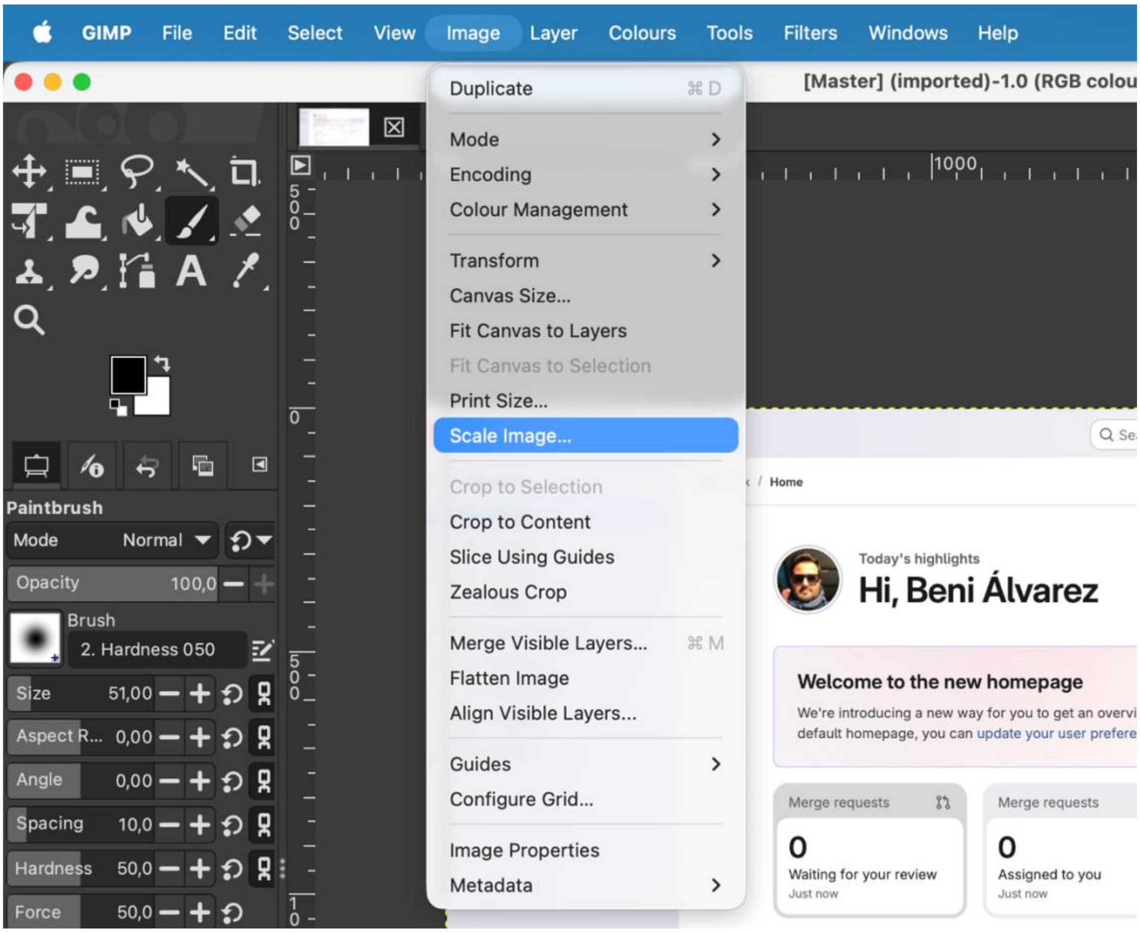
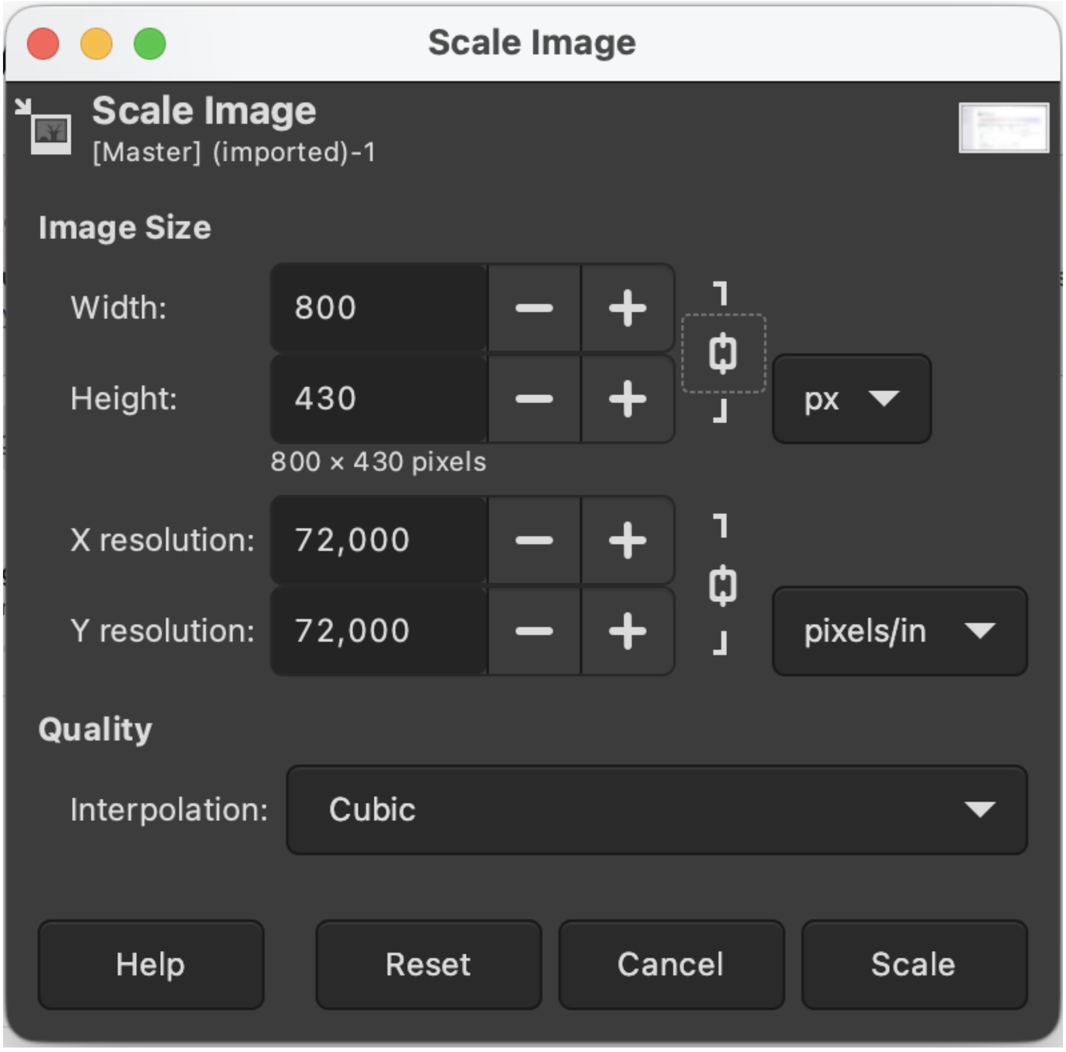
Tenemos una captura de pantalla en PNG de nuestro “home” de GitLab que acabamos de hacer de prueba que ocupa 2Mb y tiene unas dimensiones de 3438x1846 pixels (a una resolución de 144 pixels/inch). Lógicamente es un tamaño excesivo para mostrar en una web tanto por dimensiones como por peso.

Para redimensionar la imagen, vamos a utilizar GIMP (https://www.gimp.org) .

Una vez abierta la imagen en GIMO, iremos al menú “Image” y a la opción “Scale image...”



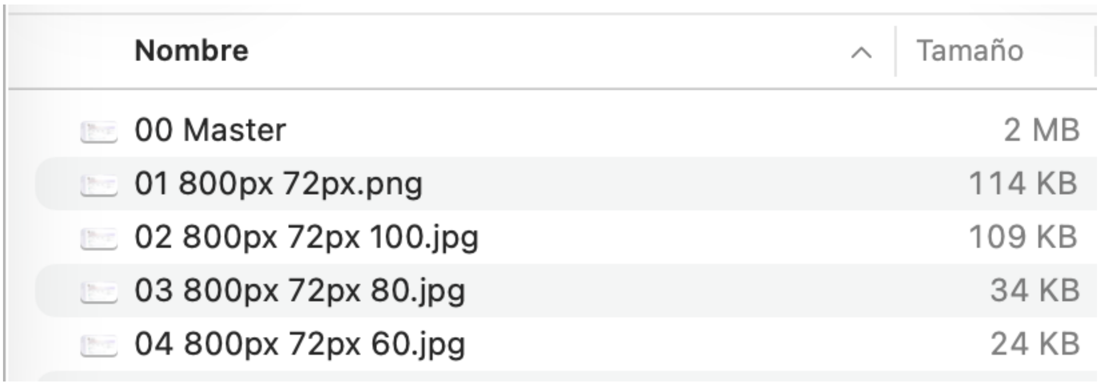

Abrimos la imagen y tomaremos como parámetros iniciales 800 px de ancho (por el alto correspondiente proporcionalmente) y a una resolución de 72 px/inch y pulsamos el botón “Scale” y veremos en pantalla la imagen mas pequeña.




Ahora en el menú, vamos a “File” y a “Export as…”.

Se nos abre una ventana donde podremos elegir la carpeta de destino, el formato de la imagen y las opciones propias de cada formato. Con esta misma imagen, vamos a hacer varias pruebas de tamaño:

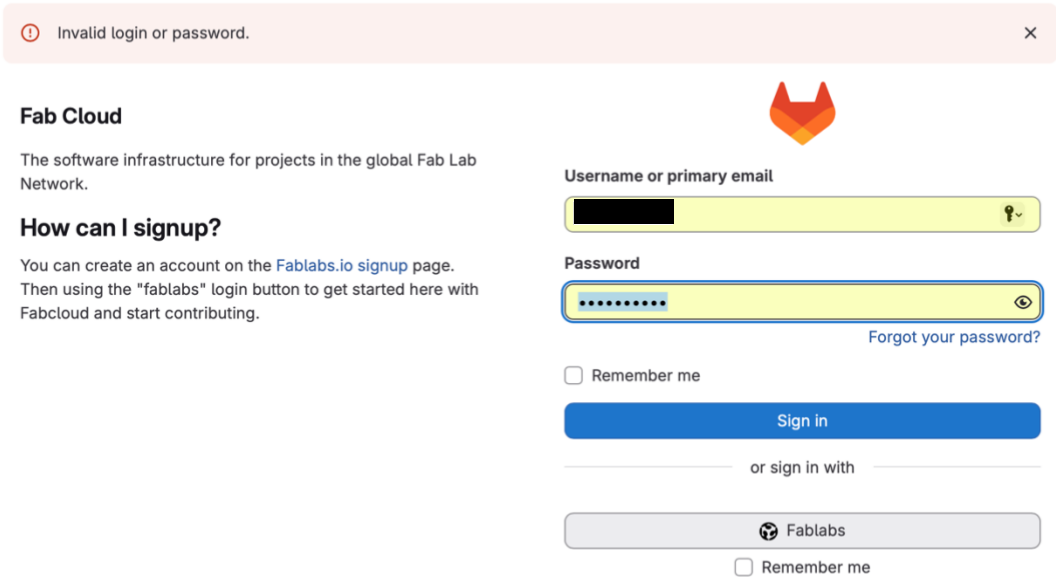
- PNG original: 2Mb
- PNG máxima compresión: 114Kb
- JPG calidad 100%: 109 Kb
- JPG calidad 80%: 34 Kb
- JPG calidad 60%: 24 Kb




Por lo que optamos en nuestro caso por reducir las imágenes y exportarlas a JPG de calidad 80.

