

P2.3

Trabajando con.. git y GitHub



git



GitHub

JUAN JORGE GALDO

1º DAW – 2020/21

CODE

Table of Contents

1.- Introducción.....	2
2.- Antes de empezar. ¿Qué es git y GitHub?.....	2
3.- Crear cuenta en GitHub.....	3
4.- Instalar y configurar git.....	4
5.- Configurar las claves SSH en GitHub.....	5
6.- Crear repositorio, clonarlo y modificarlo.....	7
7.- Conclusión.....	9

1.- Introducción

En este informe voy a reflejar los pasos seguidos para comenzar a utilizar tanto Git como GitHub en dos ordenadores diferentes. Uno es un mac y el otro es un viejo portatil al que le he instalado un linux.

2.- Antes de empezar. ¿Qué es git y GitHub?

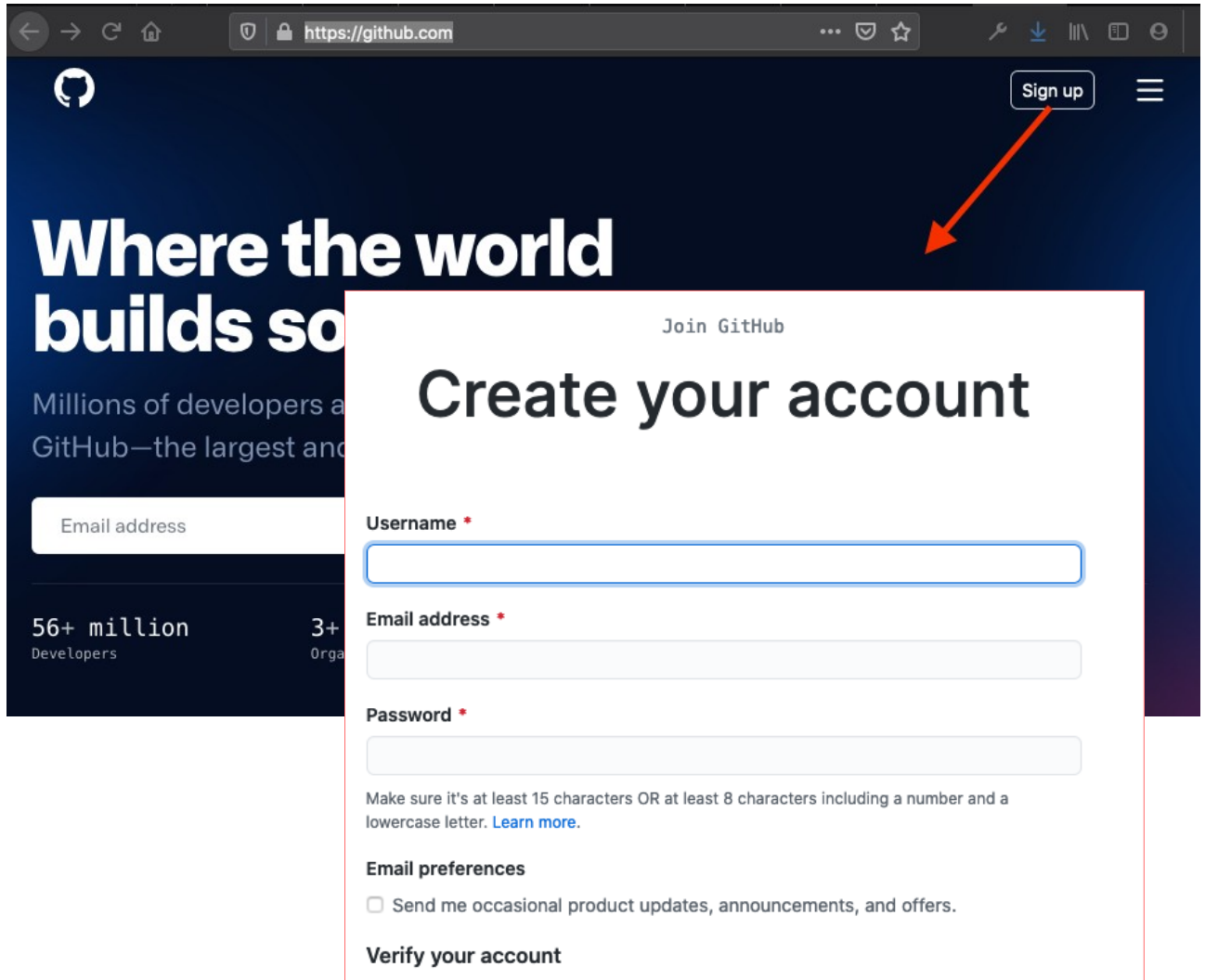
En el desarrollo de software es muy importante llevar un control de los cambios que se realizan al código (o a los ficheros de un proyecto) para, entre otras razones, asegurarse de que el equipo esté trabajando sobre la última versión, cerciorarse de que unos desarrolladores no están pisando el trabajo de otros, y para revertir cambios de la manera más sencilla posible en caso de que cualquier modificación del código derive en fallos del sistema.

