Trabajo ASW Seminario: GitHub Actions

Gabriel García Martínez - UO289097
Daniel Andina Pailos - UO287791
Ángel Moltó Álvarez - UO287747

Continuous Integration

La integración continua se basa en la automatización de las partes del proyecto, lo que implica que cualquiera puede medir la calidad del proyecto. Esto significa que cada vez que se realice algún push con código nuevo o con código cambiado se ejecutan procesos que hacen cosas como ejecutar pruebas unitarias, ejecutar un linter, que es una herramienta de programación que ayuda a analizar el código fuente en busca de posibles errores, problemas de estilo u otros problemas. Además, se menciona en el pódcast, que lo más importante para él, de este proceso, es ver si el conjunto de pruebas se ajusta a la cantidad de código que hay en el proyecto, ya que, si esto está bien, significa que es más fácil cambiar el código y saber que no se están rompiendo cosas.

Continuous Development

El desarrollo continuo es el paso siguiente a la integración continua. Esto es que una vez que el código está bien y está correctamente integrado, este código se puede llevar automáticamente a producción de una manera sencilla.

Github actions

Github actions tiene alrededor de 5 años y es la respuesta de GitHub al CI/CD. Permite definir flujos de trabajo dentro del propio repositorio y que, en respuesta a diversos eventos, se ejecutan en un contenedor los pasos de los procesos.
Se trata de una arquitectura basada en eventos. Dentro de cada repositorio, Github ha definido un directorio .github que cualquiera puede crear. En este directorio, se almacenan los diferentes YAML o .yml, que son formatos de serialización para configurar los workflows o flujos de trabajo. Permite ejecutar flujos de trabajo en momentos específicos programados (cron jobs), mediante un botón en el apartado actions de la página o incluso mediante solicitudes HTTP con el uso de webhook.

Dependabot

Un ejemplo de este tipo de archivos es dependabot, que es del propio GitHub. Dependabot realiza diferentes actuaciones. Una de las cosas que hace dependabot, es comprobar todas las dependencias de una aplicación y que estas no tengan problemas de seguridad conocidos. Además, en algunos casos, el propio dependabot se encarga de solicitar una pull-request con la solución al problema. AQUÍ EJEMPLO IPS.

Workflows

Un workflow en GitHub Actions es la implementación de un Action. Está formado por tareas automatizadas que responden a eventos como commits, pull requests, etiquetado de versiones, …

Los workflows se definen en ficheros .yml y se almacenan en .github/workflows. Parámetros:

- **name**, **on**, **jobs**, **steps, env**.

Actions en un fork | Secretos

GitHub Actions permite ejecutar flujos de trabajo en repositorios bifurcados, lo que proporciona a los colaboradores la capacidad de ejecutar automatizaciones en su propio fork antes de enviar una solicitud de extracción (pull request) al repositorio original. Esto es útil para realizar pruebas automatizadas, asegurarse de que los cambios no rompan nada y verificar la integridad del código antes de solicitar que los cambios se incorporen al repositorio principal.

Configuración de seguridad
Límite en el uso de recursos
Límite en el acceso a secretos

Contenedores | Sabores | Escalabilidad

Los contenedores en GitHub Actions son entornos de ejecución aislados que se utilizan para ejecutar tus acciones. Estos contenedores pueden ser personalizados para satisfacer las necesidades específicas del flujo de trabajo.

Características:

- **Reutilizables y portables**: Ejecutar workflows en cualquier entorno.
- **Escalabilidad**: Gran escalado. Automático frente a volúmenes grandes de trabajo.
- **Configurables**: Adaptar el entorno de ejecución a las necesidades.
- **Ejecución en paralelo**: Mejorar velocidad y rendimiento de los flujos de trabajo.

GitHub Actions proporciona varios tipos de contenedores preconfigurados, también conocidos como "sabores de contenedores".

Ubuntu, **Windows**, **macOS.**

Otros contenedores personalizados: Además de los sabores de contenedores proporcionados por GitHub, también puedes definir tus propios contenedores personalizados. Esto te permite especificar un entorno de ejecución personalizado con las herramientas, bibliotecas y configuraciones específicas que necesitas para tus acciones.

Git Scraping

Git-Scraping es una técnica poco convencional que el invitado Dave Cross menciona, su uso combina la potencia del control de versiones Git con el scraping web para la extracción y seguimiento de datos en línea.

El concepto de **Scraping Web** refiere a la práctica de extraer información de páginas web de manera automatizada, generalmente utilizando scripts o herramientas especializadas. Y por otro lado, como ya sabemos, los **actions** se pueden usar para establecer workflows que se ejecuten cada cierto tiempo

De esta manera, podemos hacer uso del servicio de contenedores de Github para definir una tarea cron, en la que si además de configurar el action para ejecutar un script de scraping, subimos el resultado de este a un repositorio, estaremos creando un repositorio de versiones para cada web que analicemos.

Artefactos

Los artefactos son los datos producto de un workflow, un archivo o recopilación de archivos producidos durante una ejecución de flujo de trabajo.

Estos son algunos de los artefactos comunes que se pueden usar:

- Archivos de registro. (LOGS)
- Resultados de prueba.
- Archivos binarios.
- Resultados de la cobertura del código

EJEMPLO DE USO REAL

Los jobs que dependen de los artefactos de un trabajo anterior deben esperar que el trabajo dependiente se complete exitosamente. Este flujo de trabajo usa la palabra clave needs para asegurarse de que job_1, job_2 y job_3 se ejecutan secuencialmente. Por ejemplo, job_2 necesita job_1 mediante la sintaxis needs: job_1.

El job 1:

Realiza un cálculo matemático y guarda el resultado en un archivo de texto denominado math-homework.txt.
Usa la acción upload-artifact para cargar el archivo math-homework.txt con el nombre del artefacto homework_pre.

El job 2:

Descarga el artefacto homework_pre cargado en el trabajo anterior. De manera predeterminada, la acción download-artifact descarga artefactos en el directorio del área de trabajo en la que se ejecuta el paso. Puedes usar el parámetro de entrada path para especificar un directorio de descarga diferente.
Lee el valor en el archivo math-homework.txt, realiza un cálculo matemático y guarda el resultado en math-homework.txt de nuevo, sobrescribiendo su contenido.
Carga el archivo math-homework.txt. A medida que los artefactos se consideran inmutables en v4, el artefacto se pasa a una entrada diferente, homework_final, como un nombre.

El job 3:
Descarga el artefacto artefacto homework.
Imprime el resultado de la ecuación matemática en el registro.

GitHub Actions Importer

Esto es una herramienta que sirve para planificar y migrar de forma automática flujos de trabajo de CI/CD a github actions, siempre que la plataforma en la que está implementado sea compatible con este sistema.

Algunos de los sistemas soportados son:

- Jenkins
- Azure DevOps
- Travis CI

Y el funcionamiento es tan sencillo como, teniendo un entorno contenerizado donde se utilice alguna de estas plataformas para realizar tareas de ci/cd, se instala el plugin de github actions importer y github cli y ya se puede empezar a usar, por ejemplo con el comando audit que nos reporta en un fichero markdown cual va a ser la complejidad de la migración entre otras cosas, además de una primera versión del fichero jobs-from-jenkins.yml, que posiblemente tenga dependencias todavía de los procesos jenkins debido a que no haya podido trasvasar los jobs en su totalidad.