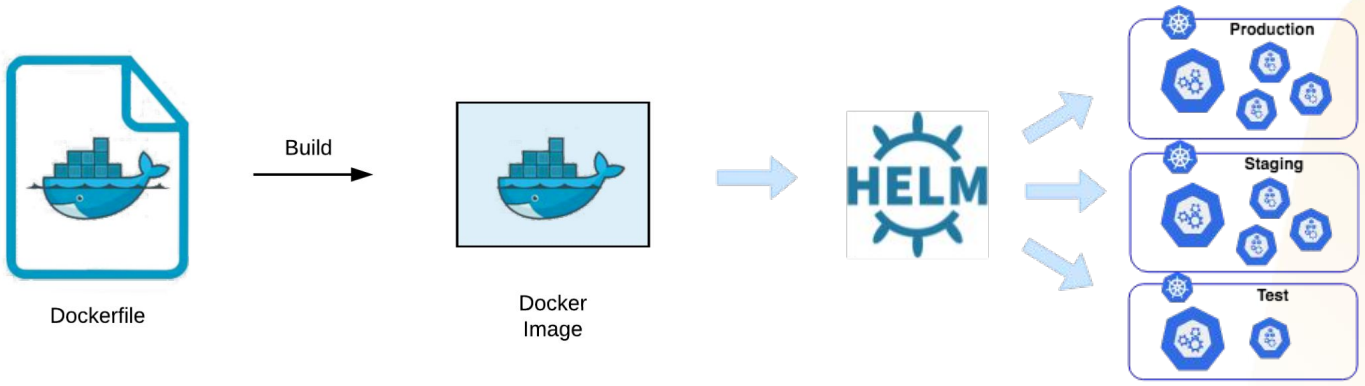
CRAFTECH

Your DevOps team

Módulo 5
CI/CD

Repaso

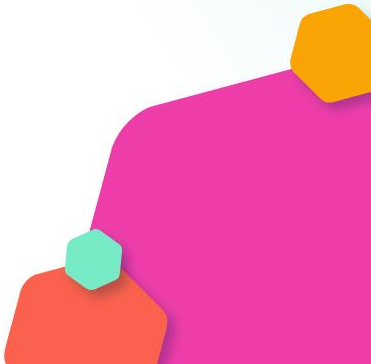
- **Docker:** dockerfile, docker-compose, build, run.
- **Kubernetes:** deploy app, exponer un servicio.



Problemas que encontramos...

- **COMPLEJO**. Sin bien, simplificado, aún se requieren una gran cantidad de pasos para actualizar un servicio.
- **POCO FIABLE**. ¿Cómo se prueban los cambios que se van a llevar a producción de una manera eficiente?
- **LENTO**. Es muy repetitivo (cansa) y hay que aprenderse la receta (traspaso de conocimiento).
- **PELIGROSO**. Se pueden cometer muchos errores (error humano).