Entrar en Gitlab:

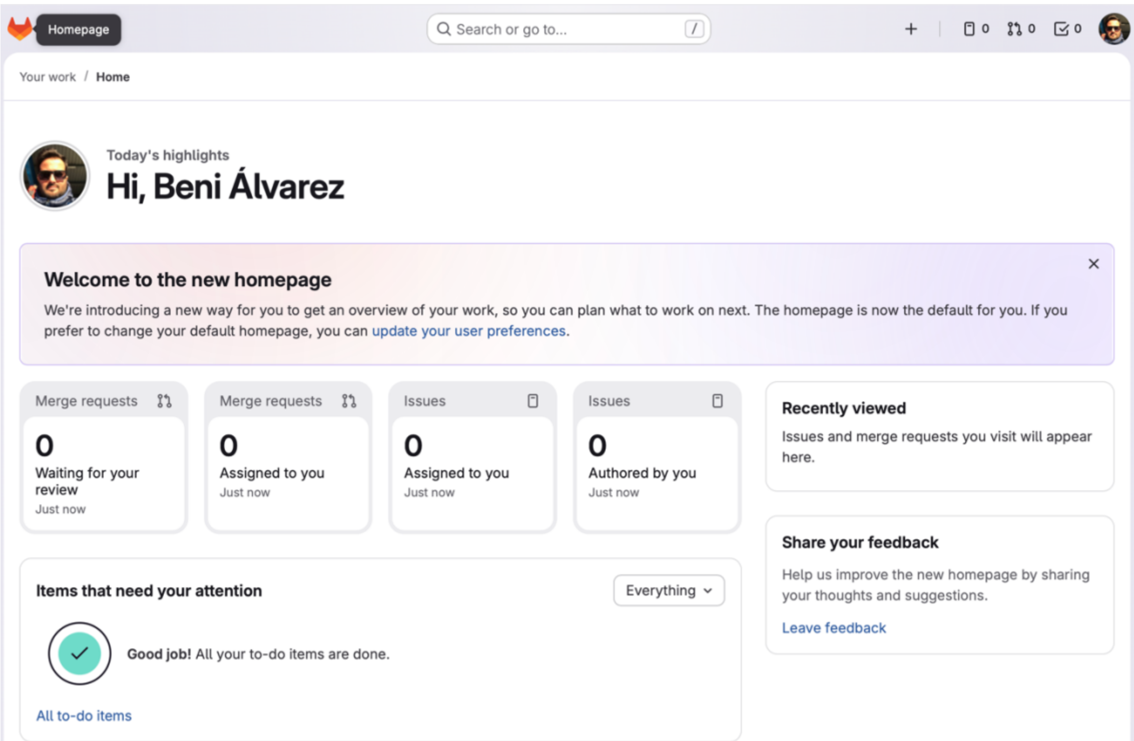
https://gitlab.fabcloud.org/



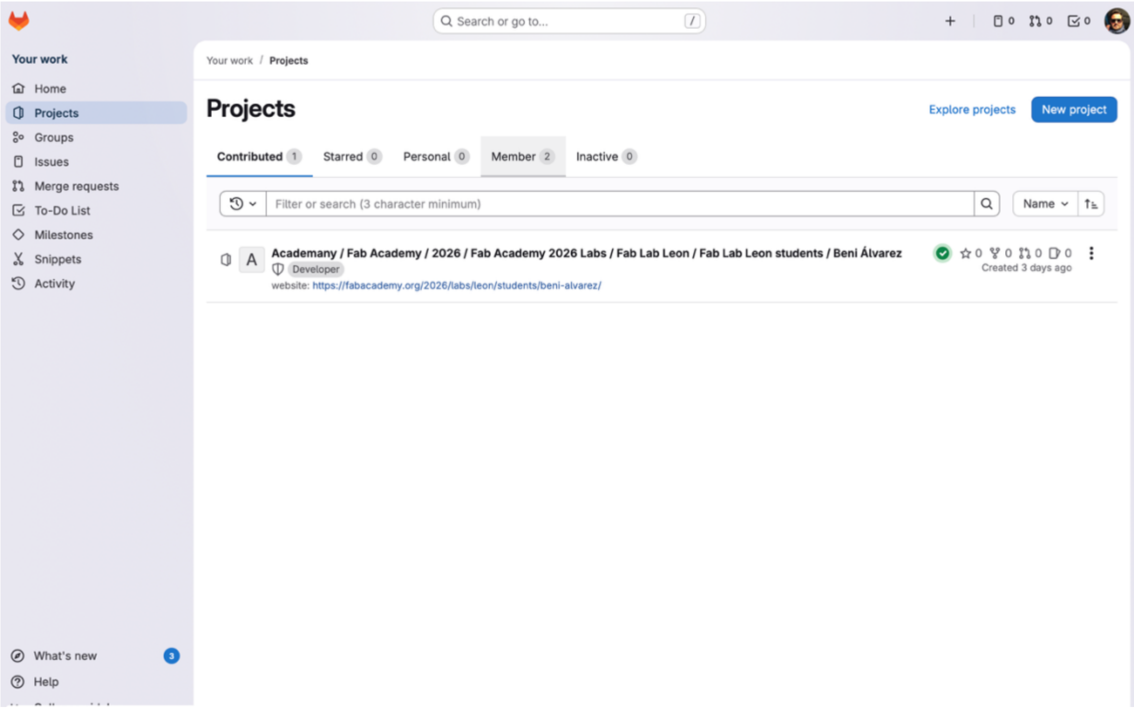

User/email & pass

Entrar pulsando “Fablabs” (no pulsar en el botón “Sign in” porque da error).

Logueado!!!



Una vez dentro, pulsamos “Projects” y se listan los proyectos en los que tenemos permisos.

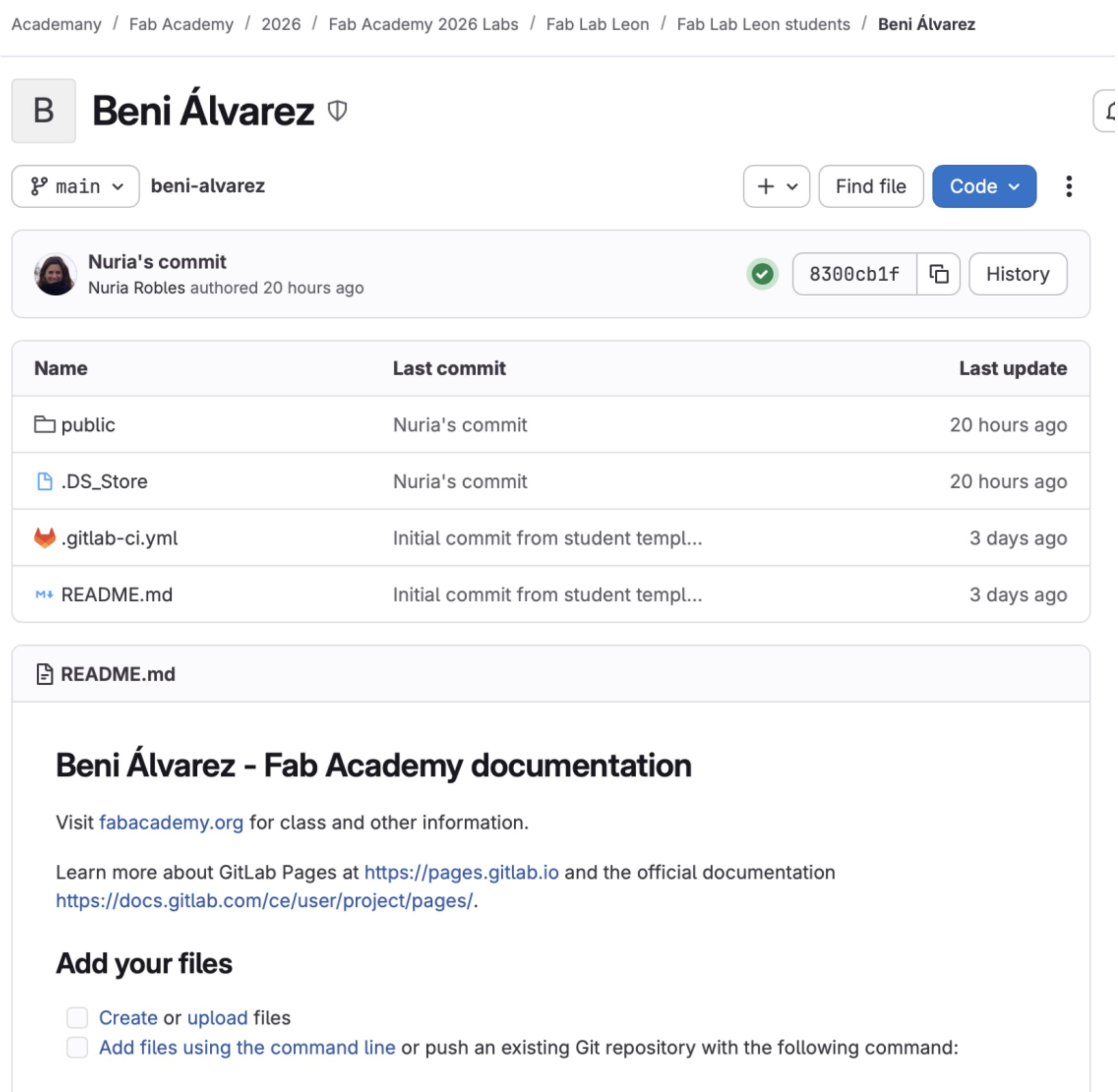



En mi caso, mi espacio personal dentro del FabAcademy 2026.

SI pulsamos sobre “[Academany / Fab Academy / 2026 / Fab Academy 2026 Labs / Fab Lab Leon / Fab Lab Leon students / Beni Álvarez]” accedemos a la parte para manegar el espacio personal.

