

## Actividad Clase 7: Mejora continua y herramientas CI/CD

Objetivo:

Aplicar conceptos de integración y entrega continua utilizando GitHub Actions y simular un flujo básico de CI/CD.

### Ejercicio 1: Simulación de un pipeline básico con GitHub Actions

Requisitos previos:

- Tener cuenta en GitHub
- Tener instalado Python 3
- Tener un repositorio nuevo en GitHub

Pasos:

1. Crear un repositorio llamado PruebaCI\_apellido.
2. Clonar el repositorio localmente.

```
MINGW64:/c/reposgit/MSII/PruebaCI
rosal@LENOVORBI MINGW64 /c/reposgit/MSII
$ git clone git@github.com:rinsaurralde/PruebaCI.git
Cloning into 'PruebaCI'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (8/8), done.

,rosal@LENOVORBI MINGW64 /c/reposgit/MSII
$ cd PruebaCI/
rosal@LENOVORBI MINGW64 /c/reposgit/MSII/PruebaCI (main)
$ ls -l
total 1
-rw-r--r-- 1 rosal 197609 33 Aug 12 11:52 README.md

rosal@LENOVORBI MINGW64 /c/reposgit/MSII/PruebaCI (main)
$
```

3. Dentro del repositorio local, crear un archivo app.py con una función simple en Python:

```
# app.py
def suma(a, b):
    return a + b
```

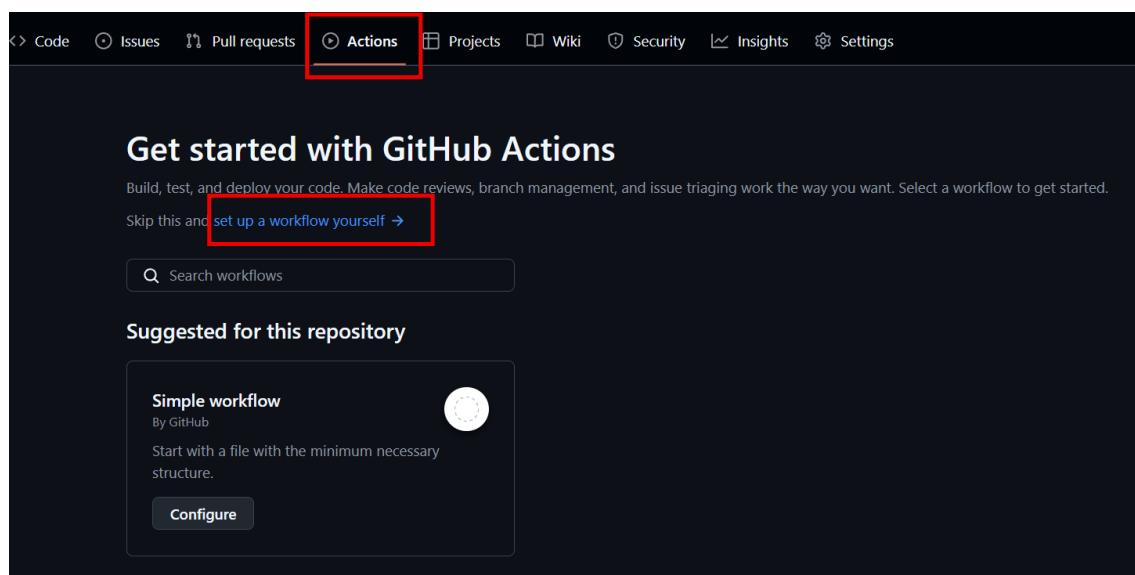
### 4. Crear un archivo de prueba test\_app.py:

```
# test_app.py
from app import suma

def test_suma():
    assert suma(2, 3) == 5
```