La solución:
automatizar el proceso de deploy

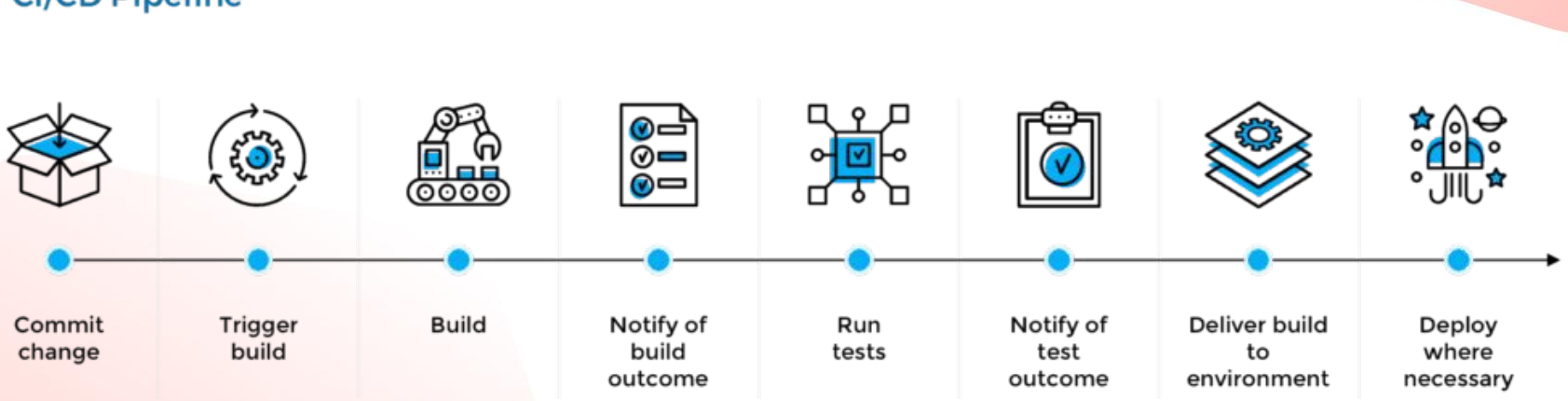


Como automatizar el proceso de deploy

CI/CD es continuous integration (CI), continuous delivery y continuous deployment (CD).

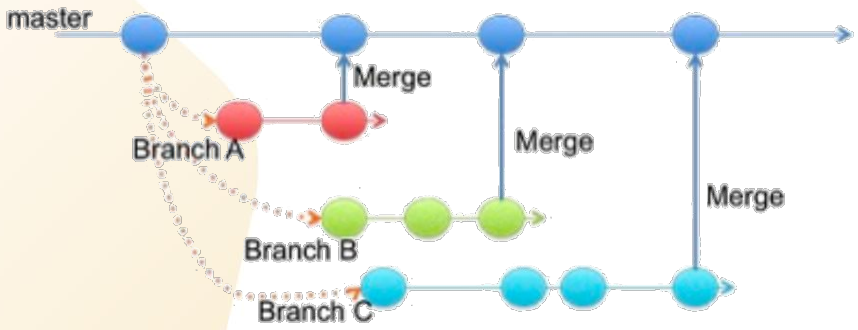
CI/CD introduce la automatización en el ciclo de vida de las aplicaciones desde la integración, el testing y el deploy.

CI/CD pipelines permiten implementar las distintas stages del ciclo de vida.



Estrategias de ramificación (branching)

Mainline Branch



Ventajas:

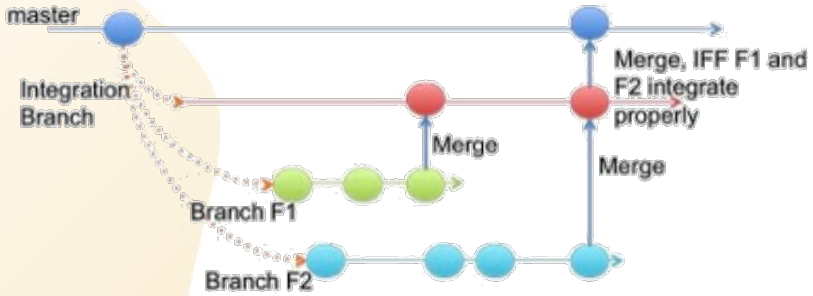
- Pocas ramas (flujos simples)
- Commits chicos
- Pruebas antes de commitear
- Si el **CD** es bueno, todos los cambios siempre listos en producción

Desventajas:

- Si los tests no son buenos, es arriesgado
- Puede servir para web pero no para software instalado (muchas actualizaciones)

Estrategias de ramificación (branching)

Branch Per Feature



Ventajas:

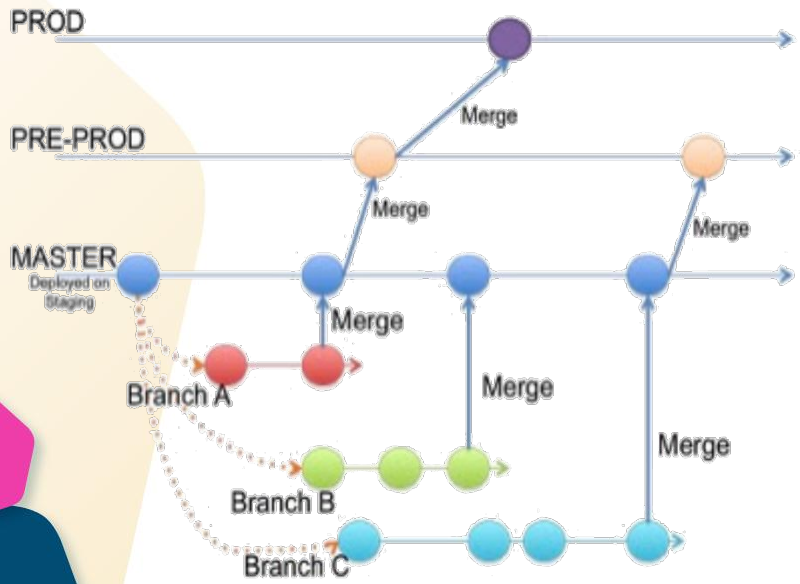
- Centrada en el deploy rápido de código
- Permite elegir cuando elegir subir los cambios a prod (gracias a la branch develop)