Fruto de esas necesidades surgen diversos sistemas de control de versiones, y Git es uno de ellos, gratuito y de software libre. Git es un Sistema de Control de Versiones Distribuido (DVCS del Inglés) lo que significa que existe un repositorio central donde se guardan los archivos, pero además, todos los miembros del equipo tienen una copia exacta en su propia máquina, lo cuál garantiza que todos los integrantes estén trabajando con la misma información.

Por su parte, GitHub es una plataforma donde alojar los proyectos de desarrollo que emplea git como su sistema de control de versiones. También tiene un matiz de red social mediante el que desarrolladores de todo el mundo pueden contactar para solucionar dudas o colaborar en proyectos. Gracias a su gran popularidad se ha convertido en el mayor repositorio de código abierto del mundo. Además de las funcionalidades de control de versiones GitHub ofrece sus propias funcionalidades como pueden ser el control de acceso, bug tracking, la gestión de tareas asociadas a cada proyecto, asignación de prioridades, creación de documentación, etc. Existen diferentes tipos de perfiles de usuario (gratuito y de pago), pero las principales funcionalidades son gratuitas.

3.- Crear cuenta en GitHub

- Acceder a la web de GitHub (<https://github.com>) y pinchar en sign up



The screenshot shows the GitHub homepage with a dark blue header. In the top right corner, there is a 'Sign up' button. An orange arrow points from this button to a white sign-up form that is overlaid on the page. The form is titled 'Join GitHub' and 'Create your account'. It contains the following fields and sections:

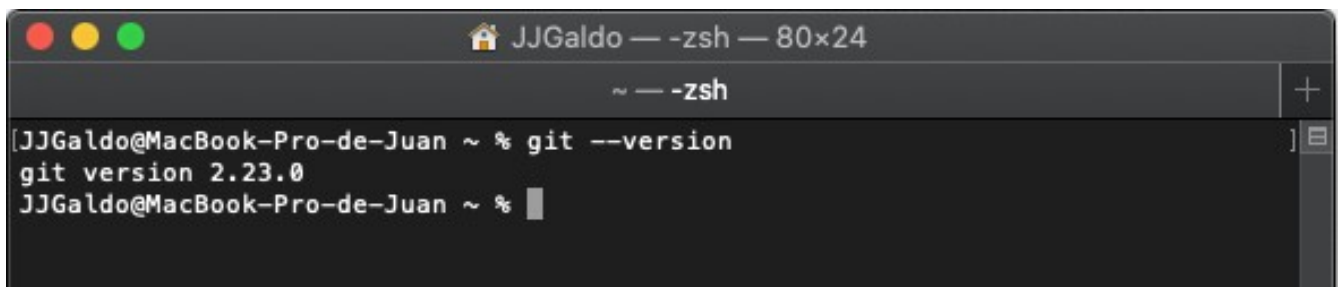
- Username ***: A text input field.
- Email address ***: A text input field.
- Password ***: A text input field.
- Password requirements**: A note stating 'Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)'
- Email preferences**: A checkbox labeled 'Send me occasional product updates, announcements, and offers.' which is currently unchecked.
- Verify your account**: A link at the bottom of the form.

