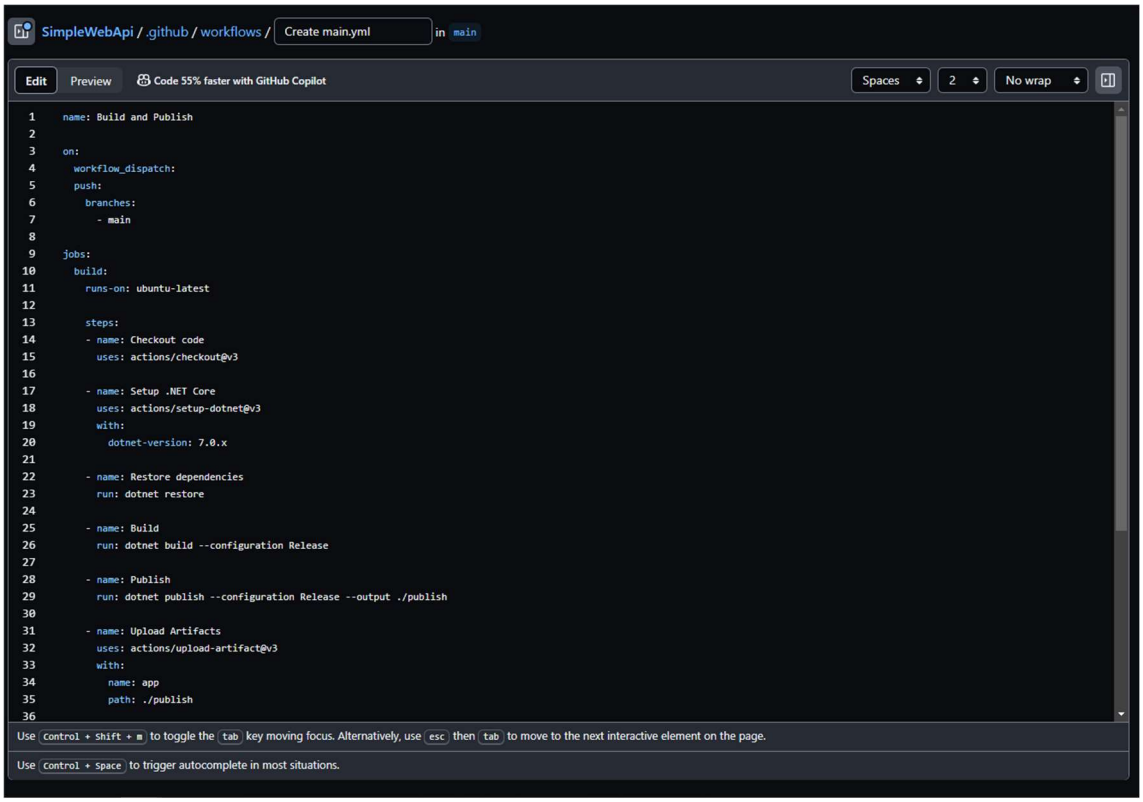
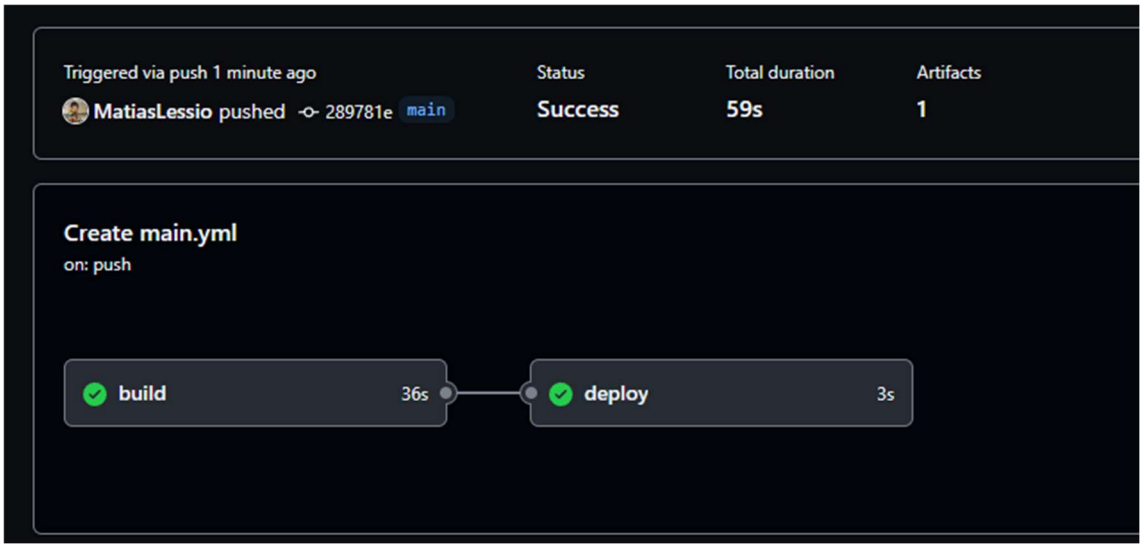


