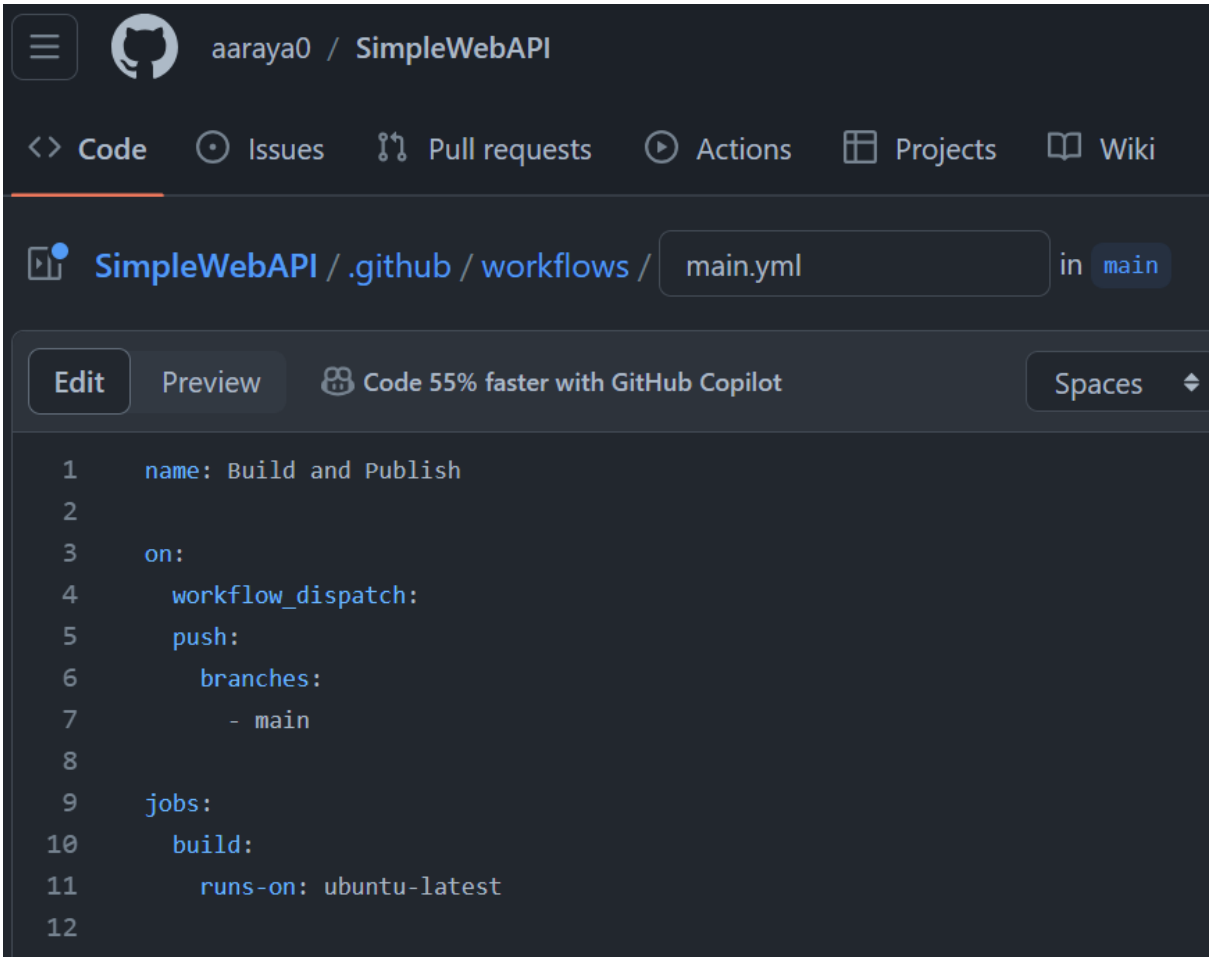
Trabajo Práctico N°8

Ejercicio 2




```
on:
  workflow_dispatch:
  push:
    branches:
      - main
```

- Define cuándo se debe ejecutar este workflow:
  - `workflow_dispatch`: Permite ejecutar el workflow manualmente desde la interfaz de GitHub.
  - `push`: Se ejecuta cuando se hace un push a la rama especificada (`main` en este caso).

Trabajo "build"

```
yamlCopy code
jobs:
```

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

- Define un trabajo llamado "build" que se ejecutará en la última versión de Ubuntu.