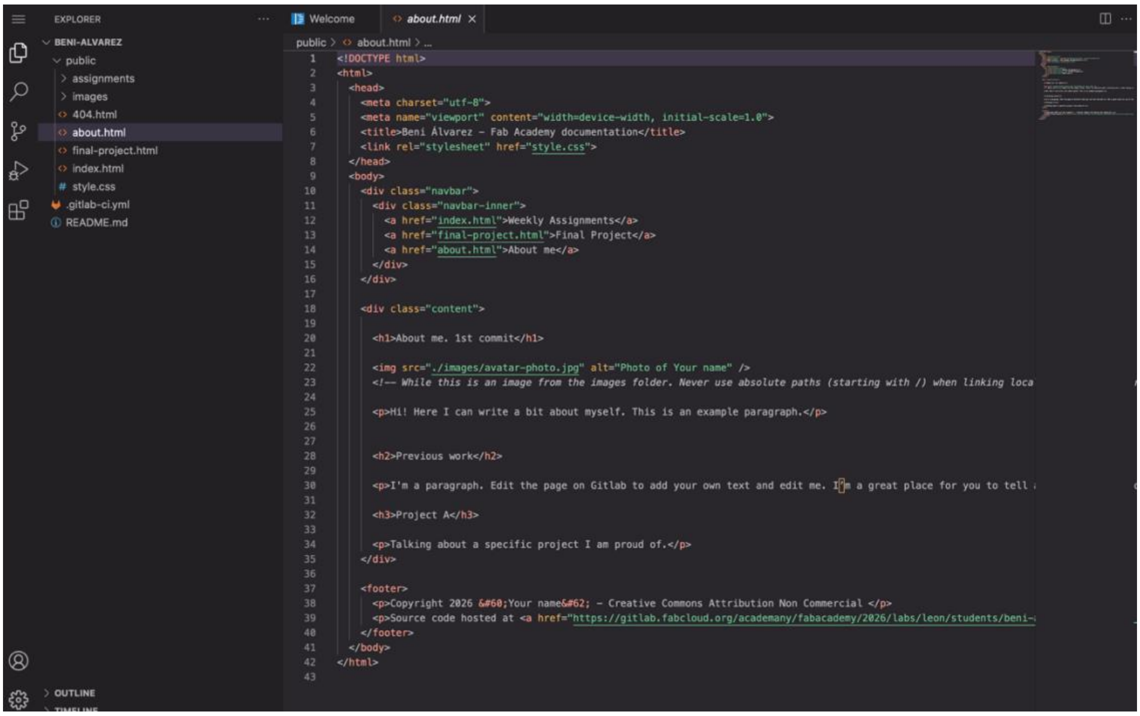
Pulsando sobre el enlace del “website”, abrimos en vestaña/ventana nueva la web pública publicada en nuestro espacio.

Entramos primero al manejo del espacio:




Aquí vemos el primer nivel de archivos que componen nuestro espacio personal y un mensaje rápido sobre la última modificación, en mi caso, un "commit" de Nuria realizado como pruebas.

Como primera prueba, vamos a hacer una modificación del código de un archivo html directamente en la web de Gitlab. Concretamente: "public/about.html". Al pulsar sobre él, nos aparece el código fuente del mismo. Pulsamos sobre el botón "Edit" y de las 2 opciones que se despliegan, entramos en en la de "Open en Web IDE". Así se ve:




Vamos a hacer varias modificaciones:
- Cambiar el título de la página (etiqueta <title>)
- Añadir un encabezado nuevo (etiqueta <h1> en este caso)
- Añadir una foto de perfil para sustituir la que viene de ejemplo

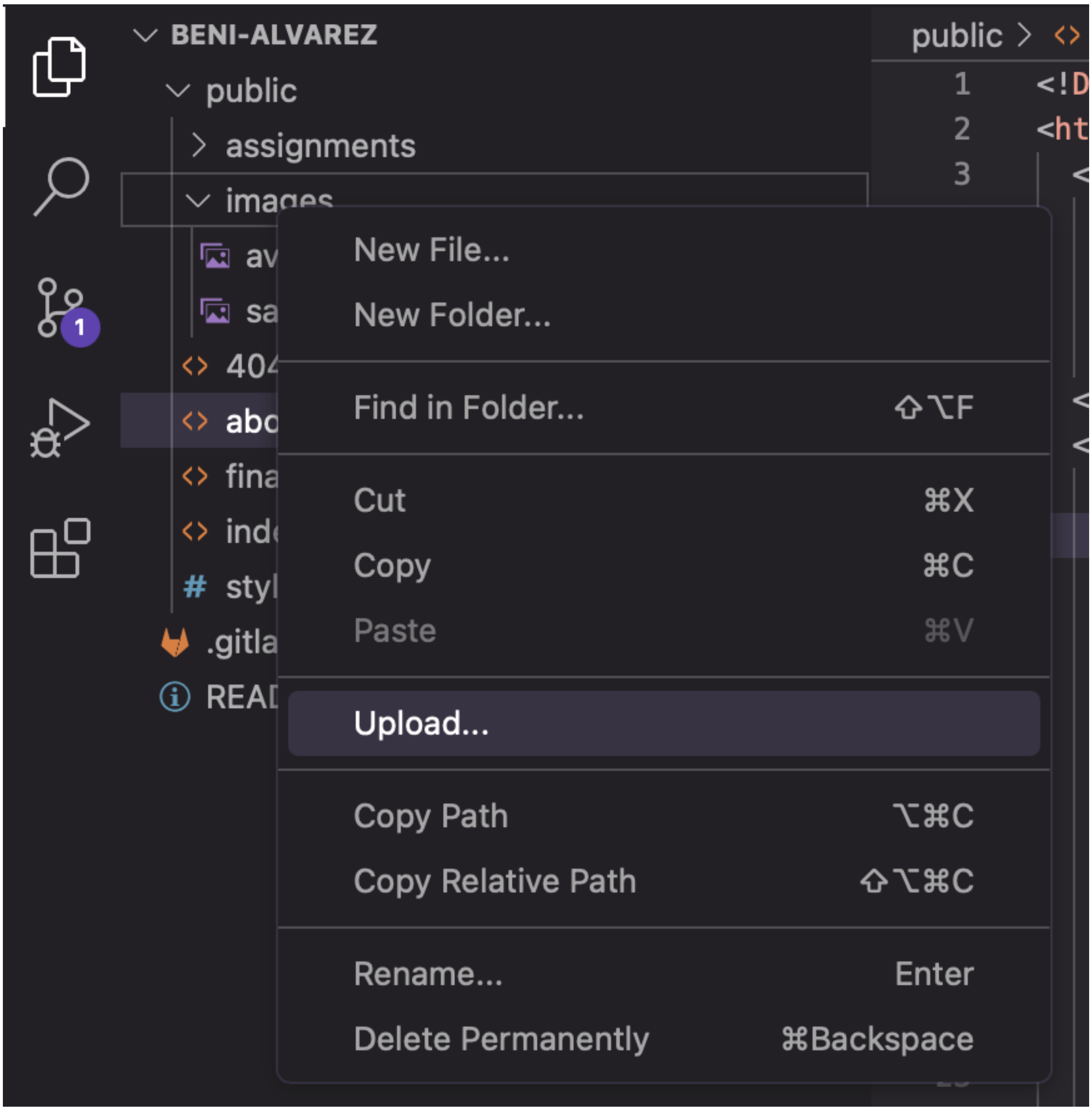
Para el caso de la foto, duplicamos la línea actual y comentamos una de ellas añadiendo “<!--" delante y poniendo “-->”, de esta forma conservamos de manera rápida el código anterior (que posteriormente borraremos).

Abrimos y editamos/recortamos una foto de perfil y la hemos llamado en este caso “face.jpg” que subiremos a la carpeta “images” posteriormente.

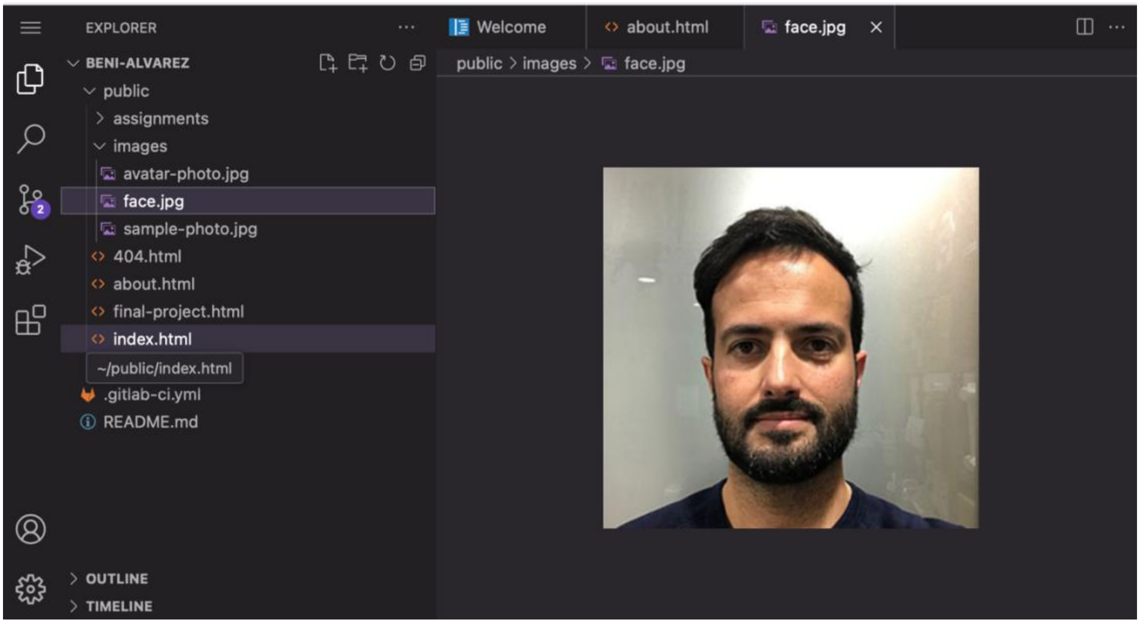
La línea actual quedaría de la siguiente forma: “<img src="./images/face.jpg" alt="This is Me. Beni Álvarez" />”

Ahora, en las opciones principales de la columna izquierda, vemos que nos sale un cambio pendiente.

Vamos a subir el archivo en la carpeta “images”, para ello, en la columna de la izquierda, vemos la estructura actual de archivos/carpetas del sitio. Pulsamos botón derecho sobre la carpeta “images” y seleccionamos la opción “Upload...” y buscamos nuestro archivo y lo subimos.

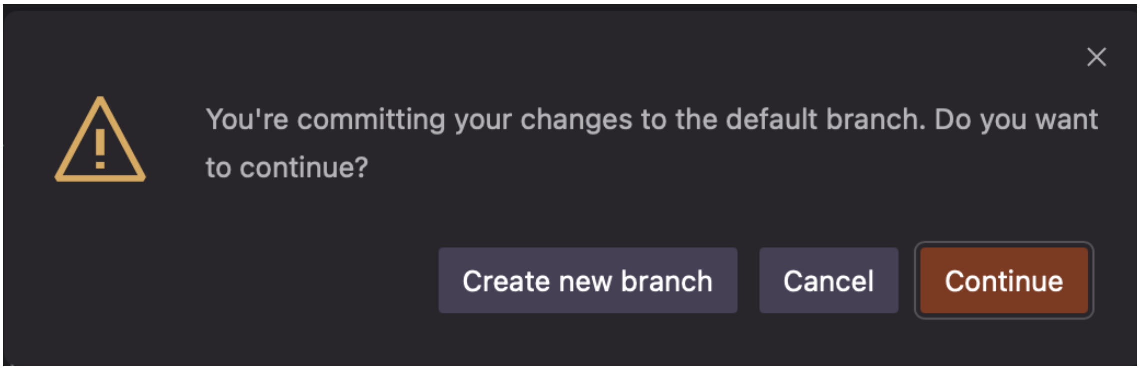



Vemos que se ha subido correctamente y que ahora nos marca que tenemos 2 cambios pendientes.

