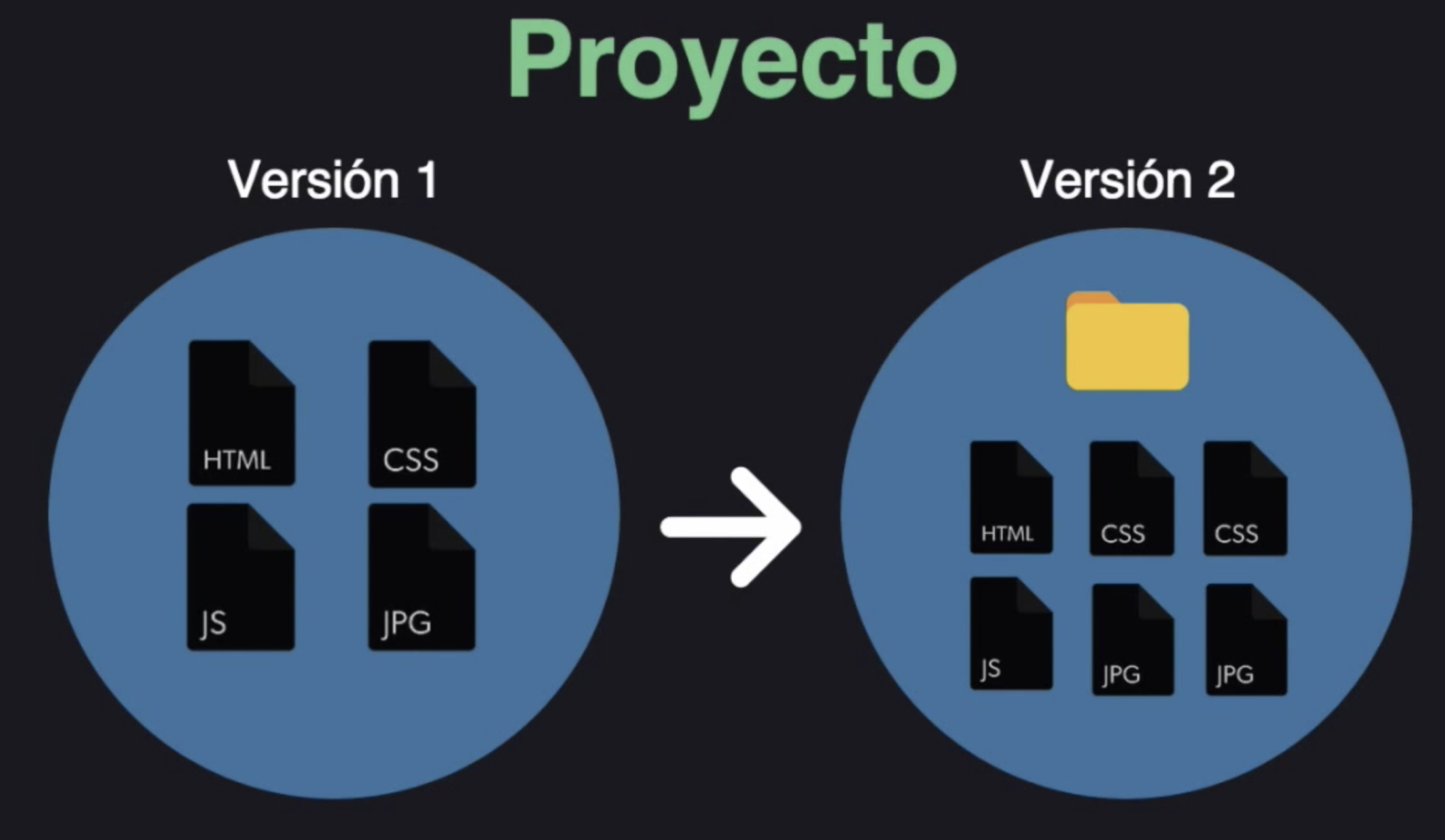
Clase 16 - Git y Github

GIT

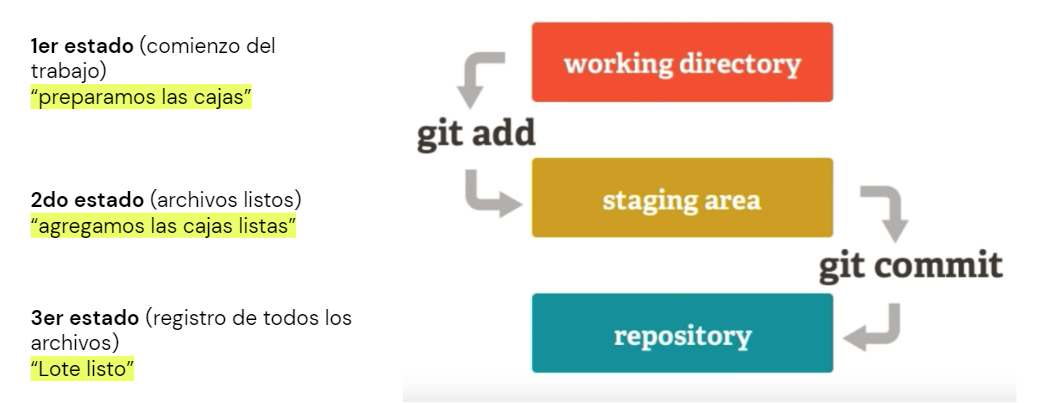
¿Qué es GIT?

