GUIA DE GIT Y GITHUB

Entorno de desarrollo

1 Tabla de contenido

1.1 Una Introducción a git y al Control de Versiones

Primero, echemos un vistazo a git. Este es un software de control de versiones para desarrolladores:

El control de versiones se refiere al proceso de guardar diferentes archivos o «versiones» a lo largo de las diferentes etapas de un proyecto (compruebe también: control de versiones de WordPress). Esto permite a los desarrolladores hacer un seguimiento de lo que se ha hecho y volver a una fase anterior si deciden que quieren revertir algunos de los cambios que han hecho.

Esto es útil por varias razones. Por ejemplo, facilita la resolución de errores y la corrección de otros errores que puedan ocurrir durante el desarrollo. También puede anotar los cambios en cada versión, para ayudar a cualquier miembro del equipo a mantenerse al día sobre lo que se ha completado y lo que aún queda por hacer.

A diferencia de la mayoría de los otros sistemas de control de versiones (VCS), git almacena cada versión guardada como una 'instantánea' en lugar de una lista de los cambios realizados en cada archivo. Puede hacer referencia a antiguas instantáneas siempre que lo necesite y las nuevas instantáneas se crean cuando se modifica el proyecto.

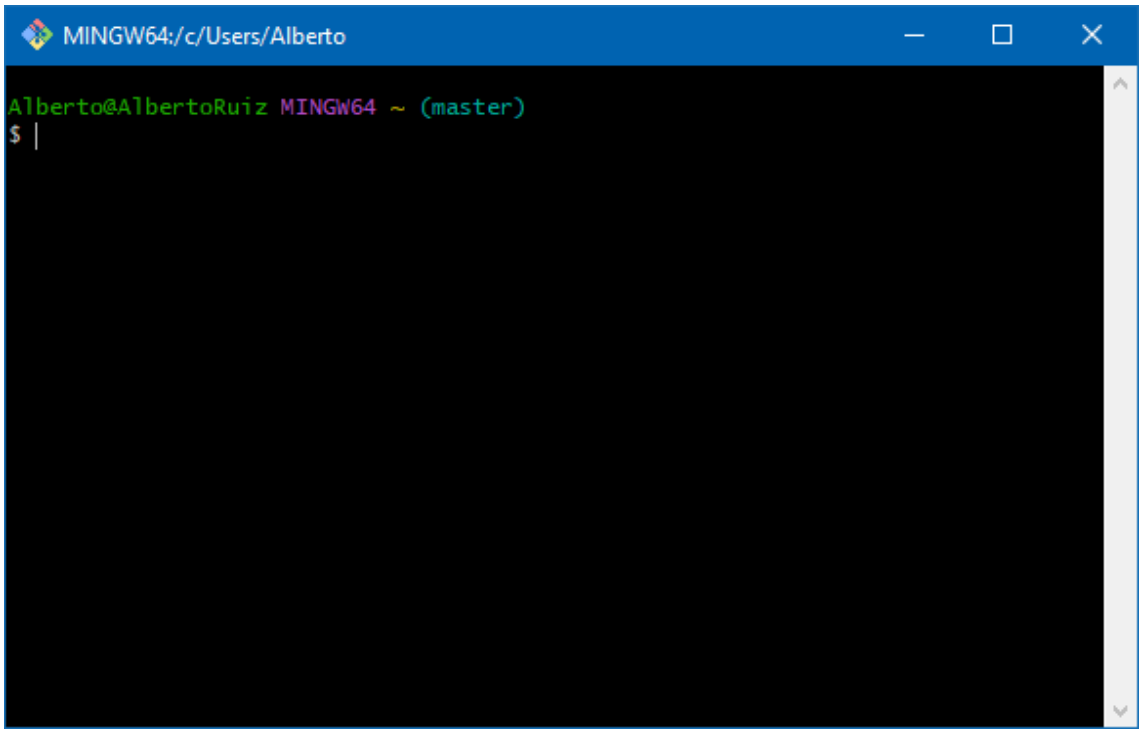
Git también le permite «empujar» y «tirar» de los cambios hacia y desde las instalaciones de otros ordenadores. Esto lo convierte en lo que se conoce como 'Sistema de control de versiones distribuido', y permite que varios desarrolladores trabajen en el mismo proyecto.

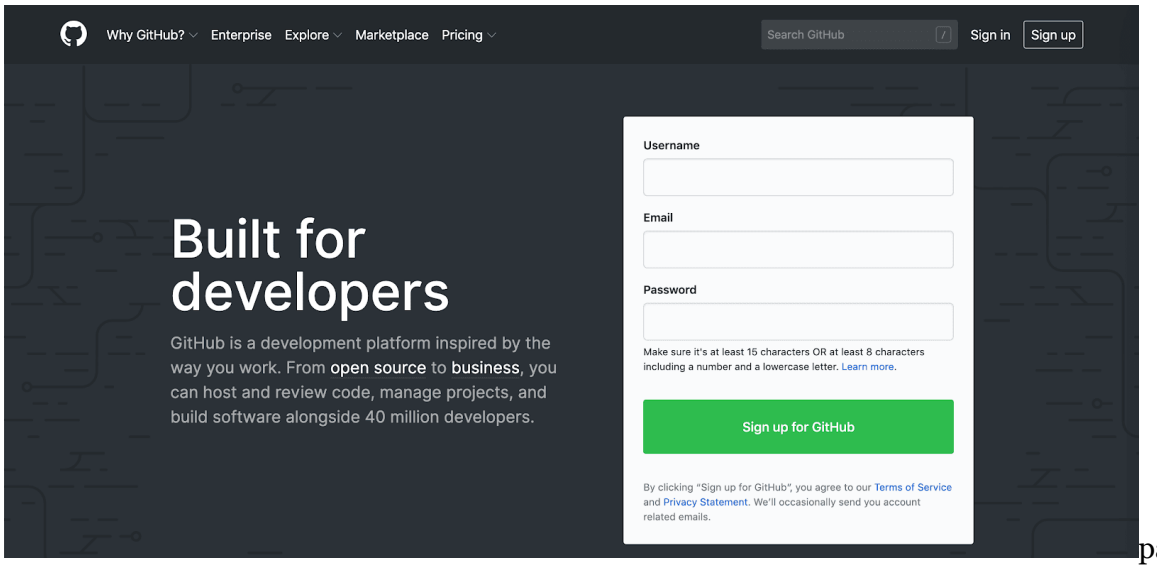
Sin embargo, hay algunos inconvenientes para manejar el desarrollo de esta manera. Como el software local está instalado en su máquina individual, git no puede leer las ediciones que otros desarrolladores puedan estar haciendo en tiempo real. Esto significa que si usted y un compañero de equipo están trabajando en un proyecto simultáneamente, no podrán ver el trabajo del otro.

Así que si no está completamente seguro de cuándo los miembros del equipo trabajarán en su proyecto, git es mejor para el uso individual. De esta manera, puede evitar conflictos o que alguien más anule accidentalmente su código.

1.2 La Diferencia Entre git y GitHub

GitHub facilita la colaboración con git. Es una plataforma que puede mantener repositorios de código en almacenamiento basado en la nube para que varios desarrolladores puedan trabajar en un solo proyecto y ver las ediciones de cada uno en tiempo real:





página de GitHub

Además, también incluye funciones de organización y gestión de proyectos. Puede asignar tareas a individuos o grupos, establecer permisos y roles para los colaboradores y usar la moderación de comentarios para mantener a todos en la tarea.

Además, los repositorios de GitHub están disponibles públicamente. Los desarrolladores de todo el mundo pueden interactuar y contribuir al código de los demás para modificarlo o mejorarlo, lo que se conoce como «codificación social». En cierto modo, esto hace que GitHub sea un sitio de redes para profesionales de la web.

Hay tres acciones principales que puede realizar cuando se trata de interactuar con el código de otros desarrolladores en GitHub:

- **Bifurcación:** El proceso de copiar el código de otra persona del repositorio para modificarlo.

- **Pull:** Cuando haya terminado de hacer cambios en el código de otra persona, puede compartirlos con el propietario original a través de una «solicitud pull».
- **Fusión:** Los propietarios pueden añadir nuevos cambios a sus proyectos a través de una fusión, y dar crédito a los contribuyentes que los han sugerido.

