

Estudiante: Villarruel Joaquin

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

- ¿Qué es GitHub?

Una comunidad que nos permite compartir los repositorios de manera publica o privada, permitiendo que nosotros o otras personas puedan acceder al código para realizar modificaciones.

- ¿Cómo crear un repositorio en GitHub?

Se debe acceder a nuestro al perfil de github, clikear sobre el botón +, nombrar el repositorio, elegir si lo queremos publico o privado y luego debemos subir los archivos desde nuestra computadora.

- ¿Cómo crear una rama en Git?

Para crear un rama se utiliza el comando `git Branch nombre-rama`.

- ¿Cómo cambiar a una rama en Git?

Para cambiar de rama se utiliza el comando `git checkout nombre-rama`.

- ¿Cómo fusionar ramas en Git?

Para funcionar ramas se utiliza el comando `git merge rama`.

- ¿Cómo crear un commit en Git?

Para crear un commit se utiliza el comando `git commit -m "Mensaje"`.

- ¿Cómo enviar un commit a GitHub?

Se envia con el comando `git push`.

- ¿Qué es un repositorio remoto?}

Un repositorio remoto es un lugar en el que almacena el historial de cambios de un proyecto, al ser remoto se almacena en un servidor en línea permitiendo colaborar con otras personas y tener una copia de seguridad del proyecto.

- ¿Cómo agregar un repositorio remoto a Git?

Se utiliza el comando `git clone url`.

- ¿Cómo empujar cambios a un repositorio remoto?

Utilizamos el comando `git push -u origin master`, solo para la primera vez, luego solo `git push`.

- ¿Cómo tirar de cambios de un repositorio remoto?

Mediante el comando `git pull`.

- ¿Qué es un fork de repositorio?

Es una copia de un repositorio existente la cual podemos guardar en nuestro perfil para realizar cambios y no afectar al repositorio original.

- ¿Cómo crear un fork de un repositorio?

Vamos al repositorio que deseamos y luego hacemos click en el botón fork, elegimos nuestra descripción, nombre y luego hacemos click en crear fork.

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

Clonamos un repositorio, creamos una nueva rama, realizamos las modificaciones, subimos las modificaciones, luego vamos al repositorio y estará la opción para crear el pull request debemos agregar la descripción y crear la solicitud.

- ¿Cómo aceptar una solicitud de extracción?

Vamos al repositorio en donde se encuentran la solicitud, entramos en la pestaña **Pull requests**, cliqueamos en **Merge pull request**, confirmamos la fusión haciendo click en **Confirm merge**.

- ¿Qué es un etiqueta en Git?

Es un marcador que se utiliza para identificar puntos específicos en el historial del repositorio

- ¿Cómo crear una etiqueta en Git?

Se utiliza el comando **git tag -a nombre -m mensaje**,

- ¿Cómo enviar una etiqueta a GitHub?

Se envía utilizando git push nombre

- ¿Qué es un historial de Git?

Es un registro con todos los cambios realizados al repositorio, tiene la información de cada commit es decir la fecha, autor, el mensaje y los cambios en los archivos.

- ¿Cómo ver el historial de Git?

Se puede ver mediante el siguiente comando **git log**

- ¿Cómo buscar en el historial de Git?

Se puede buscar por una palabra clave, autor, modificación de un archivo específico y entre fechas

- ¿Cómo borrar el historial de Git?

Mediante git reset --hard

- ¿Qué es un repositorio privado en GitHub?

Un repositorio al que únicamente las personas con acceso pueden ingresar a ver, clonar o modificar el código.

- ¿Cómo crear un repositorio privado en GitHub?

Para crearlo debemos seguir los mismos pasos como si fuésemos a crear un repositorio normal solo que seleccionando la opción de privado.

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

Debemos ir a los ajustes de nuestro repositorio e invitar a las personas mediante su nombre o mail.

- ¿Qué es un repositorio público en GitHub?

Un repositorio público a diferencia de uno privado es uno al que puede acceder cualquier persona ya sea a verlo o clonarlo.

- ¿Cómo crear un repositorio público en GitHub?

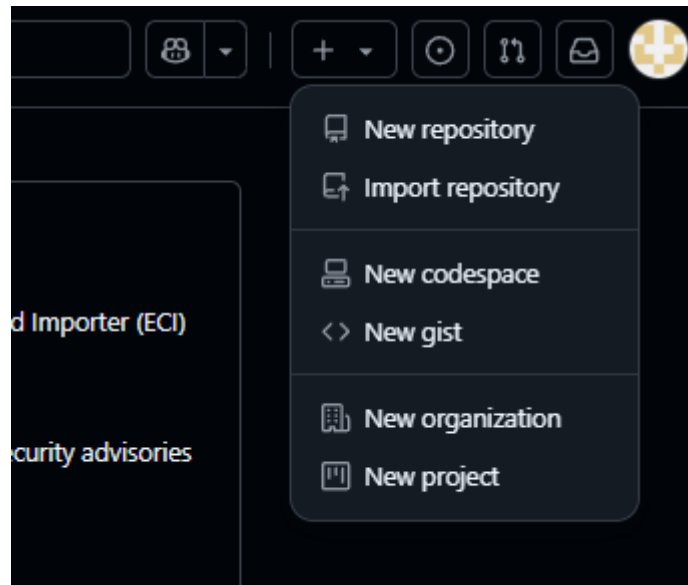
Seguimos la secuencia de pasos para crear un repositorio y al momento de estar creándolo seleccionamos la opción de público.