Git es un sistema de control de versiones gratuito y de código abierto, diseñado para manejar desde pequeños a grandes proyectos de manera rápida y eficaz. Se entiende como control de versiones a todas las herramientas que nos permiten hacer modificaciones en nuestro proyecto. Un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo.

Con GIT, podemos ir a versiones anteriores, muy útil para errores y para la organización.



GIT - Los 3 estados



Git es uno de los sistemas de control de versiones más populares entre los desarrolladores. Y parte de su popularidad se la debe a GitHub, un excelente servicio de alojamiento de repositorios de software con este sistema.

Instalar GIT en Windows

Empecemos con GIT

Buscar en su menú el Git Bash para abrir la terminal e iniciar con los comandos.

Usar para slides de texto e imagen. Si no alcanza, no sobrecargar, usar otra con el mismo título para indicar que continúa el mismo módulo.

Verificando versión Git

Escribir git --version y presionar “Enter”.

|  |
| --- |
| john@MyShopSolutions: ~$ git --version  git version 2.17.1  john@MyShopSolutions: ~$ |

Configurando Git por primera vez: tu identidad

Lo primero que deberías hacer cuando instalas Git es establecer tu nombre de usuario y dirección de correo electrónico. Esto es importante porque las confirmaciones de cambios (commits) en Git usan esta información, y es introducida de manera inmutable en los commits que envías.

Configurando Git por primera vez

* Elegir un nombre de usuario que recuerdes fácil, y el email que en la próxima clase se usará en Github.
* Establecer el nombre con el comando: git config --global user.name "Nombre Apellido"
* Establecer el correo a usar con el comando. git config --global user.email johndoe@example.com

Configurando Git por primera vez

Comprobamos los pasos en nuestra consola.

|  |
| --- |
| /\* Paso 2\*/  john@MyShopSolutions: ~$ git config --global user.name "John Doe"  /\* Paso 3\*/  john@MyShopSolutions:**~$** git config --global user.email johndoe@example.com |

Vamos a comprobar si guardamos bien el usuario usando el comando:

git config --list.

|  |
| --- |
| john@MyShopSolutions: ~$ git config --list  /\* Se puede ver el usuario, el email y otros parámetros que dependerán de cada sistema operativo \*/  user.name=John Doe  user.email=johndoe@example.com  color.status=auto  color.branch=auto  color.interactive=auto  color.diff=auto |

Comprobando tu configuración

Puedes también comprobar qué valor tiene la clave nombre en Git ejecutando: git config user.name.

Puedes consultar de la misma manera user.email.

|  |
| --- |
| john@MyShopSolutions: ~$ git config user.name  John Doe |

Obteniendo ayuda

Si alguna vez necesitas ayuda usando Git, hay tres formas de ver la página del manual (manpage) para cualquier comando de Git:

|  |
| --- |
| /\*Los tres comandos que disparan la ayuda de Git\*/  john@MyShopSolutions: ~$ git help config  john@MyShopSolutions: ~$ git config --help  john@MyShopSolutions: ~$ man git-config |

Hasta el momento hemos aprendido los primeros pasos en GIT. Tenemos funcionando en el sistema una versión de Git configurada con tu identidad. Es el momento de aprender algunos fundamentos de Git.

Creando repositorios

¿Qué es un repositorio?

Un repositorio es un espacio centralizado donde se almacena, organiza, mantiene y difunde información.

Será “la carpeta” donde guardaremos nuestro proyecto para más adelante compartirlo con el equipo a través de un repositorio en la nube (en internet, por ejemplo en Github).

Git Init

Este comando se usa para crear un nuevo repositorio en Git. Nos crea un repositorio de manera local y lo hará en la carpeta donde estamos posicionados. También se le puede pasar [nombre\_de\_la\_carpeta] y creará una con ese nombre.

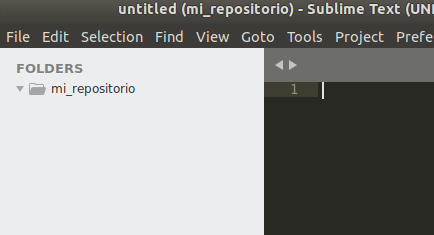
A continuación vemos el ejemplo:

|  |
| --- |
| /\* Paso 1: Me ubico en la carpeta donde quiero crear mi proyecto \*/  john@MyShopSolutions :~$ cd Documents/Proyectos\_Coder/  /\* Paso 2: Ya dentro de la carpeta inicio el proyecto con el nombre que le asigne a mi repositorio\*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/$ git init mi\_repositorio  /\* Arrojará el siguiente mensaje \*/  Initialized empty Git repository in /home/usuario/Documents/Proyectos\_Coder/mi\_repositorio/.git/  /\* Paso 3: Comprobamos que el repositorio se creó \*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/$ dir  mi\_repositorio  /\* Paso 4: Me ubico en mi repositorio \*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/$ cd mi\_repositorio  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ |

Git Status

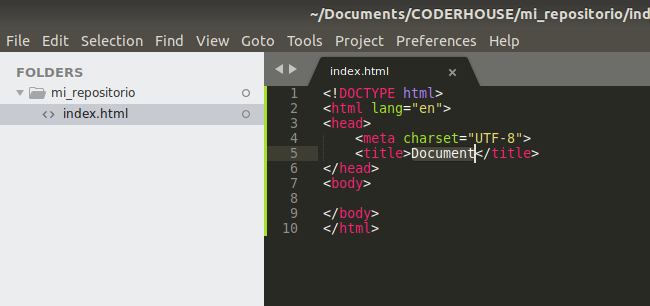
Ya hemos visto cómo inicializar un repositorio localmente utilizando git init. Ahora nos toca crear los archivos que vamos a usar en este repositorio.

