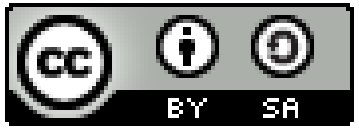
Diseño de flujos de trabajo para la Integración Continua

Asignatura: Integración Continua en el Desarrollo Ágil

Máster Universitario en Desarrollo Ágil de Software para la Web

José Ramón Hilera González



Contenido

1. Concepto de workflow o pipeline
2. Introducción a GitHub Actions
3. Definición de workflows
4. GitHub Actions Runner
5. Conclusiones

1. Concepto de workflow o pipeline

- El flujo de trabajo (workflow según GitHub, o pipeline según GitLab) de un proyecto de desarrollo es la secuencia de pasos automatizados establecidos para conseguir obtener el producto software aplicando integración, entrega y/o despliegue continuo.
- Un servidor de integración continua se encarga de ordenar la ejecución de los pasos del workflow del repositorio del proyecto cada vez que se sube un cambio (commit) al repositorio de código compartido.

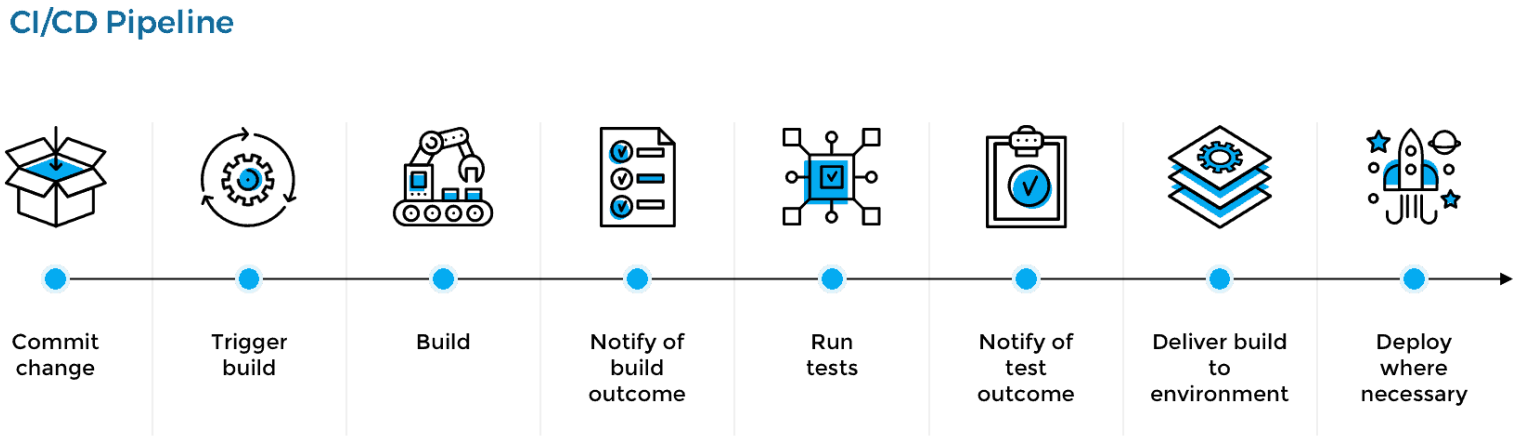


Imagen: TJ Simmons

1. Concepto de workflow o pipeline Ejecución

- Cuando se ejecuta un workflow, se van ejecutando en secuencia los pasos que lo componen.
  - Por ejemplo, la construcción o compilación (build)
- Si alguno de los pasos falla, se aborta la ejecución del workflow y se avisa a los miembros del proyecto
  - Por ejemplo, por email

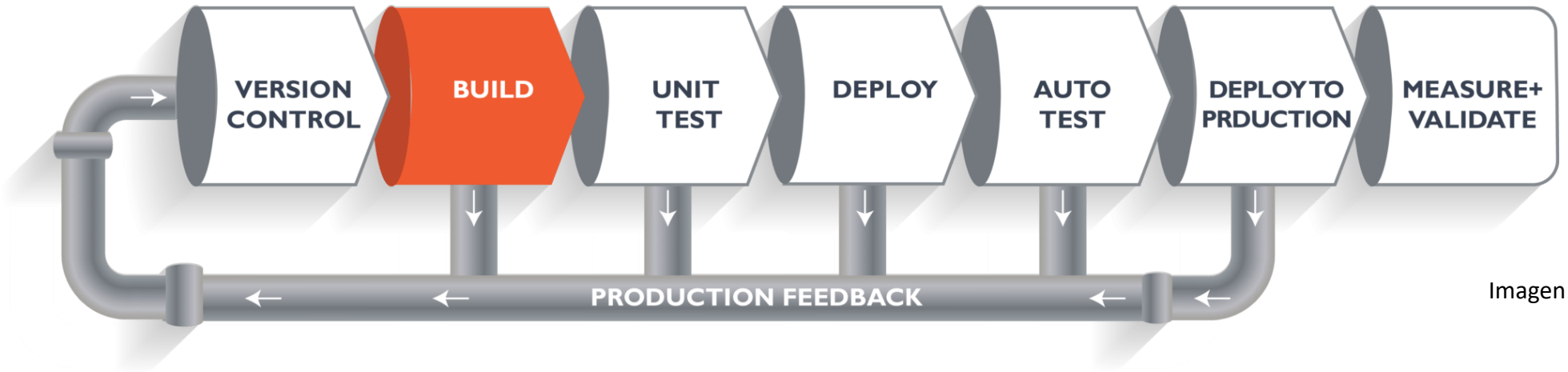


Imagen: Samarpit Tuli

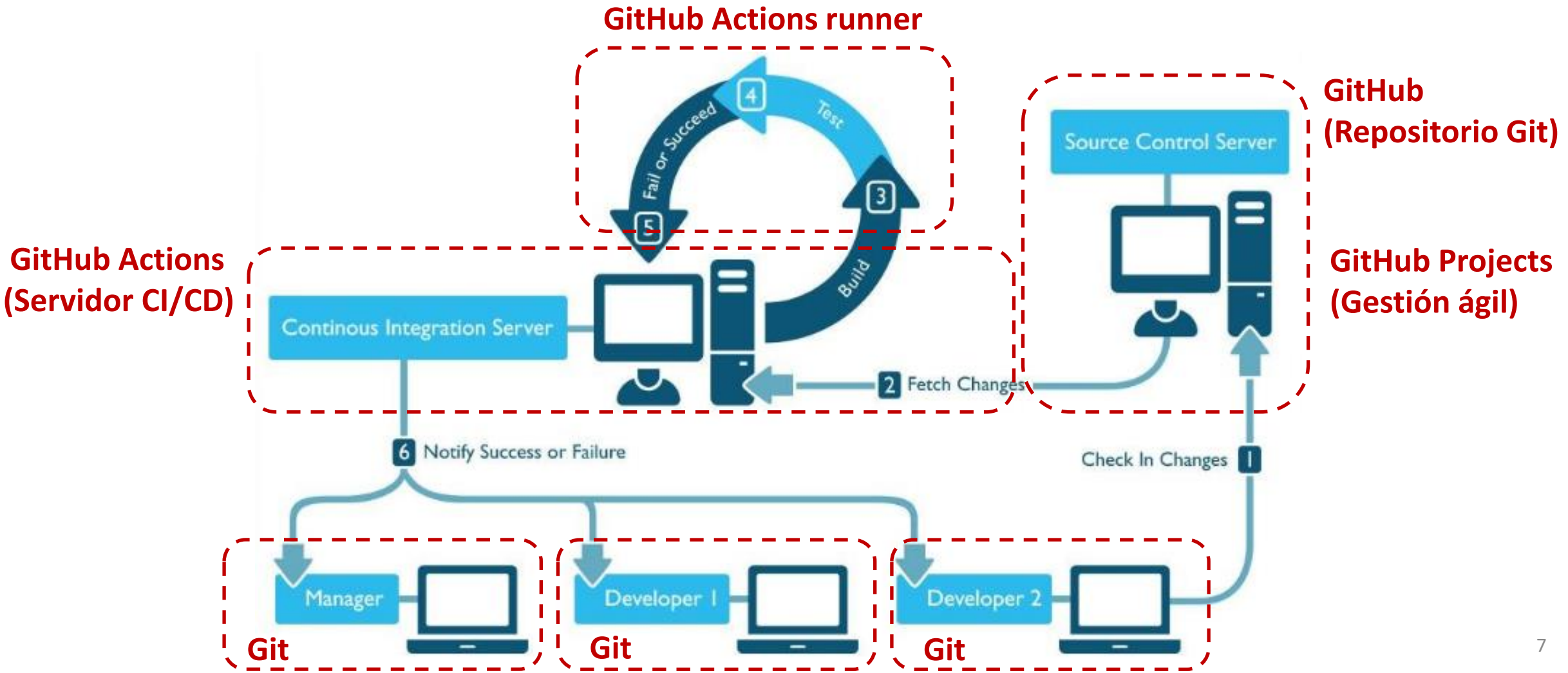
2. Introducción a GitHub Actions

- 2.1 Introducción
- 2.2 Contexto para integración continua
- 2.3 Registrarse para usar github.com
- 2.4 Repositorios
- 2.5 Creación de un repositorio
- 2.6 Comandos básicos de Git
- 2.7 Creación de un proyecto asociado a un repositorio
- 2.8 Gestión ágil de un proyecto (Scrum)
- 2.9 Servidor de Integración Continua (CI/CD)