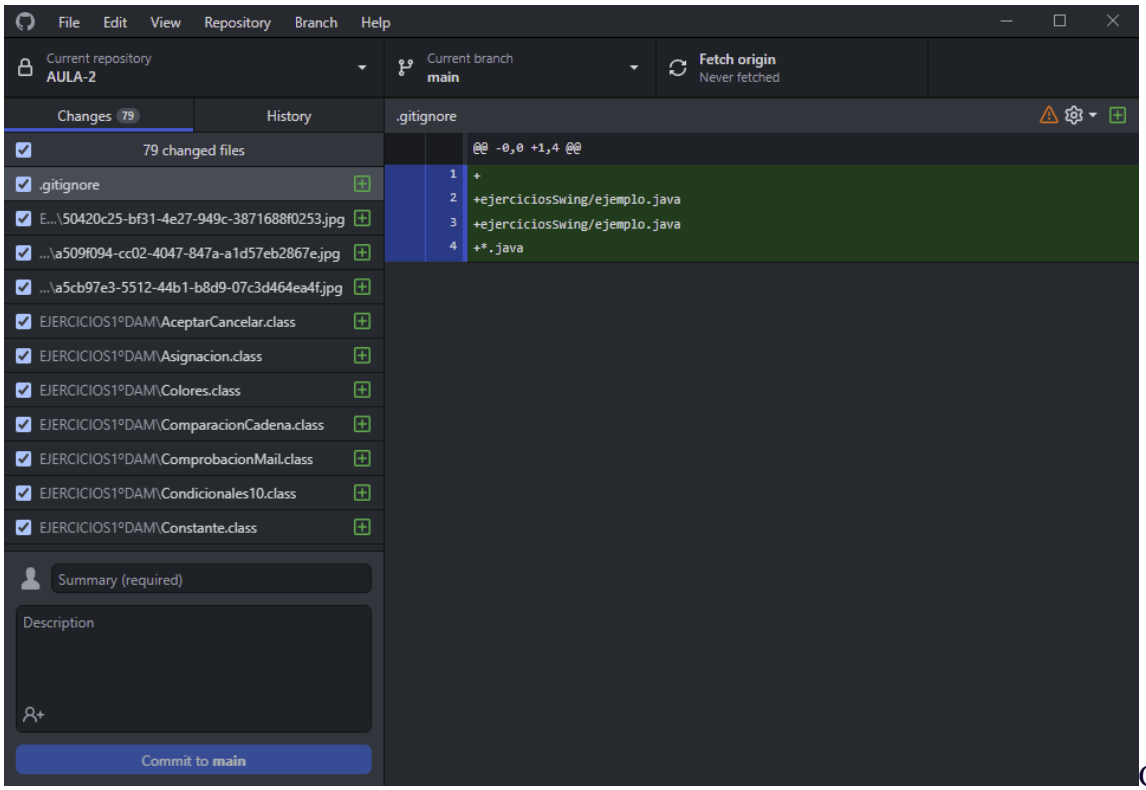
Especialmente para los nuevos desarrolladores que están tratando de construir sus currículums, esta puede ser una gran oportunidad para ganar algo de experiencia. GitHub le permite compartir proyectos en su perfil y mantiene una línea de tiempo de todos aquellos en los que has contribuido.

Para resumir la diferencia entre git vs GitHub:

1. **git es un software de VCS local** que permite a los desarrolladores guardar instantáneas de sus proyectos a lo largo del tiempo. Generalmente es mejor para uso individual.
2. **GitHub es una plataforma basada en la web que incorpora las características de control de versiones de git** para que puedan ser utilizadas de forma colaborativa. También incluye características de gestión de proyectos y equipos, así como oportunidades para la creación de redes y la codificación social.

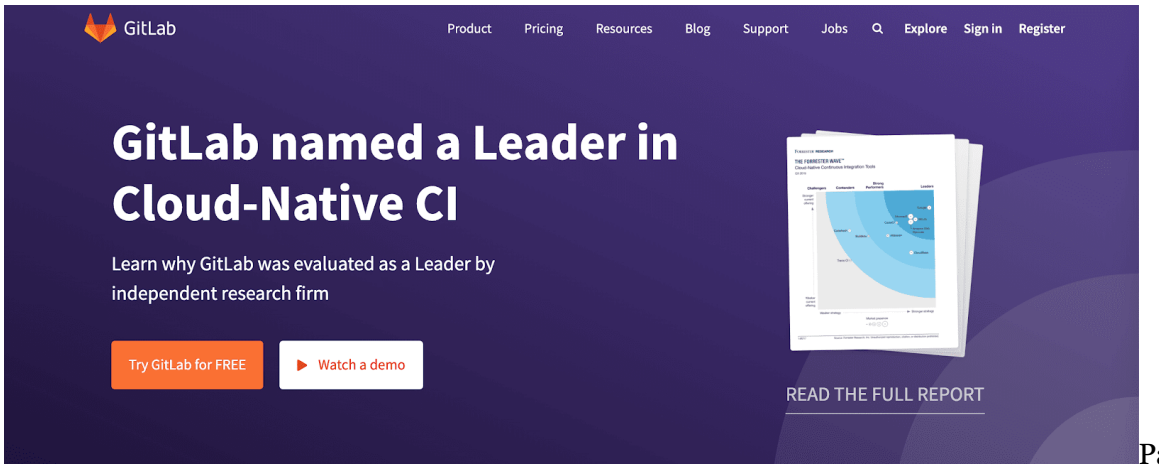
Otra opción interesante es GitHub de escritorio

Entendiendo



GitHub vs GitLab

GitHub no es el único repositorio git que puede considerar si está buscando colaborar en un proyecto de desarrollo. GitLab es otra plataforma muy similar que también vale la pena ver:



Página de GitLab

Al igual que GitHub, GitLab le permite almacenar código y usar las capacidades de control de versiones de git. Sin embargo, también proporciona permisos de usuario más matizados e incluye una integración continua (CI) incorporada. Esto elimina la necesidad de las solicitudes de extracción utilizadas en GitHub.

Dicho esto, GitLab todavía no es tan popular como GitHub. Con 40 millones de usuarios, GitHub ofrece muchas más oportunidades para la creación de redes, la codificación social e incluso para aprender de otros profesionales más experimentados en su campo.

En resumen, ambas plataformas son útiles, pero para situaciones ligeramente diferentes. Si no está interesado en trabajar con desarrolladores fuera de su equipo, GitLab puede ayudarte a acelerar un poco su flujo de trabajo. Sin embargo, GitHub puede ser la mejor ruta para aquellos que buscan crecer en sus carreras.

1.3 La Propiedad y el Costo de git vs GitHub

Dado que están tan estrechamente relacionados, tendría sentido que git y GitHub fueran propiedad de la misma empresa. Por el contrario, git es un software de código abierto, mientras que GitHub es propiedad de Microsoft.

Las plataformas de código abierto – incluyendo git y WordPress – son gratuitas de usar, modificar y distribuir.

El modelo de precios de GitHub es diferente, pero proporciona un interesante plan gratuito. De hecho, todas las características principales de GitHub son gratuitas para todos (en el pasado, el plan Individual Pro costaba 7 dólares al mes). En el caso de los equipos, el precio comienza en 4 dólares por usuario y mes. También puedes buscar las opciones de precios de Enterprise para grupos más grandes que necesiten funciones más avanzadas.

1.4 ¿Cómo Integrar git y GitHub? (en 5 pasos)

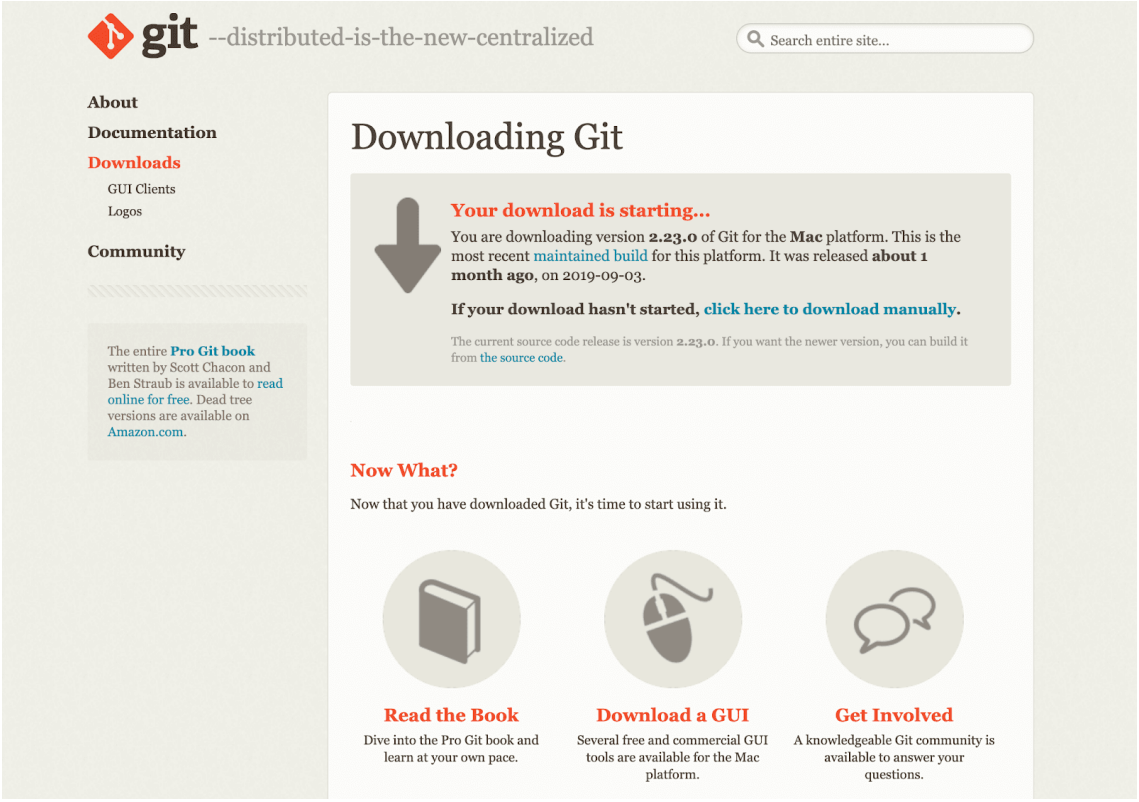
Para poder usar git y GitHub juntos para el control de versiones y la colaboración, hay algunos pasos que tendrá que dar. Echemos un vistazo a cómo funciona ese proceso.

Es importante tener en cuenta que tiene que usar git para aprovechar las ventajas de GitHub, así que querrá refrescar sus habilidades con el primero antes de intentar integrar los dos. También hemos incluido algunas instrucciones básicas a continuación para que pueda empezar.

