio remoto estén los cambios.



Como vemos, el archivo “mi_archivo.txt” ya está subido al repositorio remoto.

• Creando Branchs

o Crear una Branch

Volvemos a VSC y ejecutamos el comando **git branch prueba1**, para crear la rama que denominaremos “prueba1” y cambiamos a la nueva rama con el comando **git checkout prueba1**

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> git branch prueba1
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> git status
On branch main
Your branch is up to date with 'origin/main'.

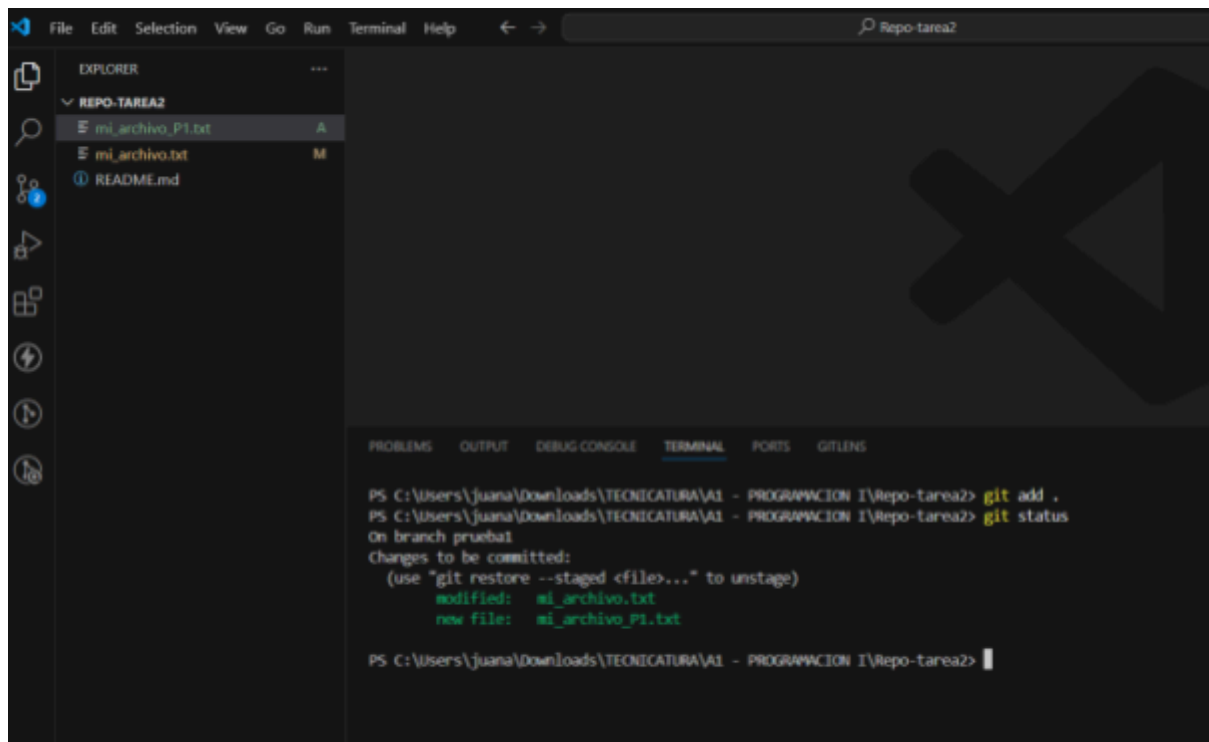
nothing to commit, working tree clean
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2> git checkout prueba1
Switched to branch 'prueba1'
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION I\Repo-tarea2>

```

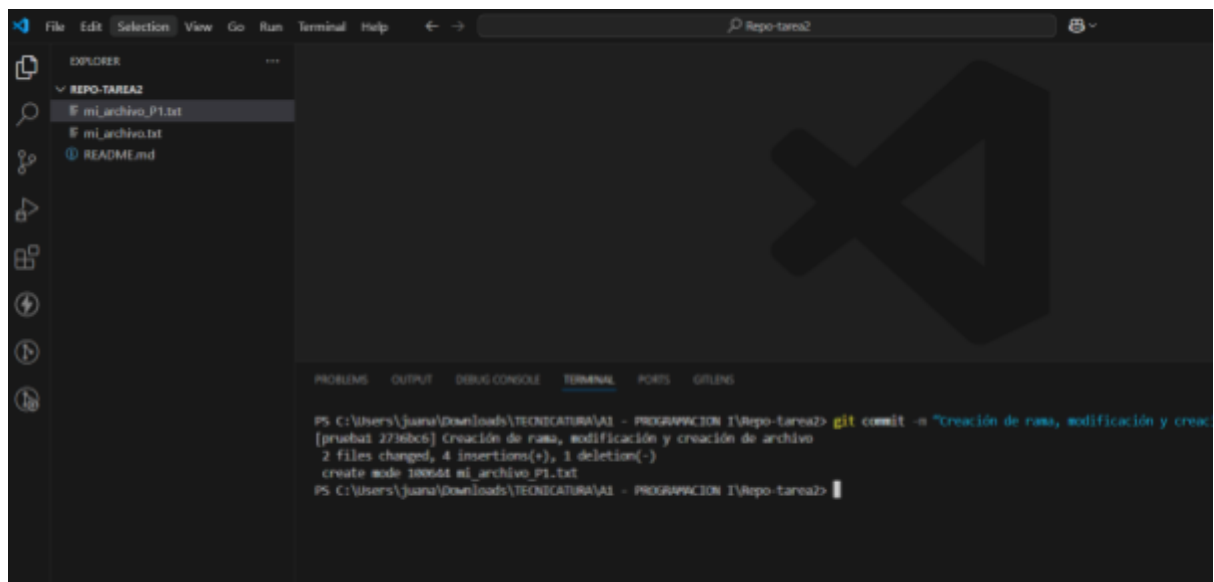
o Realizar cambios o agregar un archivo

