

Nombre y Apellido	Email
Bruno Pighin	<a href="mailto:brunopighin@hotmail.com">brunopighin@hotmail.com</a>
Jazmín Herrera	<a href="mailto:herrerajazmin934@gmail.com">herrerajazmin934@gmail.com</a>
Alan Jofre	<a href="mailto:alan91_10@hotmail.com">alan91_10@hotmail.com</a>
Aldo Manfredi	<a href="mailto:manfredialdo.1979@gmail.com">manfredialdo.1979@gmail.com</a>
Florencia Mauna	<a href="mailto:florrrmauna@gmail.com">florrrmauna@gmail.com</a>

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

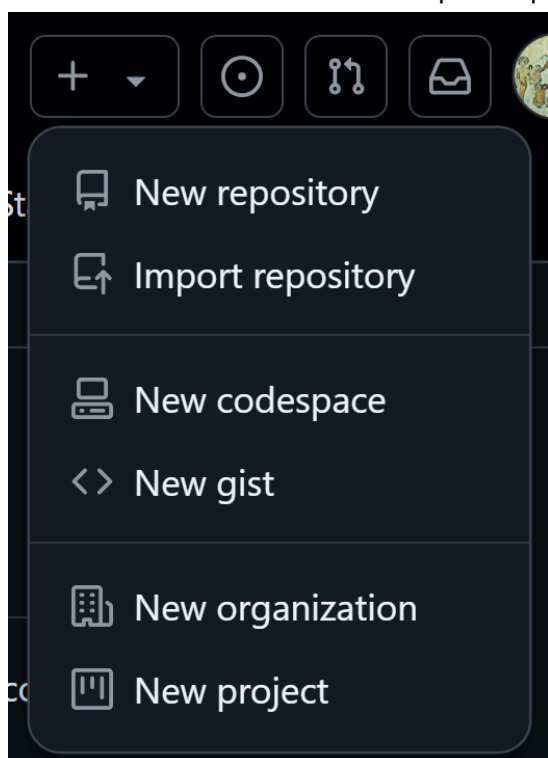
**¿Qué es GitHub?**

GitHub es una plataforma donde se puede almacenar, compartir y trabajar junto con otros usuarios para escribir código... entre otras cosas permite el compartir trabajo, o realizar trabajo colaborativo, seguir y administrar los cambios en el código, etc.

**¿Cómo crear un repositorio en GitHub?**

Crear el repositorio en GitHub se hace a través de su plataforma web <https://github.com/>.

Dentro de nuestro usuario.. en la esquina superior derecha nos ubicamos en el simbolo de suma y




## 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 \*      Repository name \*

 manfredialdo / grupoBravotest001


 grupoBravotest001 is available.


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

Description (optional)

utn tup grupo bravo

---

☒  **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.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

---

 You are creating a public repository in your personal account.

---

[Create repository](#)

Es bueno aclarar que, `git init <nombre_test>` crea un repositorio **local** llamado `<nombre_test>` en tu computadora.

¿Cómo crear una rama en Git? `git branch <nombre_rama>`

¿Cómo cambiar a una rama en Git? `git checkout <otra_rama>`

¿Cómo fusionar ramas en Git? Con otra rama: `git merge <otra_rama>`

¿Cómo crear un commit en Git? `git commit -m "mensaje descriptivo del commit"`

¿Cómo enviar un commit a GitHub? `git push origin <nombre_de_tu_rama>`

# para el primer push; `git push -u origin <nombre_de_tu_rama>`

**¿Qué es un repositorio remoto?**

¿Cómo agregar un repositorio remoto a Git? `git remote add origin <URL_del_repositorio_remoto>`

git remote add origin <URL\_del\_repositorio\_remoto> vincula tu repositorio local de Git con un repositorio remoto en un servidor (como GitHub, GitLab o Bitbucket), creando un atajo llamado "origin" que representa esa dirección web. Esto te permite enviar (push) tus cambios locales al repositorio remoto y recibir (pull) actualizaciones desde él.

**¿Cómo empujar cambios a un repositorio remoto?** git push <nombre\_del\_remoto> <nombre\_de\_tu\_rama>

**¿Cómo tirar de cambios de un repositorio remoto?** (git pull origin < nombre\_de\_la\_rama >).

Para lograr esto debemos colocar, Git pull para poder descargar cambios y Git merge para poder fusionarlo con tu rama local (git pull origin < nombre\_de\_la\_rama >).

