

Resolución TP2: Git & GitHub

Alumna Bustammante Erica

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

1. ¿Qué es GitHub?

Es una plataforma que ofrece alojamiento de repositorios de control de versiones, que permite a los desarrolladores almacenar y gestionar sus proyectos de software. Es una herramienta gratuita y de código abierto que permite el trabajo en equipo en proyectos, el intercambio de código y el trabajo conjunto de forma eficiente.

2. ¿Cómo crear un repositorio en GitHub?

Para empezar a utilizar GitHub, primero debes crear una cuenta. Puedes hacerlo en la página web de GitHub. Una vez que tengas una cuenta, puedes configurar tu perfil y empezar a crear repositorios.

Si es la primera vez utilizando esta plataforma y queremos desde nuestro git subir un repositorio a github debemos de clicar en New, una vez allí colocar el nombre, alguna que otra descripción de lo que vamos a subir, si queremos que sea público o privado, etc.



Una vez que esta todo como nos parezca, damos click en Create Repository. Nos saldrán varias líneas de comandos como:

...or create a new repository on the comand line: Nos indica cada uno de los pasos que debemos de hacer para poder mandar desde nuestro repositorio local al repositorio que hemos creado en github y enlazarlos (recomendado cuando empiezas desde 0).

...or push an existing repository from the command line: Este caso hace referencia a que ya tienes un repositorio local y solamente deseas mandarlo a la plataforma. Para ello deberás de tipear los siguientes comandos en el bash:

```
$ git remote add origin https://github.com/tucanal/nombre-repo.git
```

```
$ git push -u origin master
```

Es cuestión de refrescar la página de github y podrás ver como se ha subido nuestro repositorio.

Si por alguna razón nos vamos a otra computadora y en esta creamos alguna carpeta y necesitamos colocar en ella nuestro trabajo que hemos estado realizando en el repositorio, simplemente podemos clonar ese proyecto, seleccionamos simplemente con un click en “Clone or download” y una vez allí copiamos la dirección. Nos colocamos en la carpeta deseada para hacer la clonación y tipeamos el siguiente comando:

```
$ git clone https://github.com/tucanal/repositorio.git
```

Una vez que se ha clonado correctamente todo, podemos realizar algún cambio, es decir, modificamos el contenido de algún archivo, lo agregamos al stage y lo commiteamos. Logrado esto, deberemos colocar en el bash el siguiente comando para que los cambios que hemos realizados desde este nuevo repositorio local al de github:

```
$ git push origin master
```

(origin especifica que quieres enviar los cambios al repositorio remoto vinculado con ese nombre y master es la rama que estás enviando. Esto significa que estás subiendo los cambios de tu rama master local al repositorio remoto en la rama master.)

Si refrescamos nuestro repositorio, podremos observar los cambios que hemos realizado (recordar que hay que ingresar contraseña y usuario para poder realizar estos cambios desde otro repositorio local).

3. ¿Cómo crear una rama en Git?

La rama “master” en Git, no es una rama especial. Es como cualquier otra rama. La única razón por la cual aparece en casi todos los repositorios es porque es la que crea por defecto el comando git init.

Para crear una nueva rama, se usará el comando git branch:

```
$ git branch nuevaRama
```

Git sabe en qué rama estás mediante un apuntador especial denominado HEAD. Que estemos creando una nueva rama con `git branch` no significa que estemos en dicha rama, estaríamos en la rama master.

4. ¿Cómo cambiar a una rama en Git?

Para saltar de una rama a otra, tienes que utilizar el comando `git checkout`:

```
$ git checkout nuevaRama
```

Esto sí mueve el apuntador HEAD a la rama nuevaRama.

Para crear una nueva rama y saltar a ella, en un solo paso, puedes utilizar el comando `git checkout` con la opción `-b`:

```
$ git checkout -b nuevaRama
```

5. ¿Cómo fusionar ramas en Git?

Para fusionar ramas en Git, sigues un proceso que combina los cambios de una rama en otra.