1. Vamos a Sublime Text:



Buscamos el repositorio creado

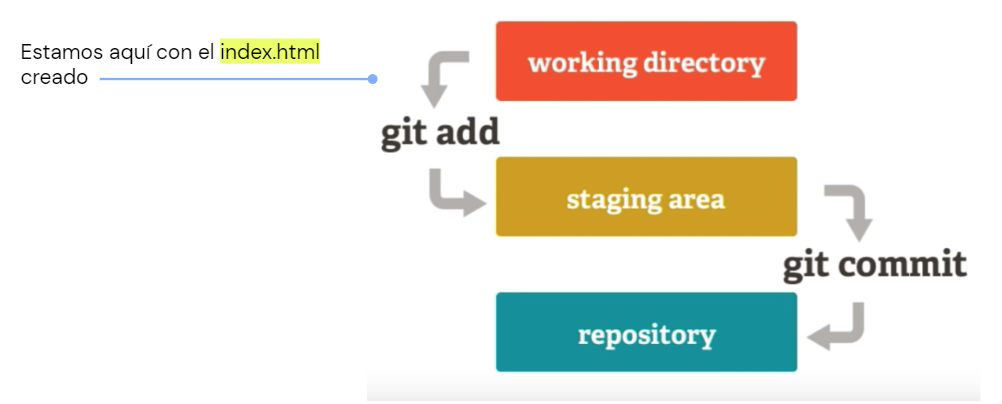
1. Creamos un archivo index.html que se guardará en el repositorio.



1. Vamos a la terminal y con git status chequeamos el estado de nuestro repositorio

|  |
| --- |
| john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git status  On branch master  No commits yet  Untracked files:  (use "git add <file>..." to include in what will be committed)  index.html  nothing added to commit but untracked files present (use "git add" to track) |

GIT: ¿recuerdan los 3 estados?



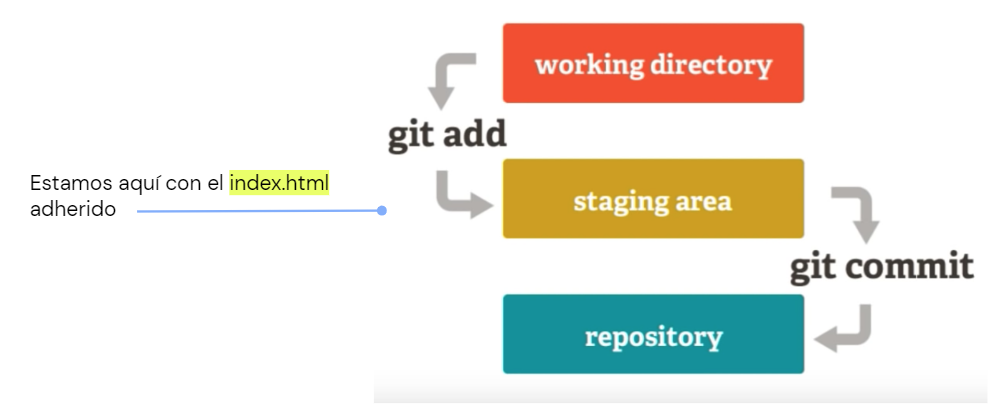
Git Add

Ahora se necesita agregar el o los archivos al Staging Area. En nuestro caso, para el index.html vamos a usar el comando git add + el nombre del archivo, lo cual permite adherir el archivo para subirlo luego al repositorio. También se puede usar git add . que adhiere todos los archivos nuevos.

Para verificar si funciono, nuevamente utilizamos git status.

|  |
| --- |
| john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git add index.html  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git status On branch master  No commits yet  Changes to be committed:  (use "git rm --cached <file>..." to unstage)  new file: index.html |

GIT: ¿recuerdan los 3 estados?



Git Commit

Una vez que nuestros archivos están en el Staging Area debemos pasarlos a nuestro repositorio local y para eso debemos usar el git commit que es el comando que nos va a permitir comprometer nuestros archivos.

