

Trabajo Práctico 8 - Herramientas de construcción de software en la nube

Repetir el ejercicio 6.1 del trabajo práctico trabajo práctico 7 para el proyecto SimpleWebAPI, pero utilizando GitHub Actions.

En GitHub, en el repositorio donde se encuentra la aplicación SimpleWebAPI, ir a la opción Actions y crear un nuevo workflow.

El nombre de archivo puede ser main.yml y tendrá un contenido similar al siguiente:

name: Build and Publish

on:

workflow_dispatch:

push:

branches:

- main

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v3

- name: Setup .NET Core

uses: actions/setup-dotnet@v3

with:

dotnet-version: 7.0.x

- name: Restore dependencies

run: dotnet restore

- name: Build

run: dotnet build --configuration Release

- name: Publish

run: dotnet publish --configuration Release --output ./publish

- name: Upload Artifacts

uses: actions/upload-artifact@v3

with:

name: app

path: ./publish

deploy:

needs: build

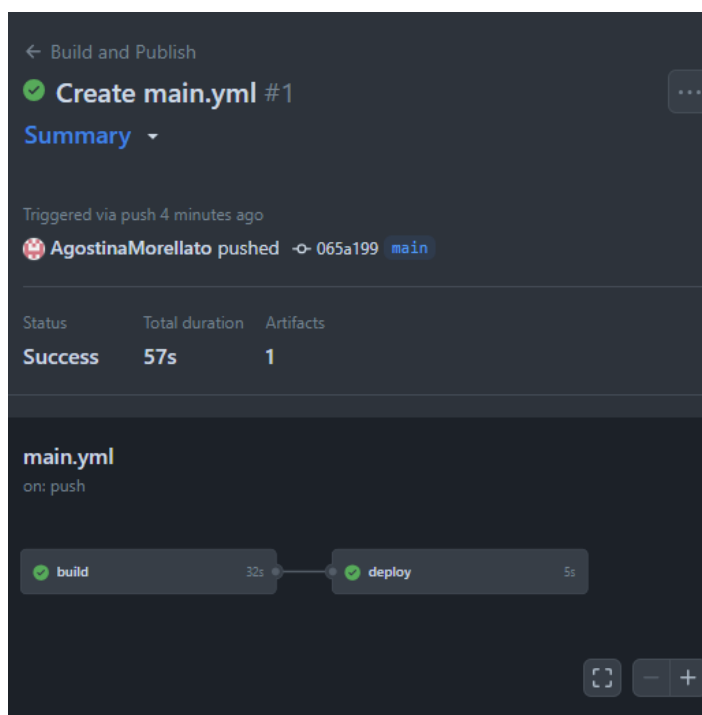
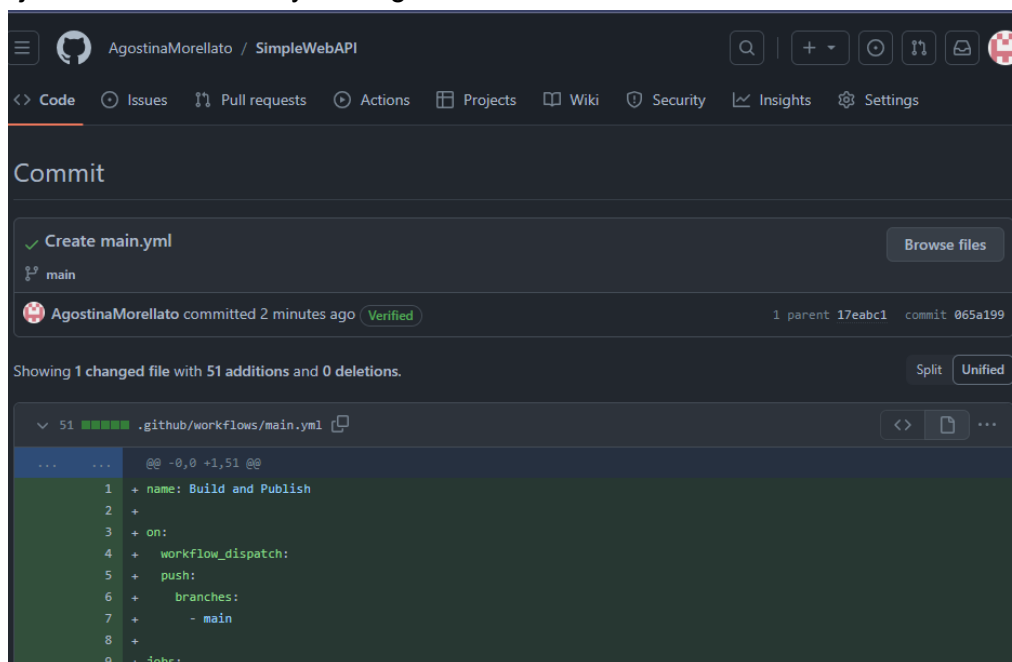
runs-on: ubuntu-latest

steps:

- name: Download Artifacts

uses: actions/download-artifact@v3
with:
 name: app
- name: Output contents
run: ls
- name: Deploy to Server
run: |
 echo "Deploy"

Guardar el archivo (hacemos commit directamente en GitHub por ejemplo) y vemos la ejecución del worflow y sus logs.



Explicar paso a paso que realiza el pipeline anterior:

name: Build and Publish: Este es el nombre del workflow y con el que se lo identificará.

workflow_dispatch: Esto permite que sea manualmente desencadenado por un usuario a través de la interfaz de GitHub.

push: Este evento se desencadena automáticamente cuando se realizan cambios en el repositorio.

branches: - main: El flujo de trabajo se ejecutará automáticamente cuando haya un push en la rama "main".

En este flujo de trabajo, hay dos trabajos definidos: **build y deploy**. Estos trabajos se ejecutarán en paralelo si son desencadenados.

BUILD

Este trabajo se encarga de construir la aplicación.

build:

runs-on: ubuntu-latest

runs-on: ubuntu-latest: Este trabajo se ejecutará en una instancia de Ubuntu.

Serie de pasos que se ejecutarán secuencialmente:

1. Checkout code: descarga el código fuente del repositorio.
2. Setup .NET Core: se configura .NET Core en el entorno.
3. Restore dependencies: restaura las dependencias del proyecto.
4. Build: compila la aplicación con la configuración "Release".
5. Publish: publica la aplicación compilada en el directorio llamado "publish".
6. Upload Artifacts: Carga los archivos resultantes (artefactos) al flujo de trabajo para su posterior uso.

DEPLOY

Este trabajo se encarga del despliegue de la aplicación.

deploy:

needs: build

runs-on: ubuntu-latest

needs: build: Este depende del trabajo "build", lo que significa que solo se ejecutará si el trabajo "build" tiene éxito.

Luego, se definen los siguientes pasos:

1. Download Artifacts: Descarga los artefactos generados en el trabajo "build" que se cargaron previamente.
2. Output contents: Muestra el contenido de la carpeta actual, lo cual es útil para verificar qué se ha descargado.
3. Deploy to Server: Este paso ejecuta un comando ficticio que simplemente imprime "Deploy". En una implementación real, aquí sería el lugar donde se realizaría el despliegue real de la aplicación en un servidor.

