

TECNICATURA SUPERIOR EN TELECOMUNICACIONES

PROYECTO INTEGRADOR I

Trabajo practico Nº 1

PARTE II

Fundamentos de Git - GitHub

Alumno/a: Romina Huk

Profesor: Cristian Gonzalo Vera



GIT

GIT es el SCV (**sistema de control de versiones**) de código abierto más utilizado que te permite rastrear los cambios realizados en los archivos. Las empresas y los programadores suelen utilizar el GIT para colaborar en el desarrollo de software y aplicaciones.

Un proyecto GIT consta de tres secciones principales: el directorio de trabajo, el área de preparación y el directorio git.

El directorio de trabajo es donde se agregan, borran y editan los archivos. Luego, los cambios son preparados (indexados) en el área de preparación. Después de que confirmes tus cambios, la instantánea de los cambios se guardará en el directorio git.

Todo el mundo puede usar GIT ya que está disponible para Linux, Windows, Mac y Solaris. El software puede tener una fuerte curva de aprendizaje, pero hay muchos tutoriales disponibles para ayudarte.

Comandos de GIT básicos

Aquí hay algunos comandos básicos de GIT que debes conocer:

- **git init** creará un nuevo repositorio local GIT. El siguiente comando de Git creará un repositorio en el directorio actual:

```
git init
```

- Como alternativa, puedes crear un repositorio dentro de un nuevo directorio especificando el nombre del proyecto:

```
git init [nombre del proyecto]
```

- **git clone** se usa para copiar un repositorio. Si el repositorio está en un servidor remoto, usa:

```
git clone nombredeusuario@host:/path/to/repository
```

- A la inversa, ejecuta el siguiente comando básico para copiar un repositorio local:

```
git clone /path/to/repository
```

- **git add** se usa para agregar archivos al área de preparación. Por ejemplo, el siguiente comando de Git básico indexará el archivo temp.txt:

```
git add <temp.txt>
```

- **git commit** creará una instantánea de los cambios y la guardará en el directorio git.

```
git commit -m "El mensaje que acompaña al commit va aquí"
```

- **git config** puede ser usado para establecer una configuración específica de usuario, como el email, nombre de usuario y tipo de formato, etc. Por ejemplo, el siguiente comando se usa para establecer un email:

```
git config --global user.email tuemail@ejemplo.com
```

- La opción -global le dice a GIT que vas a usar ese correo electrónico para todos los repositorios locales. Si quieres utilizar diferentes correos electrónicos para diferentes repositorios, usa el siguiente comando:

```
git config --local user.email tuemail@ejemplo.com
```

- **git status** muestra la lista de los archivos que se han cambiado junto con los archivos que están por ser preparados o confirmados.
- **git push** se usa para enviar confirmaciones locales a la rama maestra de `<origin>`:

```
git push origin <master>
```

Reemplaza <master> con la rama en la que quieres enviar los cambios cuando no quieras enviarlos a la rama maestra.

- **git checkout** crea ramas y te ayuda a navegar entre ellas. Por ejemplo, el siguiente comando crea una nueva y automáticamente se cambia a ella:

```
command git checkout -b <branch-name>
```

- Para cambiar de una rama a otra, sólo usa:

```
git checkout <branch-name>
```

- **git remote** te permite ver todos los repositorios remotos. El siguiente comando listará todas las conexiones junto con sus URLs:

```
git remote -v
```

- Para conectar el repositorio local a un servidor remoto, usa este comando:

```
git remote add origin <host-or-remoteURL>
```

- Por otro lado, el siguiente comando borrará una conexión a un repositorio remoto especificado:

```
g git remote <nombre-del-repositorio>
```

- **git branch** se usa para listar, crear o borrar ramas. Por ejemplo, si quieres listar todas las ramas presentes en el repositorio, el comando debería verse así:

```
git branch
```

- Si quieres borrar una rama, usa:

```
git branch -d <branch-name>
```

git pull fusiona todos los cambios que se han hecho en el repositorio remoto con el directorio de trabajo local.

```
git pull
```

GITHUB

GitHub es una plataforma de gestión y organización de proyectos basada en la nube que incorpora las funciones de control de versiones de Git. Es decir que todos los usuarios de GitHub pueden rastrear y gestionar los cambios que se realizan en el código fuente en tiempo real, a la vez que tienen acceso a todas las demás funciones de Git disponibles en el mismo lugar.