Primero, debes estar en la rama a la que quieres fusionar los cambios. Por lo general, es a la rama principal (master o main) o cualquier otra rama en la que estés integrando los cambios.

Pongamos de ejemplo que queremos fusionar la rama 'ramaNueva' a la rama master, entonces vamos a pararnos en la principal:

```
$ git checkout master
```

Una vez en la rama de destino, usa el comando `git merge` para fusionar la rama de origen en la rama actual:

```
$ git merge nuevaRama
```

Este comando incorpora los cambios de nuevaRama en master.

6. ¿Cómo crear un commit en Git?

Crear un commit en Git es el proceso mediante el cual se guardan los cambios realizados en el repositorio. Un commit captura el estado actual del código en un momento dado y lo almacena en el historial del proyecto.

Pasos para hacer un commit:

Primero realiza los cambios en los archivos de tu proyecto que desees guardar en el commit.

Luego debes agregar los cambios al área de preparación. Esto se hace con el comando `git add`. Puedes agregar archivos específicos o todos los archivos modificados:

```
$ git add archivo1 (para agregar el archivo "archivo1") o $git add . (para agregar todos los archivos)
```

Una vez que los cambios están en el área de preparación, puede hacer el commit con el comando `git commit`. Debes proporcionar un mensaje de commit que describa los cambios realizados:

```
$ git commit -m "Mensaje de los cambios"
```

Esto guarda el estado actual del código en el historial del repositorio con el mensaje correspondiente.

7. ¿Cómo enviar un commit a GitHub?

Primero debemos clonar el repositorio de GitHub a nuestra

máquina local: `git clone`

```
https://github.com/tu_usuario/tu_repositorio.git cd
```

```
tu_repositorio
```

Haz cambios en los archivos del repositorio y agrégalos:

```
git add nombre_del_archivo o git add .
```

Crea un commit de tus cambios:

```
git commit -m "Descripción de los cambios"
```

Para enviar los commits al repositorio en GitHub, debemos empujarlos con:

```
git push origin nombre_de_la_rama
```

8. ¿Qué es un repositorio remoto?

Para poder colaborar en cualquier proyecto Git, necesitas saber cómo gestionar repositorios remotos. Los repositorios remotos son versiones de tu proyecto que están hospedadas en Internet o en cualquier otra red. Puedes tener varios de ellos, y en cada uno tendrás generalmente permisos de solo lectura o de lectura y escritura. Colaborar con otras personas implica gestionar estos repositorios remotos enviando y trayendo datos de ellos cada vez que necesites compartir tu trabajo. Gestionar repositorios remotos incluye saber cómo añadir un repositorio remoto, eliminar los remotos que ya no son válidos, gestionar varias ramas remotas, definir si deben rastrearse o no y más.

9. ¿Cómo agregar un repositorio remoto a Git?

Utiliza el comando `git remote add` para vincular tu repositorio local con un repositorio remoto y asignar un nombre a ese remoto:

```
$ git remote add [nombre] [url]
```

Ponerle un nombre te ayuda a referenciarlo de manera más fácil en lugar de usar la URL completa. Puedes usar el nombre en la línea de comandos.

Para asegurarte de que el remoto se ha agregado correctamente y verificar el nombre que le has dado, usa el comando:

```
$ git remote -v
```

Esto mostrará una lista de todos los remotos configurados con sus URLs:

[nombre] [url]

Para traer toda la información del repositorio remoto que aún no tienes en tu repositorio local, usa el comando `git fetch` seguido del nombre del remoto:

```
$ git fetch [nombre]
```

Este comando descarga todos los cambios del repositorio remoto asociado con el nombre, pero no fusiona esos cambios con tu rama actual. Es útil para ver qué cambios están disponibles en el remoto.

10. ¿Cómo empujar cambios a un repositorio remoto?