Pasos del Trabajo "build"

```
yamlCopy code
    steps:
    - name: Checkout code
      uses: actions/checkout@v3
```

- `actions/checkout@v3` es una acción predefinida que clona el repositorio de GitHub en el entorno de ejecución. Esto asegura que el código esté disponible para ser procesado.

```
yamlCopy code
    - name: Setup .NET Core
      uses: actions/setup-dotnet@v3
      with:
        dotnet-version: 7.0.x
```

- Configura el entorno de ejecución para utilizar una versión específica de .NET Core (en este caso, 7.0.x).

```
yamlCopy code
    - name: Restore dependencies
      run: dotnet restore
```

- Utiliza el comando `dotnet restore` para restaurar las dependencias del proyecto.

```
yamlCopy code
    - name: Build
      run: dotnet build --configuration Release
```

- Compila el proyecto en modo de liberación (`Release`) utilizando el comando `dotnet build`.

```
yamlCopy code
    - name: Publish
      run: dotnet publish --configuration Release --output ./publish
```

- Publica la aplicación utilizando el comando `dotnet publish`, con la configuración de liberación y especificando el directorio de salida como `./publish`.

```
yamlCopy code
    - name: Upload Artifacts
      uses: actions/upload-artifact@v3
      with:
```

```
        name: app
        path: ./publish
```

- Sube los artefactos de la aplicación (el resultado de la publicación) para su uso posterior.

Trabajo "deploy"

```
yamlCopy code
  deploy:
    needs: build
    runs-on: ubuntu-latest
```

- Define un segundo trabajo llamado "deploy" que se ejecutará después de que el trabajo "build" haya terminado (`needs: build`).

Pasos del Trabajo "deploy"

```
yamlCopy code
    steps:
    - name: Download Artifacts
      uses: actions/download-artifact@v3
      with:
        name: app
```

- Descarga los artefactos de la aplicación que se generaron en el trabajo "build".

```
yamlCopy code
    - name: Output contents
      run: ls
```

- Muestra el contenido de los artefactos descargados.

```
yamlCopy code
    - name: Deploy to Server
      run: |
        echo "Deploy"
```

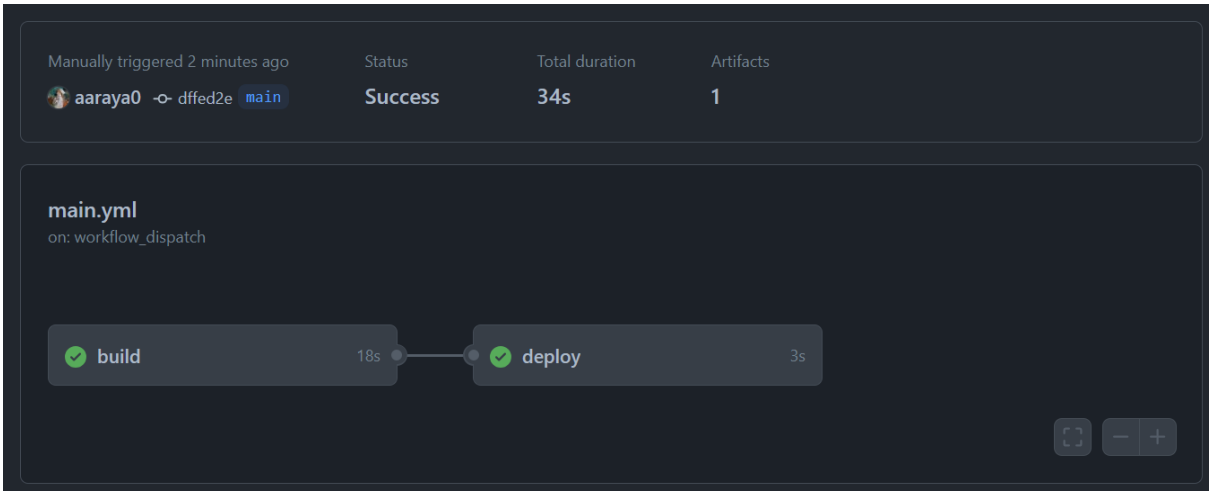
- Simula un paso de despliegue. En este caso, simplemente imprime "Deploy".