1. **Triggers (Desencadenadores):**
    - **workflow_dispatch**: Este es un desencadenador manual que permite a un usuario iniciar el flujo de trabajo de forma manual desde la interfaz de GitHub. Esto puede ser útil para ejecutar el flujo de trabajo bajo demanda.
    - **push** en la rama **main**: Este desencadenador automático inicia el flujo de trabajo cada vez que se realiza un "push" a la rama principal del repositorio. En otras palabras, cuando se realizan cambios en la rama **main**, se inicia automáticamente el flujo de trabajo.
2. **Fase de Construcción (Build):**
    - En esta fase, el flujo de trabajo se encarga de compilar y preparar la aplicación para su despliegue.
    - Comienza extrayendo el código fuente del repositorio utilizando **actions/checkout**.
    - Luego, configura la versión de .NET Core en 7.0.x utilizando **actions/setup-dotnet**.
    - A continuación, se restauran las dependencias del proyecto con **dotnet restore**.
    - La aplicación se construye con **dotnet build** utilizando la configuración "Release".
    - La aplicación se publica en un directorio llamado "publish" utilizando **dotnet publish**.

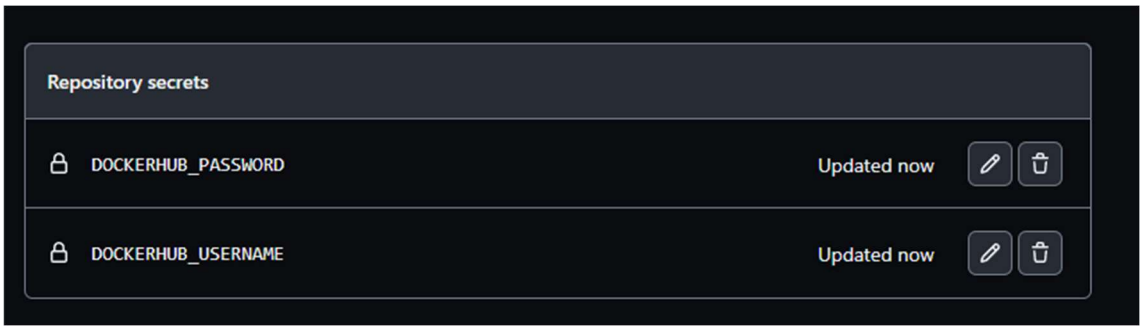
- Finalmente, los archivos generados se almacenan como un "artifact" llamado "app" mediante **actions/upload-artifact**. Los "artifacts" son datos generados por el flujo de trabajo que se pueden utilizar en fases posteriores, como la de despliegue.

