**Trabajo Práctico 8 - Herramientas de construcción de software en la nube**

1- Objetivos de Aprendizaje

- Adquirir conocimientos acerca de las herramientas de integración continua en la nube.

- Configurar este tipo de herramientas.

- Implementar procesos simples de construcción automatizada.

2- Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad Nº: 3 (Libro Continuous Delivery: Cap 3)

3- Consignas a desarrollar en el trabajo práctico:

- Para una mejor evaluación del trabajo práctico, incluir capturas de pantalla de los pasos donde considere necesario.

- En este caso existen varias herramientas con funcionalidades similares en la nube:

- [GitHub Actions](https://github.com/actions)

- [CircleCI](https://circleci.com/)

- [Travis CI](https://travis-ci.com/)

- [Codefresh](https://codefresh.io/)

- [Gitlab](https://about.gitlab.com/) - Trial de 30 días

**4- Desarrollo:**

1- Pros y Contras

- Listar los pros y contras de este tipo de herramientas

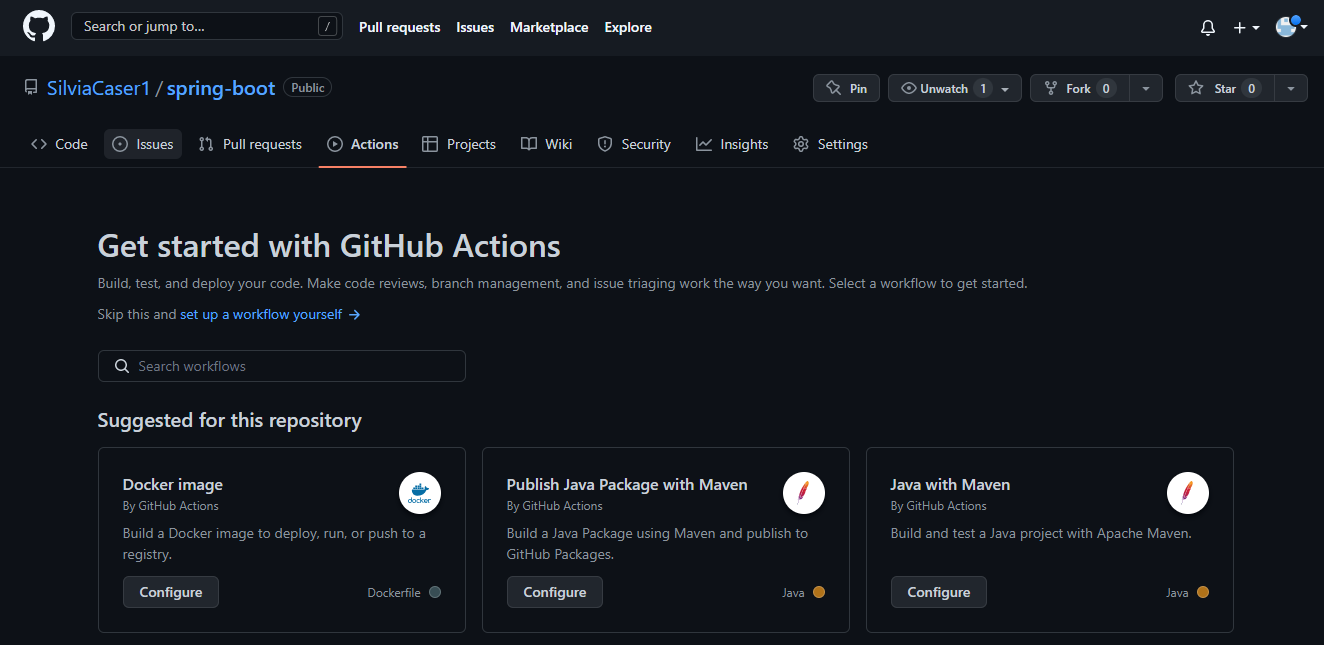
- Sacar conclusiones

|  |  |  |  |
| --- | --- | --- | --- |
| Tool | Pros | Contras | Conclusiones |
| GitHub Actions | * Servicio gratuito, aunque también tiene servicios de pago. * Búsqueda muy rápida en las estructuras de los repos. * Amplia comunidad y fácil encontrar ayuda. * Ofrece prácticas herramientas de cooperación y buena integración con Git. * Fácil integrar con otros servicios de terceros. * Trabaja también con TFS, HG y SVN. | * No es absolutamente abierto. * Tiene limitaciones de espacio, ya que no puedes exceder de 100MB en un solo archivo, mientras que los repositorios están limitados a 1GB en la versión gratis. | Realmente resulta interesante tratar las propias Actions de GitHub como herramienta de integración continua, tal y como se ha podido observar en distintos proyectos durante la investigación, obteniendo así una integración completa en el proyecto. |