Si ya está usando Git, puede saltar al paso 2. Por otro lado, los nuevos en este software de control de versiones necesitarán instalarlo y añadir algún código antes de seguir adelante.

1.4.1 Paso 1: Instalar git y Añadir un Repositorio

Primero, descargue el software git para su Sistema Operativo (OS):



Descargar git

Entonces tendrá que ejecutar el instalador. Este proceso variará dependiendo del sistema operativo que esté utilizando. Entonces puedes configurar git usando su interfaz de línea de comandos.

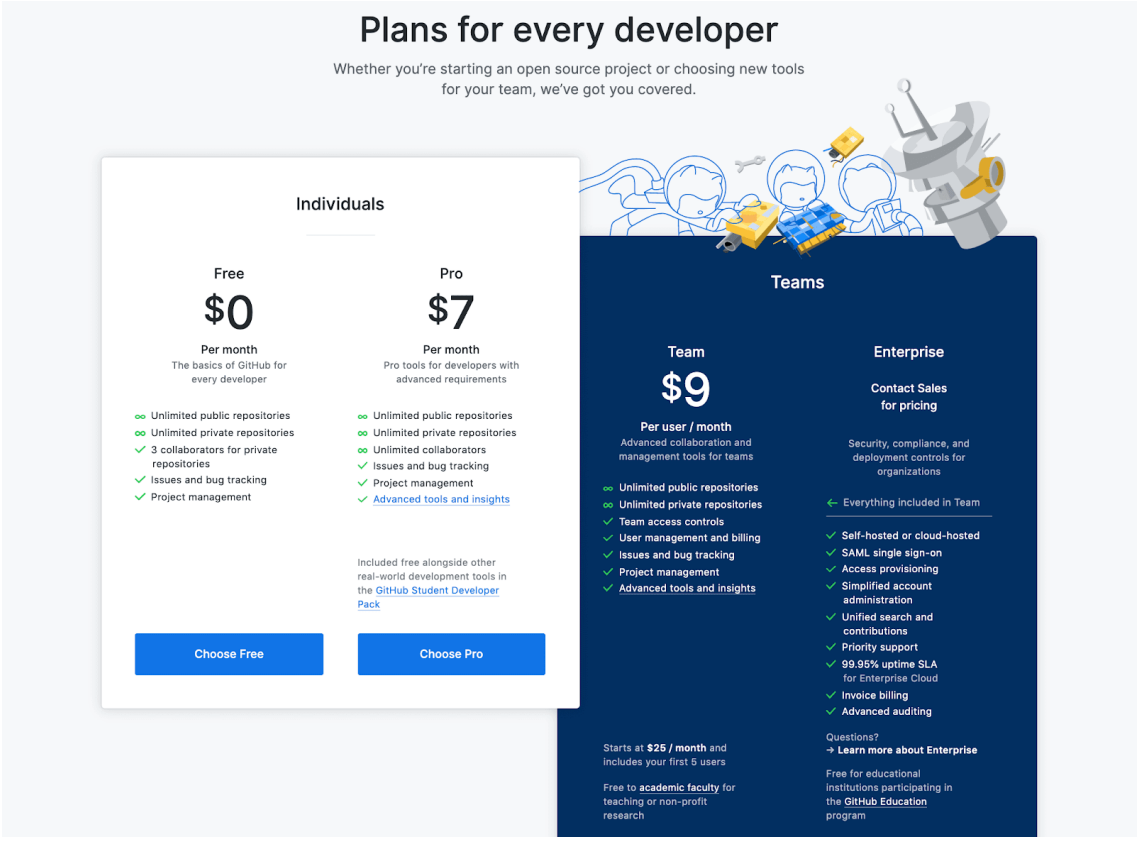
Una vez que esté en funcionamiento, hay algunos términos con los que querrá familiarizarse cuando empiece a usar el software:

- **Repositorio:** La ubicación del archivo donde se almacena su proyecto.
- **Comprometerse:** El comando utilizado para guardar los nuevos cambios en su proyecto en el repositorio.
- **Escenario:** Antes de que puedas confirmar los cambios en Git, necesitas prepararlos - esto le da la oportunidad de preparar su código antes de añadirlo formalmente a su proyecto.
- **Rama:** La parte de su proyecto que está desarrollando activamente.

Para conectar git a GitHub, necesitarás añadir un repositorio y hacer al menos una confirmación. Entonces tendrá suficiente de su proyecto establecido para empezar a trabajar en GitHub.

1.4.2 Paso 2: Crear una cuenta GitHub

A continuación, necesitarás una cuenta GitHub. Puede inscribirse en uno de ellos de forma gratuita o invertir en un plan de pago:



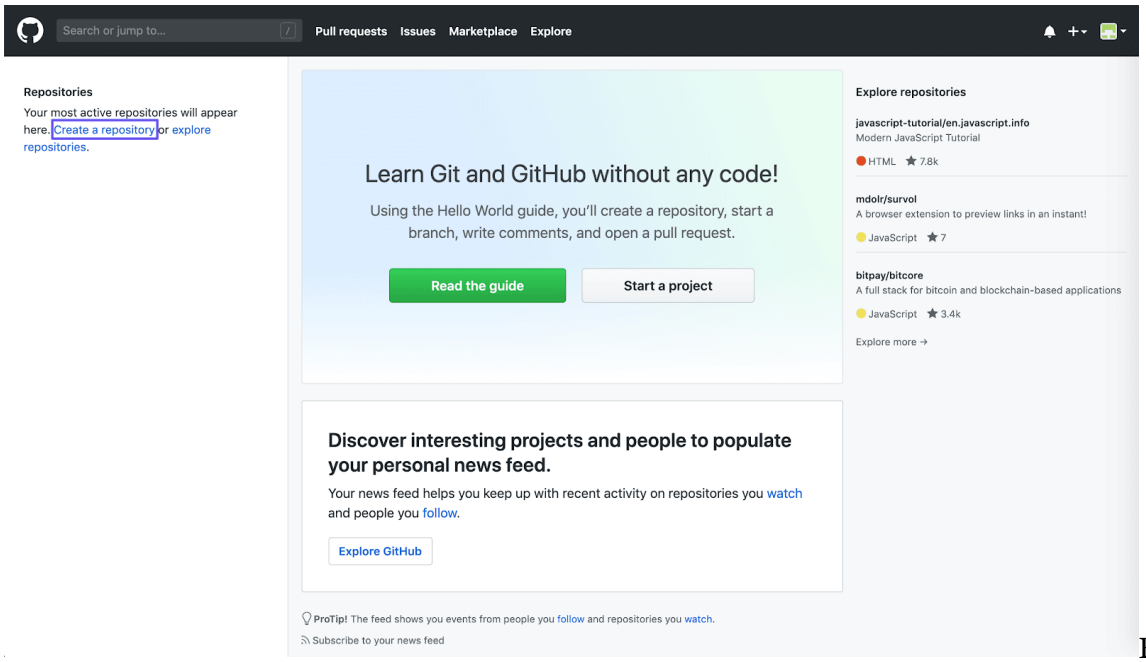
Opciones de precios de la cuenta de GitHub

Una cuenta gratuita funciona bien para los nuevos desarrolladores que buscan perfeccionar sus habilidades. Un plan profesional se adapta mejor a los freelancers y a los codificadores avanzados, mientras que las agencias querrán invertir en un plan de equipo para acceder a más herramientas de gestión de proyectos y de comunicación.

Además, GitHub ofrece descuentos para organizaciones sin fines de lucro, educadores y estudiantes.

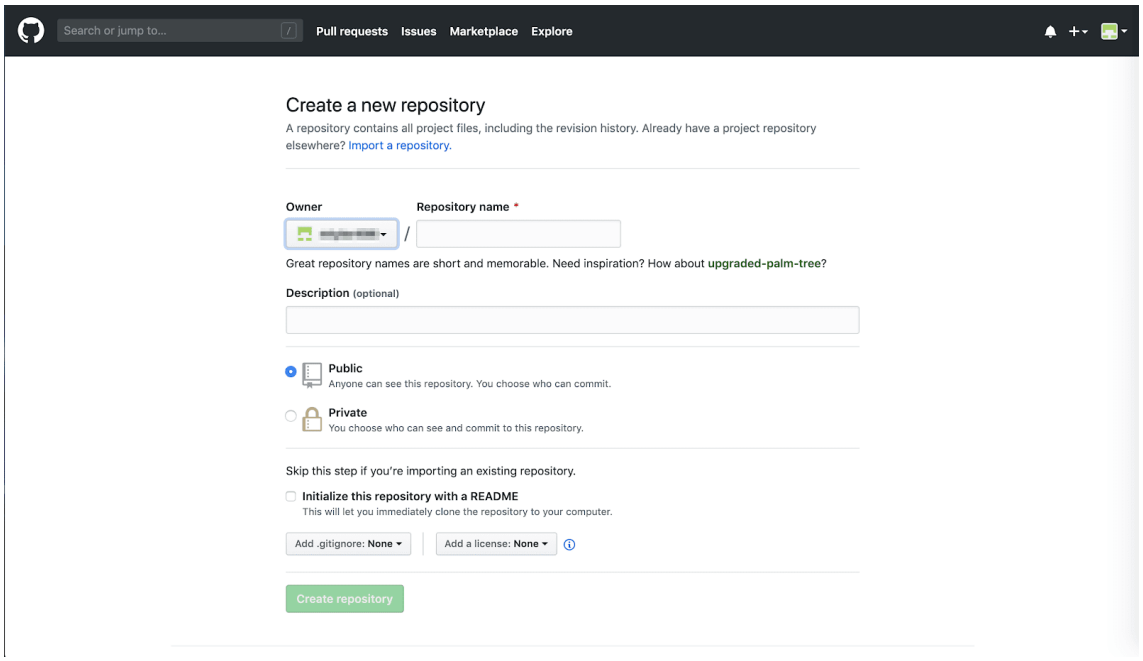
1.4.3 Paso 3: Añadir un Repositorio GitHub a su cuenta

