# GIT HUB

# git clone/fork (Clonando repositorios remotos)

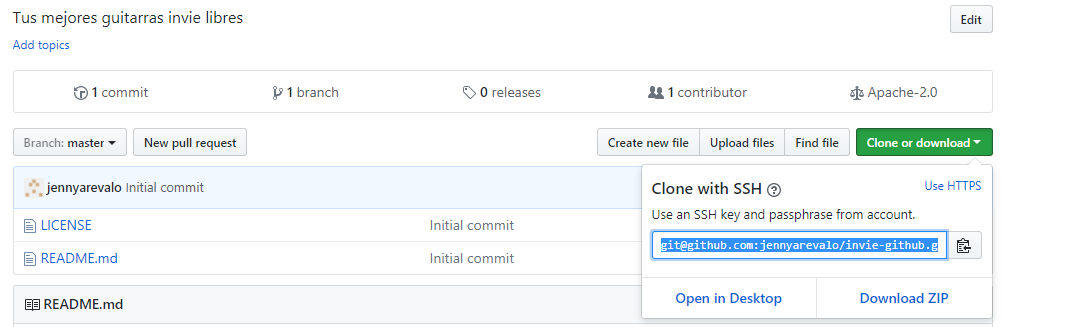
Vamos a proceder a crear nuestro primer proyecto en **GitHub.**

* Nuevo repositorio
* Asigna un nombre a tu repositorio
* Asigna una descripción a tu repositorio
* Puedes adicionar un ““read me””
* Crea el repositorio

En la parte derecha encuentras una opción que se llama clone or download si no es nuestra intención compartirlo.  
Un fork va a clonar el proyecto y nos va a crear un nuevo proyecto en el **GitHub.**

## **clone**

El comando git clone, nos permite obtener una copia, de un proyecto existente, en algunas de las plataformas de alojamiento de repositorios distribuidos ( github, gitlab, bitbucket…) trayendo desde la web, a nuestro disco duro, la replica de un repositorio ya existente, bien sea para estudiar su estructura, colaborar en el proyecto, entre otras cosas.

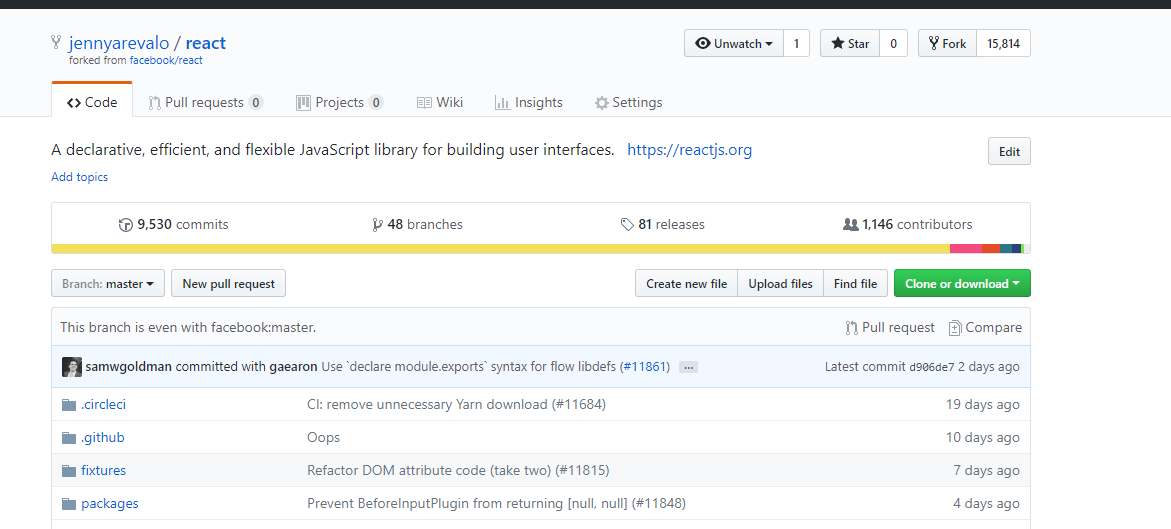






## Fork

el \*\***fork** \*\*en github nos hace una copia del proyecto en nuestro perfil/repositorios para poder hacerle lo que queramos sin afectar el proyecto original



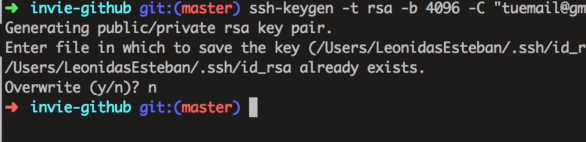
# Añadiendo una llave ssh a GitHub

Una vez configuremos bien git con el nombre de usuario y email que sea el mismo de github

Establecer una comunicación entre github y mi entorno local para que cada vez que quiera hacer un cambio y enviarlo hasta git hub a un lugar remoto para eso generamos un **llave ssh**.

