



# Introducción a GitHub Actions

Build Project





### Introducción

"Build Project" es un flujo de trabajo (workflow) en GitHub Actions que se ejecuta cuando hay cambios en las ramas dev y main.

¿Para qué sirve? Este workflow configura el entorno y construye nuestra aplicación en cada actualización de estas ramas.



# Pipeline para construir la aplicación

```
on:
 push:
   branches:
      dev
      - main
  pull_request:
   branches:
      dev
      - main
jobs:
 build-and-initialize:
    runs-on: ubuntu-latest
    steps:
      name: Checkout code
       uses: actions/checkout@v4

    name: Shutdown Ubuntu MySQL (SUDO)

        run: sudo service mysql stop
      name: Set up MySQL
        uses: mirromutth/mysql-action@v1.1
        with:
          host port: 3306
          container port: 3306
          mysql version: '8.0'
         mysql database: 'db_flight_bookings' # Database to be created
          mysql root password: '123456' # Root password
          mysql user: 'root'
          mysql password: '123456'
```



# Pipeline para construir la aplicación

```
    name: Wait for MySQL to be ready

       run:
         max_attempts=5
         attempt_num=1
          until mysql -h 127.0.0.1 -P 3306 -u root -p123456 -e "SHOW DATABASES;" || [ $attempt_num -eq
$max_attempts ]; do
            echo "Attempt $attempt_num/$max_attempts: Waiting for MySQL..."
            attempt num=$((attempt num+1))
            sleep 5
          done
          if [ $attempt_num -gt $max_attempts ]; then
            echo "MySQL is not ready after $max_attempts attempts. Exiting."
            exit 1
          fi
      - name: Set up JDK 21
       uses: actions/setup-java@v4
       with:
          java-version: '21'
          distribution: 'temurin'

    name: Create application-test.properties

          echo "spring.datasource.url=jdbc:mysql://127.0.0.1:3306/db_flight_bookings" >
src/main/resources/application-test.properties
          echo "spring.datasource.username=root" >> src/main/resources/application-test.properties
          echo "spring.datasource.password=123456" >> src/main/resources/application-test.properties
          echo "spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver" >>
src/main/resources/application-test.properties
          echo "spring.jpa.hibernate.ddl-auto=create-drop" >> src/main/resources/application-
          echo "spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect" >>
src/main/resources/application-test.properties
          echo "spring.sql.init.mode=always" >> src/main/resources/application-test.properties
      - name: Build API
        run: mvn clean install -DskipTests
```



### ¿Cuándo se activa?

#### Este flujo se activa:

- Al hacer un push en las ramas dev o main.
- Al abrir un pull request hacia estas ramas.
- Es importante limitar las ramas para mantener la automatización controlada.
- Ejemplo:

```
on:
  push:
  branches:
    - dev
    - main
  pull_request:
    branches:
    - dev
    - main
```



# Definición de trabajo (Job)

Definimos el job build-and-initialize que correrá en un entorno ubuntu-latest. Cada paso se ejecuta secuencialmente. ¿Por qué Ubuntu? Porque es un sistema compatible, rápido y seguro para correr builds en GitHub.

jobs:
 build-and-initialize:
 runs-on: ubuntu-latest



### Paso 1: Checkout del código

Primero necesitamos acceder al código fuente en el repositorio

¿Qué hace? Clona el repositorio para que los pasos siguientes puedan trabajar con el código localmente.

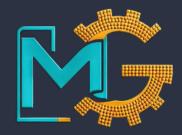
```
steps:
    # 1. Checkout the code
    - name: Checkout code
    uses: actions/checkout@v4
```



### Paso 2: Apagar MySQL en Ubuntu

Si MySQL ya está corriendo en el entorno, este comando lo detiene para evitar conflictos con nuestro contenedor de MySQL.

```
    # 2. Stop default MySQL (if running)
    name: Shutdown Ubuntu MySQL (SUDO)
    run: sudo service mysql stop
```



### Paso 3: Configuración de MySQL en un contenedor

Usamos mirromutth/mysql-action para levantar una instancia de MySQL en Docker.

Consejo: Cambia la contraseña y parámetros según tus necesidades.

```
• • •
     # 3. Set up MySQL using mirromutth/mysql-action
     - name: Set up MySQL
       uses: mirromutth/mysql-action@v1.1
       with:
         host port: 3306
                                    # Host port
         container port: 3306
                                   # Container port
         mysql version: '8.0'
                                     # MySQL version
         mysql database: 'db_flight_bookings' # Database to be created
         mysql root password: '123456' # Root password
         mysql user: 'root'
                                      # Root user
         mysql password: '123456'
```



### Paso 4: Espera a que MySQL esté listo

Este código espera hasta que MySQL esté disponible, intentando 5 veces antes de cancelar.