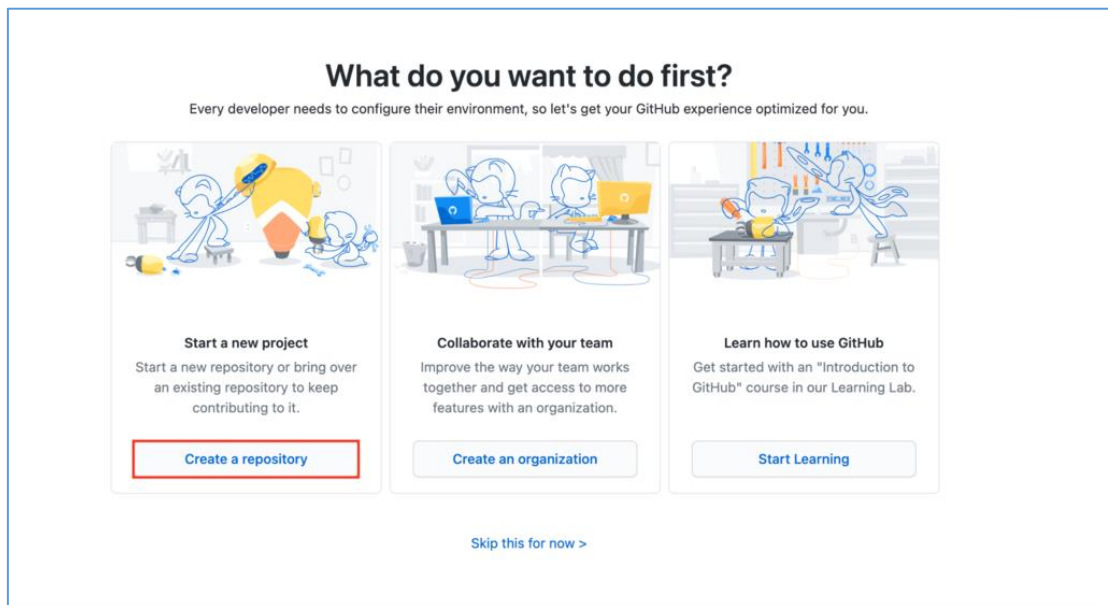
Además, la interfaz de usuario de GitHub es más fácil de usar que la de Git, lo que la hace accesible para personas con pocos o ningún conocimiento técnico. Esto significa que se puede incluir a más miembros del equipo en el progreso y la gestión de un proyecto, haciendo que el proceso de desarrollo sea más fluido.

1. Crear un Repositorio de GitHub

Un repositorio, o repo, será el eje central de tu proyecto. Puede ser un archivo o una colección de archivos que contengan código, imágenes, texto o cualquier otra cosa.

Para comenzar el proceso, sigue estos pasos:

1. Haz clic en **Create a repository** para iniciar un nuevo proyecto.



2. La sección **Owner** ya tendrá el nombre de tu cuenta. Crea un **nombre de repositorio**. Comprueba si está configurado como **Público** para que sea de código abierto, y luego marca la casilla **Add a README file**. Finalmente, haz clic en **Create repository**.