Después de que haya creado y configurado su cuenta, necesitará crear un repositorio en GitHub donde pueda almacenar su proyecto cuando lo mueva desde git. Puede hacerlo haciendo clic en el enlace correspondiente de la barra lateral izquierda:



El enlace «Crear un repositorio» en GitHub

Entonces, tendrá que elegir un nombre para su repositorio:



Nombrar un nuevo repositorio GitHub

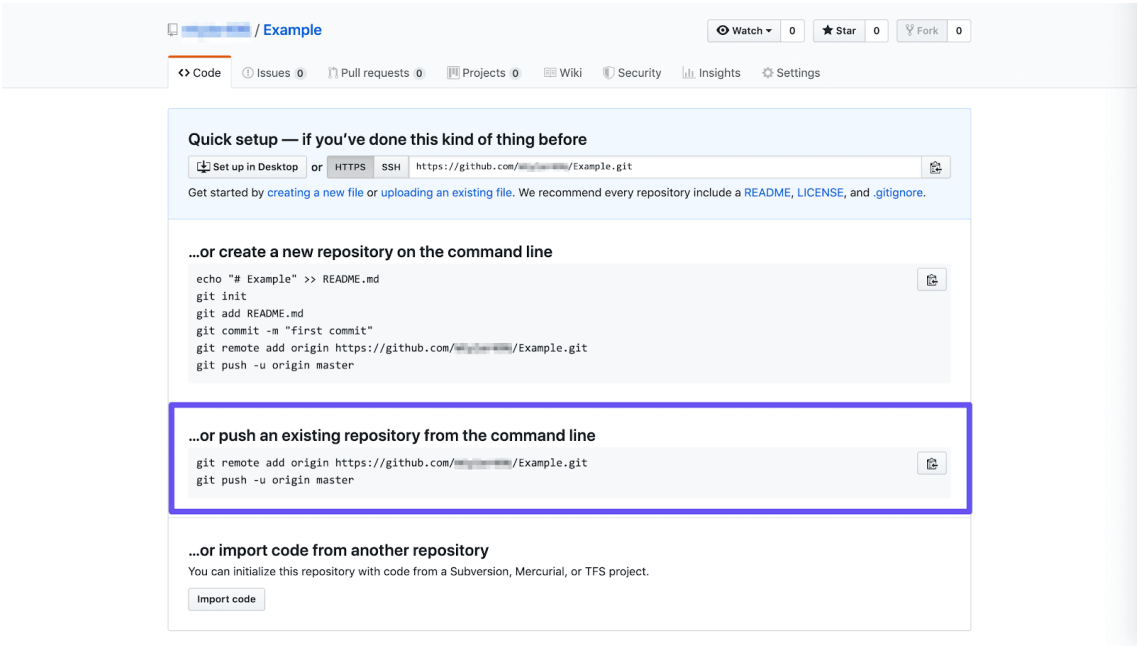
También puede decidir si quiere que el repositorio sea público, para que otros usuarios de GitHub puedan encontrarlo y contribuir a él, o si quiere mantenerlo privado.

De cualquier forma, ya que va a utilizar un proyecto existente, asegúrese de dejar la casilla **Inicializar este repositorio con una** casilla **LÉAME** desmarcada.

1.4.4 Paso 4: Empujar un Repositorio a GitHub

A continuación, tendrá la opción de añadir código a su repositorio de unas cuantas formas diferentes.

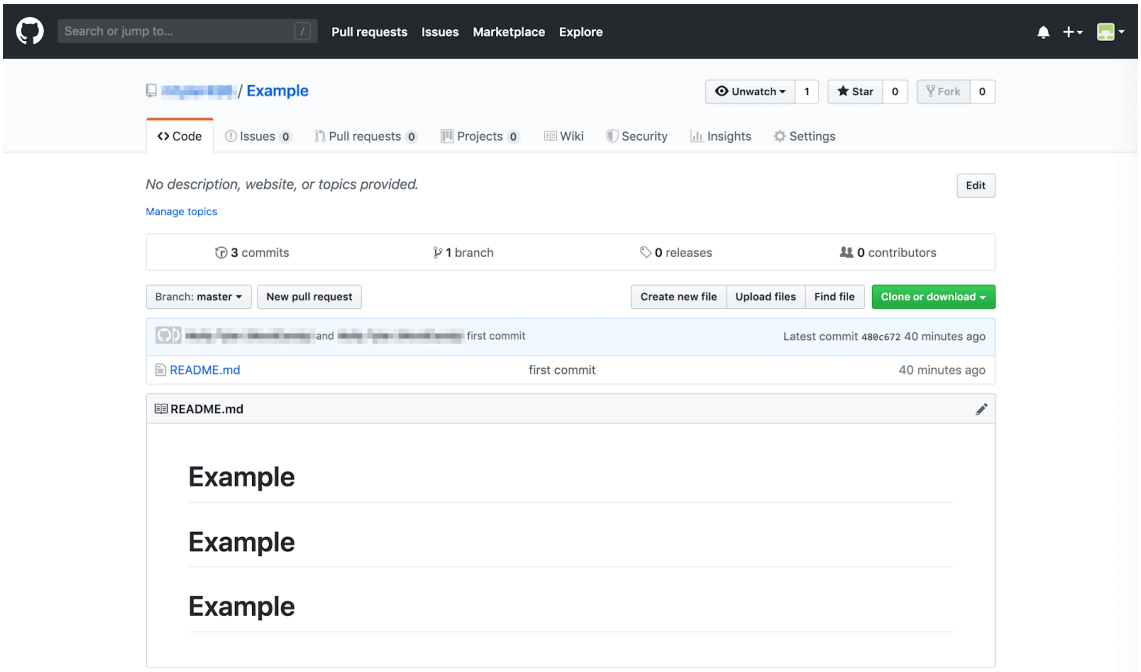
Como ya ha configurado su repositorio git, puede usar la opción de **empujar un repositorio existente desde la línea de comandos**:



Empujar un repositorio existente desde la opción de línea de comandos

El icono del portapapeles a la derecha le permite copiar los comandos que se enumeran aquí, de modo que puede pegarlos rápidamente en su interfaz de línea de comandos preferida para crear su repositorio GitHub.

Una vez hecho esto, actualice su página de GitHub:



Un nuevo repositorio GitHub listado en la página de la cuenta de usuario

Ahora debería poder ver su repositorio en GitHub. A partir de ahí, puede empezar a hacer cambios en su proyecto en línea.

También es posible enviar y fusionar solicitudes de extracción, y utilizar cualquier otra herramienta incluida en su plan.

1.4.5 Paso 5: Retirar los Cambios a git

Aunque puede ver todos los cambios que usted y otros han hecho en su proyecto en GitHub, la plataforma no tiene acceso directo a los archivos de su ordenador. Para mantener su proyecto actualizado en su computadora, necesitará hacer sus ediciones a través de git.

Para ello, simplemente introduzca git pull origin master en su interfaz de línea de comandos. Esto debería actualizar sus archivos para que todo esté sincronizado en todas las iteraciones de su proyecto.

Formas de trabajar en GitHub

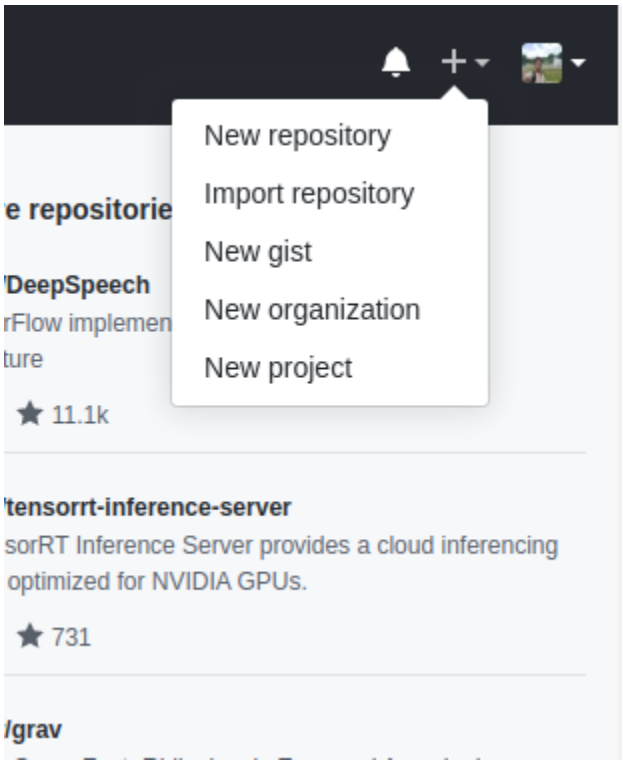
2 Trabajando con proyectos GitHub

Trabajaremos con proyectos GitHub en dos maneras.