Voy a realizar cambios en el archivo “mi_archivo.txt” y voy a crear otro archivo txt, en ambos casos usaré VSC. Posteriormente agregaré el archivo nuevo y la modificación al registro de seguimiento (**git add .**) y confirmaré los cambios (**git commit -m “Creación de rama, modificación y creación de archivo”**)

Antes de ejecutar los comandos git add y git commit, vemos en el VSC que los archivos aparecen con unas letras al costado derecho, una “M” para “mi_archivo.txt” y una “U” para “mi_archivo_P1.txt”. Esto es porque el primero es un archivo que ya estaba siendo trakeado (seguido) y se ha modificado pero todavía no se ha puesto en estado de confirmar los cambios. El segundo, es un archivo nuevo que todavía no está siendo trakeado (untracked). Ejecutamos git add y vemos que pasa.



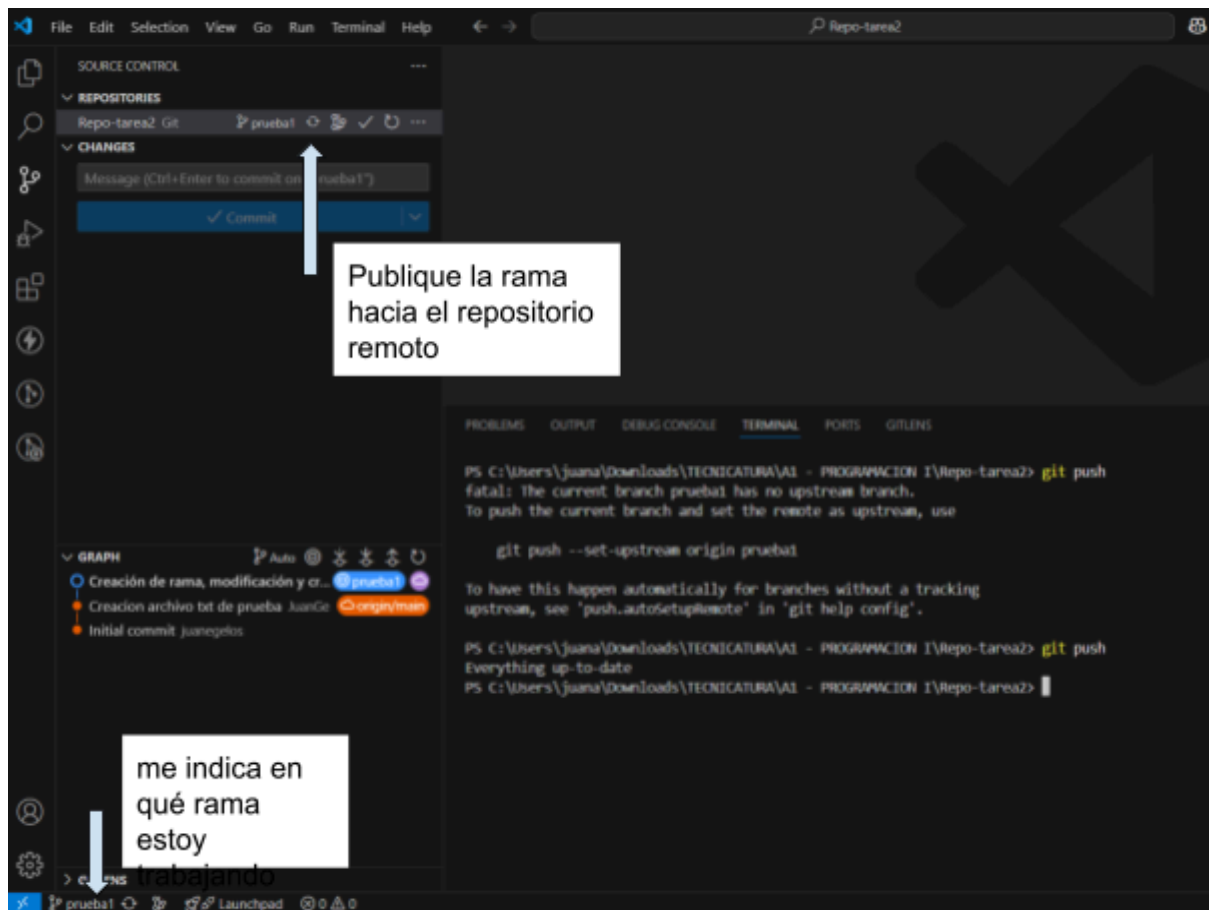
