

Clase 6: GitHub

Índice

- Git Hub: Concepto y Funcionalidad.
- Pasos para registrarse en Git Hub
- Conectando nuestro repositorio local a GitHub
- Como subir archivos.
- Como bajar archivos
- Ejemplo de conflictos que podemos llegar a tener.
- Ramas en Git. Concepto y funcionalidad.
- Creación de ramas.
- Comandos básicos.
- Documentación y glosario



Introducción

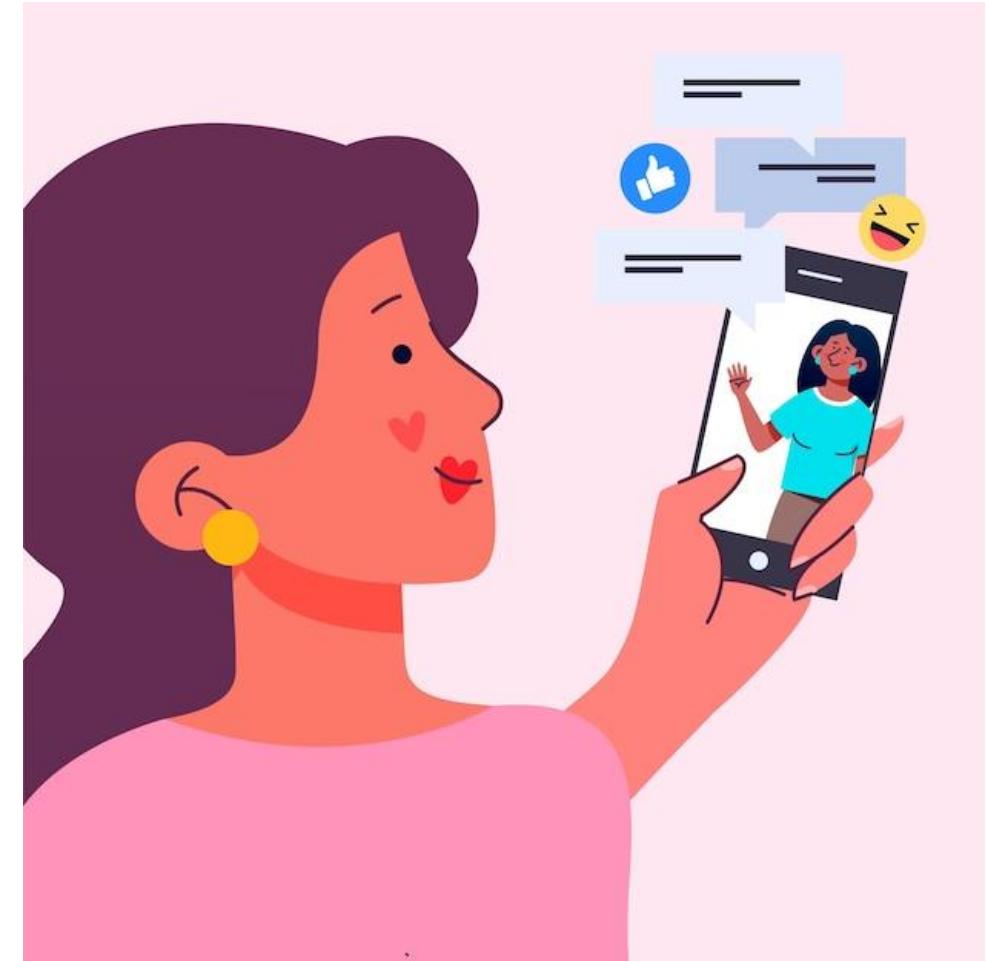
Recordemos viejos tiempos...

Hace años, las personas teníamos álbumes fotográficos en donde se guardaban todos esos **lindos recuerdos** y cuando los queríamos compartirlos con alguien mas, las personas tenían que reunirse en nuestra casa y ver los álbumes (acompañado de unos buenos mates)



Recordemos viejos tiempos...

Esto hoy en día ya no es así, mas allá de haber cambiado esta costumbre, cuando queremos ver las fotos de alguien basta con entrar a su perfil de **Instagram o facebook** y ponerte al día con lo que le a sucedido en la vida.



Recordemos viejos tiempos...

Todo esto se debe gracias al almacenamiento en la nube.

Hoy aprenderemos como hacer esto mismo con los archivos de nuestros proyectos!



Git

Hablando de nube...

Cuando hablamos de almacenamiento, seguramente piensan en opciones como Google drive o quizás Dropbox.

Estos funcionan bastante bien para la mayoría de las personas, pero cuando estamos en el ambiente de la programación, necesitamos de sistemas mucho mas estables, funcionales y rapidos.

Y es ahí, en ese momento donde **GitHub** toma protagonismo.



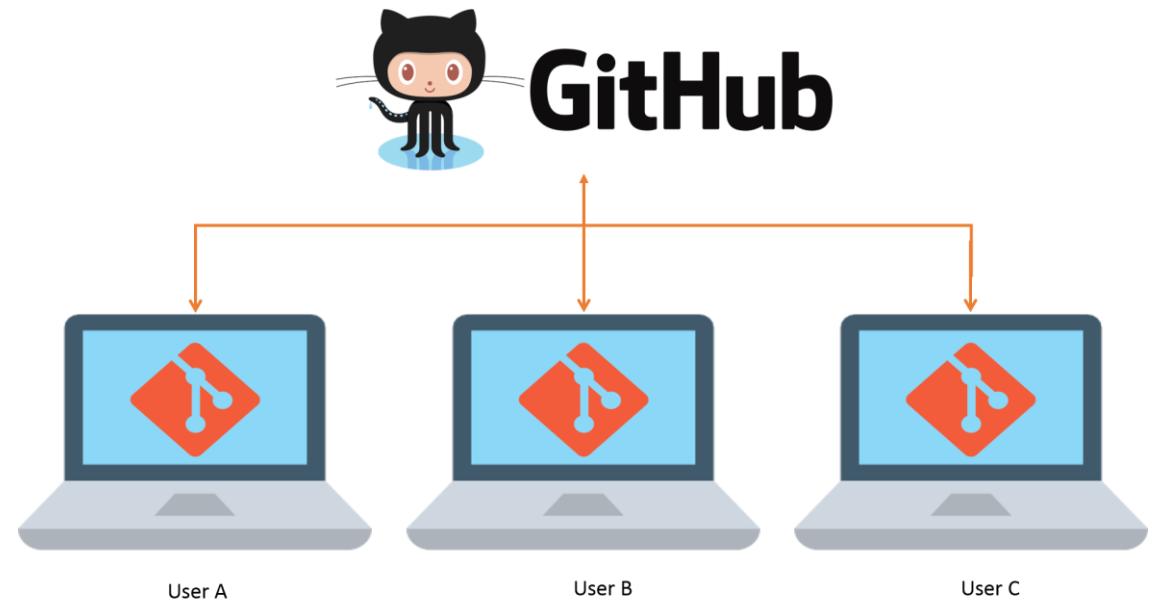
GitHub Conceptos

GitHub

¿Qué es GitHub?

¿Es lo mismo que Git?

No! GitHub es un **lugar en la nube**, un lugar donde puedes alojar tus archivos de proyectos de programación, de manera **gratuita** sin mas trabajo que crear una cuenta y disfrutar sus beneficios!



Diferencia: Git y GitHub



Herramienta de control de versiones distribuida

Herramienta de código abierto que los desarrolladores instalan localmente para gestionar el código fuente



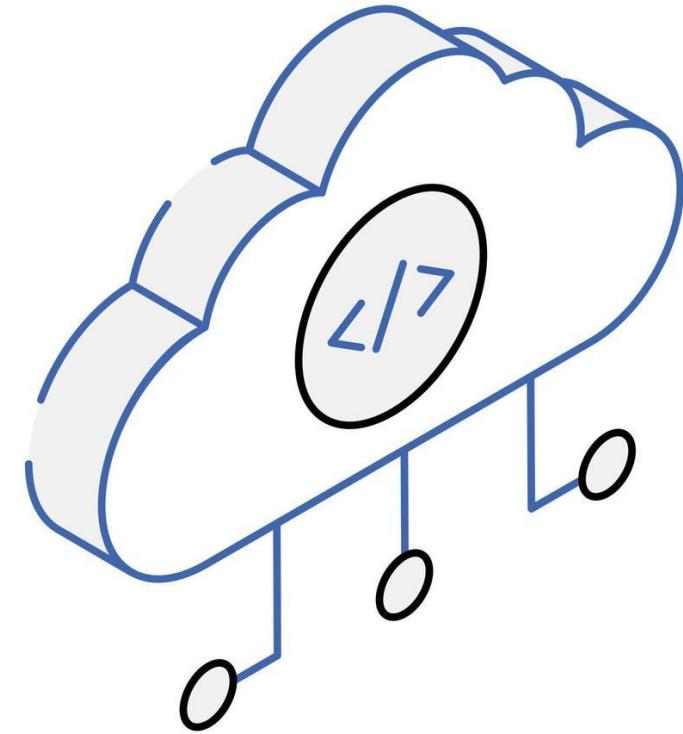
Plataforma basada en la nube

Servicio en línea al que los desarrolladores que utilizan Git pueden conectarse y cargar o descargar recursos

GitHub: Cuenta y Repo

Una vez que tengas tu cuenta en github lista, podrás crear lo que llamamos “**Repositorio**”.

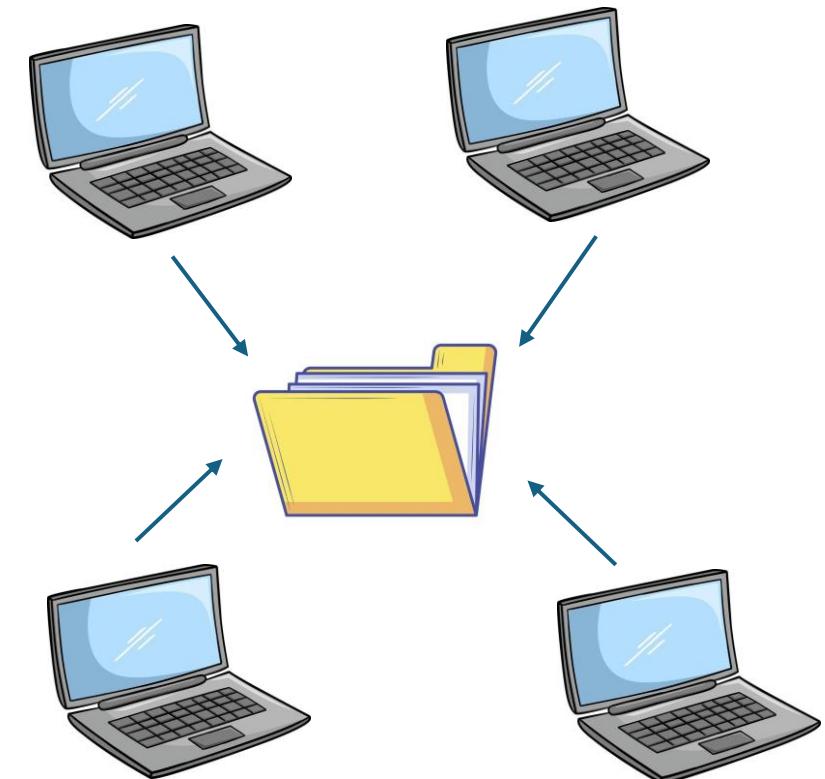
Un repositorio es un **lugar** donde se irán almacenando los archivos de tu proyecto y a través del cual podrás hacer seguimiento de los mismos.



GitHub: Cuenta y Repo

Debes tener en cuenta que por cada **repositorio corresponde un solo proyecto**.

Pero no te preocupes que en Github puedes tener la cantidad de repos que quieras, ya que los mismos son completamente **gratis**.



1 proyecto = 1 repositorio

GitHub: Cuenta y Repo

De igual manera, los repositorios que viven en GitHub los llamamos **repositorios remotos** ya que están en la nube.

En paralelo, cada persona de tu equipo va a tener una copia de ese repositorio en su computadora, a los cuales llamamos **repositorios locales**

repositorios remotos



repositorios locales

GitHub: Cuenta y Repo

Para que Git funcione bien, es necesario crear un **vinculo entre local y remoto** para que puedas mantener los backup de tus archivos locales conectados con ese lugar en la nube.

Con todo esto en mente, vamos a ver el **paso a paso en GitHub para crear nuestro primer repositorio**.

repositorios remotos



repositorios locales

GitHub

Creación de cuenta

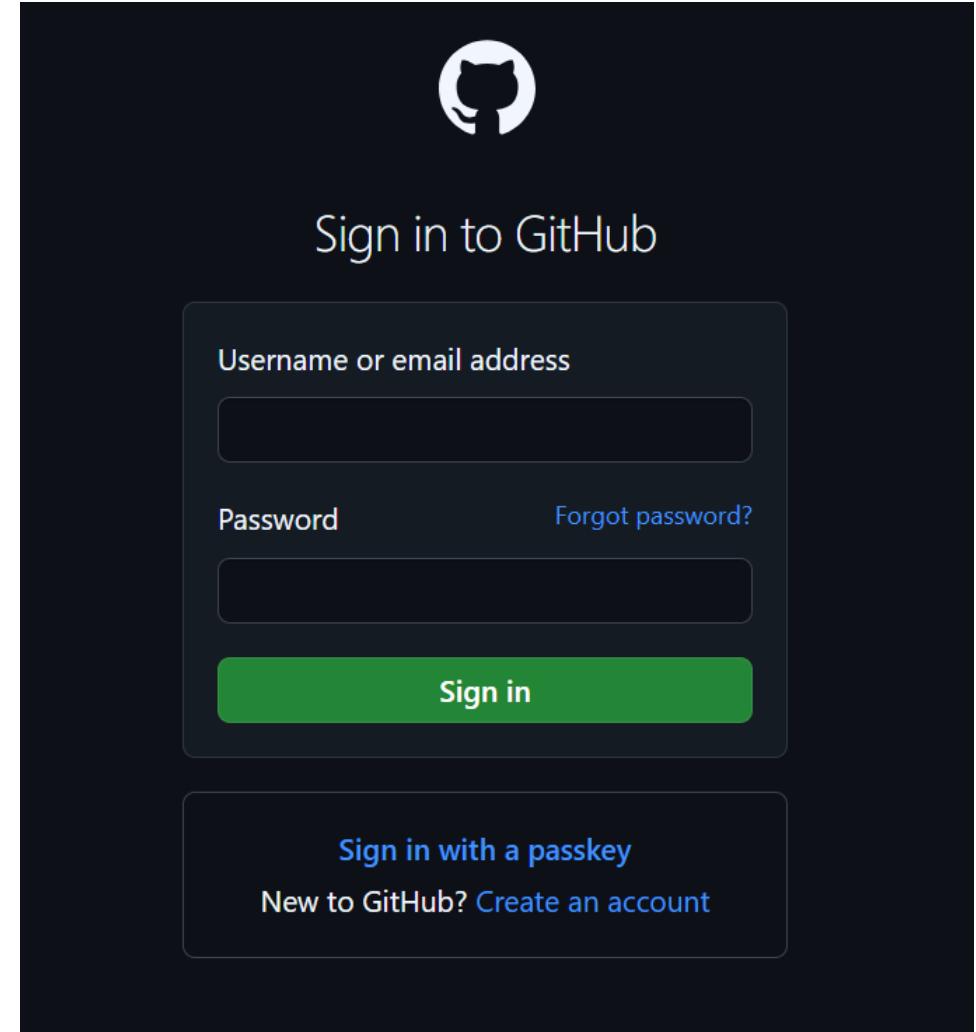
¿Cómo me registro en GitHub?