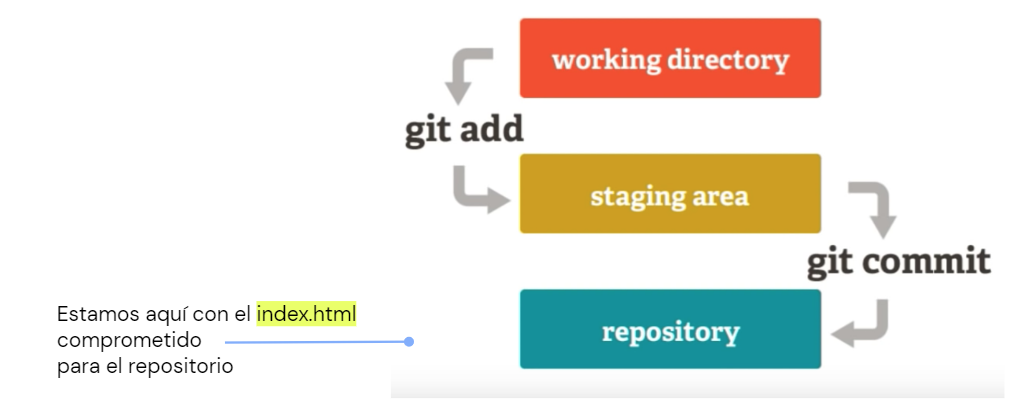
Es decir, que lo subirá al repositorio que se ha creado.

El comando es el siguiente:

git commit -m “Comentario de qué se trata el commit que se está realizando”

|  |
| --- |
| john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git commit -m "Primer archivo del repositorio"  /\* Esta sería el resultado del comando \*/  [master (root-commit) 1734915] nuevo archivo  1 file changed, 0 insertions(+), 0 deletions(-)  create mode 100644 index.html |

GIT: ¿recuerdan los 3 estados?



Git Log

Los primeros pasos a seguir:

|  |
| --- |
| /\* Con git log podemos ver los logs (historial) de lo que ha pasado en el repositorio \*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git log  commit 1734915470ce9983f703b77807a68e42166b47dd (HEAD -> master)  Author: John Doe <johndoe@example.com>  Date: Sat May 22 18:53:24 2020 -0300  Primer archivo del repositorio |

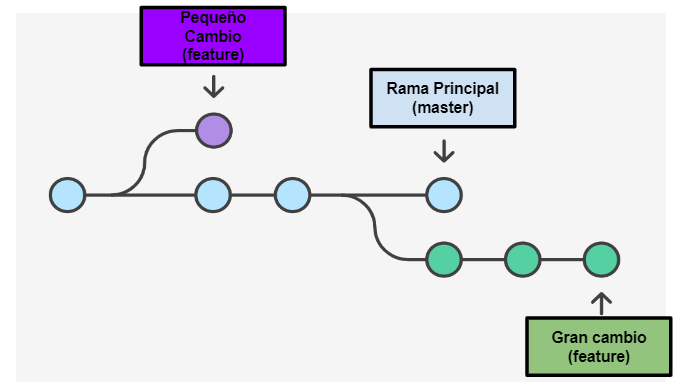
La documentación de git log es super extensa, por eso puedes indagar en el siguiente link [Git-Scm](https://git-scm.com/book/es/v1/Fundamentos-de-Git-Viendo-el-hist%C3%B3rico-de-confirmaciones)

Ramas

Para añadir una nueva función o solucionar un error (sin importar su tamaño), generas una nueva rama para alojar estos cambios. Esto te da la oportunidad de organizarte mejor con los cambios o correcciones experimentales.

Podemos crear una rama escribiendo

“git branch mi-rama”



Git Branch: creando ramas

Veamos cómo crear una rama.

|  |
| --- |
| /\* Paso 1: Verifico en cuál rama estoy \*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git branch  \*master  /\* Paso 2. Creo la rama que voy a usar para el cambio \*/ john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git branch mi\_rama  /\* Paso 3: Verifico que se creó la rama \*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git branch -l  \*master  mi\_rama  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ |

Git Branch: movernos entre ramas

¿Será muy complicado hacerlo?

|  |
| --- |
| /\* Para moverme a la rama que cree uso el comando de git checkout \*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git checkout mi\_rama  Switched to branch 'mi\_rama'  /\* Verifico nuevamente que me movi de rama \*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git branch -l  master  \*mi\_rama  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ |

Git Branch D: borrando ramas

Penúltimo paso:

|  |
| --- |
| /\* Paso 1: Me muevo a la rama principal “master” \*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git checkout master  /\* Paso 2: Verificar que se está en la rama de master \*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git branch  \*master  mi\_rama  /\* Paso 3: Procedo a borrar la rama que ya no voy a usar \*/ john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git branch -D mi\_rama  Deleted branch mi\_rama (was 6d6c28c)  /\* Paso 4: Verificar que se borró la rama\*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git branch  \*master |

Git checkouts: listar commits

Así como nos movemos entre ramas, nos podemos mover entre commits. Recuerden que al hacer cambios, adherirlos y comitearlos, se crea un historial de dichos cambios, los logs.

La posibilidad de volver a un commit en específico es una ventaja de los controladores de versiones, que permiten volver a un estado anterior si se presenta un problema, error o cambio inesperado.