Owner * / Repository name *

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

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:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

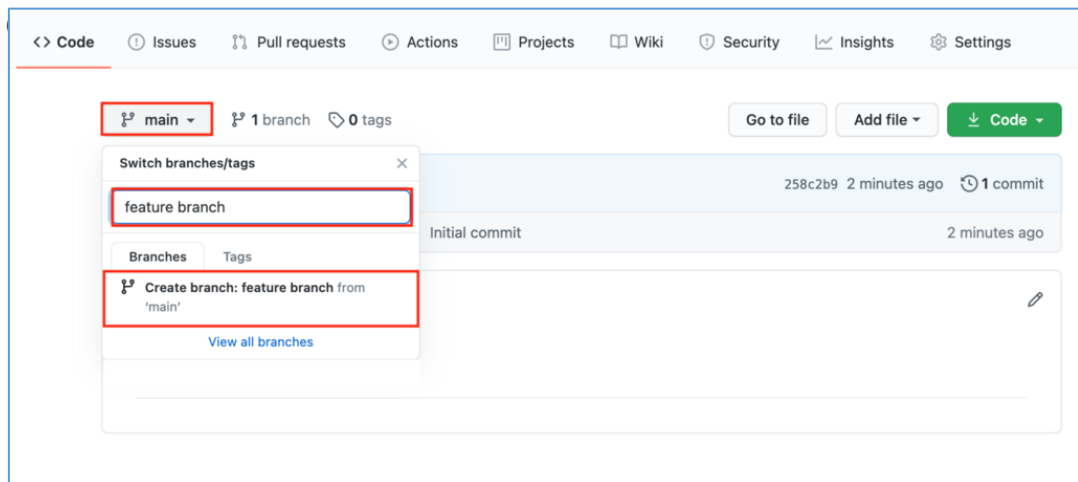
Create repository

Felicitaciones, ya has creado un nuevo repositorio que contendrá el archivo original de tu proyecto. El siguiente paso es aprender lo que puedes hacer con él.

2. Crear ramas en GitHub

Con la creación de ramas, generas diferentes versiones de un repositorio. Al hacer cambios en el proyecto en la rama de características, un desarrollador puede ver cómo afectará al proyecto maestro cuando se integre. Así es como puedes generar una rama de características:

1. Ve a tu nuevo repositorio. Pulsa el botón **main** e introduce el nombre



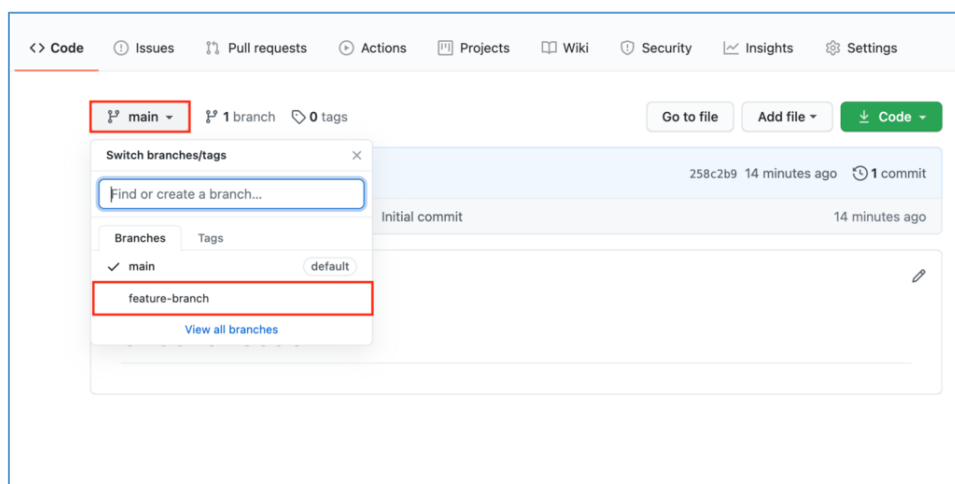
Ahora has creado una rama de características que se ve idéntica a la rama maestra. Puedes empezar a hacer cambios en ella libremente sin afectar al proyecto.