En tan solo dos simples pasos:

1. Ingresa a la URL:

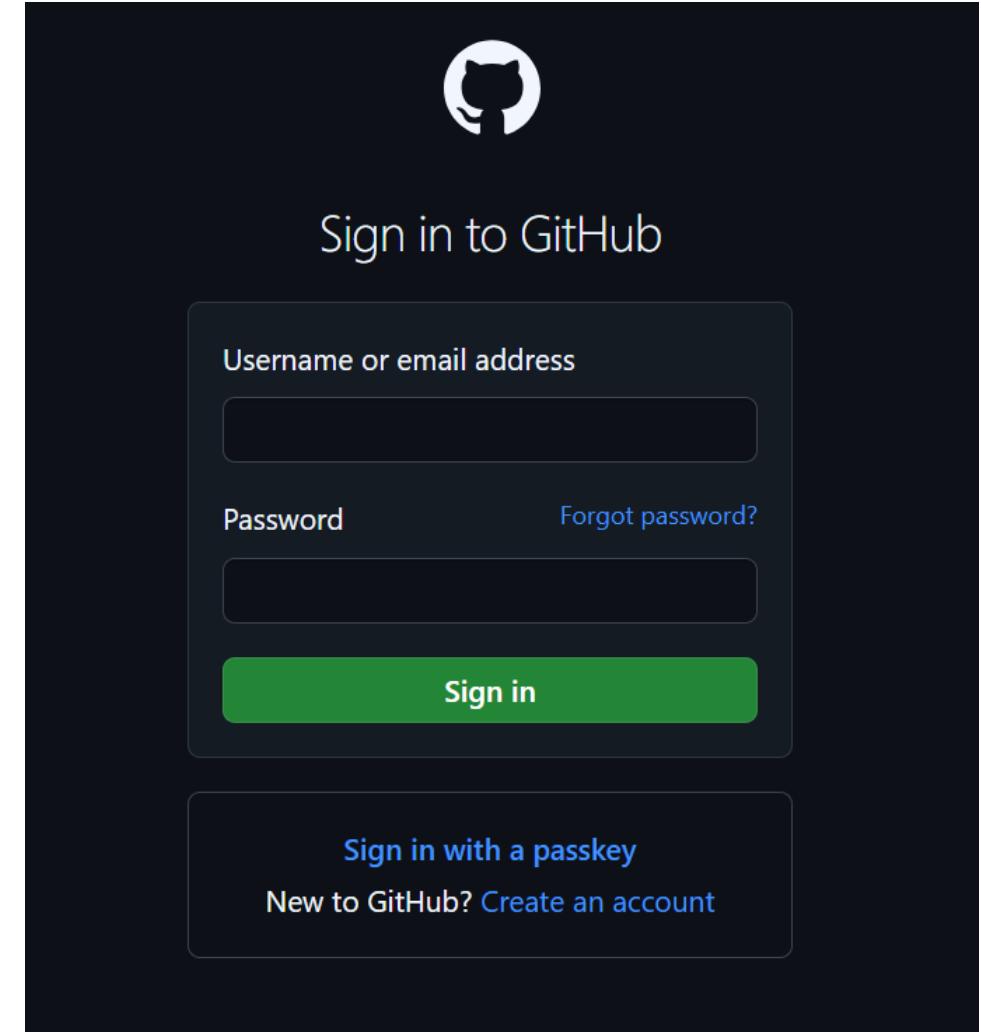
<https://github.com/login>

2. Completa el formulario de registro y listo!



¿Cómo me registro en GitHub?

Recuerda guardar este usuario y contraseña porque son los que vas a necesitar para hacer la vinculación con el repositorio local.



GitHub

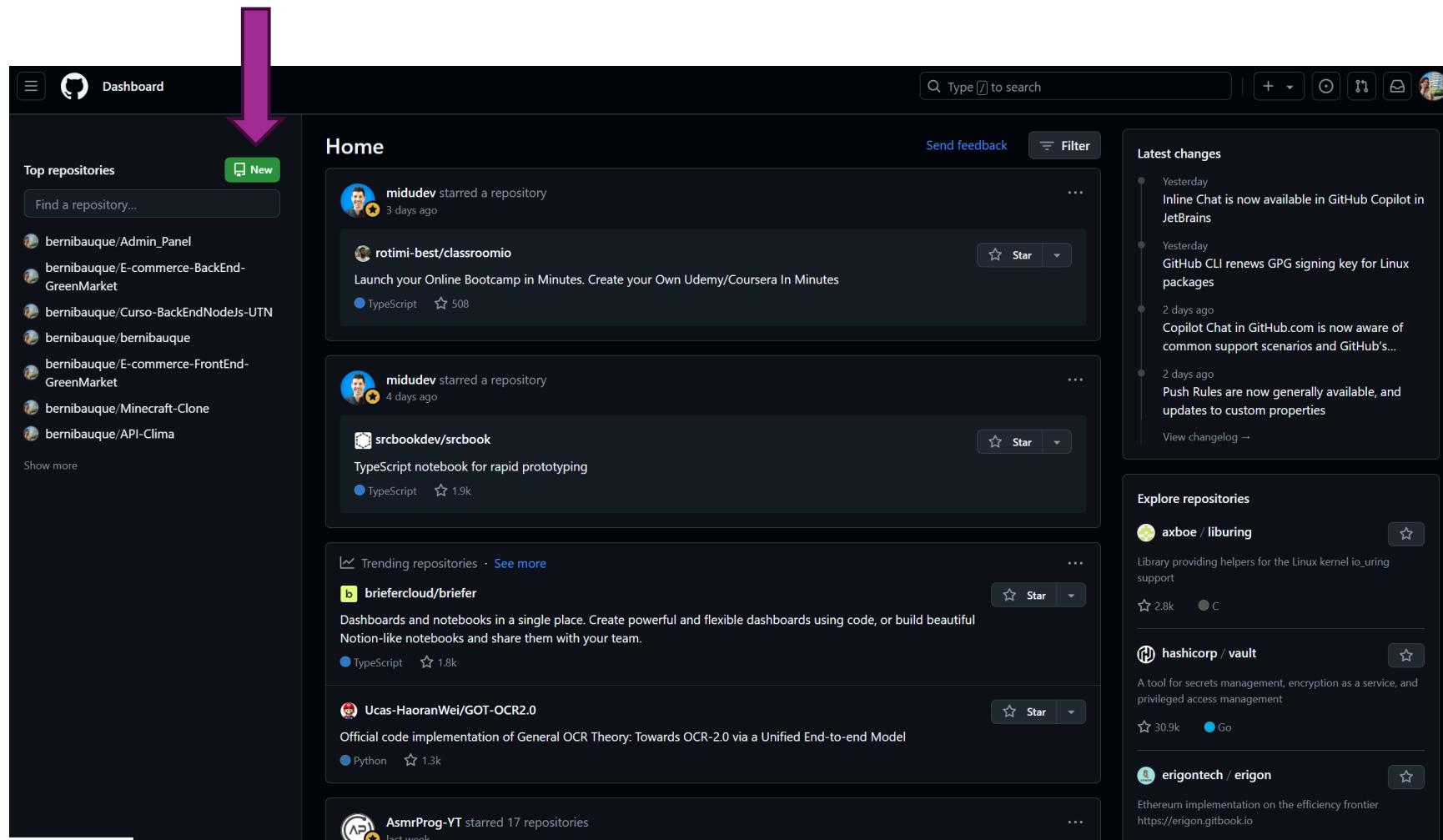
Creación de nuestro primer Repositorio remoto

GitHub

Creamos nuestro primer repositorio Remoto

Ahora, veamos cuáles son los pasos para crear nuestro primer repositorio remoto.

1. Una vez creada nuestra cuenta en GitHub hacemos click en **New**.





Creamos nuestro primer repositorio Remoto

2. Elegimos un nombre para nuestro repositorio.

En este caso yo elegí llamarlo “Mi_Proyecto” pero ustedes pueden poner un nombre que tenga que ver con el proyecto que están desarrollando.

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 *
bernibauque	/ Mi_Poryecto
<input checked="" type="checkbox"/> Mi_Poryecto is available.	



Creamos nuestro primer repositorio Remoto

3. Luego, agregamos una descripción que especifique de que trata mi proyecto.

La misma puede ser extensa, pero usualmente es algo corto, solo contando en pocas palabras de que trata el proyecto.

La descripción es algo opcional para realizar el repositorio pero es una buena práctica además que le da mejor vista para los visitantes que vean nuestros repositorios.

A screenshot of a dark-themed GitHub repository creation interface. At the top, there is a label "Description (optional)". Below it is a text input field containing the placeholder text "Aqui va la descripción del proyecto". The entire form is enclosed in a light blue rounded rectangle.

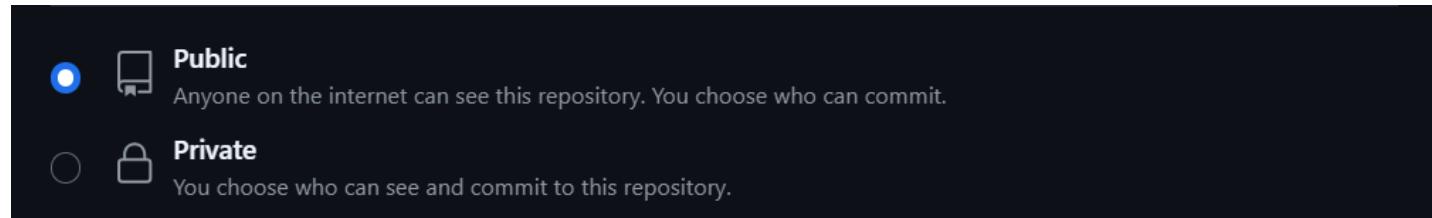
GitHub

Creamos nuestro primer repositorio Remoto

4. Después, hacemos click en la opción de “publico” ya que esta opción le dará una visibilidad total a cualquier persona que ingrese a nuestro GitHub.

Esto es bueno para que los recruiters que entren a nuestro perfil puedan ver todos los proyectos que desarrollamos.

Si por lo contrario en un futuro tienen proyectos que no pueden compartir, pueden usar la opción “private” para que solo ustedes puedan verlos.

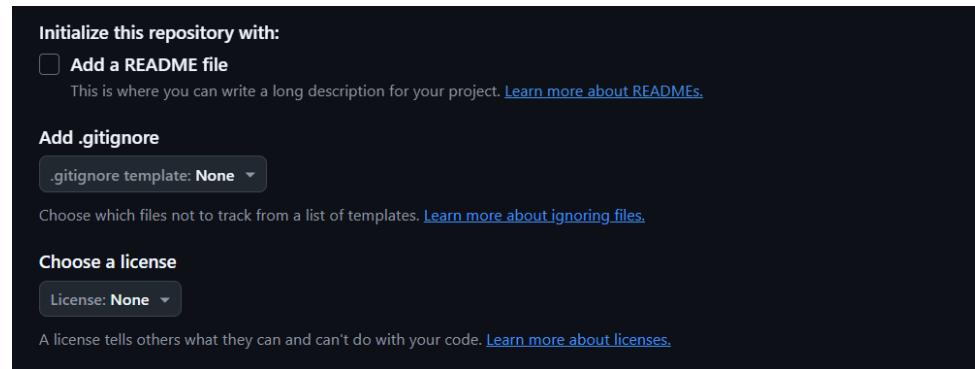




Creamos nuestro primer repositorio Remoto

5. Podemos elegir si queremos inicializar el repositorio con un readme, pero por lo general yo suelo aconsejarles no tildar esa opción.

Tampoco dar a la opción de Add .gitignore ni choose a license. Dejar ambas opciones en NONE.





Creamos nuestro primer repositorio Remoto

4. Por ultimo, damos click en **Create repository** y listo!

Ya tenemos nuestro repositorio.

Esta es la vista completa que deberían tener de su 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 * **Repository name ***

bernibauque / Mi_Proyecto
Mi_Proyecto is available.

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

Description (optional)
Aqui va la descripción del proyecto

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

GitHub

Creamos nuestro primer repositorio Remoto

Recordá que GitHub (en la nube) vamos a alojar una versión online y compartida de nuestros archivos.

Cada persona en nuestro equipo, a través de los **comandos** de git, se descargara los cambios y subirá las actualizaciones.

repositorios remotos



repositorios locales

GitHub

Tip Importante:

Las empresas de desarrollo, se fijan bastante en los perfiles de github de sus posibles contrataciones.

Así que les recomiendo que comiencen a crear un lindo perfil, que les ayudara y sumara un montón a la hora de buscar trabajo.



GitHub

Conectando nuestro primer Repositorio local

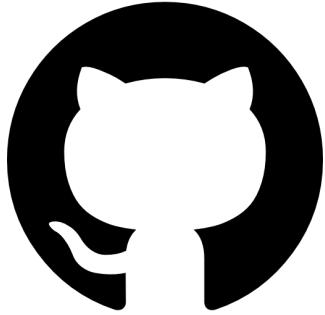
Conectando!

Llego el momento importante en nuestro flujo de trabajo! **Vamos a conectar el repositorio local con el repositorio remoto.**

Para hacer esto, vas a necesitar previamente haber creado un repositorio en GitHub.