2.1 Introducción

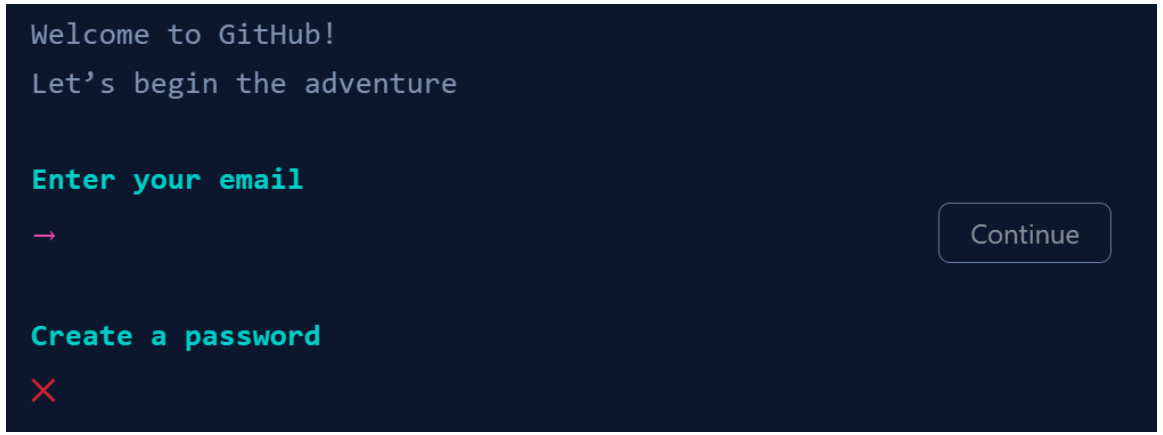
- GitHub es una aplicación web en github.com que incluye
  - Repositorio Git remoto
  - Gestor de proyectos ágiles (Scrum)
  - Servidor de integración continua (CI/CD)
- Se puede usar desde github.com gratuitamente para proyectos públicos
  - https://docs.github.com

2.2 Contexto para integración continua



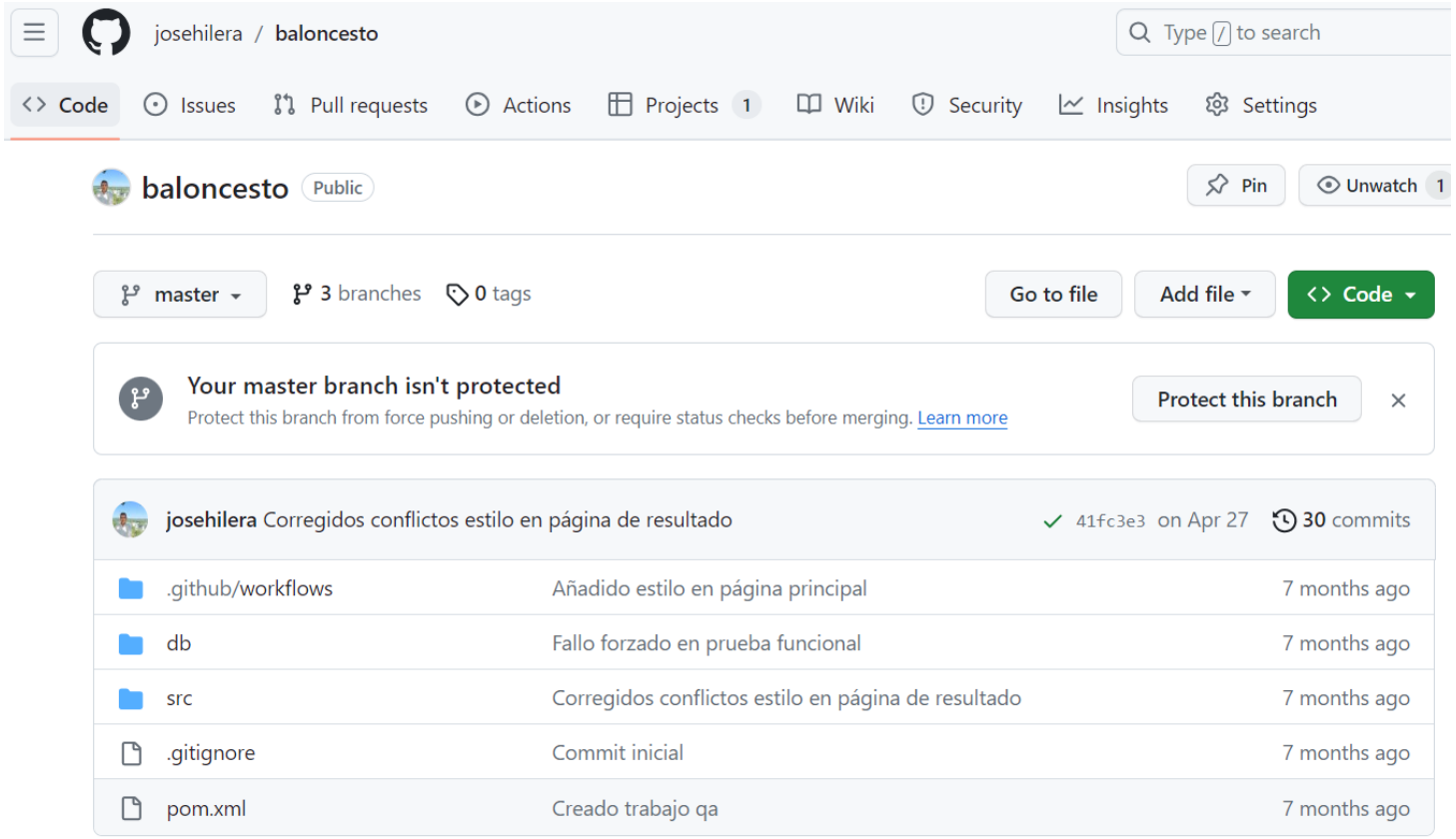
2.3 Registrarse para usar GitHub

- Para crear una cuenta en github.com hay que acceder a https://github.com y seleccionar:
  - Sign in > Create an account
- Existen diferentes planes y precios
  - https://docs.github.com/es/get-started/learning-about-github/githubs-plans
- Con una cuenta personal gratuita se puede trabajar con:
  - Repositorios públicos, pudiendo utilizar toda la funcionalidad
  - Repositorios privados, con una funcionalidad limitada



