

Mi Primera Prueba Real de CI/CD con Docker

¿En qué consiste este ejercicio?

El objetivo es crear un sistema automatizado donde tú solo te encargues de programar. Al subir tu código a **GitHub**, un "robot" (GitHub Actions) se activará automáticamente, empaquetará tu aplicación en un contenedor de **Docker** y lo subirá a la nube (**Docker Hub**) y tu ordenador convertido en un servidor, realizará el despliegue automático. Esto es exactamente lo que ocurre en las empresas: el código viaja de GitHub a servidores en la nube de forma totalmente transparente.

Conceptos Clave

- **CI (Integración Continua):** Es la automatización de la construcción y prueba de tu código cada vez que haces un cambio. Asegura que el nuevo código "encaje" bien.
- **CD (Despliegue Continuo):** Es llevar ese código automáticamente a un entorno de producción (un servidor) para que los usuarios puedan usarlo.

Checklist de Requisitos

Antes de tocar el código, asegúrate de tener:

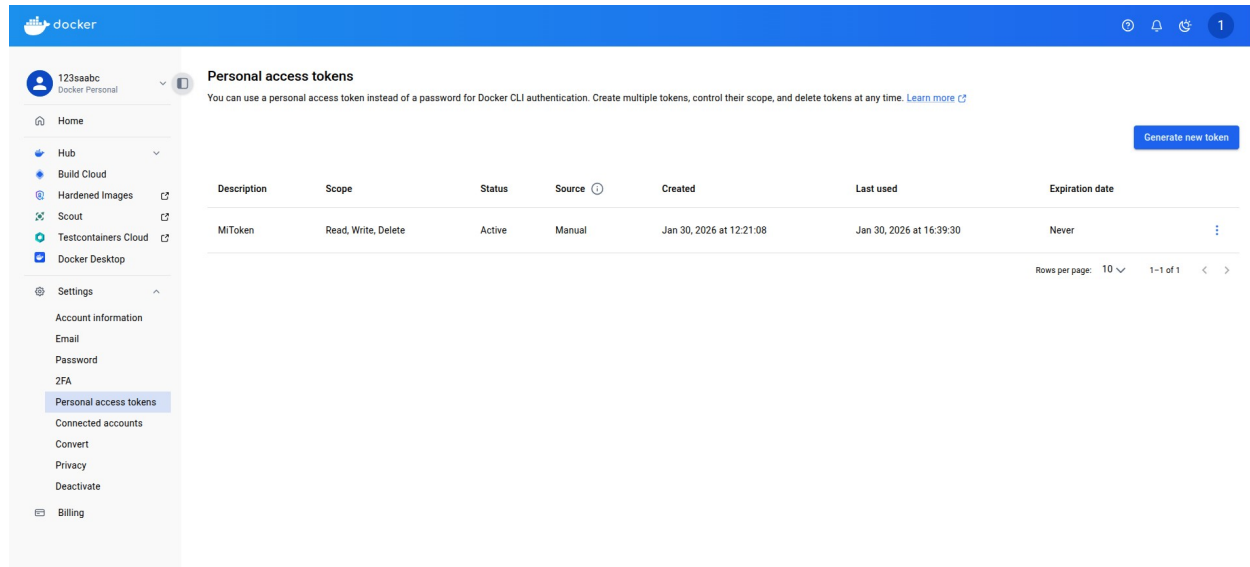
- ☐ Una cuenta en [GitHub](#).
- ☐ Una cuenta en [Docker Hub](#).
- ☐ Configurar el ordenador local en runner.
- ☐ Git instalado en tu ordenador.
- ☐ Un editor de código (como VS Code).

Paso 1: Preparación en Docker Hub (Las llaves)

Para que GitHub pueda subir cosas a tu cuenta de Docker, necesitamos una "llave especial" (Token).

1. **Entra en Docker Hub** y haz clic en tu foto (arriba a la derecha) -> **Account Settings**.
2. Ve a **Security** -> **Personal Access Tokens** -> **New Access Token**.
3. **Configuración:**
 - Nombre: GitHub-Actions-CI
 - Permisos: **Read, Write, Delete**.
4. **¡Copia el Token!** Dale al botón de copiar y guárdalo en un bloc de notas temporalmente. No podrás volver a verlo.
5. **Anota tu Usuario:** Tu usuario de Docker Hub no es tu email, es el ID que aparece bajo tu