| CicleCI | * Configuración inicial automática con GitHub, detectando el lenguaje y creando una configuración básica para el mismo. * Configuración sencilla mediante YAML. * Soporte SSH, para acceder a la máquina virtual. * Interfaz de usuario sencilla y intuitiva. * Soporte Docker. * Paralelización de tests. | * Opciones limitadas en el plan gratis. * Precio alto, aunque en nuestro caso no es realmente una desventaja. * Necesidad de conocer YAML. * Existen otras opciones con mejor documentación. | Es la herramienta más popular en la actualidad, y la que en un principio presenta mejores características de cara al proyecto. Además se trata de un competidor directo de Travis, por lo que resultará interesante analizarlo. |
| Travis CI |  |  |  |
| Codefresh |  |  |  |
| GitLab | * Plan gratuito y sin limitaciones, aunque tiene planes de pago. * Es de licencia de código abierto. * Permite el autohospedaje en cualquier plan. * Está muy bien integrado con Git. | * Su interfaz puede ser algo más lenta con respecto a la competencia. * Existen algunos problemas habituales con los repositorios. |  |

**2- Configurando GitHub Actions**

- Repetir el ejercicio 6 del trabajo práctico [trabajo práctico 7] para el proyecto \*\*spring-boot\*\*, pero utilizando GitHub Actions.

- En GitHub, en el repositorio donde se encuentra la aplicación \*\*spring-boot\*\*, ir a la opción \*\*Actions\*\* y crear un nuevo `workflow`.



- El nombre de archivo puede ser build.xml y tendrá un contenido similar al siguiente (el path donde se encuentra el código puede ser diferente):

# This is a basic workflow to help you get started with Actions

name: CI

# Controls when the workflow will run

on:

# Triggers the workflow on push or pull request events but only for the master branch

push:

paths:

- 'proyectos/spring-boot/\*\*'

branches: [ master ]

pull\_request:

paths:

- 'proyectos/spring-boot/\*\*'

branches: [ master ]

# Allows you to run this workflow manually from the Actions tab

workflow\_dispatch:

# A workflow run is made up of one or more jobs that can run sequentially or in parallel

jobs:

# This workflow contains a single job called "build"

build:

# The type of runner that the job will run on

runs-on: ubuntu-latest

# Steps represent a sequence of tasks that will be executed as part of the job

steps:

# Checks-out your repository under $GITHUB\_WORKSPACE, so your job can access it

- uses: actions/checkout@v2

# Install Java JDK with maven

- name: Set up JDK 8

uses: actions/setup-java@v2

with:

java-version: '8'

distribution: 'adopt'