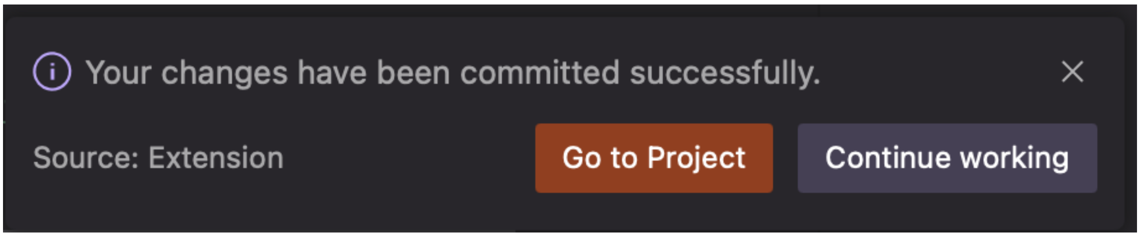



Pulsamos sobre el botón del "Source control" (nos aparecen los cambios pendientes) y pulsamos el botón "Commit and push to 'main'"

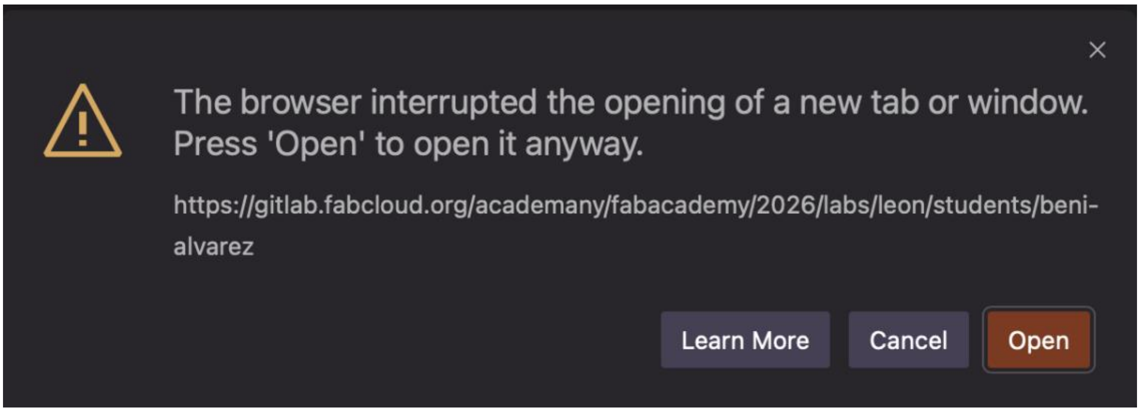
Nos sale la advertencia de confirmación y la aceptamos pulsando "Continue"



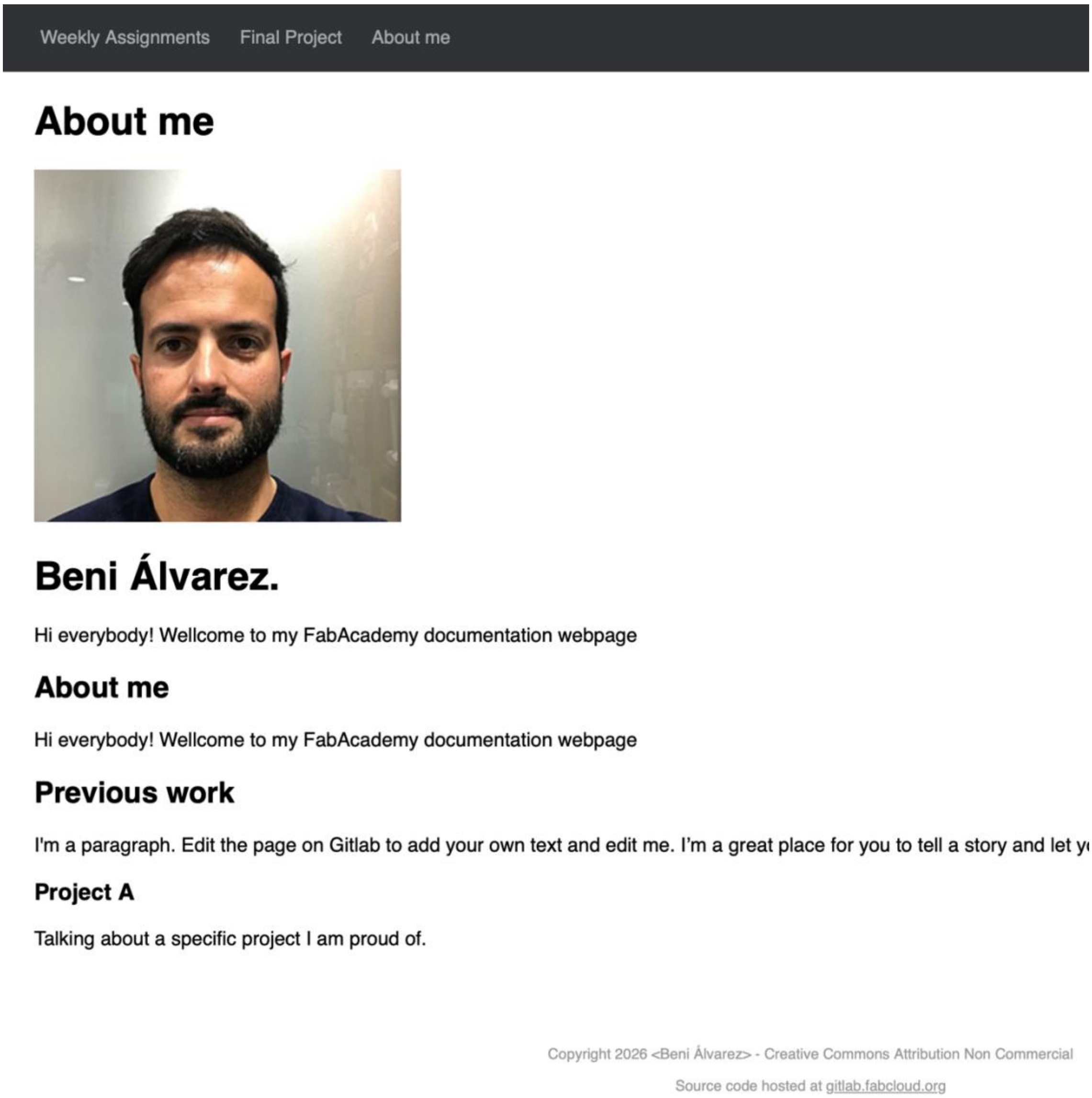

En la parte inferior derecha, sale una ventana emergente en el que indica el estado de la acción que acabamos de realizar. Una vez concluida, nos da el aviso con las opciones de cerrar o con dos accesos directos: uno al proyecto y otro a continuar.



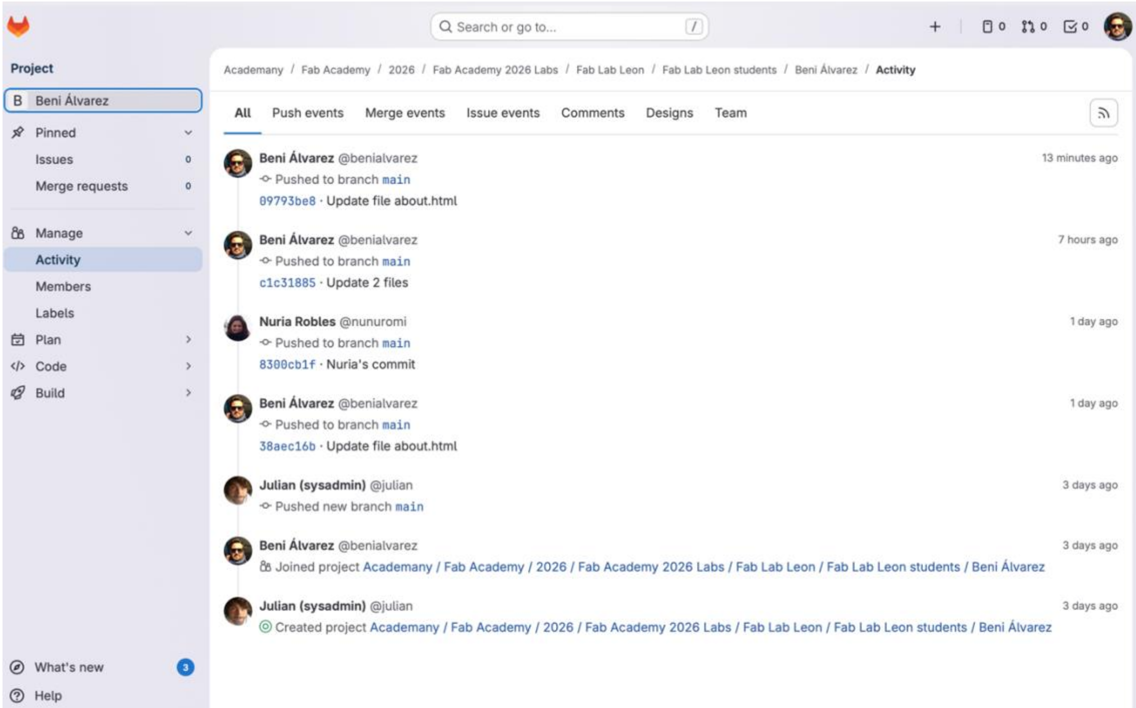

Pulsamos sobre "Go to de Project" para comprobar y visionar los cambios realizados. Nos sale otro mensaje de confirmación y pulsamos "Open" para ver nuestro sitio:




Y desde el menú de la web, accedemos al “Abot me” para ver los cambios realizados:




Si vamos a “Manage” y “Activity” vemos el histórico de modificaciones:



Search or go to...
Project
Beni Álvarez
Pinned
Issues 0
Merge requests 0
Manage
Activity
Members
Labels
Plan
Code
Build
What's new 3
Help
Academany / Fab Academy / 2026 / Fab Academy 2026 Labs / Fab Lab Leon / Fab Lab Leon students / Beni Álvarez / Activity
All
Push events
Merge events
Issue events
Comments
Designs
Team
Beni Álvarez @benialvarez
Pushed to branch main
09793be8 · Update file about.html
13 minutes ago
Beni Álvarez @benialvarez
Pushed to branch main
c1c31885 · Update 2 files
7 hours ago
Nuria Robles @nunuromi
Pushed to branch main
8300cb1f · Nuria's commit
1 day ago
Beni Álvarez @benialvarez
Pushed to branch main
38aec16b · Update file about.html
1 day ago
Julian (sysadmin) @julian
Pushed new branch main
3 days ago
Beni Álvarez @benialvarez
Joined project Academany / Fab Academy / 2026 / Fab Academy 2026 Labs / Fab Lab Leon / Fab Lab Leon students / Beni Álvarez
3 days ago
Julian (sysadmin) @julian
Created project Academany / Fab Academy / 2026 / Fab Academy 2026 Labs / Fab Lab Leon / Fab Lab Leon students / Beni Álvarez
3 days ago

Vamos a descargar y configurar gitlab en local.