Consejo: Si tienes problemas, aumenta max\_attempts o revisa la conexión.

```
# 4. Wait for MySQL to be ready
- name: Wait for MySQL to be ready
run: |
    max_attempts=5
    attempt_num=1
    until mysql -h 127.0.0.1 -P 3306 -u root -p123456 -e "SHOW DATABASES;" ||
[ $attempt_num -eq $max_attempts ]; do
    echo "Attempt $attempt_num/$max_attempts: Waiting for MySQL..."
    attempt_num=$((attempt_num+1))
    sleep 5
    done
    if [ $attempt_num -gt $max_attempts ]; then
    echo "MySQL is not ready after $max_attempts attempts. Exiting."
    exit 1
    fi
```



### Paso 5: Configuración de JDK 21

La acción setup-java instala la versión 21 de Java, necesaria para construir nuestra API. ¿Por qué Java 21? Es una versión moderna y estable, adecuada para aplicaciones complejas.

```
# 5. Set up JDK 21
- name: Set up JDK 21
uses: actions/setup-java@v4
with:
    java-version: '21'
    distribution: 'temurin'
fi
```



### Paso 6: Crear applicationtest.properties

Este archivo configura la conexión con MySQL y otros parámetros de Spring Boot.

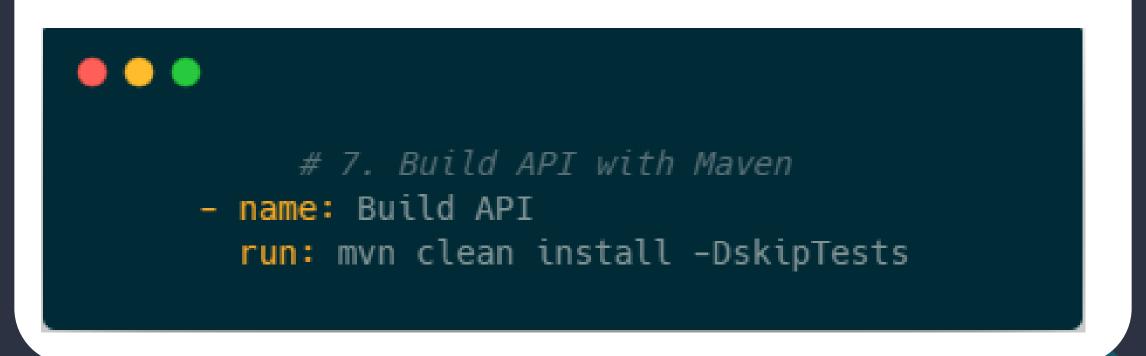
Tip: Asegúrate de no exponer datos sensibles en este archivo.

```
# 6. Create application-test.properties
- name: Create application-test.properties
run: |
echo "spring.datasource.url=jdbc:mysql://127.0.0.1:3306/db_flight_bookings" >
src/main/resources/application-test.properties
echo "spring.datasource.username=root" >> src/main/resources/application-test.properties
echo "spring.datasource.password=123456" >> src/main/resources/application-test.properties
echo "spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver" >>
src/main/resources/application-test.properties
echo "spring.jpa.hibernate.ddl-auto=create-drop" >> src/main/resources/application-test.properties
echo "spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect" >>
src/main/resources/application-test.properties
echo "spring.sql.init.mode=always" >> src/main/resources/application-test.properties
```