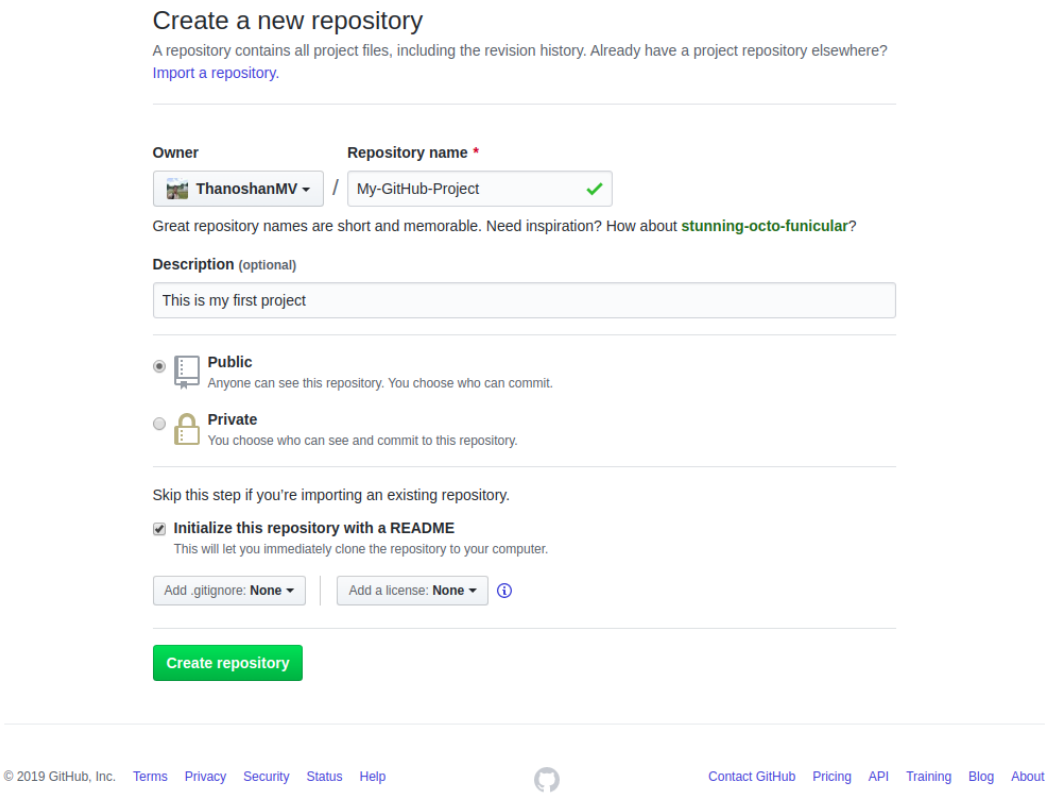
2.1.1 Tipo 1: Crear el repositorio, clonarlo en tu PC y trabajar en él. (Recomendado)

El tipo 1 involucra la creación de un repositorio totalmente nuevo en GitHub, clonarlo en nuestra computadora, trabajar en nuestro proyecto y enviarlo de regreso.

Crea un nuevo repositorio haciendo clic en el botón de "Nuevo repositorio" en la página web de GitHub.



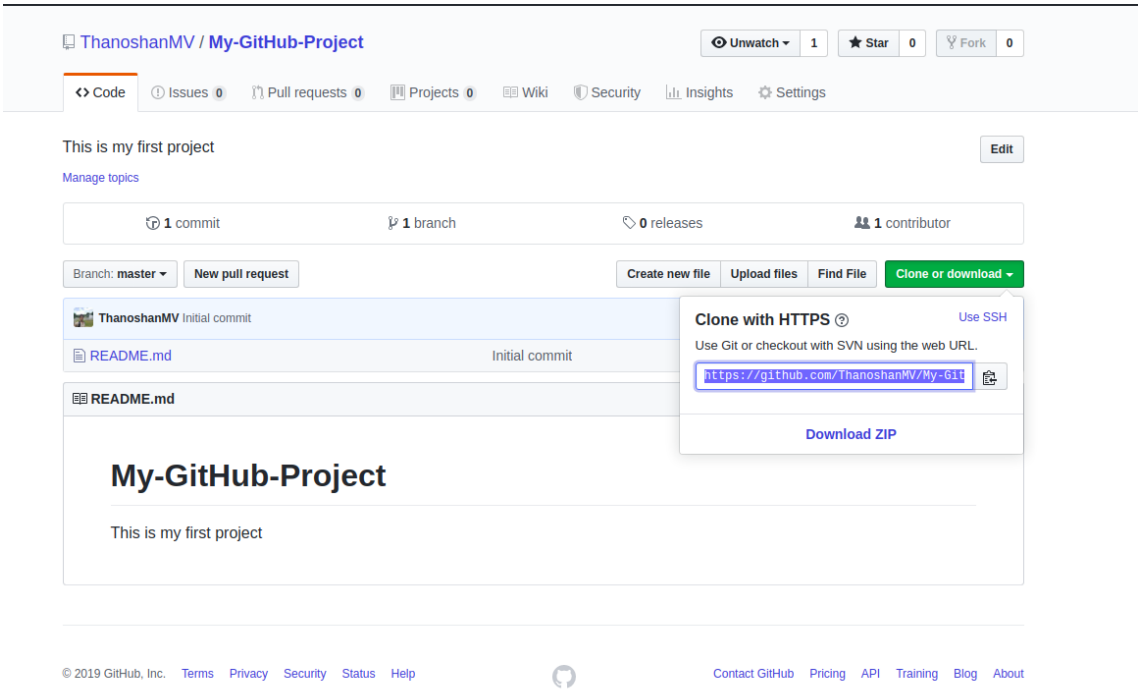
Elije un nombre para tu primer repositorio, agrega una pequeña descripción, marca la opción "Inicializar este repositorio con un README", y haz clic en el botón "Crear Repositorio".



Bien hecho! Tu primer repositorio de GitHub fue creado.

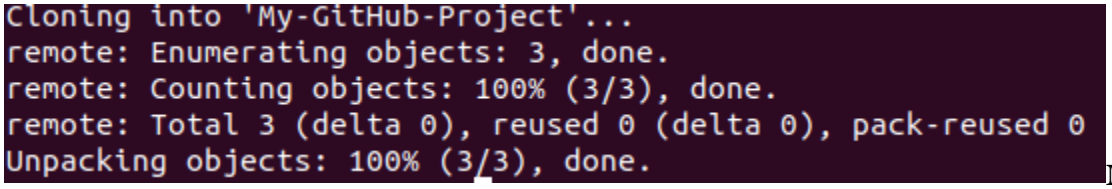
Tu primera misión es obtener una copia del repositorio en tu computadora. Para hacer eso, necesitas "clonar" el repositorio en tu computadora.

Clonar un repositorio significa que estás tomando un repositorio que está en el servidor y lo estás clonando a tu computadora – es lo mismo que descargarlo. En la página del repositorio, necesitas obtener la dirección "HTTPS".



Una vez que tienes la dirección del repositorio, necesitas utilizar tu terminal. Usa el siguiente comando en tu terminal. Cuando estés listo puedes ingresar esto:

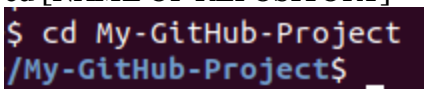
git clone [DIRECCION HTTPS]
Este comando realizará una copia local del repositorio alojado en la dirección dada.



Mensaje de salida del comando "git clone"

Ahora, tu repositorio está en tu computadora. Necesitas moverte en él con el siguiente comando.

cd [NAME OF REPOSITORY]



Como puedes ver en la imagen de arriba, el nombre de mi repositorio es "My-GitHub-Project" y este comando me hizo ir al directorio específico.

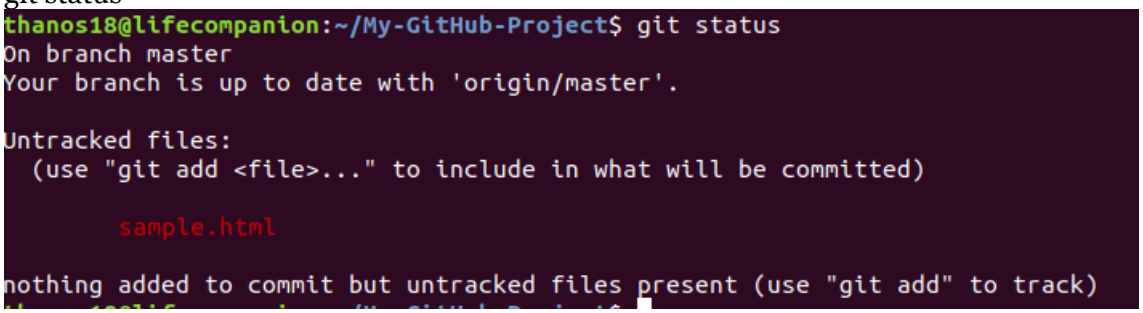
Ahora, en ese folder podemos crear archivos, trabajar en ellos y guardarlos localmente. Para guardarlos en un lugar remoto — como GitHub – tenemos que hacer un proceso llamado "commit". Para hacer esto, regresa a tu terminal. Si la cerraste, como dije anteriormente, usa el comando 'cd'.

cd [NAME OF REPOSITORY]

Ahora, en la terminal, estás en el directorio de tu repositorio. Hay 4 pasos en un commit: 'status', 'add', 'commit' y 'push'. Todos los siguientes pasos deben ejecutarse dentro de tu proyecto. Repasemos uno por uno.