Desventajas:

- Mantener actualizada la feat branch
- nombres de las branches
- Limpieza de las branches

Estrategias de ramificación (branching)

Environment Based Branching



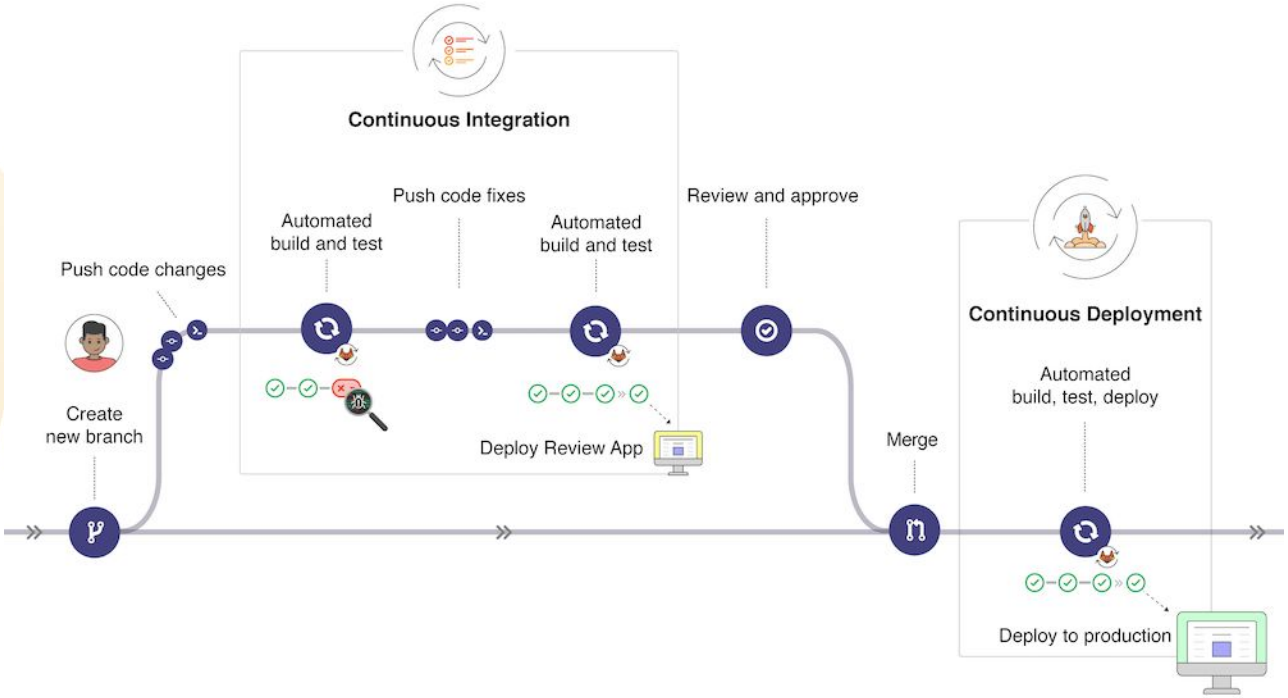
Ventajas:

- Pocas branches
- Menos merge
- Detección rápida de errores
- Facilidad para integrar