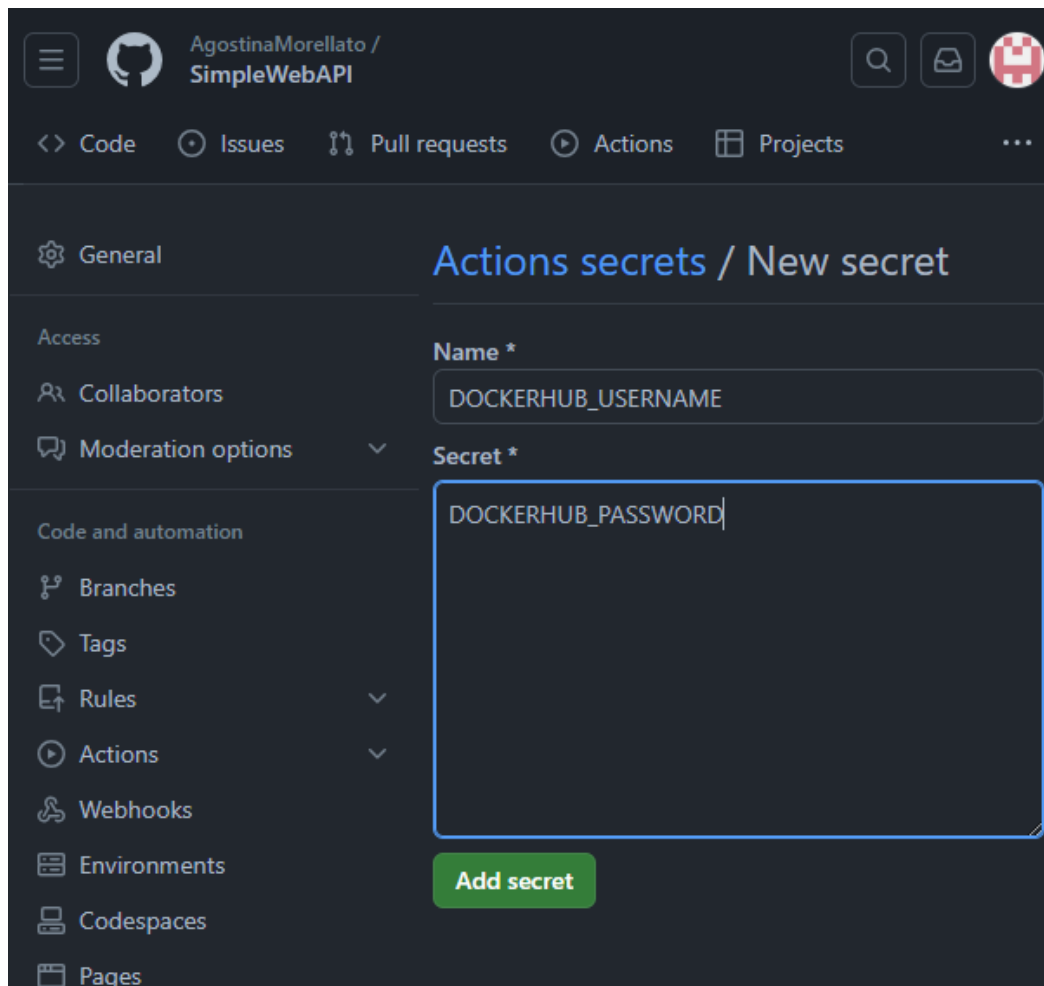
3- Configurar un workflow en GitHub Actions para generar una imagen de Docker de SimpleWebApi y subirla a DockerHub

En GitHub Actions generar una acción que genere una imagen de docker con nuestra aplicación SimpleWebAPI y la suba a DockerHub

Generar secretos y los pasos necesarios para subir la imagen a Docker Hub. Referencia

Paso 1: Configurar las credenciales de Docker Hub en tu repositorio de GitHub:

En tu repositorio de GitHub, ve a "Settings" (Configuración) > "Secrets" (Secretos). Haz clic en "New repository secret" (Nuevo secreto del repositorio). Define dos secretos: uno para el nombre de usuario de Docker Hub y otro para la contraseña de Docker Hub. Puedes nombrar estos secretos como DOCKERHUB_USERNAME y DOCKERHUB_PASSWORD, respectivamente.



The screenshot shows the GitHub interface for the repository 'SimpleWebAPI' by user 'AgostinaMorellato'. The 'Actions' tab is selected in the top navigation bar. On the left sidebar, under 'Code and automation', the 'Secrets' option is highlighted. The main content area is titled 'Actions secrets / New secret'. It contains a form with two fields: 'Name *' with the value 'DOCKERHUB_USERNAME' and 'Secret *' with the value 'DOCKERHUB_PASSWORD'. A green 'Add secret' button is located at the bottom of the form.

Paso 2: Crear un workflow para construir y subir la imagen de Docker:

name: Docker Image CI

on:

workflow_dispatch:

push:

branches: ["main"]

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Build the Docker image

run: docker build . --file Dockerfile --tag \${{ secrets.DOCKERHUB_USERNAME }}/\${{ secrets.DOCKERHUB_USERNAME }}/simple-web-api-gh:latest

- name: Log in to Docker Hub

run: docker login -u \${{ secrets.DOCKERHUB_USERNAME }} -p \${{ secrets.DOCKERHUB_PASSWORD }}