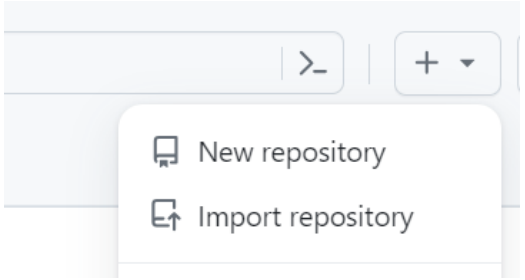
2.4 Repositorios

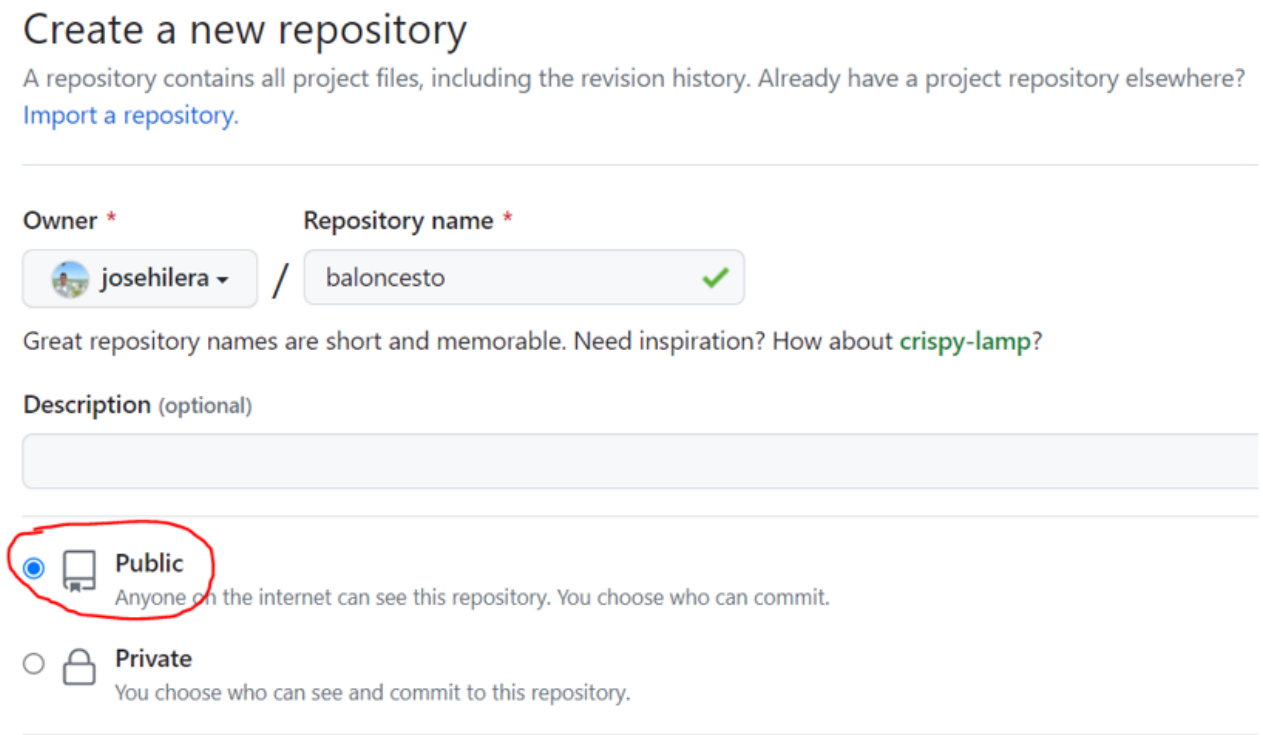
- Un repositorio en GitHub es un repositorio Git remoto, compuesto por archivos y carpetas
  - Organizados en commits y ramas (branches)
- Documentación:
  - https://docs.github.com/es/repositories/creating-and-managing-repositories/about-repositories



2.5 Creación de un repositorio

- Un nuevo repositorio se crea seleccionando la opción “New repository” desde la página principal de nuestra cuenta
  - Se recomienda crear repositorios públicos





2.6 Comandos básicos de Git (1/2)

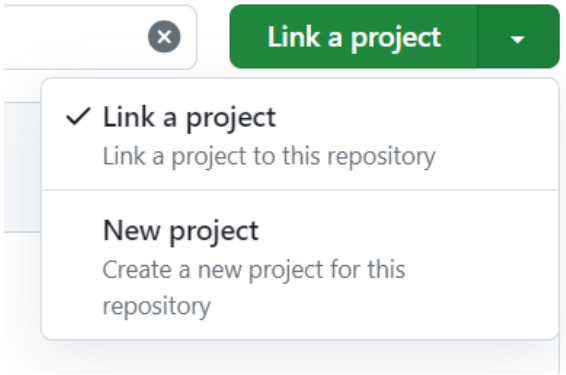
- Crear repositorio local en la carpeta actual
  - `git init`
- Copiar un repositorio local en uno remoto
  - `git remote add origin https://gitlab.com/josehilera/baloncesto.git`
- Consolidar cambios en repositorio local
  - `git add .`
- Ver estado del repositorio local
  - `git status`
- Guardar cambios creando commit en repositorio local
  - `git commit -m "Commit inicial"`
- Enviar commit a repositorio remoto
  - `git push`
  - `git push origin`
  - `git push origin rama`
  - `git push –u origin rama`

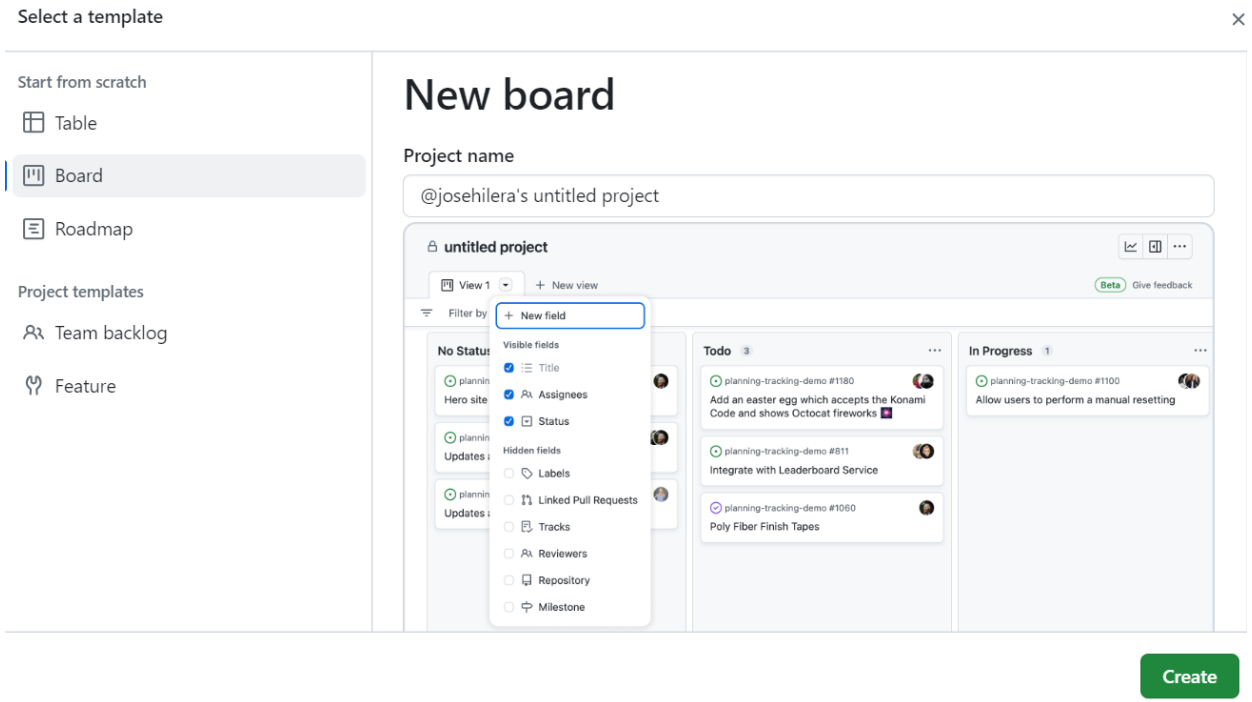
2.6 Comandos básicos de Git (2/2)

- Crear rama en repositorio local
  - `git branch rama`
- Ver ramas en repositorio local
  - `git branch`
- Cambiar de rama
  - `git checkout rama`
- Subir cambios de la rama local actual a una rama remota
  - `git push origin rama`
- Actualizar el repositorio local con la versión actual de una rama del repositorio remoto
  - `git pull origin rama`
- Clonar un repositorio remoto en la carpeta actual del ordenador local
  - `git clone https://gitlab.com/josehilera/baloncesto.git`
- Más comandos en la documentación oficial de Git (incluye libro en español)
  - https://git-scm.com/doc