1. “status”: La primer cosa que necesitas hacer es revisar los archivos que has modificado. Para hacer esto, puedes escribir el siguiente comando para hacer aparecer una lista de cambios.

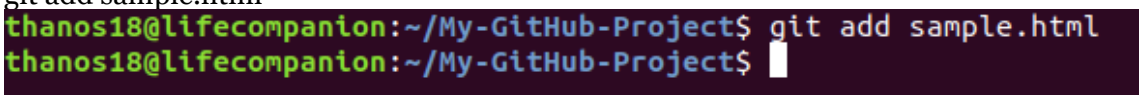
git status



2. “add”: Con la ayuda de la lista de cambios, puedes agregar todos los archivos que quieras cargar con el siguiente comando,

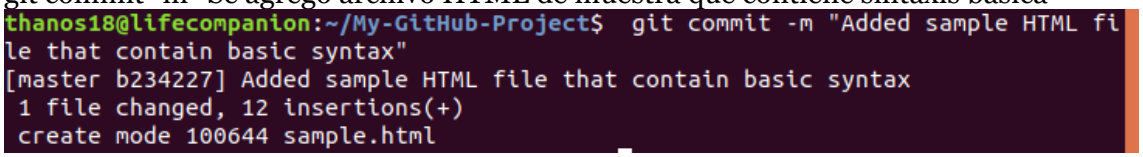
git add [NOMBRE DE ARCHIVO] [NOMBRE DE ARCHIVO] [...]
En nuestro caso, agregaremos un archivo HTML simple.

git add sample.html



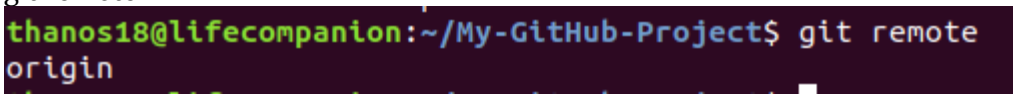
3. “commit”: Ahora que hemos agregado los archivos de nuestra elección, necesitamos escribir un mensaje para explicar lo que hemos hecho. Este mensaje puede ser útil después si queremos revisar el historial de cambios. Aquí hay un ejemplo de lo que podemos poner en nuestro caso.

git commit -m "Se agregó archivo HTML de muestra que contiene síntaxis basica"



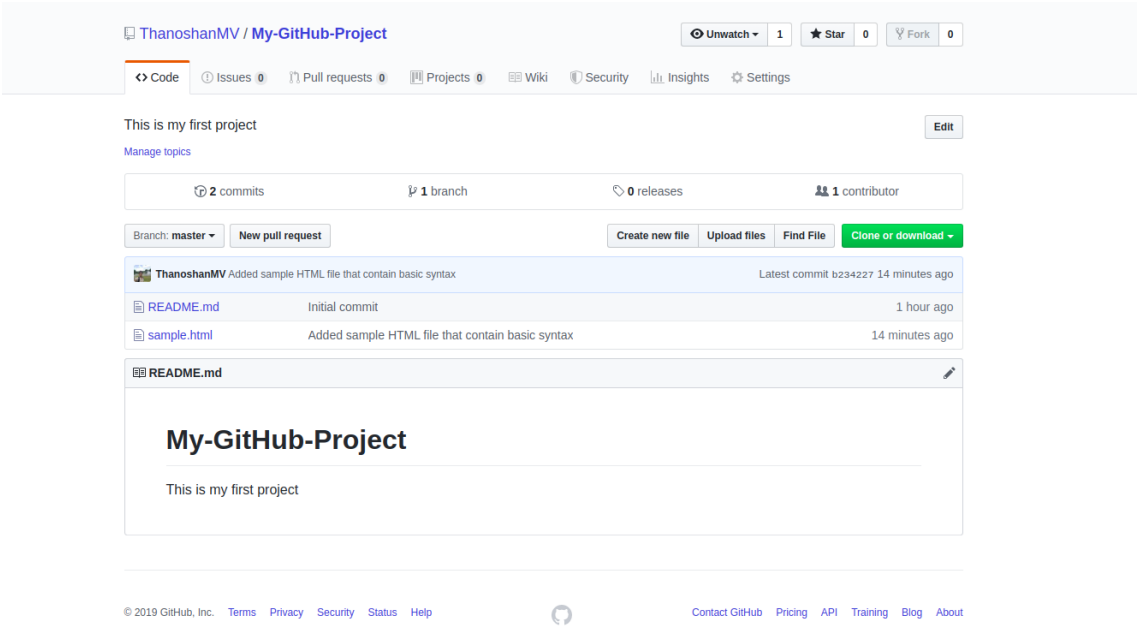
4. “push”: Ahora podemos poner nuestro trabajo en GitHub. Para hacer eso, necesitamos 'enviar' nuestros archivos a Remote. Remote es una instancia duplicada de nuestro repositorio que vive en algún otro lugar en un servidor remoto. Para hacer esto, debemos saber el nombre del Remote (En general, Remote es nombrado origen). Para encontrar ese nombre, escribe el siguiente comando.

git remote



Como puedes ver en la imagen, dice que el nombre de nuestro remote es origin. Ahora podemos 'enviar' de manera segura nuestro trabajo con el siguiente comando.

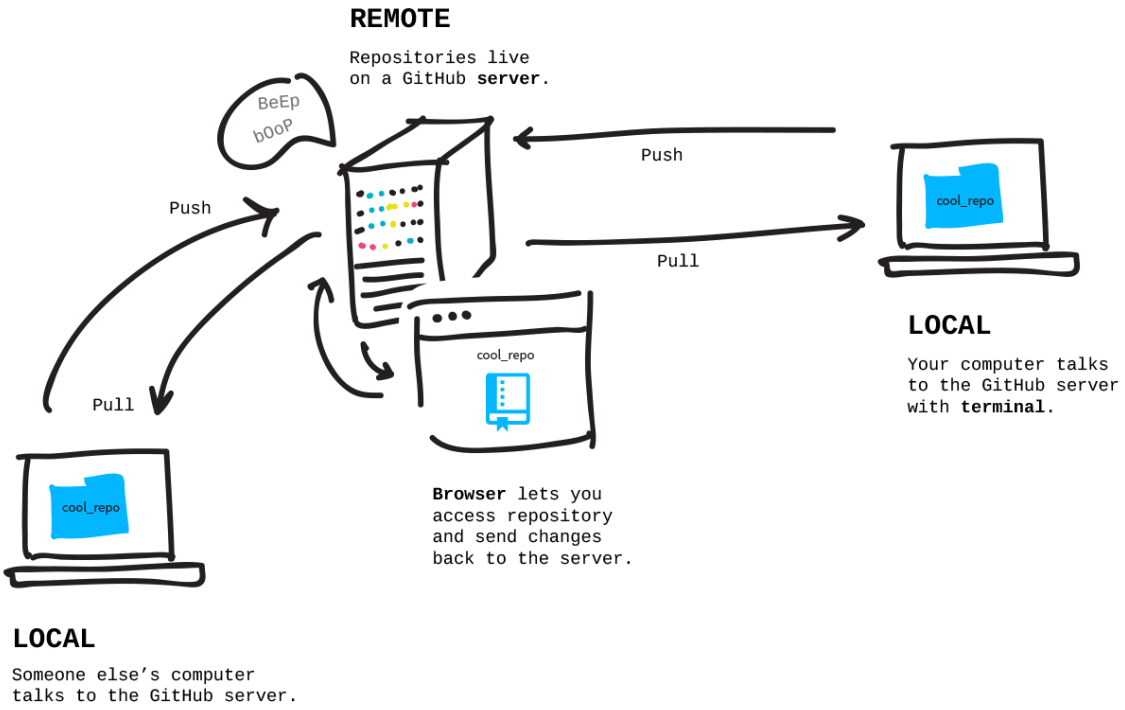
git push origin master
Ahora, si vamos a nuestro repositorio en la página web de GitHub, podemos ver el archivo sample.html que hemos enviado a Remote — GitHub!



NOTA: A veces cuando estás usando comandos Git en la terminal, te puede llevar al editor de texto VIM (un editor de texto basado en CLI). Para deshacerte de esto, tienes que escribir

:q
y presionar ENTER.

```
type  :q<Enter>
```



Describes how pull & push work

Pulling es el acto de recibir de GitHub.

Pushing es el acto de enviar a GitHub.

2.1.2 *Tipo 2: Trabajar en tu proyecto localmente y después crear el repositorio en GitHub y enviarlo a remote.*

El tipo 2 permite hacer un nuevo repositorio de un folder existente en nuestra computadora y enviarlo a GitHub. En muchos casos es posible que ya hayas creado algo en tu computadora y repentinamente quieras convertirlo en un repositorio en GitHub.

Voy a explicarte esto con un proyecto web de formulario de encuesta que hice antes y que no fue agregado a GitHub.