Recuerda que este PDF es un PASO a PASO gigante! Así que créalo con tiempo y sin apuro!





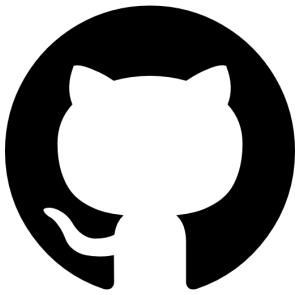
Conectando!

Cuando creas un repositorio en github, el mismo queda con una URL Única.

Es mas, esa URL tiene una **estructura similar a esta:**

```
https://github.com/mi-usuario/nombre-del-repo.git
```

En la misma, puedes ver que esta tu **nombre** y el **nombre que le diste al repositorio**



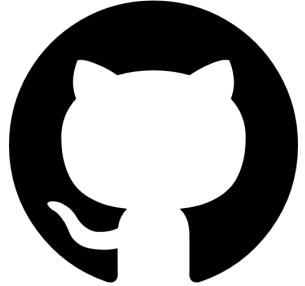
Conectando!

Con esta URL en el portapapeles, es decir **copiada**, vas a ir a la terminal y copiar el comando:

```
$ git remote add origin https://github.com/mi-usuario/nombre-del-repo.git
```

Debes pegar la url del repositorio de GitHub después de la palabra `origin` y apretar enter.

Si todo sale bien, la terminal no debería arrojarte ningún mensaje.



Conectando!

Pero para verificar que todo quedo perfecto, puedes ejecutar el comando:

```
$ git remote -v
```

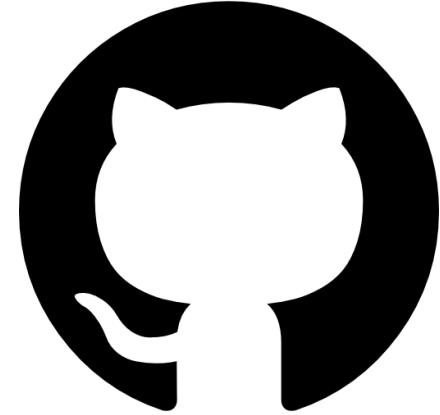
Y te debe devolver lo siguiente:

```
origin - https://github.com/mi-usuario/mi-repositorio.git (fetch)  
origin - https://github.com/mi-usuario/mi-repositorio.git (push)
```

Conectando!

Básicamente, lo que este comando acaba de hacer es **indicarle al repositorio local con que repositorio remoto se deberá sincronizar.**

Así de esa manera, cuando quieras enviar los archivos de tu computadora a GitHub o traer los archivos de GitHub a tu maquina, **los repositorios ya estarán conectados entre si.**



```
$ git remote -v
```

GitHub

Conectando nuestro repositorio local a GitHub

Resumen:

Con toda la configuración inicial lista y con el repositorio en GitHub ya creado, GitHub te mostrará algunas instrucciones.

- Busca la URL del repositorio que aparece justo debajo del título (será algo como <https://github.com/tu-usuario/TuArchivo.git>).
- En la terminal de VS Code, añade el repositorio remoto que acabas de crear en GitHub a tu repositorio local usando el siguiente comando:
git remote add origin https://github.com/tu-usuario/TuArchivo.git
- Esto establece la conexión entre tu repositorio local y GitHub. Reemplaza tu-usuario y TuArchivo con el nombre de tu usuario y tu repositorio.

Explicacion con imagenes Paso a Paso

GitHub

Conectando nuestro repositorio local a GitHub

Pasos anteriores a hacerlo:

Para poder hacer esto, debemos tener:

- Una cuenta en GitHub.
- Un repositorio local que utilizaremos para poder conectarlo.

Luego debemos inicializar un repositorio. Para esto, ejecutemos **git init** en la carpeta que queramos conectar el repositorio.

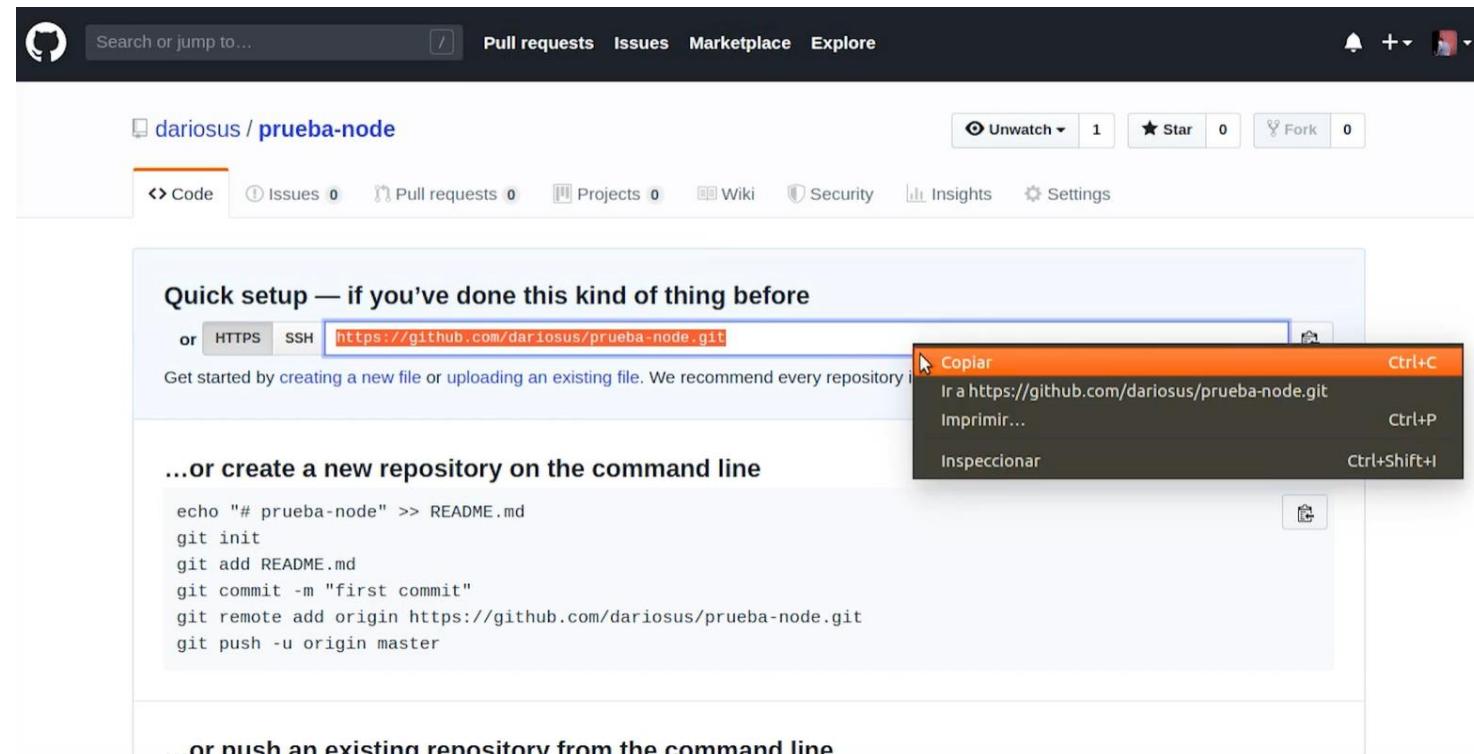
Luego tenemos que indicar al repositorio nuestros usuario ejecutando dos comandos:

- **git config user.name “mi usuario”** (escribimos nuestro nombre de usuario).
- **git config user.email “miCorreo@email.com”** (escribimos nuestra dirección de correo).

Conectando!

Entonces, para entender mejor, volvamos a donde nos quedamos en la creación del repositorio.

Apenas creamos nuestro repositorio en Github vemos una pantalla de este estilo, donde muy a mano nos dará esta URL que debemos copiar



Conectando!

La URL de la slice anterior es solo la url del repo, por lo que en la terminal debemos utilizar además el comando origin. Es por eso que copiamos este comando y lo pegaremos en la terminal:

The screenshot shows a GitHub repository page for 'dariosus/prueba-node'. At the top, there's a search bar and navigation links for Pull requests, Issues, Marketplace, and Explore. Below that, the repository name 'dariosus / prueba-node' is displayed, along with statistics: 1 pull request, 0 issues, 0 pull requests, 0 projects, 0 wiki, 0 security, 0 insights, and 0 settings. A 'Code' tab is selected. A 'Quick setup — if you've done this kind of thing before' section provides instructions for cloning the repository via HTTPS or SSH, and for creating a new repository on the command line. The command-line section includes a code editor with the following text:

```
echo "# prueba-node" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/dariosus/prueba-node.git
git push -u origin master
```

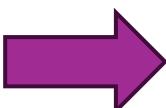
The screenshot shows a terminal session on a Linux system (sus@AR-BE-FS-00). The user runs several commands to initialize a new Git repository and add it to GitHub:

```
sus@AR-BE-FS-00:~/nodejs$ git init
Initialized empty Git repository in /home/sus/nodejs/.git/
sus@AR-BE-FS-00:~/nodejs$ ls
images public src
sus@AR-BE-FS-00:~/nodejs$ ls -a
. . . .git images public src
sus@AR-BE-FS-00:~/nodejs$ git config user.name "dariosus"
sus@AR-BE-FS-00:~/nodejs$ git config user.email "dario15th@gmail.com"
sus@AR-BE-FS-00:~/nodejs$ git remote add origin https://github.com/dariosus/prue
ba-node.git
sus@AR-BE-FS-00:~/nodejs$
```

A purple arrow points from the GitHub repository URL in the quick setup section of the previous screenshot to the 'origin' URL in the terminal command.

Conectando!

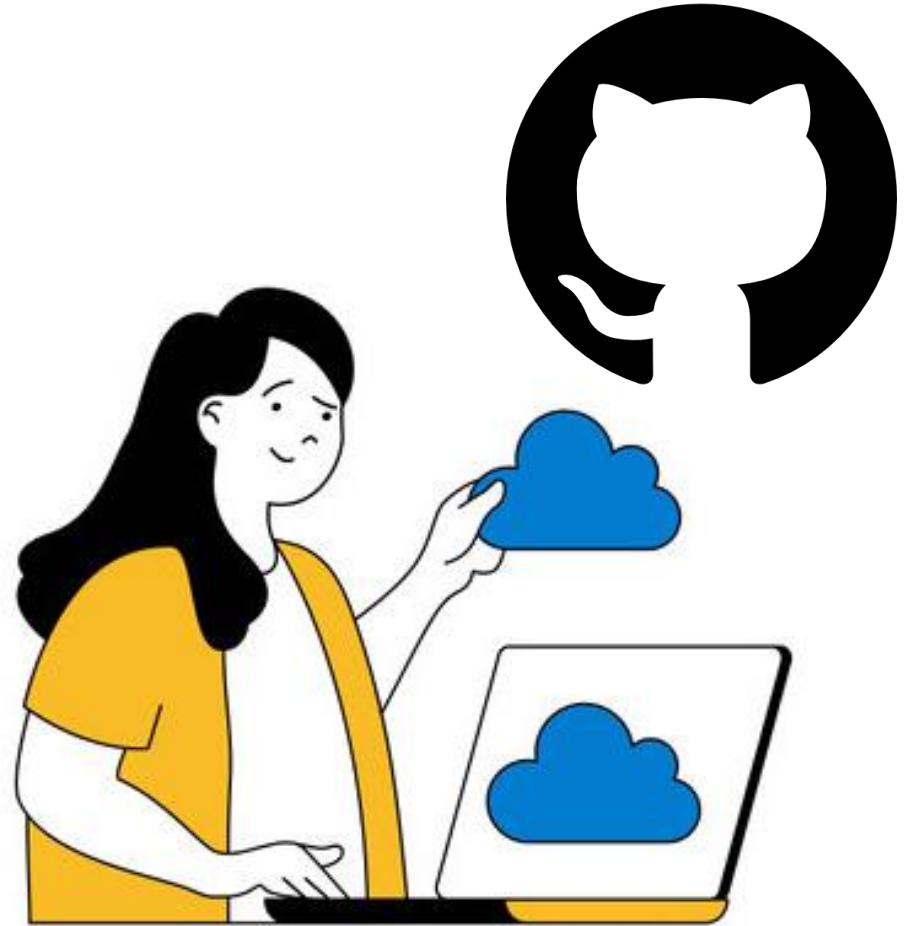
Para verificar que todo salió correctamente, hacemos el comando **git remote -v** que nos devolverá los dos comandos siguientes:



```
sus@AR-BE-FS-00:~/nodejs$ git init
Initialized empty Git repository in /home/sus/nodejs/.git/
sus@AR-BE-FS-00:~/nodejs$ ls
images public src
sus@AR-BE-FS-00:~/nodejs$ ls -a
. .. .git images public src
sus@AR-BE-FS-00:~/nodejs$ git config user.name "dariosus"
sus@AR-BE-FS-00:~/nodejs$ git config user.email "dario15th@gmail.com"
sus@AR-BE-FS-00:~/nodejs$ git remote add origin https://github.com/dariosus/prueba-node.git
sus@AR-BE-FS-00:~/nodejs$ git remote -v
origin https://github.com/dariosus/prueba-node.git (fetch)
origin https://github.com/dariosus/prueba-node.git (push)
sus@AR-BE-FS-00:~/nodejs$
```

Conectando!

De esta manera, ya tenes todo conectado para comenzar a agregar archivos a tu repositorio local y enviarlos posteriormente al repositorio remoto en la nube!



Subiendo archivos

Subiendo Archivos

Seguro te estarás preguntando **como finalmente llevar los archivos de tu repositorio local a tu repositorio remoto en GitHub?**

Es muy sencillo, pero primero recapitulemos un poco lo que venimos haciendo.

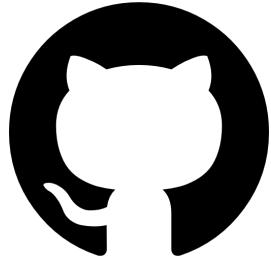
Todas las cosas que hicimos hasta el momento fueron desde nuestra computadora:

- ✓ Creamos el repositorio
- ✓ Agregamos archivos al mismo
- ✓ Commiteamos distintos mensajes
- ✓ Vimos como se encuentra el estado de nuestro repositorio y sus archivos

Subiendo Archivos

Ahora es momento de llevar esos archivos a la nube (directamente a GitHub)

Este proceso debería ser bastante sencillo y no presentar ningún tipo de problema porque si tenes debidamente vinculados tu repositorios, este proceso lo vas a llevar a cabo fácilmente.



