

Plantilla Nivel de Conocimiento

Nombre del instructor	Adolfo Centeno Tellez		
Área temática	Ciencia de datos		
Rol	Data Scientist		
Competencia	Full Stack		
Subcompetencia	Frontend		
Objetivo Módulo Conocimiento			
Módulo Conocimiento (5h en total)			
	Duración	Objetivos	Subtemas
Tema 5 - Despliegue en produccion de una aplicacion en Angular	1 hora	Analizar los mecanismos para el despliegue en producción de una aplicación en Angular	1.-configuración de github, Instalación y configuración de Docker 2.- Workflows en github para despliegue continuo

Introducción al tema

En los temas anteriores hemos desarrollado una aplicación en Angular 18 para crear nuestro propio CV en un sitio web, hemos usado componentes, servicios, una base de datos no relacional, pruebas unitarias, de integración y end-to-end.

Al finalizar este módulo adquirirás habilidades en la configuración de github, creación de repositorios, así como subir el código de nuestro proyecto a github, además de publicar nuestro proyecto Angular en Github Pages.

Aprender como instalar, configurar y comandos básicos de Docker, creación de imágenes, contenedores y publicación a Docker Hub.

Como tema avanzado crearemos un conjunto de scripts en github llamados workflows que nos permitirán automatizar la integración y despliegue continuo de nuestro proyecto. Este Workflow además creará automáticamente una imagen de Docker que se publicará al contenedor de imágenes denominado Docker Hub. Finalmente vas a desplegar tu imagen de Docker Hub en una nube pública.

SUBTEMA 1: Configuración de github, Instalacion y configuracion de Docker

Creación de una versión distribuible de nuestro proyecto

Para publicar nuestro proyecto debemos construir una versión compatible con cualquier nube de cómputo, que solo tenga HTML, CSS y Javascript. Para lograrlo usaremos Angular CLI para crear esta versión distribuible y compatible con cualquier nube.

En la raíz del proyecto

```
$ ls
```

```
[→ mycv git:(master) ✘ ls
README.md          cypress           node_modules      public          tsconfig.json
angular.json        cypress.config.ts  package-lock.json  src            tsconfig.spec.json
coverage           karma.conf.js     package.json     tsconfig.app.json
→ mycv git:(master) ✘ ]
```

Creamos la versión distribuible del proyecto y lo guardamos en una carpeta **docs** (ya que **Github Pages** requiere ese nombre para publicar un proyecto).

```
$ ng build --output-path=docs
```

La operación anterior nos genera una carpeta **docs/browser**. Requerimos que el index.html, css y js, se encuentren en la raíz de la carpeta **docs**, las movemos a un nivel anterior.

```
$ mv docs/browser/*.* docs
```

```
$ ls docs
```

```

+ mycv git:(master) ✘ ng build --output-path=docs
Initial chunk files | Names | Raw size | Estimated transfer size
| main | 581.93 kB | 149.71 kB
| - | 173.66 kB | 41.93 kB
| polyfills | 34.52 kB | 11.28 kB
| styles | 0 bytes | 0 bytes
| Initial total | 790.10 kB | 202.92 kB

Lazy chunk files | Names | Raw size | Estimated transfer size
| chunk-DT56ZVVI.js | index-esm | 29 bytes | 29 bytes

Application bundle generation complete. [5.174 seconds]

⚠ WARNING bundle initial exceeded maximum budget. Budget 512.00 kB was not met by 278.10 kB with a total of 790.10 kB.

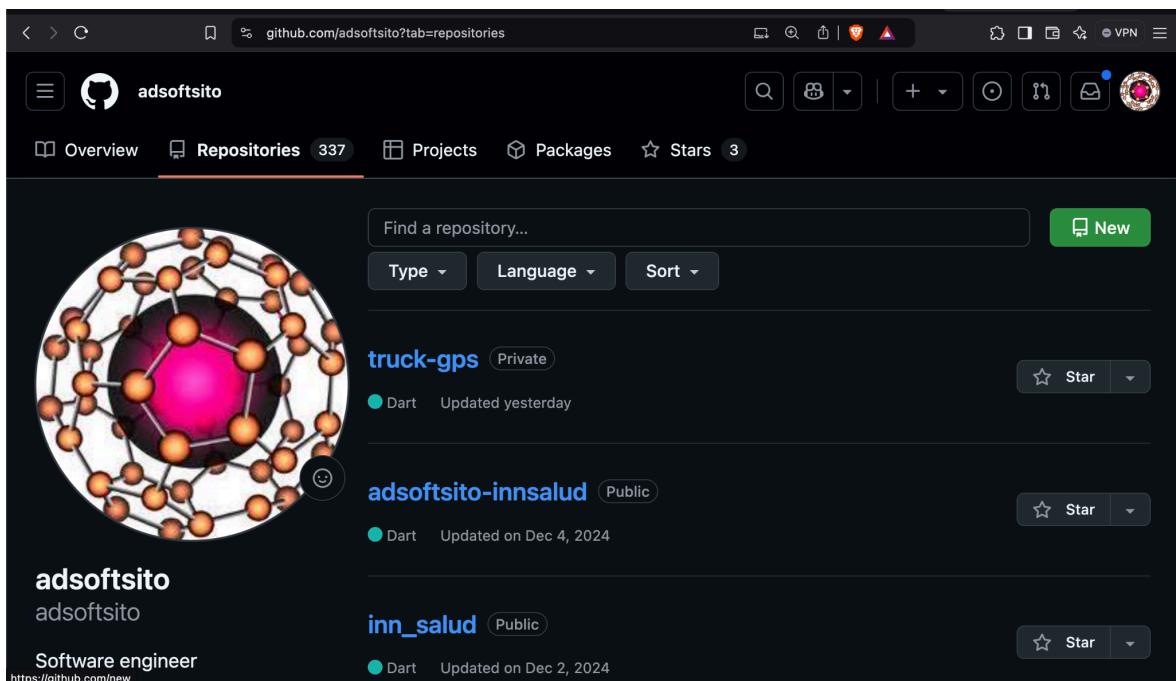
Output location: /Users/adsoft/mycv/docs

+ mycv git:(master) ✘ ls docs
3rdpartylicenses.txt browser
+ mycv git:(master) ✘ mv docs/browser/*.* docs
+ mycv git:(master) ✘ ls docs
3rdpartylicenses.txt chunk-DT56ZVVI.js main-DHG2USRS.js
browser favicon.ico polyfills-FFHMD2TL.js
chunk-4ZHOUZTK.js index.html styles-5INURTSO.css

```

Subir el proyecto Angular a un repositorio

Creamos un repositorio **público**, en <https://github.com>, Click en New



Para que nuestro proyecto pueda publicarse usando el hosting de Github Pages, debe tener un nombre especial: **tu-usuario-git.github.io**.

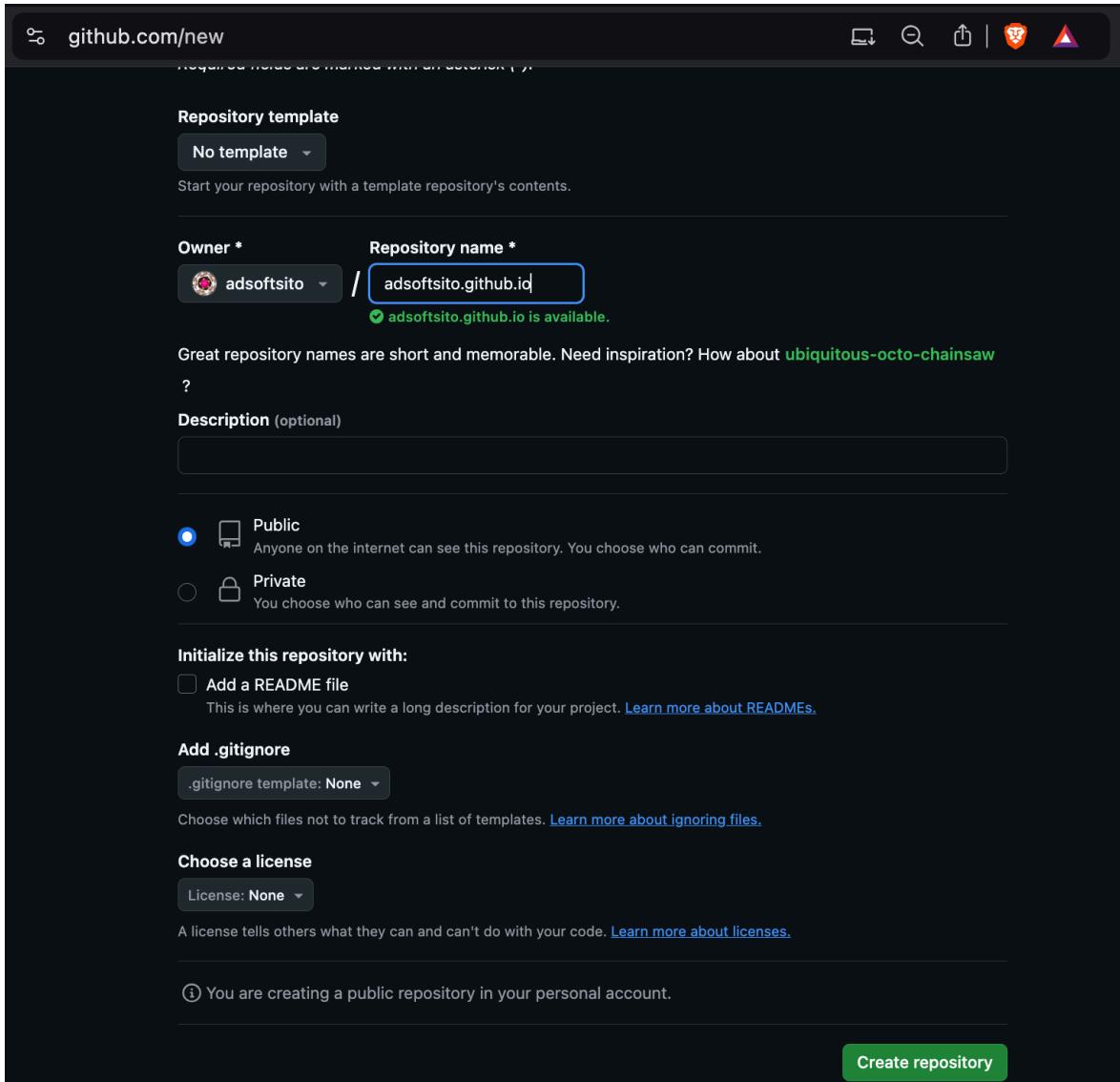
Ejemplos:

- **adsoftsito.github.io**
- **octocat.github.io**
- **juanperez.github.io**

Además :

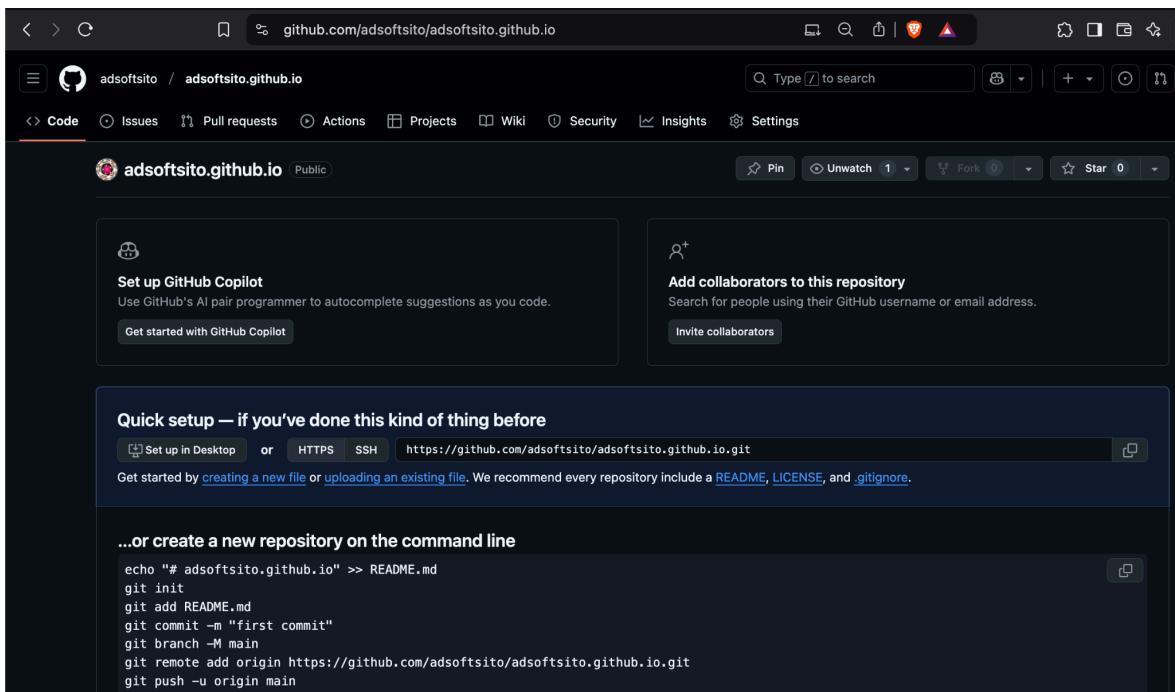
- Debe ser público
- No debe crear README,
- No debe crear .gitignore
- No seleccionar ningún tipo de licencia

Click en Create Repository



The screenshot shows the GitHub 'Create Repository' form. The 'Repository name' field contains 'adsoftsito.github.io'. A green checkmark indicates that 'adsoftsito.github.io' is available. The 'Owner' dropdown is set to 'adsoftsito'. The 'Repository name' field is highlighted with a blue border. Below the repository name, there is a note: 'adsoftsito.github.io is available.' Under 'Description (optional)', there is a large empty text area. The 'Visibility' section shows 'Public' selected (indicated by a blue dot) and 'Private' unselected. The 'Public' option has a description: 'Anyone on the internet can see this repository. You choose who can commit.' The 'Initialize this repository with:' section contains an unchecked checkbox for 'Add a README file'. The 'Add .gitignore' section shows '.gitignore template: None'. The 'Choose a license' section shows 'License: None'. A note at the bottom left says: 'You are creating a public repository in your personal account.' At the bottom right is a green 'Create repository' button.

Nuestro nuevo repositorio, luce así.



Es momento de subir nuestro proyecto al repositorio recién creado, copiamos el comando de github.com con la url de nuestro proyecto:

git remote add origin https://github.com/<usuario>/<tu-usuario>/github.io

Abrimos la consola, nos movemos a la raíz de nuestro proyecto, ejecutamos el comando (ejemplo)

```
$ pwd
```

```
$ ls -la
```

```
→ mycv git:(master) pwd
/Users/adsoft/mycv
→ mycv git:(master) ls -la
total 1232
drwxr-xr-x  23 adsoft  staff      736 Jan  7 18:02 .
drwxr-x---+ 81 adsoft  staff     2592 Jan  9 12:19 ..
-rw-r--r--@  1 adsoft  staff     6148 Jan  2 18:55 .DS_Store
drwxr-xr-x  3 adsoft  staff      96 Dec 26 01:33 .angular
-rw-r--r--   1 adsoft  staff     274 Dec 26 01:30 .editorconfig
drwxr-xr-x 12 adsoft  staff     384 Jan  7 18:03 .git
-rw-r--r--   1 adsoft  staff     587 Dec 26 01:30 .gitignore
drwxr-xr-x  5 adsoft  staff     160 Dec 26 01:30 .vscode
-rw-r--r--   1 adsoft  staff    1065 Dec 26 01:30 README.md
-rw-r--r--   1 adsoft  staff    4457 Jan  7 16:58 angular.json
drwxr-xr-x  4 adsoft  staff      128 Jan  2 18:55 coverage
drwxr-xr-x  7 adsoft  staff     224 Jan  6 15:24 cypress
-rw-r--r--   1 adsoft  staff     264 Jan  6 13:46 cypress.config.ts
drwxr-xr-x 11 adsoft  staff     352 Jan  7 18:03 docs
-rw-r--r--   1 adsoft  staff    1421 Dec 31 02:09 karma.conf.js
drwxr-xr-x 677 adsoft  staff   21664 Jan  6 13:46 node_modules
-rw-r--r--   1 adsoft  staff   573630 Jan  6 13:50 package-lock.json
-rw-r--r--   1 adsoft  staff     1219 Jan  6 13:46 package.json
drwxr-xr-x  3 adsoft  staff      96 Dec 26 01:30 public
drwxr-xr-x  7 adsoft  staff     224 Dec 26 17:04 src
-rw-r--r--   1 adsoft  staff     424 Dec 26 01:30 tsconfig.app.json
-rw-r--r--   1 adsoft  staff    1021 Dec 26 01:30 tsconfig.json
-rw-r--r--   1 adsoft  staff     434 Dec 26 01:30 tsconfig.spec.json
→ mycv git:(master) █
```

\$ **git remote add origin <https://github.com/adsoftsito/adsoftsito.github.io.git>**

Verificamos que nuestro proyecto local, ya apunte a nuestro repositorio en github. deberá retornar el **origin** de nuestro repositorio.