Como ya mencioné, cuando se ejecuta cualquier comando Git, tenemos que asegurarnos que estamos en el directorio correcto en la terminal.

```
thanos18@lifecompanion:~/Code-School/WebDev/Survey Form Project$
```

De manera predeterminada, cualquier directorio en nuestra computadora no es un repositorio Git – pero lo podemos convertir en un repositorio Git ejecutando el siguiente comando en la terminal.

git init

```
thanos18@lifecompanion:~/Code-School/WebDev/Survey Form Project$ git init
Initialized empty Git repository in /home/thanos18/Code-School/WebDev/Survey For
m Project/.git/
```

Después de convertir nuestro directorio en un repositorio Git, la primera cosa que necesitamos hacer es revisar los archivos que tenemos utilizando el siguiente comando.

git status

```
thanos18@lifecompanion:~/Code-School/WebDev/Survey Form Project$ git status
On branch master

No commits yet

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

        index.html
        style.css

nothing added to commit but untracked files present (use "git add" to track)
```

Así que hay dos archivos en ese directorio que necesitamos "agregar" a nuestro Repo.

git add [FILENAME] [FILENAME] [...]
NOTA: Para "agregar" todos los archivos en nuestro repositorio podemos usar el siguiente comando.

git add .
Después de que el área de preparación (el proceso de Add) está completo, podemos revisar si nuestros archivos fueron agregados satisfactoriamente o no ejecutando git status
Si esos archivos en particular están en verde como la imagen de abajo, haz hecho tu trabajo!

```
thanos18@lifecompanion:~/Code-School/WebDev/Survey Form Project$ git add .
thanos18@lifecompanion:~/Code-School/WebDev/Survey Form Project$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   index.html
        new file:   style.css
```

Después tenemos que hacer "commit" con una descripción.

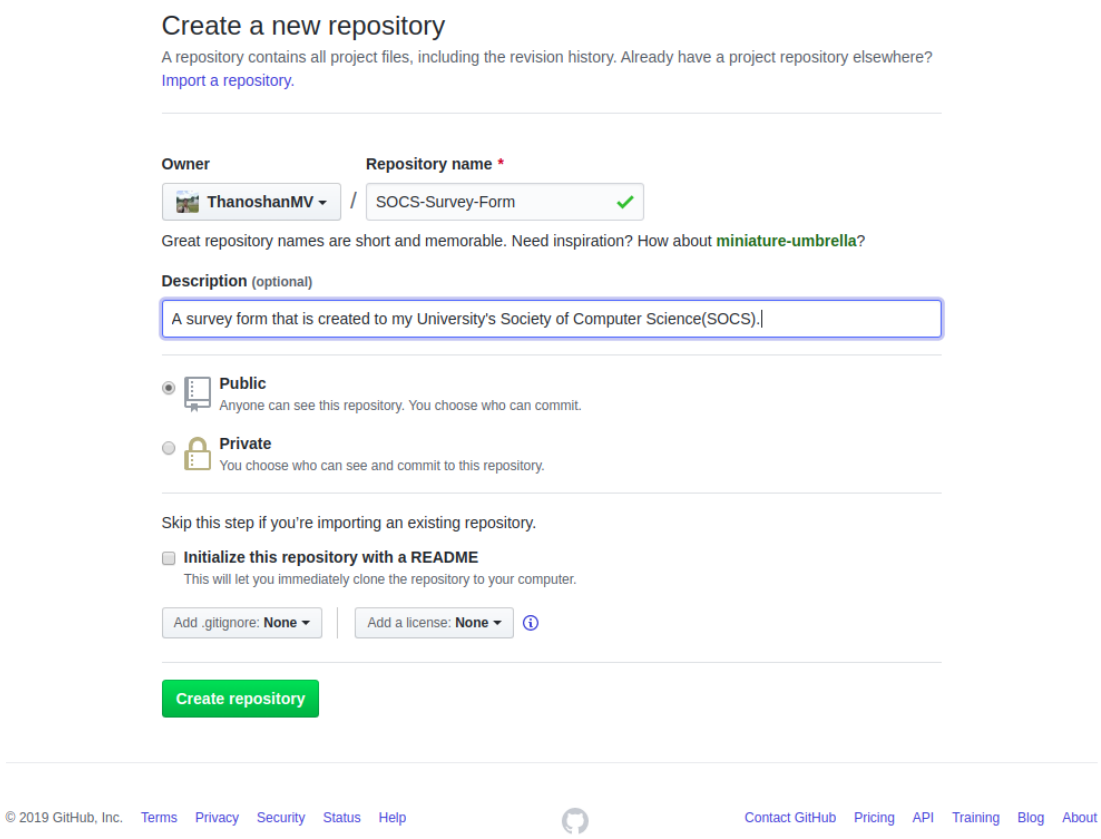
git commit -m "Agregando un formulario web de encuesta"

```
thanos18@lifecompanion:~/Code-School/WebDev/Survey Form Project$  git commit -m
"Adding web Survey form"
[master (root-commit) aa0a70a] Adding web Survey form
 2 files changed, 306 insertions(+)
 create mode 100644 index.html
 create mode 100644 style.css
```

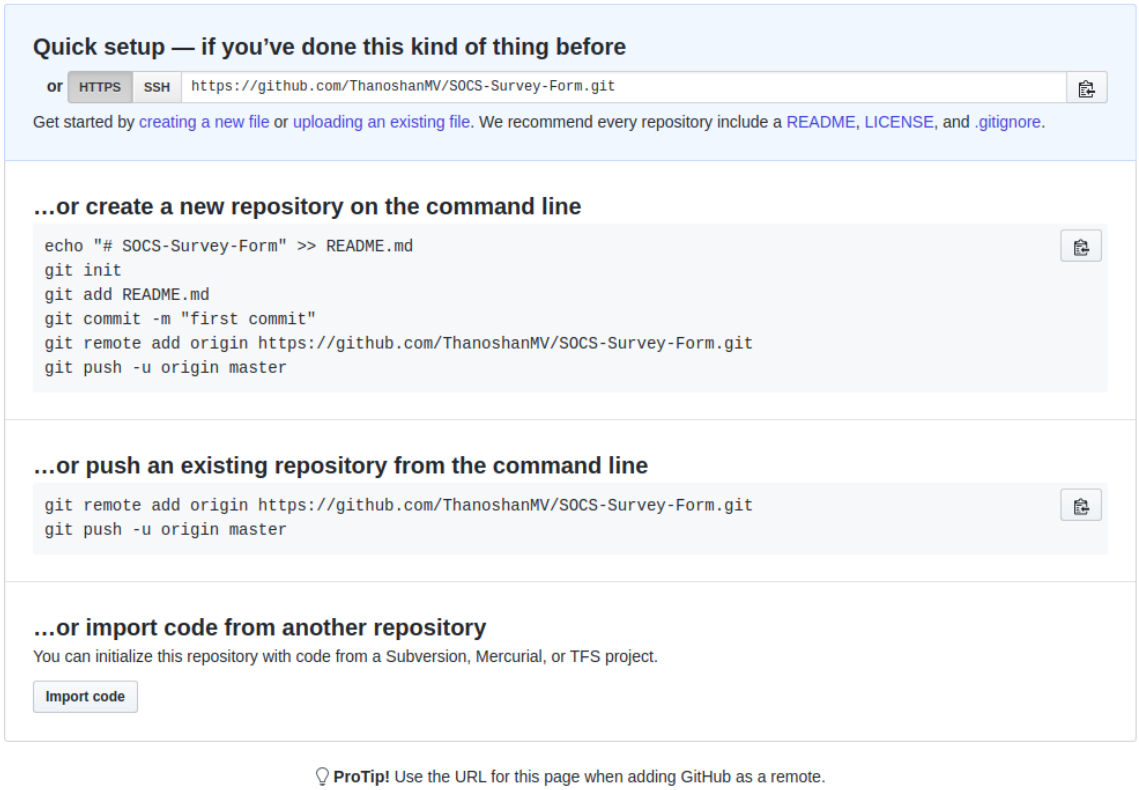
Si mi repositorio inició en GitHub y lo descargo a mi computadora, un remote ya estará unido a él (Tipo 1). Pero si estoy iniciando mi repositorio en mi computadora, no tiene un remote asociado con él, así que necesito agregarlo (Tipo 2).

Así que para agregar ese remote, primero tenemos que ir a GitHub. Crear un nuevo repositorio y nombrarlo como queramos almacenarlo en GitHub. Después hacer clic en el botón "Crear repositorio".

NOTA: En el tipo 2, por favor no inicialices el repositorio con un archivo README cuando crees un nuevo repositorio en la página web de GitHub.



Después de hacer clic en el botón "Crear repositorio" encontrarás la siguiente imagen como una página web.



Copia la dirección HTTPS. Ahora crearemos el remote para nuestro repositorio.

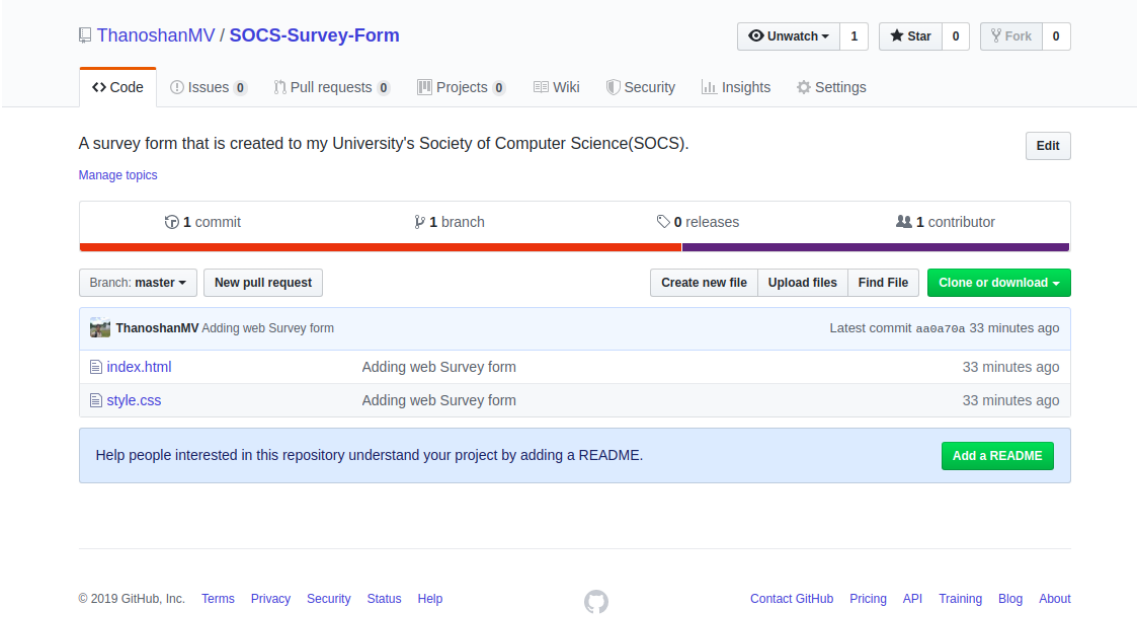
git remote add origin [DIRECCION HTTPS]
Después de ejecutar este comando, podemos revisar si hemos agregado el remote satisfactoriamente o no con el siguiente comando

git remote
Y si genera "origin", haz agregado el remote a tu proyecto.