Subiendo Archivos

Es importante que tengas en cuenta que siempre que deseas subir tus archivos a github, tenes que tener previamente commiteado TODO para poder lograrlo.

Es decir, **cuando subimos las cosas a la nube, solo se suben los commits ya cerrados.**



Antes de subir a Github -> Tener todo comiteado

Subiendo Archivos

Teniendo esto en claro, **Git necesita que le indiques que deseas subir los archivos.** Y para ello, vas a hacer uso del comando:

```
$ git push
```

Push en castellano significa “empujar”, pero técnicamente lo que sucede con este comando es que esta **solicitando a git que “inserte”**, es decir, que envíe los archivos que tenes en tu repo local al repositorio remoto.



Subiendo Archivos

Git Push, necesita adicionalmente otro conjunto de palabras:

```
$ git push origin master
```

Desde Octubre 2020, github cambio el nombre de la rama master para los repositories nuevos. Ahora esta rama se llama main. Por lo tanto tu comando debera ser asi:

git push origin main

Subiendo Archivos

Este comando significa entonces que querés llevar los archivos de tu repositorio local a tu repositorio remoto.

Para Git, el repositorio remoto se llama “origin”

Y que adicionalmente quierés que se inserten los mismos en la **rama principal que se llama master**.

Rama? Si, ahora lo vemos en profundidad, perdón el spoiler.



\$ git push origin master

Subiendo Archivos

Si volvemos a la computadora en lo que habíamos quedado y suponiendo que queremos subir tres archivos que creamos con anterioridad, podemos hacer el comando que acabamos de aprender:

Creación de archivos

Nuestro comando nuevo seguido
de mi user y password

```
sus@AR-BE-FS-00:~/nodejs
sus@AR-BE-FS-00:~/nodejs$ git status
En la rama master
nothing to commit, working directory clean
sus@AR-BE-FS-00:~/nodejs$ git log
commit e57bb028acf8a6602eda1254444a0a8ae72f601a
Author: dariosus <dario15th@gmail.com>
Date:   Fri Oct 11 17:56:54 2019 -0300

    Agregue un console log

commit e5784481f3921c327e320bfd72a5542e1a0cb190
Author: dariosus <dario15th@gmail.com>
Date:   Fri Oct 11 17:50:41 2019 -0300

    Cree el archivo de funciones

commit 289a52e762c39a47bf8e80ca77cfef597bbf63c0a
Author: dariosus <dario15th@gmail.com>
Date:   Fri Oct 11 17:49:46 2019 -0300

    Creando los 3 archivos principales
sus@AR-BE-FS-00:~/nodejs$ git push origin master
Username for 'https://github.com': dariosus
Password for 'https://dariosus@github.com':
```

Subiendo Archivos

Luego de dar enter se actualiza la terminal y nos devuelve los siguiente, explicando que se logro con éxito el comando

```
sus@AR-BE-FS-00: ~/nodejs

commit e5784481f3921c327e320bfd72a5542e1a0cb190
Author: dariosus <dario15th@gmail.com>
Date:   Fri Oct 11 17:50:41 2019 -0300

    Cree el archivo de funciones

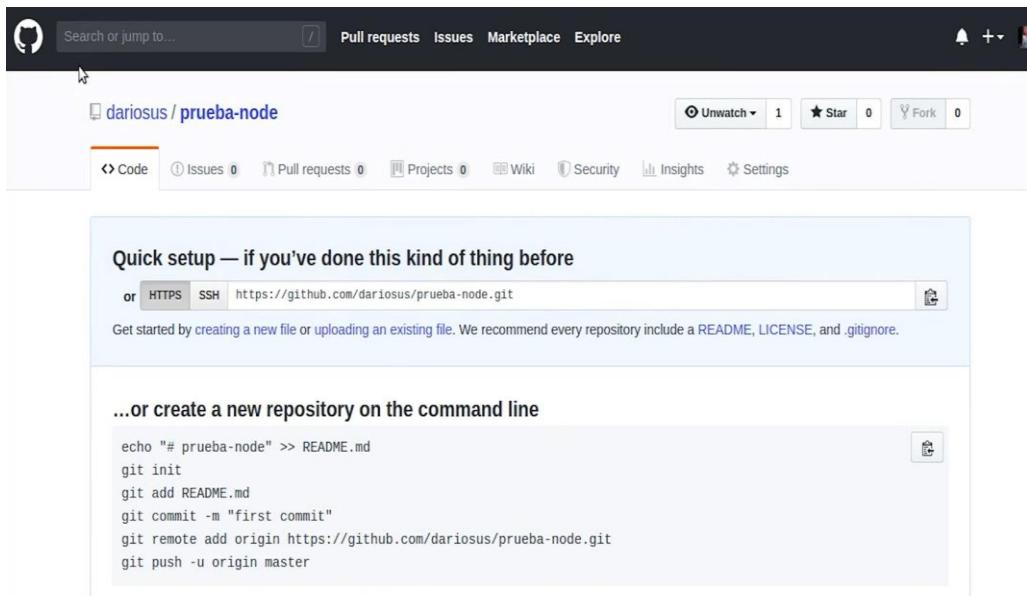
commit 289a52e762c39a47bf8e80ca77cfef597bbf63c0a
Author: dariosus <dario15th@gmail.com>
Date:   Fri Oct 11 17:49:46 2019 -0300

    Creando los 3 archivos principales
sus@AR-BE-FS-00:~/nodejs$ git push origin master
Username for 'https://github.com': dariosus
Password for 'https://dariosus@github.com':
Counting objects: 8, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (8/8), 662 bytes | 0 bytes/s, done.
Total 8 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/dariosus/prueba-node.git
 * [new branch]      master -> master
sus@AR-BE-FS-00:~/nodejs$
```

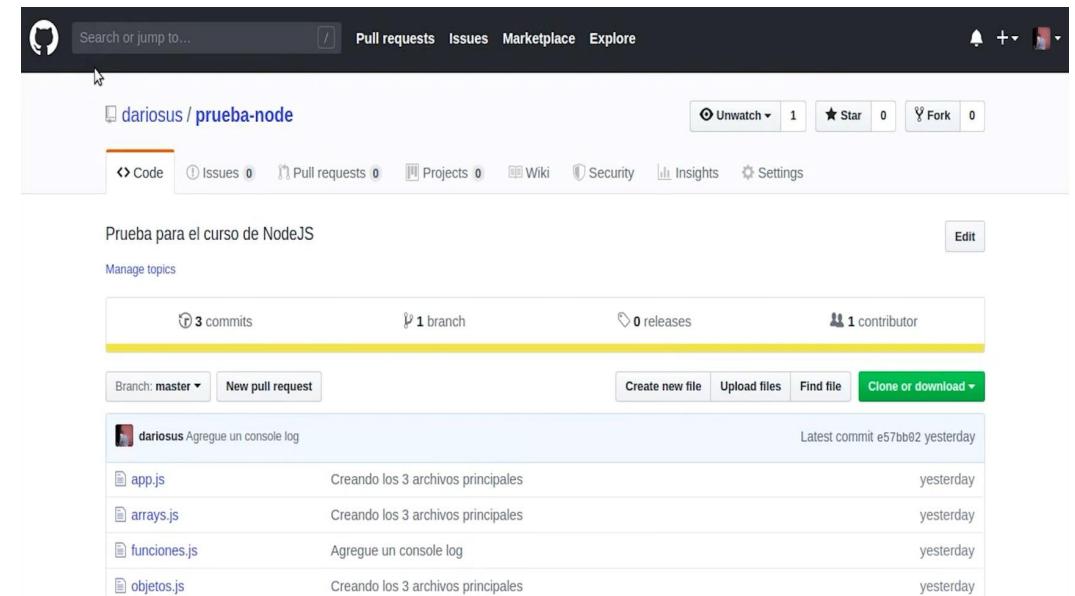
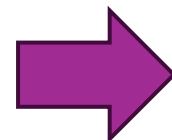
Subiendo Archivos

Si volvemos a la pagina online y recargamos, veremos que se actualizo! Es decir que nuestro repositorio ya no esta vacio.

Veamos mas a profundidad



The screenshot shows the initial setup of a new GitHub repository named 'prueba-node'. It includes sections for 'Quick setup' (with instructions for HTTPS and SSH), a command-line guide for creating a repository, and a file browser interface.

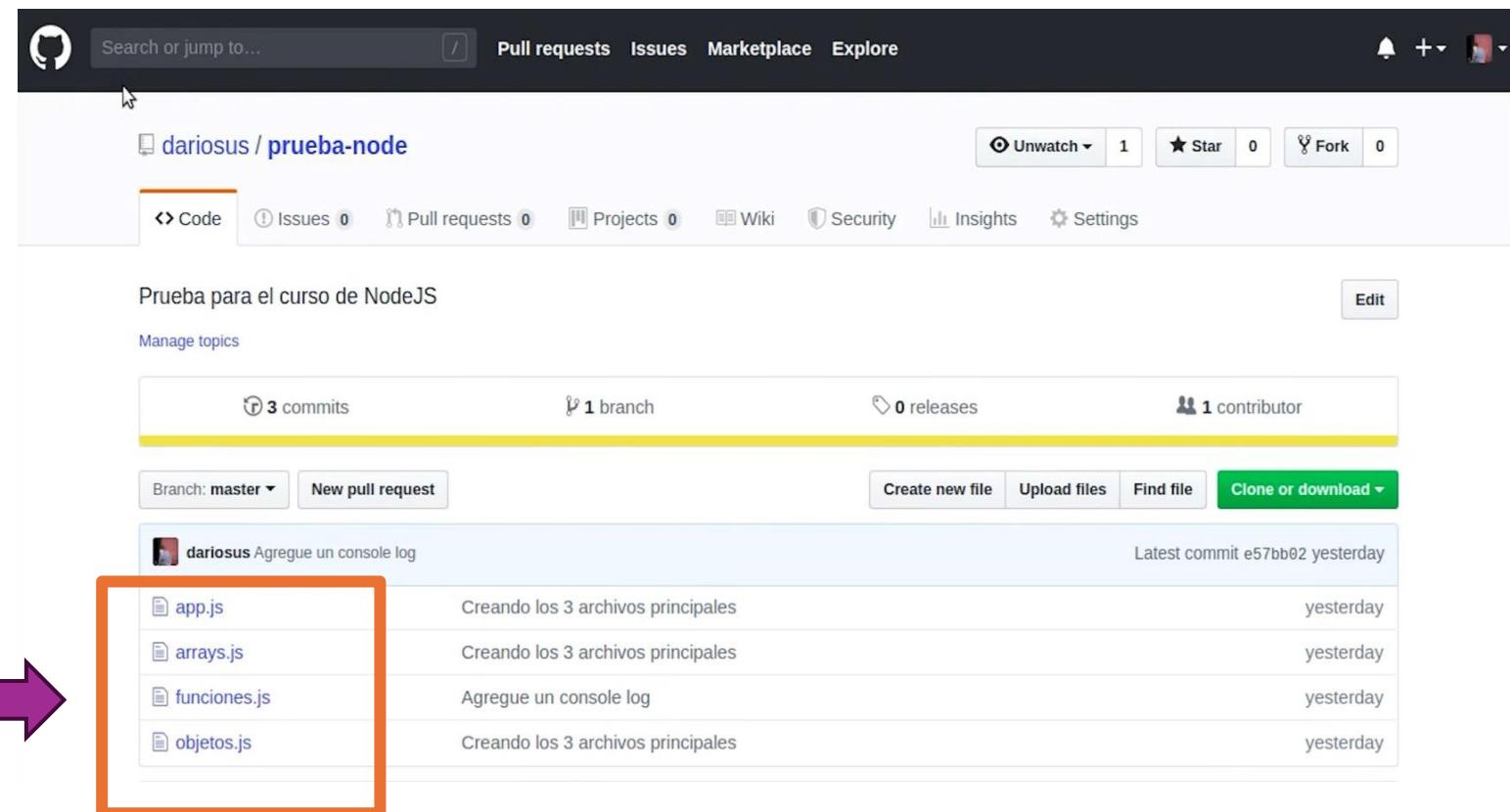
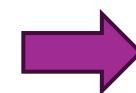


The screenshot shows the same repository after files have been uploaded. The repository name is now 'Prueba para el curso de NodeJS'. The commit history lists four files: 'app.js', 'arrays.js', 'funciones.js', and 'objetos.js', all added by the user 'dariosus' yesterday. The repository summary indicates 3 commits, 1 branch, 0 releases, and 1 contributor.

Subiendo Archivos

Estos son los archivos que subimos anteriormente.