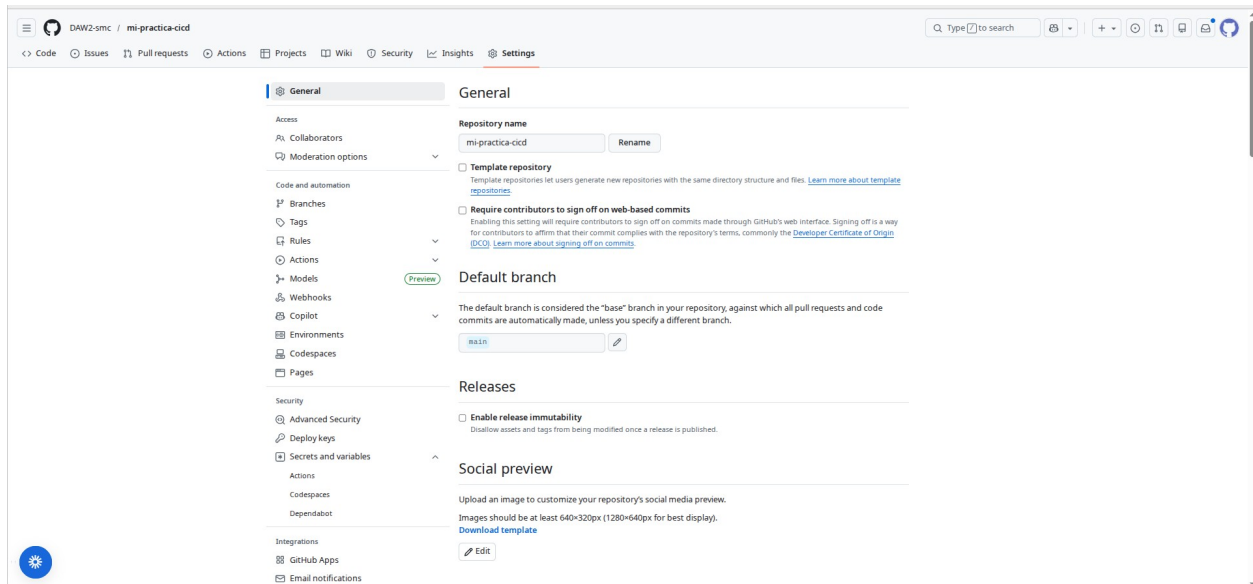
nombre en el perfil.



Paso 2: Configurar los Secretos en GitHub (Donde guardar las llaves)

IMPORTANTE: No confundas el Settings de tu perfil con el del proyecto.

1. Crea un repositorio nuevo y público que se llame mi-practica-cicd
2. Entra en la página de tu repositorio en GitHub (ej: mi-practica-cicd).
3. En la barra superior de pestañas del repositorio, haz clic en **Settings** ⚙️ (es la última a la derecha, junto a *Insights*).
4. En el menú lateral izquierdo, busca la sección **Security** -> **Secrets and variables** -> **Actions**.



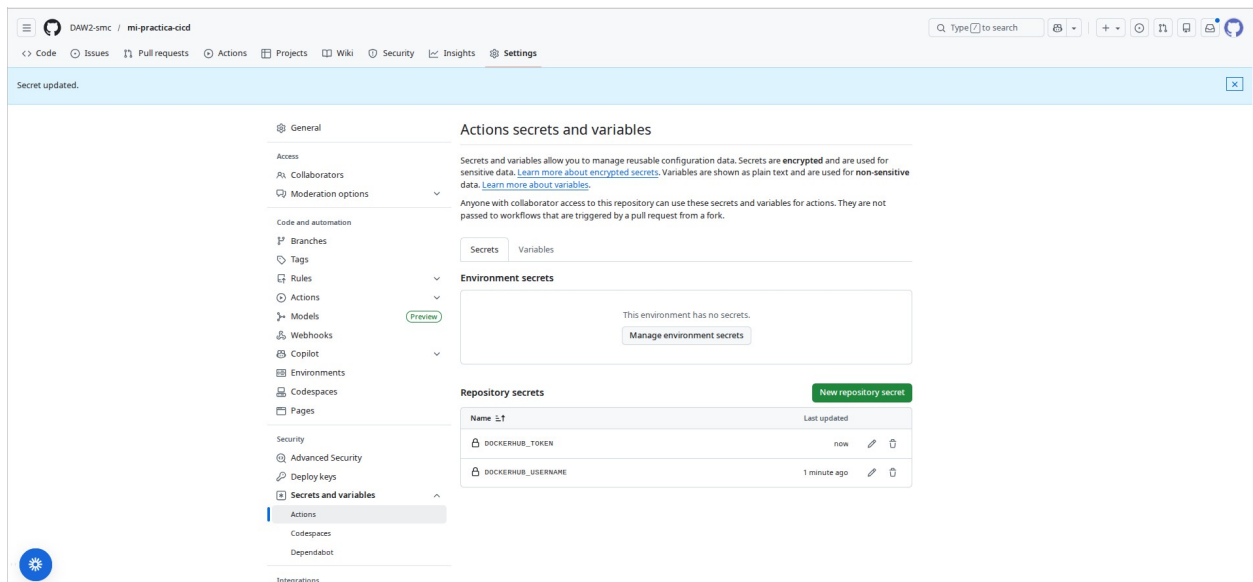
5. Pulsa el botón verde **New repository secret** para crear dos secretos:

- **Primer Secreto:**

- Name: DOCKERHUB_USERNAME
- Secret: (Tu usuario de Docker Hub, el ID corto).