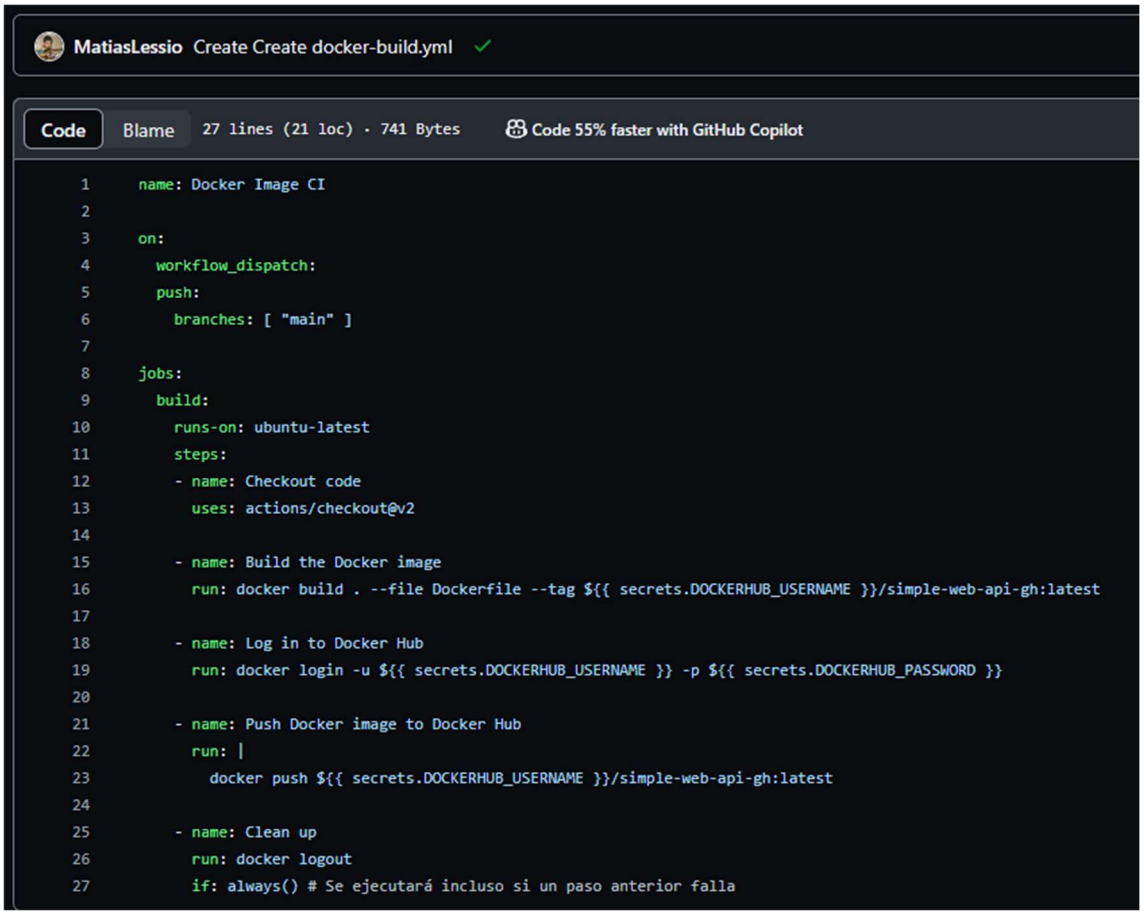
3. **Fase de Despliegue (Deploy):**

    - Esta fase comienza después de que la fase de construcción se haya completado satisfactoriamente. Esto se logra mediante la declaración **needs: build**, que asegura que los artefactos generados en la fase de construcción estén disponibles en esta fase.
    - Los artefactos generados en la fase de construcción se descargan utilizando **actions/download-artifact**.
    - Luego, se muestra el contenido del directorio actual con el comando **ls**.
    - Finalmente, se ejecuta un comando personalizado, que en este caso simplemente imprime "Deploy."

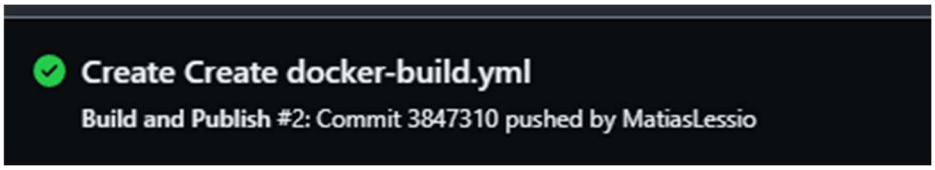


3- Configurar un worflow en GitHub Actions para generar una imagen de Docker de SimpleWebApi y subirla a DockerHub






```yaml
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 }}/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 }}/simple-web-api-gh:latest

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




```
C:\Users\Matias>docker pull matiaslessio/simplewebapi:simplewebapi
simplewebapi: Pulling from matiaslessio/simplewebapi
Digest: sha256:c6a55b9353b54e36beacbeda1d1b09bb14db23ecb63885be65a1b510798e31a9
Status: Image is up to date for matiaslessio/simplewebapi:simplewebapi
docker.io/matiaslessio/simplewebapi:simplewebapi
```

```
D:\Facu\ingsw3\SimpleWebApi>docker run --name myapi -d -p 9000:90 matiaslessio/simplewebapi:simplewebapi
9f8288987a1ec29d1a19157ba623baf82b6467f367a5b6157d7999129f58d493
```

```
[{"date":"2023-09-27","temperatureC":20,"temperatureF":67,"summary":"Mild"},{"date":"2023-09-
28","temperatureC":37,"temperatureF":98,"summary":"Balmy"},{"date":"2023-09-
29","temperatureC":50,"temperatureF":121,"summary":"Scorching"},{"date":"2023-09-
30","temperatureC":34,"temperatureF":93,"summary":"Mild"},{"date":"2023-10-
01","temperatureC":45,"temperatureF":112,"summary":"Chilly"}]
```