En mi caso, uso un ordenador con Mac OS 26.2 (Enero 2026) y voy a utilizar en caso de ser necesario el siguiente tutorial de la web Gitlab para macOS: https://docs.gitlab.com/topics/git/how_to_install_git/?tab=Global+setup

Tengo que descargar y configurar el cliente local de gitlab en mi ordenador y posteriormente para configurarlo, hay q generar y añadir un "SSH key" para emparejar con mi cuenta de Gitlab.

Para hacerlo, usaremos la app "Terminal" dentro de las utilidades de sistema.

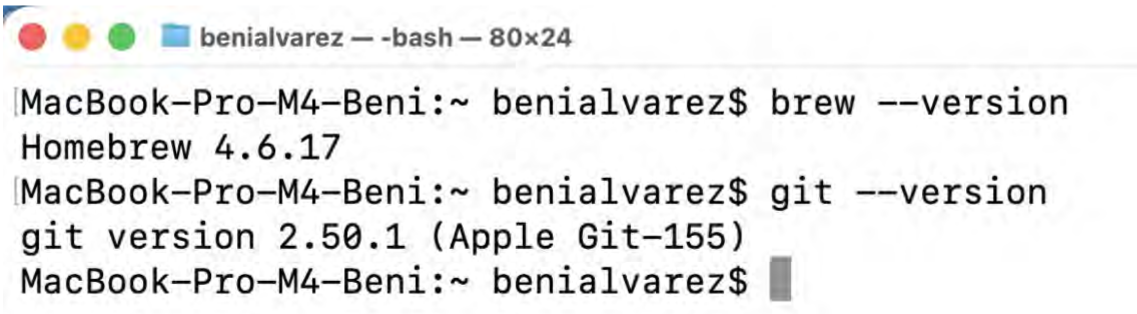
Un requisito previo para instalar el cliente Git en macOS es tener instalda la app "Homebrew". En mi caso, creo recordar que tengo instaladas ambas por necesidades anteriores que sinceramente no recuerdo.

Ejecuto los comando para comprobar si tengo instalado "homebrew" y "Git"

`brew --version`

# git --version

Comprobo que tengo instalado:
Homebrew 4.6.17 y git version 2.50.1 (Apple Git-155)




Compruebo en la web https://git-scm.com que la versión actual disponible es la 2.52.0

Ejecuto el comando de actualización de Git:

# brew upgrade git

Y automáticamente y de forma previa, se actualiza primero "Homebew"

```
benialvarez — ruby • bash -p /opt/homebrew/Library/Homebrew/brew.sh upgrade git — 80×24
MacBook-Pro-M4-Beni:~ benialvarez$ brew --version
Homebrew 4.6.17
MacBook-Pro-M4-Beni:~ benialvarez$ git --version
git version 2.50.1 (Apple Git-155)
MacBook-Pro-M4-Beni:~ benialvarez$ brew upgrade git
==> Auto-updating Homebrew...
Adjust how often this is run with `$HOMEBREW_AUTO_UPDATE_SECS` or disable with
`$HOMEBREW_NO_AUTO_UPDATE=1`. Hide these hints with `$HOMEBREW_NO_ENV_HINTS=1` (
see `man brew`).
==> Downloading https://ghcr.io/v2/homebrew/core/portable-ruby/blobs/sha256:1c98
fa49eacc935640a6f8e10a2bf33f14cfc276804b71ddb658ea45ba99d167
######################################################################## 100.0%
==> Pouring portable-ruby-3.4.8.arm64_big_sur.bottle.tar.gz
```

Al terminal la actualización sale (entre otras cosas) este error: “Error: git not installed”

```
benialvarez — -bash — 80×24
Swift
xcsift: Swift tool to parse xcodebuild output for coding agents
xleak: Terminal Excel viewer with an interactive TUI
zsv: Tabular data swiss-army knife CLI

You have 24 outdated formulae installed.

Error: git not installed
```

Compruebo que ahora homebrew ha subido a la versión Homebrew 5.0.11. y que Git está en la misma versión.

Esto es porque la versión de Git instalada en el sistema es la propia de Xcode / Command Line Tools y no ha sido instalada por homebrew.

Lo confirmo ejecutando el siguiente comando:

which git

```
benialvarez — -bash — 80×24
MacBook-Pro-M4-Beni:~ benialvarez$ git --version
git version 2.50.1 (Apple Git-155)
MacBook-Pro-M4-Beni:~ benialvarez$ which git
/usr/bin/git
MacBook-Pro-M4-Beni:~ benialvarez$
```

Veo que está instalado en /usr/bin/git en vez de en /opt/homebrew/bin/git

Así que procedo a instalar Git gestionado por Homebrew (recomendado) ejecutando el comando:

brew install git

```
benialvarez — -bash — 80×40

MacBook-Pro-M4-Beni:~ benialvarez$ brew install git
✔ JSON API formula.jws.json                          Downloaded   32.0MB/ 32.0MB
✔ JSON API cask.jws.json                             Downloaded   15.3MB/ 15.3MB
==> Fetching downloads for: git
✔ Bottle Manifest git (2.52.0_1)                     Downloaded   20.9KB/ 20.9KB
✔ Bottle Manifest pcre2 (10.47)                      Downloaded   10.4KB/ 10.4KB
✔ Bottle Manifest libiconv (1.18)                    Downloaded    7.5KB/  7.5KB
✔ Bottle Manifest gettext (0.26_1)                   Downloaded   13.8KB/ 13.8KB
✔ Bottle libiconv (1.18)                             Downloaded    1.6MB/  1.6MB
✔ Bottle pcre2 (10.47)                               Downloaded    2.4MB/  2.4MB
✔ Bottle gettext (0.26_1)                            Downloaded    9.6MB/  9.6MB
✔ Bottle git (2.52.0_1)                              Downloaded   21.5MB/ 21.5MB
==> Installing dependencies for git: gettext, pcre2 and libiconv
==> Installing git dependency: gettext
==> Pouring gettext--0.26_1.arm64_tahoe.bottle.tar.gz
🍺  /opt/homebrew/Cellar/gettext/0.26_1: 2,428 files, 29.5MB
==> Installing git dependency: pcre2
==> Pouring pcre2--10.47.arm64_tahoe.bottle.tar.gz
🍺  /opt/homebrew/Cellar/pcre2/10.47: 244 files, 7.3MB
==> Installing git dependency: libiconv
==> Pouring libiconv--1.18.arm64_tahoe.bottle.tar.gz
🍺  /opt/homebrew/Cellar/libiconv/1.18: 32 files, 2.9MB
==> Installing git
==> Pouring git--2.52.0_1.arm64_tahoe.bottle.1.tar.gz
==> Caveats
The Tcl/Tk GUIs (e.g. gitk, git-gui) are now in the `git-gui` formula.
Subversion interoperability (git-svn) is now in the `git-svn` formula.
==> Summary
🍺  /opt/homebrew/Cellar/git/2.52.0_1: 1,733 files, 59.7MB
==> Running `brew cleanup git`...
Disable this behaviour by setting `HOMEBREW_NO_INSTALL_CLEANUP=1`.
Hide these hints with `HOMEBREW_NO_ENV_HINTS=1` (see `man brew`).
==> Caveats
Bash completion has been installed to:
  /opt/homebrew/etc/bash_completion.d
==> git
The Tcl/Tk GUIs (e.g. gitk, git-gui) are now in the `git-gui` formula.
Subversion interoperability (git-svn) is now in the `git-svn` formula.
MacBook-Pro-M4-Beni:~ benialvarez$
```

Fuerzo la comprobación de la versión instalada por homebrew con el comando:

ls -l /opt/homebrew/bin/git
/opt/homebrew/bin/git --version

Y confirmo que en el homebrew tengo instalada la versión 2.52.2 (la última)

```
benialvarez — -bash — 80×40
MacBook-Pro-M4-Beni:~ benialvarez$ ls -l /opt/homebrew/bin/git
lrwxr-xr-x  1 benialvarez  admin  30 23 ene.  16:46 /opt/homebrew/bin/git -> ../
Cellar/git/2.52.0_1/bin/git
MacBook-Pro-M4-Beni:~ benialvarez$ /opt/homebrew/bin/git --version
git version 2.52.0
MacBook-Pro-M4-Beni:~ benialvarez$
```

**Problema:**

**Git de Homebrew sí se instaló** (se ve `git 2.52.0_1` en `/opt/homebrew/Cellar/...`).
Lo que pasa es que desde la app "Terminal" **sigue encontrando primero el Git de Apple** porque el **PATH** está priorizando `/usr/bin` antes que `/opt/homebrew/bin`

Busco documentación y ejecuto el siguiente comando para priorizar la versión de Git instalada por homebrew en vez de la Git de Apple.

echo 'eval "$(/opt/homebrew/bin/brew shellenv)"' >> ~/.bash_profile

eval "$(/opt/homebrew/bin/brew shellenv)"

hash -r

Ejecuto "which git" y "git –version" y compruebo que los cambios han tenido éxito.

```
benialvarez — -bash — 80×40
MacBook-Pro-M4-Beni:~ benialvarez$ which git
/opt/homebrew/bin/git
MacBook-Pro-M4-Beni:~ benialvarez$ git --version
git version 2.52.0
MacBook-Pro-M4-Beni:~ benialvarez$
```

CONFIGURACIÓN DE GITLAB en local

Empezamos a configurar la aplicación local de Git, continuando con el manual de GitLab Docs (https://docs.gitlab.com/topics/git/how_to_install_git/?tab=Global+setup)

Primero, configuramos el usuario y el email con los siguientes comandos:

git config --global user.name "my_user"

git config --global user.email "my_email"

Y ejecuto este otro comando para comprobar que los cambios han sido correctos:

git config --global –list

```
benialvarez — -bash — 83x40
MacBook-Pro-M4-Beni:~ benialvarez$ git config --global --list
user.name=[redacted]
user.email=[redacted]
MacBook-Pro-M4-Beni:~ benialvarez$
```

Ahora necesitamos generar una clave privada para poder firmar los "commits" y poder enviarlos al repositorio remoto, pero antes de eso, primero comprobaremos que no tenemos ya una creada ejecutando el siguiente comando:

ls -al ~/.ssh

```
benialvarez — -bash — 83x40
MacBook-Pro-M4-Beni:~ benialvarez$ ls -al ~/.ssh
total 16
drwx------    4 benialvarez  staff   128 13 feb.   2025 .
drwxr-xr-x+ 48 benialvarez  staff  1536 23 ene.  18:10 ..
-rw-------    1 benialvarez  staff  1107 21 nov.   2024 known_hosts
-rw-------    1 benialvarez  staff   365 20 nov.   2024 known_hosts.old
MacBook-Pro-M4-Beni:~ benialvarez$
```

En mi caso, no tengo ninguno creado por lo que seguimos el tutorial correspondiente en GitLab (https://docs.gitlab.com/user/ssh/) .

