

CI/CD GitOps

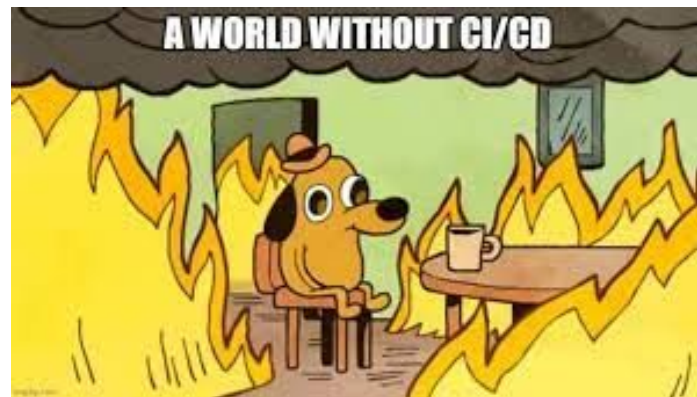
UPC - Cloud Computing Architecture

Índice - ¿Qué conceptos vamos a ver?

- Introducción
- Flujos de trabajo con Git
- Continuous integration (CI)
- Continuous delivery / deployment (CD)
 - Estrategias de despliegue
- Pipelines CI/CD
- Plataformas CI/CD
 - Github Actions
- GitOps

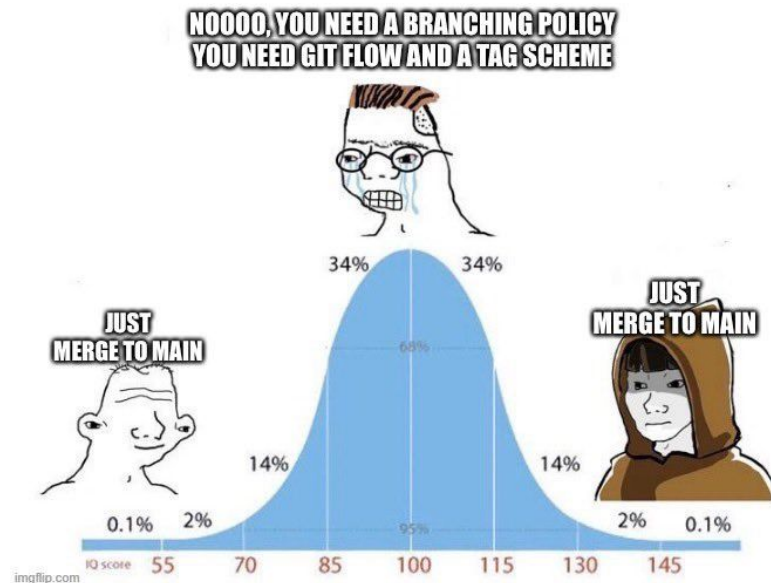
Introducción

- Retos
 - Equipos grandes, cambios de código constante
 - Deadlines exigentes
 - Personal nuevo o con poca experiencia
- Agilidad vs Calidad
 - Flujos de trabajo con GIT
 - CI/CD



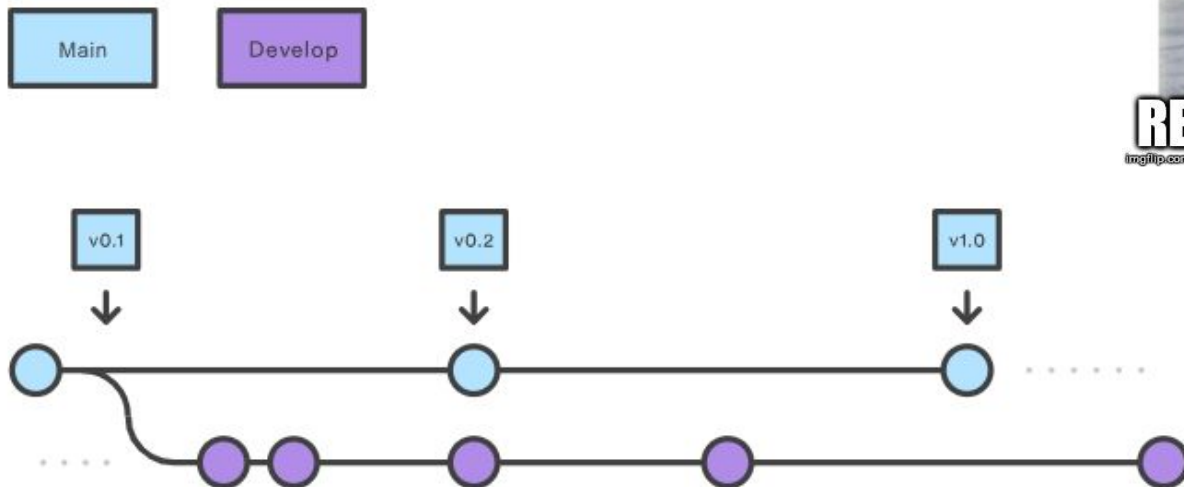
Flujos de trabajo con GIT - GitFlow

- Modelo de trabajo con ramas (branch)
- Define
 - Ramas
 - Main
 - Develop
 - Features
 - HotFix
 - Procesos
 - Hacer una release
 - Integrar funcionalidades nuevas
 - Parchear una release con un hotfix
- Útil para equipos de desarrollo grandes
- Ideado para reducir conflictos durante la integración