\$ **git remote -v**

```
→ mycv git:(master) ✘ git remote add origin https://github.com/adsoftsito/adsoftsito.github.io.git
→ mycv git:(master) ✘ git remote -v
origin  https://github.com/adsoftsito/adsoftsito.github.io.git (fetch)
origin  https://github.com/adsoftsito/adsoftsito.github.io.git (push)
→ mycv git:(master) ✘ █
```

Agregamos todos nuestros archivos del proyecto con: **git add .**

\$ **git add .**

```
[→ mycv git:(master) ✘ git add .]
```

Creamos una nueva versión del proyecto con **git commit**

```
$ git commit -m "deploy angular cv app"
```

```
[→ mycv git:(master) ✘ git commit -m "deploy angular cv app"

[master fb45031] deploy angular cv app
 67 files changed, 5552 insertions(+), 2748 deletions(-)
 create mode 100644 cypress.config.ts
 create mode 100644 cypress/e2e/spec.cy.ts
 create mode 100644 cypress/fixtures/example.json
 create mode 100644 cypress/support/commands.ts
 create mode 100644 cypress/support/component-index.html
 create mode 100644 cypress/support/component.ts
 create mode 100644 cypress/support/e2e.ts
 create mode 100644 cypress/tsconfig.json
 create mode 100644 karma.conf.js
 create mode 100644 src/app/certificates/certificates.component.css
 create mode 100644 src/app/certificates/certificates.component.html
 create mode 100644 src/app/certificates/certificates.component.spec.ts
 create mode 100644 src/app/certificates/certificates.component.ts
 create mode 100644 src/app/compute/compute.spec.ts
 create mode 100644 src/app/compute/compute.ts
 create mode 100644 src/app/currencies/currencies.spec.ts
 create mode 100644 src/app/currencies/currencies.ts
 create mode 100644 src/app/education/education.component.css
 create mode 100644 src/app/education/education.component.html
 create mode 100644 src/app/education/education.component.spec.ts
 create mode 100644 src/app/education/education.component.ts
 create mode 100644 src/app/greet/greet.spec.ts
 create mode 100644 src/app/greet/greet.ts
 create mode 100644 src/app/header/header.component.css
 create mode 100644 src/app/header/header.component.html
 create mode 100644 src/app/header/header.component.spec.ts
 create mode 100644 src/app/header/header.component.ts
 create mode 100644 src/app/interests/interests.component.css
```

NOTA: En caso de que sea la primera vez que usas git en tu computadora, lanzará un error y recomendará la ejecución de 2 comandos, antes de realizar tu primer commit.

```
$ git config --global user.name "Your github Name"
```

```
$ git config --global user.email "your email"
```

Después de estos 2 comandos, repetir el comando:

```
$ git commit -m "deploy angular cv app"
```

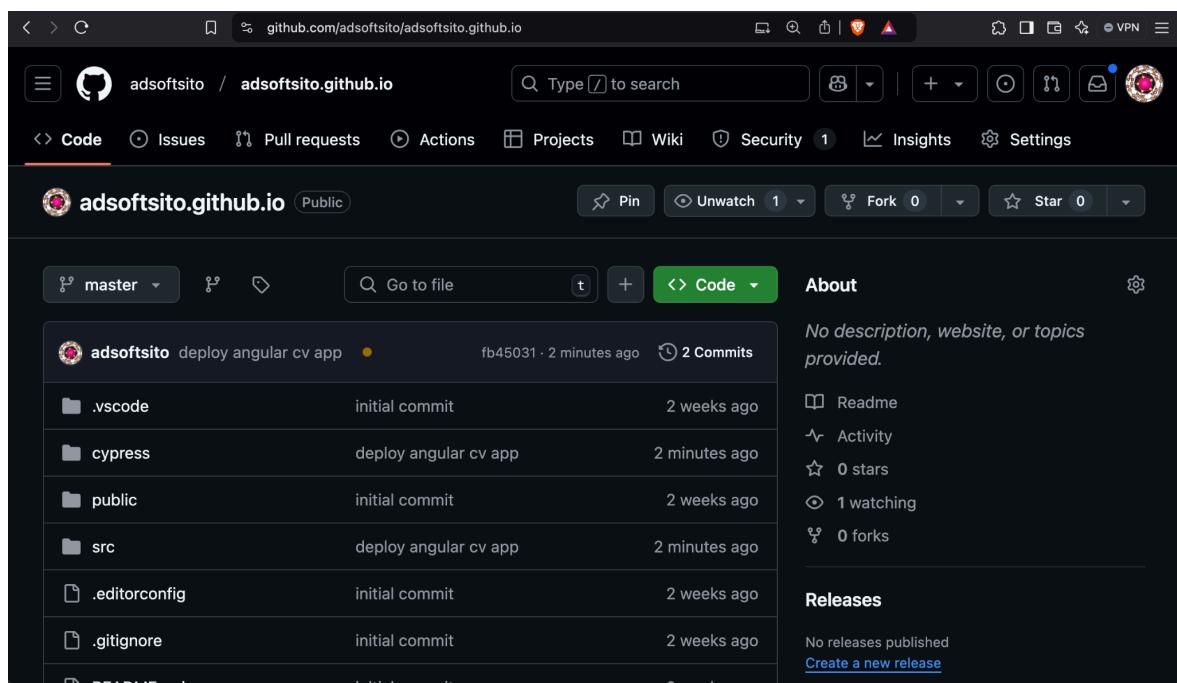
Como último paso, subimos nuestra versión de código a github.com en el branch master.

```
$ git push origin master
```

```
→ mycv git:(master) git push origin master
Enumerating objects: 117, done.
Counting objects: 100% (117/117), done.
Delta compression using up to 4 threads
Compressing objects: 100% (107/107), done.
Writing objects: 100% (117/117), 163.57 KiB | 4.81 MiB/s, done.
Total 117 (delta 27), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (27/27), done.
To https://github.com/adsoftsito/adsoftsito.github.io.git
 * [new branch] master -> master
→ mycv git:(master)
```

NOTA: En **Windows**, saldrá una pantalla de autenticación, puedes loguearte por web o token, de preferencia usar web. En **mac** y **linux**, pedirá usuario y contraseña, como usuario es el mismo de github y como password usar el token de github que creamos en el **tema 1**.

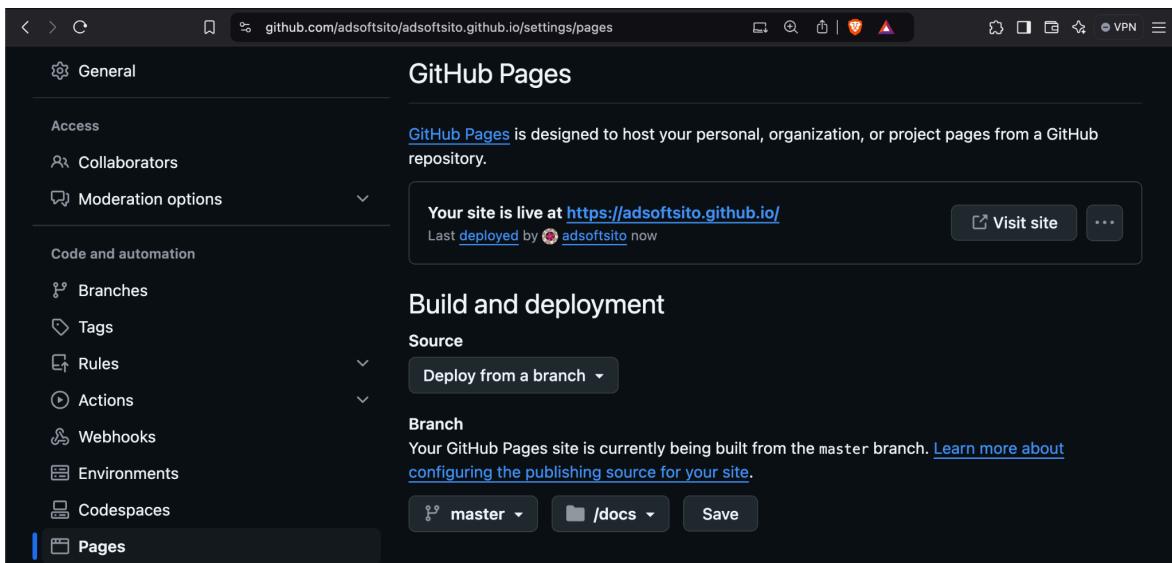
Nuestro repositorio en github debe lucir así con el código de nuestro proyecto Angular:



Configuramos github pages.

Probaremos desplegar nuestro proyecto angular en el servicio de hosting gratuito de Github.

Ir a settings -> pages



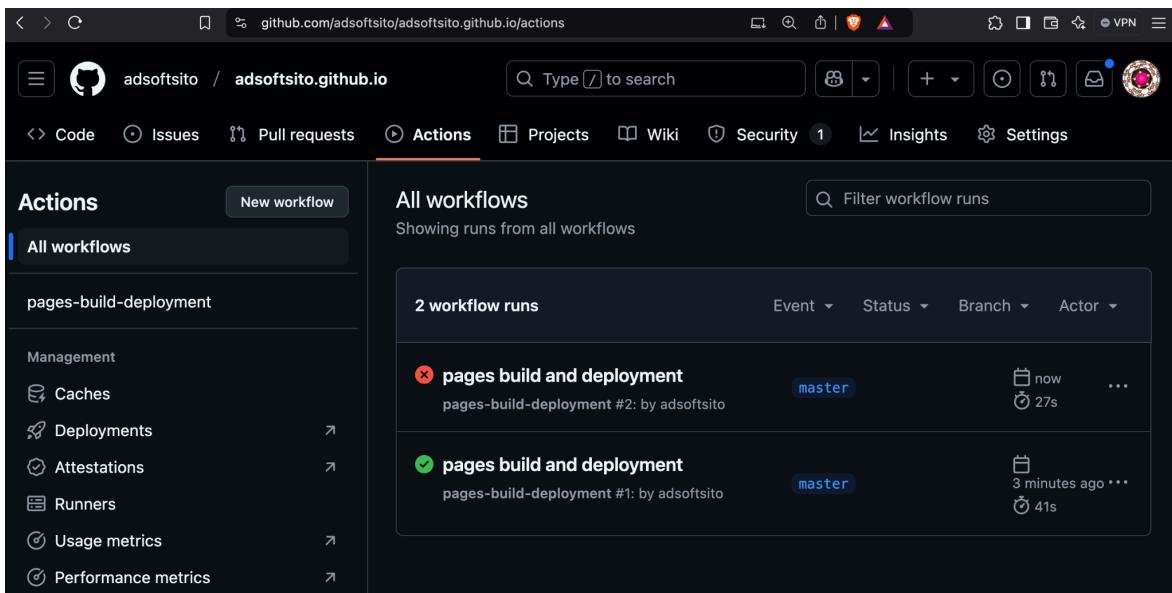
The screenshot shows the GitHub Pages settings interface. On the left, there's a sidebar with options like General, Access, Collaborators, Moderation options, Code and automation, Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, and Pages. The Pages option is selected and highlighted with a blue bar at the bottom. The main content area is titled "GitHub Pages". It says "GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository." Below that, it shows "Your site is live at <https://adsoftsito.github.io/>" and "Last deployed by adsoftsito now". There's a "Visit site" button and a three-dot menu. Under "Build and deployment", there's a "Source" dropdown set to "Deploy from a branch", a "Branch" dropdown set to "master", and a "Save" button.

Configuramos el **branch master** como la fuente del despliegue, y la carpeta **docs** (donde se encuentra el código html, css y js)

Presionar click en el botón **Save**, esto disparará un evento de despliegue automático de nuestro proyecto en el hosting de Github Pages, podemos visualizarlo en el menú **Actions**.

Ahora, hacemos click en menú **Actions**

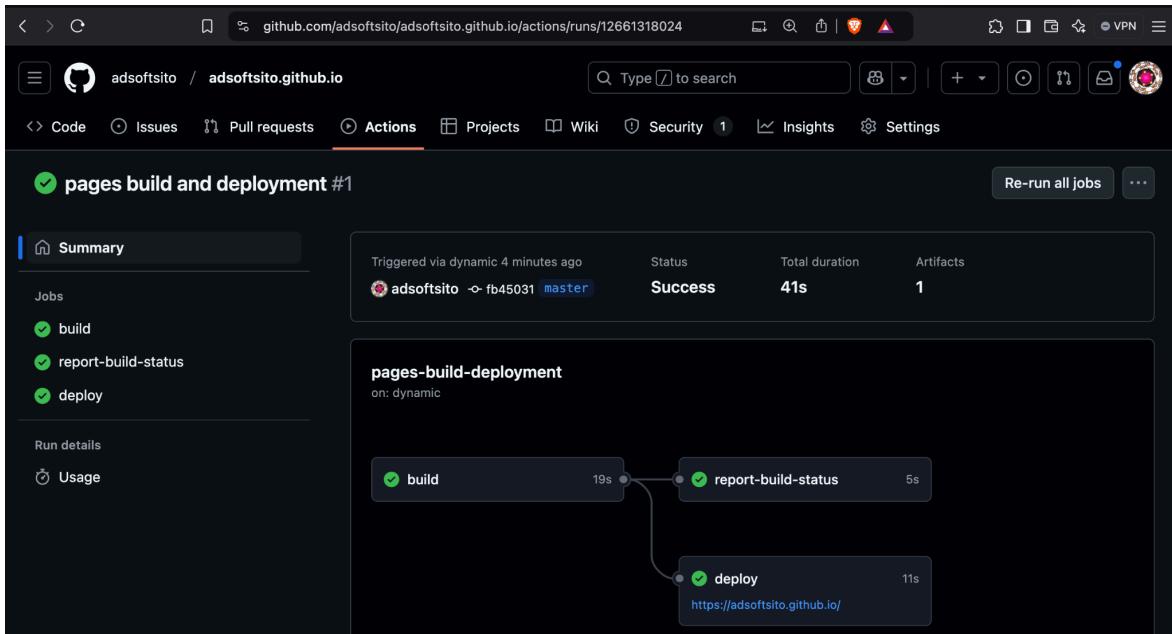
Plantilla Nivel de Conocimiento



The screenshot shows the GitHub Actions interface for the repository 'adsoftsito/adsoftsito.github.io'. The 'Actions' tab is selected. On the left, a sidebar lists various GitHub features like 'Management', 'Caches', 'Deployments', and 'Metrics'. The main area displays 'All workflows' with two runs listed:

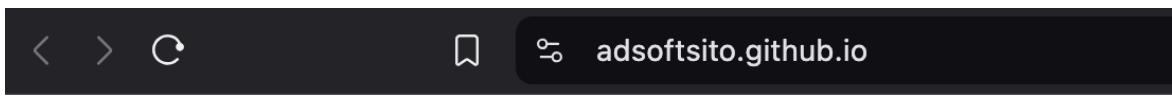
- pages-build-and-deployment #2:** Status: Failed (red), Triggered now, Duration: 27s. It failed at the 'build' step.
- pages-build-and-deployment #1:** Status: Success (green), Triggered 3 minutes ago, Duration: 41s. It completed successfully with steps: build (19s), report-build-status (5s), and deploy (11s) to <https://adsoftsito.github.io/>.

Podemos visualizar todos los workflows en ejecución o terminados. Para verificar que el despliegue damos click en el último workFlow, para inspeccionar que se completen correctamente las 3 tareas: **build, report-build-status y deploy**.



The screenshot shows the detailed view of the successful workflow run #1. The summary indicates it was triggered via dynamic 4 minutes ago, ran on branch master, and completed with a success status in 41s, producing 1 artifact. The workflow definition 'pages-build-deployment' on 'dynamic' includes three jobs: build, report-build-status, and deploy. The 'build' job took 19s, 'report-build-status' took 5s, and 'deploy' took 11s, resulting in the URL <https://adsoftsito.github.io/>.

Verificamos que se haya completado con éxito. En el nodo **deploy** está el url de nuestro proyecto en angular en producción.