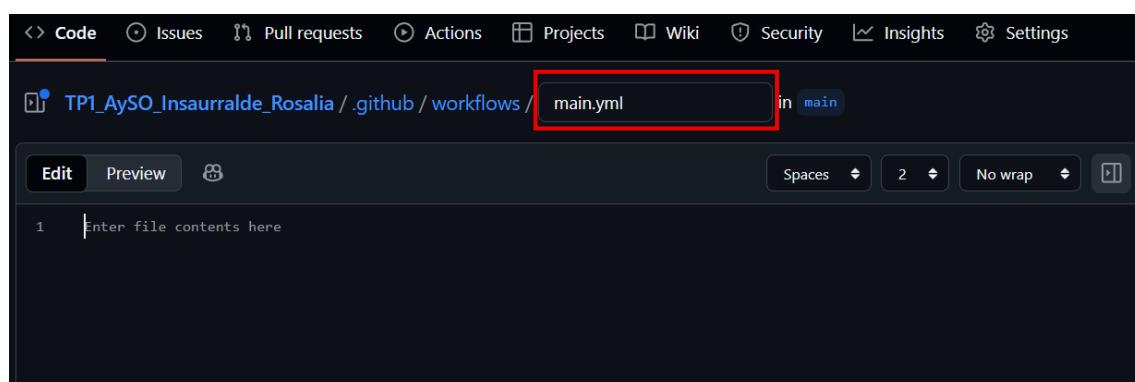
### 5. Crear la estructura del workflow:

En el repositorio remoto, agregar el archivo. Seleccionar “Actions” :



The screenshot shows the GitHub Actions setup interface. At the top, there's a navigation bar with links: Code, Issues, Pull requests, Actions (which is highlighted with a red box), Projects, Wiki, Security, Insights, and Settings. Below the navigation bar, the title "Get started with GitHub Actions" is displayed. A sub-header says "Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started." There's a "Skip this and set up a workflow yourself →" link. A search bar labeled "Search workflows" is present. Under the heading "Suggested for this repository", there's a card for "Simple workflow By GitHub". The card describes it as starting with a file with the minimum necessary structure and includes a "Configure" button. The entire screenshot has a dark theme.

.github/workflows/python-ci.yml



The screenshot shows the GitHub workflow editor. The top navigation bar is identical to the one in the previous screenshot. Below it, the URL "TP1\_AySO\_Insaurralde\_Rosalia / .github / workflows / main.yml" is shown, with "main.yml" highlighted by a red box. The editor interface includes tabs for "Edit", "Preview", and "Raw". On the right, there are buttons for "Spaces", "2", "No wrap", and a copy icon. The main area is a code editor with the following content:

```
1 Enter file contents here
```

### 6. Copiar el siguiente contenido en python-ci.yml:

```
name: Python CI
```

```
on: [push]
```

```
jobs:
```

```
  build:
```

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

```
steps:
```

- uses: actions/checkout@v3

- name: Instalar Python

- uses: actions/setup-python@v4

- with:

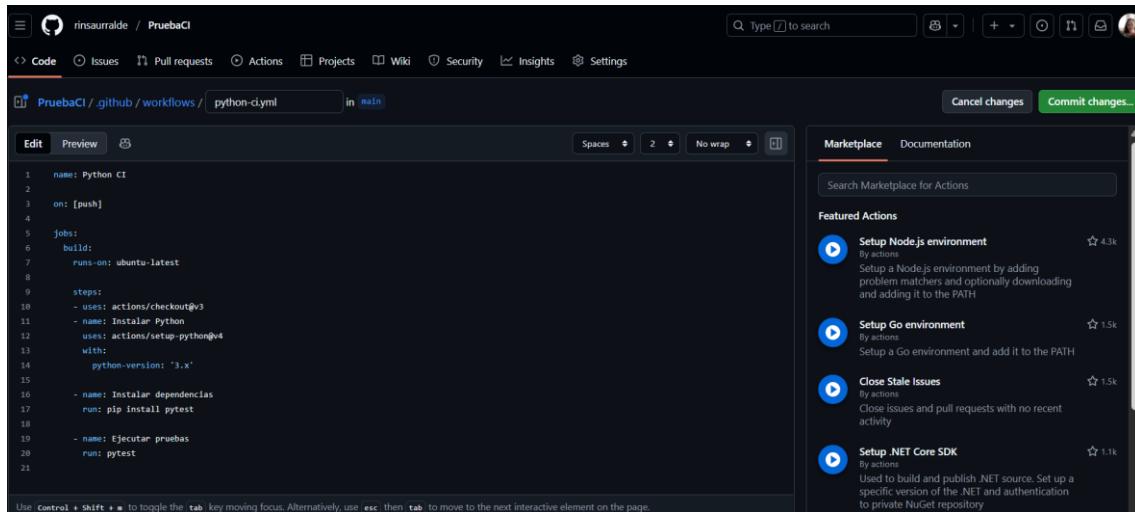
- python-version: '3.x'