- Rellenar los datos y contestar al mail de confirmación

4.- Instalar y configurar git

Instalación en Mac:

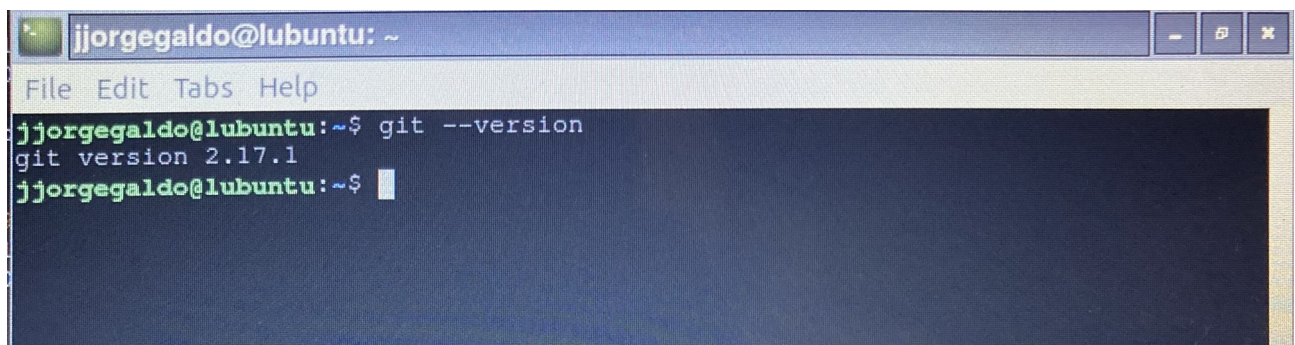
- abrir la terminal y escribir el comando: `$ git --version`
- Si no se tiene instalada se instalará después de confirmar un mensaje.
- Una vez finalizada la instalación, y sólo una vez, configuramos nuestro equipo en git para enviar información de propietario de cambios. Para ello:
 - `$ git config --global user.name "nombre_usuario"`
 - `$ git config --global user.mail "direccion@dominio.xx"`
 - `$ git config --global core.editor Nombre_editor_por_defecto`
 - De esta forma hemos configurado git con los datos que vamos a emplear en nuestro equipo cuando realicemos cambios a los ficheros. El último punto es para configurar el editor de texto que git abrirá por defecto a la hora de realizar commit o cualquier otra acción



```
JJGaldo — -zsh — 80x24
~ — -zsh
JJGaldo@MacBook-Pro-de-Juan ~ % git --version
git version 2.23.0
JJGaldo@MacBook-Pro-de-Juan ~ %
```

Instalación en Linux:

- Abrir la terminal y escribir el comando: `$ apt-get install git`
- Una vez finalizada la instalación realizamos los mismos pasos que en Mac



```
jjorgegaldo@lubuntu: ~
File Edit Tabs Help
jjorgegaldo@lubuntu:~$ git --version
git version 2.17.1
jjorgegaldo@lubuntu:~$
```

5.- Configurar las claves SSH en GitHub

Las claves SSH se utilizan para autorizar el acceso de equipos (ordenadores) a los repositorios, asegurando que es un usuario autorizado, pero sin la necesidad de estar solicitando el nombre de usuario y la contraseña en cada interacción que se haga con el repositorio central. Para ello generamos una clave única en el equipo desde el que trabajamos, y validamos dicha clave en nuestro usuario de GitHub. De esta manera, cuando nos conectemos desde el equipo con la clave dada de alta, el sistema no nos solicitará nuestros datos para poder interactuar con el repositorio del servidor. Podemos dar de alta tantas claves (equipos) como queramos.