			
Adolfo Centeno Ser ingeniero de software		adsoft@live.com.mx 2721908413 Orizaba Ver. Mexico @adsoft.sito	
<ul style="list-style-type: none">• nov-2023 - nov-2024 <i>Puebla, Mexico</i> Backend Developer kubeet SA de CV Logros: graphql api, REST api• ene-2020 - oct-2023 <i>Orizaba, Mexico</i> Frontend Developer Waves Lab Logros: web plataform, android app, ios app		skills works!	
education works!		certificates works!	
languages works!		interests works!	

Ya tienes algunas herramientas que te permitirán construir la versión distribuible de tu proyecto Angular, crear un repositorio en github, subir el código y publicarlo en el hosting gratuito de Github Pages.

Sala de inspiración

El conocimiento que has adquirido hasta ahora, te permite publicar tu proyecto en un hosting gratuito de github Pages

- ¿Podrías publicarlo en otra nube que conozcas?
- ¿Crees que el html, css y js creado por npm run build sea compatible con cualquier hosting o nube comercial?

Sala de pruebas

Lee cada una de las preguntas y selecciona la respuesta que consideres correcta de acuerdo a tu experiencia hasta ahora.

1. *Comando de angular CLI para generar la versión distribuible en la carpeta www*

Opción a: <i>ng build</i>	Incorrecta: <i>El comando de Angular CLI correcta para generar la versión distribuible de un proyecto en una carpeta www es:</i> <i>ng build --output-path=www</i>
Opción b: <i>ng build --output-dir=www</i>	Incorrecta: <i>El comando de Angular CLI correcta para generar la versión distribuible de un proyecto en una carpeta www es:</i>

	<i>ng build --output-path=www</i>
Opción c: <i>ng build --path=www</i>	Incorrecta: <i>El comando de Angular CLI correcta para generar la versión distribuible de un proyecto en una carpeta www es:</i> <i>ng build --output-path=www</i>
Opción d: <i>ng build --output-path=www</i>	CORRECTA: <i>El comando de Angular CLI correcta para generar la versión distribuible de un proyecto en una carpeta www es:</i> <i>ng build --output-path=www</i>

2. Comando en git para agregar el **origin** de un repositorio cuyo url es : <https://github.com/adsoftsito/angular18demo.git>

Opción a: git remote origin https://github.com/adsoftsito/angular18demo.git	Incorrecta El comando git correcta para agregar un repositorio a una carpeta local es: git remote add origin https://github.com/adsoftsito/angular18demo.git
Opción b: git add origin https://github.com/adsoftsito/angular18demo.git	Incorrecta El comando git correcta para agregar un repositorio a una carpeta local es: git remote add origin https://github.com/adsoftsito/angular18demo.git
Opción c:	

Plantilla Nivel de Conocimiento

	CORRECTA
<pre>git remote add origin https://github.com/adsoftsito/angular18demo.git</pre>	<p>El comando git correcta para agregar un repositorio a una carpeta local es:</p> <pre>git remote add origin https://github.com/adsoftsito/angular18demo.git</pre>
Opción d: <pre>git origin https://github.com/adsoftsito/angular18demo.git</pre>	<p>Incorrecta:</p> <p>El comando git correcta para agregar un repositorio a una carpeta local es:</p> <pre>git remote add origin https://github.com/adsoftsito/angular18demo.git</pre>

3.- Comando git para agregar todos los archivos pendientes de subir al repositorio.

Opción a: <pre>git add.</pre>	<p>Incorrecta:</p> <p>El comando git correcto para agregar todos los archivos pendientes es:</p> <p><i>git add .</i></p>
Opción b: <pre>git add *</pre>	<p>Incorrecta:</p> <p>El comando git correcto para agregar todos los archivos pendientes es:</p> <p><i>git add .</i></p>
Opción c: <p><i>git add .</i></p>	<p style="text-align: center;">CORRECTA</p> <p>El comando git correcto para agregar todos los archivos pendientes es:</p> <p><i>git add .</i></p>

<p>Opción d:</p> <pre>git add</pre>	<p>Incorrecta:</p> <p>El comando git correcto para agregar todos los archivos pendientes es:</p> <p><i>git add .</i></p>
--	---

SUBTEMA 2: Workflows en github para despliegue continuo

Docker es una plataforma de código abierto para desarrollar aplicaciones en entornos virtuales ligeros se conocen como contenedores, en esta sección aprenderemos a instalar y configurar Docker con el propósito de crear una imagen de Docker personalizado donde podamos distribuir nuestra aplicación en cualquier nube comercial que soporte Docker. Posteriormente automatizamos este proceso usando workflows de github.

Instalación de Docker.

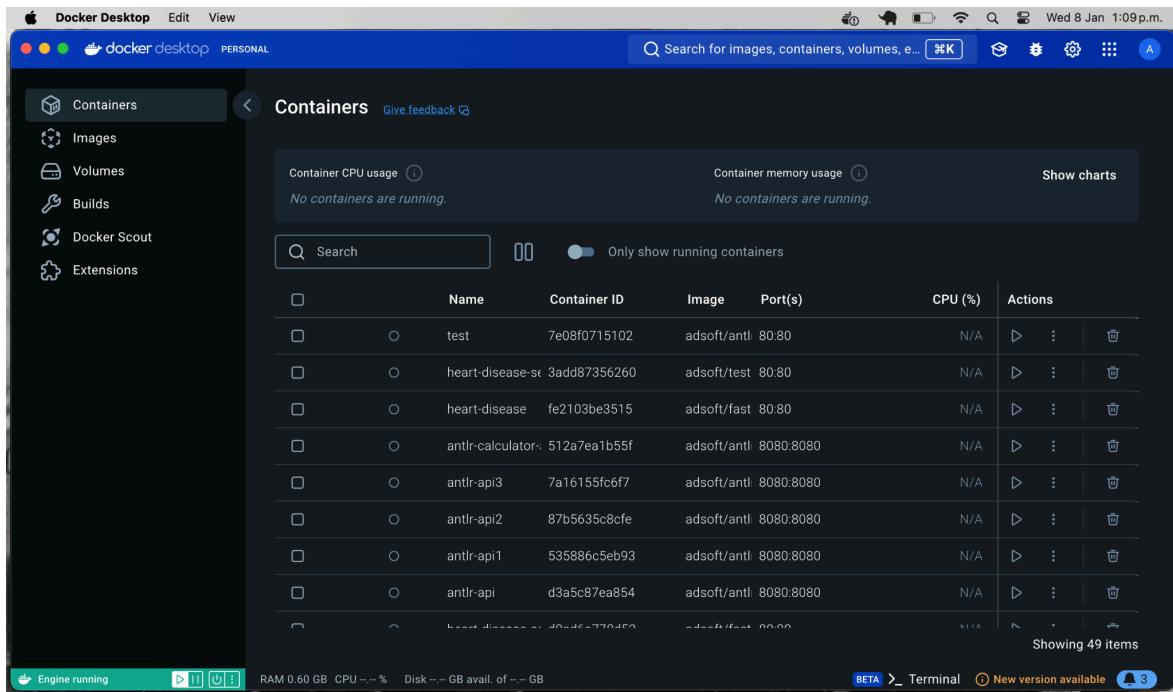
windows	<p>1.- Habilitar WSL 2 y Hyper-V: Abrimos el Panel de Control -> Programas -> Programas y Características -> Turn Windows features on or off. Check en “Hyper-V” y “Windows Subsystem for Linux.” Click OK y reiniciar windows.</p> <p>2.- Instalar WSL, abrir PowerShell como Administrador y ejecuta el comando: wsl --install</p> <p>Este comando instalará WSL 2 y requiere reiniciar windows</p> <p>3.- Descargar Docker Desktop: Docker Desktop for Windows - x86_64</p>
mac	<p>1.- Instalar Docker Desktop para Intel: https://desktop.docker.com/mac/main/amd64/Docker.dmg?utm_source=docker&utm_medium=webreferral&utm_campaign=docs-driven-download-mac-amd64</p>

	<p>o bien</p> <p>2.- Instalar Docker Desktop para ARM: https://desktop.docker.com/mac/main/arm64/Docker.dmg?utm_source=docker&utm_medium=webreferral&utm_campaign=docs-driven-download-mac-arm64</p>
linux	<p>Debian/Ubuntu</p> <pre>\$ sudo apt-get update \$ sudo apt-get install ca-certificates curl \$ sudo install -m 0755 -d /etc/apt/keyrings \$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc \$ sudo chmod a+r /etc/apt/keyrings/docker.asc \$ echo \ "deb [arch=\$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \ \$(. /etc/os-release && echo "\$VERSION_CODENAME") stable" \ sudo tee /etc/apt/sources.list.d/docker.list > /dev/null \$ sudo apt-get update \$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin</pre> <p>Redhat/Fedora/Centos</p> <pre>\$ sudo dnf -y install dnf-plugins-core \$ sudo dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo \$ sudo dnf install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin \$ sudo systemctl enable --now docker</pre>



Probar la instalación de Docker.

En caso de Windows y Mac asegurarse que Docker Desktop está iniciado



Probamos creando un contenedor con una imagen llamada: **hello_world**, ya disponible Docker Hub.

\$ sudo docker run hello-world

```
Last login: Wed Jan  8 10:08:35 on console
↳ ~ sudo docker run hello-world
Password:

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Si tenemos la salida anterior, nuestro motor de Docker está funcionando correctamente.

Principales comandos de Docker

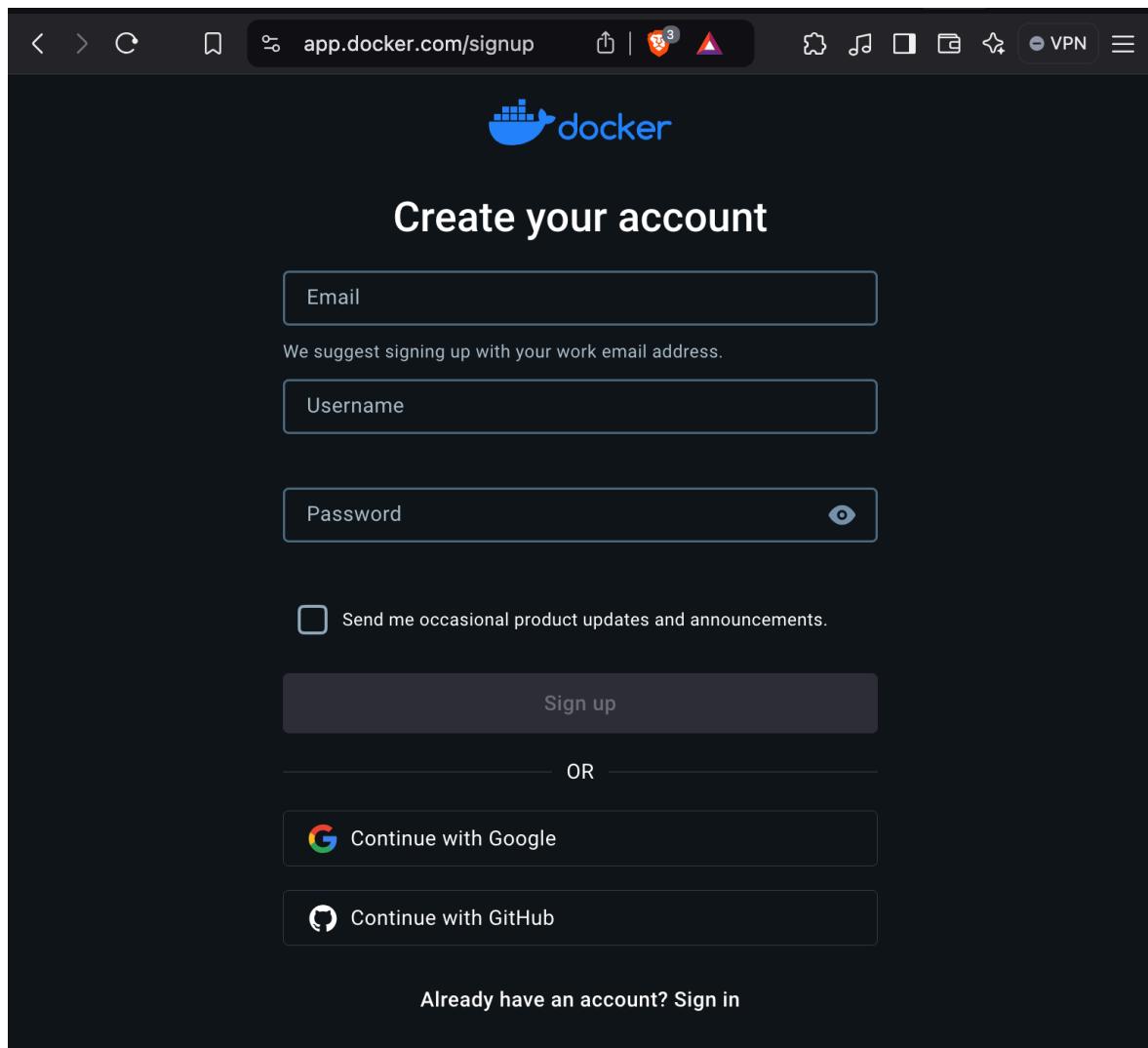
docker pull	descarga una imagen del contenedor de imágenes Docker Hub
docker ps	muestra la lista de contenedores activos
docker ps -a	lista los contenedores activos e inactivos
docker images	lista las imágenes docker descargadas
docker run	crea un contenedor nuevo a partir de una imagen
docker kill	elimina un contenedor activo
docker start	arranca un contenedor que exista pero este inactivo
docker stop	detiene un contenedor activo, pero no lo elimina
docker cp	copia un conjunto de archivos del host a un contenedor activo
docker commit	crea una nueva imagen a partir de un contenedor activo

docker exec	ejecuta un comando sobre un contenedor activo
docker login	hace login con las credenciales de Docker Hub
docker push	si estamos logueados, sube una imagen local a Docker Hub

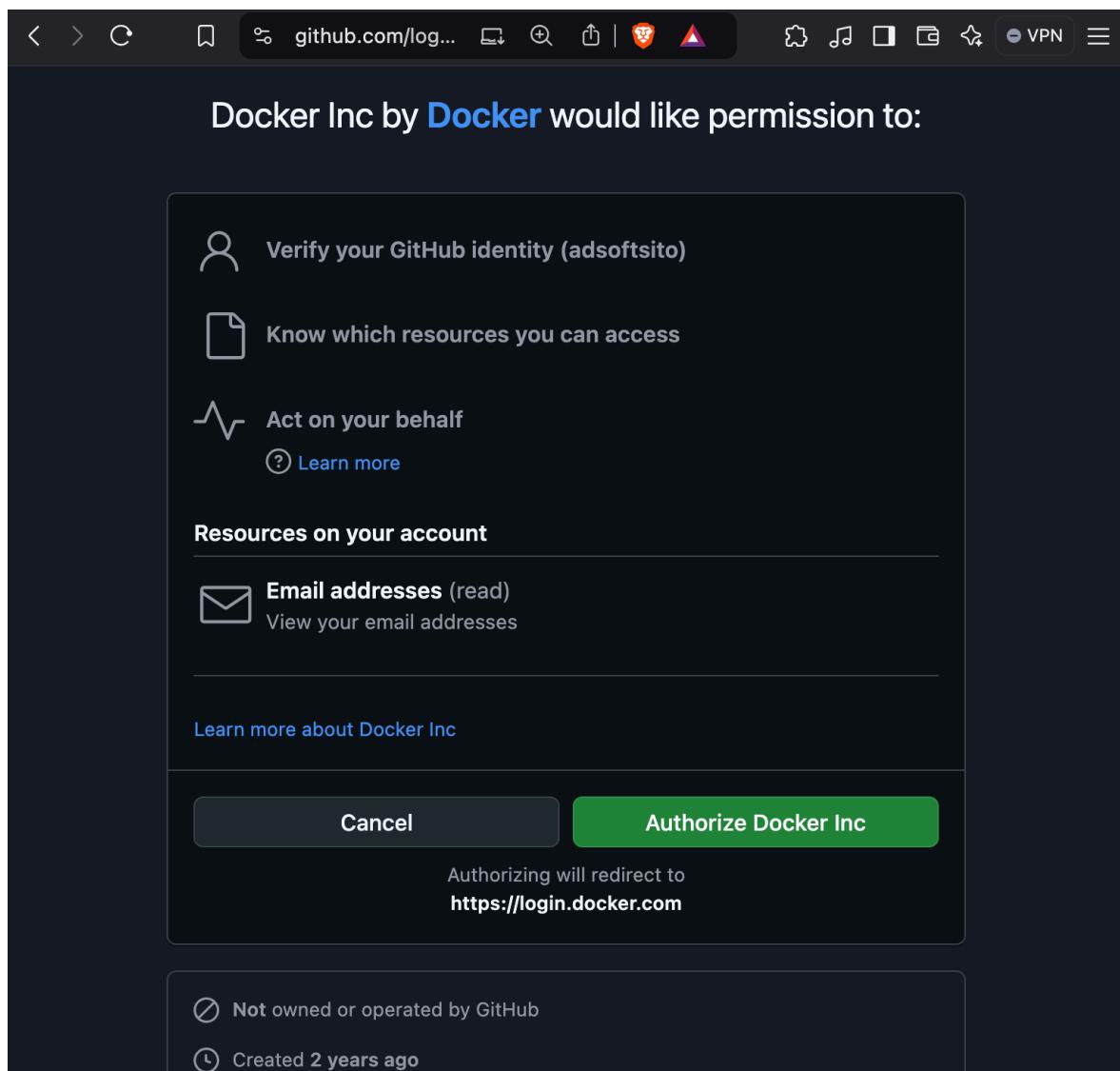
Docker Hub

Docker Hub es un registro de imágenes de contenedores que permite a los desarrolladores y contribuidores de código abierto encontrar, usar y compartir sus imágenes de contenedores. Es la biblioteca más grande de imágenes de contenedores del sector IT, alojando una gran cantidad de imágenes oficiales y de proyectos de código abierto. Docker Hub ofrece tanto repositorios públicos como privados, permitiendo a los usuarios compartir imágenes con la comunidad o mantenerlas para uso exclusivo. Además, se integra con sistemas de CI/CD para automatizar el proceso de construcción y despliegue de imágenes de contenedores.

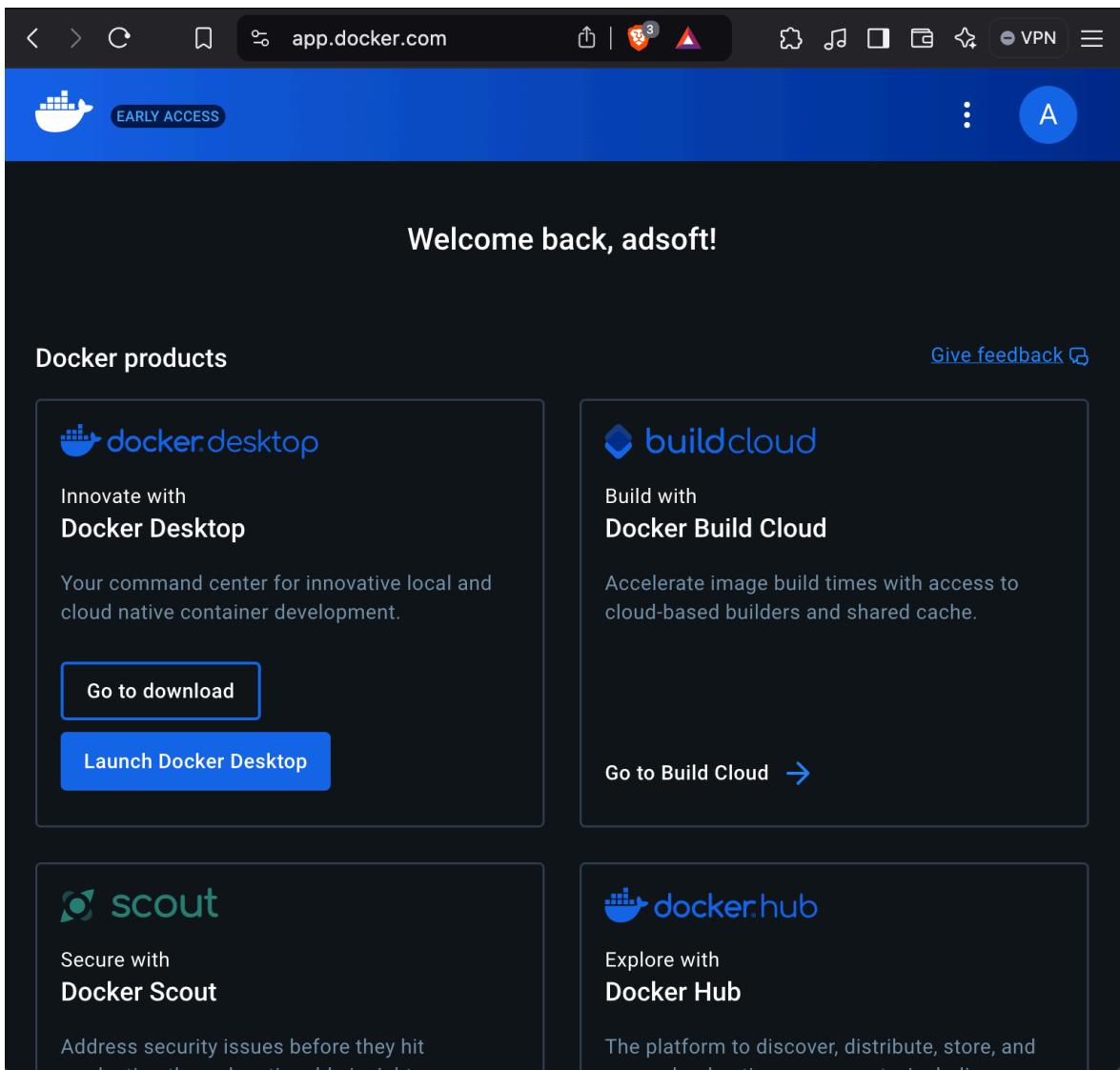
Creamos una cuenta en Docker Hub en <https://app.docker.com/signup>



Seleccionamos **Continue with Github**.



Autorizamos en **Authorize Docker Inc.**



The screenshot shows a web browser window with the URL app.docker.com in the address bar. The page is titled "Welcome back, adsoft!" and features a "Give feedback" button. It displays four main product cards:

- Docker desktop**: Innovate with Docker Desktop. Your command center for innovative local and cloud native container development. Includes "Go to download" and "Launch Docker Desktop" buttons.
- buildcloud**: Build with Docker Build Cloud. Accelerate image build times with access to cloud-based builders and shared cache. Includes a "Go to Build Cloud" button.
- scout**: Secure with Docker Scout. Address security issues before they hit production through actionable insights across your entire application stack.
- docker hub**: Explore with Docker Hub. The platform to discover, distribute, store, and reuse cloud-native components, including Docker images, Dockerfiles, and Docker Compose files.

Click en Docker Hub, para ir al dashboard de **repositorios**.

Name	Last Pushed	Contains	Visibility	Size
adsoft/tensorflow-flowers-model	about 2 months ago	IMAGE	Public	1.1GB
adsoft/tensorflow-linear-model	about 2 months ago	IMAGE	Public	1.1GB
adsoft/antlr-python	about 2 months ago	IMAGE	Public	1.1GB
adsoft/testmodel	2 months ago	IMAGE	Public	1.1GB
adsoft/heartmodel	2 months ago	IMAGE	Public	1.1GB

[Create a repository](#)

[Create an organization](#)

[Create and manage users and grant](#)

Creación de un contenedor Docker localmente.

Como primer paso, descargamos la imagen oficial del servidor web Apache llamada **httpd**. Este servidor lo usaremos para crear nuestro servidor web en un contenedor.

\$ docker pull httpd