En el caso de trabajar con una rama específica, como main, deberá colocarse de la siguiente forma: (git pull origin main).

**¿Qué es un fork de repositorio?**

Cuando nos encontramos en Github muchas veces nos puede parecer interesante algún repositorio, pero no es de nuestro usuario, para esto Github tiene un botón llamado Fork el cual se encuentra en la parte superior derecha. Al hacer clic allí podremos generar una copia exacta en nuestro usuario de aquel repositorio que nos interesó, también podremos modificarlo a nuestro gusto sin problema, ya que no afectará al repositorio original, sino que irá cualquier modificación a nuestra copia local. Finalmente, vamos a copiar ese repositorio que se encuentra en nuestro canal para poder moverlo a una carpeta que elijamos y continuar trabajando desde nuestro repositorio local.

**¿Cómo crear un fork de un repositorio?**

Para crear un fork de un repositorio, debes ingresar a Github y encontrar aquel usuario que te interese su repositorio . Una vez allí, puedes hacer un Fork del repositorio (haciendo clic en el extremo derecho).Luego lo clonaremos en una carpeta y, si hay un archivo que creemos que puede ser mejorado, lo modificamos. Luego lo agregamos al stage y hacemos un commit y finalmente enviamos los cambios al fork que creamos utilizando git push origin master.

Finalmente si deseamos que el autor del repositorio original vea la modificación que hicimos. Debemos avisarle, ya que los cambios que realizamos no se reflejarán en el repositorio original, se dirigen a nuestra copia local.

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

Para realizar la solicitud de Pull request, nos iremos a la pestaña de Solicitudes de pull request donde haremos clic en new pull request. Aparecerá una ventana que mostrará un resumen de las modificaciones que hemos realizado en comparación con el código original. Haremos clic en Crear solicitud de extracción, donde veremos el título, allí agregaremos un mensaje general y un poco más abajo hay suficiente espacio para explicar las razones por las cuales consideramos que el cambio que hemos hecho sería beneficioso para agregar al repositorio original.

**¿Cómo aceptar una solicitud de extracción?**

El creador del repositorio podrá visualizar en sus solicitudes de extracción el aviso que le hemos enviado, lo que le permitirá revisarlo y, si lo considera adecuado, efectuar la modificación correspondiente, además de tener la opción de responder al usuario que sugirió dicho cambio. La ventaja de este proceso es que si el usuario original determina que esta alteración es beneficiosa y

no ocasiona conflictos con la rama principal de su repositorio local remoto, podrá hacer clic en Merge pull request, integrando así en su repositorio las modificaciones realizadas por un usuario.

### **¿Qué es una etiqueta en Git?**

**¿Cómo crear una etiqueta en Git?** git tag <nombre\_de\_la\_etiqueta>

**¿Cómo enviar una etiqueta a GitHub?** git push origin <nombre\_etiqueta>

### **¿Qué es un historial de Git?**

**¿Cómo ver el historial de Git?** git log o git log --oneline

**¿Cómo buscar en el historial de Git?** git log --grep='<patrón\_de\_búsqueda>'

**¿Cómo borrar el historial de Git?** git reset --hard <commit\_anterior\_al\_que\_deseo\_eliminar>

mejor... git revert <commit\_a\_revertir>

### **¿Qué es un repositorio privado en GitHub?**

Un repositorio comprende una colección de archivos (o carpetas) de un proyecto en conjunto con su historial de revisión. En su carácter de “privado”, este repositorio solo puede ser visto por su propietario y/o aquellas personas a las que se les haya concedido acceso previo. Se suele usar para proyectos de desarrollo interno como por ejemplo los de una universidad o una empresa.

### **¿Cómo crear un repositorio privado en GitHub?**

Iniciar sesión

En la esquina superior derecha, hacer click en el icono de perfil y seleccionar “your repositories”.

Hacer click en el botón verde que dice “New” para crear un repositorio nuevo.

Ingresar un nombre y una descripción del repositorio que queremos crear.

Seleccionamos la opción de “private” para que el repositorio sea privado.

Podemos elegir entre un README o un .gitignore para empezar.

Hacer click en el botón verde que dice “Create repository”.

### **¿Cómo invitar a alguien a un repositorio privado en GitHub?**

Vamos a nuestro repositorio en GitHub.