3. Entender los commits de GitHub

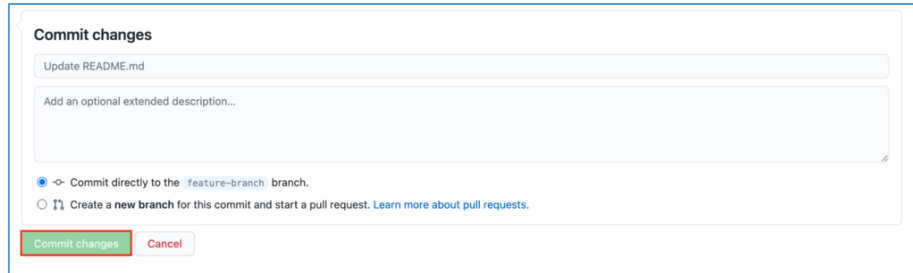
Los commits son la forma en que se denominan los cambios guardados en GitHub. Cada vez que cambies el archivo de la rama de características, tendrás que hacer un **Commit** para mantenerlo.

A continuación te explicamos cómo hacer y confirmar un cambio:

1. Accede a la rama de características haciendo clic en **main** y seleccionando tu rama recién creada en el menú desplegable.



2. Haz clic en el «icono del lápiz» para empezar a editar el archivo. Cuando hayas terminado, escribe una breve descripción de los cambios realizados. Haz clic en **Commit changes**.



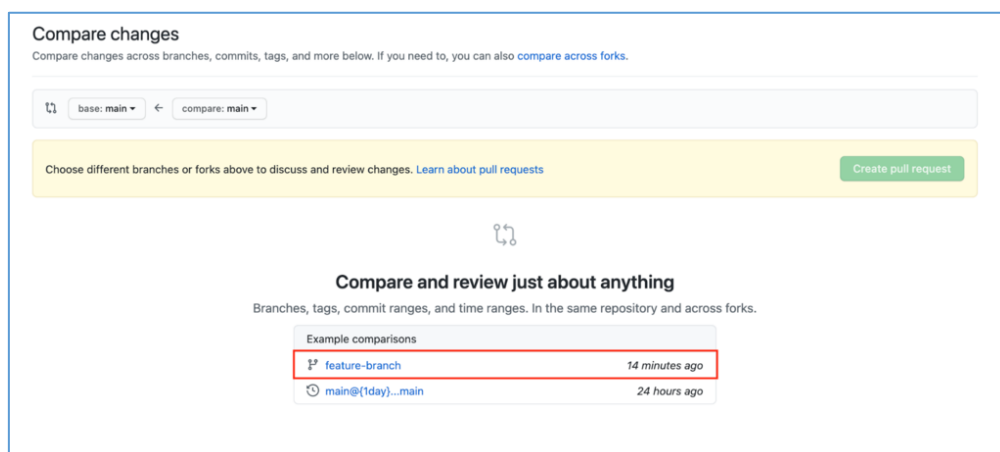
The screenshot shows the 'Commit changes' dialog in GitHub. At the top, it says 'Commit changes'. Below that is a text input field containing 'Update README.md'. Underneath is a larger text area for an optional extended description. At the bottom, there are two radio buttons: the first is selected and labeled 'Commit directly to the feature-branch branch.', and the second is labeled 'Create a new branch for this commit and start a pull request. Learn more about pull requests.' At the very bottom are two buttons: 'Commit changes' (highlighted with a red box) and 'Cancel'.

4. Crear solicitudes de extracción en GitHub

Para proponer los cambios que acabas de hacer a otros desarrolladores que trabajan en el mismo proyecto, debes crear una **solicitud de extracción**. Estas facilitan el trabajo conjunto en los proyectos, ya que son la principal herramienta de colaboración en GitHub.