NOTA: Solo recuerda que podemos indicar cualquier nombre para el remote cambiando el nombre "origin". Por ejemplo:

git remote add [NOMBRE REMOTE] [DIRECCION HTTPS]
Ahora, podemos enviar nuestro proyecto a GitHub sin ningún problema!

git push origin master
Después de completar estos pasos uno por uno, si vas GitHub puedes encontrar tu repositorio con los archivos!



Apéndice

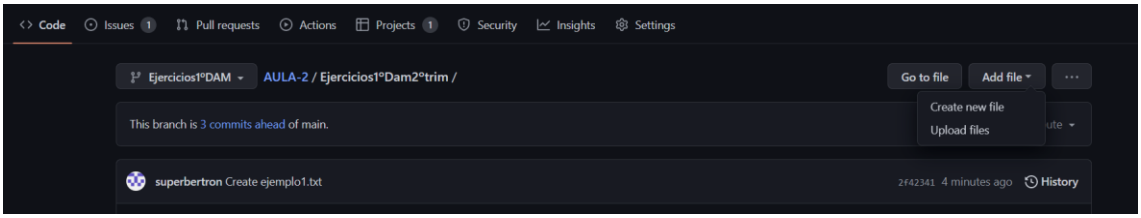
Qué son las Ramas (Branches) ?

En Git las Ramas son espacios o entornos independientes para que un Desarrollador sea Back-end, Front-end, Tester, etc. pueda usar y así trabajar sobre un mismo Proyecto sin chancar o borrar el conjunto de archivos originales del proyecto, dándonos flexibilidad para desarrollar nuestro proyecto de manera mas organizada.

Cada Rama o Branch que creamos con Git, esta compuesta por un archivo compuesto de 40 caracteres de una suma de control SHA-1.

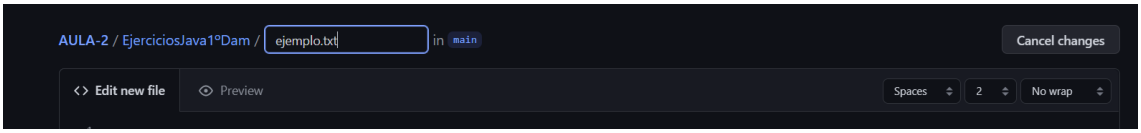
En github se crea de la siguiente manera

1º pulsamos en Add file y despues en créate new file



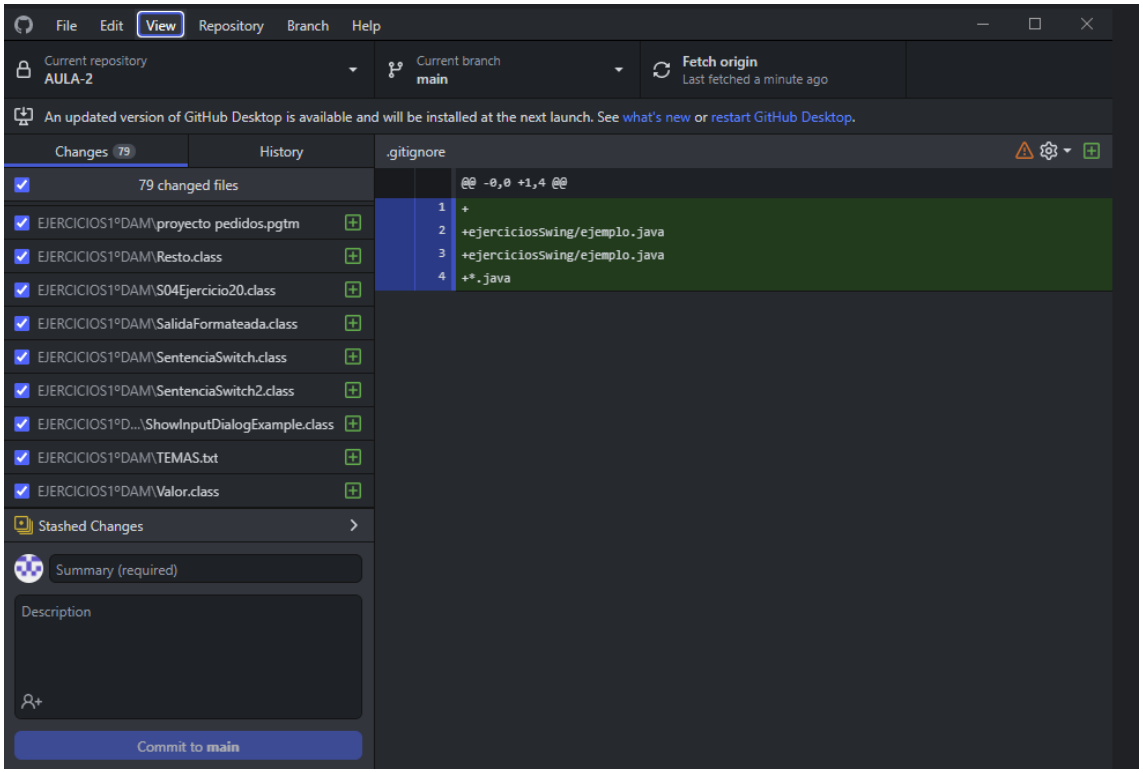
Si queremos crear un Branch tendremos que escribir un nombre y despues un /

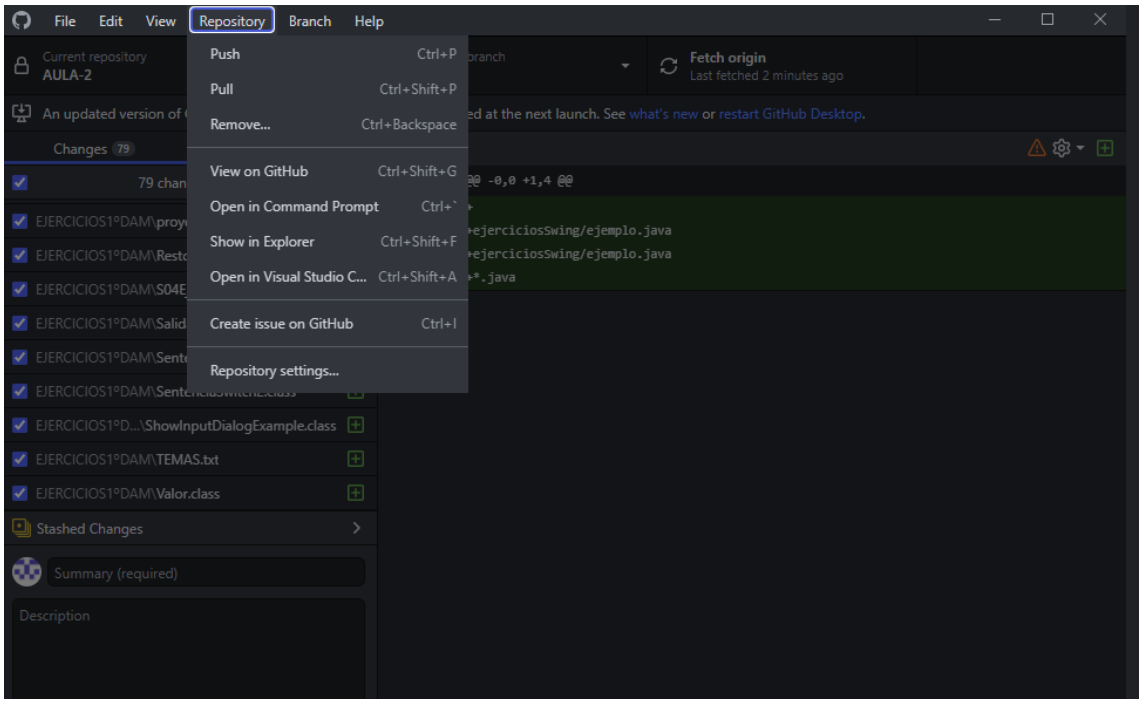
Luego el nombre de un fichero



Si usamos Git

git pull es uno de los muchos comandos que se encargan de "sincronizar" el contenido remoto. El comando git remote se utiliza para especificar los extremos remotos sobre los que operarán los comandos de sincronización. El comando git push sirve para cargar contenido en un repositorio remoto





Insertar título tema o módulo
Insertar otro título

3 Escriba el título del Capítulo 1