En caso de que fueran carpetas aparecerían con ese formato y podríamos entrar en las mismas y explorarlas libremente.



dariosus / prueba-node

Prueba para el curso de NodeJS

Manage topics

3 commits 1 branch 0 releases 1 contributor

Branch: master New pull request

Create new file Upload files Find file Clone or download

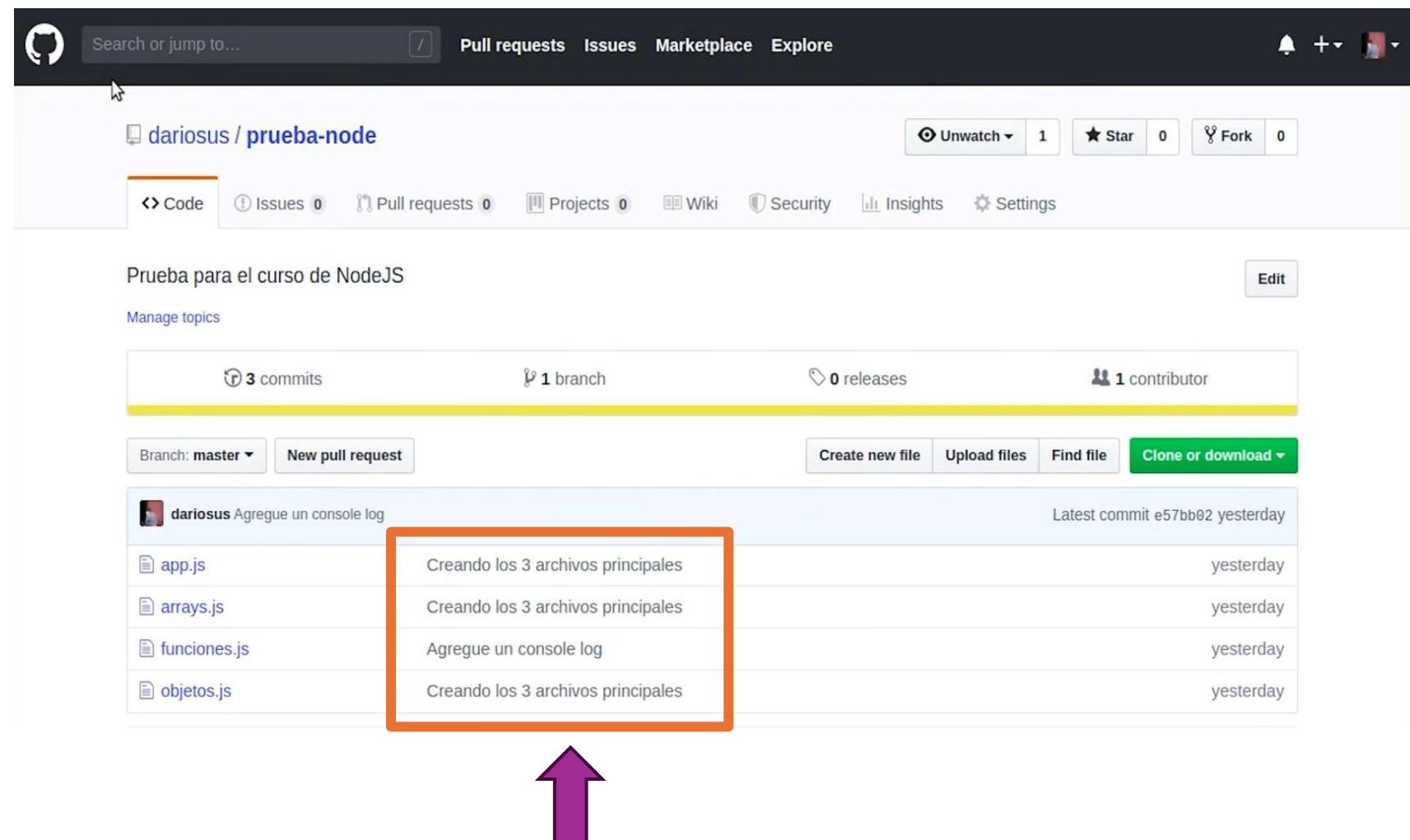
dariosus Agregue un console log

Latest commit e57bb02 yesterday

File	Message	Date
app.js	Creando los 3 archivos principales	yesterday
arrays.js	Creando los 3 archivos principales	yesterday
funciones.js	Agregue un console log	yesterday
objetos.js	Creando los 3 archivos principales	yesterday

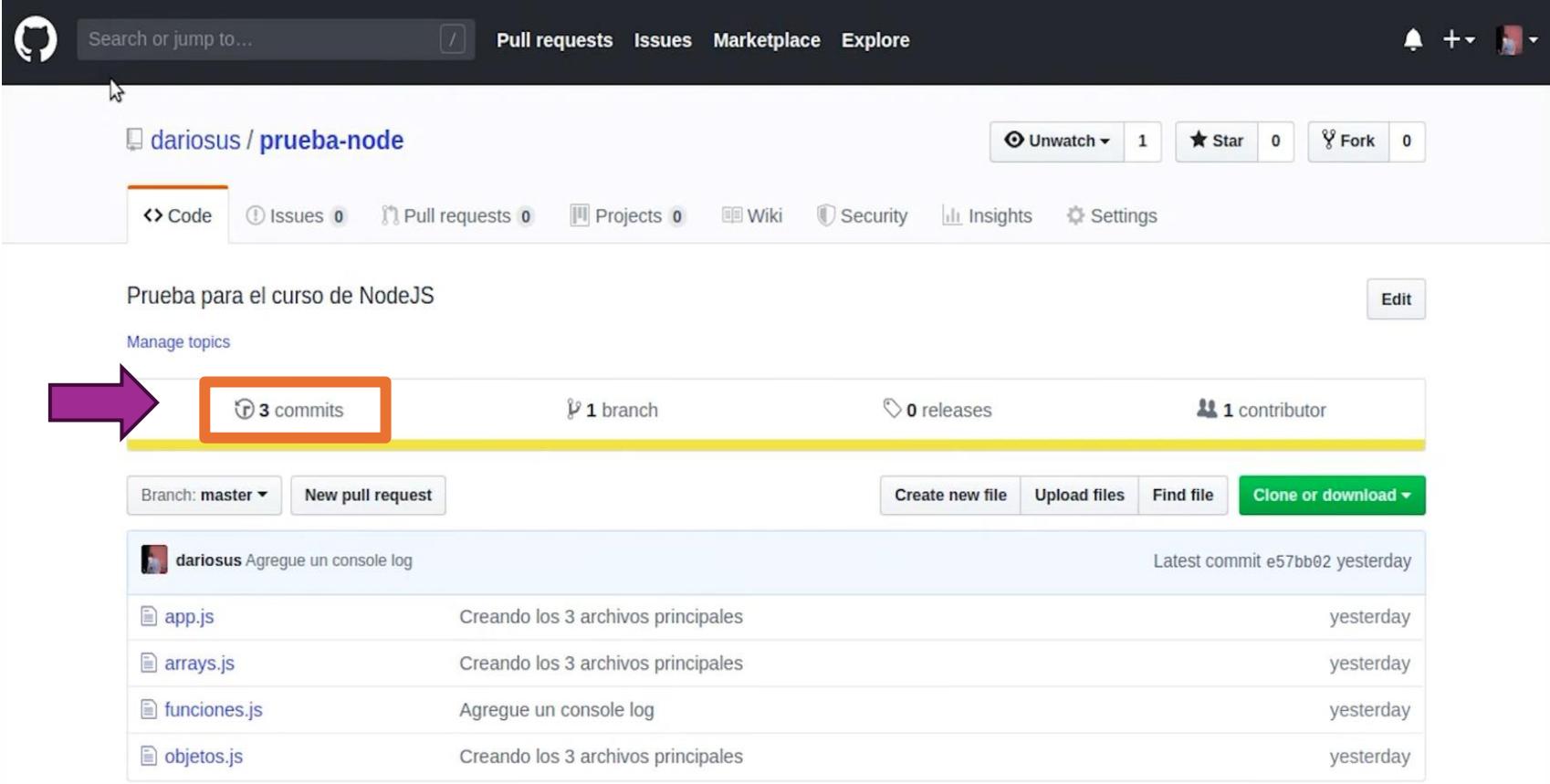
Subiendo Archivos

Podemos ver a la derecha de cada archivo, que aparece el ultimo mensaje de commit en el que fue modificado.



Subiendo Archivos

Si quisiéramos ver mejor estos, podemos ingresar a “commits” y verlos en mejor profundidad.

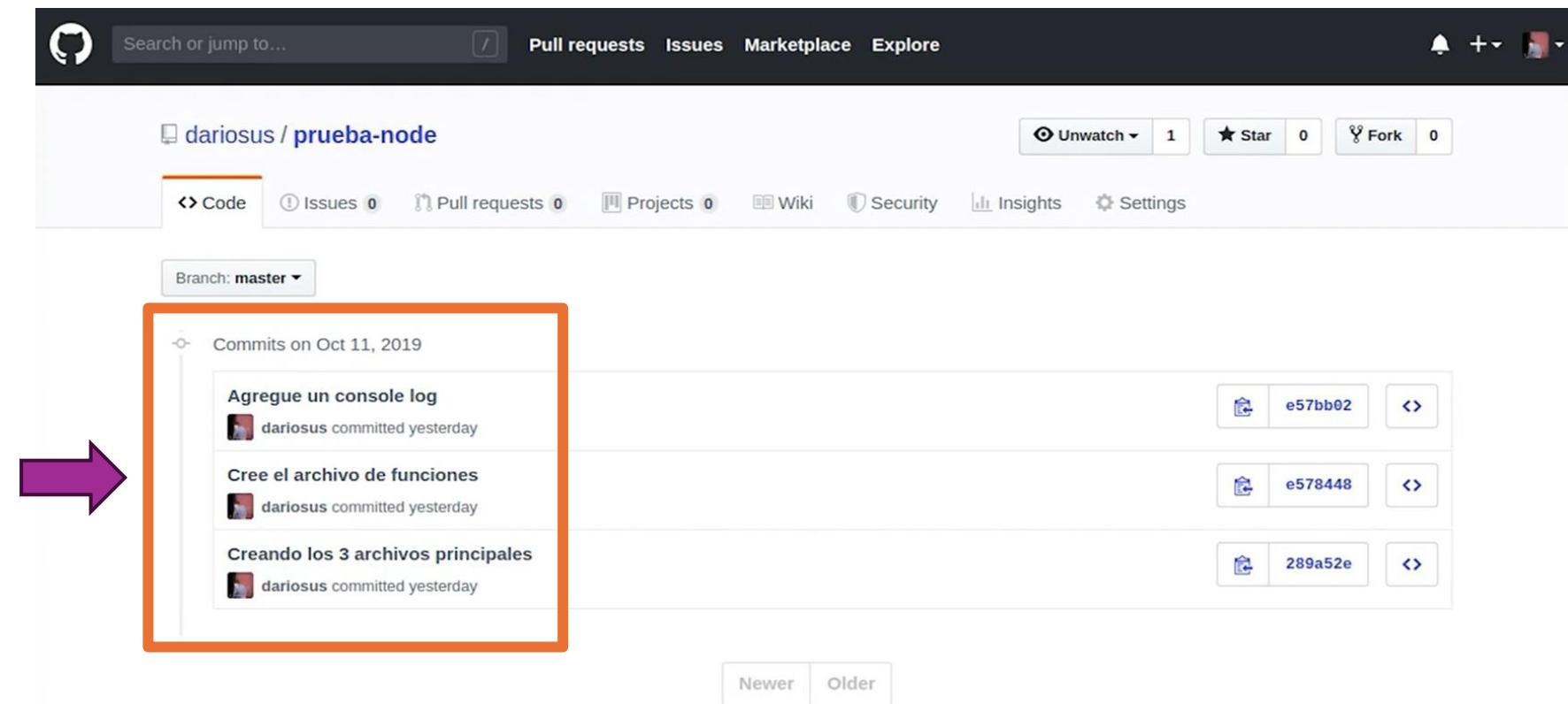


The screenshot shows a GitHub repository page for 'dariosus / prueba-node'. The top navigation bar includes 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below the repository name, there are buttons for 'Unwatch', 'Star', 'Fork', and a 'Code' tab which is currently selected. A purple arrow points to the '3 commits' button, which is highlighted with a red border. Other metrics shown include 0 issues, 0 pull requests, 0 projects, 0 wiki, 0 security, 0 insights, and 1 contributor. The commit history lists four commits made by 'dariosus' yesterday:

File	Message	Date
app.js	Creando los 3 archivos principales	yesterday
arrays.js	Creando los 3 archivos principales	yesterday
funciones.js	Agregue un console log	yesterday
objetos.js	Creando los 3 archivos principales	yesterday

Subiendo Archivos

Aquí tendremos el historial completo, con una línea de tiempo, observando cada uno de los cambios que se realizo en este repositorio, quien los hizo y en que momento



Subiendo Archivos

Ademas, podemos entrar en un commit en particular para observar cual fue el cambio que fue hecho.

Por ejemplo, aquí nos dice que se creo vacio “funciones.js”

The screenshot shows a GitHub commit page for the repository 'dariosus / prueba-node'. The commit message is 'Cree el archivo de funciones'. It was made by 'dariosus' yesterday. The commit hash is 'e5784481f3921c327e320bfd72a5542e1a0cb190'. The commit details show '1 parent 289a52e' and 'commit e5784481f3921c327e320bfd72a5542e1a0cb190'. Below the commit message, it says 'Showing 1 changed file with 0 additions and 0 deletions.' A file named 'funciones.js' is listed with a status of 'No changes.' At the bottom, there is a comment section with a placeholder 'Leave a comment' and a rich text editor.

Subiendo Archivos

En este otro ejemplo vemos como se agrego una línea al código (en este caso un console.log)

Si por el contrario borráramos líneas se verían en rojo

The screenshot shows a GitHub commit page for the repository 'dariosus / prueba-node'. The commit message is 'Agregue un console log' and it was made by 'dariosus' yesterday. The commit hash is 'e57bb028acf8a6602eda1254444a0a8ae72f601a'. The commit details show a single change in the file 'funciones.js': a new line of code '+ console.log(1);' has been added. The GitHub interface highlights this addition in green. Below the commit, there are 0 comments and a 'Lock conversation' button. At the bottom, there are 'Write' and 'Preview' buttons, and a rich text editor toolbar.