Comenzamos listando.

|  |
| --- |
| /\* Para ver los commits realizados, los listamos con el comando git log --oneline para verlos en una sola línea\*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git log --oneline  /\* Se listan todos los cambios que se han realizado sobre el index.html \*/  fc59b88 (HEAD -> nueva\_rama) Ahora agregamos un título  6bcff19 Agregar un texto al index.html  41e6121 (master) Primer archivo del repositorio  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ |

Git checkout: mover a un commit

|  |
| --- |
| /\* Supongamos que me equivoqué en agregar el título, quiero volver al punto anterior del texto, busco el número de commit y muevo hacia ese punto \*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git checkout 6bcff19  Note: checking out 6bcff19.  You are in 'detached HEAD' state. You can look around, make experimental  changes and commit them, and you can discard any commits you make in this  state without impacting any branches by performing another checkout.  If you want to create a new branch to retain commits you create, you may  do so (now or later) by using -b with the checkout command again. Example:  git checkout -b <new-branch-name>  HEAD is now at 6bcff19... Agregar un texto al index.html |

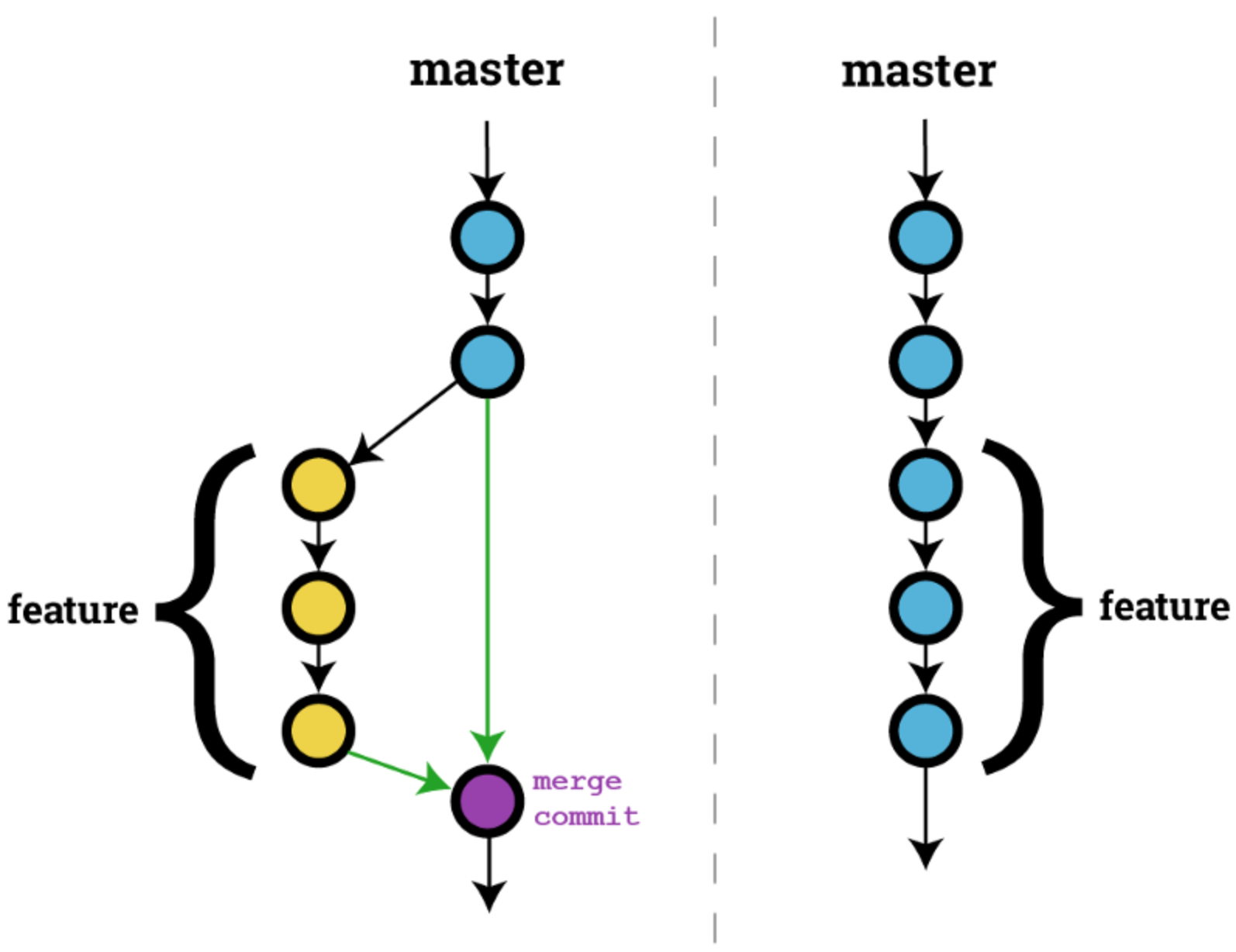
Git checkout: movernos a un commit

|  |
| --- |
| /\* Si verifico donde estoy parado co git branch se puede observar que se está en el commit y el index.html ha cambiado\*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git branch  \* (HEAD detached at 6bcff19)  master  nueva\_rama  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ |

Git: fusionar (merge)

Una vez que tenemos una rama (o más), podemos experimentar características nuevas.

Para luego FUSIONARLAS con la rama MASTER.