Antes de empujar tus cambios, es una buena práctica obtener los últimos cambios del repositorio remoto para evitar conflictos:

```
git pull origin nombre_de_la_rama
```

Empuja tus cambios al repositorio remoto:

```
git push origin nombre_de_la_rama
```

11. ¿Cómo tirar de cambios de un repositorio remoto?

Como mencionamos en el punto anterior, para tirar los cambios del repositorio remoto Usa el comando `git pull` para descargar y fusionar los cambios del repositorio remoto con tu rama local:

```
git pull origin nombre_de_la_rama
```

Por ejemplo, si estás trabajando en la rama `main`:

```
git pull origin main
```

12. ¿Qué es un fork de repositorio?

En muchas ocasiones podemos ver en github repositorios que son de nuestro agrado y donde nosotros queremos tener una copia y hacerle algunas modificaciones, para ello github nos ofrece la implementación de fork.

Buscamos algún repositorio que nos parezca interesante. Este puede ser cualquier proyecto de código abierto que quieras modificar o utilizar como base para tu propio trabajo.

En la página del repositorio, busca el botón "Fork" ubicado en la parte superior derecha de la página. Haz clic en el botón "Fork". Esto creará una copia del repositorio en tu propia cuenta de GitHub. Esta copia será independiente del repositorio original. Cualquier cambio que realices en tu fork no afectará al repositorio original, los cambios que hagamos no irán al repositorio original del autor de ese proyecto, si no a nuestra copia local. Entonces solo resta clonar ese repositorio que está en nuestro canal para poder trasladarlo a alguna carpeta deseada y seguir trabajando desde nuestro repositorio local.

13. ¿Cómo crear un fork de un repositorio?

Para entender esto puedes por ejemplo crear una cuenta adicional en github para que puedas desde allí hacer un Fork de alguno de los repositorios de tu cuenta original, clonarlo en alguna carpeta y en algún archivo que nosotros notamos que podría mejorarse algo, procedemos a realizar algún cambio, lo agregamos al stage y lo commiteamos (todo esto desde nuestro repositorio local remoto) y luego mandamos los cambios a el forking que hicimos con un git push origin master.

Ese cambio que hemos hecho queremos que sea visto por el creador del repositorio original (en este caso nosotros, desde nuestra cuenta original), debemos hacérselo notar puesto que en el repositorio original los cambios que hicimos previamente, no se verán reflejados.

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

Para hacer el pull request nos dirigiremos a la solapa de Pull requests allí daremos click en new pull request, veremos una ventana a modo de resumen en donde se reflejarán los cambios que hemos hecho nosotros en comparación al repositorio original (el código original, mejor dicho). Daremos click en Create pull request donde veremos el asunto (colocamos algún mensaje global) y más abajo tenemos suficiente lugar para poder explayarnos en mencionar por qué ese cambio que hemos realizado nosotros, sería considerado como algo que a el repositorio original le vendrían bien agregarlo.

15. ¿Cómo aceptar una solicitud de extracción?

El autor del repositorio verá en sus pull requests el mensaje que le hemos enviado, para que lo pueda observar y si lo considera realizar el cambio pertinente (además de poder responderle al usuario que le ha propuesto ese cambio). Lo bueno de todo esto es que si el usuario original considera que esta modificación es buena y no genera conflictos con la rama maestra de su repositorio local remoto, puede clickear en Merge pull request y de esta manera sumará a su repositorio los cambios que hizo un usuario (en modo de ayuda).

16. ¿Qué es una etiqueta en Git?

Como muchos VCS, Git tiene la posibilidad de etiquetar puntos específicos del historial como importantes.

Esta funcionalidad se usa típicamente para marcar versiones de lanzamiento (v1.0, por ejemplo).

17. ¿Cómo crear una etiqueta en Git?

Git utiliza dos tipos principales de etiquetas: ligeras y anotadas.

Una etiqueta ligera es muy parecida a una rama que no cambia - simplemente es un puntero a un commit específico.