## Generar una Llave ssh

Crear llave  
ssh-keygen -t rsa -b 4096 -C "Email de github"

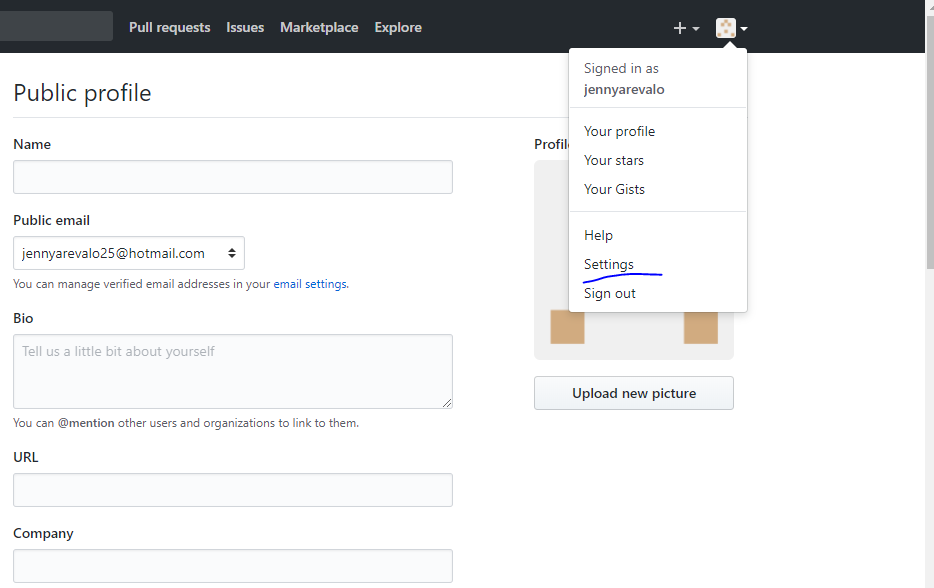
4

# Copiar llave

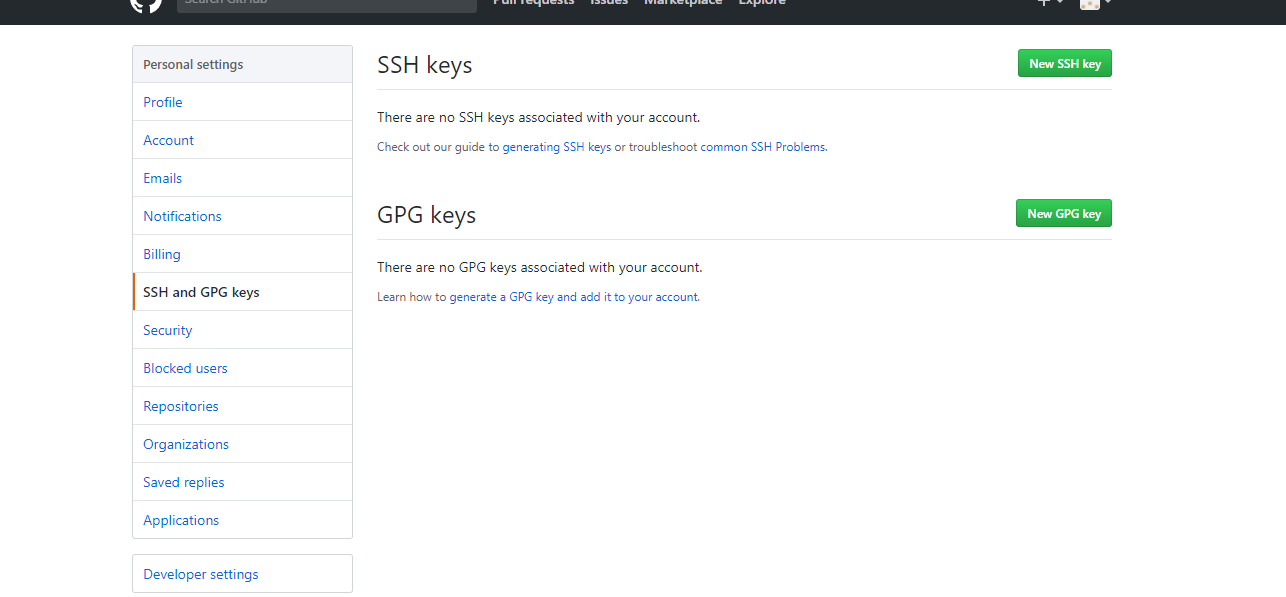
Con esto vamos a copiar el contenido de lo que tenemos un nuestra llave publica que se encuentra en esta ubicacion