cache: maven

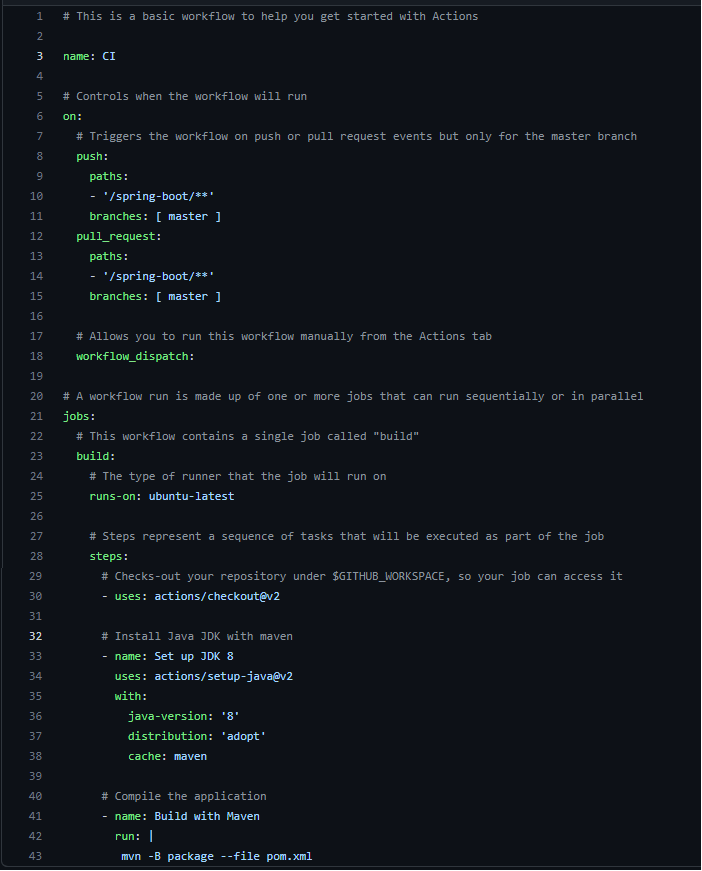
# Compile the application

- name: Build with Maven

run: |

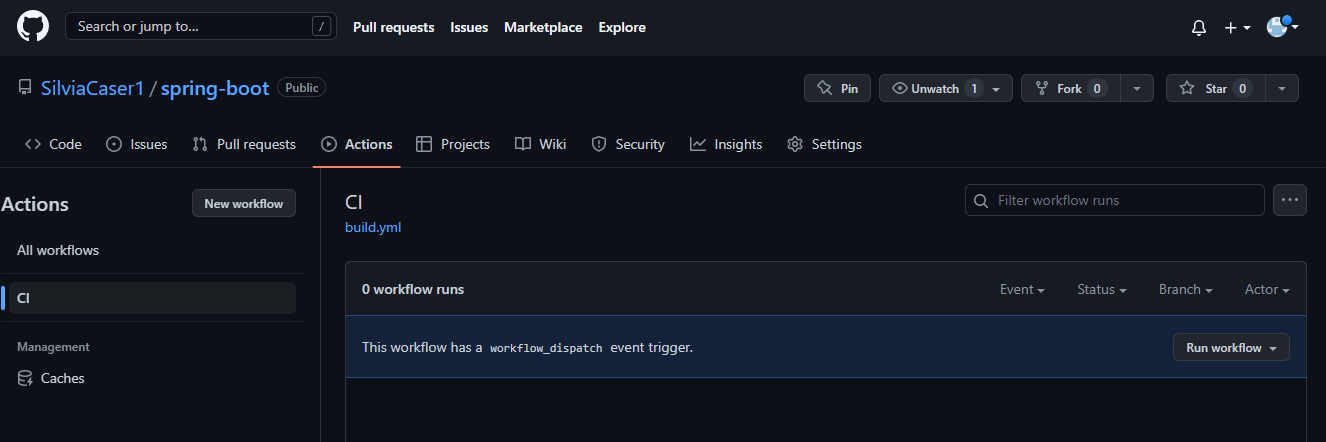
cd proyectos/spring-boot/

mvn -B package --file pom.xml

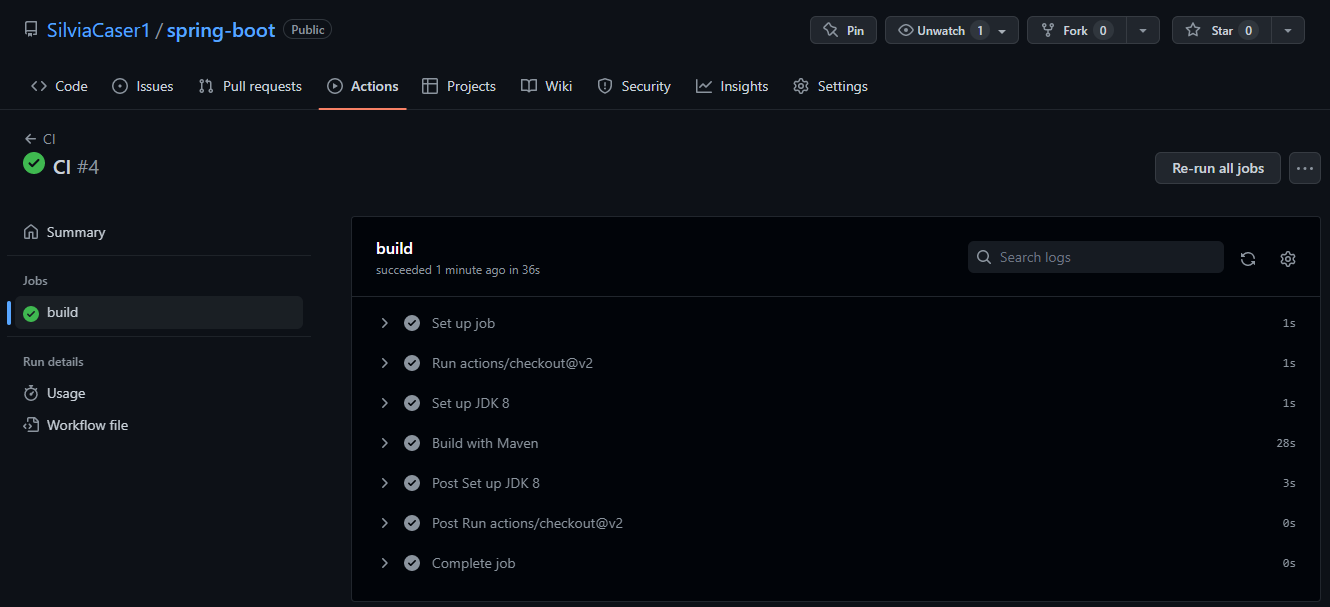


- Guardar el archivo (hacemos commit directamente en GitHub por ejemplo) y ejecutamos manualmente el pipeline.

Se hace click en Start commits