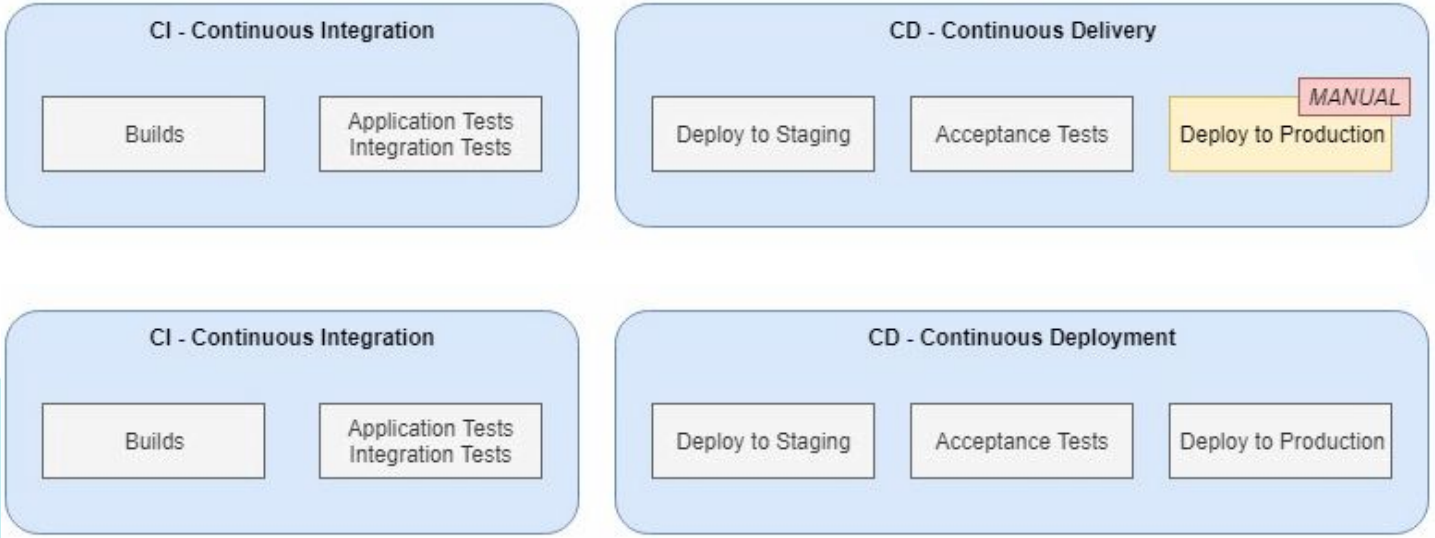
Desventajas:

- Para los desarrolladores puede ser engorroso porque constantemente deben descargar cambios de otros devs.
- código sucio
- Errores de otras personas te frenan

¿Qué es **CI** y que es **CD**?



Continuous **Delivery** vs **Deployment**



Plataformas

Las siguientes plataformas implementan pipelines **CICD**:

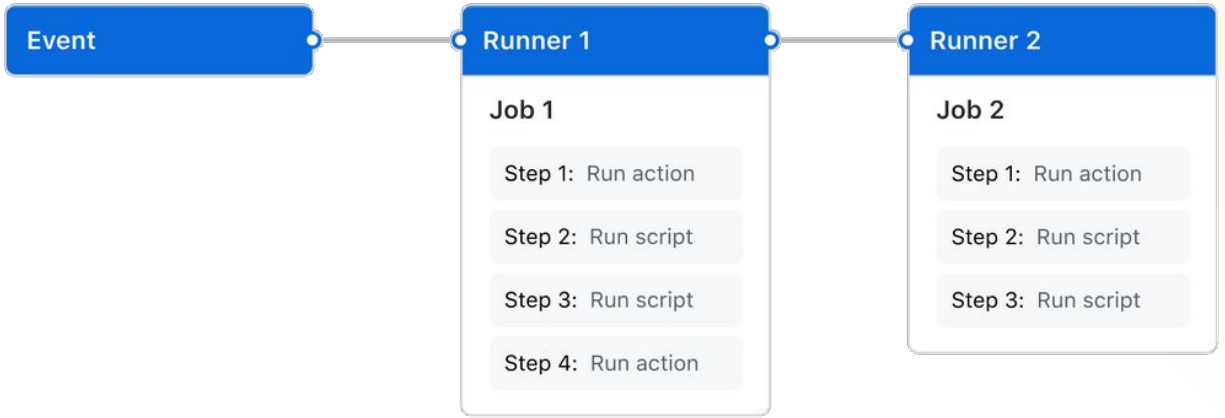
- GitLab CICD
- Github Actions
- Bitbucket Pipelines
- Circle CI
- Jenkins
- Azure Pipelines
- Google Cloud Build
- AWS: CodeBuild, CodeDeploy & CodePipelines
- **Integraciones:** Atlantis, ArgoCD, Dependabot

Github Actions

GitHub Actions

GitHub Actions es una plataforma de CI/CD que permite automatizar pipelines de build, test y deployment. Se pueden crear workflows que buildeen y testeen cada pull request que se hace al repositorio o deployen a producción los pull requests que se mergearon a producción.