pbcopy < ~/.ssh/id\_rsa.pub

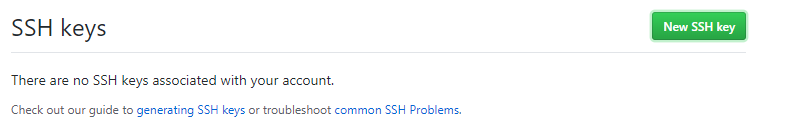
-Luego ir a



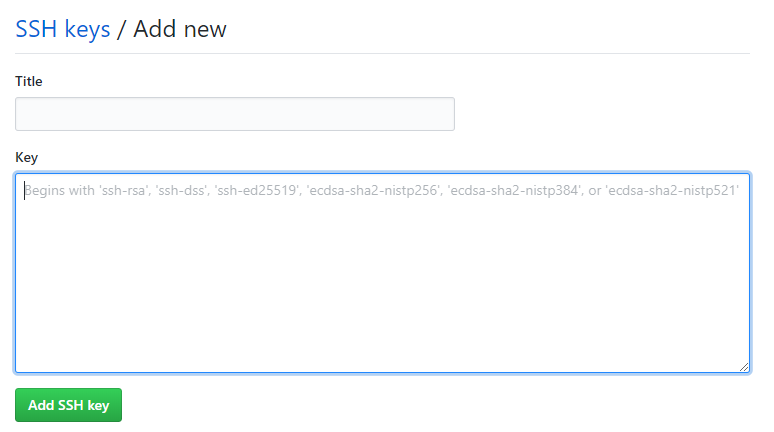
-Luego dar click en



-Dar click sobre



-Poner llave



# git remote (Añadiendo un repositorio remoto a uno local)

Tenemos que crear una conexión entre tu repositorio local y tu repositorio remoto para que puedas compartirlo con la comunidad. Para esto vamos a usar un nuevo comando que en este caso es **git remote.** Por convención lo nombramos “origin”. Para borrarlo podemos usar **git remote remove** y el nombre del repositorio.

**git remote add [origin] [SSH/HTTPS] : funciona para añadir un repositorio remoto a nuestro repositorio local.**

## Lista las conexiones existentes.

git remote -v

## Elimina una conexión con algún repositorio.

git remote remove [origin]

# git pull/fetch (Trayendo cambios desde el repositorio remoto)

Vamos a compartir nuestro código con el repositorio remoto. Antes de poder enviar cambios tengo que tener los mismos cambios en el repositorio remoto que en el repositorio local.  
Para traer los cambios del repositorios remoto vamos a usar el comando **git fetch.** Este comando recibe dos parámetros 1. desde donde y hacia donde van.  
Luego de traer los cambios, debemos mezclarlos con la rama que estamos usando en local. Para esto usamos git merge.

## Fetch

* Trae los cambios del repositorio remoto pero no los mezcla con el master del local

#Git fetch origin master

Donde **origin** es: el nombre del repositorio remoto

Y **master** es la rama que quiero traer de ese repositiorio

* Para mezclarlos con el master del repositorio local

#git merge origin/**master**

**si** obtienes un error usar la siguiente linea

# git merge origin/master --allow-unrelated-histories

* Otra manera de traer los cambios del repositorio remoto al local
* **git pull**: es como ejecutar los 2 pasos anteriores(fetch y merge) en uno solo

$ git pull origin **master**

# git push (Enviando cambios al repositorio remoto)

Para enviar nuestro código al repositorio remoto vamos a usar el comando git push. Así logramos que otras personas tengan acceso a nuestro proyecto.  
Cuando esto termine tendremos acceso a nuestro trabajo desde la plataforma de **GitHub.**  
Lo primero que debemos hacer es enviar nuestros tags en el proyecto.  
también recordemos que podemos eviar ramas, lo que también posdemos hacer a través del uso del comando git push [nombre de la rama].