2.7 Creación de un proyecto asociado a un repositorio

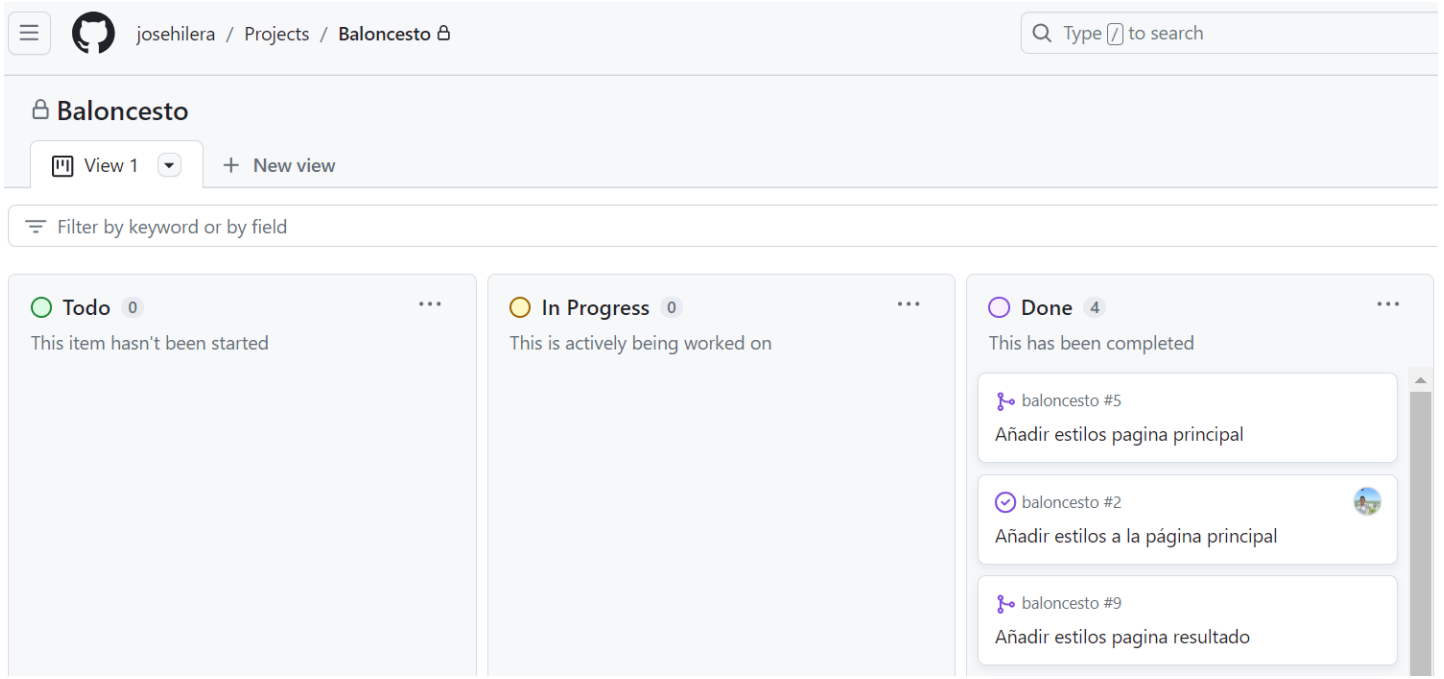
- En un repositorio se puede crear un proyecto para la gestión del desarrollo del software del repositorio





2.8 Gestión ágil de un proyecto (Scrum)

- En GitHub los siguientes conceptos son equivalente a los conocidos de Scrum
  - Issue = User Story
  - Board = Board
  - Milestone = Sprint
  - Issue list = Product Backlog
- Documentación:
  - https://docs.github.com/es/issues/planning-and-tracking-with-projects



2.9 Servidor de Integración Continua (CI/CD) GitHub Actions

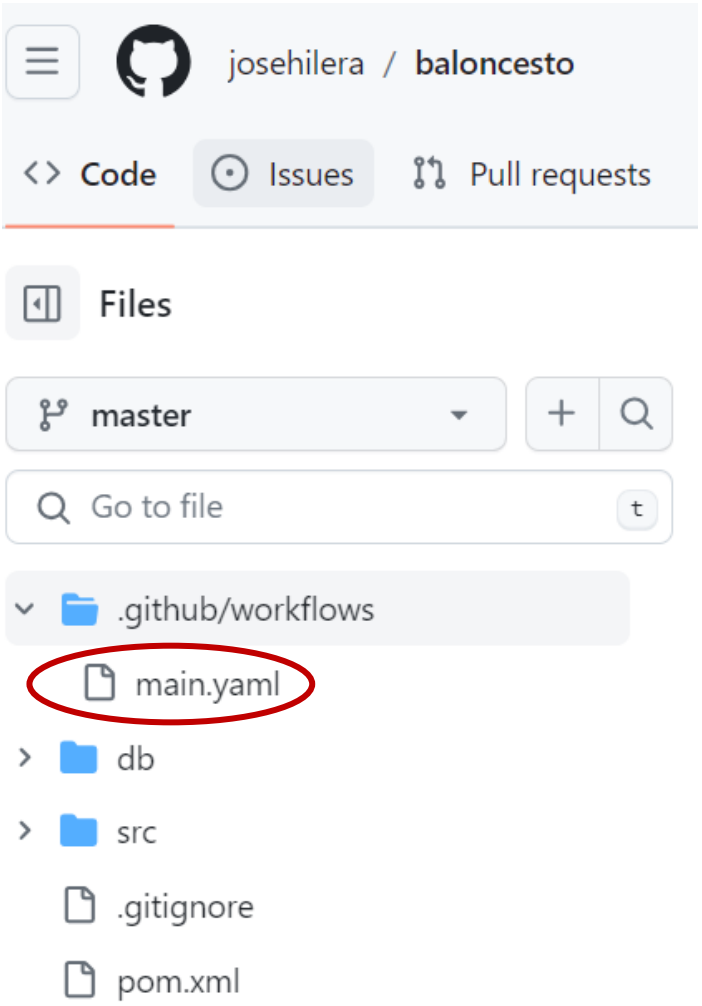
- GitHub incluye la funcionalidad “GitHub Actions”, que permite que actúe como servidor de integración continua, para que todos los miembros del proyecto puedan integrar fácilmente sus cambios en el código fuente que se guarda en el repositorio.
- Permite configurar el flujo de trabajo (workflow) del proyecto en el archivo `main.yaml`, ubicado en la carpeta `.github\workflows\` del propio repositorio
  - Definiendo cada uno de los trabajos (jobs) a realizar
- Re-ejecuta automáticamente el workflow cada vez que se suben al repositorio cambios en el código fuente del repositorio, realizando integración continua
- Documentación: https://docs.github.com/es/actions

3. Definición de workflows

- 3.1 Archivo .github\workflows\main.yaml
- 3.2 Lenguaje de definición de un workflow
- 3.3 Elementos básicos
- 3.4 Ejecución del workflow de un repositorio
- 3.5 Elementos “needs” y “if”
- 3.6 Elemento “uses”

3.1 Archivo .github\workflows\main.yaml

- El workflow de un repositorio se define en el archivo main.yaml ubicado en la carpeta .github\workflows del repositorio
- Ejemplo de proyecto: https://github.com/josehilera/baloncesto



3.2 Lenguaje de definición de un workflow

- Referencia del lenguaje de definición de workflows
  - https://docs.github.com/es/actions/using-workflows/workflow-syntax-for-github-actions
- Elementos habituales en un workflow
  - name, on, jobs, runs-on, steps, uses, run, needs, continue-on-error, if, with, …

baloncesto / .github / workflows / main.yaml

josehilera Añadido estilo en página principal ✓

Code | Blame | 70 lines (64 loc) · 1.99 KB

```yaml
name: CI

on: push

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: descargar repostorio
        uses: actions/checkout@v3
      - name: pruebas-unitarias
        run: mvn test
      - name: empaquetar
        run: mvn package -DskipTests=true
```