- ¿Cómo compartir un repositorio público en GitHub?

Lo podemos compartir mediante el link del mismo

2)

- Crear un repositorio.



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 * JoaquinVillarruel / **Repository name *** Repo-Ejemplo

✓ Repo-Ejemplo is available.

Great repository names are short and memorable. Need inspiration? How about **cautious-funicular** ?

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

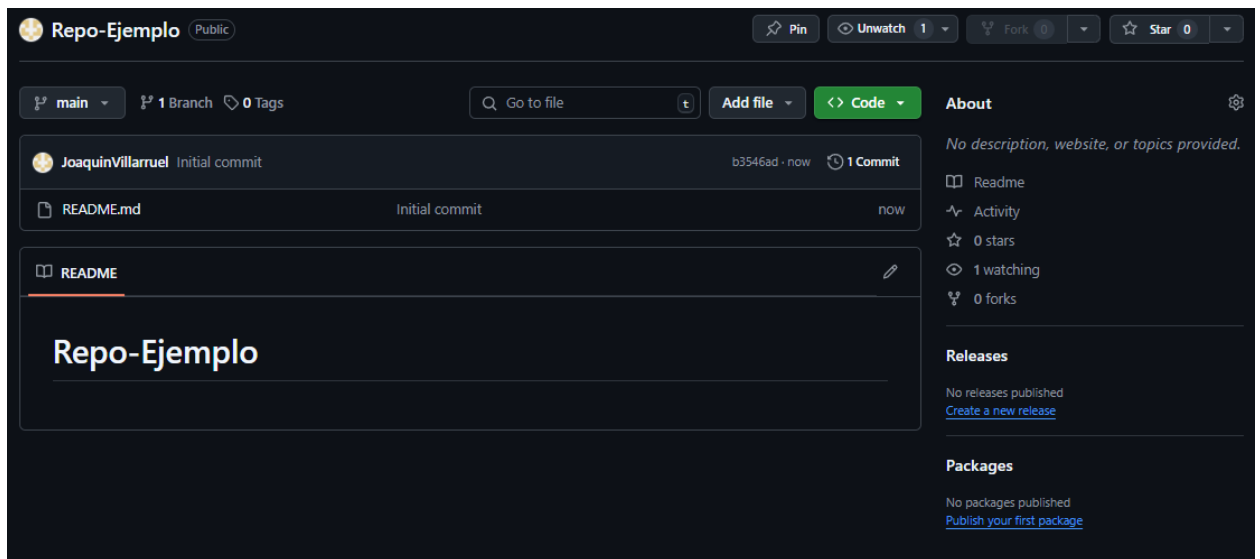
License: **None**

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

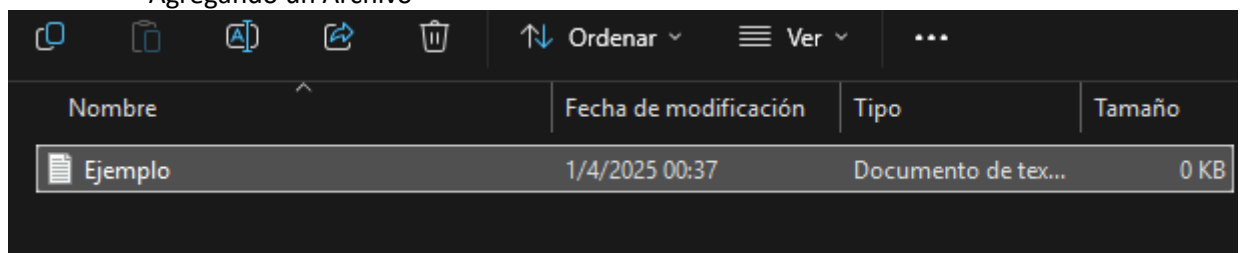
This will set **main** as the default branch. Change the default name in your [settings](#).