Donde origin es el nombre del repositorio remoto

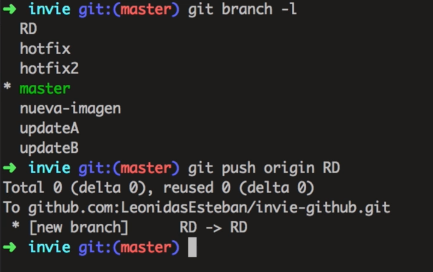
Y master es la rama del repositorio remoto

## Para enviar otras ramas o etiquetas a el repositorio remoto

* Enviar tags de la rama master al repositorio remoto

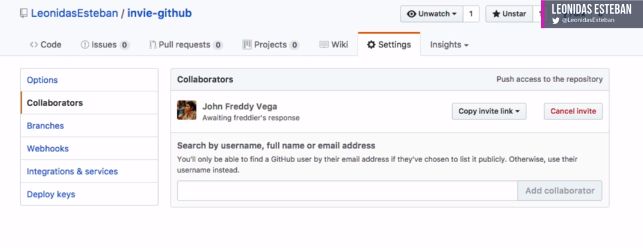


* Enviar otra rama

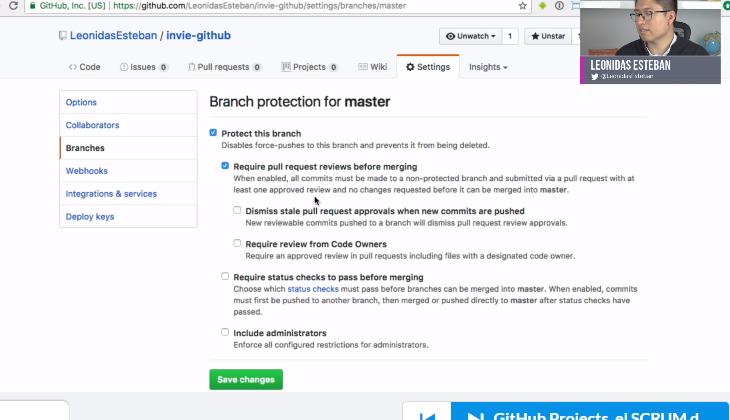


# Explorando la plataforma de GitHub

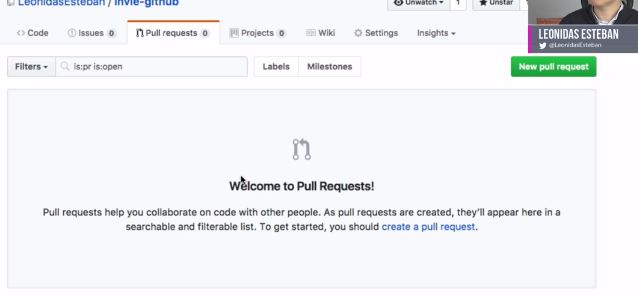
* Para agregar colaboradores a mi proyecto