```
|> ~ docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
fd674058ff8f: Pull complete
8c3081b233c7: Pull complete
4f4fb700ef54: Pull complete
172b239db5c2: Pull complete
bbff13f6be42: Pull complete
d7382fd3e491: Pull complete
Digest: sha256:72f6e24600718dddef131de7cb5b31496b05c5af41e9db8707df371859a350bb
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview h
  ttpd
|> ~
```

Verificamos que la imagen de Apache, esté guardada localmente.

\$ docker images httpd

```
[→ ~ docker images httpd
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
httpd          latest   4ce47c750a58  5 months ago  147MB
→ ~ ]
```

Ahora, creamos un contenedor llamada **http_base**, que estará disponible en el puerto 8080 y tomamos como base la imagen **httpd**.

```
$ docker run -d --name http_base -p 8080:80 httpd
[→ ~ docker run -d --name http_base -p 8080:80 httpd
edf46bb283029c708b26b75c18a56c36e5dadc96824364522f0331a0706a7a88
```

Verificamos que nuestro contenedor esté en ejecución.

```
$ docker ps
[→ ~ docker ps
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
edf46bb28302   httpd     "httpd-foreground"  14 seconds ago  Up 13 seconds  0.0.0.0:8080->80/tcp  http_base
```

Probamos la ejecución del contenedor localmente.

```
$ curl localhost:8080
[→ ~ curl localhost:8080
<html><body><h1>It works!</h1></body></html>
→ ~ ]
```

También podemos probar nuestro contenedor en el navegador en la dirección local: **localhost:8080**



It works!

En la raíz de nuestro proyecto Angular, verificamos la carpeta **docs** de nuestro proyecto tenga nuestros archivos **html, css y js**

\$ **ls -la docs**

```
[→ mycv git:(master) ls -la docs
total 1656
drwxr-xr-x 11 adsoft staff 352 Jan 7 18:03 .
drwxr-xr-x 23 adsoft staff 736 Jan 7 18:02 ..
-rw-r--r-- 1 adsoft staff 20778 Jan 7 18:02 3rdpartylicenses.txt
drwxr-xr-x 2 adsoft staff 64 Jan 7 18:03 browser
-rw-r--r-- 1 adsoft staff 173656 Jan 7 18:02 chunk-4ZHOUZTK.js
-rw-r--r-- 1 adsoft staff 29 Jan 7 18:02 chunk-DT56ZVVI.js
-rw-r--r-- 1 adsoft staff 15086 Jan 7 18:02 favicon.ico
-rw-r--r-- 1 adsoft staff 528 Jan 7 18:02 index.html
-rw-r--r-- 1 adsoft staff 581929 Jan 7 18:02 main-DHG2USRS.js
-rw-r--r-- 1 adsoft staff 34519 Jan 7 18:02 polyfills-FFHMD2TL.js
-rw-r--r-- 1 adsoft staff 0 Jan 7 18:02 styles-5INURTS0.css
```

Copiamos todos los archivos de nuestro proyecto al contenedor en ejecución **http_base** en la carpeta **/usr/local/apache2/htdocs**, en esta ruta el servidor web Apache almacena el sitio web a visualizar.

\$ **docker cp docs/. http_base:/usr/local/apache2/htdocs**

```
[→ mycv git:(master) docker cp docs/. http_base:/usr/local/apache2/htdocs
Successfully copied 835kB to http_base:/usr/local/apache2/htdocs
```

Verificamos que el contenedor en ejecución, ya contenga los archivos del proyecto. Ejecutamos **docker exec** con el comando **ls -la /usr/local/apache2/htdocs** sobre el contenedor activo **http_base**.

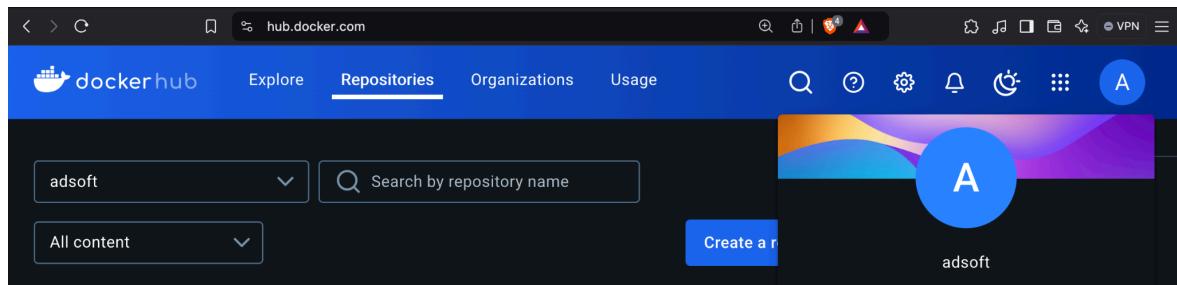
\$ **docker exec http_base ls -la /usr/local/apache2/htdocs**

```
[→ mycv git:(master) docker exec http_base ls -la /usr/local/apache2/htdocs
total 844
drwxr-xr-x 1 root root 4096 Jan 8 19:43 .
drwxr-xr-x 1 www-data www-data 4096 Dec 24 22:18 ..
-rw-r--r-- 1 501 dialout 20778 Jan 8 00:02 3rdpartylicenses.txt
drwxr-xr-x 2 501 dialout 4096 Jan 8 00:03 browser
-rw-r--r-- 1 501 dialout 173656 Jan 8 00:02 chunk-4ZHOUZTK.js
-rw-r--r-- 1 501 dialout 29 Jan 8 00:02 chunk-DT56ZVVI.js
-rw-r--r-- 1 501 dialout 15086 Jan 8 00:02 favicon.ico
-rw-r--r-- 1 501 dialout 528 Jan 8 00:02 index.html
-rw-r--r-- 1 501 dialout 581929 Jan 8 00:02 main-DHG2USRS.js
-rw-r--r-- 1 501 dialout 34519 Jan 8 00:02 polyfills-FFHMD2TL.js
-rw-r--r-- 1 501 dialout 0 Jan 8 00:02 styles-5INURTS0.css
```

Creamos una nueva imagen a partir del contenedor en ejecución, ya que contiene nuestro proyecto copiado. La nueva imagen está formada por:

usuario_dockerhub/nombreImagen:version

El usuario de Docker Hub podemos verlo en <https://hub.docker.com>



Ejecutamos docker commit con el parámetro **--change** y un mensaje entre “ ”, el nombre del contenedor activo, y nombre de la imagen nueva.

```
$ docker commit --change "ENV MODEL_NAME http_base" http_base
your-docker-user/mycv:latest
```

```
[→ mycv git:(master) docker commit --change "ENV MODEL_NAME http_base" http_base adsoft/mycv:latest
sha256:e02792c80b75dc23a0c85f009a9b3b940eb1253eee21ba984f4bd9658ce33667]
```

Verificamos la correcta creación de la nueva imagen.

```
$ docker images your-docker-user/mycv:latest
```

```
[→ mycv git:(master) docker images adsoft/mycv:latest
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
adsoft/mycv     latest   e02792c80b75   About a minute ago   148MB]
```

Probamos la nueva imagen, creando un nuevo contenedor en puerto 8081

```
$ docker run -dit --name angular-app -p 8081:80 -d your-docker-user/mycv:latest
```

```
[→ mycv git:(master) docker run -dit --name angular-app -p 8081:80 -d adsoft/mycv:latest
361e0f48154a553a1167952f4bfa53992bbc8d54ea671742bee2e8a1f1bd215b]
```

Verificamos que el contenedor nuevo llamado **angular-app**, esté en ejecución.