Sin embargo, las etiquetas anotadas se guardan en la base de datos de Git como objetos enteros. Tienen un checksum; contienen el nombre del etiquetador, correo electrónico y fecha; tienen un mensaje asociado; y pueden ser firmadas y verificadas con GNU Privacy Guard (GPG). Normalmente se recomienda que crees etiquetas anotadas, de manera que tengas toda esta información; pero si quieres una etiqueta temporal o por alguna razón no estás interesado en esa información, entonces puedes usar las etiquetas ligeras.

18. ¿Cómo enviar una etiqueta a GitHub?

Primero, debes crear una etiqueta en tu repositorio local. Puedes crear una etiqueta anotada (que incluye información adicional como el autor y la fecha) o una etiqueta ligera (simplemente un puntero a un commit):

Ej. etiqueta ligera: `git tag v1.0`

Podes verificar las etiquetas que has creado localmente con:

`git tag`

Una vez que has creado la etiqueta en tu repositorio local, necesitas empujarla al repositorio remoto en GitHub. Puedes hacer esto con el siguiente comando: `git push origin v1.0`

(origin es el nombre del repositorio remoto (por defecto suele ser origin) y v1.0 es el nombre de la etiqueta.) Para empujar todas las etiquetas creadas, usar:

`git push origin --tags`

19. ¿Qué es un historial de Git?

El historial de Git es una secuencia de todos los cambios realizados en un repositorio de Git. Cada cambio en el repositorio se guarda como un commit, y cada commit contiene información sobre el estado del proyecto en un momento específico, incluyendo:

Identificador del commit

Autor

Fecha de realización

Mensaje enviado

20. ¿Cómo ver el historial de Git?

Esto lo conseguimos con el comando de Git:

`git log`

Con tipear este comando en el bash de Git podremos apreciar el histórico de commits, estando situados en la carpeta de nuestro proyecto. El listado de commits estará invertido, es decir, los últimos realizados aparecen los primeros.

El comando `git log --oneline` es una forma compacta de visualizar el historial de commits en un repositorio Git. Muestra un resumen conciso de los commits recientes, con cada commit representado en una sola línea.

Si tu proyecto ya tiene muchos commits, quizás no quieras mostrarlos todos, ya que generalmente no querrás ver cosas tan antiguas como el origen del repositorio. Para ver un número de logs determinado introducimos ese número como opción, con el signo "-" delante (-1, -8, -12...). Por ejemplo, esto muestra los últimos tres commits: `git log -3`

Si queremos que el log también nos muestre los cambios en el código de cada commit podemos usar la opción -p. Esta opción genera una salida mucho más larga, por lo que seguramente nos tocará movernos en la salida con los cursores y usaremos CTRL + Z para salir. `git log -2 -p`

21. ¿Cómo buscar en el historial de Git?

Para buscar en el historial de commits de Git, puedes utilizar varios comandos y opciones que te permiten filtrar y localizar commits específicos.

Para buscar commits que contengan una palabra o frase específica en el mensaje de commit, usa `git log` con la opción `--grep`: `git log --grep="palabra clave"`

Para buscar commits que han modificado un archivo específico, usa `git log` seguido del nombre del archivo: `git log -- nombre_del_archivo`

Para buscar commits en un rango de fechas específico, usa las opciones `--since` y `--until`:

`git log --since="2024-01-01" --until="2024-01-31"`

Para encontrar commits hechos por un autor específico, usa `--author`:

`git log --author="Nombre del Autor"`

22. ¿Cómo borrar el historial de Git?

El comando `git reset` se utiliza sobre todo para deshacer las cosas, como posiblemente puedes deducir por el verbo. Se mueve alrededor del puntero HEAD y opcionalmente cambia el índice o área de ensayo y también puede cambiar opcionalmente el directorio de trabajo si se utiliza `- hard`. Esta última opción hace posible que este comando pueda perder tu trabajo si se usa incorrectamente, por lo que asegúrese de entenderlo antes de usarlo.