- name: Instalar dependencias

- run: pip install pytest

- name: Ejecutar pruebas

- run: pytest

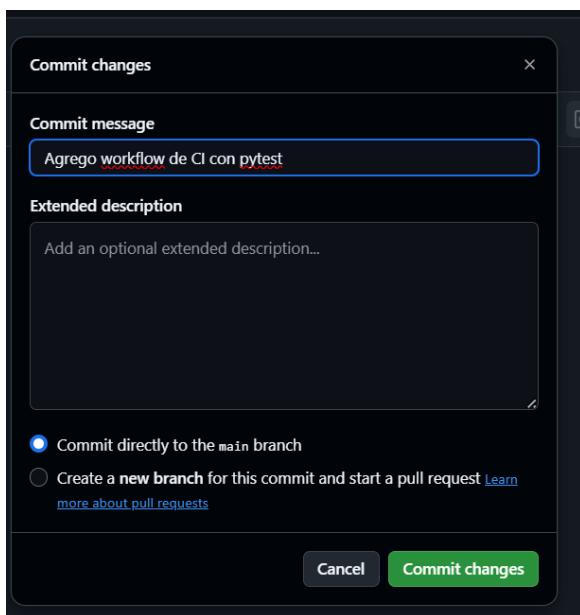


```

1 name: Python CI
2
3 on: [push]
4
5 jobs:
6   build:
7     runs-on: ubuntu-latest
8
9   steps:
10    - uses: actions/checkout@v3
11    - name: Instalar Python
12      uses: actions/setup-python@v4
13      with:
14        python-version: '3.x'
15
16    - name: Instalar dependencias
17      run: pip install pytest
18
19    - name: Ejecutar pruebas
20      run: pytest

```

The screenshot shows the GitHub Actions workflow editor for a repository named 'PruebaCI'. The workflow file is named 'python-ci.yml' and is set to trigger on pushes ('on: [push]'). It contains one job named 'build' which runs on an Ubuntu latest runner. The job consists of several steps: 1) Checkouts the repository ('actions/checkout@v3'), 2) Sets up Python using the 'actions/setup-python@v4' action with version '3.x', 3) Installs dependencies with 'pip install pytest', and 4) Runs tests with 'pytest'. The right side of the interface shows the GitHub Marketplace with featured actions like 'Setup Node.js environment', 'Setup Go environment', 'Close Stale Issues', and 'Setup .NET Core SDK'.