```
$ docker ps
```

```
[→ mycv git:(master) docker ps
CONTAINER ID  IMAGE          COMMAND           CREATED        STATUS         PORTS          NAMES
361e0f48154a  adsoft/mycv:latest  "httpd-foreground"  35 seconds ago  Up 35 seconds  0.0.0.0:8081->80/tcp  angular-app
10b4db745d4e  httpd          "httpd-foreground"  2 minutes ago   Up 2 minutes   0.0.0.0:8080->80/tcp  http_base]
```

Probamos el contenedor del puerto 8081

\$ curl localhost:8081

```
➜ mycv git:(master) curl localhost:8081
<!doctype html>
<html lang="en" data-critters-container>
<head>
  <meta charset="utf-8">
  <title>Mycv</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link rel="stylesheet" href="styles-5INURTSO.css"></head>
<body>
  <app-root></app-root>
  <link rel="modulepreload" href="chunk-4ZHOUZTK.js"><script src="polyfills-FFHMD2TL.js" type="module"></script><script src="main-DHG2USR.js" type="module"></script></body>
</html>
```

Probamos en el navegador, nuestro contenedor docker con nuestro proyecto Angular en ejecución.

Adolfo Centeno		adsoft@live.com.mx 2721908413 Orizaba Ver. Mexico @adsoft.sito
Ser ingeniero de software		
<ul style="list-style-type: none"> nov-2023 - nov-2024 <i>Puebla, Mexico</i> Backend Developer kubeet SA de CV Logros: graphql api, REST api ene-2020 - oct-2023 <i>Orizaba, Mexico</i> Frontend Developer Waves Lab Logros: web plataform, android app, ios app 	skills works!	
education works!	certificates works!	
languages works!	interests works!	

Ahora debemos publicar nuestra imagen Apache httpd modificada con nuestro proyecto Angular en Docker Hub, para realizar este proceso primero debemos hacer login. Usamos el usuario y contraseña de Docker Hub.

\$ docker login

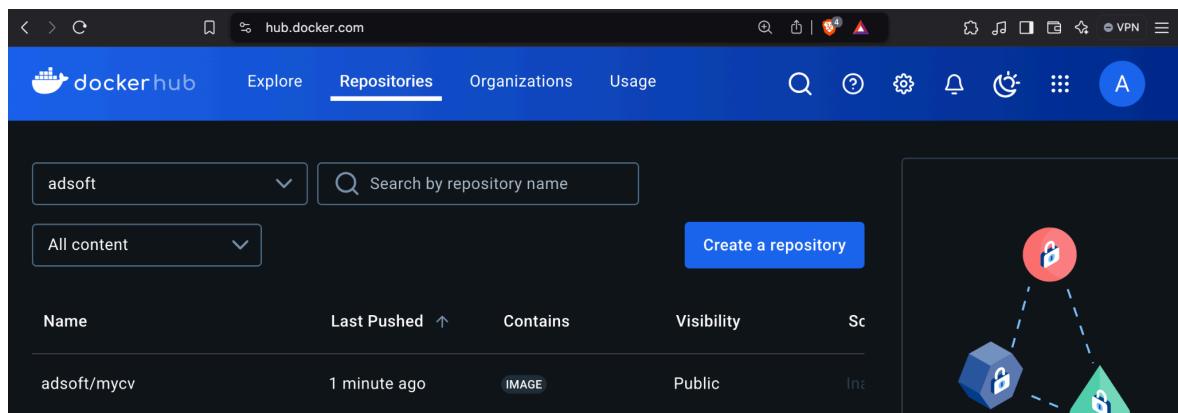
```
[→ mycv git:(master) docker login
Log in with your Docker ID or email address to push and pull images from Docker Hub. If you don't have a
Docker ID, head over to https://hub.docker.com/ to create one.
You can log in with your password or a Personal Access Token (PAT). Using a limited-scope PAT grants bett
er security and is required for organizations using SSO. Learn more at https://docs.docker.com/go/access-
tokens/
Username: adsoft
Password:
Login Succeeded
→ mycv git:(master) ]
```

Subimos la imagen del proyecto a **Docker Hub** con el comando **docker push**.

\$ docker push your-docker-user/mycv:latest

```
[→ mycv git:(master) docker push adsoft/mycv:latest
The push refers to repository [docker.io/adsoft/mycv]
6d47d65a4911: Pushed
52168ee29b83: Layer already exists
ca565a60a706: Layer already exists
5dceadbfb1901: Layer already exists
5f70bf18a086: Layer already exists
fa884c5dde25: Layer already exists
8b296f486960: Layer already exists
latest: digest: sha256:d2be27874c8be61647de81538d09c2ce363b217af52632fe8af786eee65e26f7 size: 1782
```

Verificamos en docker hub nuestra nueva imagen.



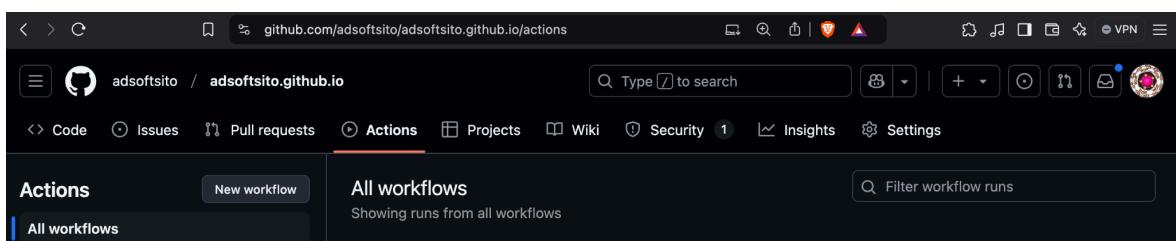
Despliegue continuo con Workflows

Los GitHub Workflows son una colección de acciones recurrentes que se ejecutan automáticamente en un repositorio de GitHub cuando ocurren ciertos eventos. Estos flujos de trabajo están compuestos por varios trabajos (jobs), que a su vez se dividen en una serie de pasos (steps). Los GitHub Workflows pueden configurarse para que se ejecuten de forma programada, en respuesta a eventos específicos o incluso de manera manual.

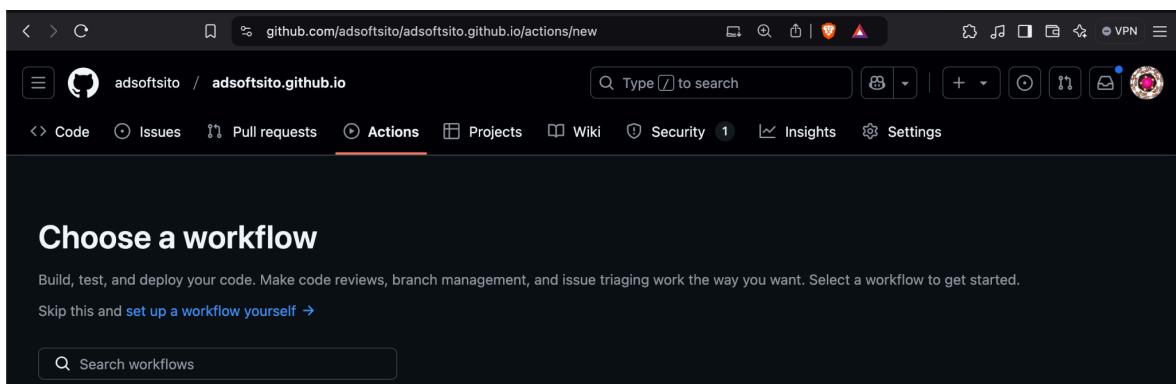
Los GitHub Workflows son una parte esencial de la plataforma de integración continua conocida como **GitHub Actions**, que permite automatizar tareas en el ciclo de vida de desarrollo de software, como la comprobación automática del código y la notificación de cambios en el repositorio.

Ingresamos a **github.com** y creamos un **Workflow** en nuestro repositorio.

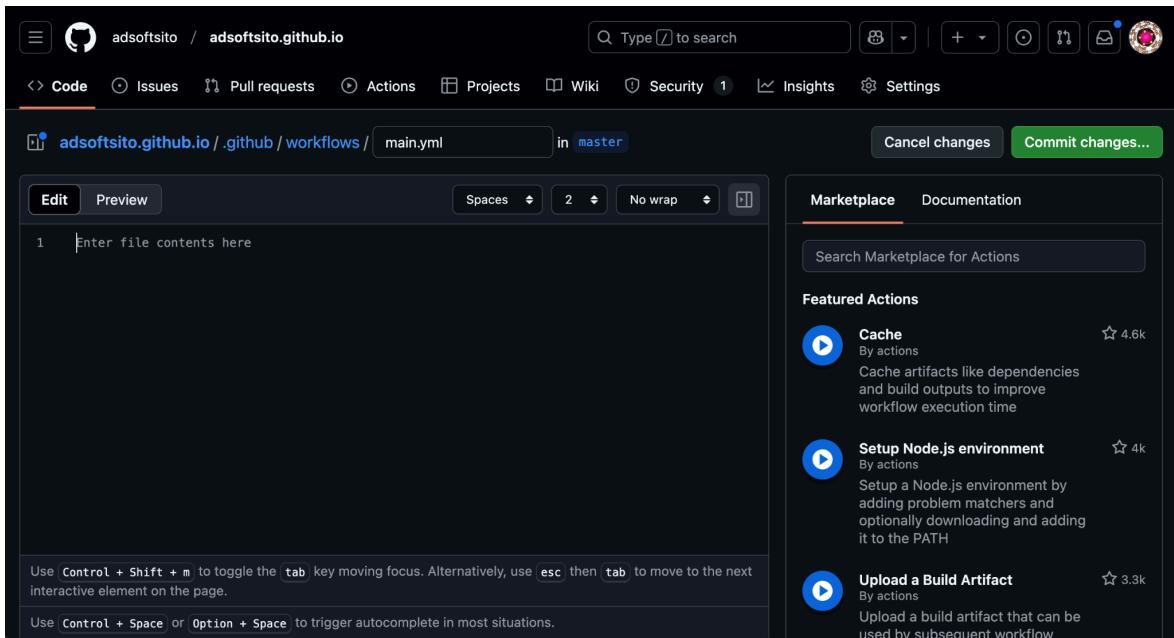
En el menú **Actions -> New Workflow**



Damos click en: **set up a workflow yourself.**



Aparecerá un editor para teclear el código completo de nuestro Workflow



Introducir el siguiente código con los scripts con cada uno de los pasos que realizamos manualmente, para que el Workflow las ejecute de forma automática cada vez que realicemos un cambio en el código.

NOTA: Este archivo se llamará main.yml, es un conjunto de scripts con formato python, por tanto debe respetarse la indentación para que no marque ningún error.

```
name: Angular_DockerImage
```

```
on:
```

```
push:
```

```
  branches: [ master ]
```

```
pull_request:
```

```
  branches: [ master ]
```

```
jobs:
```

```
angular_deploy :
```

```
  runs-on: ubuntu-latest
```

steps:

- uses: actions/checkout@v2

- name: Use Node.js 16.x

uses: actions/setup-node@v1

with:

node-version: 20.x

- name: npm install

run: npm ci

- name: test

run: |

npm test -- --no-watch --no-progress --browsers=ChromeHeadlessCI

- name: build

run: |

npm run build --output-path=docs

- name: docker login

env:

DOCKER_USER: \${{secrets.DOCKER_USER}}

DOCKER_PASSWORD: \${{secrets.DOCKER_PASSWORD}}

run: |

docker login -u \$DOCKER_USER -p \$DOCKER_PASSWORD

- name: Download and run the Docker base image

run: docker run -d --name http_base httpd

- name: copy model to the Docker image

```
run: docker cp docs/. http_base:/usr/local/apache2/htdocs
```

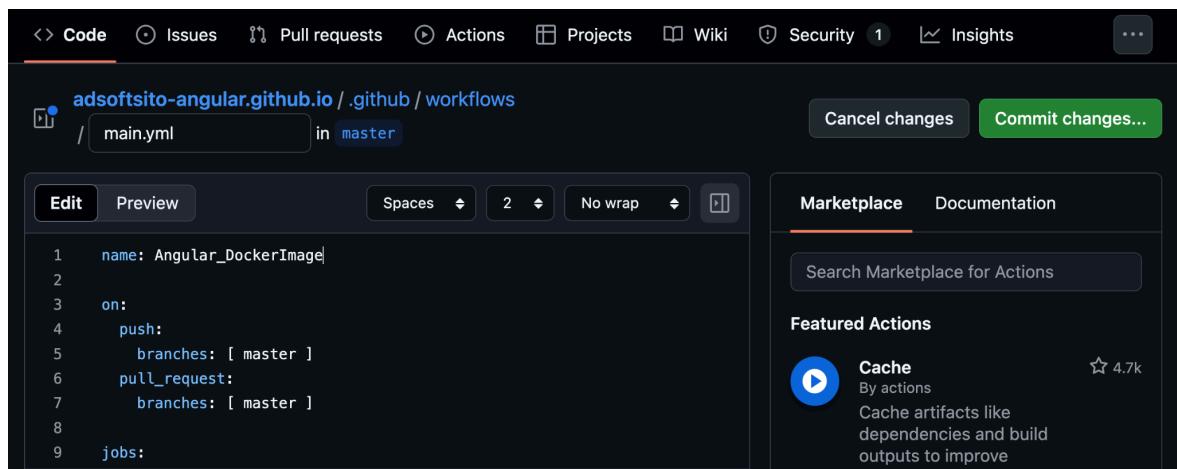
```
- name: Build the custom Docker image
```

```
run: docker commit --change "ENV MODEL_NAME http_base" http_base
${{secrets.DOCKER_USER}}/mycv:${{ github.sha }}
```

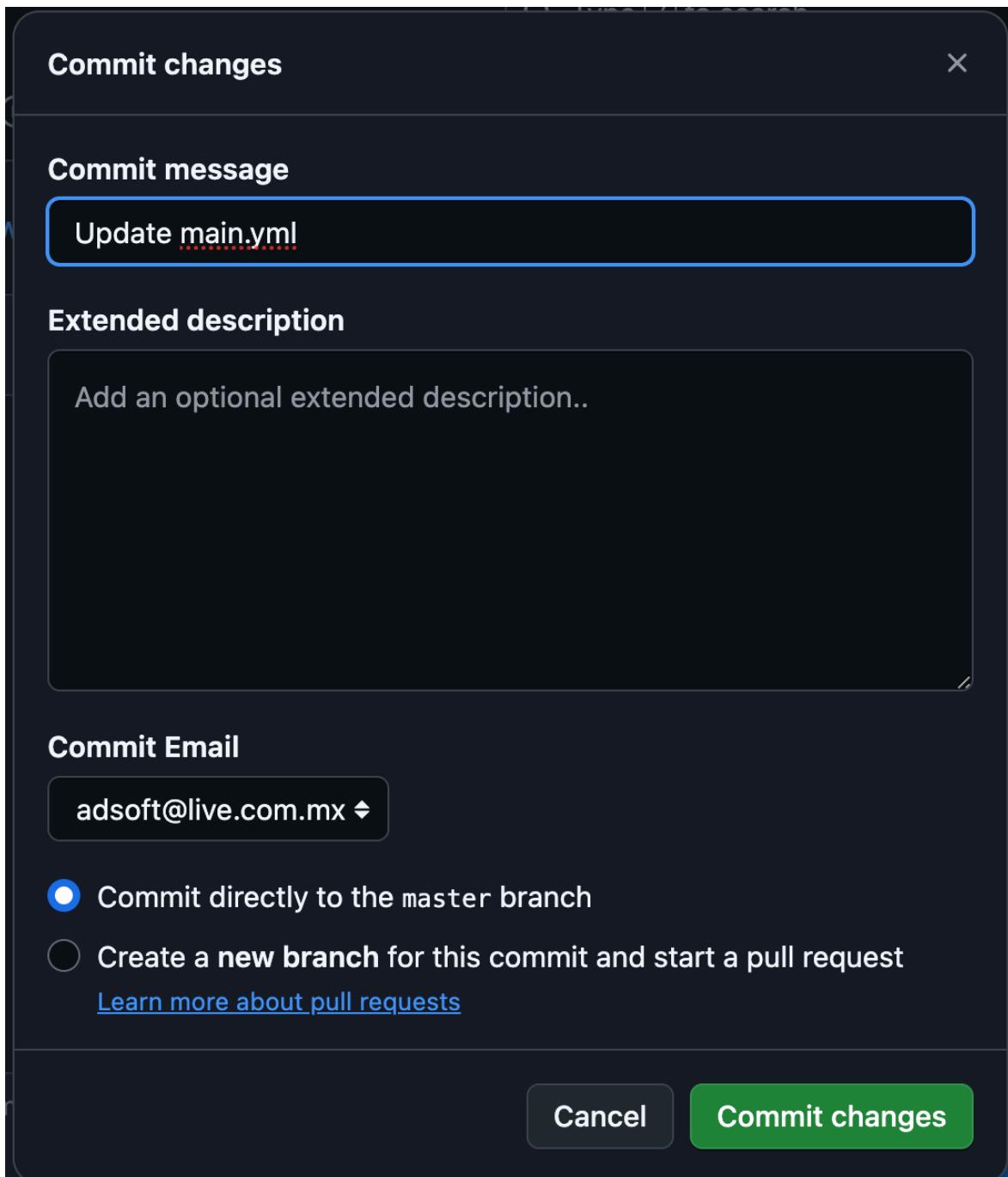
```
- name: Docker Push
```