GitHub

Subiendo Archivos

Veamos por ejemplo el repositorio de VSC que es publico y todos lo podemos ver:

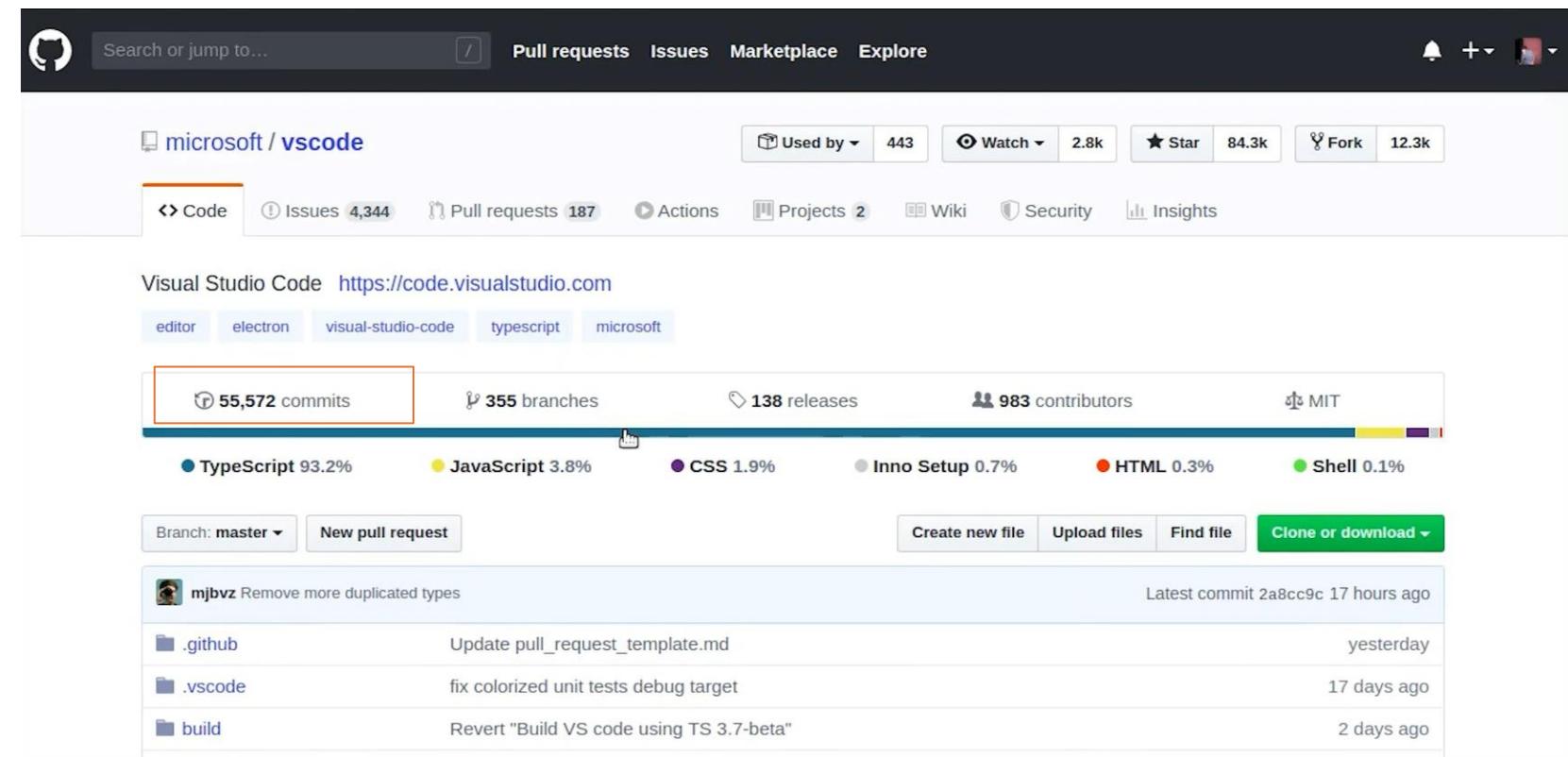
<https://github.com/microsoft/vscode>

The screenshot shows the GitHub repository page for `microsoft / vscode`. The page includes a search bar, navigation links for Pull requests, Issues, Marketplace, and Explore, and a header with metrics: Used by 443, Watch 2.8k, Star 84.3k, Fork 12.3k. Below the header are tabs for Code, Issues (4,344), Pull requests (187), Actions, Projects (2), Wiki, Security, and Insights. The main content area displays the repository details: Visual Studio Code (<https://code.visualstudio.com>), tags (editor, electron, visual-studio-code, typescript, microsoft), and a file distribution chart showing TypeScript (93.2%), JavaScript (3.8%), CSS (1.9%), Inno Setup (0.7%), HTML (0.3%), and Shell (0.1%). The repository has 55,572 commits, 355 branches, 138 releases, and 983 contributors. A green button at the bottom right says "Clone or download". The commit history lists recent changes: "Remove more duplicated types" by mjbvz (yesterday), "Update pull_request_template.md" by .github (yesterday), "fix colorized unit tests debug target" by .vscode (17 days ago), and "Revert 'Build VS code using TS 3.7-beta'" by build (2 days ago). The latest commit was 2a8cc9c, 17 hours ago.

GitHub

Subiendo Archivos

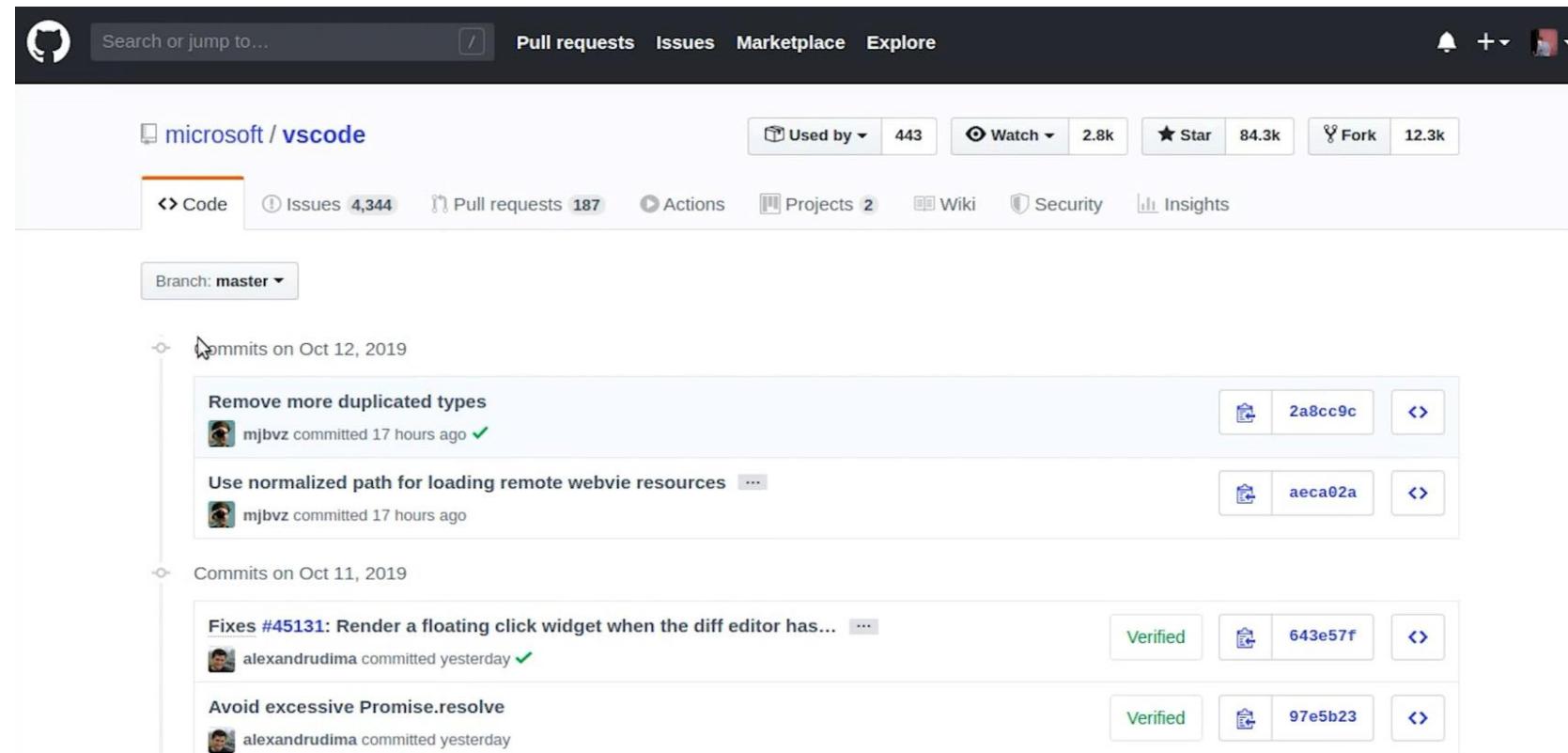
En el mismo podemos ver estadísticas de las tecnologías que se usaron para programar y también podemos ver la cantidad de commits que tiene!



Subiendo Archivos

Por lo general los repositorios de github son públicos, ya que algo que se fomenta desde git es el **open source**

Que nos permite ver código ajeno, inspirarnos y porque no colaborar entre todos.

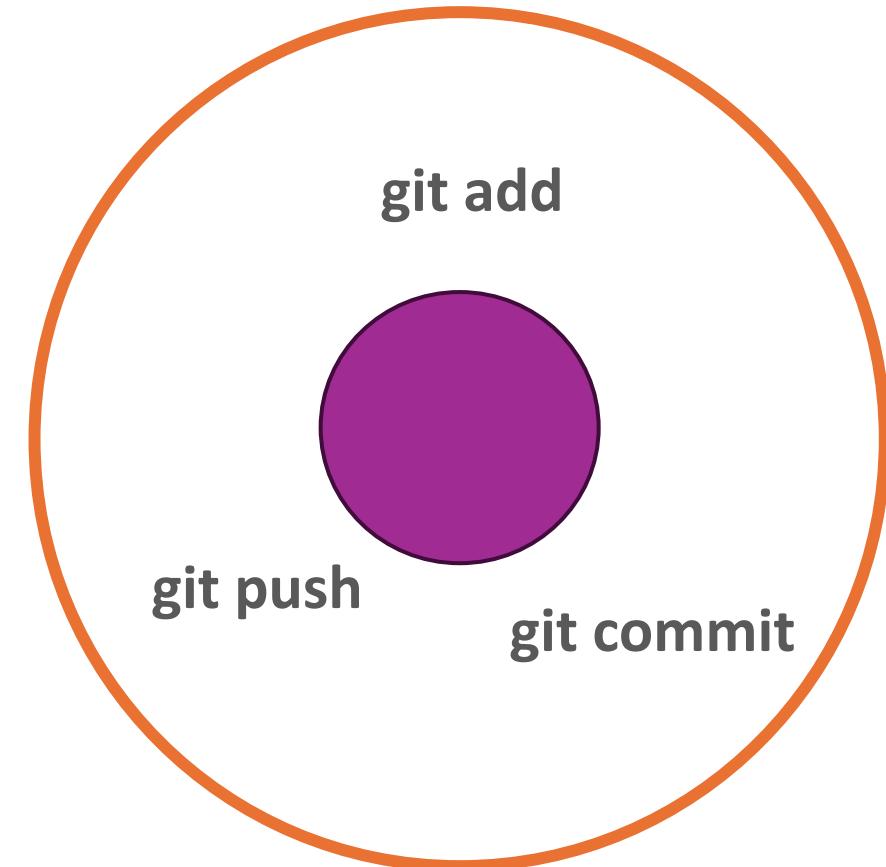


Subiendo Archivos

Ya veras que cuando le tomes ritmo, el proceso de git add, git commit y git push se volverá un ciclo constante y mas manejable.

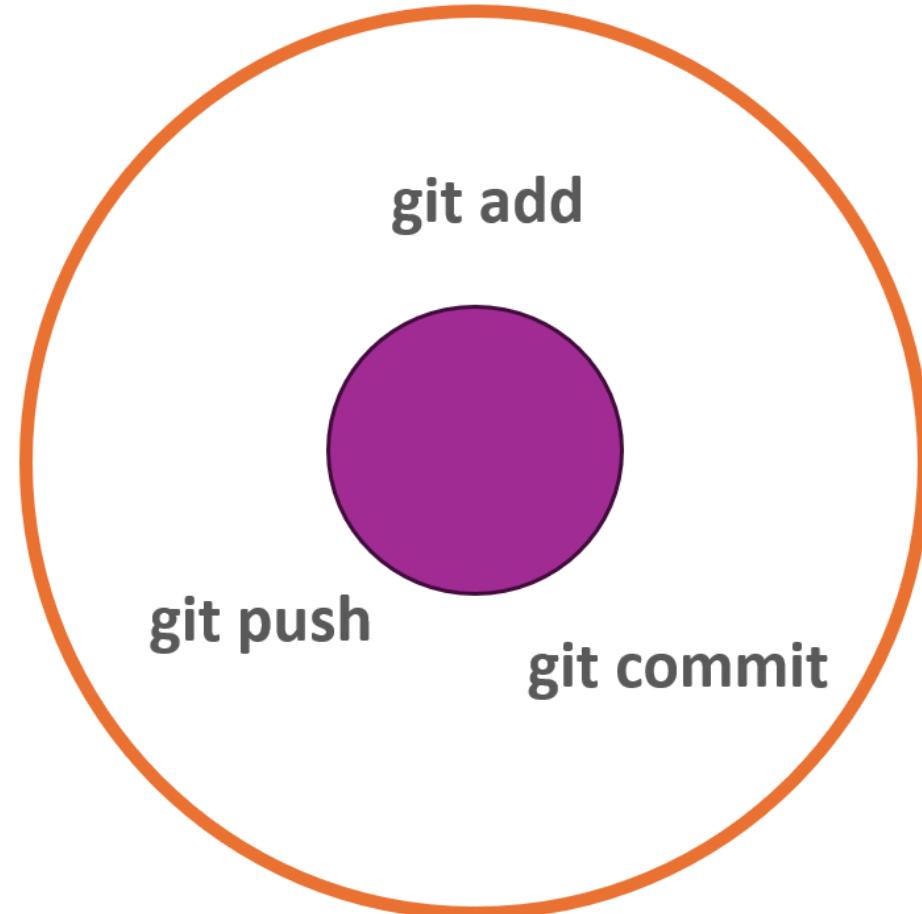
Una pregunta muy recurrente es...

Cuando debo hacer un commit? Y la verdad que esa desicion la tomas vos, pero se recomienda que un commit sea realizado **luego de haber agregado una cantidad de funcionalidades adicionales al código, siempre que sea una versión estable!**



Subiendo Archivos

Porque no queremos subir una versión inestable que si la descarga un compañero introduce nuevos errores, o que cosas que antes andaban bien dejen de funcionar.



Resumiendo: Subiendo archivos

Resumiendo:

Agregar y commitear los archivos al repositorio

- Para asegurarte de que todos los archivos del proyecto están listos para subirse, primero agrégalos al "índice" de Git (etapa de preparación) usando:

git add .

El punto (.) significa que agregará todos los archivos de tu carpeta de proyecto.

- A continuación, guarda tus cambios en el historial de Git con un commit:

git commit -m "Primer commit: Cambios"

El mensaje entre comillas ("Primer commit: Cambios") es opcional, pero es una buena práctica agregar descripciones breves que expliquen lo que se hizo en el commit.

