

## DevOps e Integración Continua

### PRÁCTICA 3 - Construcción de un Pipeline con Github Actions

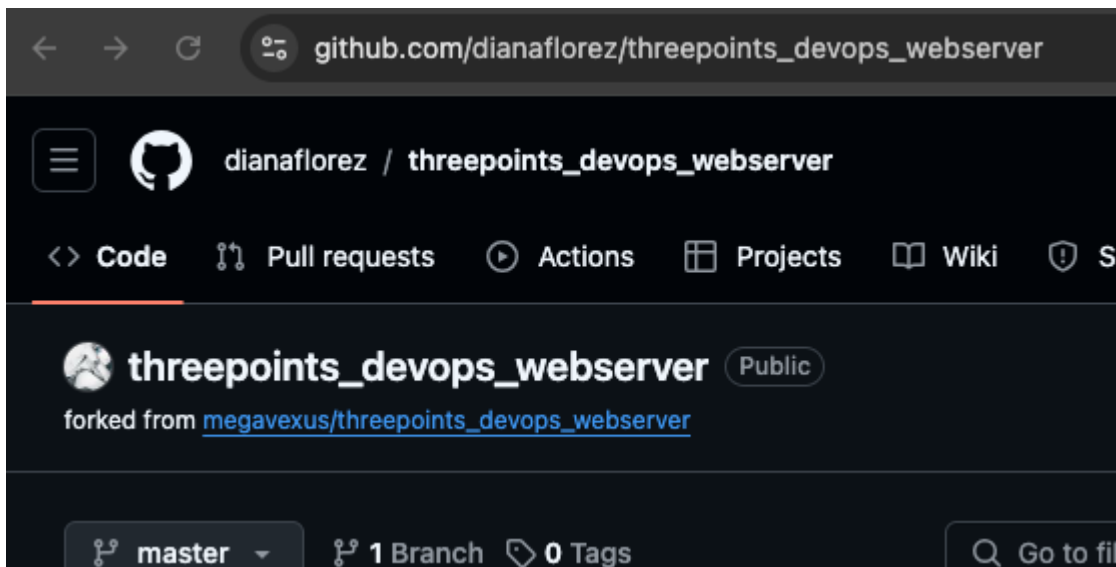
Presentado a: **Javier Gutiérrez Navío**  
Presentado Por: **Diana Cristy Florez Bravo**

#### Objetivo

El objetivo de esta práctica es aprender los fundamentos de uso de los pipelines en Github mediante Github Actions, y entender las diferencias y similitudes con los pipelines declarativos de Jenkins anteriormente dados.

**0 – Preparación: Forqueo del repositorio a utilizar** Para poder realizar esta práctica, vamos a usar un proyecto dockerizado básico de node. Podéis encontrarlo de las siguientes maneras: - En el repositorio [https://github.com/megavexus/threepoints\\_devops\\_webserver](https://github.com/megavexus/threepoints_devops_webserver) - En la asignatura, siguiendo el path: Actividades de Evaluación > Laboratorio de Prácticas > Proyecto Node. Haría falta o bien, realizar un Fork del proyecto, o subir a un repositorio propio el proyecto. Deberá incluirse el link del repositorio del alumno. Nota: Si en la práctica 1 se ha hecho esto, podría reutilizarse el mismo repositorio.