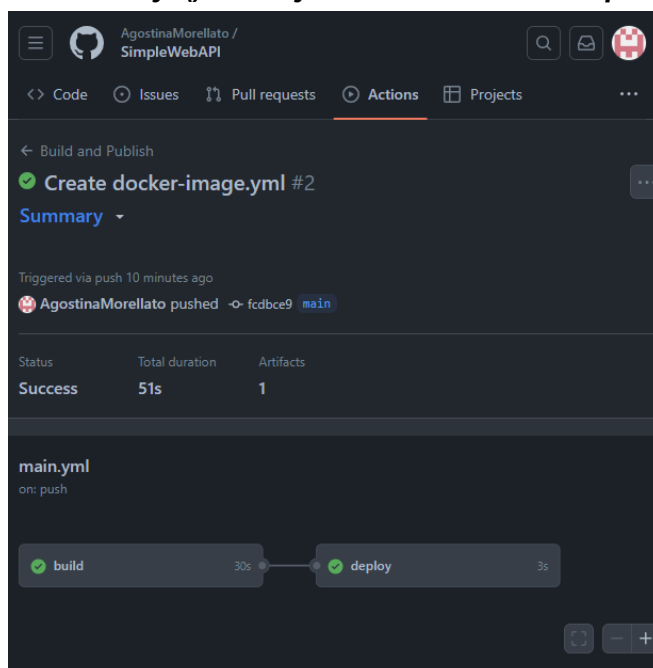
- name: Push Docker image to Docker Hub

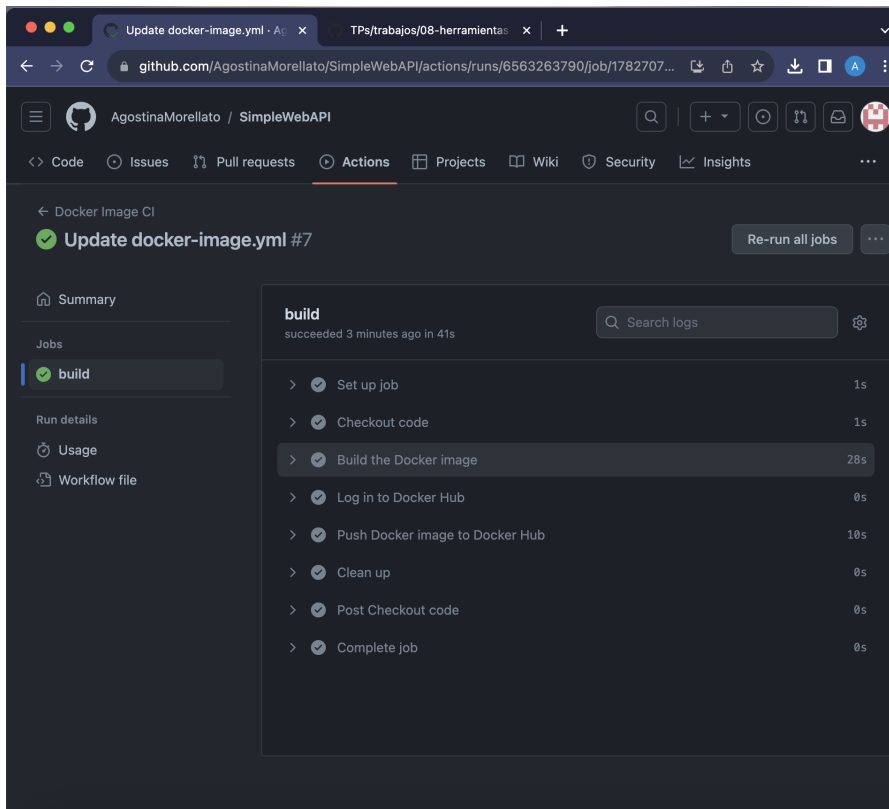
run: |
docker push \${{ secrets.DOCKERHUB_USERNAME }}/\${{ secrets.DOCKERHUB_USERNAME }}/simple-web-api-gh:latest