Para ello, vamos a utilizar el algoritmo "ED25519" asociado a la dirección de email registrada en GitLab ejecutando el siguiente comando:

ssh-keygen -t ed25519 -C benialvarez@gmail.com

Nos preguntará en que directorio queremos almacenar las claves, si aceptamos el directorio por defecto (/.ssh/id_ed2519) pulsamos "intro".
Nos pedirá una contraseña de cifrado y nos pide confirmarla.
Si todo es correcto, nos la habrá generado una clave privada y una pública.

```
MacBook-Pro-M4-Beni:~ benialvarez$ ssh-keygen -t ed25519 -C "benialvarez@gmail.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/Users/benialvarez/.ssh/id_ed25519):
Enter passphrase for "/Users/benialvarez/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/benialvarez/.ssh/id_ed25519
Your public key has been saved in /Users/benialvarez/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:LWCFNrDJR\[redacted]8EYfLG2zUuRAcz0 [redacted]
The key's randomart image is:
+--[ED25519 256]--+
|    .oo+o..o.+ o.|
[redacted]
|          .. .   |
+----[SHA256]-----+
MacBook-Pro-M4-Beni:~ benialvarez$
```

(lógicamente he censurado la clave privada que me ha generado).

Ahora, arrancaremos el agente y añadimos la clave con el siguiente comando:

eval "$(ssh-agent -s)"

ssh-add --apple-use-keychain ~/.ssh/id_ed25519

```
MacBook-Pro-M4-Beni:~ benialvarez$ eval "$(ssh-agent -s)"
Agent pid 24652
MacBook-Pro-M4-Beni:~ benialvarez$ ssh-add --apple-use-keychain ~/.ssh/id_ed25519
Enter passphrase for /Users/benialvarez/.ssh/id_ed25519:
Identity added: /Users/benialvarez/.ssh/id_ed25519 (benialvarez@gmail.com)
MacBook-Pro-M4-Beni:~ benialvarez$
```

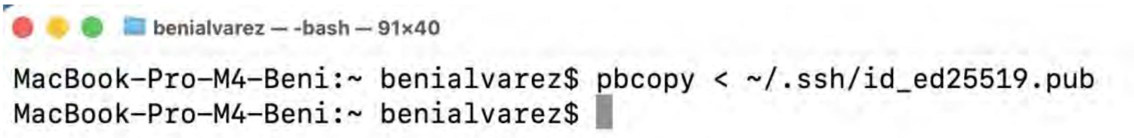
Y podemos comprobar que está cargada ejecutando:

ssh-add -l

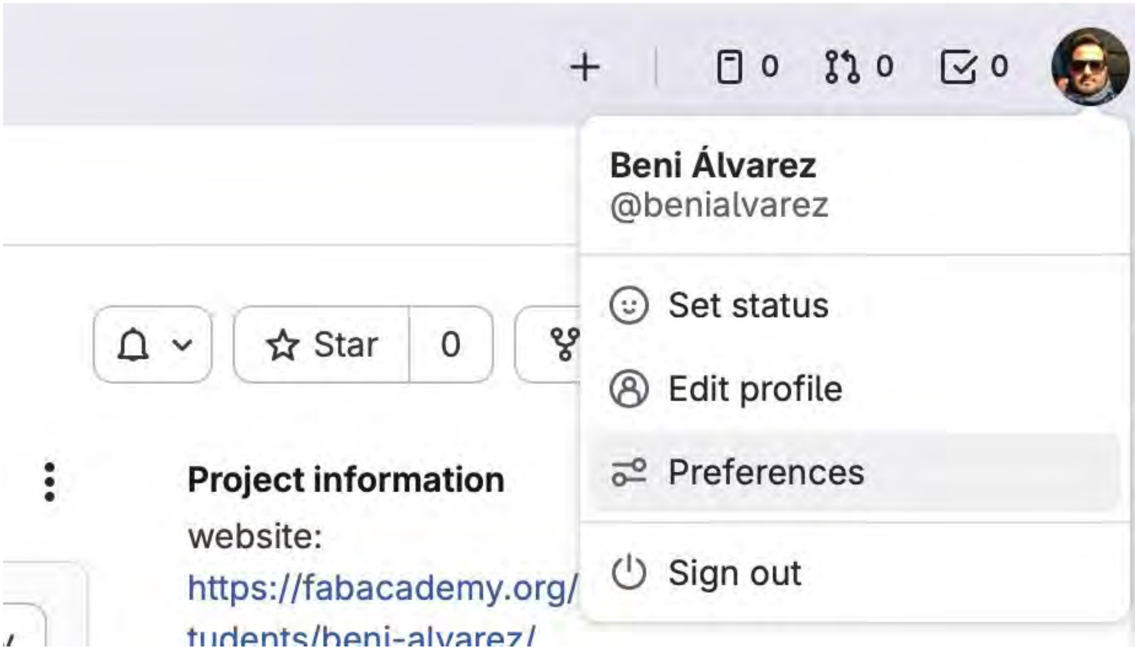
```
MacBook-Pro-M4-Beni:~ benialvarez$ ssh-add -l
256 SHA256:LWCFNrDJRV[redacted]8EYfLG2zUuRAcz0 [redacted](ED25519)
MacBook-Pro-M4-Beni:~ benialvarez$
```

Lo siguiente es copiarnos la clave pública en el portapapeles para poder pegarla en la web de GitLab. Para ello, ejecutamos el siguiente comando:

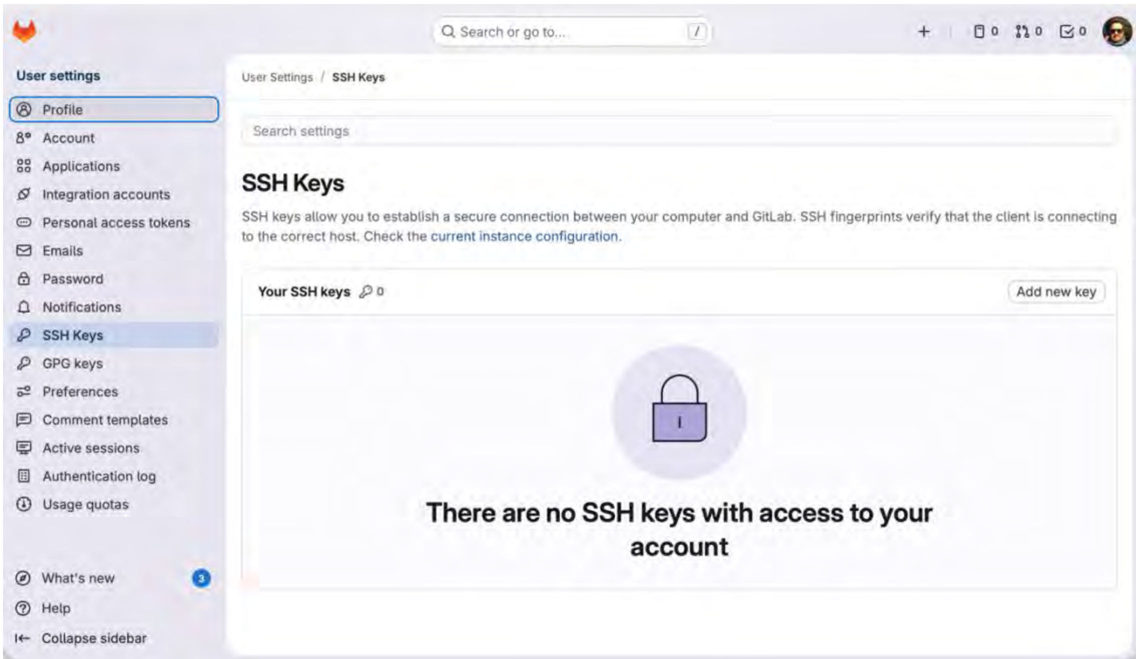
pbcopy < ~/.ssh/id_ed25519.pub




Ahora vamos a la web, y accedemos a los ajustes/preferencias, pulsado en nuestro icono de usuario en la parte superior derecha.

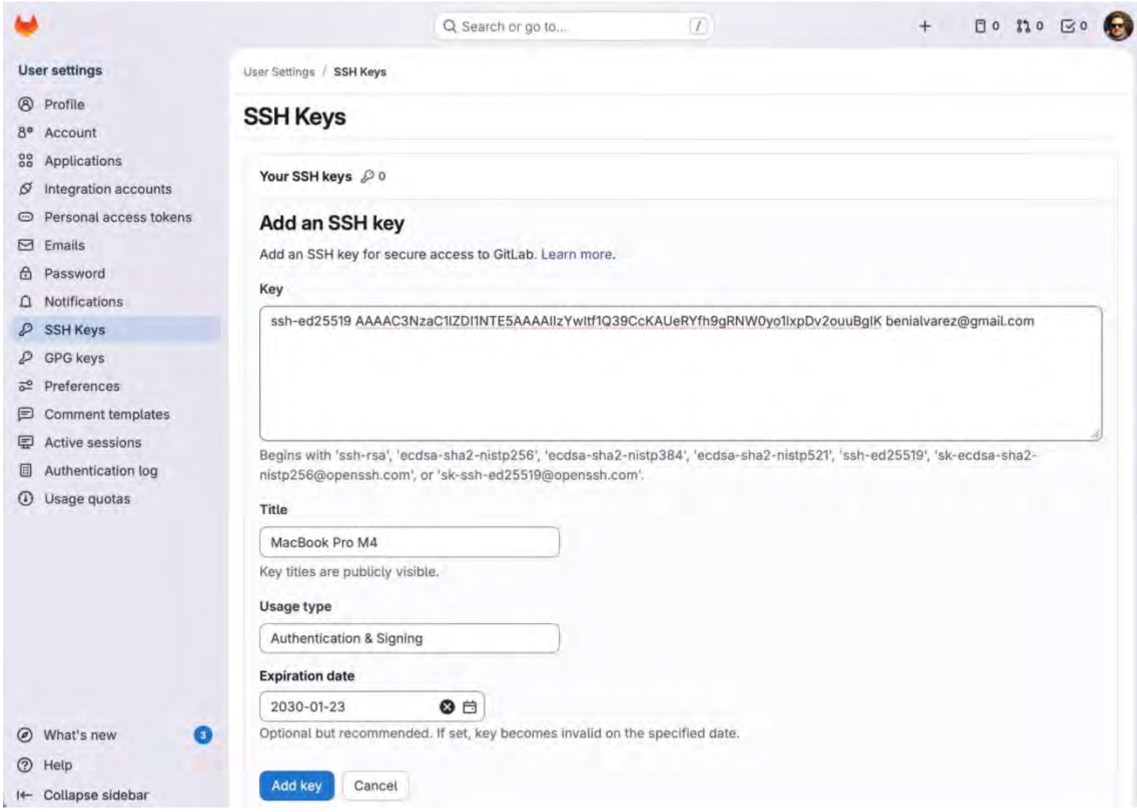



Una vez allí, en el menú de la columna de la izquierda, pulsamos sobre la opción “SSH Keys” y pulsamos el botón “Add new key”




Se nos abre un formulario para cumplimentar los datos de la clave:

En el campo “key” pegamos del portapapeles la clave pública que nos habíamos copiado y cumplimentamos los siguientes campos.