Ahora cambiaron las letras a “M” y “A” y me dice que tengo cambios a confirmar. Ejecuto git commit y vemos que pasa:



Vemos que nos dice que 2 archivos cambiaron, 4 inserciones y 1 eliminación. Como se ve, han desaparecido las letras a la derecha de los nombres de los archivos.

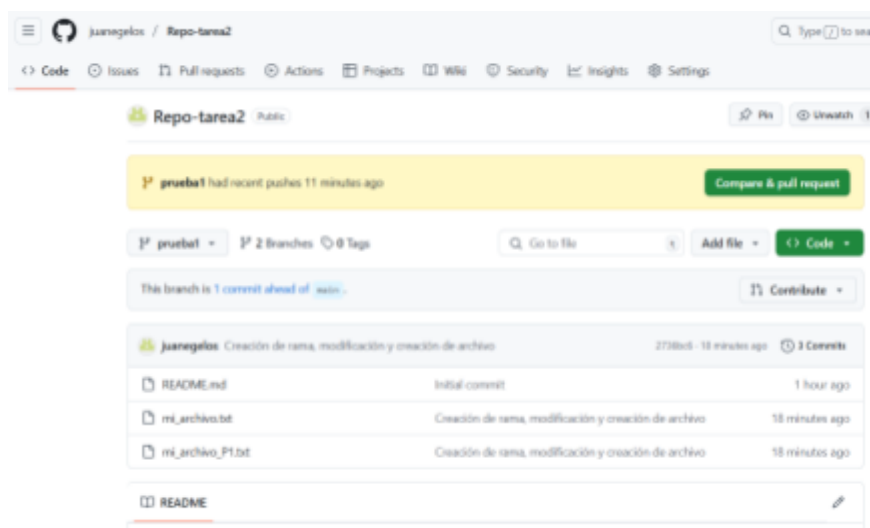
o Subir la Branch

Posicionado sobre la rama “prueba1”, ejecuto el comando **git push** para subir la rama y las modificaciones al repositorio remoto



Al hacerlo me dio un error porque no había publicado la rama en el repositorio remoto. Me propone ejecutar previamente el comando **git push --set-upstream origin prueba1**. Yo lo hice desde “Source Control” en VSC, publique la rama hacia el repositor, haciendo que ambos se sincronizaran. Posteriormente ejecute el comando git push y subí todos los cambios.

Verifico que mi repositorio remoto tiene todos los cambios, las dos ramas y todos los archivos.