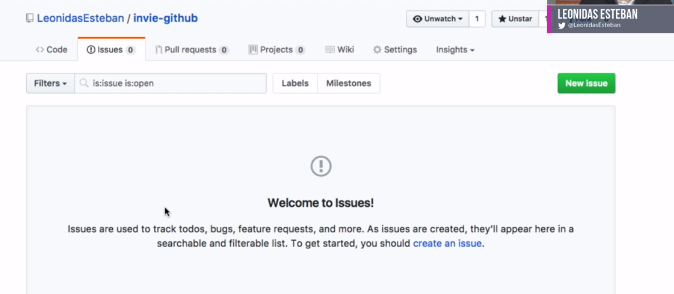
* Proteger la rama master



* Si la gente envía cambios llegan a

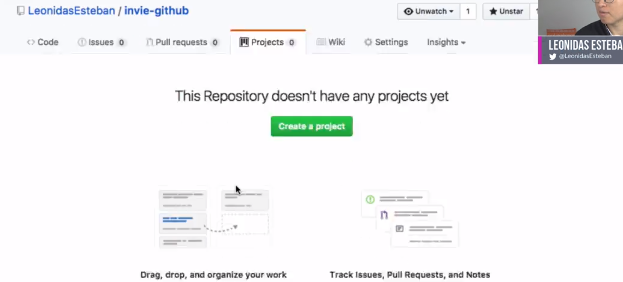


* Donde manejamos los problemas



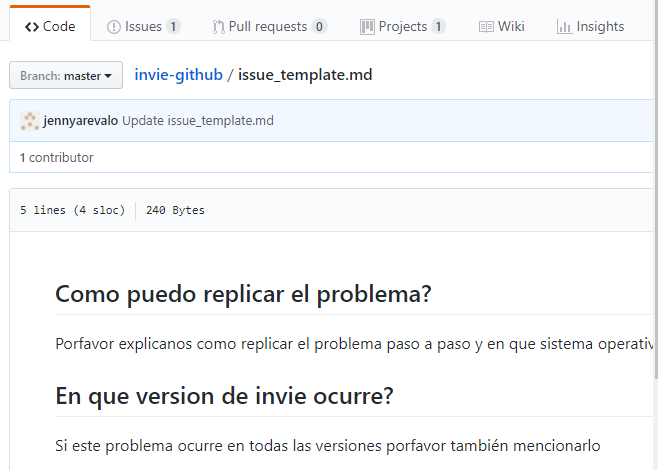
# GitHub Projects, el SCRUM de GitHub

**GitHub** tiene unas herramientas increíbles para nuestros proyectos. Una de estas herramientas es el **projects** de **GitHub,** la cual nos da la posibilidad de tener un mayor control del desarrollo de nuestro proyecto.  
Aquí como en la metodología de **SCRUM** tenemos la posibilidad de saber qué tenemos pendiente por hacer, cuáles son las features que se encuentran en desarrollo, identificar los bugs que emergen de nuestro proyecto, y los features que debemos revisar.

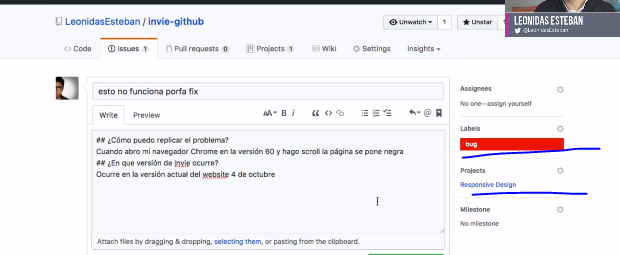
* 4 **TODO:** Son las cosas por hacer.
* **WIP:** Work in Progress. En que estamos trabajando.
* **Bugs:** Cosas que hay que arreglar de manera prioritaria.
* **Waiting for review:** nos sirve para cuando trabajamos con los pulls requests, ya que se espera que alguien revise nustro codigo.
* **Done:** Cuando la tarea ha finalizado.

# Creando un template para issues

Lo ideal es que siempre que creemos un proyecto tengamos un template para enviar pull requests o cuando tengamos issues.



* Crear un issue

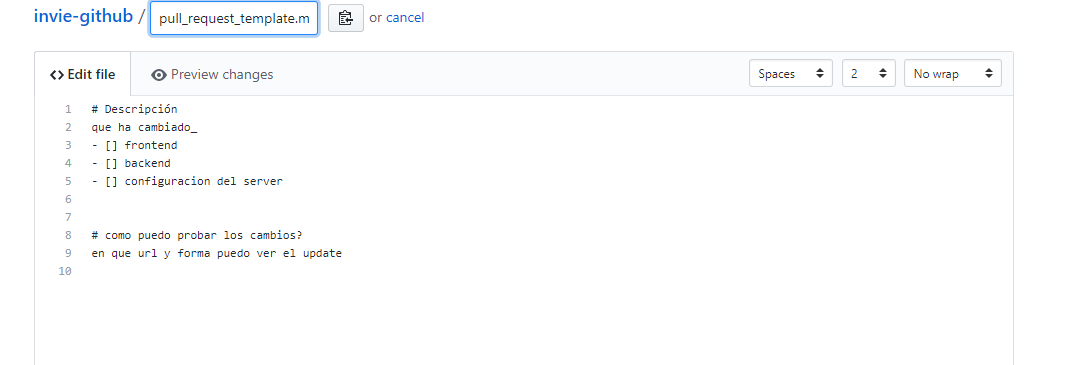


# Creando un template para Pull Request

GitHub permite usar templates a la hora de generar Pull Request

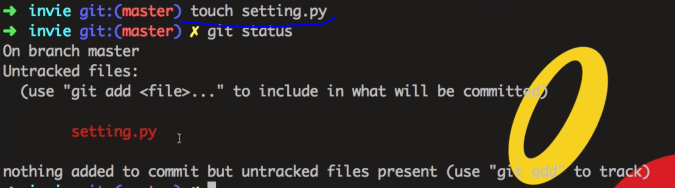
1. Crear archivo llamado **pull\_request\_template.md**
2. Definir los lineamientos usando sintaxis *markdown*
3. Hacer commit

Ahora todo será mas ordenado cuando generen contribuciones a tu proyecto.