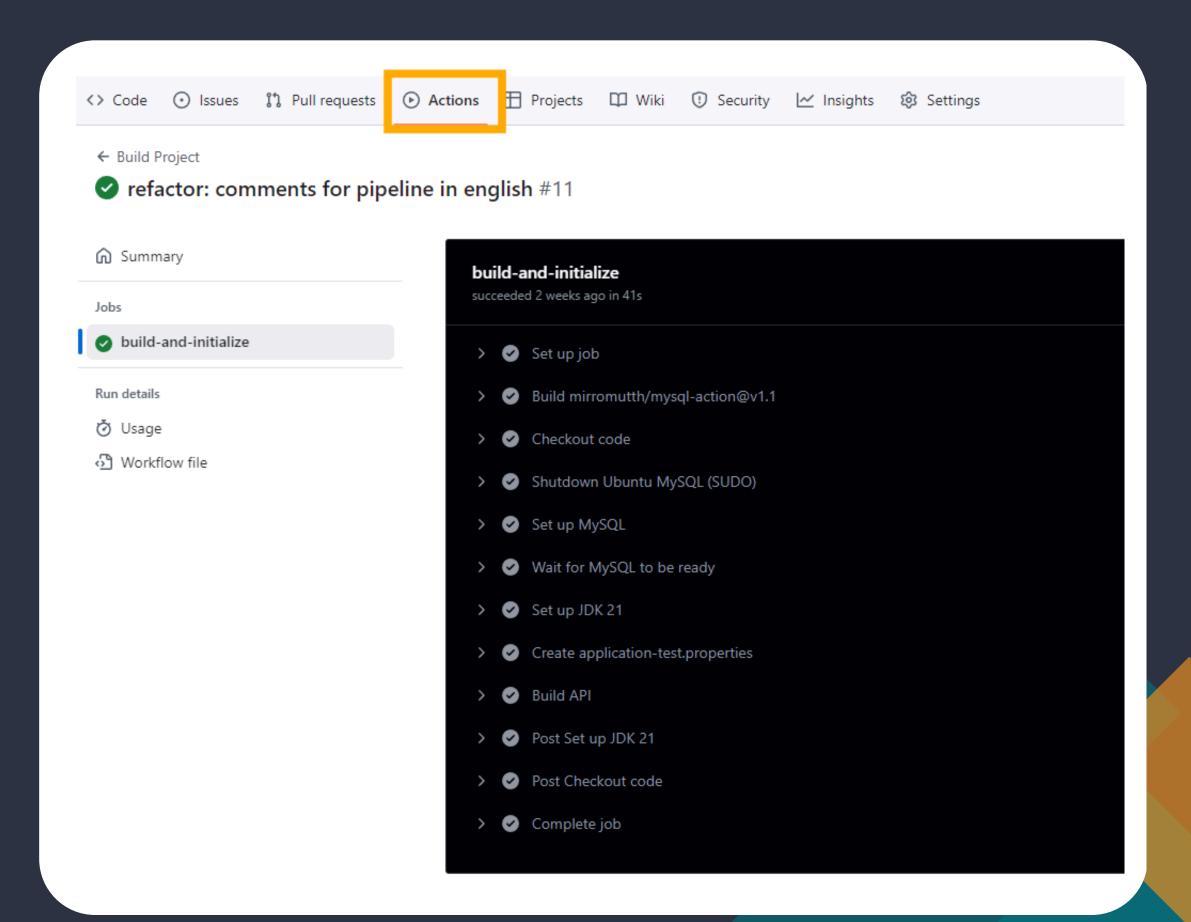
### Paso 7: Construir la API

Finalmente, usamos Maven para construir la API. Consejo: Usa -DskipTests solo en entornos de prueba. En producción, permite que los tests se ejecuten.





# Cómo revisar la ejecución en GitHub Actions





# Cómo revisar la ejecución en GitHub Actions

Accede a la pestaña de 'Actions':

- 1 En la barra superior de tu repositorio de GitHub, busca y haz clic en la pestaña 'Actions'. Ahí encontrarás una lista de todos los workflows que has configurado para tu proyecto.
- 2 Selecciona el workflow 'Build Project':

En la lista de workflows a la izquierda, selecciona 'Build Project' o el nombre del workflow que quieras revisar. Esto te permitirá ver todas las ejecuciones recientes de ese workflow.

Revisar el estado de ejecución:

Puedes ver la lista de intentos o ejecuciones (runs) recientes. Aquí se muestran las ramas afectadas y si cada intento ha sido exitoso (con un ícono verde de verificación) o fallido (con un ícono rojo).



# Cómo revisar la ejecución en GitHub Actions

4 Detalles de un intento específico:

Haz clic en un intento para ver los detalles completos del pipeline. En esta vista, puedes seguir los pasos ejecutados, como la instalación de MySQL, la configuración de JDK, y la construcción de la API. Además, podrás ver si cada paso se ejecutó correctamente o presentó errores.

5 Correcciones o ajustes:

Si encuentras errores, desde esta misma vista puedes depurar el problema revisando los logs de cada paso. Si todo ha salido bien, verás un estado final de 'Complete job' en verde.



### Conclusión y Siguientes Pasos

Este workflow es una excelente base para proyectos Java con bases de datos. Te animo a explorar:

- Otros actions para pruebas (testing).
- Configuración de CI/CD para despliegues automáticos.

¡Con estos pasos, ya tienes tu flujo de trabajo listo para construcciones en GitHub Actions!

# Si te ha gustado, comparte!