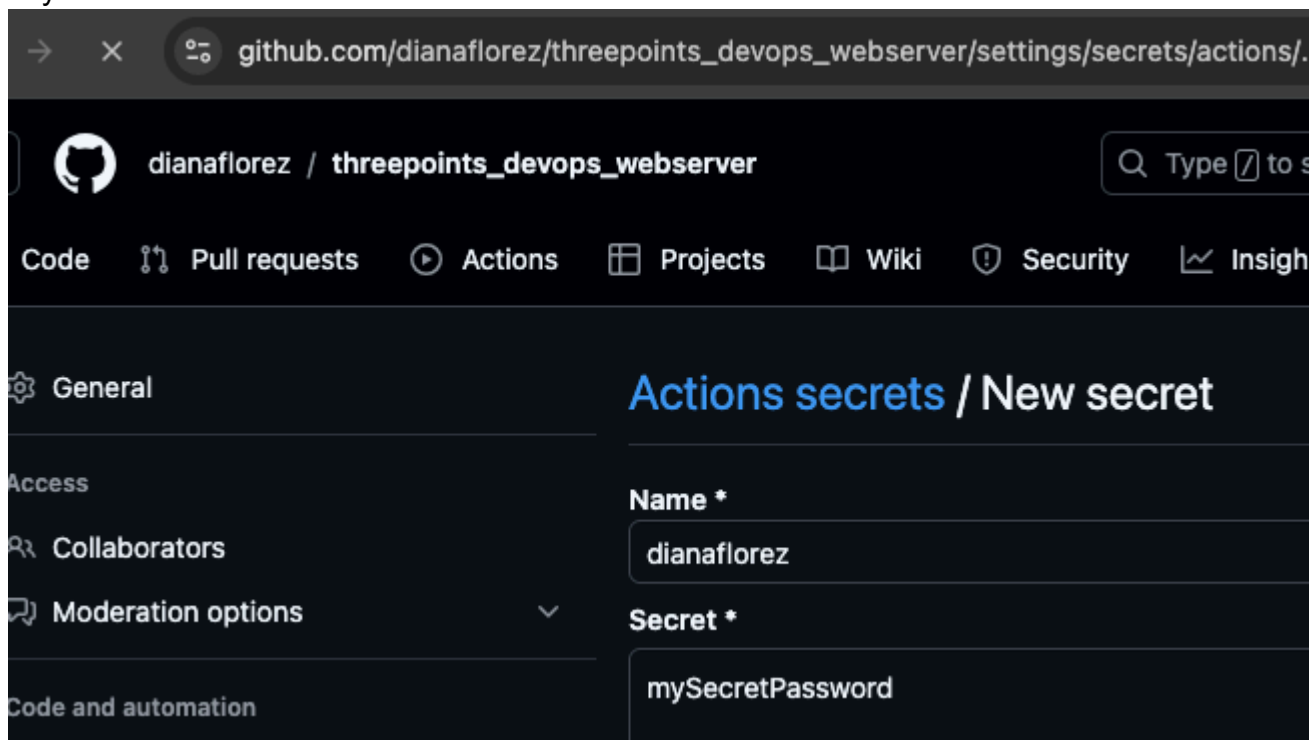
[https://github.com/dianaflorez/threepoints\\_devops\\_webserver](https://github.com/dianaflorez/threepoints_devops_webserver)



#### 1 – Creación de un pipeline declarativo básico de CI

Durante este primer ejercicio, crearemos un pipeline básico que construya el proyecto de un repositorio de git. Para ello, se usará el proyecto en Github, que deberíamos haber subido a un repositorio personal (y accesible públicamente) cada uno, en el ejercicio previo. Primero hay que dar de alta dos secretos que serán las credenciales, llamados “USERNAME” y “PASSWORD”.

El valor de USERNAME tiene que ser el nombre del alumno, y el de PASSWORD, "mySecretPassword".



Habr  que crear un nuevo workflow de github, que consista en un pipeline b sico (que llamaremos continous-integration-.yaml) que realice ciertas comprobaciones antes de construir y lanzar el Docker con el proyecto que tenemos. Este workflow se debe de ejecutar cuando se haga un Push a la rama "master". El pipeline debe de realizar los siguientes pasos: (Nota: La fase de hacer checkout no ser a necesario, debido a que ya tienes el c digo en el propio repositorio)

1. Ejecuci n de pruebas de calidad de c digo. Al igual que en la anterior pr ctica, este paso lo haremos mockeado por posibles problemas t cnicos de la m quina virtual en algunos ordenadores. Ejecutaremos un "echo" en el sistema para imprimir "Ejecuci n de pruebas de SAST". Este paso se debe de llamar "Pruebas de SAST".
2. Crear un archivo de credenciales. Basado en el modelo del archivo "credentials.ini.tpl", hay que crear un archivo con la misma estructura, llamado "credentials.ini" y cambiando las variables de \${USERNAME} y \${PASSWORD} por los secretos dados de alta anteriormente.
3. Construcci n del container de Docker. Para ejecutar esto, hace falta ejecutar el comando 'docker build -t devops\_ws . --tag devops\_ws:\$(date +%s)' Este paso se debe de llamar "Build docker image".