- **Segundo Secreto:**

- Name: DOCKERHUB_TOKEN
- Secret: (El token que copiaste en el Paso 1 que empieza por dckr_pat_...).



**** Estamos metiendo en GitHub las claves (usuario y token) de DockerHub, porque el robot lo que va a hacer es que cuando haya cambios en el código en el repositorio común de desarrolladores GITHUB, va a crear una imagen y subirlo al HUB de DOCKER de forma automática.**

Paso 3: Configurar el Runner en ordenador Local

⚠ **IMPORTANTE: Estructura de Carpetas**

NUNCA instales el Runner dentro de la carpeta de tu repositorio de Git. Esto causará errores de subida por el peso de los archivos.

- **Carpeta de Código (Git):** .../mi-proyecto/ (Solo archivos de texto: html, yaml).
- **Carpeta del Runner:** ~/actions-runner/ (Fuera de la carpeta de Git).

Aún incluyendo el fichero de .ignore, la carpeta de runner debe estar fuera de la carpeta que se sincroniza con gitHub.

3.1.Registrar tu ordenador como Runner en GitHub

- En tu repo de GitHub, ve a **Settings** -> **Actions** -> **Runners**.
- Pulsa **New self-hosted runner**.
- Selecciona tu sistema operativo (**Windows**, **macOS** o **Linux**).
- GitHub te dará una lista de comandos. **¡No cierres esa pestaña!**

3.2.Configurar el Runner en tu PC

- Abre una terminal en tu ordenador.
- Crea una carpeta llamada runner-github (ej: mkdir runner-github && cd runner-github).
- Copia y pega los comandos de **Download** que te dio GitHub.
- Copia y pega el comando de **Configure** (el que empieza por ./config.sh o config.cmd).
 - **Importante:** Cuando te pregunte por **Labels**, escribe: mi-pc-local.
 - A lo demás, puedes darle a **Enter** para aceptar los valores por defecto.
- Finalmente, arranca el runner con: ./run.sh (o run.cmd en Windows).
 - Deberías ver: ✓ Connected to GitHub. Listening for Jobs.

```
Archivo Editar Ver Buscar Terminal Ayuda
2/runner-github$
sara@max11:~/Escritorio/UT06.Documentos y Sistemas de Control de Versiones/02.EJ2/runner-github$
2/runner-github$ curl -O actions-runner-linux-x64-2.331.0.tar.gz -L https://github.com/actions/runner/releases/download/v2.331.0/actions-runner-linux-x64-2.331.0.tar.gz
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left  Speed
100 212M  100 212M    0     0  41.5M      0  0:00:05  0:00:05 --:--:-- 43.5M
sara@max11:~/Escritorio/UT06.Documentos y Sistemas de Control de Versiones/02.EJ2/runner-github$ echo "5fcc01bd540ba5c3f1291c2803658ebd3cedb3836489eda3be357d41bfcf28a7 actions-runner-linux-x64-2.331.0.tar.gz" |
shasum -a 256 -c
actions-runner-linux-x64-2.331.0.tar.gz: OK
sara@max11:~/Escritorio/UT06.Documentos y Sistemas de Control de Versiones/02.EJ2/runner-github$ tar xzf ./actions-runner-linux-x64-2.331.0.tar.gz
sara@max11:~/Escritorio/UT06.Documentos y Sistemas de Control de Versiones/02.EJ2/runner-github$ ./config.sh --url https://github.com/DAM2-snc/mi-practica-cicd --token BXXK2FTC2WNJLGSZB03NER33P6JYS

-----
GitHub Actions
Self-hosted runner registration
-----

# Authentication

✓ Connected to GitHub

# Runner Registration

Enter the name of the runner group to add this runner to: [press Enter for Default]

Enter the name of runner: [press Enter for max11]

This runner will have the following labels: 'self-hosted', 'Linux', 'X64'
Enter any additional labels (ex. label-1,label-2): [press Enter to skip] mi-pc-local

✓ Runner successfully added

# Runner settings

Enter name of work folder: [press Enter for _work]

✓ Settings Saved.
sara@max11:~/Escritorio/UT06.Documentos y Sistemas de Control de Versiones/02.EJ2/runnsasara@max11:~/Escritorio/UT06.Documentos y Sistemas de Control de Versiones/02.EJ2/runner-github$ ./run.sh
sara@max11:~/Escritorio/UT06.Documentos y Sistemas de Control de Versiones/02.EJ2/runner-github$ ./run.sh

✓ Connected to GitHub

Current runner version: '2.331.0'
2026-02-01 17:07:35Z: Listening for Jobs
```

Paso 4: El Proyecto en Local (Tu PC)

4.1. Crear carpeta en local

Bash

```
mkdir mi-proyecto-cicd-ej2 && cd mi-proyecto-cicd-ej2
```

4.2 Crear archivo index.html

- index.html: Tu página web sencilla.

```
<> index.html > ...
1  |<!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>Mi Primera Web con CI/CD</title>
5  </head>
6  <body>
7  |   <h1>¡Hola! Este sitio se actualiza solo.</h1>
8  |   <p>Aprendiendo Docker y GitHub Actions.</p>
9  </body>
10 </html>
```