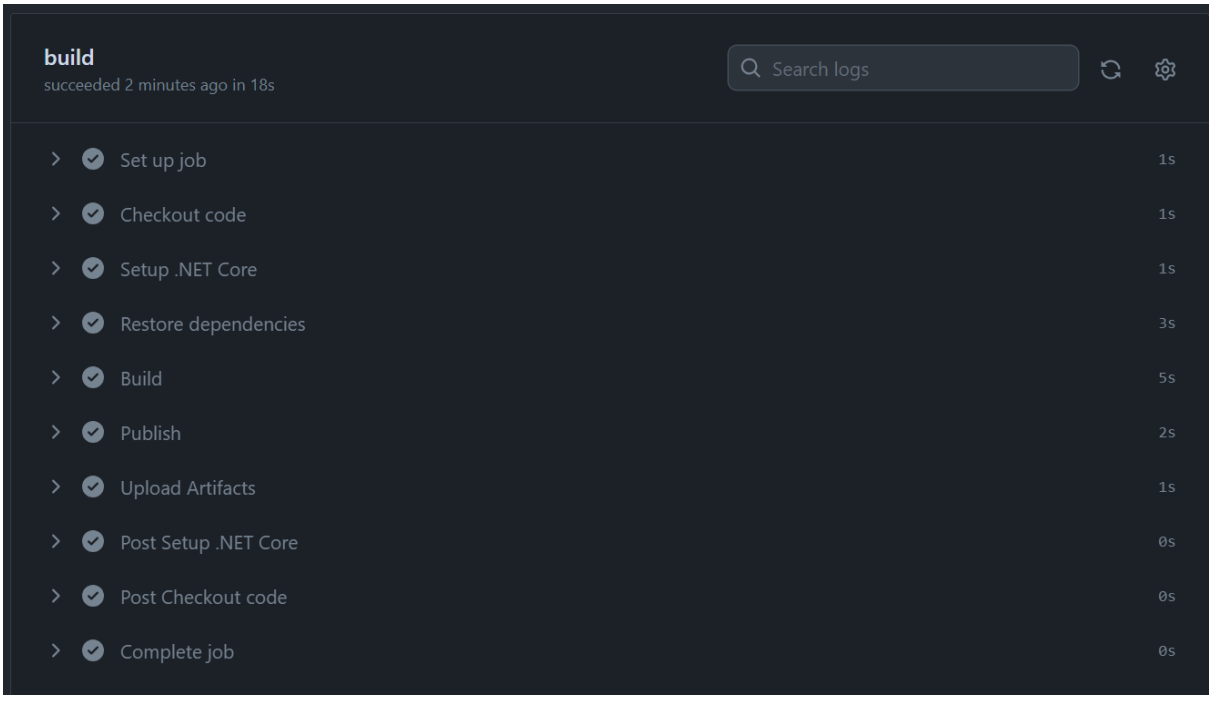
Este pipeline realiza los siguientes pasos:

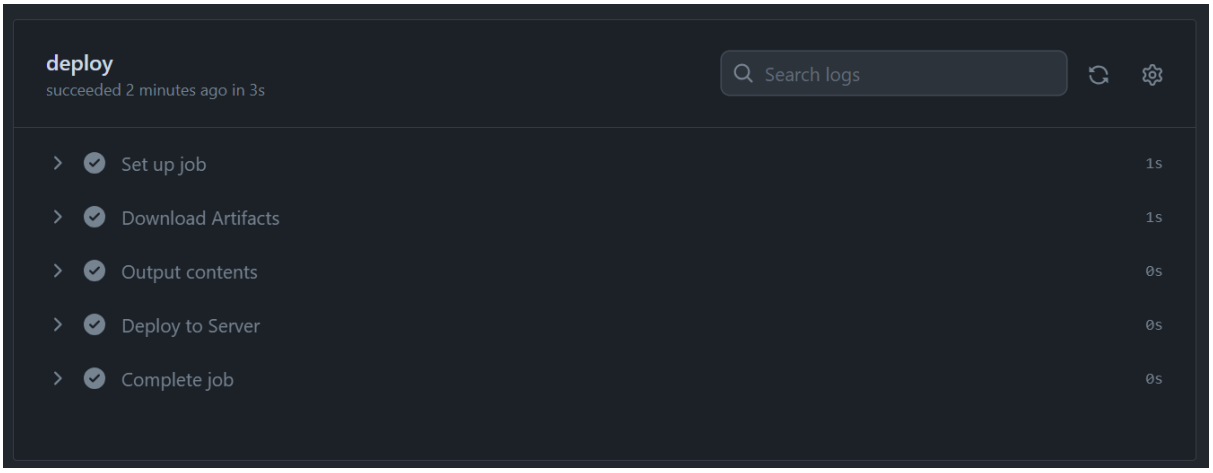
- Clona el repositorio.
- Configura el entorno para .NET Core 7.0.x.
- Restaura las dependencias del proyecto.
- Compila el proyecto en modo de liberación.
- Publica la aplicación en un directorio específico.