3) Realizar la siguiente actividad:

Paso 1: Crear un repositorio en GitHub

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere [Import a repository](#).


Required fields are marked with an asterisk (*).

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner *

 juanegelos ▾

Repository name *

conflict-exercise

✔ conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about [shiny-potato](#) ?

Description (optional)

Repositorio de Programación 1 para la Tecnicatura Universitaria a distancia de la UTN.

☒  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

Paso 2: Clonar el repositorio a tu máquina local

```
Windows PowerShell
PS C:\Users\juana\downloads\tecnicatura\A1 - PROGRAMACION I> git clone https://github.com/juanegelos/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
PS C:\Users\juana\downloads\tecnicatura\A1 - PROGRAMACION I> |
```

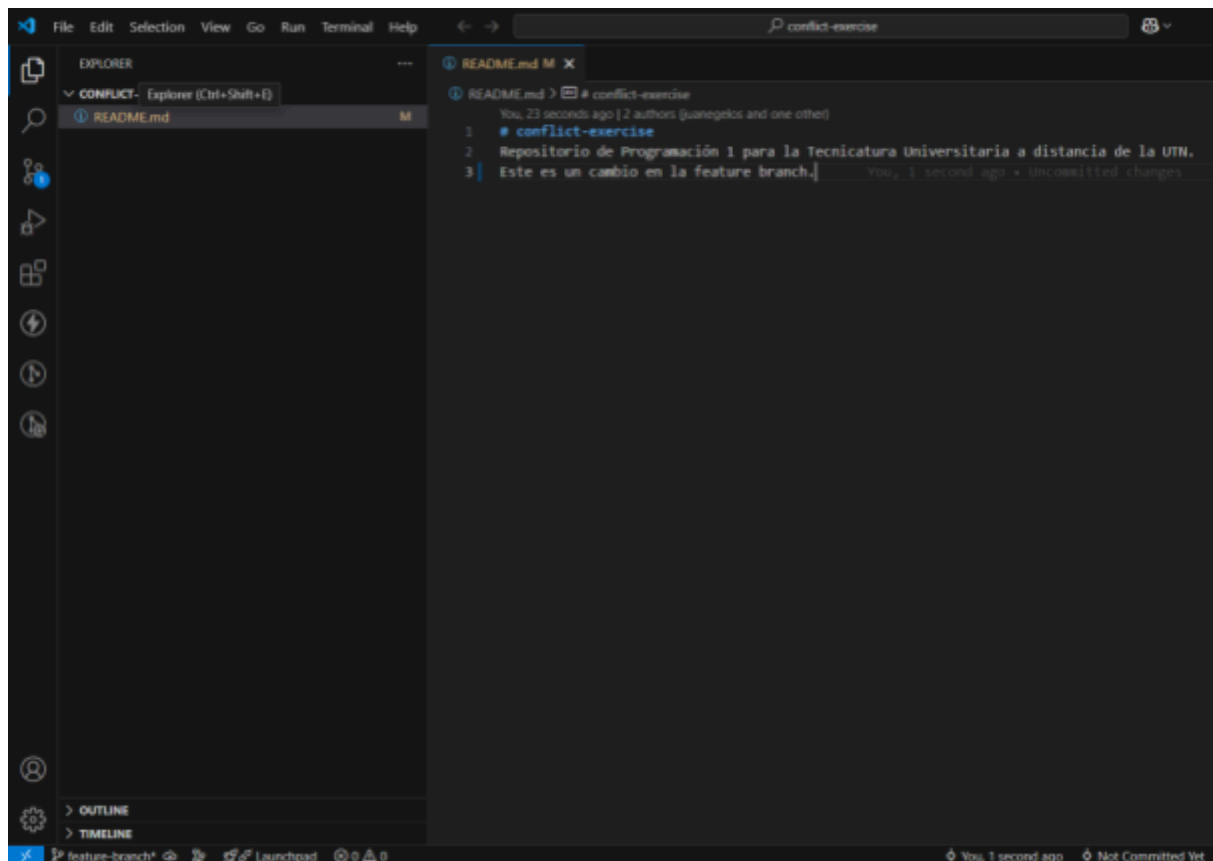