En la parte superior derecha, nos dirigimos a Settings.

En el menú seleccionamos Collaborators and teams.

En la parte de Collaborators seleccionamos Add people.

Escribimos el nombre de usuario o el correo electrónico de la persona a la que queremos invitar.

Presionamos Add.

### **¿Qué es un repositorio público en GitHub?**

Un repositorio comprende una colección de archivos (o carpetas) de un proyecto en conjunto con su historial de revisión. Su carácter de “público” lo convierte en contenido accesible para cualquier usuario de GitHub (e incluso para personas que no están registradas en la plataforma).

Esta versión es útil para proyectos de código abierto o cuando se quiere compartir el código con la comunidad.

### **¿Cómo crear un repositorio público en GitHub?**

Iniciar sesión

En la esquina superior derecha, hacer click en el icono de perfil y seleccionar “your repositories”.

Hacer click en el botón verde que dice “New” para crear un repositorio nuevo.

Ingresar un nombre y una descripción del repositorio que queremos crear.

Seleccionamos la opción de “public” para que el repositorio sea privado.

Podemos elegir entre un README o un .gitignore para empezar.

Podemos elegir una licencia.

Hacer click en el botón verde que dice “Create repository”.

### **¿Cómo compartir un repositorio público en GitHub?**

Para compartir un repositorio público en GitHub lo podemos hacer de varias maneras:

Compartimos en enlace del repositorio: copiando la URL desde la barra de navegación o enviándola por correo, red social, etc)

Haciendo un fork: otros usuarios pueden hacer un fork del repositorio, es decir, pueden copiarlo y modificarlo en su propia cuenta.

Usando Git para clonar: compartimos el comando para clonar el repositorio.

Activando GitHub Pages: si es un repositorio de una página web, se puede activar desde Settings > Pages y compartir la URL de la web generada.

## **2. Realizar la siguiente actividad:**

Crear un repositorio.

**a - Dale un nombre al repositorio.** grupoBravotest001


**b - Elije el repositorio sea público.**

**c - Inicializa el repositorio con un archivo.**

Con la sesión iniciada en github... se crea este repo con las siguientes características


**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 \*  / Repository name \*   
grupoBravotest001 is available.

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

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

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

Choose a license

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

This will set `main` as the default branch. Change the default name in your [settings](#).

☐ You are creating a public repository in your personal account.

[Create repository](#)

git clone https://github.com/manfredialdo/grupoBravotest001.git tp2

cd tp2

# Agregando un Archivo

touch mi-archivo.txt

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

# Agregar todos los archivos al área de staging:

git add .

# Realizar el primer commit:

git commit -m "Agregando mi-archivo.txt"

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

git push -u origin main

y para las siguientes git push origin main

**F- # Agregar el repositorio remoto (origin)**

git remote add origin https://github.com/manfredialdo/grupoBravotest001.git

**# Subir los cambios al repositorio remoto (intentando con 'main' primero):**

git push -u origin main

**# Subir los cambios al repositorio remoto (intentando con 'main' primero)**

git push origin main

**# Verificar el estado del repositorio (nuevamente)**

git status

## **G - # Creando Branchs**

**# Crear una nueva branch (ejemplo: 'mi-nueva-rama')**

git checkout -b mi-nueva-rama

## **H- Realizar cambios o agregar un archivo**

**# Realizar cambios (editar archivos, crear nuevos archivos, etc.) con el visual studio**

**# Agregar los cambios al área de staging**

git add .

**# Realizar un commit en la nueva Branch**

git commit -m "Segundo commit tp2actividad2"

## **I - Subir la Branch**

**# Subir la nueva branch al repositorio remoto**

git push origin mi-nueva-rama

**# Volver a la rama principal** (opcional, si quieres seguir trabajando en main/master)

git checkout main # o git checkout master

**# Subir los cambios de la rama principal al repositorio remoto**

git push origin main # o git push origin master

## **3. Realizar la siguiente actividad:**

**Paso 1: Crear un repositorio en GitHub** <https://github.com/manfredialdo/conflict-exercise.git>

Ve a GitHub e inicia sesión en tu cuenta.

Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.