GitHub provee **máquinas virtuales** con Linux, Windows, y macOS para correr los workflows, o se permite hostear **runners** propios on-premise o en una cloud.



Workflow

Un workflow es un proceso automatizado configurable que corre uno o más jobs. Workflows se definen en archivos YAML que son parte del repositorio y van a ser triggereados por un evento en el repositorio, manualmente o scheduleados.

Se definen en el directorio **.github/workflows** dentro del repositorio.

Events

Un **evento** es una actividad específica en un repositorio que triggerea un workflow.

Puede ser cuando se crea una pull request, se abre una issue, se pushea un commit, etc.

Jobs

Un **job** es una serie de **steps** en un workflow que son ejecutados por el mismo runner.

Cada step va a ejecutar un comando de bash o una actions.

Estos son ejecutados en orden y dependen del resto.

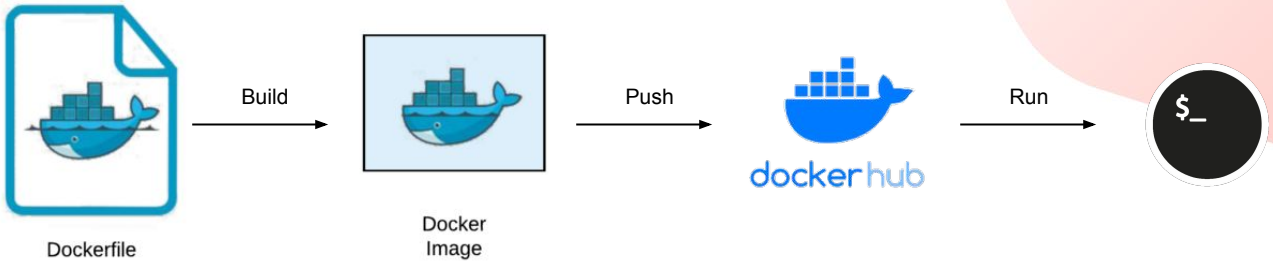
Actions

Una **action** es una aplicación custom de GitHub Actions que hace una tarea específica. Uno puede escribir sus propias actions, o puede buscar una específica en el Marketplace.

```yaml
name: learn-github-actions
on: [push]
jobs:
  check-bats-version:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '14'
      - run: npm install -g bats
      - run: bats -v
```

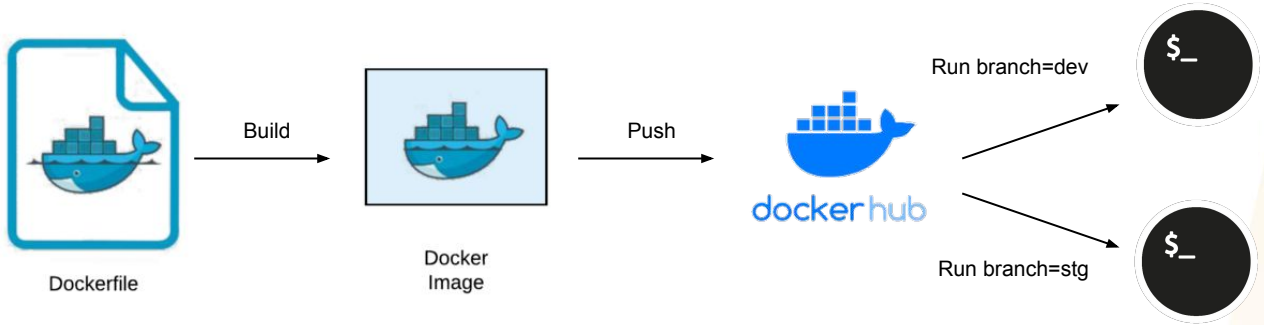
Hands On

Pipeline simple: docker build, push & run.



Hands On

Pipeline con branching: docker build, push & run en base a la branch.



Tarea

Para esta semana:

- Crear un chart de nginx. (revisar: helm create).
- CI/CD GH Actions:
  - Build y push a Docker Hub de una imagen básica de un nginx.
  - Prueba de la imagen instalando el chart en minikube (helm upgrade).