



**UNIVERSIDAD  
CATÓLICA  
DE CÓRDOBA**  
JESUITAS

**Ingeniería de Software 3  
TRABAJO PRÁCTICO N°8**

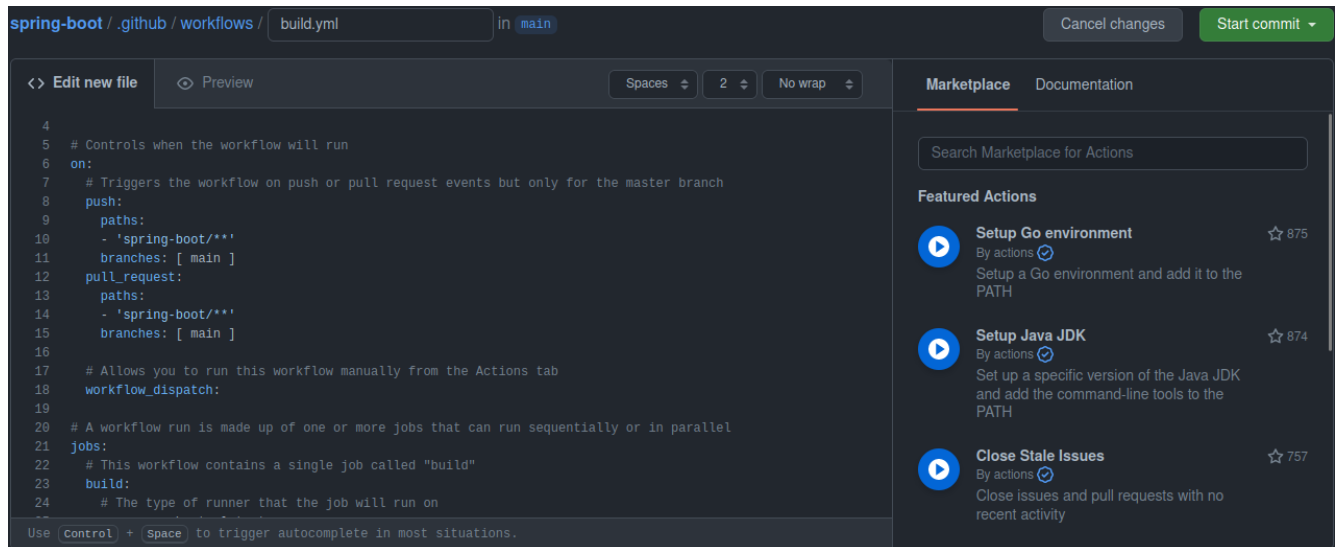
**Profesor:** Ing. Fernando Bono

**Alumno:** Menel Angelo (1804789)

## **EJERCICIO 1:**

- Github Actions:
  - Pros:
    - GitHub actions are just consecutive docker runs. Very easy to reason about and debug. Reproducing the build environment for [container-based Travis is possible](#), but more difficult. On GitHub actions it's just a docker build docker run away.
    - The individual *actions* in a *workflow* are isolated by default. You can use a completely different computing environment for, say, compilation and testing. Other traditional CI would run all "stages" (~ actions) in the same computing environment. Again, GitHub actions are much easier to reason about and debug.
    - The main.workflow spec (a subset of the HCL and really just a directed acyclic graph) is [open source](#). The whole thing is a pretty thin wrapper around Docker anyway, so platform lock-in is arguably minimal.
    - You have ready access to the GitHub API with (somewhat limited) authentication out of the box.
    - There *might* be a vibrant community (marketplace?) where people can share actions. For example, I'm reusing deploy actions build by different people in different ecosystems.
  - Cons:
    - No native caching. You get *image* and *layer caching*, but nothing else. To build artifacts, you have to roll your own cache (via AWS, Azure, etc. ...), which can be a lot of work.
    - **No support for pull requests from forks.** It's again a bit complicated, and understandable from a security standpoint, but it's currently not possible to run actions a) against the secrets of the receiving repo of a fork PR (base), and/or b) against the *would-be* merge result of a fork PR (that's what travis does). For a workflow that involves forks, that makes GitHub actions largely unusable as CI/CD tools.
    - The quality and breadth of published GitHub actions (at least on the marketplace) is still pretty low / limited.

## EJERCICIO 2:

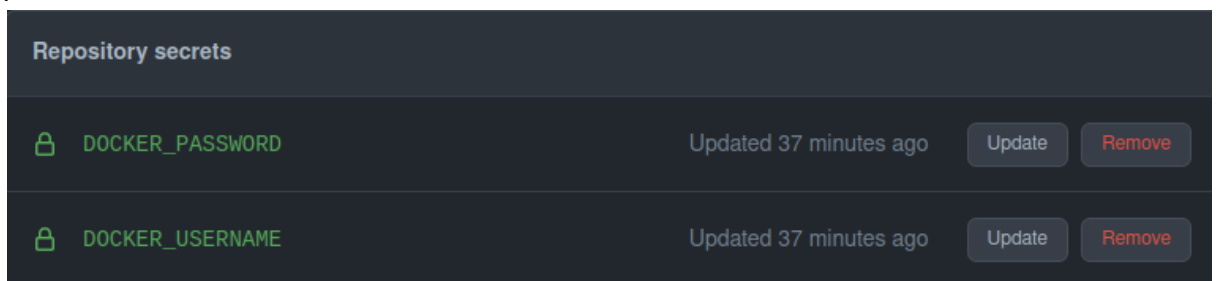


The screenshot shows the GitHub Actions workflow editor for the file `build.yml` in the `spring-boot` repository. The workflow is configured to run on the `main` branch. It includes a `push` trigger and a `pull_request` trigger, both limited to the `main` branch. The workflow consists of a single job named `build` that runs on the `ubuntu-latest` runner. The job includes a `workflow_dispatch` trigger and a `jobs` section with a single job named `build`. The job is configured to run on the `ubuntu-latest` runner. The workflow is currently in a draft state, with buttons for `Cancel changes` and `Start commit` visible at the top right. The right sidebar shows the `Marketplace` tab with a search bar and a list of featured actions: `Setup Go environment` (875 stars), `Setup Java JDK` (874 stars), and `Close Stale Issues` (757 stars).

```
4
5 # Controls when the workflow will run
6 on:
7   # Triggers the workflow on push or pull request events but only for the master branch
8   push:
9     paths:
10      - 'spring-boot/**'
11     branches: [ main ]
12   pull_request:
13     paths:
14      - 'spring-boot/**'
15     branches: [ main ]
16
17 # Allows you to run this workflow manually from the Actions tab
18 workflow_dispatch:
19
20 # A workflow run is made up of one or more jobs that can run sequentially or in parallel
21 jobs:
22   # This workflow contains a single job called "build"
23   build:
24     # The type of runner that the job will run on
```

## EJERCICIO 3:

- El ejercicio planteado está en Github Action [build.yml](#).
- A diferencia de la referencia que nos dejó en el TP8, borre de environment el **REGISTRY** y de **IMAGE\_NAME** borre el `${{ github.repository }}` para poder modificar el nombre del username de docker hub.
- En **Settings>Secrets>Actions>New repository secrets** configurar el username y password de Docker Hub.



- Estos secrets.DOCKER\_USERNAME y secrets.DOCKER\_PASSWORD los uso para loguearme en docker hub.