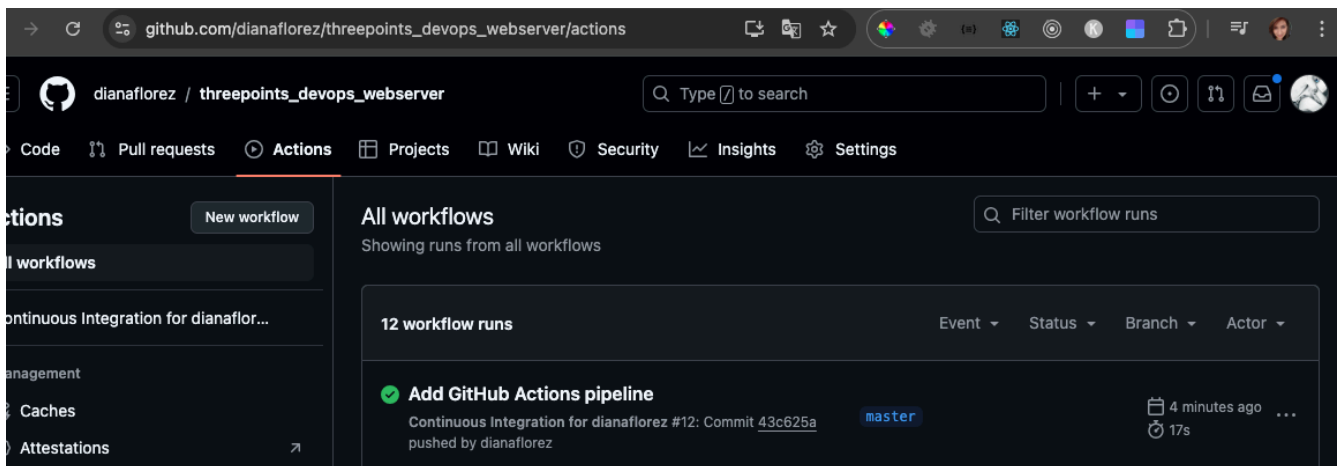
El Job se debe de entregar: - El archivo del workflow. - Captura de pantalla de los secretos dados de alta. - Capturas del pipeline ejecutado correctamente. - Capturas del archivo "credentials.ini" generado. - Capturas de pantalla del log de la ejecuci n.

```

Welcome | ! continuous-integration-diana-florez.yml U x
.github > workflows > ! continuous-integration-diana-florez.yml
1  name: Continuous Integration for diana_florez
2
3  on:
4    push:
5      branches:
6        - master
7
8  jobs:
9    build:
10     runs-on: ubuntu-latest
11
12     steps:
13       # Paso 1: Pruebas de SAST (mockeado con un echo)
14       - name: Pruebas de SAST
15         run: echo "Ejecución de pruebas de SAST"
16
17       # Paso 2: Crear archivo de credenciales
18       - name: Crear archivo de credenciales
19         run: |
20           echo "[Credentials]" > credentials.ini
21           echo "username={{ secrets.USERNAME }}" >> credentials.ini
22           echo "password={{ secrets.PASSWORD }}" >> credentials.ini
23
24       # Paso 3: Build docker image
25       - name: Build docker image
26         run: docker build -t devops_ws . --tag devops_ws:$(date +%s)
27

```

Tube inconvenientes con Build docker image y para que corra con el paso 3 me sale error