## 7. Hacer commit y push. Ir a la pestaña Actions del repositorio y observar la ejecución automática del pipeline.

```

rosal@LENOVORBI MINGW64 /c/repoGit/MSII/PruebaCI (main)
$ ls -l
total 3
-rw-r--r-- 1 rosal 197609 33 Aug 12 11:52 README.md
-rw-r--r-- 1 rosal 197609 33 Aug 12 11:58 app.py
-rw-r--r-- 1 rosal 197609 67 Aug 12 12:00 test_app.py

rosal@LENOVORBI MINGW64 /c/repoGit/MSII/PruebaCI (main)
$ git add .
warning: in the working copy of 'app.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'test_app.py', LF will be replaced by CRLF the next time Git touches it

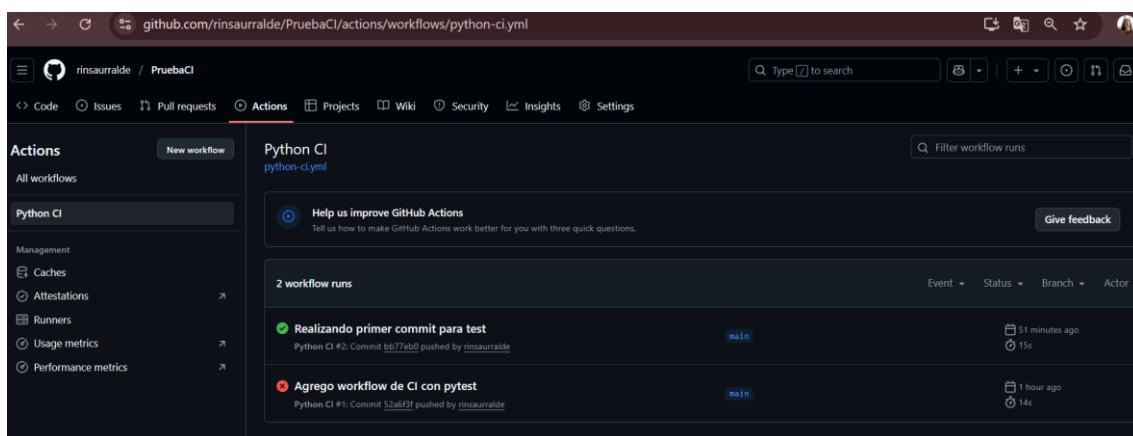
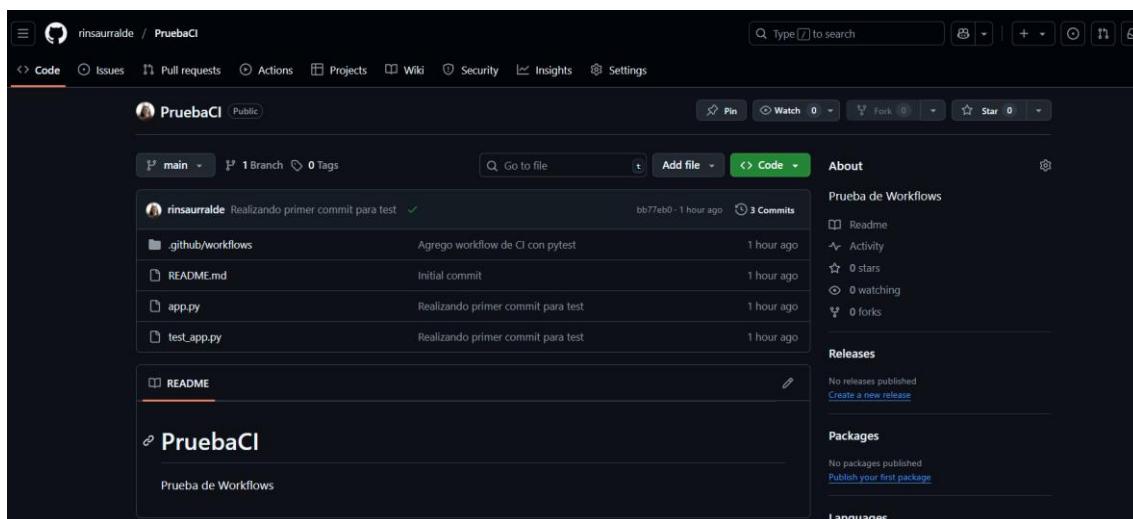
rosal@LENOVORBI MINGW64 /c/repoGit/MSII/PruebaCI (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   app.py
    new file:   test_app.py

rosal@LENOVORBI MINGW64 /c/repoGit/MSII/PruebaCI (main)
$ git commit -m "Realizando primer commit para test"
[main bb77eb0] Realizando primer commit para test
 2 files changed, 7 insertions(+)
 create mode 100644 app.py
 create mode 100644 test_app.py

rosal@LENOVORBI MINGW64 /c/repoGit/MSII/PruebaCI (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 452 bytes | 226.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:rinsaurralde/PruebaCI.git
  52a6f3f..bb77eb0  main -> main

rosal@LENOVORBI MINGW64 /c/repoGit/MSII/PruebaCI (main)
$ 
```



## Ejercicio 2: Análisis de mejora continua

1. Analizar qué pasaría si la función tuviera un error.
2. Modificar la función suma() para que devuelva incorrectamente a - b y observar cómo falla el pipeline.
3. Corregir el error y volver a hacer push.