Se corre el build manualmente



- Explicar que realiza el pipeline anterior.

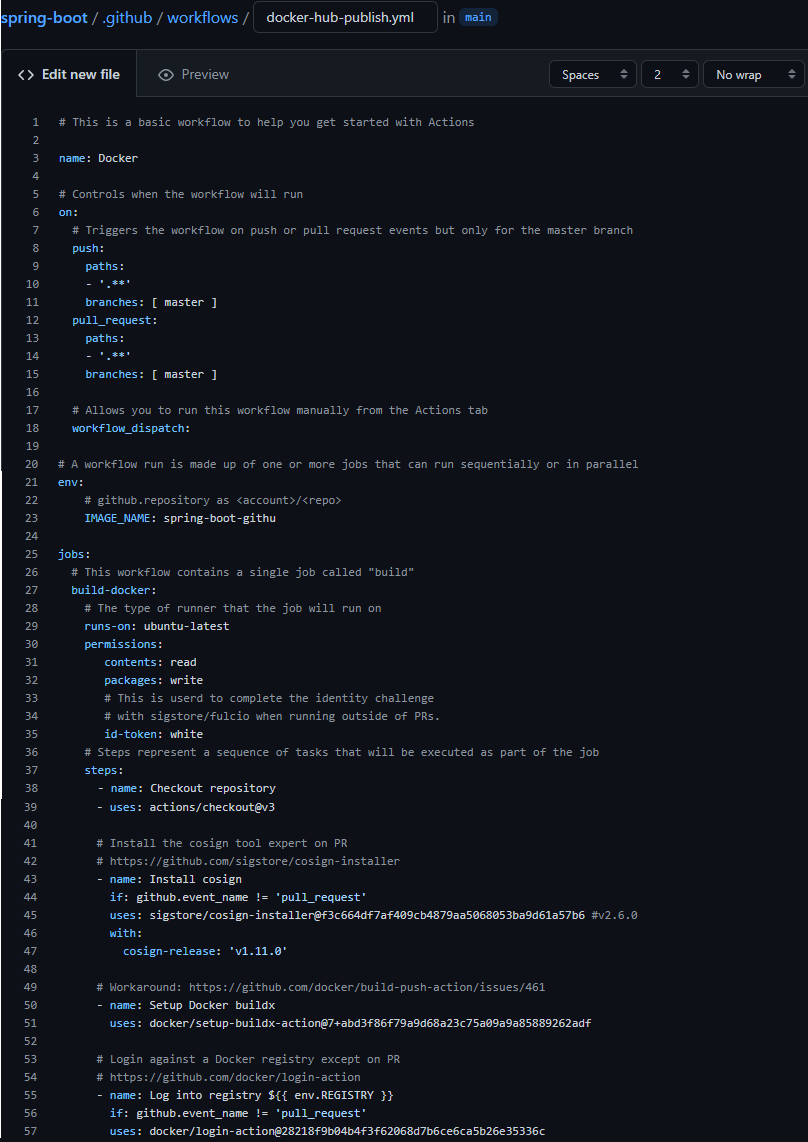
Se define la ruta en Github del repo spring-boot. Lo siguiente, es hace uso de la actions/checkout@2 que permite acceder al repo de spring-boot continuando con la instalación de Java JDK 8 con Maven. Termina compilando y corriendo una app con Maven.