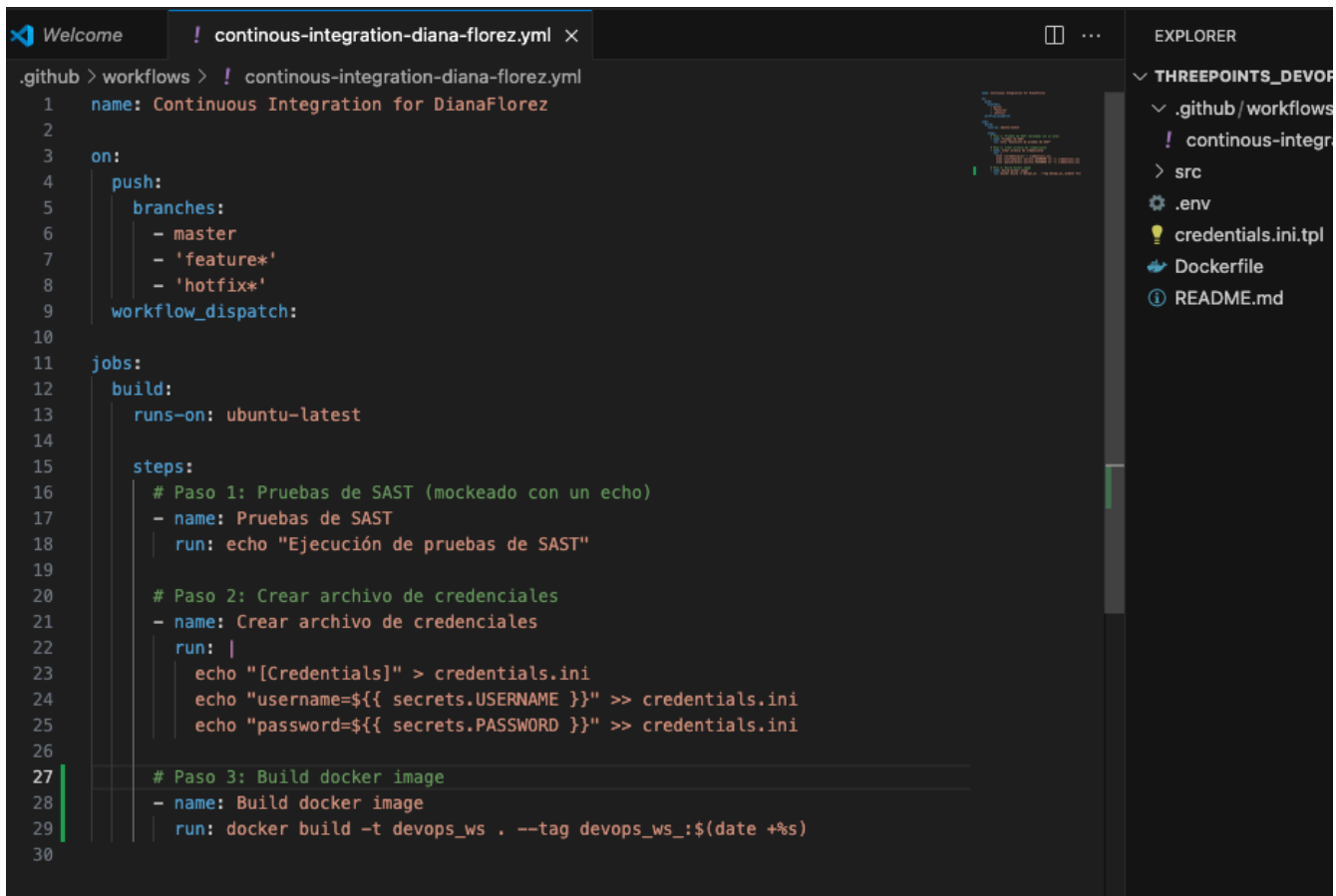


## 2 – Hacer que las acciones se ejecuten, además de al pushear en la rama “master”, en las ramas que empiecen por “feature” y “hotfix (2 puntos)