```
run: docker push ${{secrets.DOCKER_USER}}/mycv:${{ github.sha }}
```

Para guardar, Clic en **Commit changes**.

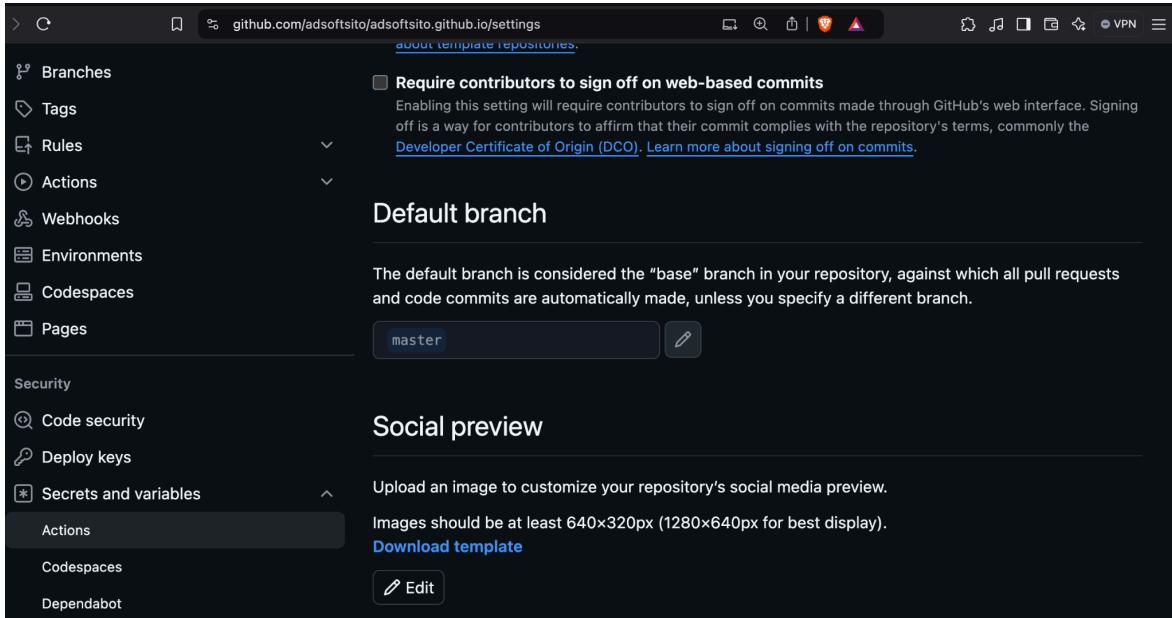


Click en **Commit changes** nuevamente.



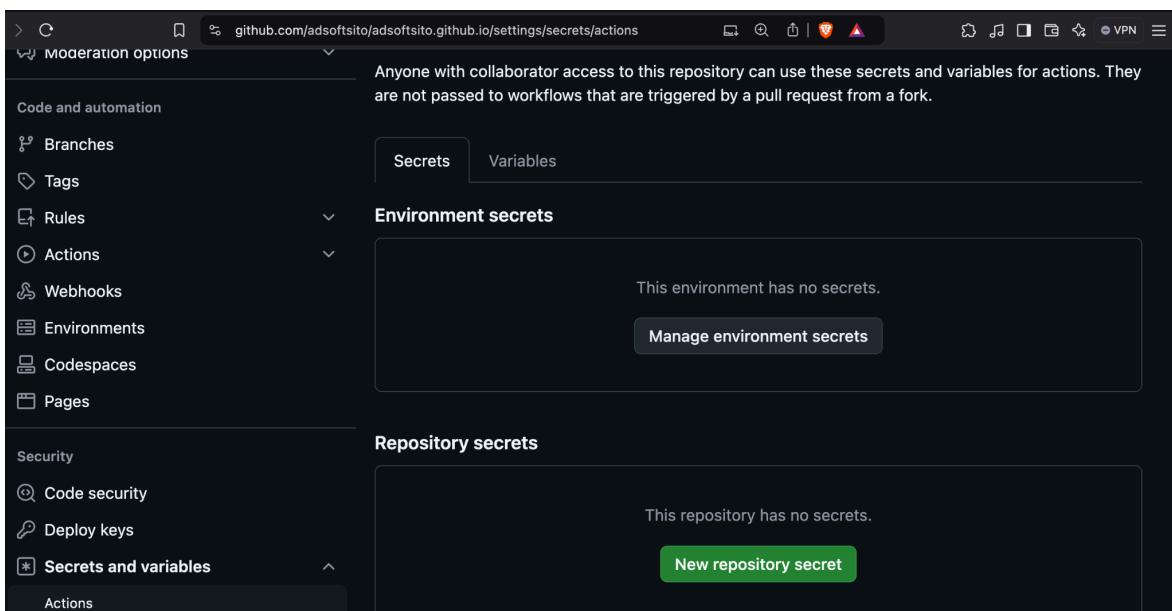
Para que nuestro workflow se ejecute con éxito, requiere crear 2 variables secretas llamadas **Secrets**, el usuario y password de nuestra cuenta de Docker Hub, esto es porque nuestro Workflow creará automáticamente una imagen con nuestro código HTML, CSS y JS, y la publicará en Docker Hub sin intervención humana por tanto debemos hacer login de forma automática.

Ir al menu **Settings -> Secrets and variables -> Actions**.



The screenshot shows the GitHub repository settings page for 'adsoftsito/adsoftsito.github.io'. The left sidebar includes options like Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages, Security, Code security, Deploy keys, Secrets and variables, Actions, Codespaces, and Dependabot. The main content area is titled 'Default branch' and shows the 'master' branch selected. Below it is the 'Social preview' section, which allows users to upload an image for their repository's social media preview. A button labeled 'Edit' is visible.

Posteriormente, click en **New repository secret**.



The screenshot shows the GitHub repository settings page for 'adsoftsito/adsoftsito.github.io' under the 'Secrets and variables' section. The left sidebar remains the same. The main content area is divided into two sections: 'Environment secrets' and 'Repository secrets'. Both sections indicate that they have no secrets. A green 'New repository secret' button is located in the 'Repository secrets' section.

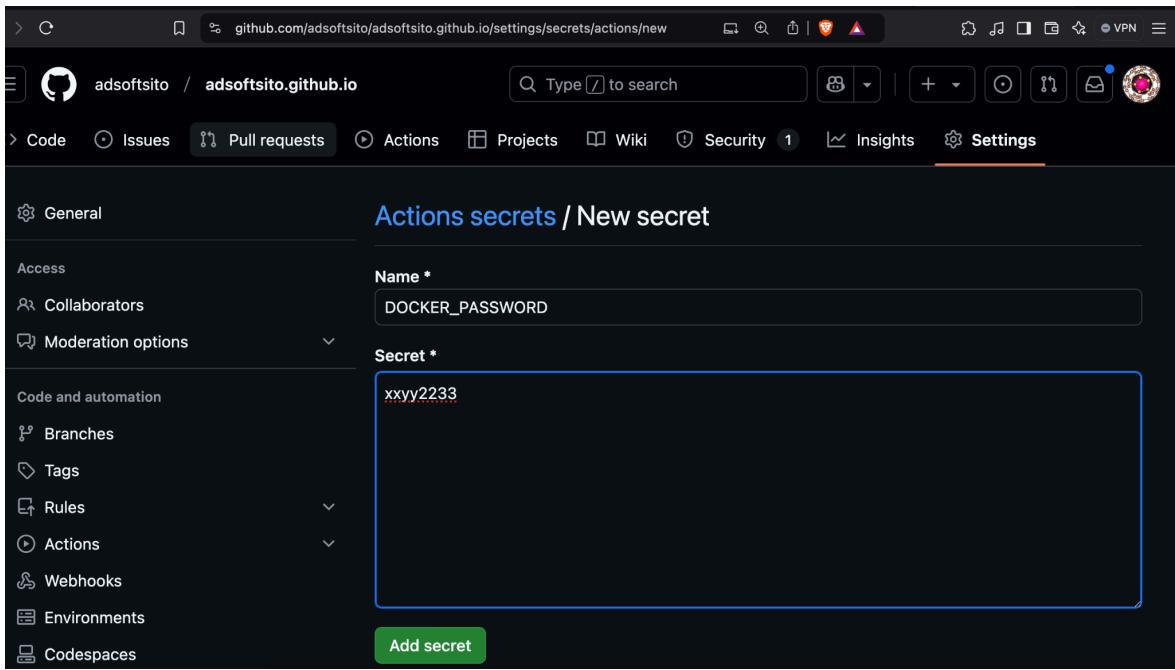
Creamos el secret **DOCKER_USER**

The screenshot shows the GitHub Actions secrets creation interface. On the left, there's a sidebar with options like General, Access, Collaborators, and Moderation options. Under Code and automation, there are branches, tags, rules, actions, webhooks, environments, and codespaces. The main area is titled "Actions secrets / New secret". It has fields for "Name *" (set to DOCKER_USER) and "Secret *" (containing the value adsoft). A green "Add secret" button is at the bottom.

Aparece nuestro nuevo secret en la lista de **Repository secrets**

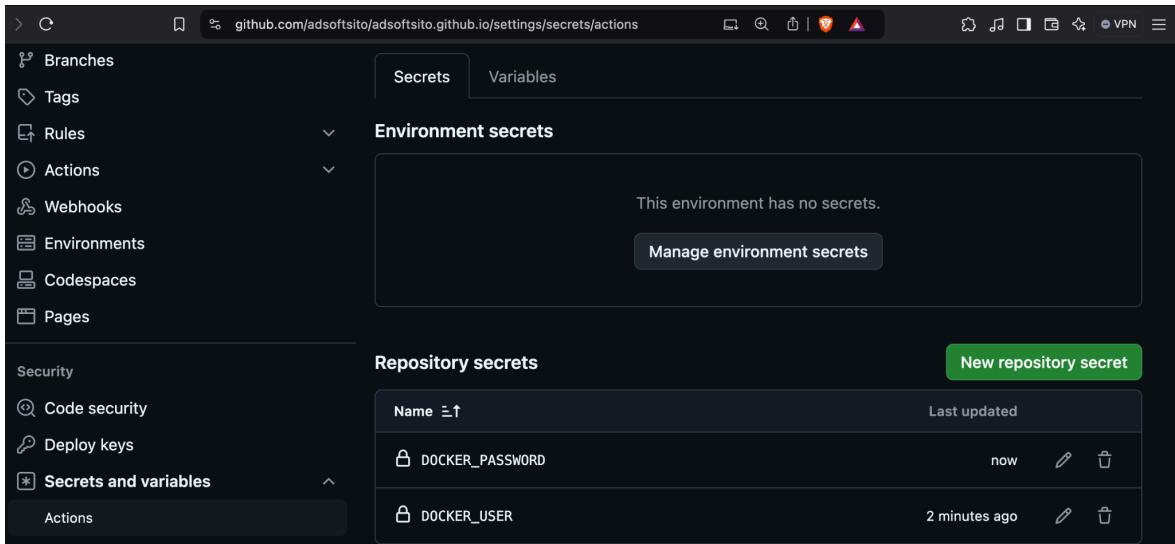
The screenshot shows the GitHub repository secrets list. The sidebar includes Rules, Actions, Webhooks, Environments, Codespaces, and Pages. Under Security, it lists Code security, Deploy keys, and Secrets and variables. The main area shows a section for "ENVIRONMENT SECRETS" which says "This environment has no secrets." and a "Manage environment secrets" button. Below this is a section for "Repository secrets" with a green "New repository secret" button. A table lists one secret: Name is DOCKER_USER, Last updated is now, and there are edit and delete icons.

Creamos ahora, el secret **DOCKER_PASSWORD**



The screenshot shows the GitHub Actions secrets creation interface. On the left, there's a sidebar with options like General, Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces), and Pages. The main area is titled "Actions secrets / New secret". It has fields for "Name *" (set to DOCKER_PASSWORD) and "Secret *" (containing the value xxxy2233). A green "Add secret" button is at the bottom.

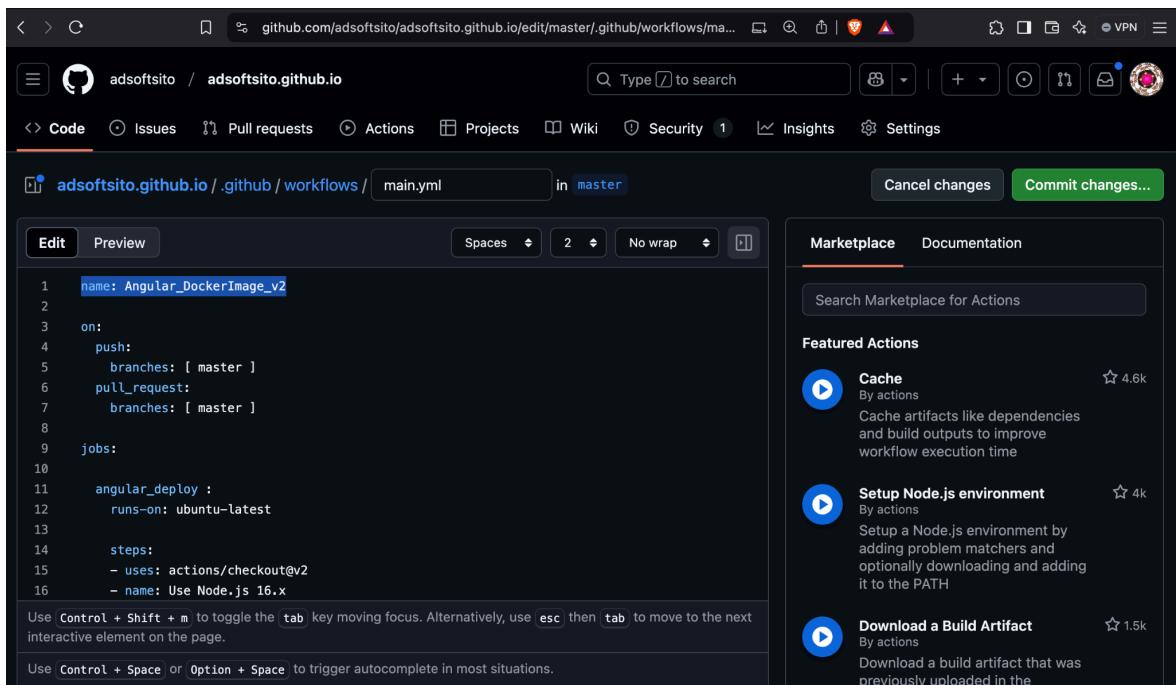
Tenemos 2 secrets en **Repository Secrets**.



The screenshot shows the GitHub Repository secrets list. The sidebar includes Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, and Pages. Under "Secrets and variables", there's a "Repository secrets" section. It lists two secrets: "DOCKER_PASSWORD" (created now, last updated now) and "DOCKER_USER" (created 2 minutes ago, last updated 2 minutes ago). There are "Edit" and "Delete" icons next to each entry. A green "New repository secret" button is visible at the top right of the secrets list.

Disparamos nuevamente nuestro Workflow, realizando cualquier cambio el código del Workflow que no afecte su funcionamiento. Ejemplo cambiar la primera línea:

name: Angular_DockerImage_v2



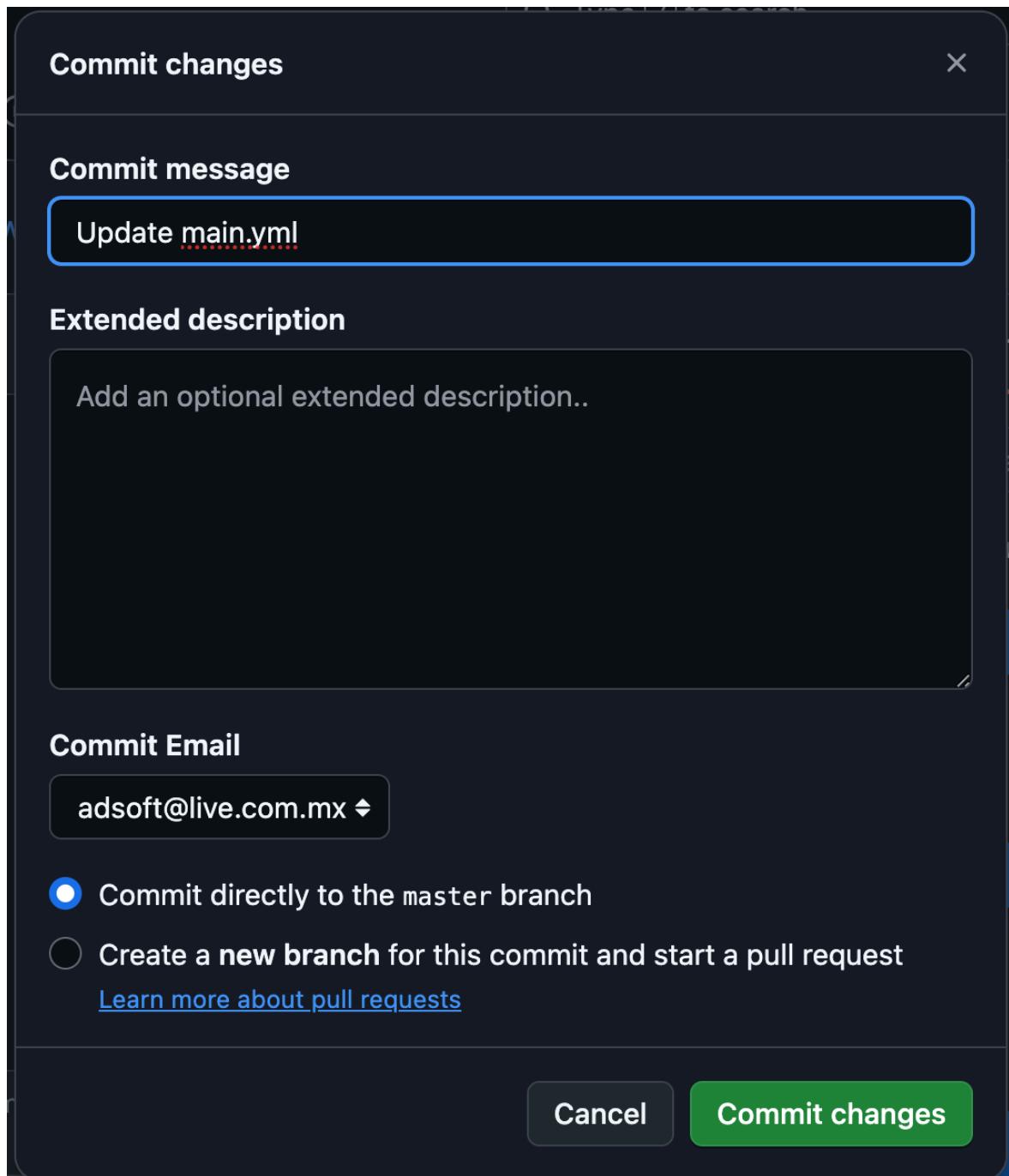
The screenshot shows a GitHub repository page for 'adsoftsito.github.io'. The user is editing the 'main.yml' file in the '.github/workflows' directory. The code in the editor is:

```
1 name: Angular_DockerImage_v2
2
3 on:
4   push:
5     branches: [ master ]
6   pull_request:
7     branches: [ master ]
8
9 jobs:
10
11   angular_deploy :
12     runs-on: ubuntu-latest
13
14     steps:
15       - uses: actions/checkout@v2
16       - name: Use Node.js 16.x
```

Below the editor, there are instructions for keyboard navigation and autocomplete. To the right of the editor is a sidebar titled 'Marketplace' which lists 'Featured Actions':

- Cache** By actions ★ 4.6k: Cache artifacts like dependencies and build outputs to improve workflow execution time.
- Setup Node.js environment** By actions ★ 4k: Setup a Node.js environment by adding problem matchers and optionally downloading and adding it to the PATH.
- Download a Build Artifact** By actions ★ 1.5k: Download a build artifact that was previously uploaded in the

Click en Commit changes.



Click en Commit changes nuevamente.

Verificar que el Workflow se dispare en el Menú **Actions**.

Plantilla Nivel de Conocimiento

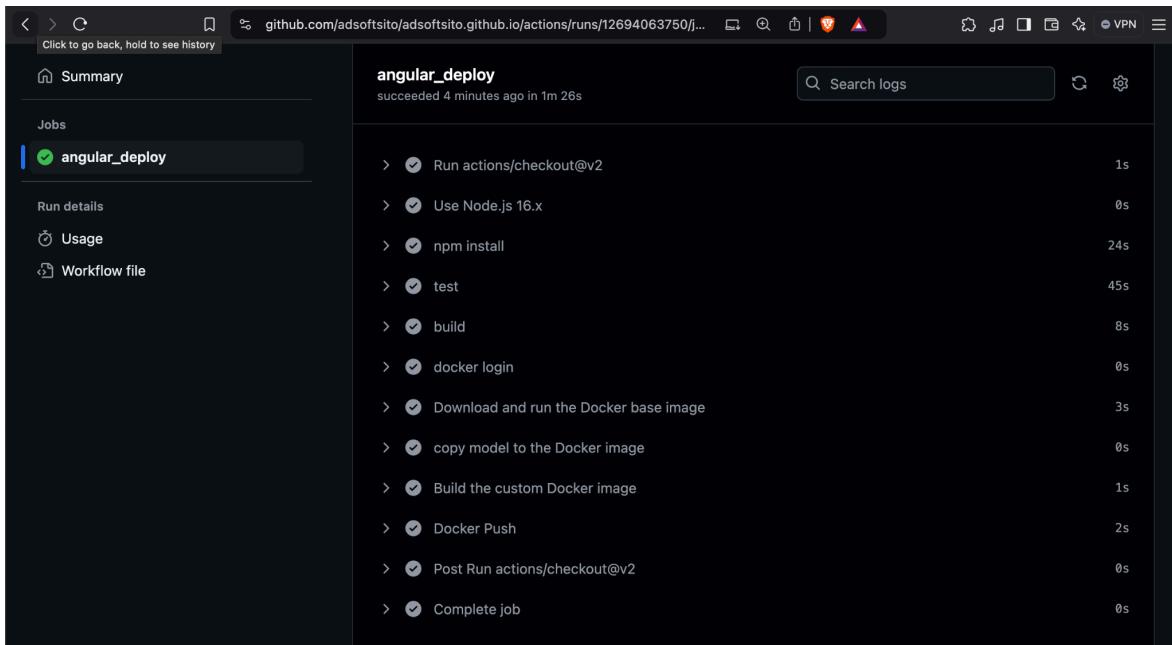
The screenshot shows the GitHub Actions interface for the repository `adsoftsito/adsoftsito.github.io/actions`. The left sidebar lists workflows: `Angular_DockerImage`, `pages-build-deployment`, `Management`, and `Caches`. The `All workflows` tab is selected. The main area displays "All workflows" with "Showing runs from all workflows". It shows 13 workflow runs. One run is highlighted: `Update main.yml` (Angular_DockerImage #4). The status is `master`, it was pushed by `adsoftsito`, and its status is `Queued`.

Revisamos los workflows, clic en el más reciente o que se encuentre en ejecución (**Update main.yml**) en nuestro caso.

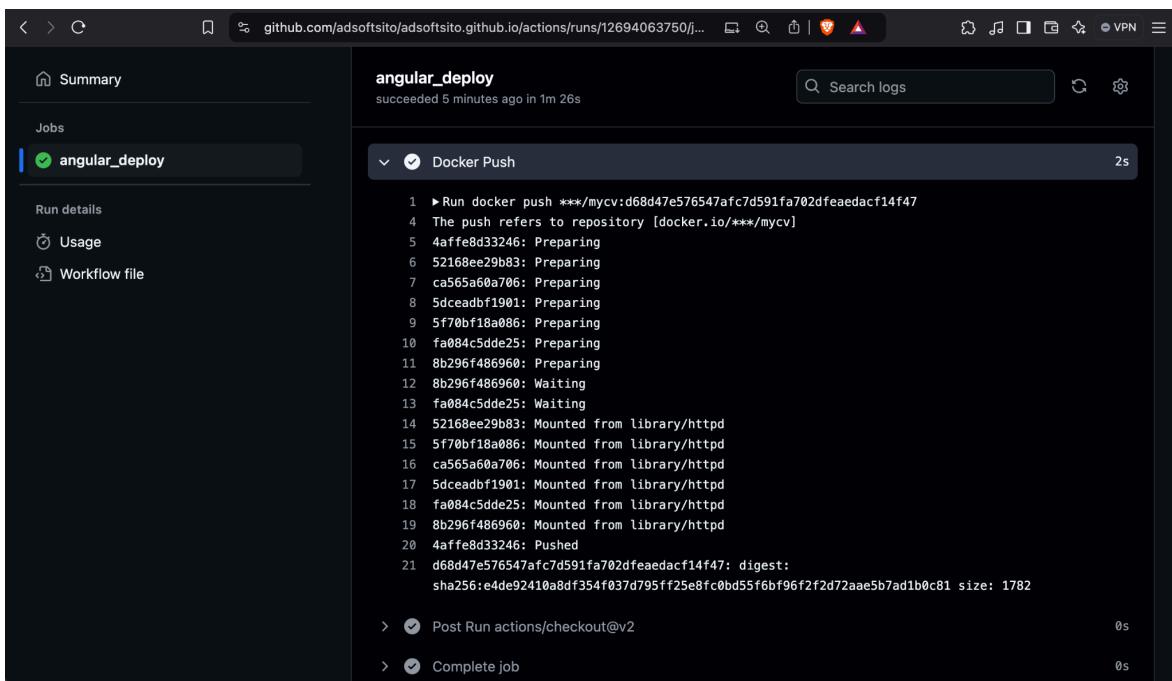
Posteriormente podemos inspeccionar los **Jobs**, click en **angular_deploy**.

The screenshot shows the details of the workflow run `Update main.yml #4` for the `Angular_DockerImage` workflow. The left sidebar shows the run summary, jobs (`angular_deploy`), run details, usage, and workflow file. The main area shows the run triggered via push 1 minute ago, pushed by `adsoftsito` to the `master` branch, and is currently `In progress`. A detailed view of the `main.yml` file shows the `angular_deploy` job, which took 1m 21s to complete.

Deberá ejecutar cada paso del Workflow sin errores. En caso de un error en cualquier paso, el proceso se cancela totalmente.



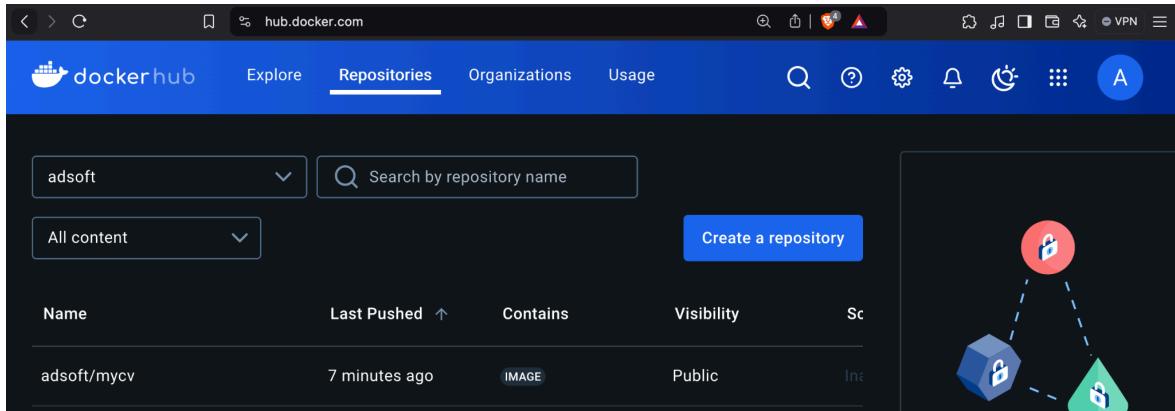
Podemos visualizar cada paso en detalle, presionando click en alguna tarea, ejemplo: **Docker push**



Nuestra imagen de docker es creada automáticamente por el Workflow, deberá aparecer en Docker Hub y podemos inspeccionar el detalle de nuestra imagen, presionando clic en el nombre de la imagen.

Cada que realicemos un cambio en nuestro código, y hagamos docker push para guardar los cambios, el Workflow se dispara automáticamente, verificará

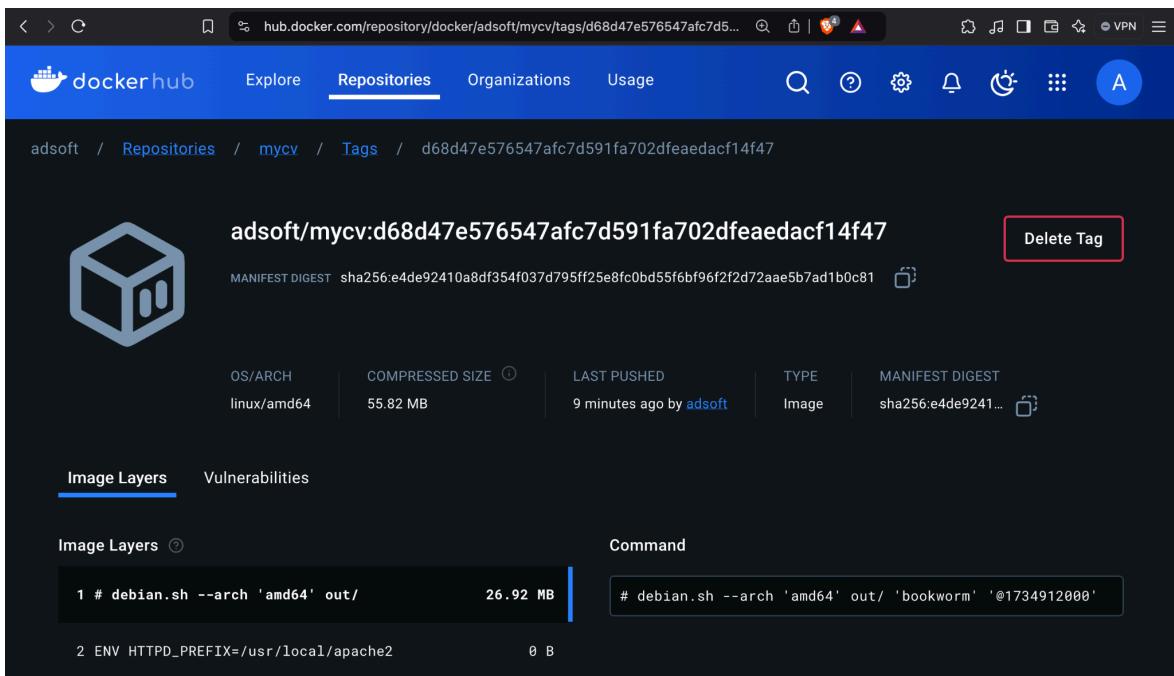
pasen las pruebas y creará una nueva versión de la imagen Docker, finalmente la subirá nuevamente a Docker Hub con otra etiqueta (tag o versión).



Para verificar el nombre de la imagen, presione click en el último Tag de la sección tags.

Tag	OS	Type	Pulled	Pushed
d68d47e576547afc7...	Image		8 minutes ago	8 minutes ago

Copiamos el nombre completo de nuestra imagen, esta información la usaremos para desplegarla en alguna nube pública.



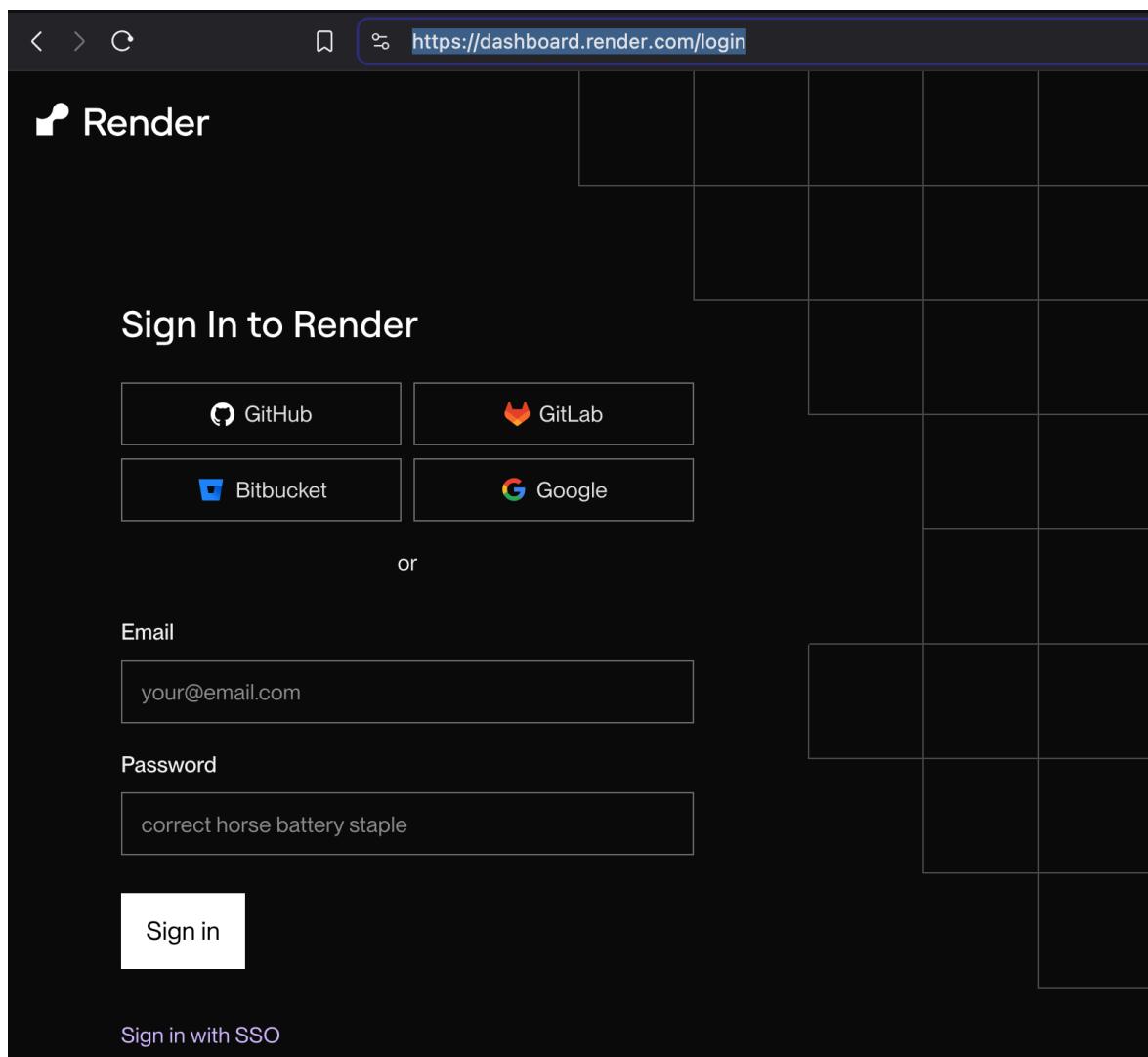
Copiamos el nombre de la imagen, un ejemplo seria:

adsoft/mycv:d68d47e576547afc7d591fa702dfeaeadcf14f47

Una vez creada nuestra imagen de docker con la última versión de nuestro código podemos publicarla en nubes como: Google Cloud Platform, Microsoft Azure, Amazon, Digital Ocean, **Render**, entre otras.

Para nuestro curso usaremos la nube gratuita Render, ingresar:

<https://dashboard.render.com/login>



En **Sign in Render**, hacemos login con nuestra cuenta de **Github**, posteriormente podemos visualizar el dashboard de nuestra cuenta.

Plantilla Nivel de Conocimiento

The screenshot shows the dashboard.render.com interface. On the left, there's a sidebar with 'adsoft's Workspace' and a 'Projects' section highlighted. The main area is titled 'Overview' and features a 'Projects' section with a callout box: 'Get organized with Projects' which says 'An easier way to organize your resources and collaborate with team members.' It includes a 'Create your first project' button and a 'Learn more' link. There are also 'Invite your team' and '+ Add new' buttons.

Creamos un nuevo servicio, click en **+ New**, seleccionamos **Web Service**.

The screenshot shows the 'New Project' dialog from dashboard.render.com. The 'Project' dropdown menu is open, and 'Web Service' is selected. Other options visible in the menu include 'Static Site', 'Private Service', 'Background Worker', 'Cron Job', 'PostgreSQL', 'Redis', and 'Blueprint'. The main area of the dialog shows a 'Get organized with Projects' callout and a 'Create your first project' button.

Seleccionamos **Existing image**, y copiamos el nombre completo de nuestro imagen de docker hub, click en **Connect**.

You are deploying a Web Service

Source Code

Git Provider Public Git Repository Existing Image

Image URL Deploy an image from a Docker registry

adsoft/mycv:d68d47e576547afc7d591fa702dfaeadacf14f47

Credential (Optional)

No credential

Connect →

En **Name**, Personalizamos el nombre de nuestro servicio.

You are deploying a Web Service

Source Code

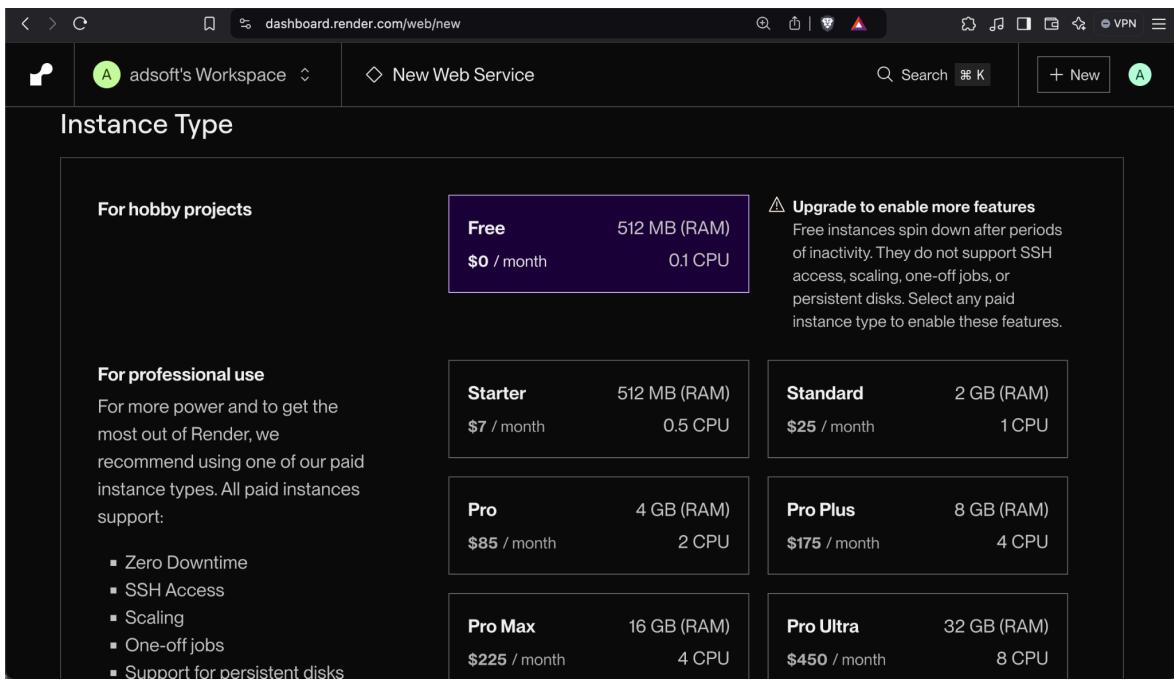
docker.io/adsoft/mycv:d68d47e576547afc7d591fa702dfaeadacf14f47 Edit

Name

A unique name for your web service.

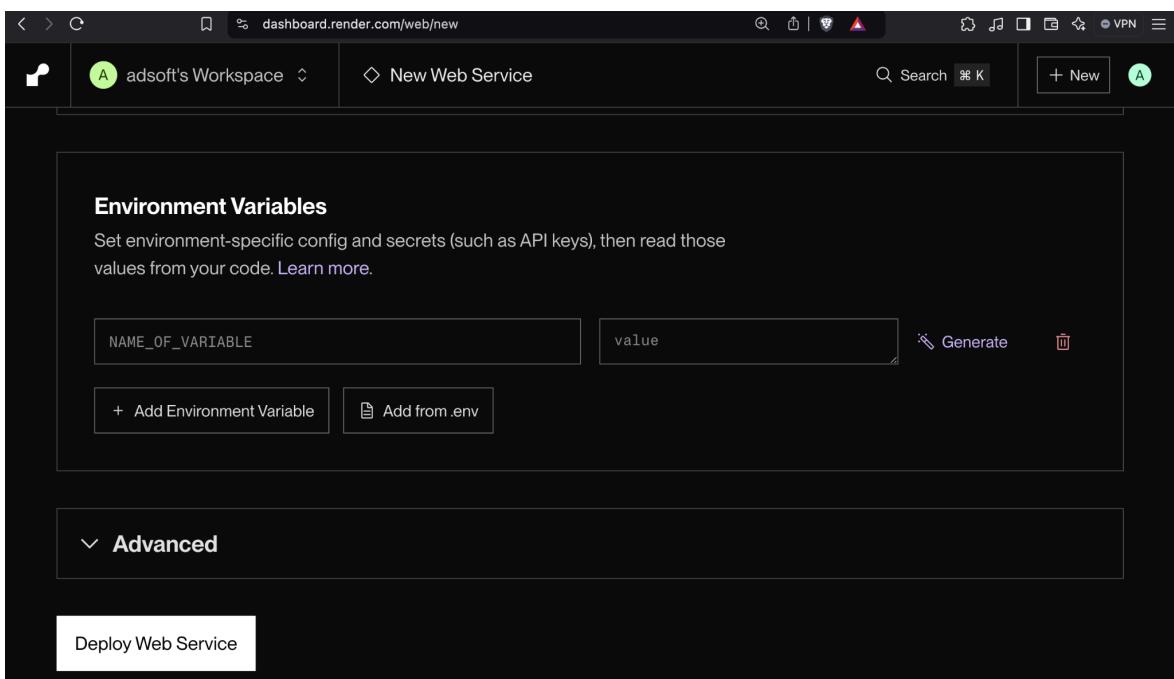
mycv_adsoft