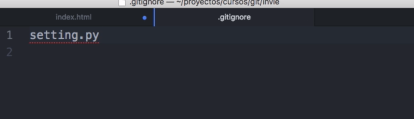
# .gitignore (Ignorando archivos no deseados)

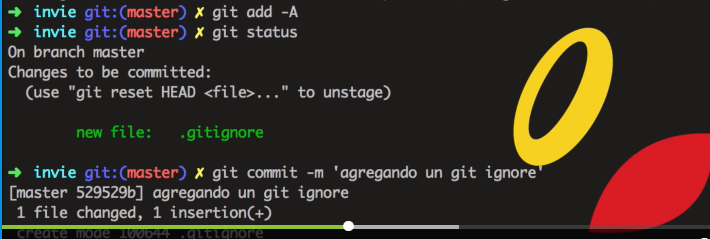
Si tienes archivos que no pueden ser públicos, como archivos de configuración con contraseñas, lo ideal es que no los subas a tu repositorio, estos archivos los puedes poner en el archivo .gitignore.



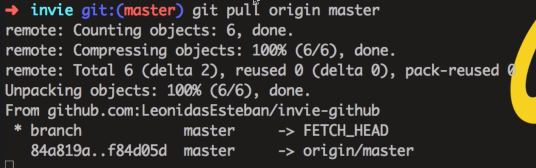
* Crear archivo .gitignore

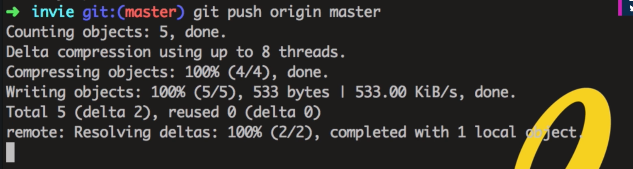


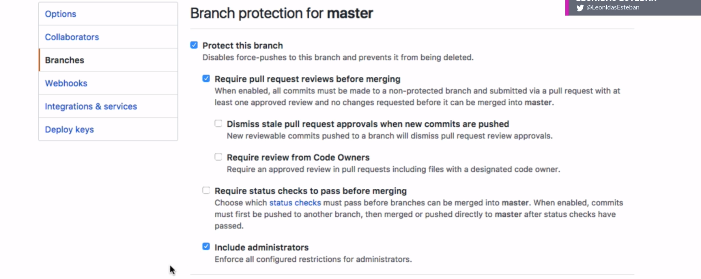




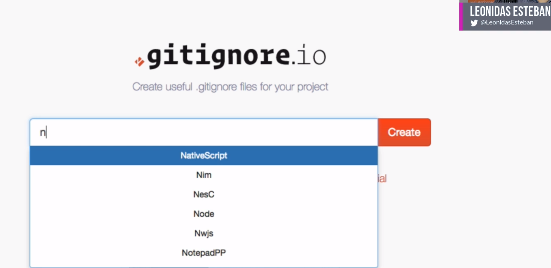


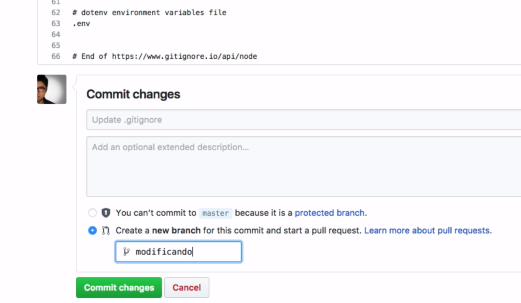






No deja enviar push a master

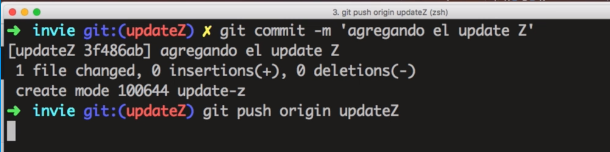




# Pull Request (Colabora a proyectos externos)

Un Pull Request es una solicitud para que el dueño del repositorio realice los cambios que estas proponiendo. Estos nunca se hacen a la rama master, para evitar inconvenientes.