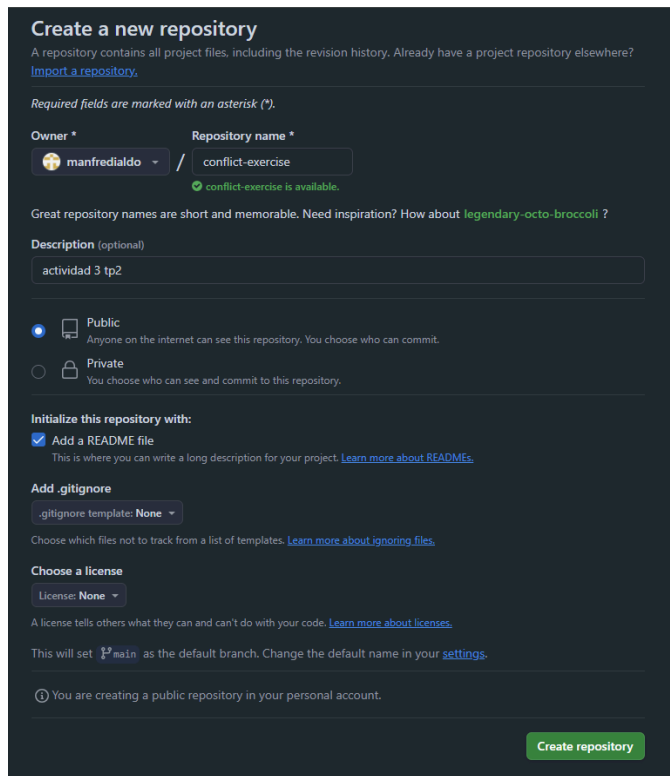
Asigna un nombre al repositorio, por ejemplo, conflict-exercise.

Opcionalmente, añade una descripción.

Marca la opción "Initialize this repository with a README".

Haz clic en "Create repository".

<https://github.com/manfredialdo/conflict-exercise.git>



The screenshot shows the GitHub 'Create a new repository' page. At the top, it says 'Create a new repository' and 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, it states 'Required fields are marked with an asterisk (\*)'. The 'Owner' field is set to 'manfredialdo' and the 'Repository name' field is 'conflict-exercise', with a green checkmark indicating 'conflict-exercise is available.'. A suggestion for repository names is provided: 'Great repository names are short and memorable. Need inspiration? How about [legendary-octo-broccoli](#) ?'. The 'Description' field is optional and contains 'actividad 3 tp2'. Under 'Visibility', 'Public' is selected, with the note 'Anyone on the internet can see this repository. You choose who can commit.'. Under 'Initialize this repository with:', 'Add a README file' is checked, with a note 'This is where you can write a long description for your project. [Learn more about READMEs.](#)'. The '.gitignore' section shows a template of 'None'. The 'Choose a license' section shows a license of 'None'. A note at the bottom states 'This will set `main` as the default branch. Change the default name in your [settings](#).'. A final note says 'You are creating a public repository in your personal account.'. A green 'Create repository' button is at the bottom right.

**Paso 2:** Clonar el repositorio a tu máquina local Copia la URL del repositorio (usualmente algo como <https://github.com/manfredialdo/conflict-exercise.git>).

<https://github.com/manfredialdo/conflict-exercise.git> Abre la terminal o línea de comandos en tu máquina. Clona el repositorio usando el comando:

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

Entra en el directorio del repositorio:

```
cd conflict-exercise
```

**Paso 3: Crear una nueva rama y editar un archivo**

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

```
echo "Este es un cambio en la feature branch." >> README.md
```

```
git add README.md
```

```
git commit -m "Paso 3: Added a line in feature-branch"
```

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

```
git checkout main
```

```
echo "Este es un cambio en la main branch." >> README.md
```



```
git add README.md
```

```
git commit -m "Paso 4: Added a line in main branch"
```

#### **# Paso 5: Hacer un merge y generar un conflicto**

```
git merge feature-branch
```

#### **# Paso 6: Resolver el conflicto**

```
# (Aquí abrimos el archivo README.md con un editor de texto en la terminal)
```

```
# Eliminamos lo verde de README.mc.
```

```
# Guarda y cierra el archivo.
```

```
git add README.md
```

```
git commit -m "Paso 6: Resolved merge conflict"
```

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

```
git push origin main
```

```
git push origin feature-branch
```

#### **# Paso 8: Verificar en GitHub**

```
# (Este paso se realiza a través del navegador) https://github.com/manfredialdo?tab=repositories
```