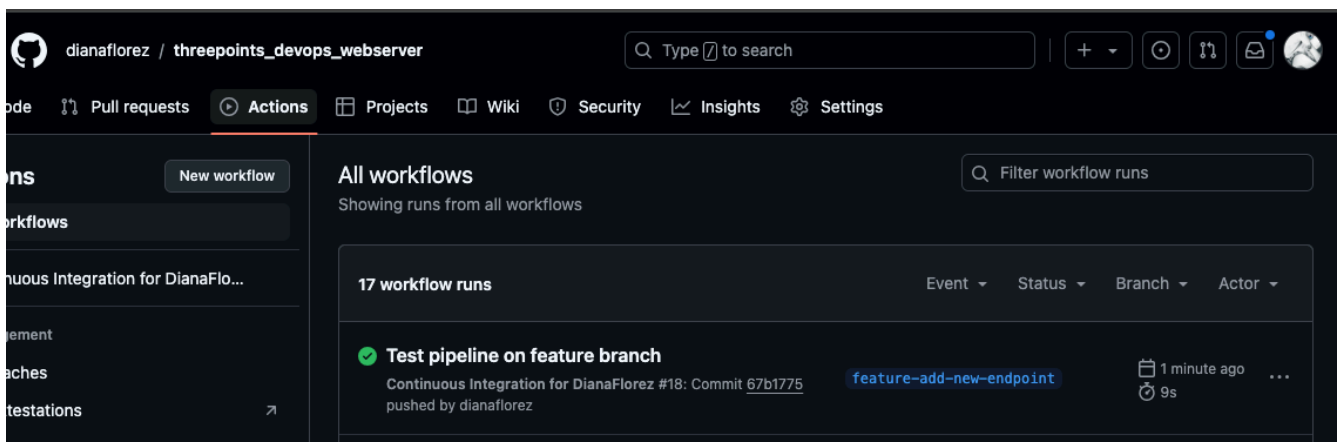
Durante el siguiente punto, basándose en el pipeline creado en el anterior ejercicio, tenemos que modificarlo para que, además de ejecutarse en la rama “master”, se ejecute en aquellas que EMPIECEN por “feature” o por “hotfix”.

Por ejemplo, debería ejecutarse cuando se haga push a la rama “master” o a la rama “feature-addjs-support”, pero no si se hace push a la rama “develop”.

- Código del pipeline que ejecuta esto.
- Captura de la pantalla de la ejecución exitosa en una rama que empiece por “feature”.



```
1 name: Continuous Integration for DianaFlorez
2
3 on:
4   push:
5     branches:
6       - master
7       - 'feature*'
8       - 'hotfix*'
9   workflow_dispatch:
10
11 jobs:
12   build:
13     runs-on: ubuntu-latest
14
15     steps:
16       # Paso 1: Pruebas de SAST (mockeado con un echo)
17       - name: Pruebas de SAST
18         run: echo "Ejecución de pruebas de SAST"
19
20       # Paso 2: Crear archivo de credenciales
21       - name: Crear archivo de credenciales
22         run: |
23           echo "[Credentials]" > credentials.ini
24           echo "username=${{ secrets.USERNAME }}" >> credentials.ini
25           echo "password=${{ secrets.PASSWORD }}" >> credentials.ini
26
27       # Paso 3: Build docker image
28       - name: Build docker image
29         run: docker build -t devops_ws . --tag devops_ws:${date +%s}
```



### 3 – Crear un segundo pipeline para analizar el código en caso de una Pull Request y no permita el mergeo

Vamos a crear para este ejercicio un segundo Workflow, que debe de ejecutarse a la hora de recibir un PULL REQUEST en el repositorio. Esta acción está pensada para impedir el merge request en caso de que el nuevo código vaya a introducir fallos.

En este caso, para simplificar, impediremos el merge request siempre, simulando un análisis no exitoso. Para realizar esto, usaremos steps condicionales para realizar este quality gate de forma artificial. Comprobaremos si existe un archivo llamado "allow\_pull\_request.txt".

Si ese archivo existe, imprimirá por la pantalla "Permitiendo Pull Request". Si no existe, imprimirá "Impidiendo Pull Request" y fallará la ejecución (exit 1).

Este workflow debe de ejecutar las siguientes acciones:

1. Analizar el código. En un workflow real, se ejecutaría un SonarQube u otro analizador de código, pero en este caso sólo habrá que imprimir "Realizando análisis de SAST"
2. Comprobar se cumplen los requisitos. Durante este job, usualmente se comprobaría si los resultados del análisis han superado los requisitos de nuestro quality gate. En este caso, lo que haremos será mirar si existe el archivo "allow\_pull\_request.txt". Si existe, generará una variable booleana llamada "allow\_pull" que valga True. Si no, esta variable valdrá False.
3. Permitir la build si allow\_pull vale True. Este paso deberá mirar la variable generada anteriormente, y si vale True, imprimir por la pantalla "Permitiendo Pull Request".
4. Impedir la build si allow\_pull vale False. Este paso deberá mirar la variable generada en el paso 2, y si vale False, imprimir por la pantalla "Build fallida. Impidiendo Pull Request" y salir con un exit 1.