Es la base de la colaboración Open Source en github

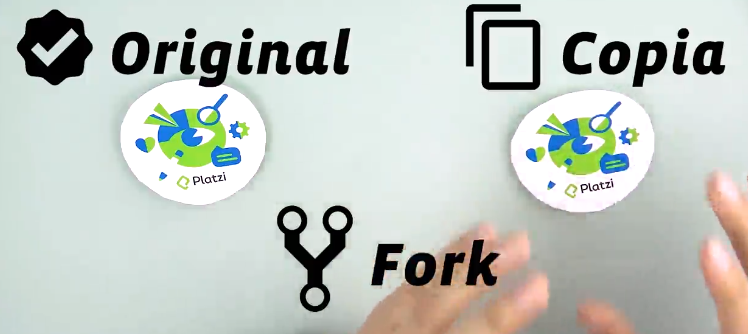


# Issues y Milestones (Reportando y monitoreando errores eficientemente)

**Issues**: Sirve para reportar un problema o sugerir algún cambio que para el repositorio.

Los issues no son problemas. son tareas que se pueden asignar a un participante del desarrollo.  
**Milestones**: Forma para agrupar Issues o Pull Request.

# ¿Cómo funciona un Pull Request en GitHub?

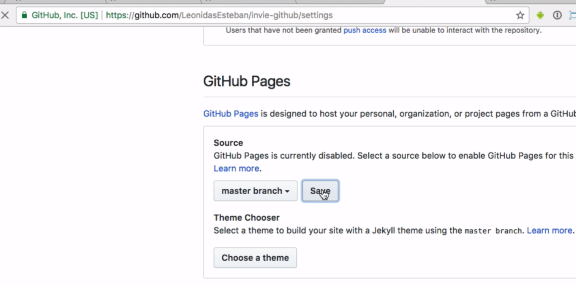






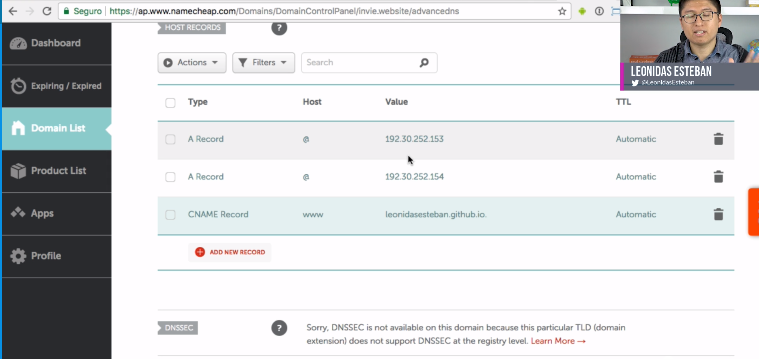
# GitHub Pages, hosting gratuito de archivos estáticos

**GitHub** nos permite publicar nuestros proyectos a tráves de **GitHub Pages.**  
Los **GitHubs** pages almacenan código estático.



# Dominios personalizados en Github

**GitHub** me permite personalizar la URL de mi **GitHub Pages**, primero debo comprar el dominio, nosotros usamos [namecheap.com](http://namecheap.com/).  
En el dashboard de namecheap vamos a manage nuestro dominio, vamos a configurar los DNS.  
En **GitHub** vamos a settings y creamos un custom URL.  
Creamos en el proyecto el archivo CNAME y en el ponemos la URL del dominio que compramos.  
La ip de GitHub es 192.168.252.153 - 192.168.252.154.



# GitHub Desktop, todo lo que necesitas de Git en una sola herramienta gráfica

Interfaz de escritorio en [desktop.github.com](http://desktop.github.com/) para que puedas trabajar gráficamente desde tu PC. Desarrollada en Electron. Lo primero que debes hacer es iniciar sesión, luego clonar el proyecto.

* Puedes ver todo el historial de tu proyecto de una manera muy amigable.
* Quien hace los cambios, quien los aprueba.  
  -También puedes navegar en todos tus repositorios.
* Puedes hacer cambios.

# Enviando invie a producción (invie.com)

# Desafío: Crea un repositorio público configurando un template de Issues y Pull Rquests

Crea tu propio repositorio con tus propios templates de Issues y Pull Request.