3.3 Elementos básicos

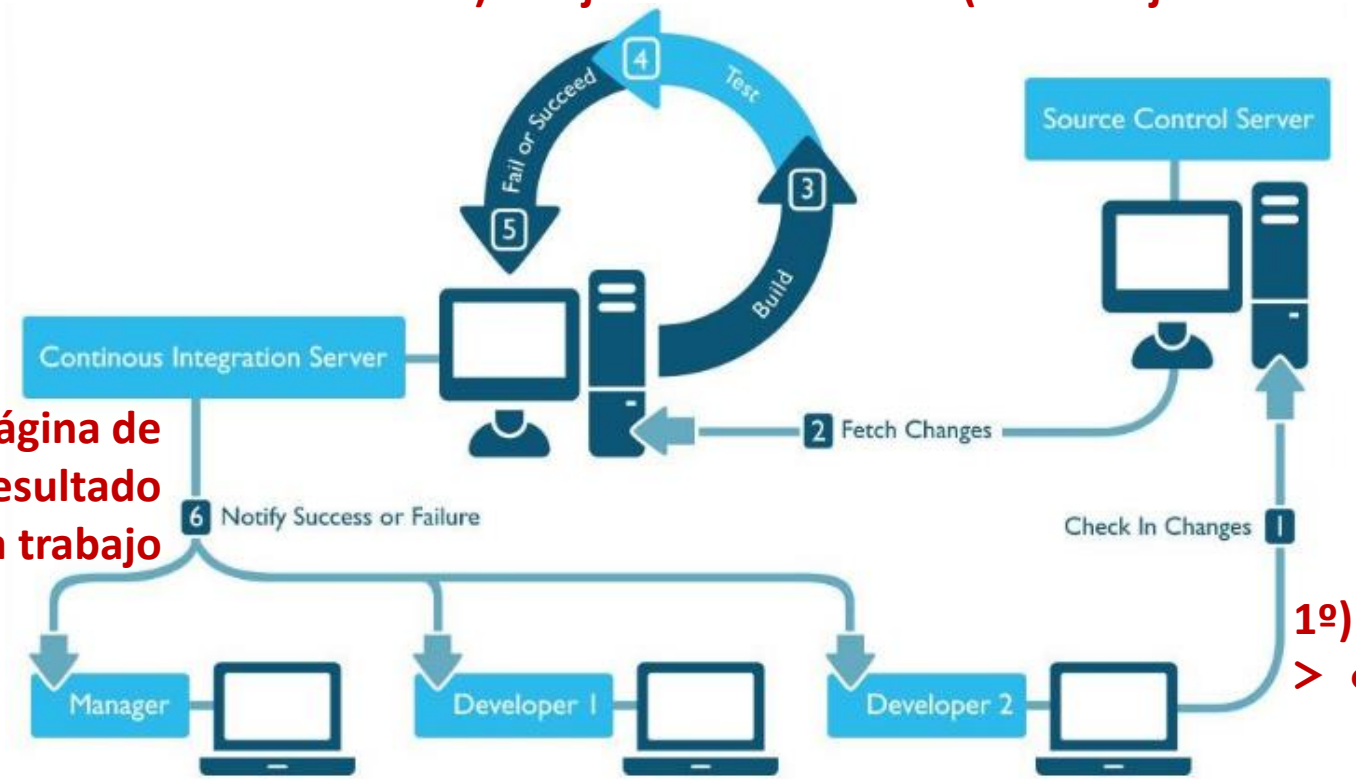
- Se pueden incluir comentarios con “#”
- Primero, en “name” se indica el nombre del workflow
- Después, en “on” se indica cuándo debe ejecutarse el workflow, por ejemplo, cuando se reciba un “push”
- A continuación se definen los trabajos en la sección “jobs”
  - En cada trabajo se usa “runs-on” para indicar el runner que ejecutará el trabajo, que puede ser un runner de github.com o se pueden crear runners propios
  - Se usa “steps” para indicar los pasos que forman el trabajo
  - En cada paso se indica con ”run” la secuencia de comandos que debe ejecutar el runner encargado de ejecutar el trabajo
    - En el ejemplo. Se indica que se compile el código del repositorio, con el comando “mvn package” de Maven, que deberá estar instalado en el runner
  - Si se re-utilizan funciones (actions) creadas por terceros se incluye su nombre y versión en “uses”

```
# Comentario
name: CI
on: push
jobs:
 build:
  runs-on: ubuntu-latest
  steps:
   - name: descargar repositorio
     uses: actions/checkout@v3
   - name: pruebas-unitarias
     run: mvn test
   - name: empaquetar
     run: mvn package
...
```

3.4 Ejecución del workflow de un repositorio

- Una vez configurado el workflow en main.yaml, cada vez que un desarrollador ejecute "git push origin" desde su ordenador local hacia el repositorio remoto, GitHub Actions desencadena la ejecución en secuencia de los trabajos definidos.

2º) Se ejecuta el workflow (los trabajos indicados en main.yaml)



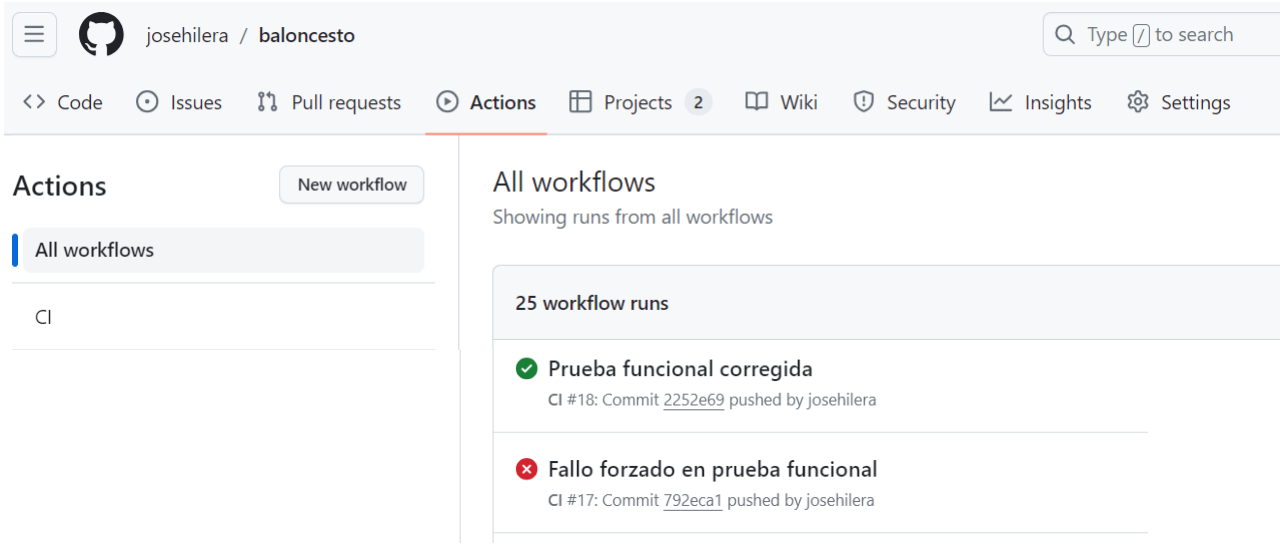
3º) Se muestra en la página de de GitHub Actions el resultado de cada trabajo

1º) Desarrollador ejecuta:

```
> git push origin
```

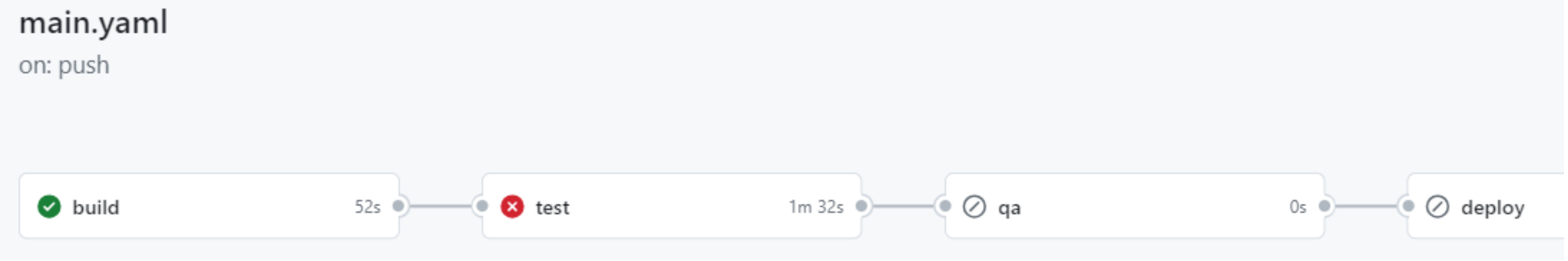
3.4 Ejecución del workflow de un repositorio Resultado global de la ejecución

- En la sección Actions, se puede ver la última ejecución del workflow (y otras anteriores) y se indica si terminó con éxito, si todos los trabajos finalizaron sin fallar ninguno