Así mismo, deberíamos guardar este archivo generado como un nuevo artefacto en el Job. Como entregables de este ejercicio, será necesario:

- Código del Workflow que ejecuta la acción.
- Captura de la pantalla de la ejecución exitosa, osea, con el archivo existente, y que permita el pull request.
- Captura de la pantalla fallida, ósea, sin el archivo, impidiendo el pull request.

```
! continuous-integration-diana-florez.yml | ! pull-request-analysis.yml U x
github > workflows > ! pull-request-analysis.yml
1  name: Pull Request Analysis for DianaFlorez
2
3  on:
4    pull_request:
5      branches:
6        - master
7        - 'feature*'
8        - 'hotfix*'
9
10 jobs:
11   analyze:
12     runs-on: ubuntu-latest
13     steps:
14       # Paso 1: Realizando análisis de SAST
15       - name: Realizando análisis de SAST
16         run: echo "Realizando análisis de SAST"
17
18       # Paso 2: Comprobar si existe el archivo allow_pull_request.txt
19       - name: Comprobar requisitos
20         id: check_file
21         run: |
22           if [ -f allow_pull_request.txt ]; then
23             echo "allow_pull=true" >> $GITHUB_ENV
24           else
25             echo "allow_pull=false" >> $GITHUB_ENV
26           fi
27
28       # Paso 3: Permitir Pull Request si allow_pull es True
29       - name: Permitir Pull Request
30         if: env.allow_pull == 'true'
31         run: echo "Permitiendo Pull Request"
32
33       # Paso 4: Impedir Pull Request si allow_pull es False
34       - name: Impedir Pull Request
35         if: env.allow_pull == 'false'
36         run: |
37           echo "Build fallida. Impidiendo Pull Request"
38           exit 1
39
40       # Guardar el archivo como artefacto (opcional)
41       - name: Subir artefacto
42         if: env.allow_pull == 'false'
43         uses: actions/upload-artifact@v3
44         with:
45           name: resultado_pull_request
```

github.com/dianaflorez/threepoints\_devops\_webserver/actions

dianaflorez / threepoints\_devops\_webserver

Code Pull requests 1 Actions Projects Wiki Security Insights Settings

Workflow run deleted successfully.

### Actions

New workflow

All workflows

Continuous Integration for DianaFlo...

Pull Request Analysis for DianaFlorez

Management

Caches

Attestations

Runners

### All workflows

Showing runs from all workflows

Filter workflow runs

24 workflow runs

	Event	Status	Branch	Actor
🟢 pipeline para analizar el código en caso de una ...	Continuous Integration for DianaFlorez #24: Commit 6e48d8f pushed by dianaflorez	feature-pull-request-check	now	9s
🔴 Feature pull request check	Pull Request Analysis for DianaFlorez #1: Pull request #1 opened by dianaflorez	feature-pull-request-check	4 minutes ago	11s

github.com/dianaflorez/threepoints\_devops\_webserver/actions

☰

dianaflorez / threepoints\_devops\_webserver

🔍 Type / to search

+ ▾

🔄

🔗

📧

👤

<> Code

🔗 Pull requests 1

🎬 Actions

📁 Projects

📖 Wiki

🛡 Security

📈 Insights

⚙ Settings

Actions

New workflow

All workflows

Continuous Integration for DianaFlo...

Pull Request Analysis for DianaFlorez

Management

🗄 Caches

📄 Attestations

All workflows

Showing runs from all workflows

🔍 Filter workflow runs

25 workflow runs

Event ▾ Status ▾ Branch ▾ Actor ▾

✔ Bloquear Pull Request

Continuous Integration for DianaFlorez #25: Commit [6948a64](#) [feature-pull-request-check](#)

📅 1 minute ago

⌚ 9s

...