- name: Clean up

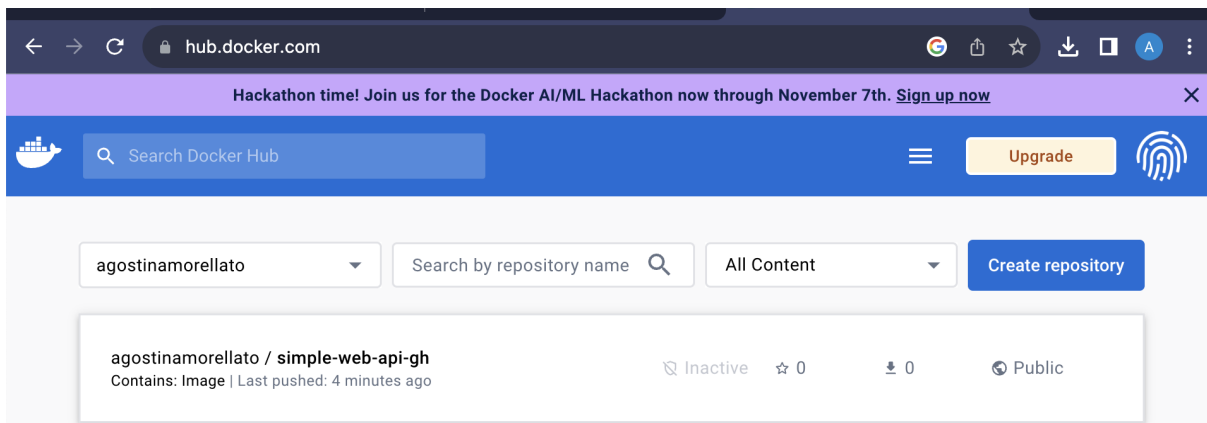
run: docker logout

if: always() # Se ejecutará incluso si un paso anterior falla





Paso 3: Verificar en DockerHub que la imagen ha sido subida



Paso 4: Descargar la imagen

docker pull agostinamorellato/simple-web-api-gh:latest

```

[agostinamorellato@Agostinas-MacBook-Pro Practicos % docker pull agostinamorellato/simple-web-api-gh:latest
latest: Pulling from agostinamorellato/simple-web-api-gh
e67fdae35593: Pull complete
0ab66724116f: Pull complete
14ccddebb1bc: Pull complete
5e265b51b431: Pull complete
3bda6efdfc5b: Pull complete
cf4222aa1f5d: Pull complete
4f4fb700ef54: Pull complete
13824f46db50: Pull complete
Digest: sha256:d1b0020e4e169330c75e2c03e3ade8bd8d113dc117374ef49960aac9d793fdcd
Status: Downloaded newer image for agostinamorellato/simple-web-api-gh:latest
docker.io/agostinamorellato/simple-web-api-gh:latest

What's Next?
View summary of image vulnerabilities and recommendations → docker scout quickview agostinamorellato/simple-web-api-gh:la
test
agostinamorellato@Agostinas-MacBook-Pro Practicos %