1- Desde la Terminal: obtenemos la clave SSH y la incluimos en el SSH-Agent(donde lo guarda el equipo)

Desde el **Mac** se hace de la siguiente manera:

```

/Users/JJGaldo
JJGaldo@MacBook-Pro-de-Juan ~ % ssh-keygen -t ed25519 -C "juanjorgegaldo@gmail.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/Users/JJGaldo/.ssh/id_ed25519): Sin escribir nada le das a enter y guarda el archivo en la
/Users/JJGaldo/.ssh/id_ed25519 already exists. ruta que tiene por defecto
Overwrite (y/n)? y Yo tenia uno hecho, pero lo sebreescribi
Enter passphrase (empty for no passphrase): Yo puse contraseña, pero es opcional. Si no quieres, le das a enter sin escribir nada
Enter same passphrase again:
Your identification has been saved in /Users/JJGaldo/.ssh/id_ed25519.
Your public key has been saved in /Users/JJGaldo/.ssh/id_ed25519.pub.
The key fingerprint is:
SHA256:Ppd7M0/CcnPgPAiQYQ0zIlfv2oVU2AB5AUZ27I9Xr9E juanjorgegaldo@gmail.com
The key's randomart image is:
+---[ED25519 256]---+
|. oo@B+-.|
| o ++==..|
|. =0|
| oo.. .|
|.ooS. o|
| oooo =.E|
|. .oo=oB|
| ooX.+|
| o.B.|
+-----[SHA256]-----+
JJGaldo@MacBook-Pro-de-Juan ~ % ls -al ~/.ssh
total 16
drwx----- 4 JJGaldo staff 128 18 dic 12:03 .
drwxr-xr-x+ 60 JJGaldo staff 1920 18 dic 11:54 ..
-rw----- 1 JJGaldo staff 464 18 dic 12:05 id_ed25519
-rw-r--r-- 1 JJGaldo staff 106 18 dic 12:05 id_ed25519.pub

```

```

~/Documents/QuePasa/QuePasa - zsh
drwxr-xr-x+ 60 JJGaldo staff 1920 18 dic 11:54 ..
-rw----- 1 JJGaldo staff 464 18 dic 12:05 id_ed25519
-rw-r--r-- 1 JJGaldo staff 106 18 dic 12:05 id_ed25519.pub
JJGaldo@MacBook-Pro-de-Juan ~ % eval "$(ssh-agent -s)" Para saber que el SSH-Agent está corriendo en segundo plano. Muestra el número de proceso
Agent pid 31658
JJGaldo@MacBook-Pro-de-Juan ~ % open ~/.ssh/config Consulta si existe el archivo config.
The files /Users/JJGaldo/open and /Users/JJGaldo/.ssh/config do not exist.
JJGaldo@MacBook-Pro-de-Juan ~ % touch ~/.ssh/config Crea el archivo config
JJGaldo@MacBook-Pro-de-Juan ~ % open ~/.ssh/config
The file /Users/JJGaldo/open does not exist.
JJGaldo@MacBook-Pro-de-Juan ~ % open ~/.ssh/config Abre para editar el archivo config y le pegas lo que te aparece a continuación(respecta las líneas):
JJGaldo@MacBook-Pro-de-Juan ~ % open ~/.ssh/config Si no has puesto contraseña: UserKeychain no
JJGaldo@MacBook-Pro-de-Juan ~ % open ~/.ssh/config
The file /Users/JJGaldo/open does not exist.
JJGaldo@MacBook-Pro-de-Juan ~ % open ~/.ssh/config
JJGaldo@MacBook-Pro-de-Juan ~ % ssh-add -K ~/.ssh/id_ed25519 Añade la clave al ssh-agent
Enter passphrase for /Users/JJGaldo/.ssh/id_ed25519: Copia en portapapeles el contenido de la clave.
Identity added: /Users/JJGaldo/.ssh/id_ed25519 (juanjorgegaldo@gmail.com) Es lo que hay que pegar en la página de GitHub,
JJGaldo@MacBook-Pro-de-Juan ~ % pbcopy < ~/.ssh/id_ed25519.pub en el campo Key al pulsar New SSH key