Seleccionamos el tipo de instancia: **free**



Instance Type	RAM	CPU	Pricing
Free	512 MB (RAM)	0.1 CPU	\$0 / month
Starter	512 MB (RAM)	0.5 CPU	\$7 / month
Standard	2 GB (RAM)	1 CPU	\$25 / month
Pro	4 GB (RAM)	2 CPU	\$85 / month
Pro Plus	8 GB (RAM)	4 CPU	\$175 / month
Pro Max	16 GB (RAM)	4 CPU	\$225 / month
Pro Ultra	32 GB (RAM)	8 CPU	\$450 / month

Finalmente clic en **Deploy Web Service**



Environment Variables
 Set environment-specific config and secrets (such as API keys), then read those values from your code. Learn more.

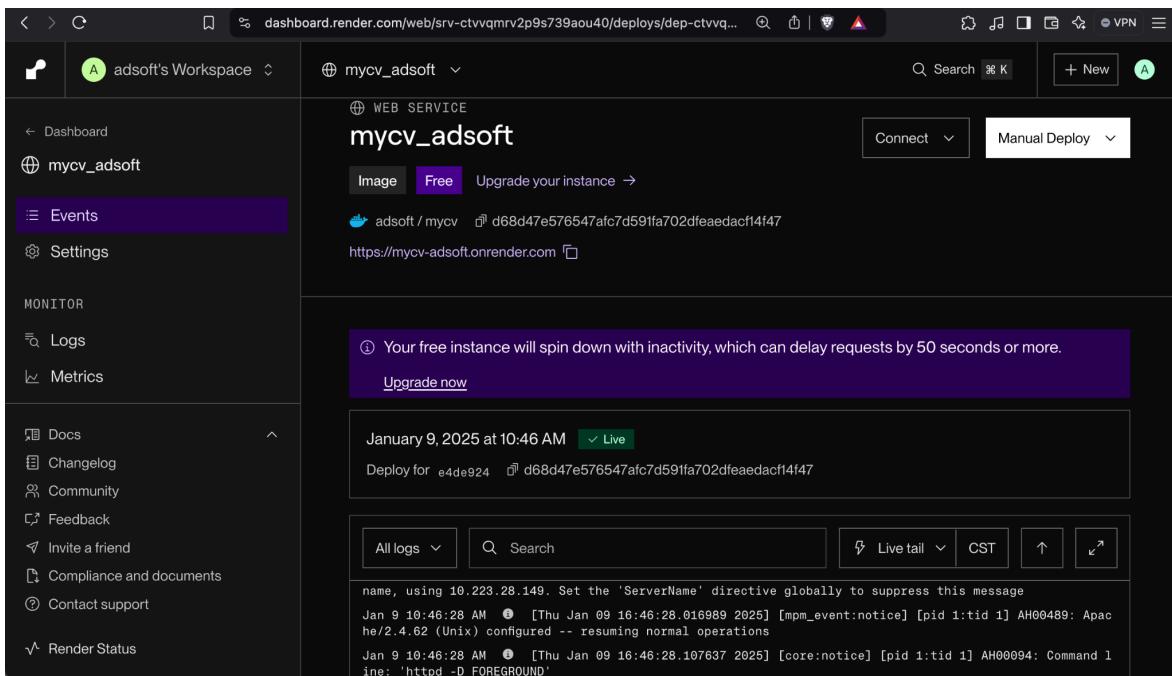
NAME_OF_VARIABLE	value	Generate	☰
+ Add Environment Variable	Add from .env		

Advanced

Deploy Web Service

Esperamos que se despliegue el proyecto y alcance el estatus **live**.

Verificamos el URL de nuestro servicio: <https://mycv-adsoft.onrender.com>



Probamos nuestro proyecto en producción en Chrome, con nuestra imagen tomada de Docker Hub.

Adolfo Centeno Ser ingeniero de software		adsoft@live.com.mx 2721908413 Orizaba Ver. Mexico @adsoft.sito
<ul style="list-style-type: none"> • nov-2023 - nov-2024 <i>Puebla, Mexico</i> Backend Developer kubeet SA de CV Logros: graphql api, REST api • ene-2020 - oct-2023 <i>Orizaba, Mexico</i> Frontend Developer Waves Lab Logros: web platform, android app, ios app 		skills works!
education works!		certificates works!
languages works!		interests works!

Cada que se genere una nueva imagen en Docker Hub, debemos actualizar la versión o tag en la nube de **render.com** para que se actualicen los cambios en producción.

Sala de inspiración

Todas los proyectos de software deben ser puestos en producción, ya sean de frontend, backend o aplicaciones móviles, el uso Docker y Workflows son

tecnologías ampliamente usadas en la industria para realizar despliegues continuos de proyectos de software.

Sala de pruebas

Lee cada una de las preguntas y selecciona la respuesta que consideres correcta

1.- *comando docker para descarga una imagen llamada ubuntu*

Opción a: <i>docker run ubuntu</i>	Incorrecta: <i>El comando correcto de docker para descargar una imagen de Docker Hub es:</i> <i>docker pull ubuntu</i>
Opción b: <i>docker push ubuntu</i>	Incorrecta: <i>El comando correcto de docker para descargar una imagen de Docker Hub es:</i> <i>docker pull ubuntu</i>
Opción c: <i>docker pull ubuntu</i>	CORRECTA: <i>El comando correcto de docker para descargar una imagen de Docker Hub es:</i> <i>docker pull ubuntu</i>
Opción d: <i>docker start ubuntu</i>	Incorrecta: <i>El comando correcto de docker para descargar una imagen de Docker Hub es:</i> <i>docker pull ubuntu</i>

2.- Comando para crear un nuevo contenedor llamada **test**, a partir de una imagen llamada **debian**

Opción a: <i>docker start test -d debian</i>	Incorrecta: El comando para crear un nuevo contenedor es: <i>docker run test -d debian</i>
Opción b: <i>docker test -d debian</i>	Incorrecta: El comando para crear un nuevo contenedor es: <i>docker run test -d debian</i>
Opción c: <i>docker push test -d debian</i>	Incorrecta: El comando para crear un nuevo contenedor es: <i>docker run test -d debian</i>
Opción d: <i>docker run test -d debian</i>	CORRECTA: El comando para crear un nuevo contenedor es: <i>docker run test -d debian</i>

3.- Comando en docker para subir una imagen llamada **octocat/mycv:latest** a Docker Hub.

Opción a: <i>docker push octocat/mycv:latest</i>	CORRECTA El comando correcto para subir una imagen a Docker Hub es:
--	---

	<code>docker push octocat/mycv:latest</code>
Opción b: <code>docker run octocat/mycv:latest</code>	Incorrecta: El comando correcto para subir una imagen a Docker Hub es: <code>docker push octocat/mycv:latest</code>
Opción c: <code>docker commit octocat/mycv:latest</code>	Incorrecta: El comando correcto para subir una imagen a Docker Hub es: <code>docker push octocat/mycv:latest</code>
Opción d: <code>docker pull octocat/mycv:latest</code>	Incorrecta: El comando correcto para subir una imagen a Docker Hub es: <code>docker push octocat/mycv:latest</code>

Pruébate (de los 3 temas)

Ahora vamos a realizar algunos cambios en tu código en Angular y lo actualizaremos en producción.

1. Actualiza el componente Header, actualiza datos de tu misión en la vida, redes sociales o número de celular.
2. Prueba con `ng test` que no existan pruebas rotas
3. Ejecuta `ng serve` para validar los cambios en frontend
4. Sube tus cambios a github
5. Verifica que el WorkFlow se ejecute correctamente, no mande error de pruebas rotas

6. Comprueba que el Workflow genere una nueva versión de la imagen en Docker Hub
7. Copia el id de la nueva versión o tag generada.
8. Actualiza en render.com nuestro service con el nuevo tag de la imagen Docker
9. Verifica se actualice el servicio y se reflejen los cambios.

Ideas para llevar

Actualmente los procesos de integración continua y despliegue continuo (CI/CD) es una práctica ampliamente usada en la industria de software. Involucra prácticas de la ingeniería de software para automatizar procesos usando tecnologías entre ellas el uso de repositorios, scripts automatizados, docker, nubes de cómputo, entre otros.

Todo esto forma el concepto **devops** que ha permitido mejorar las prácticas de producción de software de calidad en los tiempos y costos presupuestados.

Referencias

<https://arxiv.org/pdf/2308.16774v3>

<https://medium.com/@nagarjunmallesh/introduction-to-github-workflows-a-comprehensive-overview-2f5b89522578>

<https://www.geeksforgeeks.org/github-workflows/>

Material de consulta

<https://pages.github.com/>

<https://docs.github.com/en/actions/writing-workflows>

<https://docs.docker.com/get-started/>

<https://www.geeksforgeeks.org/docker-instruction-commands/>