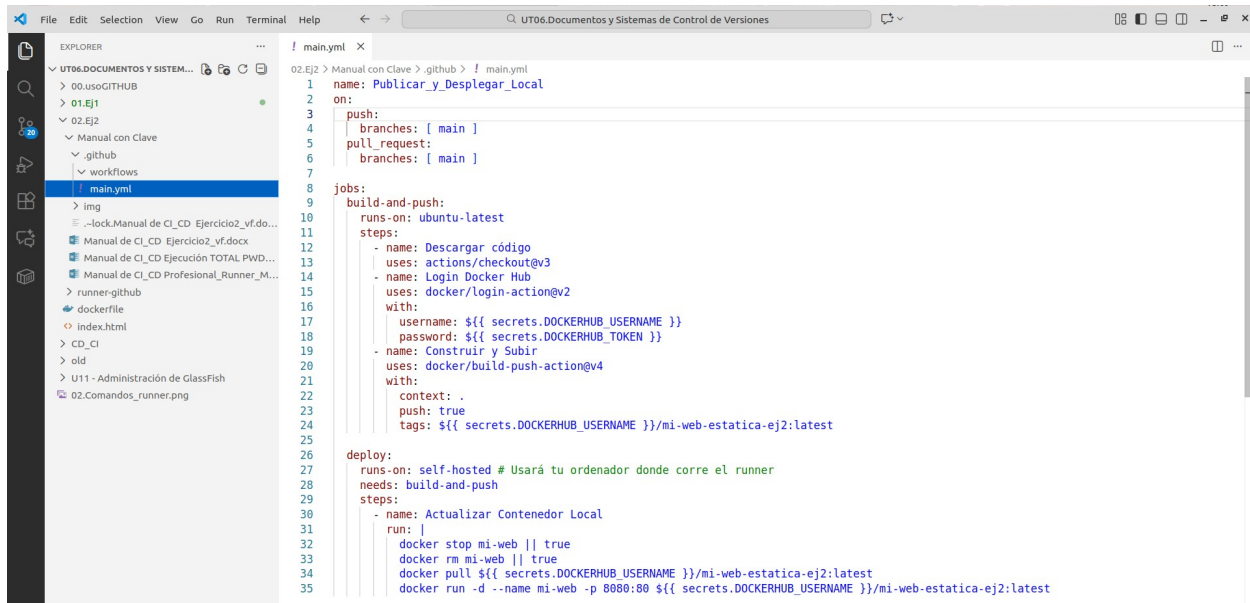
4.3 Crear archivo dockerfile

- Dockerfile: Las instrucciones para que Docker sepa cómo montar tu web.

```
dockerfile > ...
1 # Usamos un servidor web muy ligero llamado Nginx
2 FROM nginx:alpine
3
4 # Copiamos nuestro archivo html dentro de la carpeta del servidor
5 COPY index.html /usr/share/nginx/html/index.html
```

4.4 Crear el workflow en un archivo main.yml

- main.yml: Este archivo es el "cerebro" que le dice a GitHub: "Usa mis secretos de Docker Hub para hacer login y subir mi imagen".
- Tengo que crear una carpeta oculta .github y dentro una carpeta workflow donde creo el fichero.



```
1 name: Publicar_y_Desplegar_Local
2 on:
3   push:
4     branches: [ main ]
5   pull_request:
6     branches: [ main ]
7
8 jobs:
9   build-and-push:
10    runs-on: ubuntu-latest
11    steps:
12      - name: Descargar código
13        uses: actions/checkout@v3
14      - name: Login Docker Hub
15        uses: docker/login-action@v2
16        with:
17          username: ${ secrets.DOCKERHUB_USERNAME }
18          password: ${ secrets.DOCKERHUB_TOKEN }
19      - name: Construir y Subir
20        uses: docker/build-push-action@v4
21        with:
22          context: .
23          push: true
24          tags: ${ secrets.DOCKERHUB_USERNAME }}/mi-web-estatica-ej2:latest
25
26   deploy:
27     runs-on: self-hosted # Usará tu ordenador donde corre el runner
28     needs: build-and-push
29     steps:
30      - name: Actualizar Contenedor Local
31        run: |
32          docker stop mi-web || true
33          docker rm mi-web || true
34          docker pull ${ secrets.DOCKERHUB_USERNAME }}/mi-web-estatica-ej2:latest
35          docker run -d --name mi-web -p 8080:80 ${ secrets.DOCKERHUB_USERNAME }}/mi-web-estatica-ej2:latest
```

name: Publicar_y_Desplegar_Local

on:

push:

branches: [main]

pull_request:

branches: [main]

```

jobs:
  build-and-push:
    runs-on: ubuntu-latest
    steps:
      - name: Descargar código
        uses: actions/checkout@v3
      - name: Login Docker Hub
        uses: docker/login-action@v2
        with:
          username: ${ secrets.DOCKERHUB_USERNAME }
          password: ${ secrets.DOCKERHUB_TOKEN }
      - name: Construir y Subir
        uses: docker/build-push-action@v4
        with:
          context: .
          push: true
          tags: ${ secrets.DOCKERHUB_USERNAME }/mi-web-estatica-ej2:latest

  deploy:
    runs-on: self-hosted # Usará tu ordenador donde corre el runner
    needs: build-and-push
    steps:
      - name: Actualizar Contenedor Local
        run: |
          docker stop mi-web || true
          docker rm mi-web || true
          docker pull ${ secrets.DOCKERHUB_USERNAME }/mi-web-estatica-ej2:latest
          docker run -d --name mi-web -p 8080:80 ${ secrets.DOCKERHUB_USERNAME }/mi-
          web-estatica-ej2:latest

```

Paso 5: Sincronizar GitLocal y GitHub

Los ficheros que hemos creado en local, ahora queremos compartirlos con nuestro equipo en GitHub, que es donde colaboramos juntos. Para eso, donde tengo los fichero de index.html [...] abro un terminal y realizo los pasos para enviar los datos al GitHub.

```

Bash
git init
git add .
git commit -m "Estructura inicial"
git branch -M main
git remote add origin URLRepositorio

```

git push -u origin main

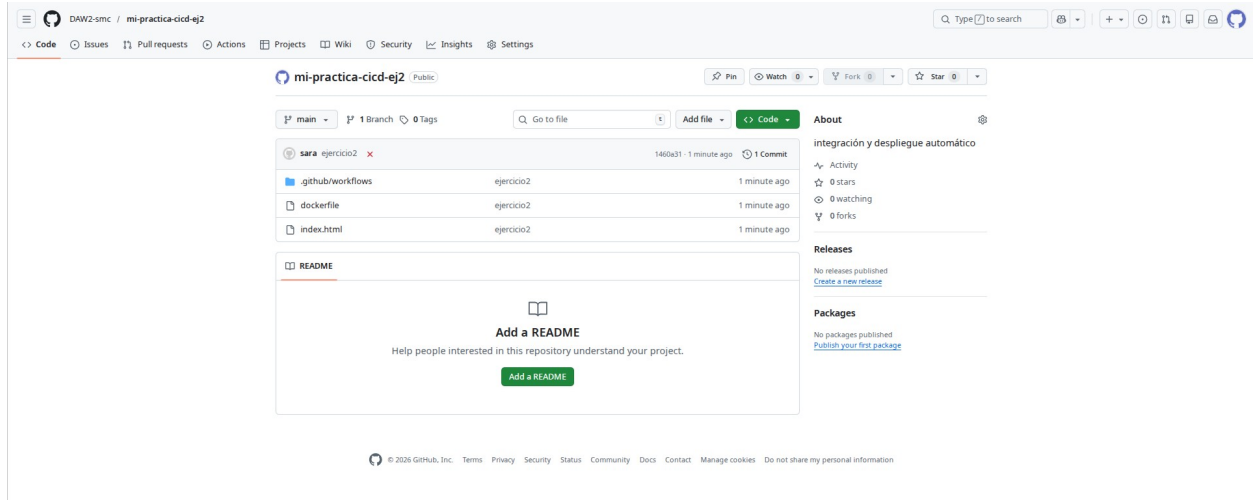
Cuando lo hagamos vemos en nuestro GITHUB la información actualizada:

The screenshot shows the GitHub repository page for 'mi-practica-cicd'. The repository is public and has 1 branch (main) and 0 tags. The commit history shows a single commit by 'sara' titled 'Mi primer commit' made 1 hour ago. The repository contains files: '.github/workflows', 'dockerfile', and 'index.html'. The README section is empty, with a prompt to 'Add a README'. The right sidebar shows 'About' (no description), 'Releases' (no releases published), 'Packages' (no packages published), and 'Languages' (HTML 100.0%).

The screenshot shows the GitHub repository page for 'mi-practica-cicd-ej2'. The repository is public and has 1 branch (main) and 0 tags. The commit history shows a single commit by 'sara' titled 'Mi primer commit' made 1 hour ago. The repository contains files: '.github/workflows', 'dockerfile', and 'index.html'. The README section is empty, with a prompt to 'Add a README'. The right sidebar shows 'About' (no description), 'Releases' (no releases published), 'Packages' (no packages published), and 'Languages' (HTML 100.0%).