```

Paso 5: Crear el contenedor

docker run --name myapi -d -p 8080:80 agostinamorellato/simple-web-api-gh

Paso 6: Navegar a <http://localhost:8080/weatherforecast>

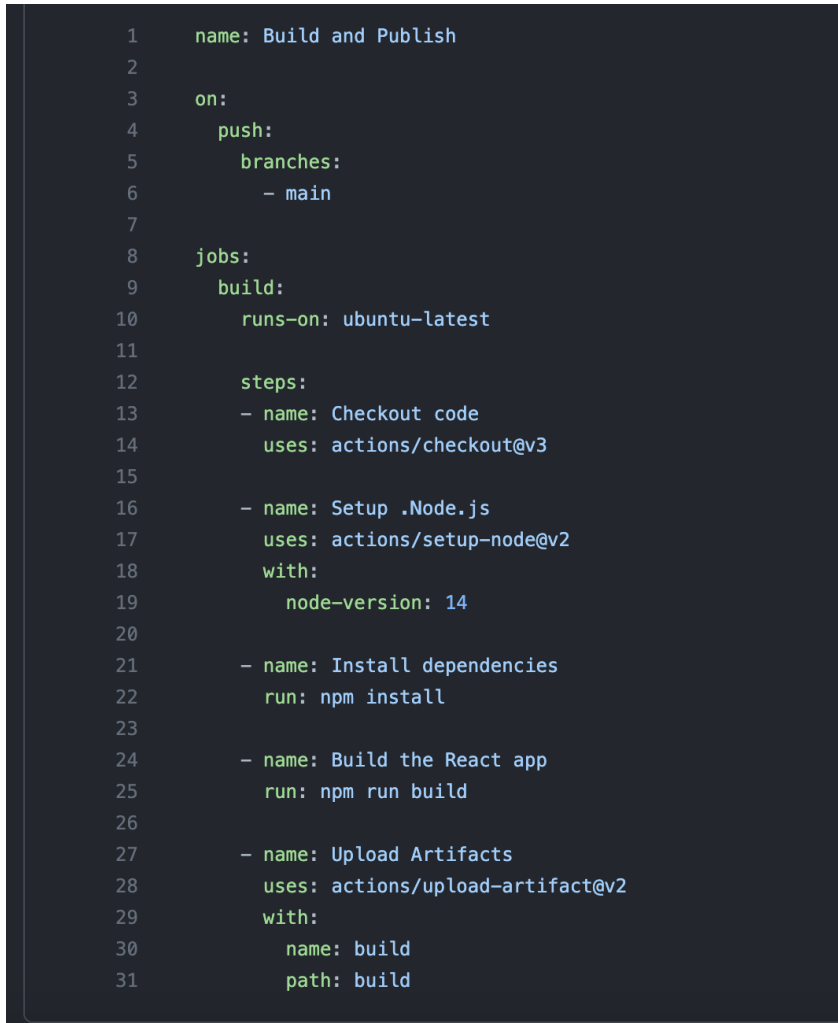


A screenshot of a web browser window. The address bar shows 'localhost:8080/weatherforecast'. The page content displays a JSON array of weather forecast data for five consecutive days. Each day's entry includes a date, temperature in Celsius and Fahrenheit, and a summary.

```
[{"date": "2023-10-19", "temperatureC": 37, "temperatureF": 98, "summary": "Balmy"}, {"date": "2023-10-20", "temperatureC": 22, "temperatureF": 71, "summary": "Mild"}, {"date": "2023-10-21", "temperatureC": -14, "temperatureF": 7, "summary": "Sweltering"}, {"date": "2023-10-22", "temperatureC": 46, "temperatureF": 114, "summary": "Fresquito"}, {"date": "2023-10-23", "temperatureC": 43, "temperatureF": 109, "summary": "Mild"}]
```

4- Crear una GitHub Action que genere los artefactos para el proyecto React

En GitHub Actions generar una acción que genere los artefactos para el Ejercicio 2 del TP 5




A screenshot of a GitHub Actions workflow file. The workflow is named 'Build and Publish' and is triggered on a push to the 'main' branch. It consists of a single job named 'build' that runs on 'ubuntu-latest'. The job has five steps: 'Checkout code', 'Setup .Node.js', 'Install dependencies', 'Build the React app', and 'Upload Artifacts'.

```
1  name: Build and Publish
2
3  on:
4    push:
5      branches:
6        - main
7
8  jobs:
9    build:
10     runs-on: ubuntu-latest
11
12     steps:
13       - name: Checkout code
14         uses: actions/checkout@v3
15
16       - name: Setup .Node.js
17         uses: actions/setup-node@v2
18         with:
19           node-version: 14
20
21       - name: Install dependencies
22         run: npm install
23
24       - name: Build the React app
25         run: npm run build
26
27       - name: Upload Artifacts
28         uses: actions/upload-artifact@v2
29         with:
30           name: build
31           path: build
```

5- Crear una GitHub Action que genere una imagen de Docker para el proyecto React y lo suba a DockerHub

→ github.com/AgostinaMorellato/SimpleWebAPI/new/main?filename=.github%2Fworkflows%2Fmain.yml&workflow_template=t

SimpleWebAPI / .github / workflows / in

Edit Preview  Code 55% faster with GitHub Copilot

```
1  name: Build and Upload Docker Image CI
2
3  on:
4    push:
5      branches:
6        - main
7
8  jobs:
9    build:
10     runs-on: ubuntu-latest
11
12     steps:
13       - name: Checkout Repository
14         uses: actions/checkout@v2
15
16       - name: Build the Docker image
17         run: docker build . --file Dockerfile --tag ${ secrets.DOCKERHUB_USERNAME }}/simple-web-api-gh:latest
18
19       - name: Log in to Docker Hub
20         run: docker login -u ${ secrets.DOCKERHUB_USERNAME }} -p ${ secrets.DOCKERHUB_PASSWORD }}
21
22       - name: Push Docker image to Docker Hub
23         run: |
24           docker push ${ secrets.DOCKERHUB_USERNAME }}/simple-web-api-gh:latest
25
26       - name: Clean up
27         run: docker logout
28         if: always() # Se ejecutará incluso si un paso anterior falla
29
30
31
```