CI/CD de Microservicios con GitHub Actions para entornos MultiCloud

MAURICIO OSPINA VERGARA 20•02•25

Rockwell Automation

expanding human possibility®

ROADMAP

CI/CD DE MICROSERVICIOS CON GITHUB ACTIONS PARA ENTORNOS MULTICLOUD

GITOPS CON GITHUB ACTIONS (?)

MLOPS EN GITHUB ACTIONS (?)

STEP 1 | STEP 2 | STEP 3 | STEP 4 | STEP 5

DESPLIEGUE DE IAC EN GITHUB ACTIONS

GITHUB ACTIONS CON COPILOT (?)

Rockwell Automation

¿QUÉ ES CI/CD?

- **CI (Integración continua):** Es la práctica de integrar el código de todos los desarrolladores con pruebas automáticas para detectar errores temprano.

- **CD:**

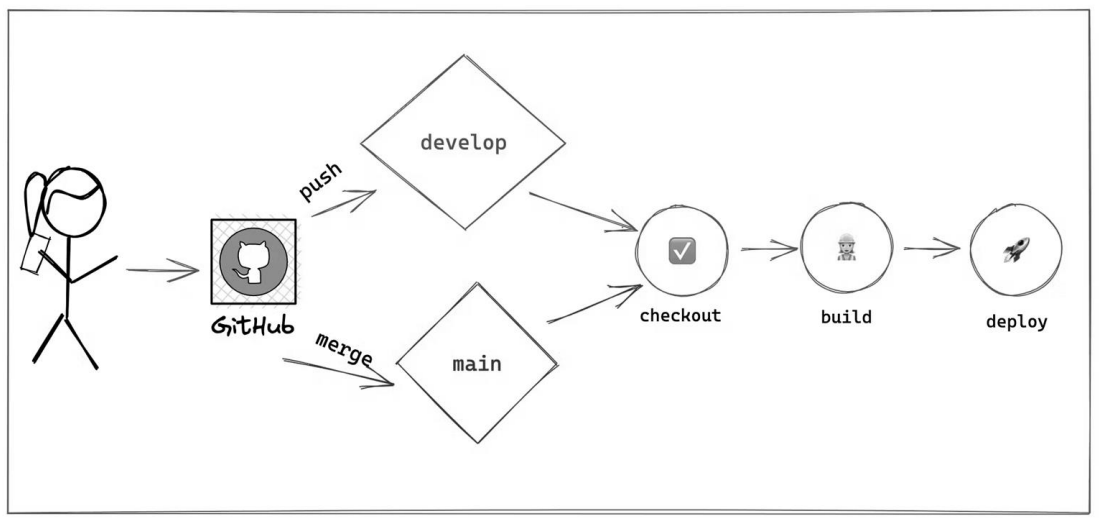
- Entrega Continua: El código está siempre listo para ser desplegado a producción, pero el despliegue se hace manualmente.

- Despliegue Continuo: El código se despliega automáticamente a producción sin intervención humana tras pasar las pruebas.

Ok, here we go again

¿POR QUÉ ES IMPORTANTE?

- **Reducción** de errores.
- Mejora la **velocidad** de desarrollo.
- Aumenta la **calidad** del código.

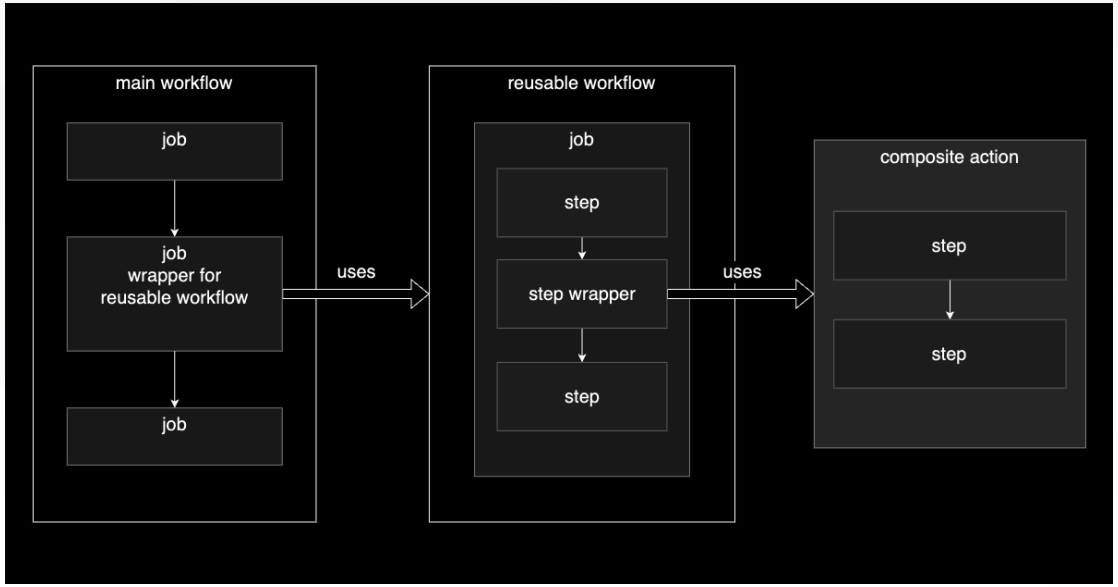


¿GitHub Actions en el mundo del desarrollo?