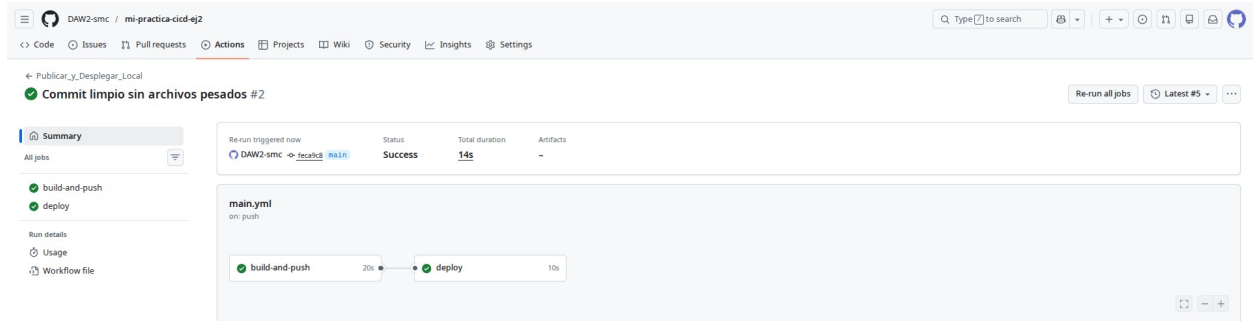
Overlaid on the screenshot is a terminal window showing the following commands and output:

```
sara@max11:~/home/sara/Escritorio/UT06.Documentos y Sistemas de Control de Versiones$
create node 100644 runner-github/run-helper.cnd.template
create node 100755 runner-github/run-helper.sh
create node 100755 runner-github/run-helper.sh.template
create node 100755 runner-github/run.sh
create node 100755 runner-github/safe_sleep.sh
create node 100755 runner-github/svc.sh
sara@max11:~/Escritorio/UT06.Documentos y Sistemas de Control de Versiones/02.Ej2$
25 git branch -M main
sara@max11:~/Escritorio/UT06.Documentos y Sistemas de Control de Versiones/02.Ej2$
25 git remote add origin https://github.com/DAW2-smc/ml-practica-cicd-ej2
fatal: remoto origin ya existe.
sara@max11:~/Escritorio/UT06.Documentos y Sistemas de Control de Versiones/02.Ej2$
25 git remote add origin2 https://github.com/DAW2-smc/ml-practica-cicd-ej2
sara@max11:~/Escritorio/UT06.Documentos y Sistemas de Control de Versiones/02.Ej2$
25 git push -u origin2 main
Username for 'https://github.com': DAW2-smc
Password for 'https://DAW2-smc@github.com':
Enumerando objetos: 4986, listo.
Contando objetos: 100% (4986/4986), listo.
Compresión de la usando hasta 8 hilos
Comprimiendo objetos: 100% (4717/4717), listo.
error: RPC fallo; HTTP 408 curl 22 The requested URL returned error: 408
fatal: el remoto se colgó de manera inesperada
Escribiendo objetos: 5K (250/4986), 250.70 MiB | 14.84 MiB/s
```

Paso 6: Menú ACTIONS en GitHub

Cuando hemos hecho el push, además de subirse los ficheros, se ha ejecutado el código del main.yml creando un job, que ha podido terminar ok y se muestra en verde o terminar con errores y nos dice qué ha fallado.



Cuando se han ejecutado los dos pasos, mágicamente se publica nuestra web:
<http://localhost:8080/>

¡Hola! Este sitio es el ejercicio 2.

Aprendiendo Docker y GitHub Actions.

```
sara@max11:~$ docker ps
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	POR
TS						
8d7611914e17	123saabc/mi-web-estatica-ej2:latest	mi-web-ej2	"/docker-entrypoint...."	11 minutes ago	Up 11 minutes	0.0

```
.0.0:8080->80/tcp, [::]:8080->80/tcp
```

```
sara@max11: /home/sara/Escritorio/UT06.CDCI/02.Ej2/runner-github
```

Archivo Editar Ver Buscar Terminal Ayuda