Existen distintas formas de utilizarlo:

- `git reset` -> Quita del stage todos los archivos y carpetas del proyecto.
- `git reset nombreArchivo` -> Quita del stage el archivo indicado.
- `git reset nombreCarpeta/` -> Quita del stage todos los archivos de esa carpeta.
- `git reset nombreCarpeta/nombreArchivo` -> Quita ese archivo del stage (que a la vez está dentro de una carpeta).
- `git reset nombreCarpeta/*.extensión` -> Quita todos los archivos que cumplan con la condición indicada previamente dentro de esa carpeta del stage.

23. ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un tipo de repositorio en el que el contenido solo es accesible para usuarios específicos que han sido autorizados. A diferencia de los repositorios públicos, donde cualquier persona puede ver y clonar el contenido, un repositorio privado limita el acceso a los colaboradores que tú elijas. Esto es útil para proyectos que contienen

información sensible o que aún están en desarrollo y no deseas que estén disponibles públicamente.

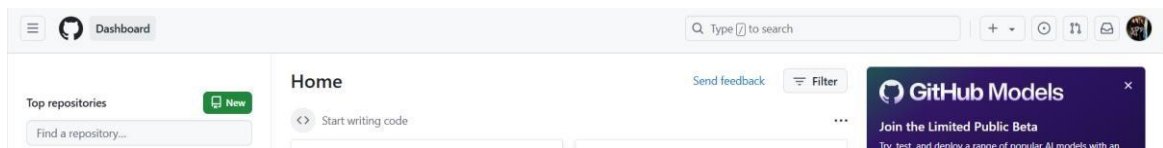
24. ¿Cómo crear un repositorio privado en GitHub?

Pasos:

Inicia sesión en GitHub

Ingresa a la página de creación de repositorios:

En la esquina superior derecha de la página principal, debes hacer clic en el botón “+” y seleccionar “New Repository” o hacer clic en “New”:



Completar la información del repositorio (nombre del repositorio, descripción) Seleccionar la configuración de privacidad:

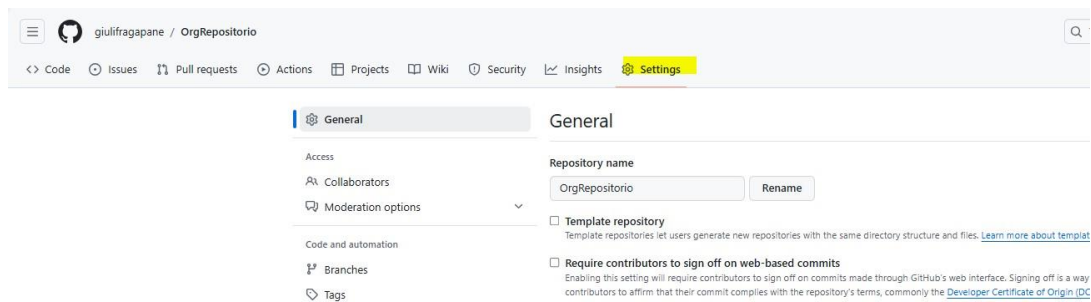


Esto asegura que el repositorio será privado y solo accesible para los colaboradores que tu elijas.