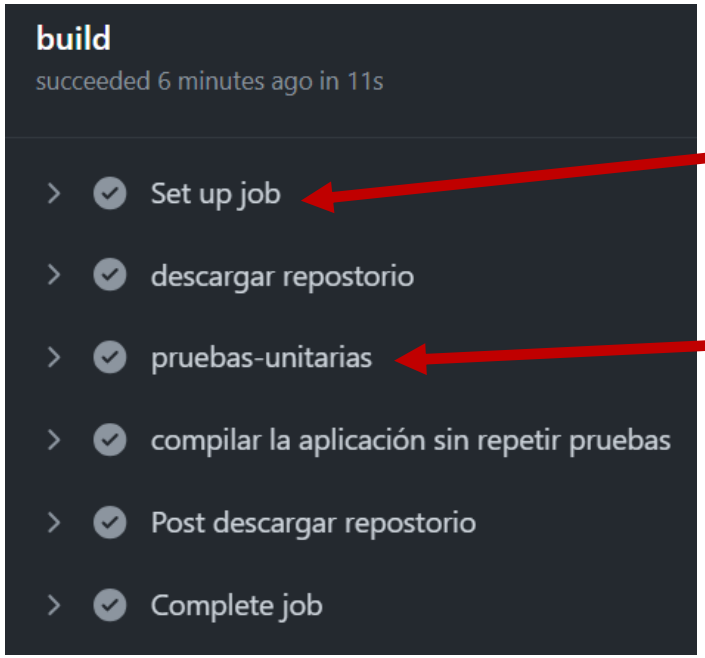
3.4 Ejecución del workflow de un repositorio Resultado de cada trabajo

- Si se selecciona una ejecución del workflow, se puede ver el resultado de los trabajos
- Cuando termina un trabajo se marca en:
  - Verde si ha finalizado con éxito
  - Rojo si se ha abortado por un error



3.4 Ejecución del workflow de un repositorio Detalle de la ejecución de un trabajo

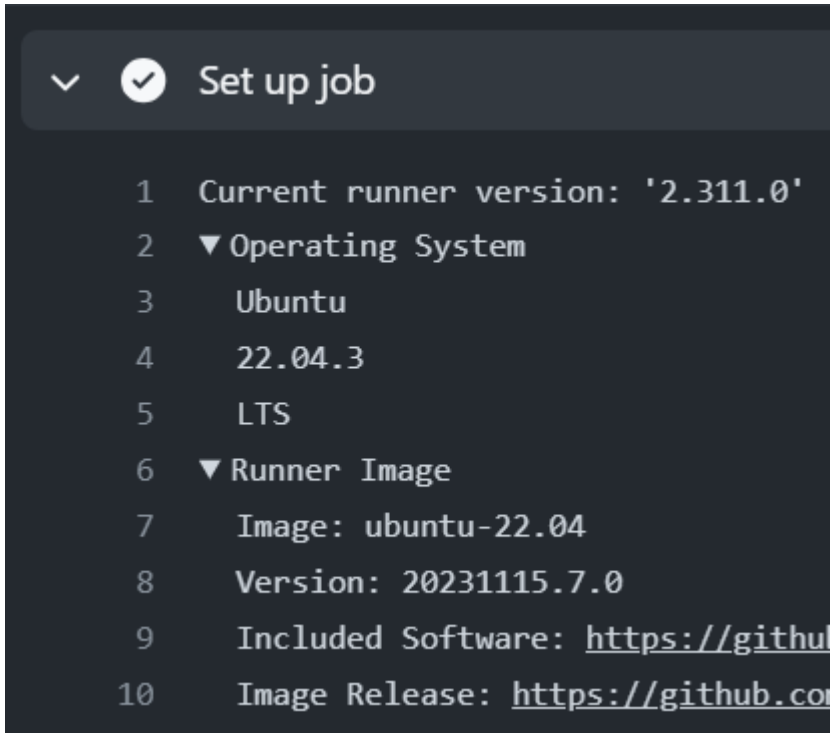
- Si se abre el trabajo, aparecen las secciones:
  - Set up job
  - Una sección por cada paso (step)
  - Complete job




```
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: descargar repositorio
        uses: actions/checkout@v3
      - name: pruebas-unitarias
        run: mvn test
      - name: compilar la aplicación sin repetir pruebas
        run: mvn package -DskipTests=true
```

3.4 Ejecución del workflow de un repositorio Detalle de la ejecución de un trabajo (Set up job)

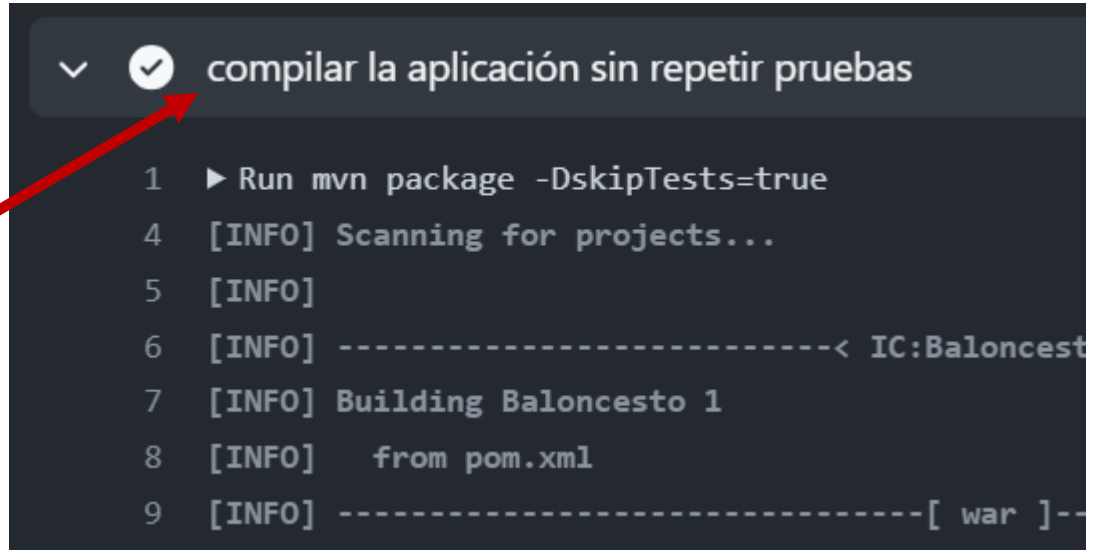
- En la sección Set up job se muestra los detalles de la maquina o contenedor que ha ejecutado el trabajo



3.4 Ejecución del workflow de un repositorio Detalle de la ejecución de un trabajo (steps)

- En cada uno de los pasos se muestra el contenido de la pantalla de la máquina o contenedor cuando ejecuta los comandos indicados en ese paso del workflow.

```
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: descargar repositorio
        uses: actions/checkout@v3
      - name: pruebas-unitarias
        run: mvn test
      - name: compilar la aplicación sin repetir pruebas
        run: mvn package -DskipTests=true
```



3.4 Ejecución del workflow de un repositorio Borrado de resultados

- Al finalizar la ejecución del trabajo, si el runner se ejecuta en un contenedor gestionado por GitHub (ej. “ubuntu-latest”), se realiza una limpieza y no se mantiene ningún archivo generado durante el trabajo.
- Si queremos mantener los archivos con los resultados, por ejemplo, de la compilación, debemos usar un runner instalado en una máquina o contenedor propio (ej. uno llamado “mi-runner”)

```
jobs:
  build:
    runs-on: ubuntu-latest
```

```
jobs:
  build:
    runs-on: mi-runner
```