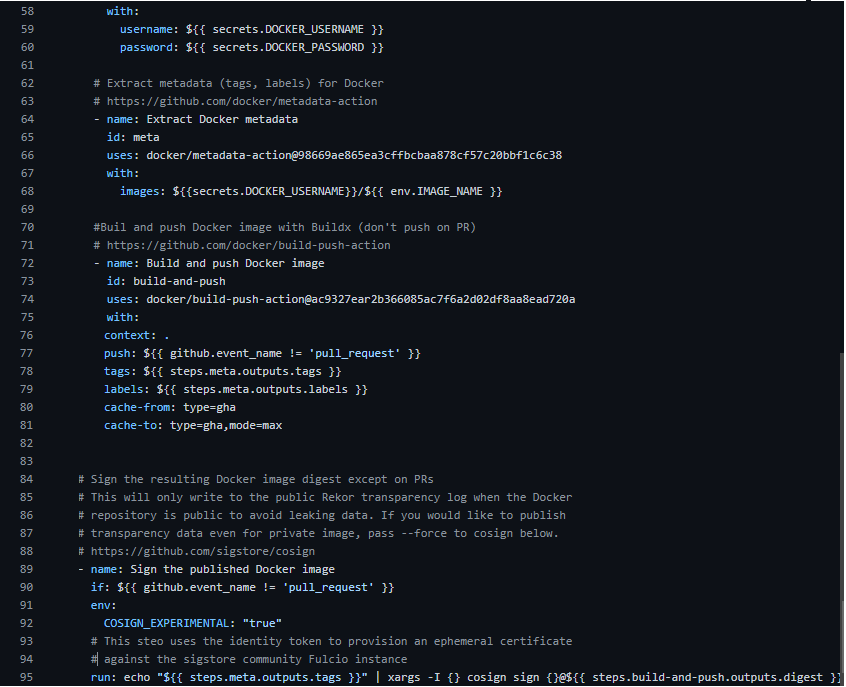
**3- Utilizando nuestros proyectos con Docker**

- Repetir el ejercicio 7 del trabajo práctico [trabajo práctico 7](07-servidor-build.md), pero utilizando GitHub Actions.