```


Desde **Linux** se utilizan estos comandos (los pasos son iguales):

- Create new SSH Key:
 - `$ ssh-keygen -t ed25519 -C "your_email@example.com"`
- Presionar intro cuando pregunte por el archivo
- Introducir la contraseña (opcional)
- Arrancar el SSH-Agent: `$ eval "$(ssh-agent -s)"`
- Añadir la clave al SSH-Agent: `$ ssh-add ~/.ssh/id_ed25519`
- Copiar clave al portapapeles: `pbcopy < ~/.ssh/id_ed25519.pub`

2- Añadir SSH Key a nuestro usuario de GitHub

The image shows three screenshots of the GitHub web interface, illustrating the steps to add an SSH key to a user account.

Top Left Screenshot: Shows the user's profile menu on the left sidebar. The user is signed in as **JJorgeGaldo**. The **SSH and GPG keys** option is highlighted with a red circle.

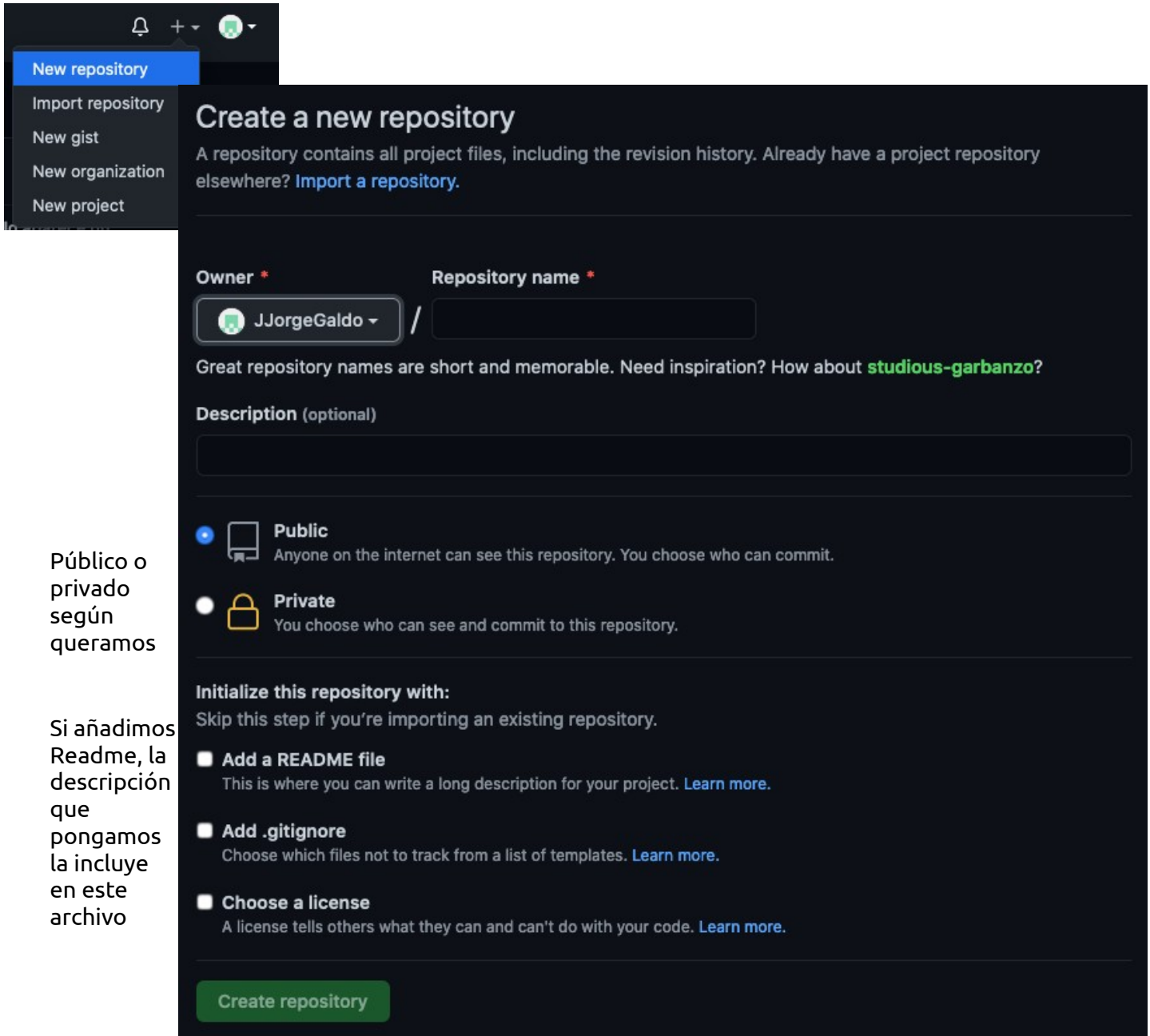
Top Right Screenshot: Shows the **SSH keys** page. A red arrow points to the **New SSH key** button. The page lists two existing SSH keys:

Key Name	SHA256 Fingerprint	Added On	Last Used	Permissions	Action
Portatil 2015	SHA256:Ppd7M0/CcnPgPAiQYQ0zI1fv2oVU2AB5AUZ27I9Xr9E	Added on 18 Dec 2020	Last used within the last week	Read/write	Delete
Portatil_2009_Lubuntu	SHA256:NUkz9ebI36FF0id8IwrfyDScTcYUmhelZC+APnbsL24	Added on 11 Jan 2021	Last used within the last week	Read/write	Delete

Bottom Screenshot: Shows the **SSH keys / Add new** form. The **Title** field contains the text: "Poner el nombre con el que queremos identificar al equipo". The **Key** field contains the text: "Pegar aquí la clave que habíamos copiado al portapapeles". The **Add SSH key** button is highlighted with a red circle, and a red arrow points to it with the text: "Clickar cuando esté todo cubierto".

6.- Crear repositorio, clonarlo y modificarlo


Crear repositorio en GitHub:



Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * **Repository name ***

 JJorgeGaldo /

Great repository names are short and memorable. Need inspiration? How about **studious-garbanzo**?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

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

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

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

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

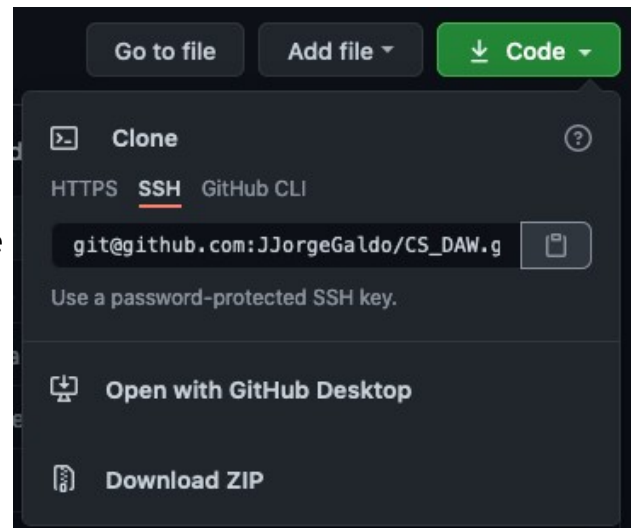
[Create repository](#)

Público o
privado
según
queramos

Si añadimos
Readme, la
descripción
que
pongamos
la incluye
en este
archivo

Clonar repositorio:

- En GitHub, dentro del repositorio que queremos clonar pinchamos en el botón que pone "Code" y, una vez seleccionado SSH, copiamos el texto que aparece justo debajo en el portapapeles.