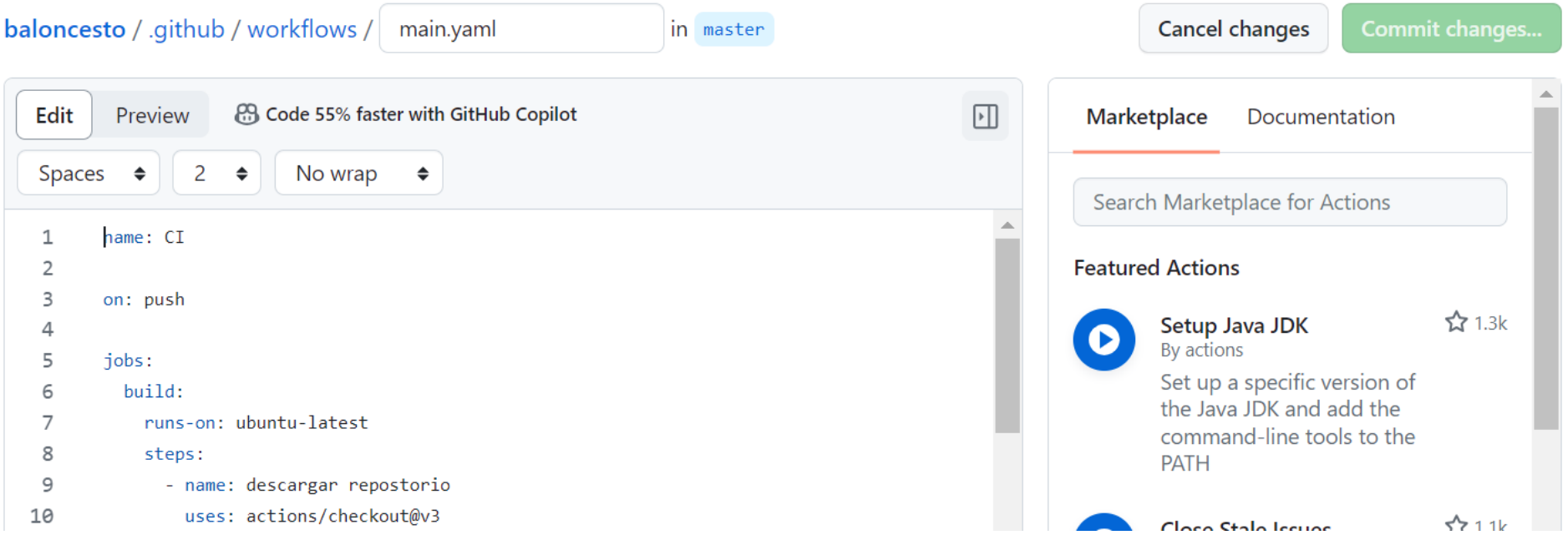
3.5 Elementos needs y if

- Para que los trabajos se ejecuten en secuencia, hay que utilizar la sección “needs” para indicar que un trabajo no se debe ejecutar hasta que haya finalizado el citado en esta sección.
- También podemos establecer una condición utilizando “if” para indicar que los pasos de un trabajo sólo se ejecuten si se cumple la condición (ej. que los cambios hayan ocurrido en la rama master).

```
jobs:
 ...
 qa:
    ...
 deploy:
    runs-on: ubuntu-latest
    needs: qa
    if: github.ref == 'refs/heads/master'
    steps:
      - name: Descargar repositorio
        uses: actions/checkout@v3
```

3.6 Elemento “uses”

- Con el elemento “uses” se pueden reutilizar funciones creadas por otros autores, denominadas “actions”, que se pueden integrar en un workflow
- Las actions se puede encontrar en el Marketplace de GitHub
  - Cuando se está editando el archivo main.yaml, se puede acceder directamente a un buscador de actions en el Marketplace



3.6 Elemento “uses” Ejemplo

- Un ejemplo puede ser la acción llamada “checkout”
  - Disponible en https://github.com/actions/checkout
- Esta acción hace que el runner que ejecuta un trabajo:
  - reciba todo el contenido del repositorio (el código fuente de la aplicación que se está desarrollando),
  - y lo guarde en la carpeta de trabajo que se indicó al crear el runner (por defecto, el nombre es _work), para poder trabajar con dichos archivos, por ejemplo, para compilarlos o para ejecutar las pruebas.

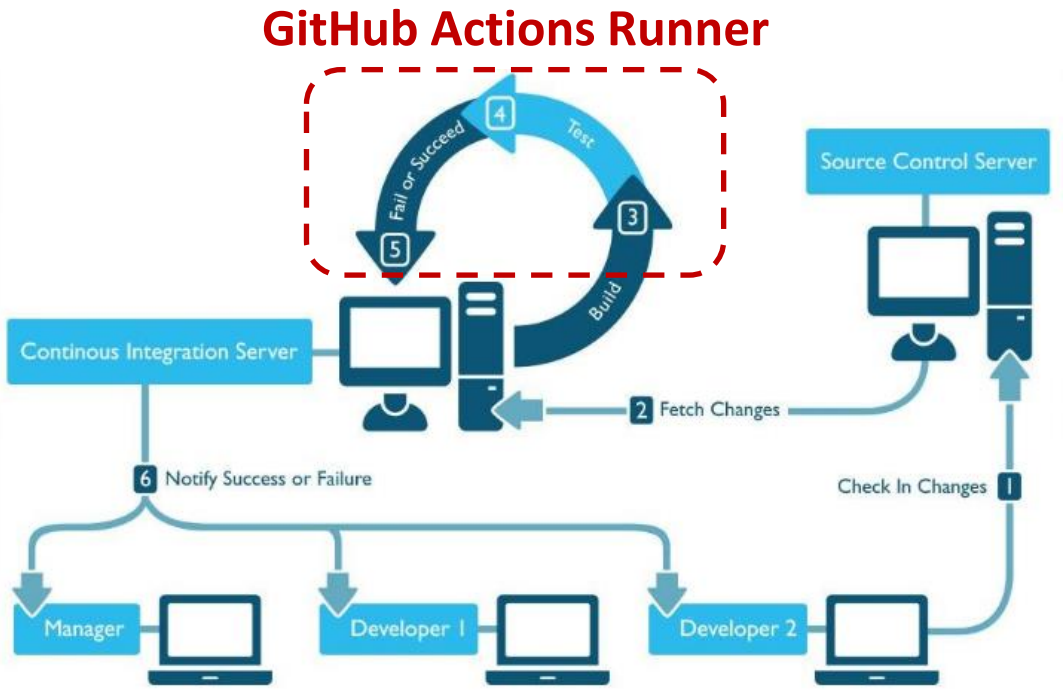
```
name: CI
on: push
jobs:
 build:
  runs-on: ubuntu-latest
  steps:
   - name: descargar repositorio
     uses: actions/checkout@v3
   - name: pruebas-unitarias
     run: mvn test
   - name: empaquetar
     run: mvn package
```

4. GitHub Actions Runner

- 4.1 Introducción
- 4.2 Utilizar un runner hospedado en GitHub
- 4.3 Utilizar un runner autohospedado
- 4.4 Instalar el programa GitHub Actions Runner

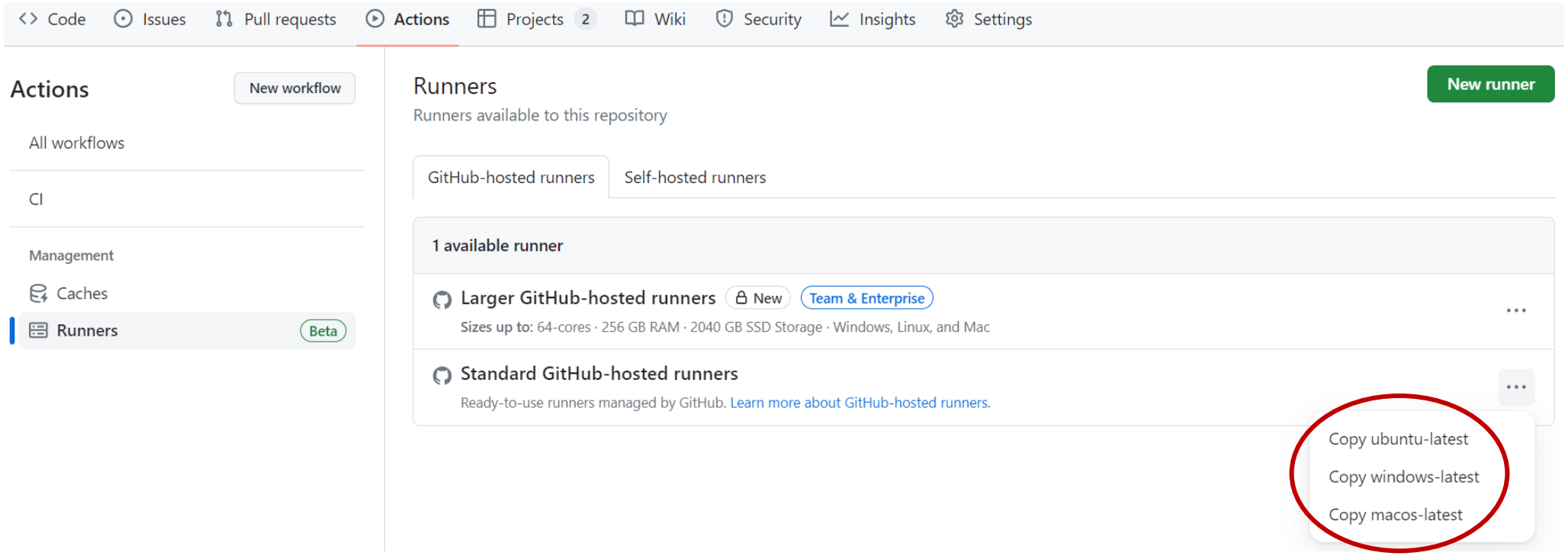
4.1 Introducción

- GitHub no ejecuta los comandos de la sección “run” de los trabajos.
- GitHub ordena a un programa llamado GitHub Actions Runner que los ejecute.
- El programa GitHub Actions Runner puede descargarse e instalarse en cualquier máquina (Linux, Windows, Mac) que tenga acceso a Internet.
  - Podemos usar runners compartidos de forma gratuita, hospedados en GitHub, pero con algunas limitaciones.
  - Pero podemos instalar y usar nuestro propio runner autohospedado sin limitaciones.