A continuación veamos cómo hacerlo…

Git Merge

|  |
| --- |
| /\* Paso 1: Ubicarse en la rama master, que es a donde quiero fusionar los cambios usando el comando de git checkout master. \*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git checkout master  /\* Paso 2: Verificar que estoy en master con git branch. Se puede observar en el archivo de index.html que no tiene ni título ni texto. \*/  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git branch  \*master  Nueva\_rama  /\* Paso 3: Realizar la fusión. Hacer el merge con el comando **git merge nueva\_rama\*/**  john@MyShopSolutions :~/Documents/Proyectos\_Coder/mi\_repositorio$ git merge nueva\_rama  Updating 41e6121..fc59b88  Fast-forward  index.html | 2 ++  1 file changed, 2 insertions(+) |

Repaso

* Git Init: indicarle que en ese directorio, donde ejecutamos este comando, será usado con GIT.
* Git Add: Agregar todos los archivos creados, modificados, eliminados al estado 2 (stage).
* Git Commit - m “mensaje”: mensaje obligatorio para mostrar que hemos cambiado, por ejemplo al estado 3.
* Git log -- online: para conocer los códigos de los commits realizados.
* Git checkout rama: para cambiar de rama e ir a un commit específico (debemos conocer su código anteriormente).
* Git merge rama: Debemos estar en un MASTER para funcionar.
* Git branch rama: creación de una rama (si queremos eliminar una rama ponemos git branch -D nombre-rama).

GitHub

¿Qué es GitHub?

Definición

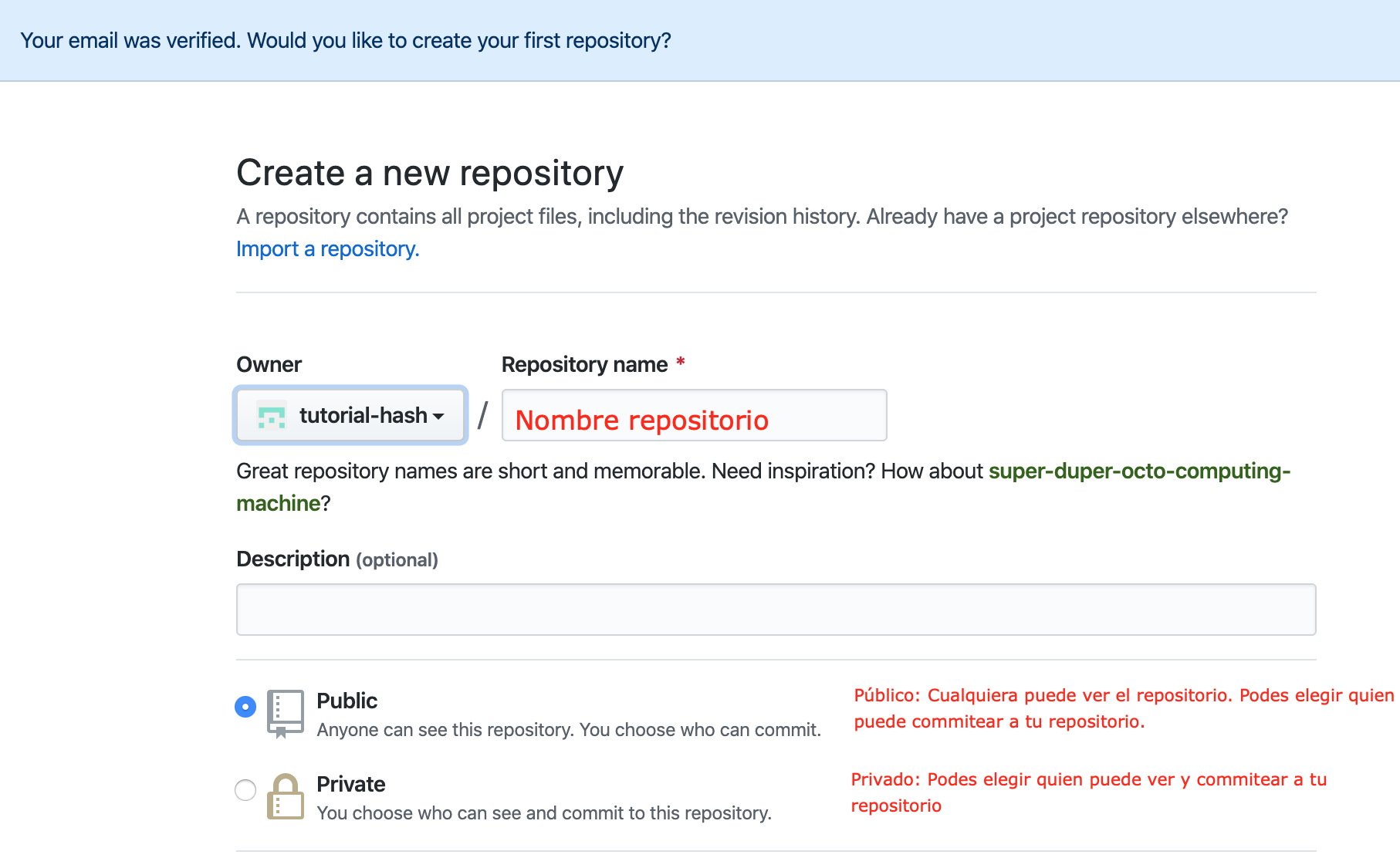
Por ahora todo lo que venía ocurriendo en Git era de manera local, no necesitábamos nada de internet para guardar nuestros commits y nuestro repositorio. Ahora queremos compartir nuestro trabajo con otros (compañeros de proyecto, clientes, etc). ¡Para eso utilizamos Github!

Github es una especie de “red social” de programadores. Con este sitio podemos subir nuestros proyectos y lograr que otras personas colaboren.