- Sube los artefactos generados.
- Descarga los artefactos para el despliegue.
- Muestra el contenido de los artefactos.
- Simula un paso de despliegue (puede ser personalizado según las necesidades reales).

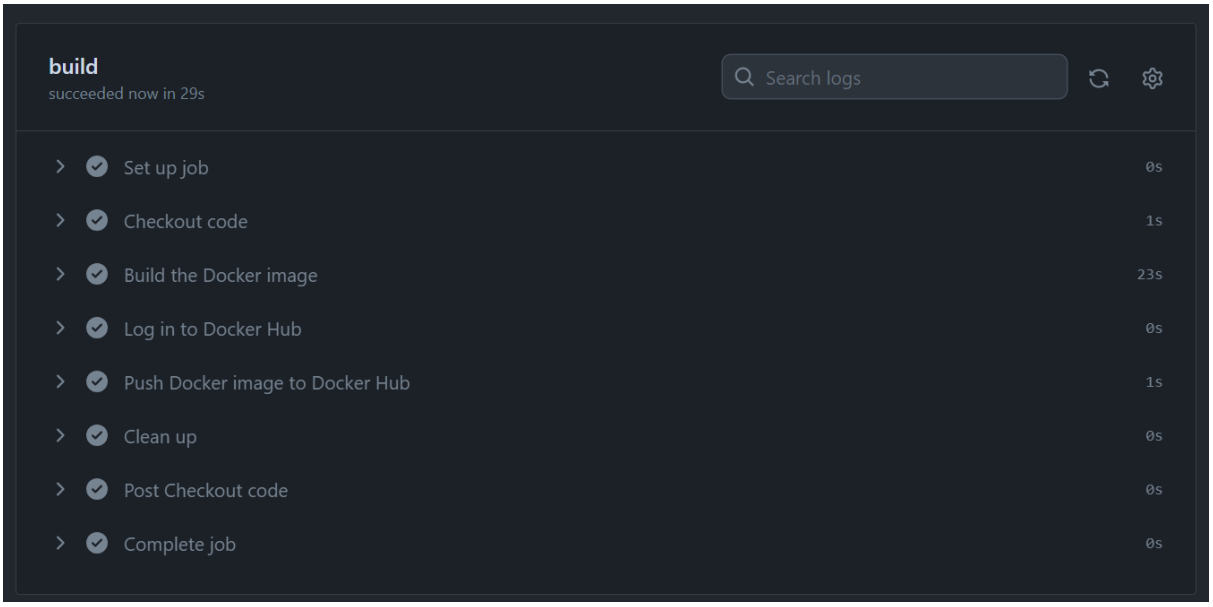
Este pipeline automatiza la construcción y publicación de la aplicación SimpleWebAPI en GitHub Actions, facilitando el proceso de desarrollo y despliegue.

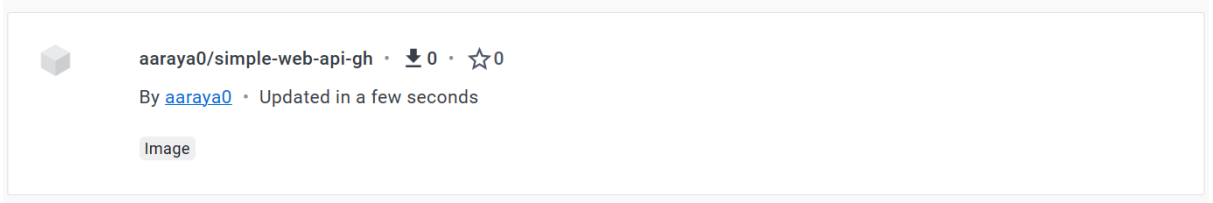





deploy
succeeded 2 minutes ago in 3s
Search logs
Set up job
1s
Download Artifacts
1s
Output contents
0s
Deploy to Server
0s
Complete job
0s

Ejercicio 3


build
succeeded now in 29s
Search logs
Set up job
0s
Checkout code
1s
Build the Docker image
23s
Log in to Docker Hub
0s
Push Docker image to Docker Hub
1s
Clean up
0s
Post Checkout code
0s
Complete job
0s