- GitHub Actions facilita la **integración y despliegue continuo** directamente en GitHub.
- Mejora la **automatización** en proyectos de software.

¿QUÉ SON WORKFLOWS, JOBS Y STEPS?

- **Workflow**: Un conjunto de acciones que se ejecutan.
- **Job**: Un conjunto de pasos que se ejecutan en el mismo entorno.
- **Step**: Acción individual dentro de un job.



ESTRUCTURA DE UN WORKFLOW (.YML)

- Sintaxis **clave** de un workflow:
  - **name**: Nombre del workflow.
  - **on**: Evento que dispara el workflow (ej., push).
  - **jobs**: Definición de jobs y sus pasos.

```yaml
name: CI Workflow
on:
  push:
    branches:
      - main
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - run: npm install
      - run: npm test
```

- Explicación: Este workflow se ejecuta cuando hay un **push** a la rama **main**. El job build se ejecuta en un entorno ubuntu-latest, y realiza los pasos de checkout, instalación de dependencias y ejecución de pruebas.

BUENAS PRÁCTICAS - OPTIMIZACIÓN

- **Acciones Reutilizables y Modulares:** Divide workflows complejos en **acciones modulares** reutilizables.
- **Versionado de Acciones:** Usa versiones fijas para evitar problemas con actualizaciones inesperadas.
- **Caché de Dependencias:** Usa actions/cache para **mejorar el rendimiento**.

Ejemplo (Caché de Dependencias):

```yaml
- name: Cache node modules
  uses: actions/cache@v2
  with:
    path: ~/.npm
    key: ${{ runner.os }}-node-${{ hashFiles('**/package-lock.json') }}
```

BUENAS PRÁCTICAS - FLEXIBILIDAD Y SEGURIDAD

- **Paralelización de Jobs:** Optimiza el tiempo ejecutando **múltiples jobs en paralelo**.
- **Manejo de Secretos:** Usa **GitHub Secrets** para almacenar claves API y otros secretos de forma segura.
- **Despliegue Condicional:** Usa condiciones para desplegar solo en ramas específicas o cuando se aprueben ciertos cambios.

```yaml
jobs:
  deploy:
    runs-on: ubuntu-latest
    if: github.ref == 'refs/heads/main'  # Despliega solo en la rama principal
    steps:
      - uses: actions/checkout@v2
      - run: ./deploy.sh
```

¿QUÉ ES UNA MATRIZ DE JOBS?

- **Matriz de jobs:** Permite ejecutar **múltiples variantes de un job en paralelo**. Ideal para probar un proyecto en diferentes versiones de entornos (por ejemplo, diferentes versiones de Node.js).

- **¿Cuándo usarlo?**

- Cuando necesitas ejecutar tus pruebas en múltiples versiones de Node.js, Python u otros entornos.
- Útil cuando tu proyecto debe ser compatible con varias versiones de un lenguaje o framework.

```yaml
jobs:
  test:
    strategy:
      matrix:
        node-version: [14, 16, 18]
    runs-on: ubuntu-latest
    steps:
      - uses: actions/setup-node@v2
        with:
          node-version: ${{ matrix.node-version }}
      - run: npm install
      - run: npm test
```

NOTIFICACIONES EN SLACK O CORREO

- **Recibe notificaciones** de éxito o fallo en los **workflows**.
- **Configura alertas** de Slack o correo electrónico.

```yaml
jobs:
  notify:
    runs-on: ubuntu-latest
    steps:
      - name: Send notification to Slack
        uses: slackapi/slack-github-action@v1.21.0
        with:
          channel-id: 'CXXXXXXX'
          slack-token: ${{ secrets.SLACK_TOKEN }}
          message: '¡Las pruebas han fallado!'
```

- **Explicación:** Este **workflow** enviará una notificación a un canal de Slack si alguna de las pruebas falla. Utiliza el token de Slack almacenado de manera segura en **GitHub Secrets**.

CASOS DE USO REALES Y HERRAMIENTAS RECOMENDADAS

- **Manejo de entornos multicloud:** Despliegue en múltiples nubes.
- **Docker y Kubernetes:** Construcción y despliegue de contenedores.
- **Automatización de tareas repetitivas:** Generación de documentación y mantenimiento de código.

- Tareas recomendadas para reutilizar:
  - **Trivy:** Escaneo de vulnerabilidades en contenedores.
  - **K3s:** Kubernetes liviano para despliegues rápidos.
  - **OWASP:** Seguridad en el desarrollo de software.
  - **SonarQube:** Análisis de calidad de código.
  - **Terraform:** Infraestructura como código.

FACTURACIÓN PARA GITHUB ACTIONS Y GITHUB

- **GitHub Actions:**
  - **Gratuito**: Para repositorios públicos.
  - **Repositorios privados**: Tienes una cuota gratuita de minutos de ejecución.
  - **Usuarios gratuitos**: 2,000 minutos al mes.
  - **GitHub Team**: 3,000 minutos al mes.
  - **GitHub Enterprise**: 50,000 minutos al mes.
  - **Minutos adicionales**: Después de exceder la cuota gratuita, se cobra por minuto de ejecución según el tipo de runner (Ubuntu, macOS, Windows).
- **GitHub:**
  - Los **repositorios públicos** son gratuitos.
  - Los **repositorios privados** pueden requerir un plan de pago dependiendo del uso y la cantidad de colaboradores.

Q & A

Rockwell Automation

expanding human possibility®

www.rockwellautomation.com