Y le damos al botón “Add key”.

Ahora, para comprobar si todo está configurado correctamente, volvemos a la app Terminal y ejecutamos el siguiente comando:

ssh -T git@gitlab.fabcloud.org

Si es la primera vez es normal que salga el mensaje *“authenticity of host can’t be established”* ya que es la primera vez que nos conectamos por ssh al servidor de GitLab y desde en el ordenador no teníamos guardada la “host key”, por lo que pulsamos “yes”.

Si todo es correcto, nos devolverá nuestro nombre de usuario, en mi caso:

“Welcome to GitLab, @benialvarez!”

```
benialvarez — -bash — 91×40
MacBook-Pro-M4-Beni:~ benialvarez$ ssh -T git@gitlab.fabcloud.org
The authenticity of host 'gitlab.fabcloud.org (13.59.248.79)' can't be established.
ED25519 key fingerprint is SHA256:JCWnquB9Cdr[redacted]i8j8KckmYNCq/M.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'gitlab.fabcloud.org' (ED25519) to the list of known hosts.
Welcome to GitLab, @benialvarez!
MacBook-Pro-M4-Beni:~ benialvarez$
```

Ahora vamos a crear una carpeta que sea la que utilicemos en nuestro repositorio local. En mi caso dentro de mi carpeta de usuario, he creado una que he llamado “FA2026”. Para vincular el proyecto actual con la carpeta local, debemos inicializar Git dentro de la carpeta ejecutando el comando:

git init

```
FA2026 — -bash — 91×40
MacBook-Pro-M4-Beni:FA2026 benialvarez$ pwd
/Users/benialvarez/FA2026
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: will change to "main" in Git 3.0. To configure the initial branch name
hint: to use in all of your new repositories, which will suppress this warning,
hint: call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
hint:
hint: Disable this message with "git config set advice.defaultBranchName false"
Inicializado repositorio Git vacío en /Users/benialvarez/FA2026/.git/
MacBook-Pro-M4-Beni:FA2026 benialvarez$
```

Con esto, hemos inicializado el repositorio de Git vacío en la carpeta FA2026.

Ahora debemos establecer el origen para este repositorio vacío que acabamos de crear. Para ello, ejecutamos el comando “git remote add origin” seguido de nuestro código de “Clone with SSH”. En mi caso quedaría de la siguiente manera:

git remote add origin
git@gitlab.fabcloud.org:academany/fabacademy/2026/labs/leon/students/beni-alvarez.git

```
FA2026 — -bash — 87×40
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git remote add origin  git@gitlab.fabcloud.org:
academany/fabacademy/2026/labs/leon/students/beni-alvarez.git
MacBook-Pro-M4-Beni:FA2026 benialvarez$
```

Para comprobar que se ha establecido correctamente, ejecutamos el siguiente comando:

git remote -v

```
FA2026 — -bash — 102×40
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git remote -v
origin  git@gitlab.fabcloud.org:academany/fabacademy/2026/labs/leon/students/beni-alvarez.git (fetch)
origin  git@gitlab.fabcloud.org:academany/fabacademy/2026/labs/leon/students/beni-alvarez.git (push)
MacBook-Pro-M4-Beni:FA2026 benialvarez$
```

Ahora debemos descargarnos y sincronizar ambos repositorios por lo que primero comprobaremos los encabezados del repo del servidor ejecutando:

git ls-remote --heads origin

Vemos que en este caso corresponden a “refs/heads/main”. Ejecutamos:

git fetch origin

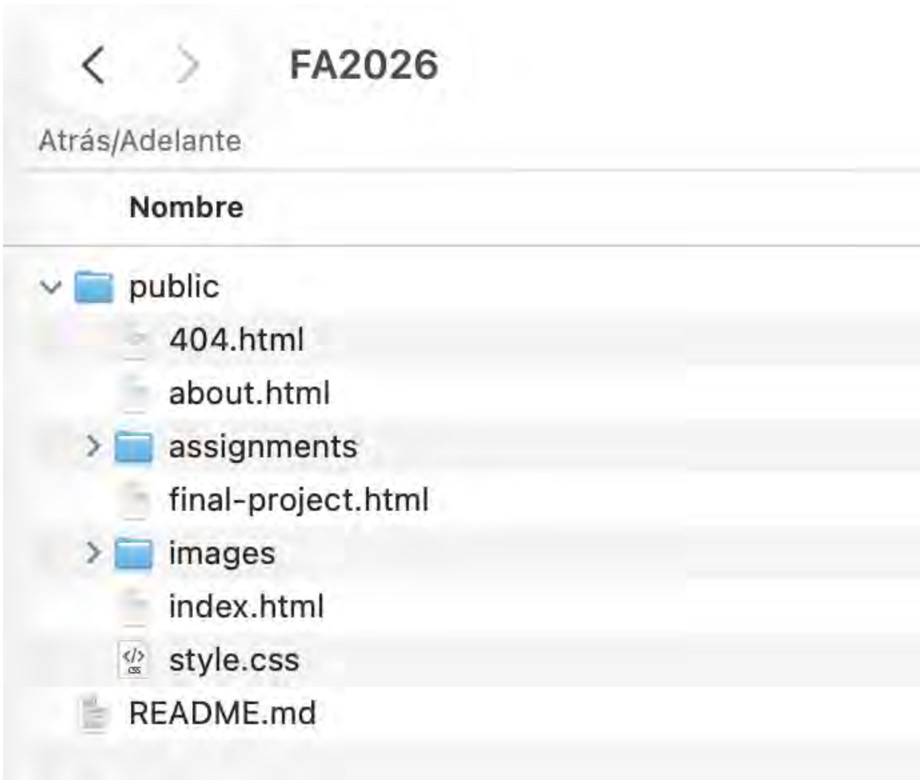
Y después:

git checkout -B main origin/main

Y deberíamos ver algo como esto:

```
FA2026 — -bash — 102×40
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git fetch origin
remote: Enumerating objects: 34, done.
remote: Total 34 (delta 0), reused 0 (delta 0), pack-reused 34 (from 1)
Desempaquetando objetos: 100% (34/34), 46.59 KiB | 221.00 KiB/s, listo.
Desde gitlab.fabcloud.org:academany/fabacademy/2026/labs/leon/students/beni-alvarez
 * [nueva rama]      main       -> origin/main
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git checkout -B main origin/main
rama 'main' configurada para rastrear 'origin/main'.
Cambiado a nueva rama 'main'
MacBook-Pro-M4-Beni:FA2026 benialvarez$
```

Por lo que si vamos a nuestra carpeta local, deberemos de ver todo el contenido del repo del servidor descargado:




Ahora, vamos a modificar un archivo en el repo local, por ejemplo, editamos el archivo “about.html” y modificamos la etiqueta “<h1>About me </h1>” por “<h1>About me LOCAL.</h1>”.

Para comprobar el estado de la sincronización entre ambos repos ejecutando el comando:

git status

```
FA2026 — -bash — 102×40
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git status
En la rama main
Tu rama está actualizada con 'origin/main'.

Cambios no rastreados para el commit:
  (usa "git add <archivo>..." para actualizar lo que será confirmado)
  (usa "git restore <archivo>..." para descartar los cambios en el directorio de trabajo)
        modificados:     public/about.html

sin cambios agregados al commit (usa "git add" y/o "git commit -a")
MacBook-Pro-M4-Beni:FA2026 benialvarez$
```

Nos indica la modificación del archivo about.html pendiente de sincronizar, para ello, ejecutamos los siguientes comandos:

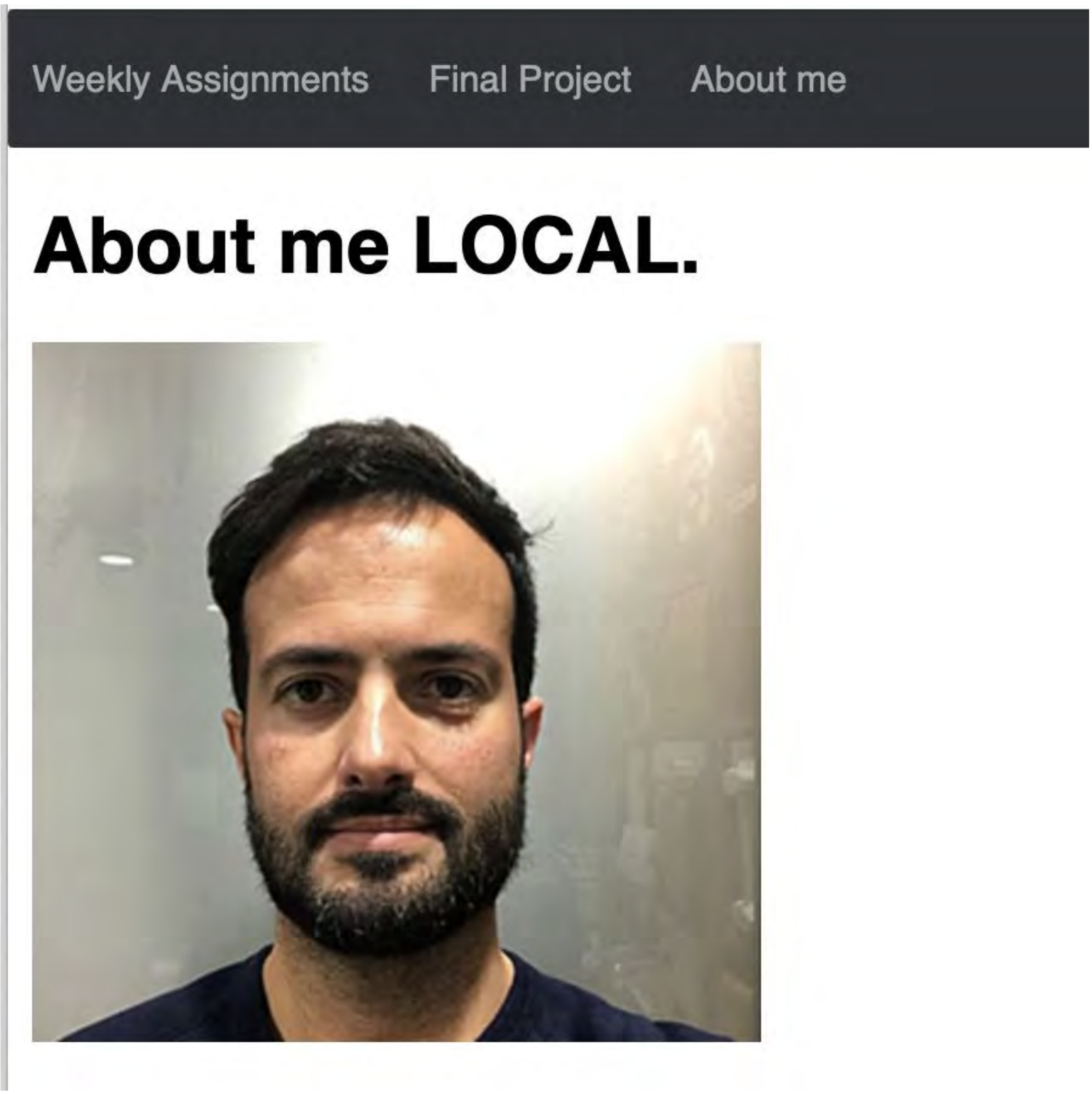
- `git add .` → añade **todos los cambios** (archivos nuevos, modificados y borrados) al *stage*