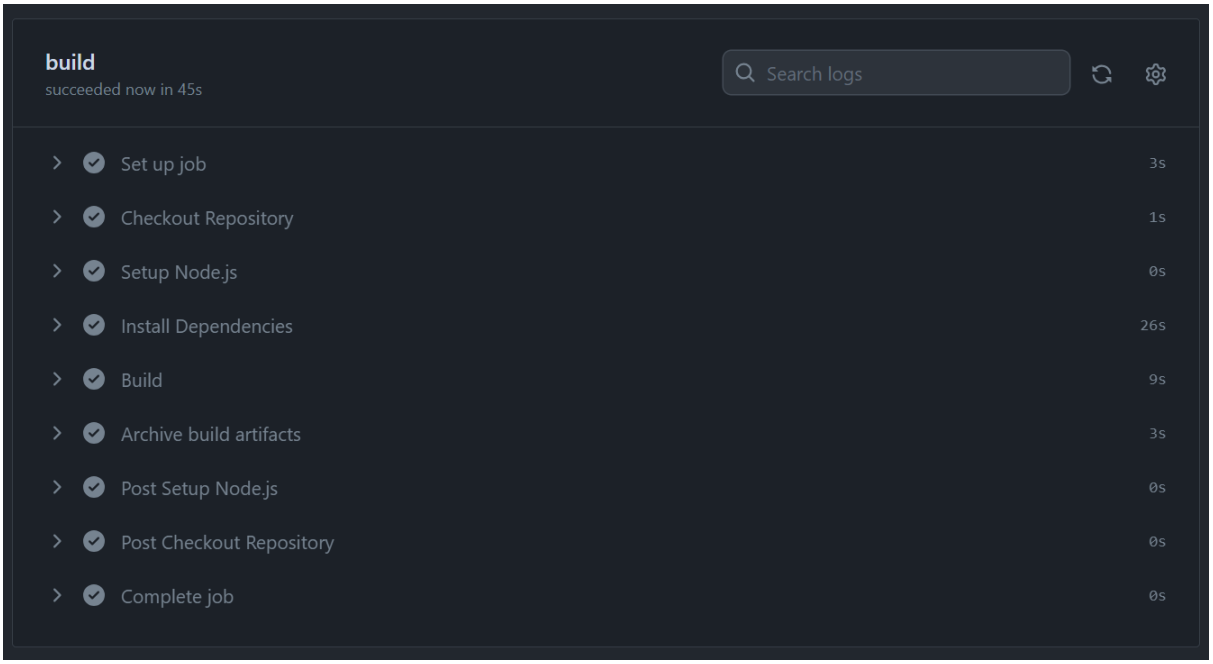

aaraya0/simple-web-api-gh
0
0
By aaraya0
Updated in a few seconds
Image

```
Usuario@DESKTOP-8G1HL0R MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3/tp8
$ docker pull aaraya0/simple-web-api-gh:latest
latest: Pulling from aaraya0/simple-web-api-gh
7dbc1adf280e: Pull complete
969d48310aaa: Pull complete
2194c6af6861: Pull complete
98003354b5af: Pull complete
a5db88be08ca: Pull complete
d2a75afc8974: Pull complete
4f4fb700ef54: Pull complete
2a63f016004c: Pull complete
Digest: sha256:85cb928fa71418fb0fb4609d716c5f848886366bf5ac1ba8fd349091253ba9fb
Status: Downloaded newer image for aaraya0/simple-web-api-gh:latest
docker.io/aaraya0/simple-web-api-gh:latest
```

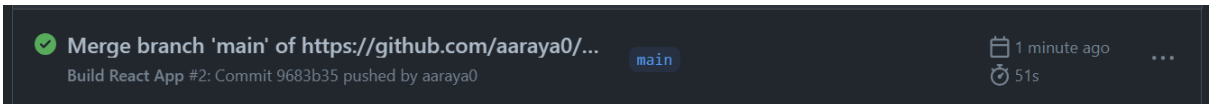
```
Usuario@DESKTOP-8G1HL0R MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3/tp8
$ docker run --name myapi -d -p 8080:80 aaraya0/simple-web-api-gh
89411dfdbb22384907c461d96b13cac2759f69b6592d7f6923d3e0c2dfc5816e
```