Paso 3: Crear una nueva rama y editar un archivo

Creo la rama feature-branch

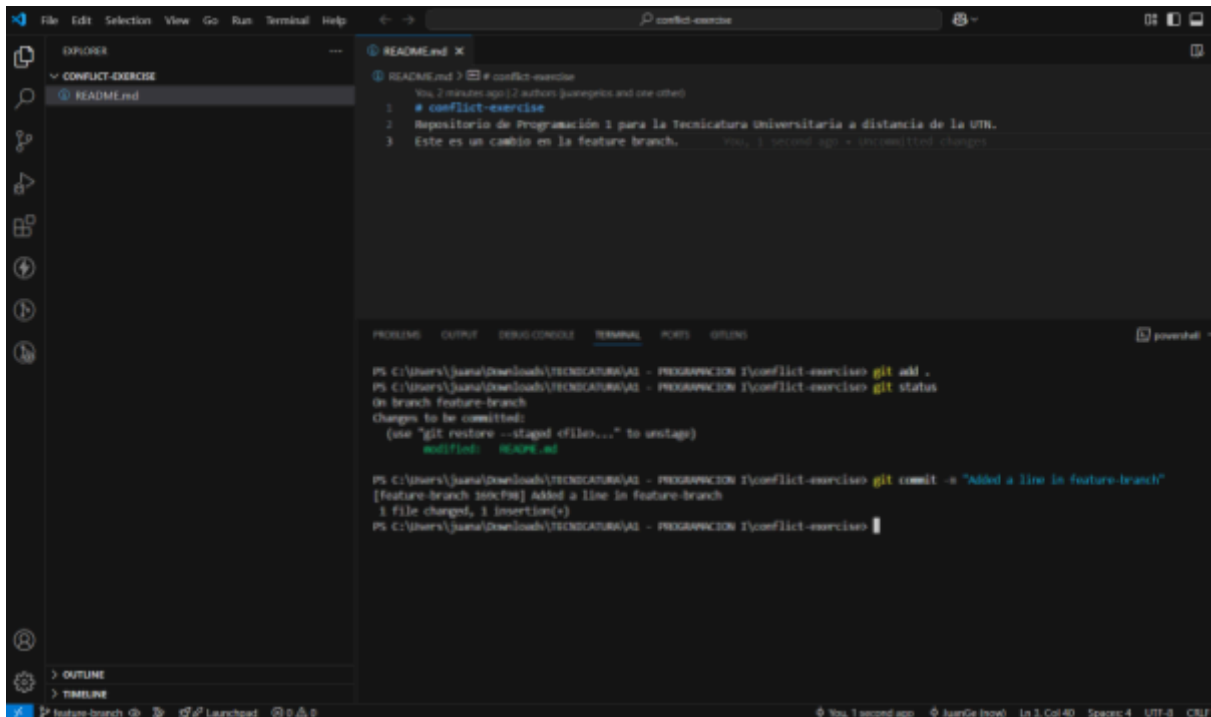
```
Windows PowerShell
PS C:\Users\juana\downloads\tecnicatura\A1 - PROGRAMACION I> git clone https://github.com/juanegelos/conflict-exercise.git
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
PS C:\Users\juana\downloads\tecnicatura\A1 - PROGRAMACION I> cd conflict-exercise
PS C:\Users\juana\downloads\tecnicatura\A1 - PROGRAMACION I\conflict-exercise> git branch feature-branch
PS C:\Users\juana\downloads\tecnicatura\A1 - PROGRAMACION I\conflict-exercise> git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
PS C:\Users\juana\downloads\tecnicatura\A1 - PROGRAMACION I\conflict-exercise> |
```

Modifico el archivo readme.md y guardo los cambios



Hago commit



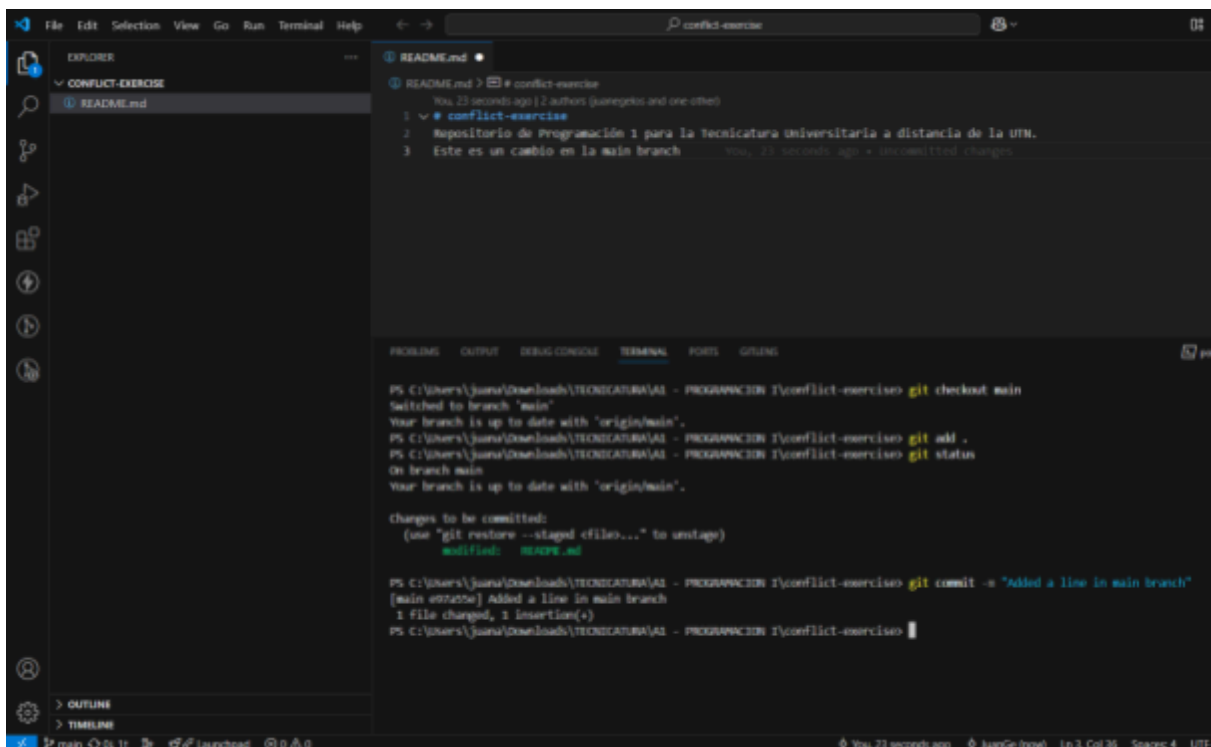