```
sara@max11:~/Escritorio/UT06.CDCI/02.Ej2/runner-github$ ./run.sh
```

```

✓ Connected to GitHub

Current runner version: '2.331.0'
2026-02-01 18:35:59Z: Listening for Jobs
2026-02-01 18:36:15Z: Running job: deploy
2026-02-01 18:36:25Z: Job deploy completed with result: Succeeded

```

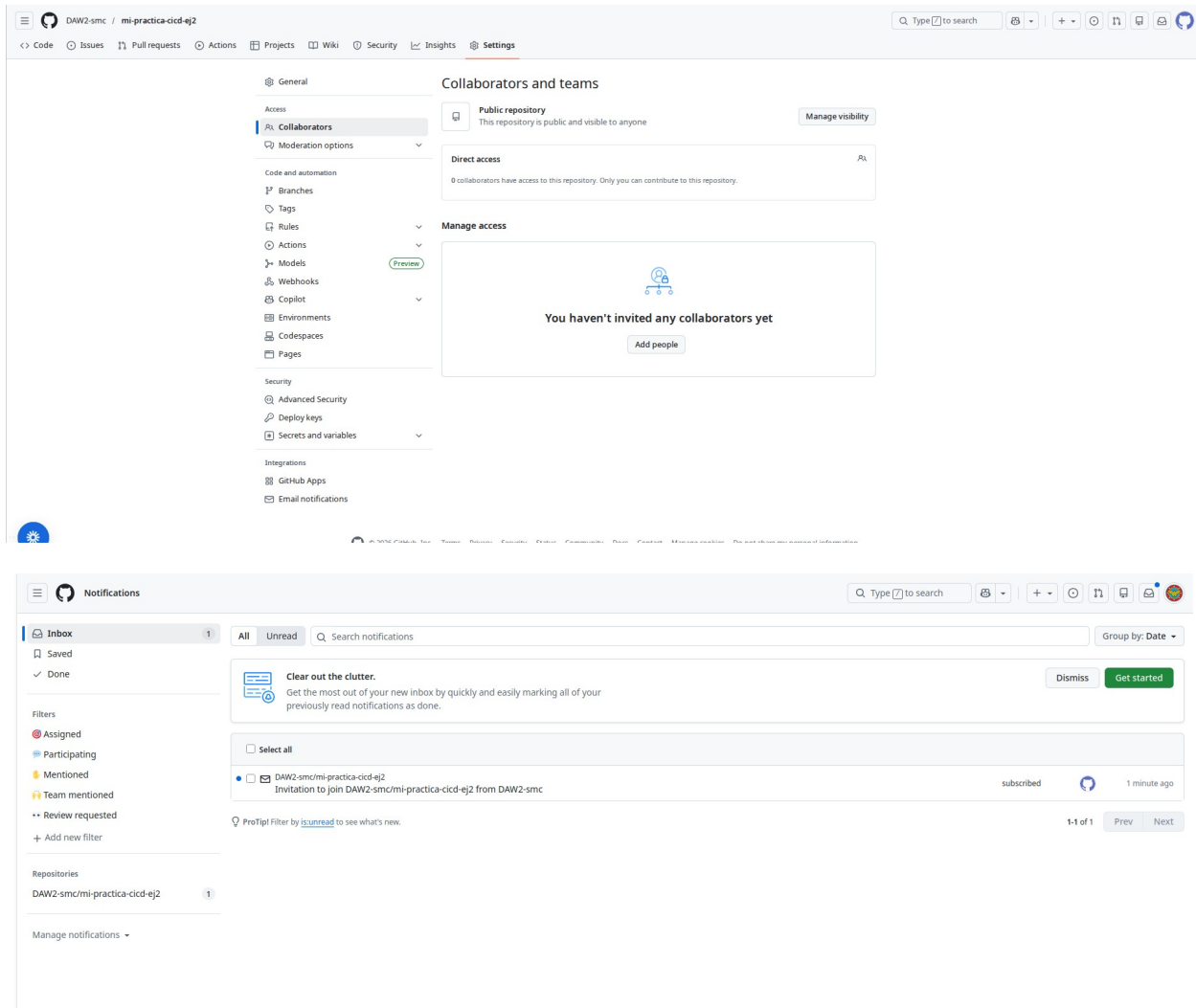
Paso 7: Pull request de un colaborador.

Para ver el CD en acción:

2. Asegúrate de que el Runner está en estado Listening en tu terminal.
3. Simula un cambio (o pídele a un colaborador que lo haga):
 - Cambia el fondo de la web en index.html.
 - Haz git push a la rama main.
4. Mira tu terminal: Verás que el Runner recibe un "Job", ejecuta los comandos de Docker, descarga la imagen y la arranca.
5. Abre en tu navegador localhost:8080. ¡Tu web se ha actualizado sola!

7.1.Damos acceso a un colaborador

Entramos en Settings del repositorio y en Colaboradores añadimos a un colaborador. Este, entrará en su cuenta de GitHub y aceptará la invitación.

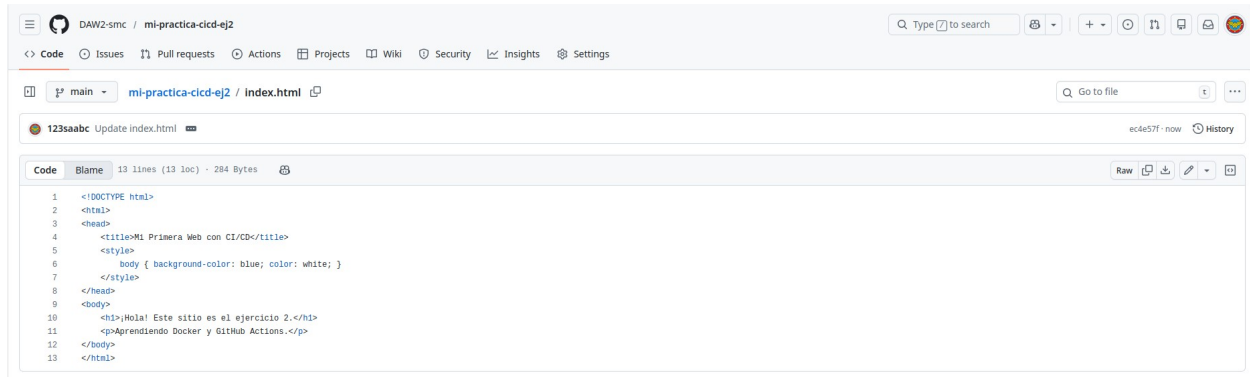


7.2.Colaborador hace cambios y actualiza GH.

Opción A: Directamente desde la web de GitHub (La más rápida)

1. **Entra con la cuenta del colaborador** en el repositorio.
2. Busca el archivo `index.html` y haz clic en el icono del **lápiz (Edit)**.
3. Modifica algo visible (por ejemplo, cambia el título `<h1>` o el color de fondo en el CSS).
4. Haz clic en el botón verde **Commit changes....**
5. Aquí tienes dos opciones:
 - **Commit directly to the main branch:** Esto disparará el CI/CD inmediatamente.
 - **Create a new branch and start a pull request:** Esto es lo más profesional. Tú (como dueño) tendrás que ir a la pestaña **Pull Requests**, revisar el cambio y

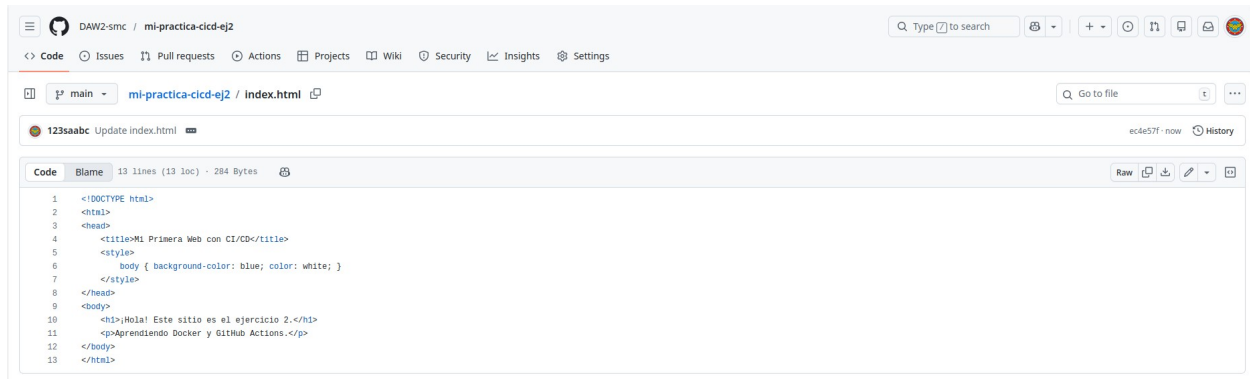
darle a **Merge**. En cuanto le des a "Merge", se disparará el proceso hacia tu ordenador.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Mi Primera Web con CI/CD</title>
5   <style>
6     body { background-color: blue; color: white; }
7   </style>
8 </head>
9 <body>
10  <h1>¡Hola! Este sitio es el ejercicio 2.</h1>
11  <p>Aprendiendo Docker y GitHub Actions.</p>
12 </body>
13 </html>
```