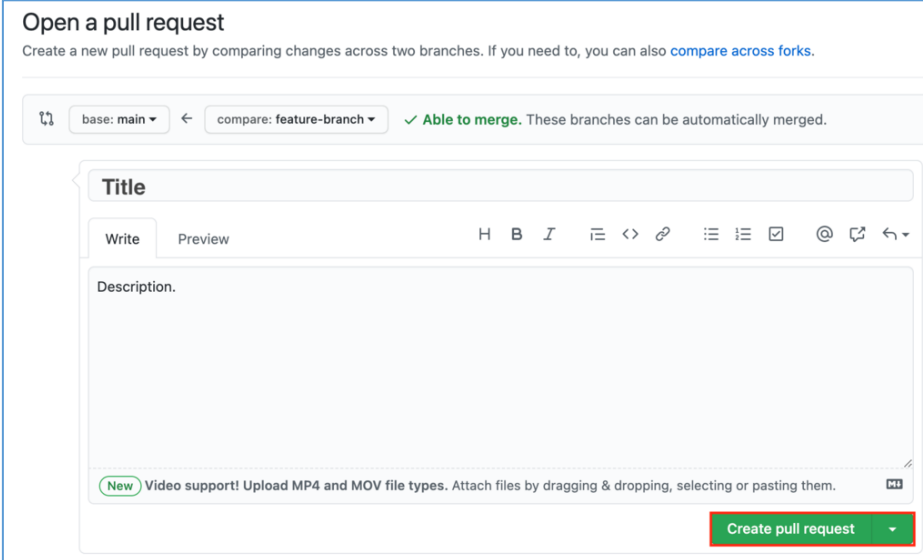
Las solicitudes de extracción te permiten ver las diferencias entre el proyecto original y tu rama de características. Es la forma de pedir a tus compañeros que las revisen. Si los otros desarrolladores lo aprueban, pueden **fusionar la solicitud de extracción**, lo que aplicará esos cambios al proyecto principal. Para hacer una solicitud de extracción sigue los siguientes pasos:

1. Haz clic en **Pull requests -> New pull request**. En **Example comparisons**, selecciona la **rama de características** en la que estabas trabajando.



The screenshot shows the 'Compare changes' page in GitHub. At the top, it says 'Compare changes' and provides a brief explanation. Below that is a search bar with 'base: main' and 'compare: main' selected. A yellow banner prompts the user to 'Choose different branches or forks above to discuss and review changes. Learn about pull requests' with a 'Create pull request' button. The main section is titled 'Compare and review just about anything' and lists 'Example comparisons'. Two comparisons are shown: 'feature-branch' (14 minutes ago) and 'main@{1day}...main' (24 hours ago). The 'feature-branch' comparison is highlighted with a red box.

2. Revisa los cambios una vez más y haz clic en **Create pull request**. En la nueva página, escribe el título y proporciona una breve descripción de lo que has trabajado para promover la fusión. Haz clic en **Create pull request**.



The screenshot shows the GitHub interface for creating a pull request. At the top, it says 'Open a pull request' and 'Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).' Below this, there are two dropdown menus: 'base: main' and 'compare: feature-branch', with a green checkmark and the text 'Able to merge. These branches can be automatically merged.' The main form has a 'Title' field, a 'Write' tab (selected) and a 'Preview' tab, and a large 'Description' text area. At the bottom right, there is a green button labeled 'Create pull request'.

Ahora otros desarrolladores podrán fusionar los cambios que hiciste con los archivos originales del proyecto.

Si quieres saber todo lo demás sobre cómo empezar en GitHub, consulta esta [guía](#) (en inglés).

Conclusión

Aunque GitHub es conocido principalmente dentro de la comunidad de ingenieros de software, puede ser utilizado en una variedad de industrias diferentes. Cualquier equipo o empresa que trabaje en diferentes proyectos que requieran desarrollo en forma de archivos puede utilizar este servicio. Por ejemplo, los equipos de contenido y marketing pueden utilizar GitHub para organizar sus proyectos. Los creativos freelance pueden utilizarlo para gestionar su trabajo cuando trabajan con otras personas.