The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project named 'CONFLICT-EXERCISE' with a file named 'README.md'. The main editor area displays the content of 'README.md', which includes a title '# conflict-exercise' and two bullet points: 'Repositorio de Programación 1 para la Tecnicatura Universitaria a distancia de la UTN.' and 'Este es un cambio en la feature branch.' The bottom panel shows the TERMINAL window with the following commands and output:

```
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION 1\conflict-exercicio> git add .
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION 1\conflict-exercicio> git status
On branch feature-branch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md

PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION 1\conflict-exercicio> git commit -m "Added a line in feature-branch"
[feature-branch 90cf56] Added a line in feature-branch
 1 file changed, 1 insertion(+)
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION 1\conflict-exercicio>
```

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

Cambio a rama main, modifico readme.md, ejecuto git add . y luego git commit -m "Added a line in main branch"



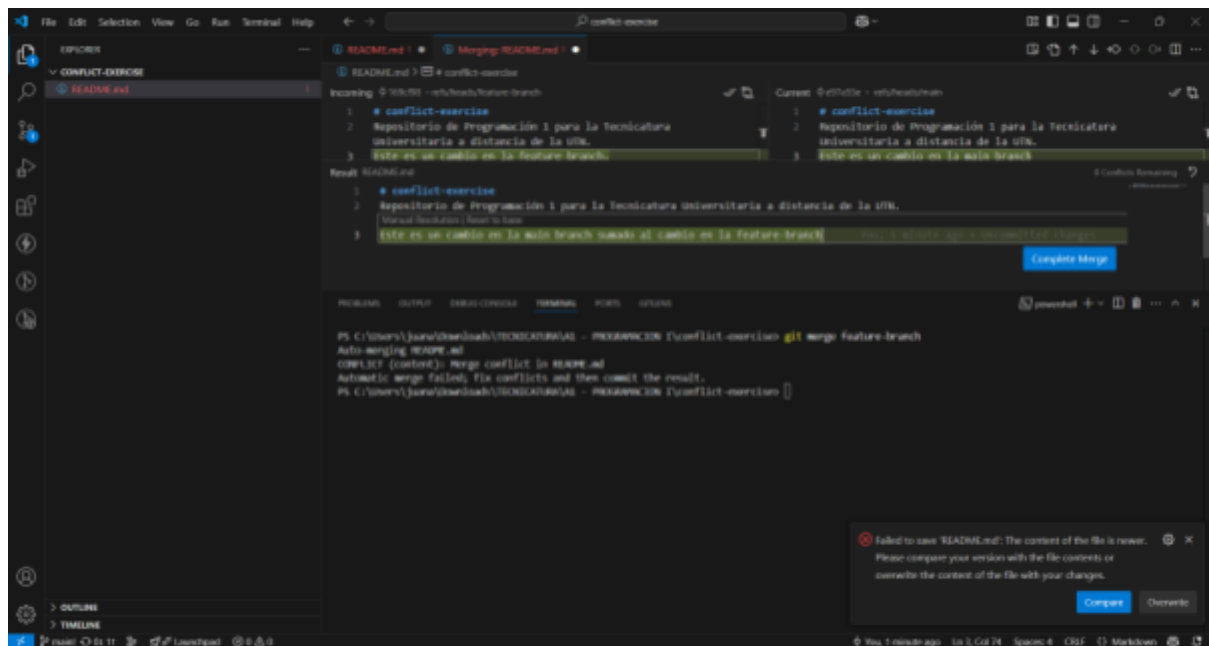
The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project named 'CONFLICT-EXERCISE' with a file named 'README.md'. The main editor area displays the content of 'README.md', which includes a title '# conflict-exercise' and two bullet points: 'Repositorio de Programación 1 para la Tecnicatura Universitaria a distancia de la UTN.' and 'Este es un cambio en la main branch'. The bottom panel shows the TERMINAL window with the following commands and output:

```
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION 1\conflict-exercicio> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION 1\conflict-exercicio> git add .