Hemos hecho un commit directamente desde el colaborador.

y desde el repositorio y el usuario maestro vemos ese ok al cambio del colaborador.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Mi Primera Web con CI/CD</title>
5   <style>
6     body { background-color: blue; color: white; }
7   </style>
8 </head>
9 <body>
10  <h1>¡Hola! Este sitio es el ejercicio 2.</h1>
11  <p>Aprendiendo Docker y GitHub Actions.</p>
12 </body>
13 </html>
```

Opción B: Desde la terminal del colaborador (La profesional)

Si tu colaborador quiere trabajar en su propio PC:

1. **Clonar:** `git clone https://github.com/DAW2-smc/tu-repo.git`
2. **Modificar:** Abrir el `index.html` y hacer los cambios.
3. **Subir:**

Bash

```
git add .
git commit -m "Cambio realizado por colaborador 123saabc"
git push origin main
```

💡 Lo que debes vigilar en TU ordenador (El Servidor)

Mientras el colaborador hace eso, estamos atentos a dos sitios:

1. **El terminal:** Donde tenemos escuchando los Jobs con `./run.sh`, aparecerá `job: deploy`.
2. **El navegador:** En cuanto el job termine, refrescamos `localhost:8080`. ¡Se verán los cambios que ha hecho el colaborador sin que tú hayas bajado nada de código!

```
sara@max11: /home/sara/Escritorio/UT06.CDCI/02.Ej2/runner-github
Archivo Editar Ver Buscar Terminal Ayuda
sara@max11:~/Escritorio/UT06.CDCI/02.Ej2/runner-github$ ./run.sh

✓ Connected to GitHub

Current runner version: '2.331.0'
2026-02-01 18:35:59Z: Listening for Jobs
2026-02-01 18:36:15Z: Running job: deploy
2026-02-01 18:36:25Z: Job deploy completed with result: Succeeded
2026-02-01 19:46:15Z: Running job: deploy
2026-02-01 19:46:22Z: Job deploy completed with result: Succeeded
█
```

¡Hola! Este sitio es el ejercicio 2.

Aprendiendo Docker y GitHub Actions.