Resumiendo

Subir el repositorio local a GitHub:

- Ahora que tu repositorio local está vinculado al repositorio remoto en GitHub, es hora de subir los archivos.
- Usa el siguiente comando para subir (hacer "push") tus archivos al repositorio de GitHub:

git push -u origin master

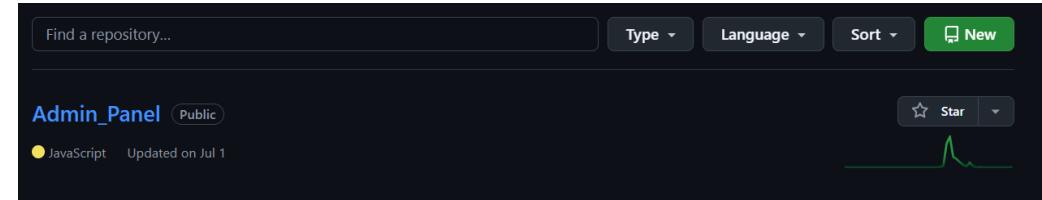
Este comando sube los archivos a la rama principal (master) del repositorio remoto.

El -u hace que Git recuerde esta configuración para que no tengas que escribir la parte de origin master la próxima vez.

Resumiendo:

Verificar la subida en GitHub

- Ve a la página del repositorio en GitHub y verifica que los archivos subidos aparecen correctamente.
- ¡Listo! Tu proyecto está ahora en GitHub y puedes compartir el enlace con otros o seguir trabajando en él.



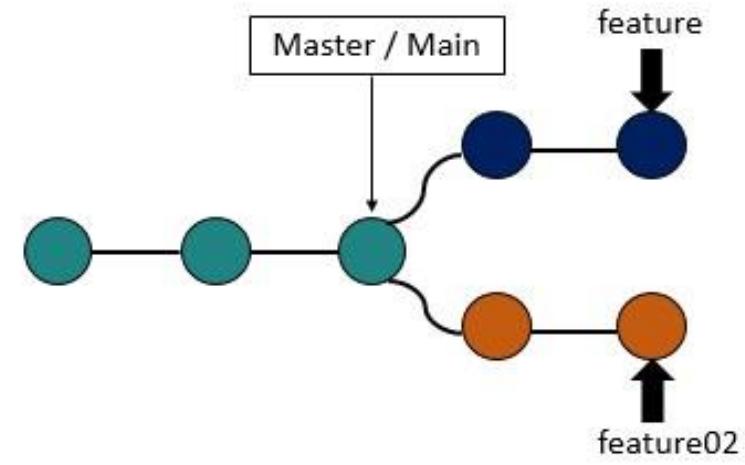
Ramas

GitHub

Ramas:

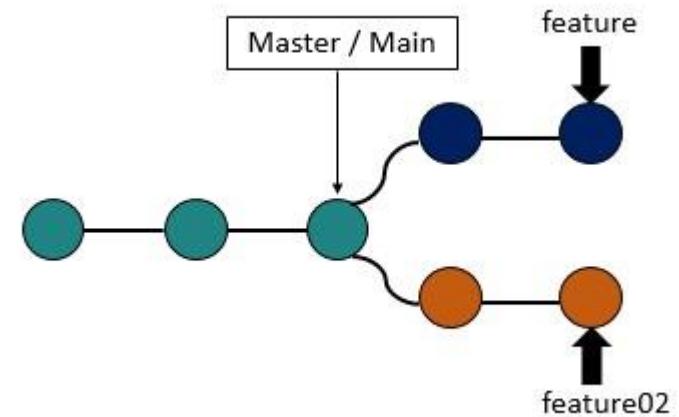
Las ramas dentro del repositorio es una copia alternativa del mismo hasta ese momento.

Podemos decir que es una línea paralela, en la cual podes ir agregando nuevas funcionalidades, sin tener que modificar la línea original de tiempo ni afectar que código hay ahí.



Ramas:

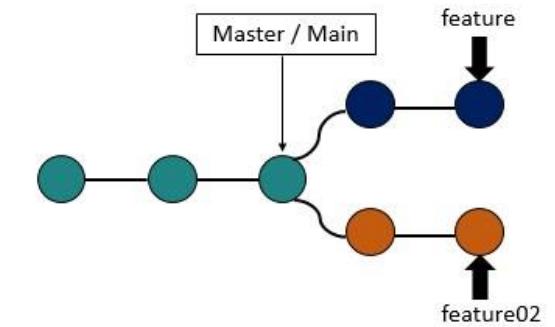
Es por así decirlo una versión dos de tu proyecto, en la cual podes probar cosas nuevas y si te gustan, después las podes fusionar con la rama principal.



GitHub

Ramas:

Este concepto si demanda un poco mas de experiencia, por lo que les recomiendo que primero practiquen subiendo los archivos a GitHub para poco a poco **incursionar en temas mas complejos.**



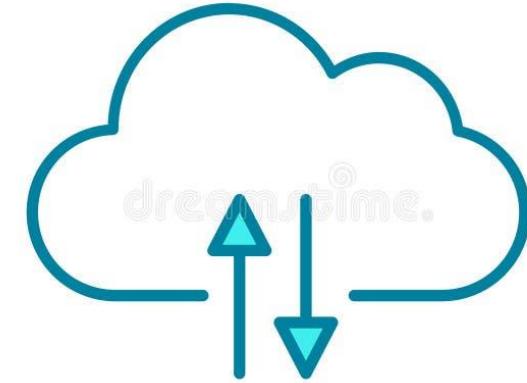
Bajando archivos

GitHub

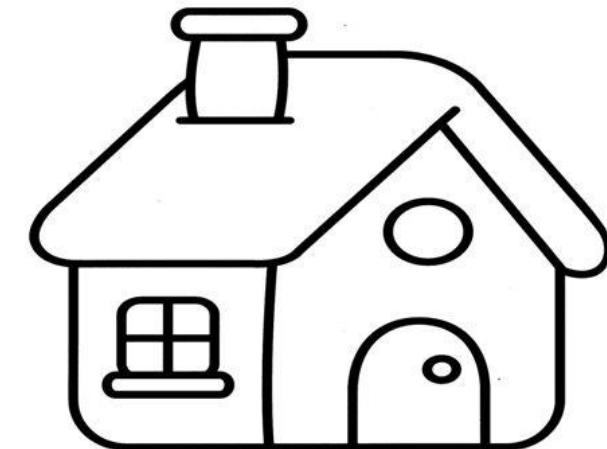
Introducción:

En ocasiones, nos solemos llevar trabajo a casa para adelantar algo del mismo y dichos avances después los necesitas de nuevo en tu espacio de trabajo.

Ahora veremos como podes bajar los proyectos que tengas en github en cualquier computadora del mundo. Y tambien como bajar los cambios de nuestros compañeros.



- ↳ Bajar los proyectos que tengas en github en cualquier computadora del mundo

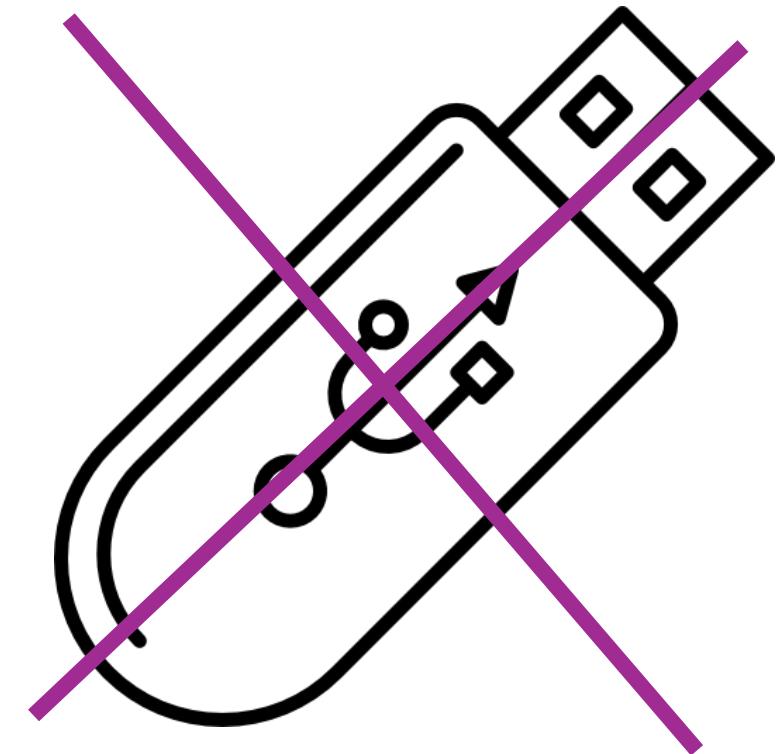


GitHub

Introducción:

Trabajando en programación nos pasará que necesitamos nuestros archivos en más de un espacio de trabajo, y como ya venimos conversando, hacerlo con un pendrive no es para nada una opción.

Por eso con git vamos a descargar en nuestras computadoras la última versión que esté en el repositorio remote.



GitHub

Introducción:

El proceso para hacer esto es super amigable, basta con ejecutar un comando desde la terminal y te dejará todo listo para comenzar a trabajar.

Este comando del que hablamos es:

```
git clone
```



GitHub

Introducción:

Este comando nos permite crear una copia exacta en la computadora de todos los archivos existentes en el repo remote.

Pero git clone necesita de una tercera parte que le sirva para indicar cual es el repositorio remoto del cual queremos hacer una copia.

Esta parte va a ser la URL exacta en la que se encuentran publicados los archivos

Crear una copia exacta en la computadora de todos los archivos existentes en un repositorio remoto



GitHub

Introducción:

Entonces, el comando completo deberia verse asi:

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

Una vez ejecutado, git se encargara de escargar todos los archivos alli presentes y los dejara listos en la computadora para poder ser utilizados.

IMPORTANTE: Este comando solo se ejecuta una sola vez, es decir la primera vez cuando queres descargar los archivos de git y cuando los mismos no esten presentes en la computadora donde estas trabajando.

Si ya tenes los archivos y los queres actualizar el comando sera otro

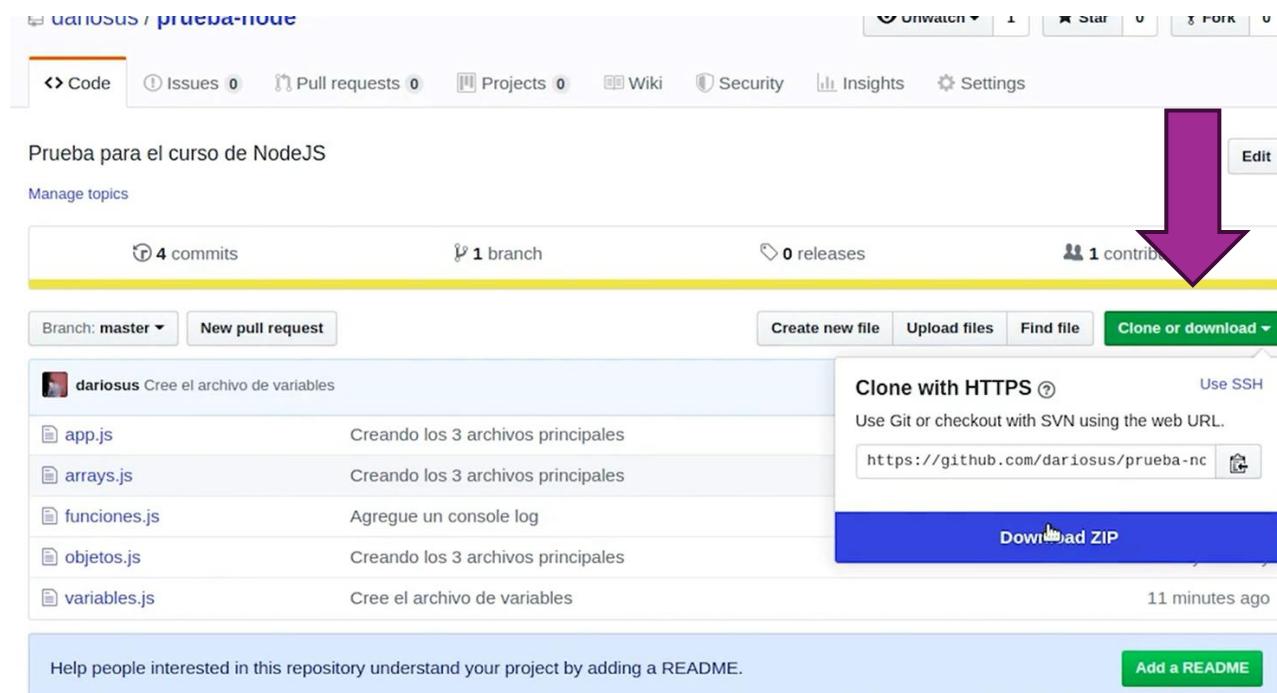
GitHub

Paso a paso para descargar los archivos de GitHub:

Estamos en nuestro repo remoto y supongamos que los quiero llevar a una compu nueva.

Vamos al boton que dice “**clone or download**”

Tenemos la opcion si queremos de descargarlos en un archive zip, pero realmente no es lo que estamos buscando porque no esta sincronizado con nuestro repo remoto.





Paso a paso para descargar los archivos de GitHub:

En cambio la opcion de clonado va a sincronizar nuestra carpeta en nuestra computadora con el repo remoto de github

Este es el hipervinculo que debemos copiar.

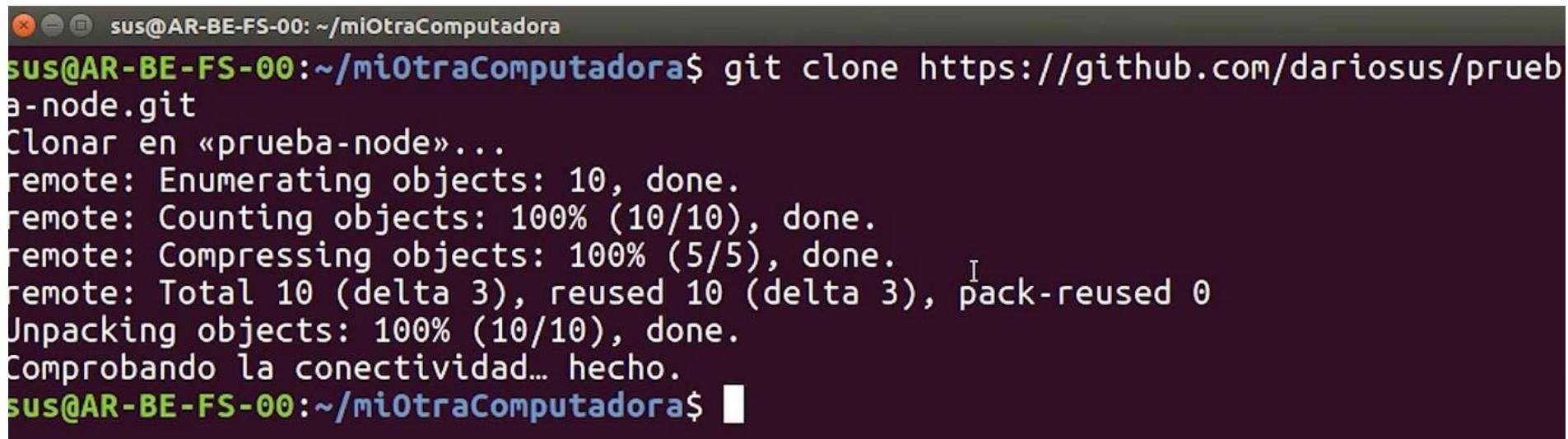
The screenshot shows a GitHub repository page for 'dariosus/prueba-node'. The page includes navigation links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Security, Insights, and Settings. Below these are statistics: 4 commits, 1 branch, 0 releases, and 1 contributor. A 'Clone or download' button is visible. On the right, there's a 'Clone with HTTPS' section with a URL: <https://github.com/dariosus/prueba-node>. A purple arrow points to this URL. The main content area lists files: app.js, arrays.js, funciones.js, objetos.js, and variables.js, each with a brief description. At the bottom, there's a note to 'Add a README'.