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION 1\conflict-exercicio> 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)
        modified:   README.md

PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION 1\conflict-exercicio> git commit -m "Added a line in main branch"
[main 90cf56] Added a line in main branch
 1 file changed, 1 insertion(+)
PS C:\Users\juana\Downloads\TECNICATURA\A1 - PROGRAMACION 1\conflict-exercicio>
```

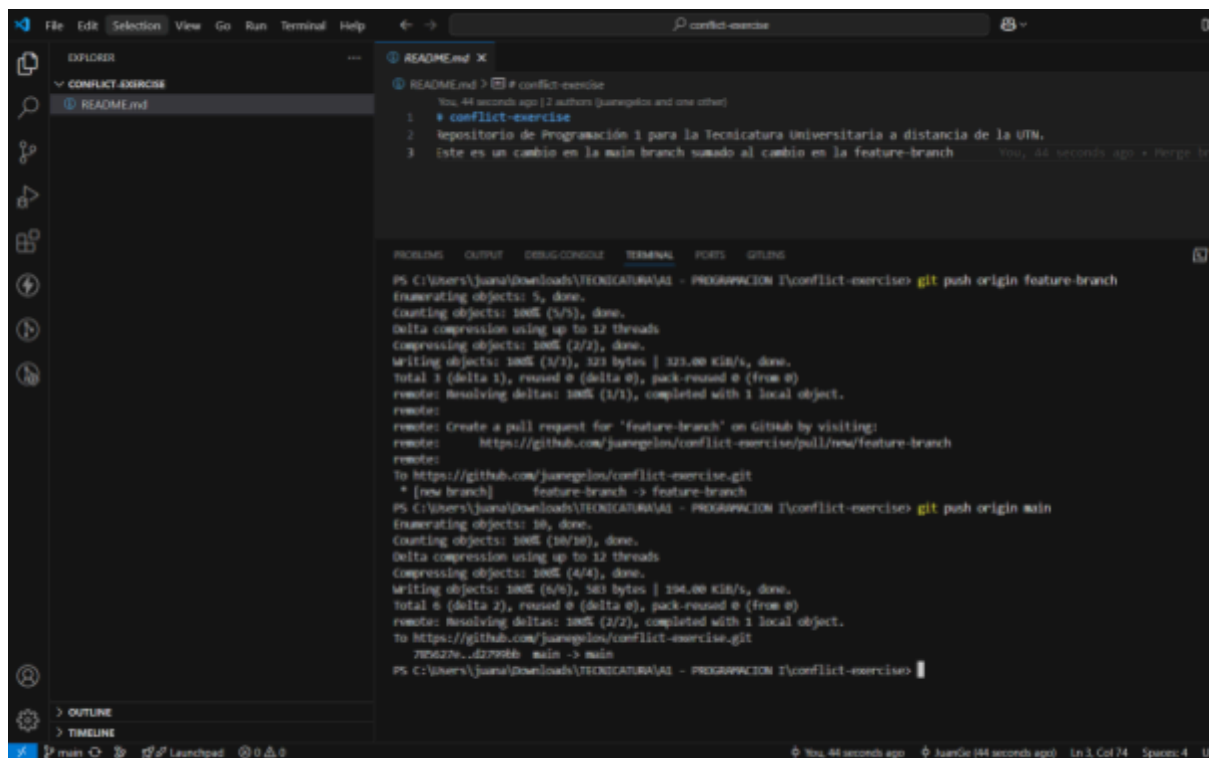
Paso 5: Hacer un merge y generar un conflicto



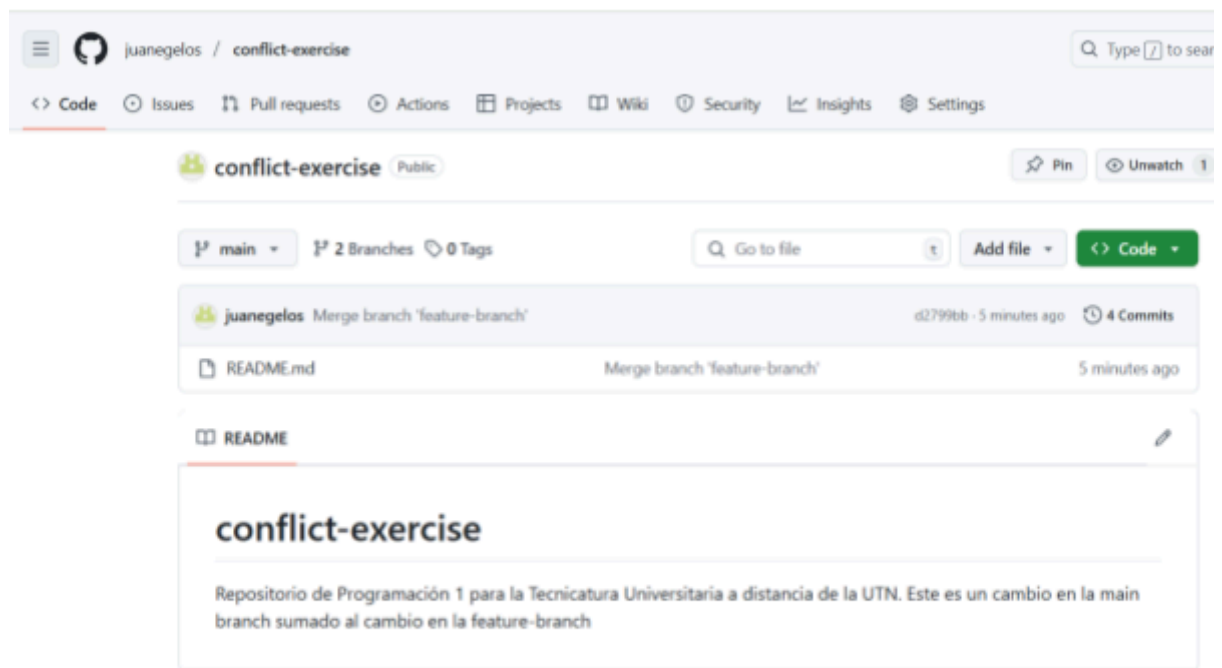
Paso 6: Resolver el conflicto

Soluciono conflicto y guardo los cambios usando `git add .` y `git commit -m "Resolved merge conflict"`

Paso 7: Subir los cambios a GitHub



Paso 8: Verificar en GitHub



Vemos que se ha creado la rama feature-branch y que el archivo readme.md ha quedado con las modificaciones que hicimos cuando resolvimos el conflicto que se generó al hacer git merge.