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

[Create repository](#)



- Agregando un Archivo



```

MINGW64:/e/tup/Programacion/tp2 github/Repo-Ejemplo

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo
$ git init
Initialized empty Git repository in E:/tup/Programacion/tp2 github/Repo-Ejemplo/.git/

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (master)
$

```

```

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (master)
$ git add .

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (master)
$

```

```

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (master)
$ git commit -m "Agregando Ejemplo.txt"
[master (root-commit) 1ec6559] Agregando Ejemplo.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Ejemplo.txt

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (master)
$

```

```

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (master)
$ git remote add origin https://github.com/JoaquinVillarruel/Repo-Ejemplo.git

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (master)
$

```

```
Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 293 bytes | 293.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/JoaquinVillarruel/Repo-Ejemplo.git
    eb1dcc6..1385830  main -> main

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (main)
$
```

- Creando Branchs

```
Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (main)
$ git checkout -b "Ejemplo"
Switched to a new branch 'Ejemplo'

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (Ejemplo)
$
```

```
Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (Ejemplo)
$ git add .
```

```
Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (Ejemplo)
$ git commit -m "Agrego Ejemplos.py"
[Ejemplo 299046d] Agrego Ejemplos.py
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 ejemplo3.py

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (Ejemplo)
$ |
```

```
Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (Ejemplo)
$ git push origin Ejemplo
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 477 bytes | 477.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'Ejemplo' on GitHub by visiting:
remote:   https://github.com/JoaquinVillarruel/Repo-Ejemplo/pull/new/Ejemplo
remote:
To https://github.com/JoaquinVillarruel/Repo-Ejemplo.git
 * [new branch]      Ejemplo -> Ejemplo

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/Repo-Ejemplo (Ejemplo)
$
```

Default				
Branch	Updated	Check status	Behind Ahead	Pull request
main 	 1 minute ago		Default	 ...

Your branches				
Branch	Updated	Check status	Behind Ahead	Pull request
Ejemplo 	 1 minute ago		1 0	 #1  ...

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".

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 * JoaquinVillarruel / **Repository name *** conflict-exercise

 conflict-exercise is available.

Great repository names are short and memorable. Need inspiration? How about **musical-spork** ?

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.](#)

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](#)

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
```

```
Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise
$ git clone https://github.com/JoaquinVillarruel/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.
```

```
Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise
$ cd conflict-exercise

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/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"
```

```
Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/conflict-exercise (feature-branch)
$ git add README.md

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/conflict-exercise (feature-branch)
$ git commit -m "Added a line in feature-branch"
[feature-branch eb23533] Added a line in feature-branch
1 file changed, 2 insertions(+), 1 deletion(-)

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/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"
```

```
Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/conflict
-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/conflict
-exercise (main)
$ git add README.md

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/conflict
-exercise (main)
$ git commit -m "Added a line in main branch"
[main 6a12242] Added a line in main branch
1 file changed, 2 insertions(+), 1 deletion(-)

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/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.

```
Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/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.

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/conflict
-exercise (main|MERGING)
$
```

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"
```

Paso 7: Subir los cambios a GitHub

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

```
git push origin main
```

- También sube la feature-branch si deseas:

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

```
Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/conflict
-exercise (main|MERGING)
$ git add README.md

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/conflict
-exercise (main|MERGING)
$ git commit -m "Resolved merge conflict"
[main e14ca2f] Resolved merge conflict

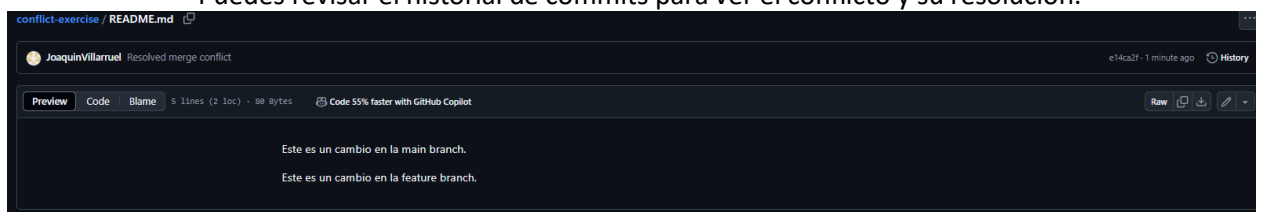
Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/conflict
-exercise (main)
$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 840 bytes | 840.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/JoaquinVillarruel/conflict-exercise.git
2f33d94..e14ca2f  main -> main

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/conflict
-exercise (main)
$ git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:   https://github.com/JoaquinVillarruel/conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/JoaquinVillarruel/conflict-exercise.git
* [new branch]      feature-branch -> feature-branch

Joaquin@DESKTOP-8UNRC1I MINGW64 /e/tup/Programacion/tp2 github/conflict-exercise/conflict
-exercise (main)
```

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.



main 2 Branches 0 Tags

Go to file

Add file

Code

JoaquinVillarruel Resolved merge conflict

e14ca2f · 1 minute ago 4 Commits

README.md

Resolved merge conflict

1 minute ago

README