GitHub

Paso a paso para descargar los archivos de GitHub:

Luego de copiarlo vamos a nuestra carpeta donde lo queremos clonar, abrimos la terminal y hacemos el comando **git clone + la url**

Asi, el comando se encargara de descargar todos los archivos en esta carpeta.



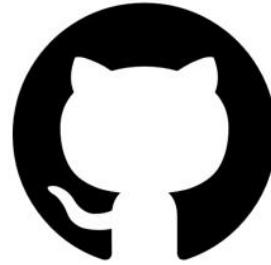
```
sus@AR-BE-FS-00: ~/miOtraComputadora
sus@AR-BE-FS-00:~/miOtraComputadora$ git clone https://github.com/dariosus/prueba-node.git
Clonar en «prueba-node»...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 10 (delta 3), reused 10 (delta 3), pack-reused 0
Unpacking objects: 100% (10/10), done.
Comprobando la conectividad... hecho.
sus@AR-BE-FS-00:~/miOtraComputadora$ █
```

GitHub

Bajando archivos

Facil no? Pero... que sucede cuando **trabajas en la maquina de tu trabajo, subir los archivos a github y despues los queres descargar en tu maquina personal?** o cuando otra persona subio cambios y queremos descargarlos?

Ya sabemos que los archivos existen en las dos compus, ahora veamos como mantener sincronizados los archivos existentes en esas computadoras



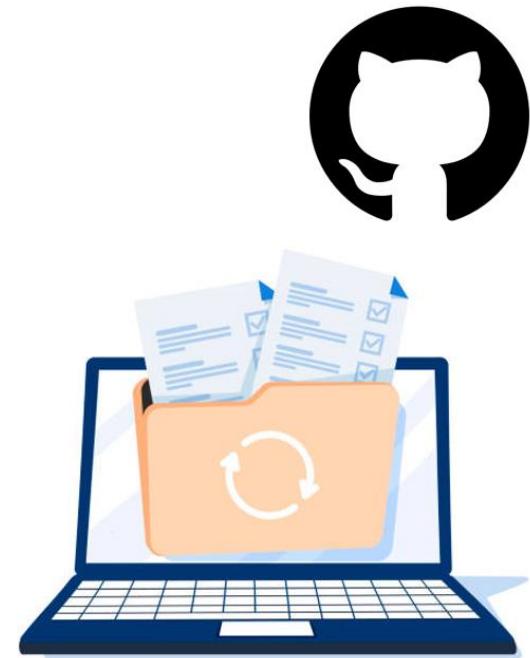
GitHub

Bajando archivos

Cuando lo que queres hacer no es clonar un repositorio, sino actualizar los archivos de uno (que ya tenes en tu computadora) el commando que deberas realizar es:

Git pull

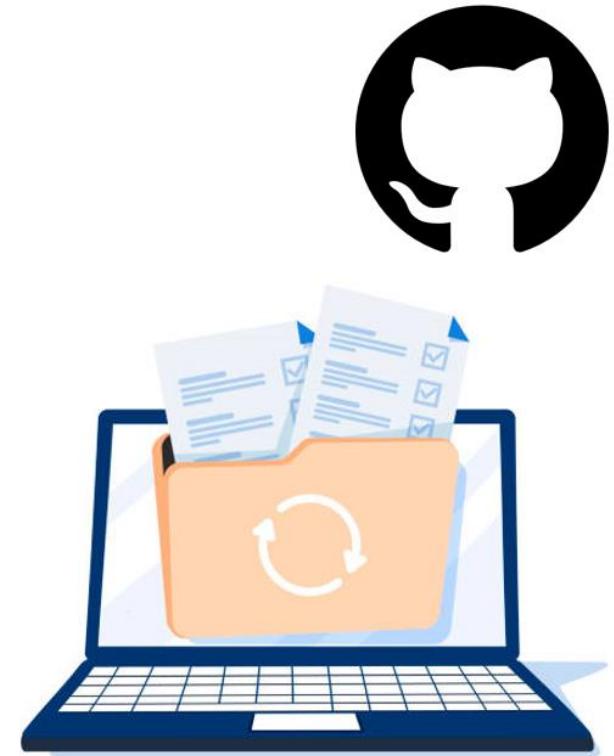
- *Pues el mismo no descargara nuevamente todos los archivos sino que solamente bajara y actualizara aquellos archivos que tuvieron algún tipo de modificación, así como también cualquier archivo nuevo existente.*





Bajando archivos

A git pull deberemos agregarle al igual que al comando anterior las palabras “**origin master**” ya que así indicamos que queremos traer desde el repo todos los archivos nuevos y con cambios en la rama principal (master)



Resumiendo: Bajando archivos

GitHub

Asegúrate de que tu repositorio local esté vinculado a GitHub:

- Primero, verifica que tu repositorio local ya esté conectado con GitHub, lo cual debiste hacer siguiendo los pasos anteriores para la conexión. Para revisar si tienes un origen remoto configurado, usa este comando:

git remote -v

- Si todo está bien, deberías ver algo como esto:

origin <https://github.com/tu-usuario/TuArchivo.git> (fetch)

origin <https://github.com/tu-usuario/TuArchivo.git> (push)

Actualizar tu repositorio local con los cambios del remoto:

- Si alguien más ha hecho cambios en el repositorio en GitHub y quieres descargarlos a tu repositorio local, usa el siguiente comando:

git pull origin master

Aquí, *origin* es el nombre del repositorio remoto (en este caso, tu repositorio en GitHub) y *master* es la rama principal de tu proyecto.

GitHub

Resolver conflictos (si los hay):

- Si tanto tú como otras personas hicieron cambios en los mismos archivos, podrías encontrarte con un conflicto de fusión (**merge conflict**).
- No te preocupes, Git te avisará si esto ocurre y te permitirá resolver los conflictos.
- Si hay un conflicto, Git te mostrará el archivo donde está el problema y te pedirá que elijas qué cambios mantener.
- Podrás ver los conflictos en el archivo con marcadores como:

```
<<<<< HEAD
```

```
# Tu cambio
```

```
=====
```

```
# Cambio hecho en GitHub
```

```
>>>>> nombre-de-la-rama
```

- *Elimina los marcadores <<<<<, =====, >>>>> y edita el archivo para resolver el conflicto. Luego, guarda el archivo y realiza un nuevo commit.*

Finalizar la actualización:

- Una vez que los cambios se han descargado correctamente (y los conflictos, si los hubo, han sido resueltos), tu repositorio local estará actualizado con los cambios del repositorio remoto.
- Siempre es una **buena práctica** hacer un pull antes de empezar a trabajar en nuevos cambios para asegurarte de que tienes la versión más actualizada del código.

Contenido Extra sobre GitHub

GitHub

Todos los comandos

Acción	Comando	Descripción
Inicializar un repositorio local	<code>git init</code>	Inicializa un repositorio de Git en la carpeta actual.
Ver el estado del repositorio	<code>git status</code>	Muestra el estado actual de los archivos en tu repositorio (cambios no guardados, etc.).
Agregar archivos al área de preparación (staging)	<code>git add .</code>	Agrega todos los archivos al área de preparación para el próximo commit.
Hacer un commit	<code>git commit -m "Mensaje"</code>	Guarda los cambios agregados al área de preparación con un mensaje descriptivo.
Ver el historial de commits	<code>git log</code>	Muestra el historial de commits del repositorio.
Ver los repositorios remotos vinculados	<code>git remote -v</code>	Lista las URL de los repositorios remotos vinculados al repositorio local.
Vincular un repositorio remoto	<code>git remote add origin https://github.com/tu-usuario/repositorio.git</code>	Conecta tu repositorio local a uno remoto en GitHub.
Subir cambios al repositorio remoto	<code>git push -u origin master</code>	Sube los commits del repositorio local a la rama principal (master) en GitHub.
Descargar cambios del repositorio remoto	<code>git pull origin master</code>	Descarga y fusiona los cambios del repositorio remoto a la rama master.
Ver diferencias entre archivos	<code>git diff</code>	Muestra los cambios que se han hecho en los archivos pero que aún no se han guardado (commits).
Ver las ramas del repositorio	<code>git branch</code>	Lista todas las ramas locales en el repositorio.
Crear una nueva rama	<code>git branch nombre-rama</code>	Crea una nueva rama en tu repositorio local.
Cambiar a una rama existente	<code>git checkout nombre-rama</code>	Cambia de la rama actual a otra rama existente en el repositorio.
Resolver conflictos de fusión	<code>Editar archivos afectados y luego git add . y git commit -m "Resuelto"</code>	Resuelve los conflictos manualmente y luego haz commit de los cambios.

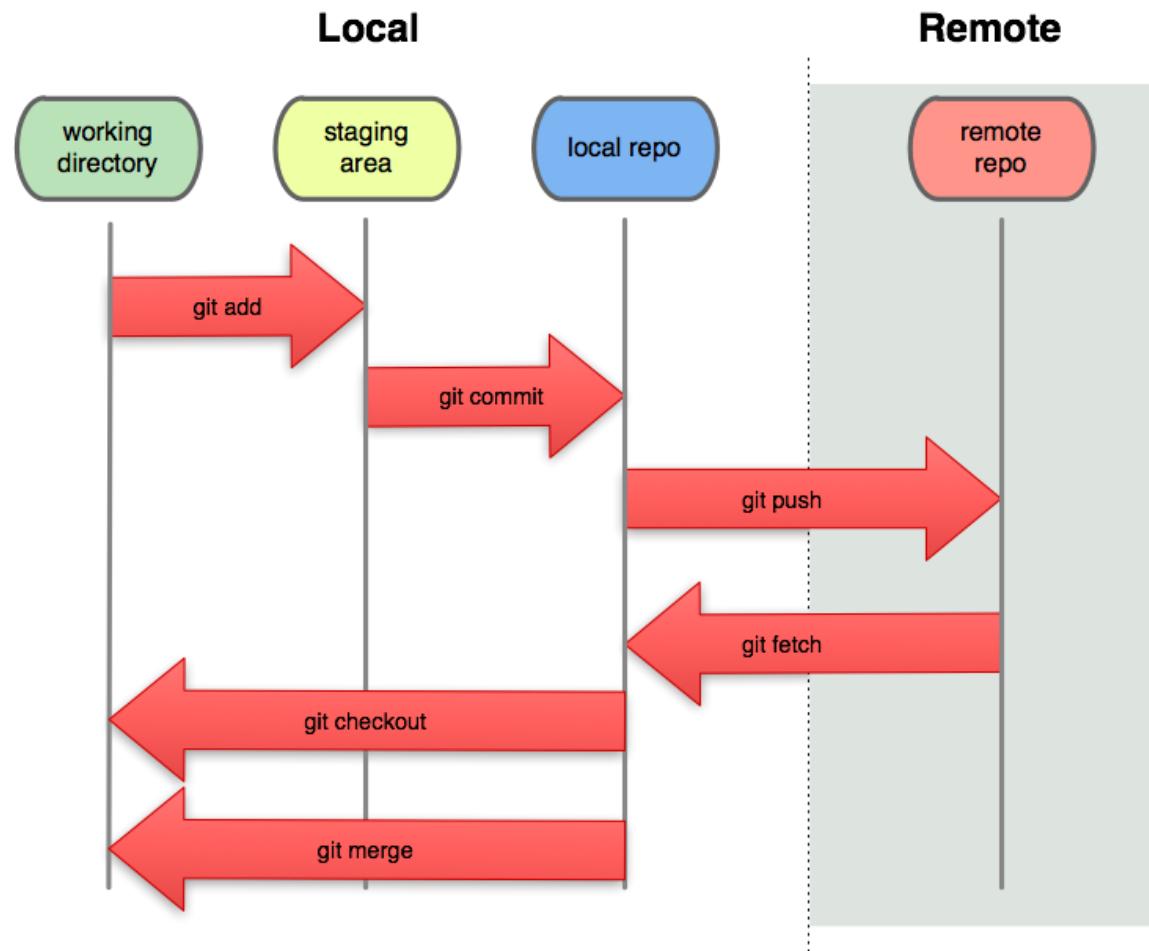
Git

En Git el ciclo de vida del archivo es:

1. Creación del archivo, por ejemplo index.html (aquí el archivo está en estado untracked).
2. Cambiamos el estado del archivo a seguimiento a través del comando **git add** index.html (especificando nombre del archivo index.html para este caso). Si el archivo ya no será modificado, pasar al paso 6.
3. Si el archivo es modificado, deja de estar en seguimiento y pasa a un estado de modificado, ya que Git detecta que hubo cambios.
4. Guardamos el archivo con **ctrl+g** (o **ctrl + s**) y luego escribimos **git add** index.html (especificando nombre del archivo index.html para este caso), el archivo pasa al stage area.
5. El archivo está listo para realizar el **commit**, que es cuando se genera el punto histórico, a través del comando **git commit**.
6. Para especificar que git no realice más seguimiento a un archivo, usamos el comando **git rm --cached** donde si bien el archivo como tal no se ha eliminado, Git procede a ignorarlo y pasa a un estado de untracked.

Git

¿Qué es?





Momento de
poner a prueba
lo aprendido!