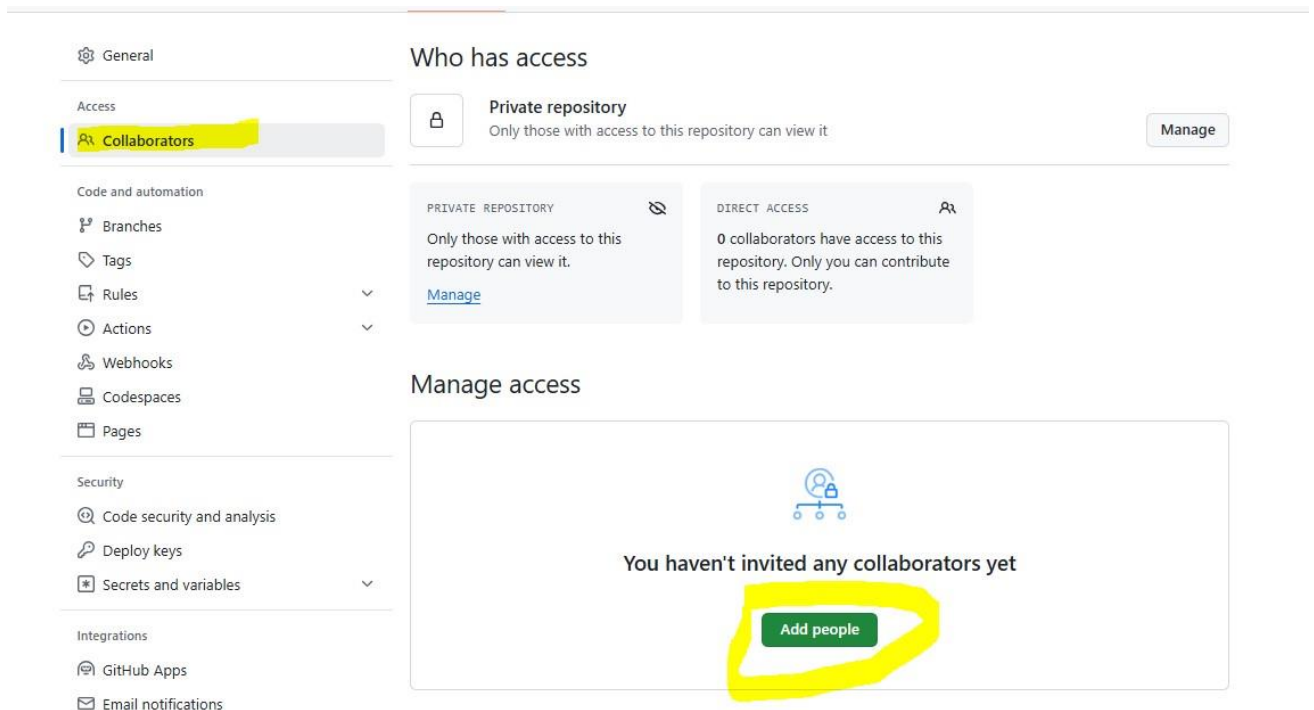
25. ¿Cómo invitar a alguien a un repositorio privado en GitHub?

Invitar a alguien a un repositorio privado en GitHub es un proceso sencillo, pero requiere permisos adecuados.

Accede al repositorio, **haz clic en la pestaña "Settings"** del repositorio. Está en la parte superior del repositorio, junto a las pestañas como "Code" y "Issues".



Selecciona **"Collaborators"** en el menú de la izquierda. Esto te llevará a la página donde puedes administrar colaboradores.



En la sección "Collaborators", haz clic en el botón **"Add people"** e ingresa el nombre de usuario de GitHub de la persona que deseas invitar. Selecciona el nivel de acceso que deseas otorgar: **Read, Triage, Write, Maintain, o Admin**. Haz clic en el botón **"Add"** para enviar la invitación.

26. ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio cuyo contenido es accesible a cualquier persona en Internet. A diferencia de un repositorio privado, que está restringido a un grupo específico de colaboradores, un repositorio público permite que cualquier persona pueda ver, clonar y, si tienen los permisos adecuados, contribuir al proyecto.

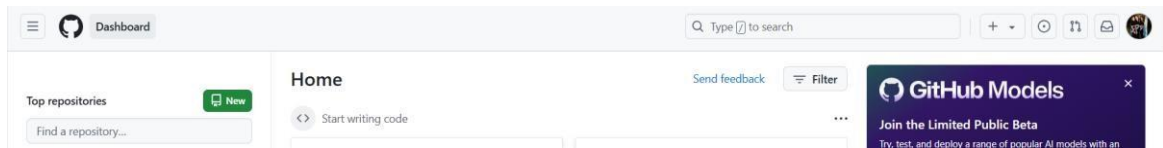
27. ¿Cómo crear un repositorio público en GitHub?