4.2 Utilizar un runner hospedado en GitHub

- En la sección Actions > Runners > GitHub-hosted runners de un repositorio, se pueden encontrar los runners hospedados en GitHub que se pueden usar de forma gratuita (ej. uno alojado en un contenedor ubuntu-latest)



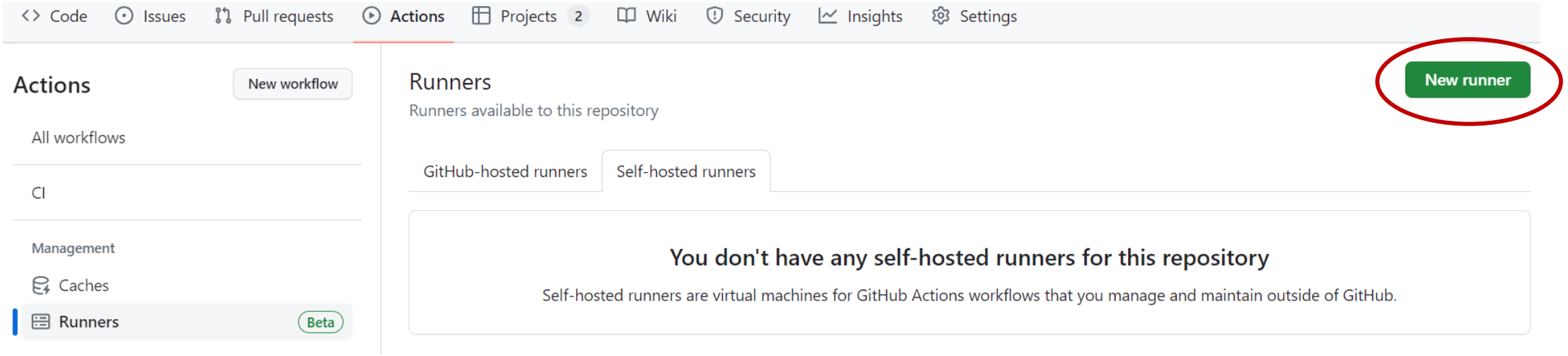
4.2 Utilizar un runner hospedado en GitHub Ejemplo

- En la sección “runs-on” de cada trabajo de un workflow se puede indicar el runner que tiene que ejecutar el trabajo.
- Por ejemplo, si se utiliza ubuntu-latest, se indicaría como:
  - runs-on: ubuntu-latest
- En un mismo workflow se pueden usar diferentes runners en diferentes trabajos.

```
name: CI
on: push
jobs:
 build:
  runs-on: ubuntu-latest
  steps:
   - name: descargar repositorio
     uses: actions/checkout@v3
   - name: pruebas-unitarias
     run: mvn test
   - name: empaquetar
     run: mvn package
...
```

4.3 Utilizar un runner autohospedado

- En la sección Actions > Runners > Self-hosted runners de un repositorio, se pueden añadir runners propios, seleccionando “New runner”
- Para poder utilizar un runner propio hay que instalar previamente en nuestra máquina el programa GitHub Actions Runner

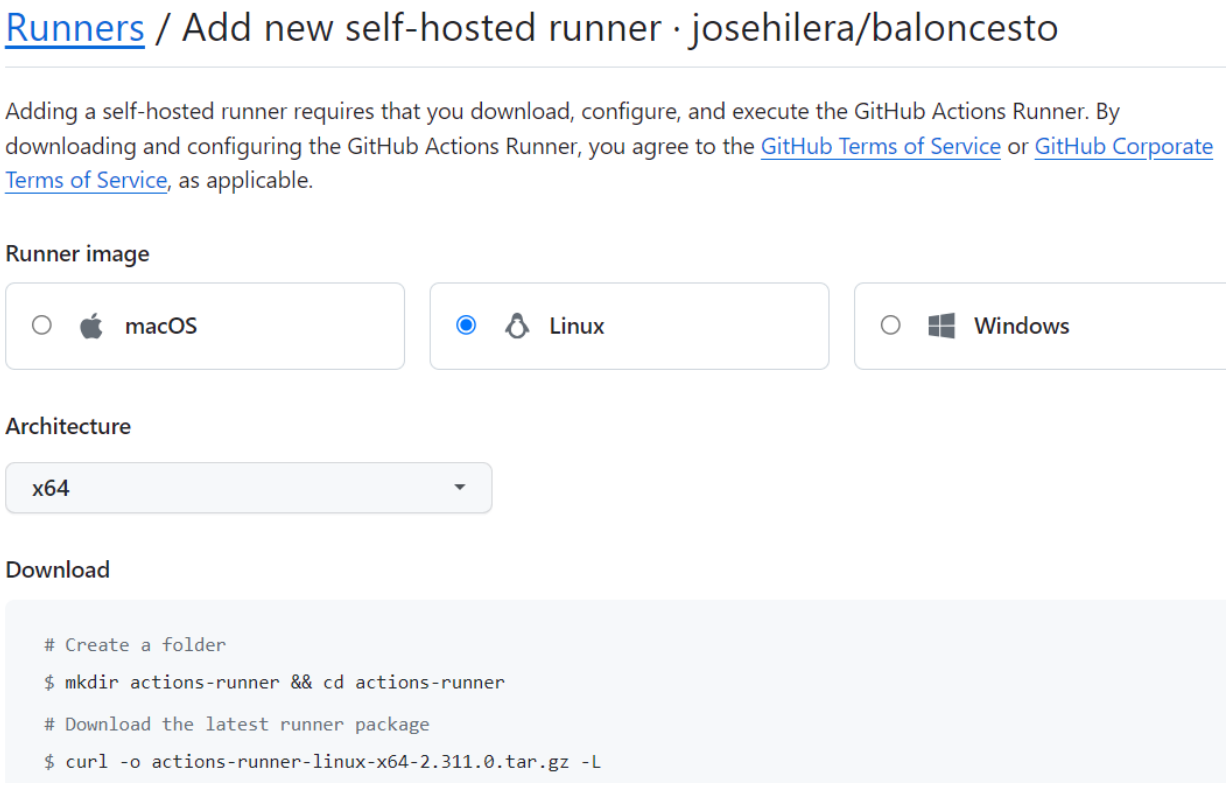


4.4 Instalar el programa GitHub Actions Runner

- Para utilizar un runner propio autohospedado, hay que instalarlo en un ordenador con acceso a Internet
- Se debe instalar el programa GitHub Actions Runner, disponible para Linux, Windows y Mac
- Se puede instalar en un contenedor Docker
- Los comandos que hay que ejecutar en nuestra máquina o contenedor para descargar e instalar un runner, dependen del sistema operativo, y se indican en la sección Actions > Runners > Self-hosted runners > New runner > New self-hosted runner

4.4 Instalar el programa GitHub Actions Runner Ejemplo

- Se ejecutan los comandos que indica GitHub en nuestra máquina, y se le da el nombre que se desee (por ejemplo, “mi-runner”)
- Una vez creado y activado en nuestra máquina, hay que indicar su nombre en el workflow




```
name: CI
on: push
jobs:
 build:
  runs-on: mi-runner
   steps:
    - name: descargar repositorio
      uses: actions/checkout@v3
    - name: pruebas-unitarias
      run: mvn test
    - name: empaquetar
      run: mvn package
...
```

5. Conclusiones

- En CI/CD un workflow (o pipeline) representa la automatización del flujo de trabajo de un proyecto, divido en trabajos
- En un repositorio de GitHub se define el workflow en el archivo .github\workflows\main.yaml
- La ejecución de los trabajos del workflow se realiza en una máquina en la que está instalado el programa GitHub Actions Runner
  - El runner puede estar hospedado en GitHub (con limitaciones si es gratuito)
  - O puede estar auto hospedado en una máquina propia del desarrollador