Ejercicio 4

```yaml
name: Build React App

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout Repository
      uses: actions/checkout@v2

    - name: Setup Node.js
      uses: actions/setup-node@v2
      with:
        node-version: 14

    - name: Install Dependencies
      run: npm install

    - name: Build
      run: npm run build

    - name: Archive build artifacts
      uses: actions/upload-artifact@v2
      with:
        name: build
        path: build
```




```
Usuario@DESKTOP-8G1HL0R MINGW64 ~/OneDrive/Escritorio/2023/Ing SW 3/tp5/my-app (main)
$ git push
Enumerating objects: 9, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 602 bytes | 301.00 KiB/s, done.
Total 5 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To https://github.com/aaraya0/my-app.git
   bb13a8f..9683b35  main -> main
```

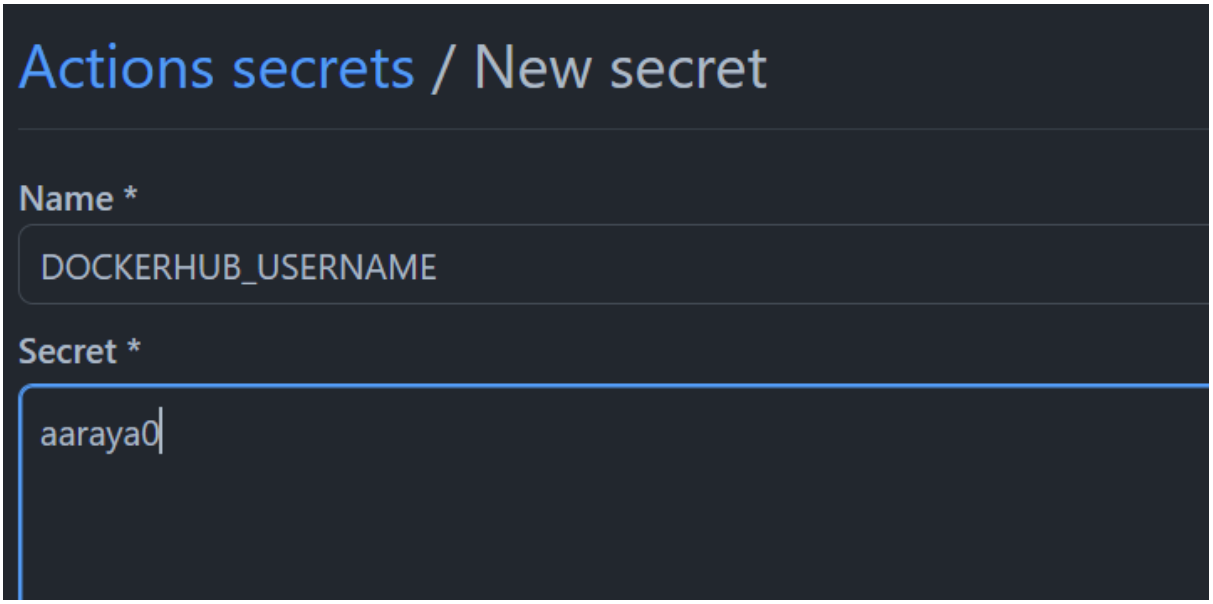


Ejercicio 5

Primero, se agregó un dockerfile al repositorio:

```
# Usa una imagen base de Node.js
FROM node:14

# Establece el directorio de trabajo en /app
WORKDIR /app

# Copia el package.json y el package-lock.json (si existe)
COPY package*.json ./

# Instala las dependencias
RUN npm install

# Copia todos los archivos del proyecto a /app
COPY . .
```

```
# Construye la aplicación React
RUN npm run build

# Configura el servidor web para servir la aplicación
EXPOSE 80
CMD [ "npx", "serve", "-s", "build", "-l", "80" ]
```




```
name: Build and Push Docker Image

on:
  push:
    branches:
      - main

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout Repository
      uses: actions/checkout@v2

    - name: Build Docker Image
      run: docker build . --file Dockerfile --tag ${{ secrets.DOCKERHUB_USERNAME }}/my-react-app: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 }}/my-react-app:latest

    - name: Clean up
      run: docker logout
      if: always() # Se ejecutará incluso si un paso anterior falla
```

Lista de pasos que se ejecutarán en el trabajo:

- `Checkout Repository`: Clona el repositorio.
- `Build Docker Image`: Construye la imagen de Docker.
- `Log in to Docker Hub`: Inicia sesión en Docker Hub.
- `Push Docker Image to Docker Hub`: Sube la imagen de Docker a Docker Hub.
- `Clean up`: Cierra sesión en Docker Hub.