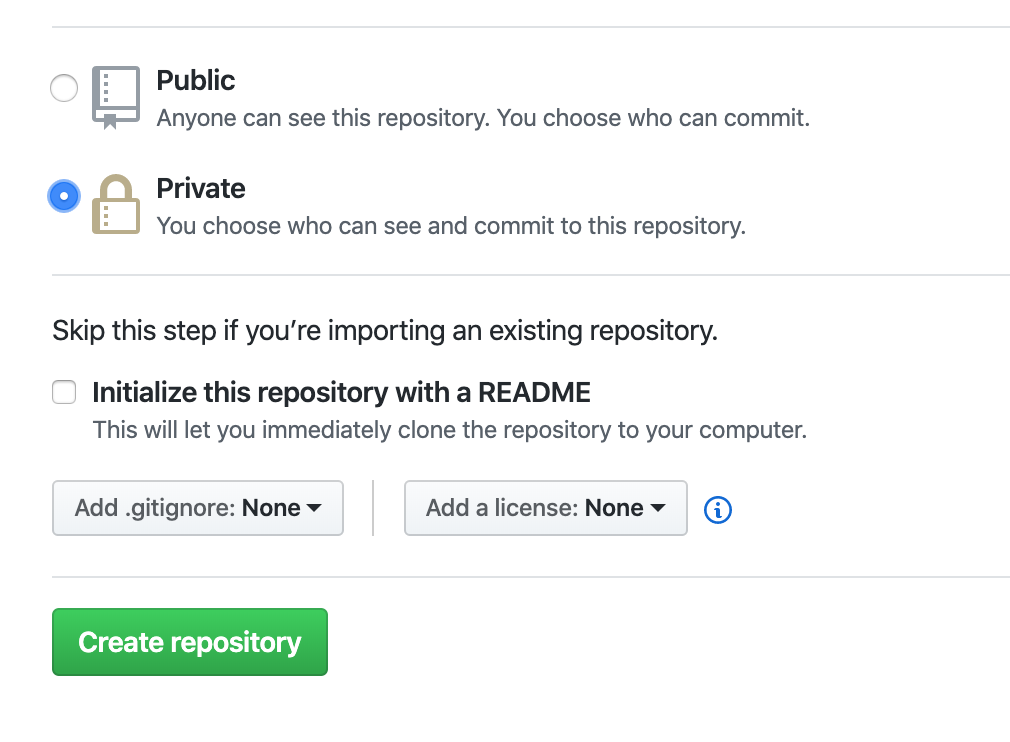
Crear nuestro repositorio en GitHub

Creando un repositorio

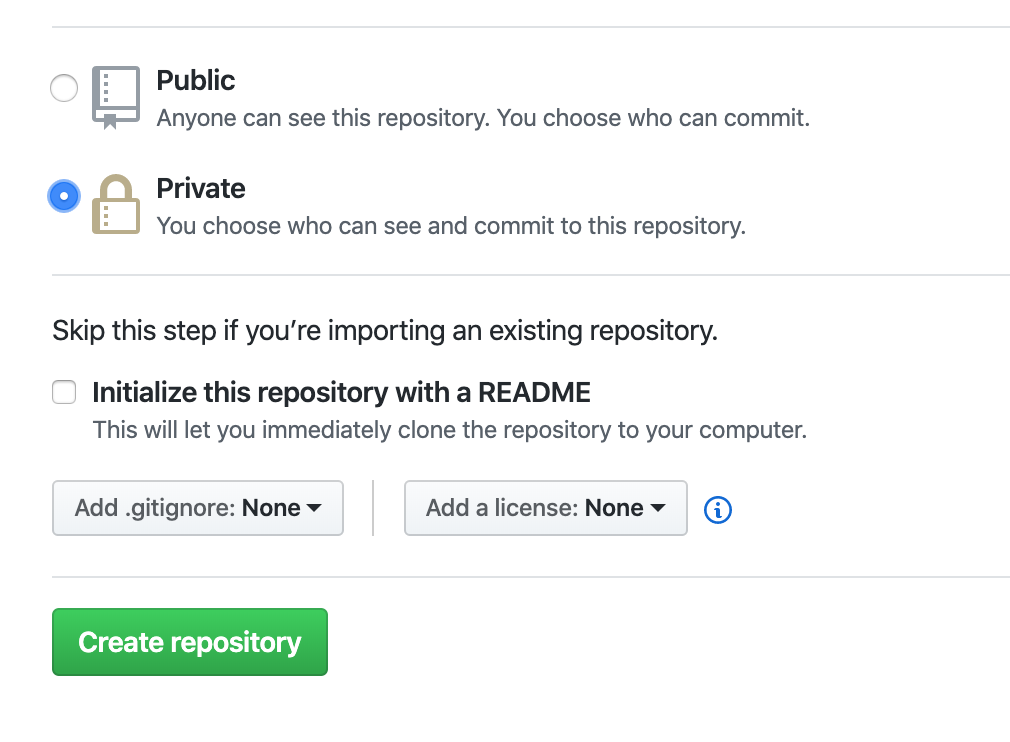
Luego de hacer clic en el enlace de verificación, aparecerá una pantalla así, que indica que tu e-mail ha sido verificado, y permite que hagas tu primer repositorio.