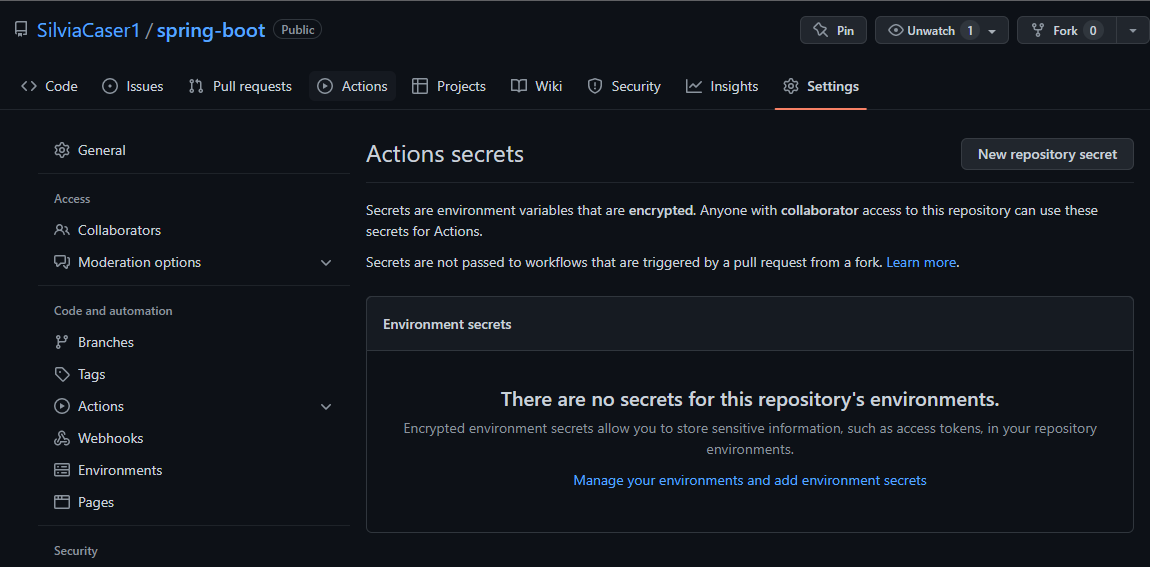
- Generar `secretos` y los `pasos` necesarios para subir la imagen a Docker Hub. [Referencia](https://github.com/actions/starter-workflows/blob/main/ci/docker-publish.yml)

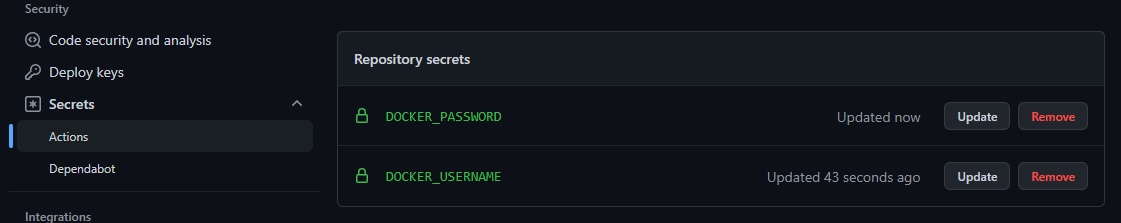
* Crear el archivo docker-hub-publish.yml





* Se generan los secretos con mis credenciales de docker





* Correr el workflow

Pendiente

* Comprobar que se subió a Docker Hub

Pendiente

**4- Opcional: Configurando CircleCI**

- De manera similar al ejercicio 2, configurar un build job para el mismo proyecto, pero utilizando CircleCI

- Para capturar artefactos, utilizar esta referencia: https://circleci.com/docs/2.0/artifacts/

- Como resultado de este ejercicio, subir el config.yml a la carpeta \*\*spring-boot\*\*

5- Opcional: Configurando TravisCI

- Configurar el mismo proyecto, pero para TravisCI. No es necesario publicar los artefactos porque TravisCI no dispone de esta funcionalidad.

- Como resultado de este ejercicio subir el archivo .travis.yml a la carpeta \*\*spring-boot\*\*

6- Opcional: Configurando Codefresh

- Configurar el mismo proyecto, pero para Codefresh.

- Como resultado de este ejercicio subir el archivo codefresh.yml a la carpeta \*\*spring-boot\*\*

7- Opcional: Configurando Gitlab

- Configurar el mismo proyecto, pero para Gitlab.

- Como resultado de este ejercicio subir el archivo .gitlab-ci.yml a la carpeta \*\*spring-boot\*\*