- `git commit -m "..."` → guarda todos esos cambios en un commit con el comentario

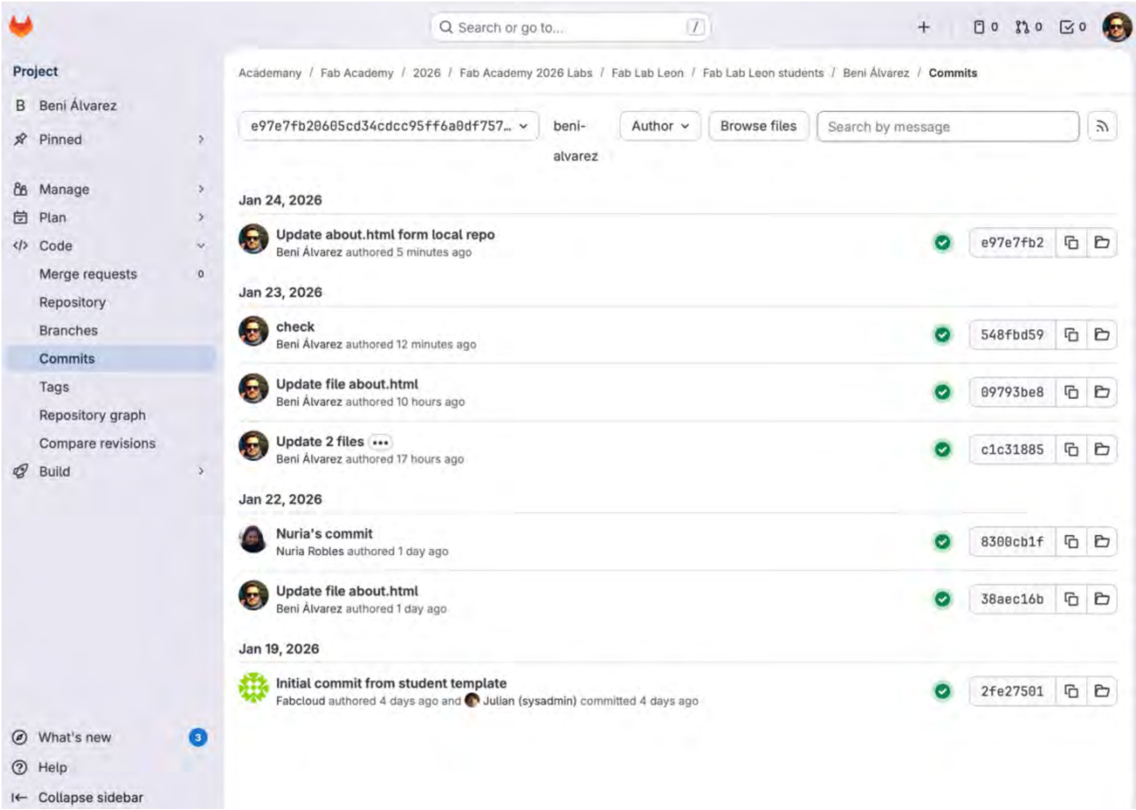
- `git push` → los sube al repositorio remoto

```
FA2026 — -bash — 102×40
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git add .
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git commit -m "Update about.html form local repo"
[main e97e7fb] Update about.html form local repo
 1 file changed, 1 insertion(+), 1 deletion(-)
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git push
Enumerando objetos: 7, listo.
Contando objetos: 100% (7/7), listo.
Compresión delta usando hasta 10 hilos
Comprimiendo objetos: 100% (4/4), listo.
Escribiendo objetos: 100% (4/4), 363 bytes | 363.00 KiB/s, listo.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
To gitlab.fabcloud.org:academany/fabacademy/2026/labs/leon/students/beni-alvarez.git
   548fbd5..e97e7fb  main -> main
MacBook-Pro-M4-Beni:FA2026 benialvarez$
```

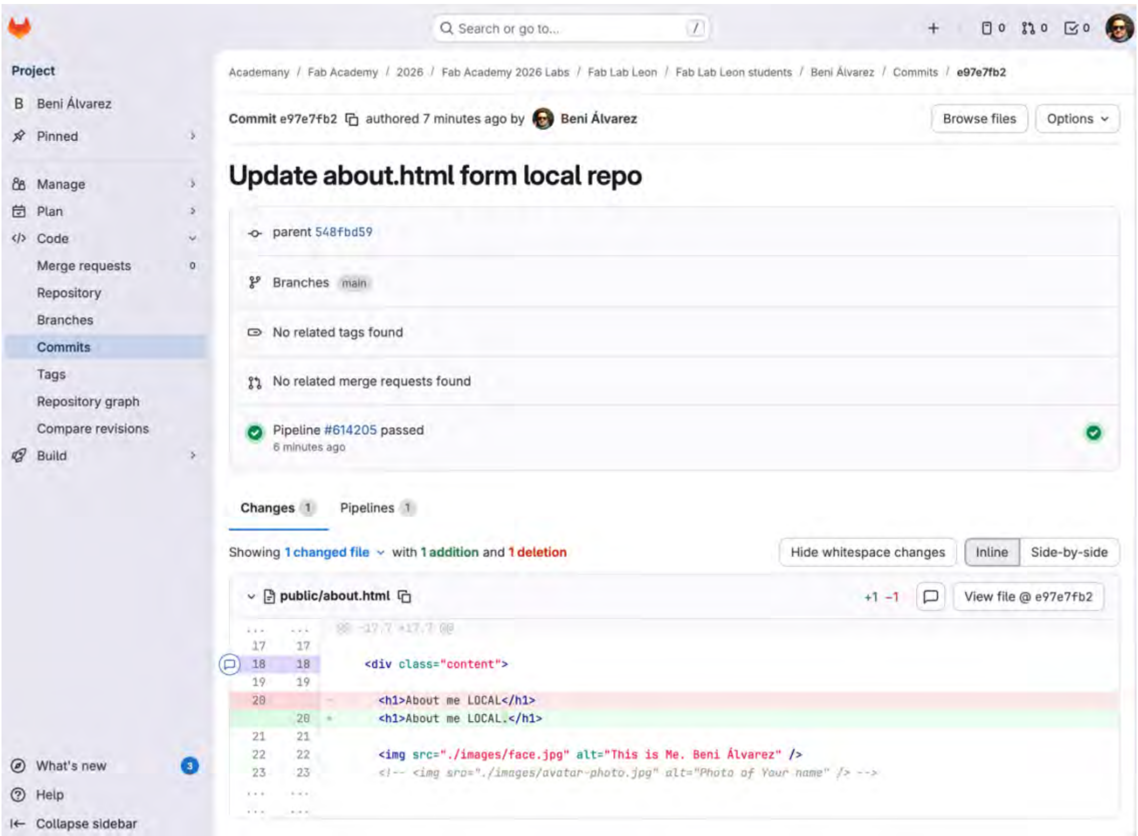
Ahora, accedemos a la web y vemos si se ha actualizado la etiqueta "<h1>About me LOCAL.</h1>" en nuestro repo del servidor.




Y si accedemos al histórico de "commits", veremos que efectivamente está el que acabamos de hacer con su comentario:




Si pulsamos sobre él, vemos los cambios que se han realizado:



Volvemos a la app Terminal y ejecutamos de nuevo “git status” para comprobar que no hay nada pendiente:

```
FA2026 — -bash — 102×40
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git status
En la rama main
Tu rama está actualizada con 'origin/main'.

nada para hacer commit, el árbol de trabajo está limpio
MacBook-Pro-M4-Beni:FA2026 benialvarez$
```

Ahora, vamos a volver a modificar la etiqueta “<h1>About me LOCAL.</h1>” por “<h1>About me SERVER.</h1>” de la página about.php en el editor web del servidor (Web IDE) para comprobar el comportamiento respecto al repo local. Para hacerlo, seguiremos los pasos más arriba indicados.

Una vez hecho el cambio en el código html y hecho el commit, comprobamos actualizando la web en navegador web que el cambio han sido efectivo y volvermos a la app Terminal y comprobamos con el comando “git status”.

```
FA2026 — -bash — 102×40
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git status
En la rama main
Tu rama está actualizada con 'origin/main'.

nada para hacer commit, el árbol de trabajo está limpio
MacBook-Pro-M4-Beni:FA2026 benialvarez$
```

Lógicamente nos dice que no hay cambios, ya que el comando “git status” no mira en el servidor, solo mira la copia en local y efectivamente en ella no hemos hecho cambios.

Por tanto, tenemos que ejecutar previamente el comando "git fetch” para que se revisen si hay cambios en el servidor y posteriormente ejecutar ya el “git status”, que nos dirá en este caso que nuestra rama local está por detrás de la del servidor y que tenemos un commit pendiente y que debemos usar “git pull” para actualizarlo en la local

```
FA2026 — -bash — 102×35
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git fetch
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
Desempaquetando objetos: 100% (4/4), 348 bytes | 69.00 KiB/s, listo.
Desde gitlab.fabcloud.org:academany/fabacademy/2026/labs/leon/students/beni-alvarez
   e97e7fb..19b97bf  main       -> origin/main
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git status
En la rama main
Tu rama está detrás de 'origin/main' por 1 commit, y puede ser avanzada rápido.
  (usa "git pull" para actualizar tu rama local)

nada para hacer commit, el árbol de trabajo está limpio
MacBook-Pro-M4-Beni:FA2026 benialvarez$
```

Por tanto, ejecutamos el comando y vemos el resultado

git pull

```
FA2026 — -bash — 102×35
MacBook-Pro-M4-Beni:FA2026 benialvarez$ git pull
Actualizando e97e7fb..19b97bf
Fast-forward
 public/about.html | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
MacBook-Pro-M4-Beni:FA2026 benialvarez$
```

Abrimos el archivo “about.html” local en un editor html y confirmamos que se ha actualizado.