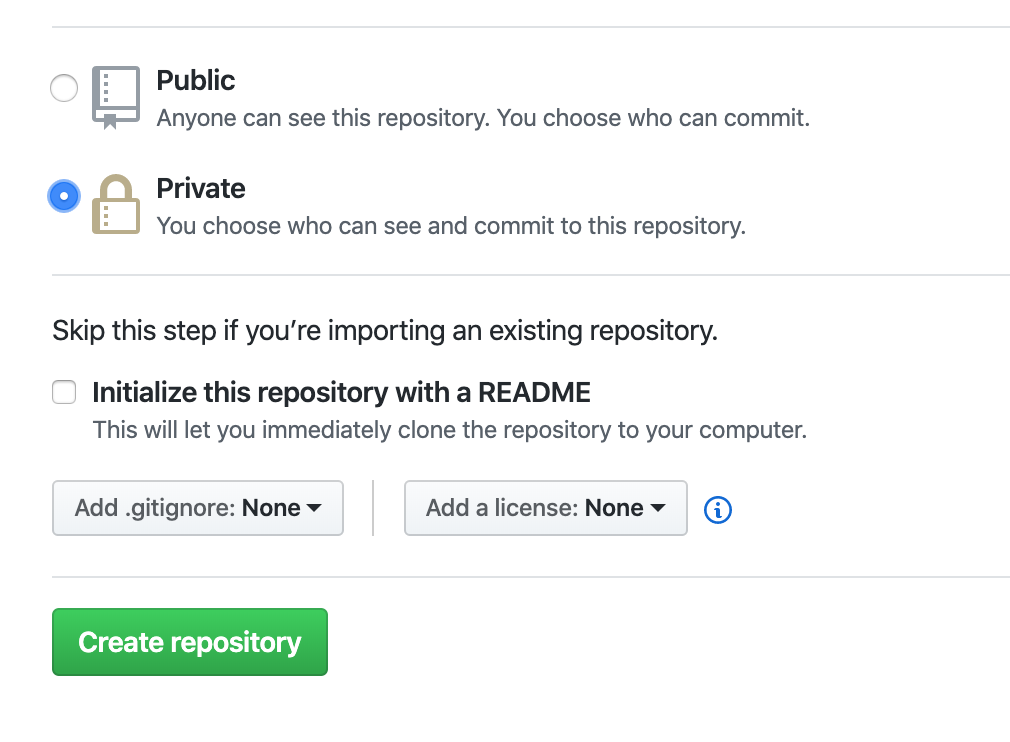
Por ejemplo, podría ser llamado “mi\_repositorio”, para que pruebes con los archivos que trabajaste en el desafío de GIT.



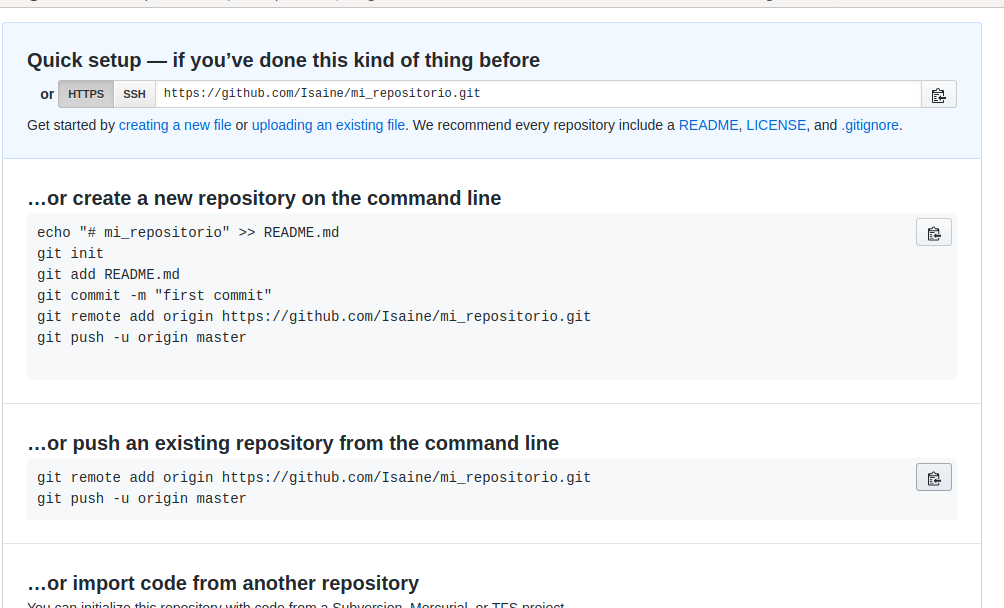
Elegimos “público” o “privado”. Si bien con privado limitamos el acceso a cualquier persona, no nos permitirá mostrar nuestro código como página web, por lo que elegimos “público”.



Luego hacemos clic en “Create repository”



Repositorio creado



¿Quieres saber más?

Te dejamos material ampliado de la clase

[Git Cheat Sheet](https://drive.google.com/file/d/10JdMNw82t9L62AaDGL__45mh1V7TTThL/view?usp=sharing) | GitHub Education

[Tutoriales de uso GitHub](https://skills.github.com/) | GitHub

[Cómo usar la integración Git en Visual Studio Code](https://www.digitalocean.com/community/tutorials/how-to-use-git-integration-in-visual-studio-code-es) | Digital Ocean