- Después nos vamos a la terminal, nos ubicamos en el directorio donde queremos clonar el repositorio y tecleamos `$ git clone pegar_portapapeles`
- Una vez terminado de copiar la carpeta, ya hemos clonado el repositorio de GitHub en nuestro equipo y podemos trabajar con el.
- Esto se hace igual en Mac y en Linux

El **flujo de trabajo** será similar en todas las plataformas:

1. Crear / editar ficheros tanto en la terminal como en entorno gráfico
2. Comando `git add nombre_fichero` Con este comando se añade el fichero al Staging area, que es un estado en el que el fichero se encuentra listo para ser confirmado y enviado al repositorio del servidor.
3. Git `commit -m "mensaje"` .Con este comando confirmamos que queremos validar los cambios realizados y obtener un nuevo commit del proyecto.
4. Si no nos ha dado fallos, ejecutamos el `git push`, que lo que hace es enviar al servidor la última versión que acabamos de confirmar (commit). De esta manera todo el equipo tendrá la última versión del proyecto
5. Después del primer acceso al repositorio, lo primero que hay que hacer es ejecutar el comando `git pull`. De esta manera nos descargaremos del servidor la última versión del proyecto en caso de que hubiera cambios con respecto a lo que tenemos guardado en nuestro equipo.

Para **consultar el estado del repositorio** hay dos comandos muy útiles:

- `git status` (muestra info de lo que tenemos pendiente de añadir al staging area, lo que está en el staging area a la espera de ser comiteado, y de archivos que tengan tratamiento especial)
- `git log` (muestra histórico de commits del repositorio con usuarios y mensajes)

7.- Conclusión

Llevo utilizando poco tiempo estas herramientas pero no hace falta ser un genio para darse cuenta de la importancia que tiene en el mundo del desarrollo de software..

Durante las navidades he jugado un poco con GitHub y he encontrado algo interesante. Se trata de los proyectos. Me gustó mucho porque los puedes asociar o no a repositorios existentes, añadir tareas, asignar etiquetas, asignar usuarios a determinadas tareas, y, en general, me parece un muy buen sistema para gestionar los proyectos, porque incluso se puede ir creando una cola de trabajo, y se puede tener todo bajo control. Me parece super útil. Yo he creado varios proyectos con temas que quiero hacer y alguno ya lo tengo empezado (el busca-minas que te comentaba el otro día por ejemplo)

Otra cosa que me ha gustado (GitHub) es la cantidad ingente de información gratuita hecha por desarrolladores profesionales. Todavía me tiene sorprendido y es algo que se tiene que aprovechar, porque tenemos acceso al código explicado por sus creadores de una infinidad de proyectos/lenguajes/tecnologías diferentes.. Es increíble. Relacionado con esto último, también me parece interesante la parte social que implica GitHub. Es muy interesante el poder contactar con profesionales de una manera tan directa.

Estas navidades he migrado todos los ejercicios hechos en las diferentes asignaturas del ciclo a GitHub, y mi intención es utilizarlo para todo lo relacionado con esto. De esta manera me olvido de andar con pendrives de aquí para allá, y siempre tendré accesible las últimas versiones de los ejercicios.

Otra cosa que me gusta es que en cierto modo obliga a organizar los contenidos de una manera más estructurada. Es importante tener un método a la hora de crear las carpetas y nombrarlas para no perderse luego buscando la información.

Con respecto a git, me parece que da mucha tranquilidad saber que estás trabajando en la versión correcta, y que si hay algún problema se pueda revertir.

Estoy deseando avanzar en el uso de estas herramientas y aprender más para poder sacarle más partido, porque ahora mismo ya me está gustando y solo estoy viendo la superficie. Supongo que cuando me dedique a esto más profesionalmente le encontraré un montón de utilidades.

En fin, me parecen dos temas vitales para el desarrollo, tanto para trabajar, como para formación continua, como para networking.