Pasos:

Inicia sesión en GitHub

Ingresa a la página de creación de repositorios:

En la esquina superior derecha de la página principal, debes hacer clic en el botón “+” y seleccionar “New Repository” o hacer clic en “New”:



Completar la información del repositorio (nombre del repositorio, descripción)

Seleccionar la configuración de privacidad:

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 *

 giulifragapane /

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

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.](#)

 You are creating a public repository in your personal account.

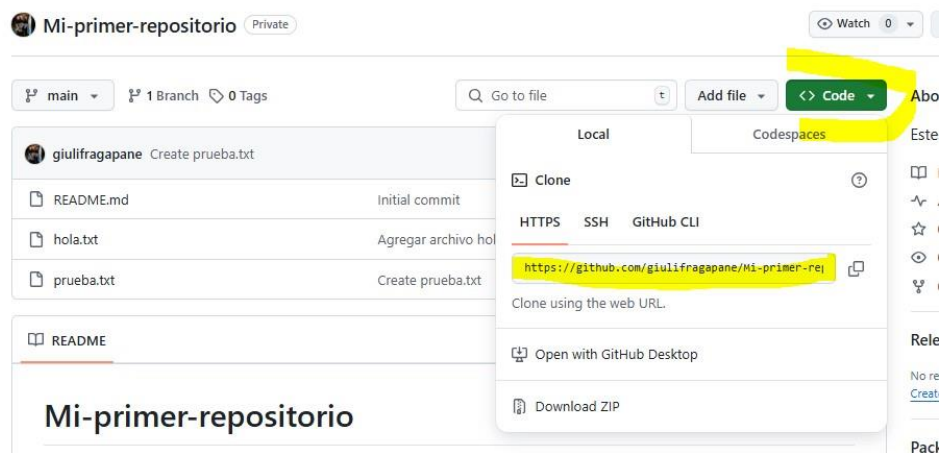
Create repository

Esto asegura que el repositorio será público.

28. ¿Cómo compartir un repositorio público en GitHub?

La forma más sencilla de compartir tu repositorio es proporcionar el enlace directo al mismo.

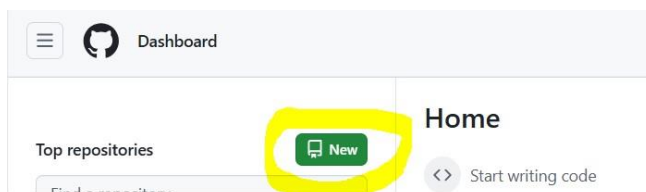
Accede a tu repositorio, copia la URL de tu repositorio que se encuentra en un cuadro de texto que dice "<> Code":



Puedes copiar la URL directamente haciendo clic en el botón de copiar a la derecha de la URL.

2) Realizar la siguiente actividad:

Crear un repositorio.



Dale un nombre al repositorio.

Elije el repositorio sea público.


Inicializa el repositorio con un archivo.

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 *

 giulifragapane

Repository name *


OrgRepositorio

✓ OrgRepositorio is available.


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

Description (optional)

Este es mi repositorio de Organización Empresarial

☒  **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.](#)

- **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).**

```
C:\Users\Giuliana>git clone https://github.com/giulifragapane/OrgRepositorio.git
Cloning into 'OrgRepositorio'...
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.
```

```
C:\Users\Giuliana>cd OrgRepositorio
```

```
C:\Users\Giuliana\OrgRepositorio>echo "Este es mi archivo (Giuliana)" > mi-archivo.txt
```

```
C:\Users\Giuliana\OrgRepositorio>git add .
```

```
C:\Users\Giuliana\OrgRepositorio>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
```

```
C:\Users\Giuliana\OrgRepositorio>git commit -m "Agregando mi archivo.txt"
```

```
[main e1cd9dc] Agregando mi archivo.txt
```

```
1 file changed, 1 insertion(+)
```

```
create mode 100644 mi-archivo.txt
```