Flujos de trabajo con GIT - GitFlow

- Main
 - Almacena el histórico de releases oficial
- Develop
 - Se utiliza para integrar funcionalidades nuevas



COMMITING ON MASTER



REGRETING 10 SEC LATER

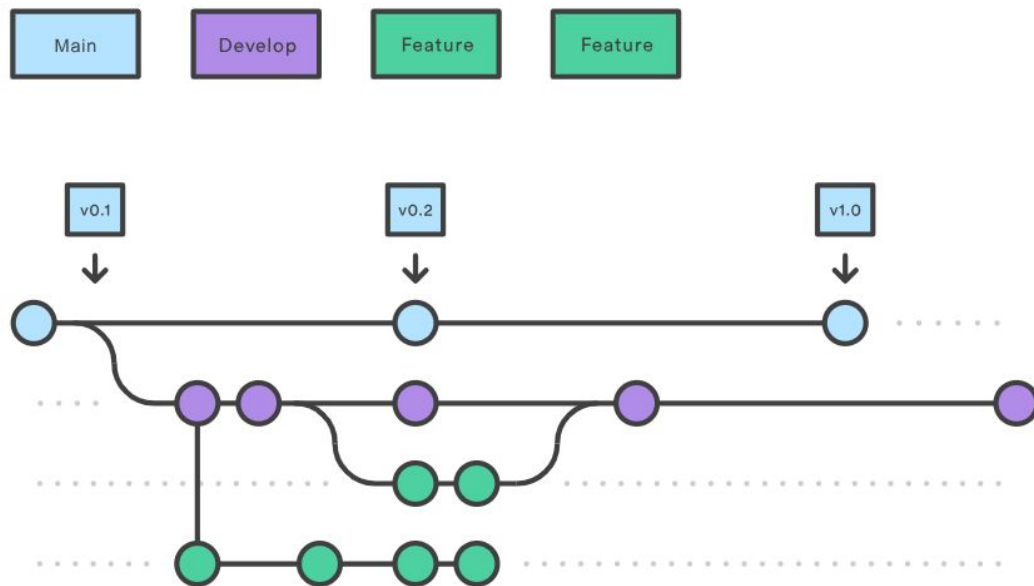
imgflip.com



Flujos de trabajo con GIT - GitFlow

- Feature

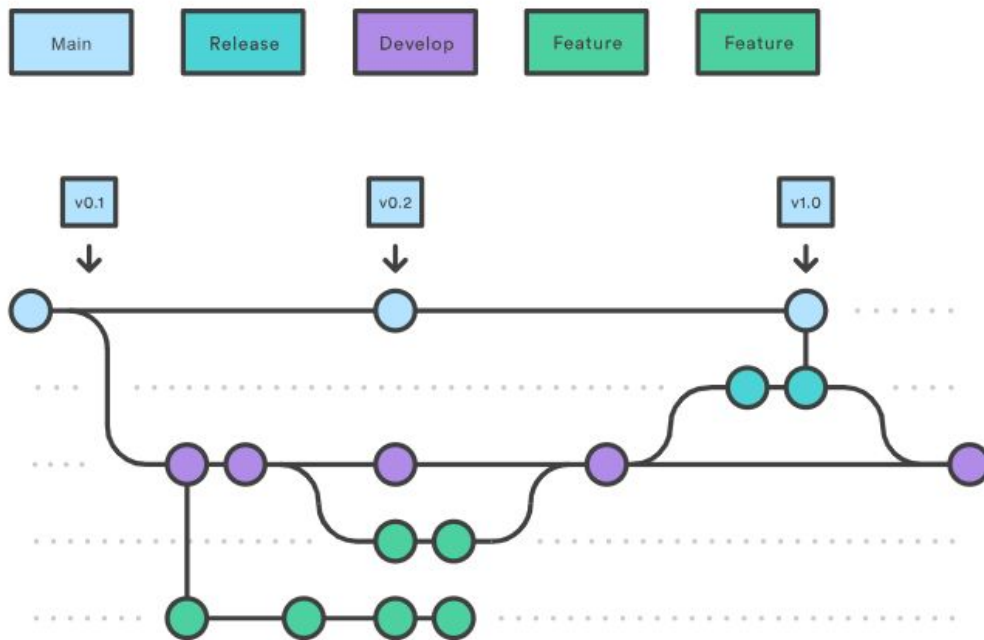
- Nacen de develop
- Donde los developers implementan los cambios
- Finalización:
 - Merge a develop
 - Se elimina la rama



Flujos de trabajo con GIT - GitFlow

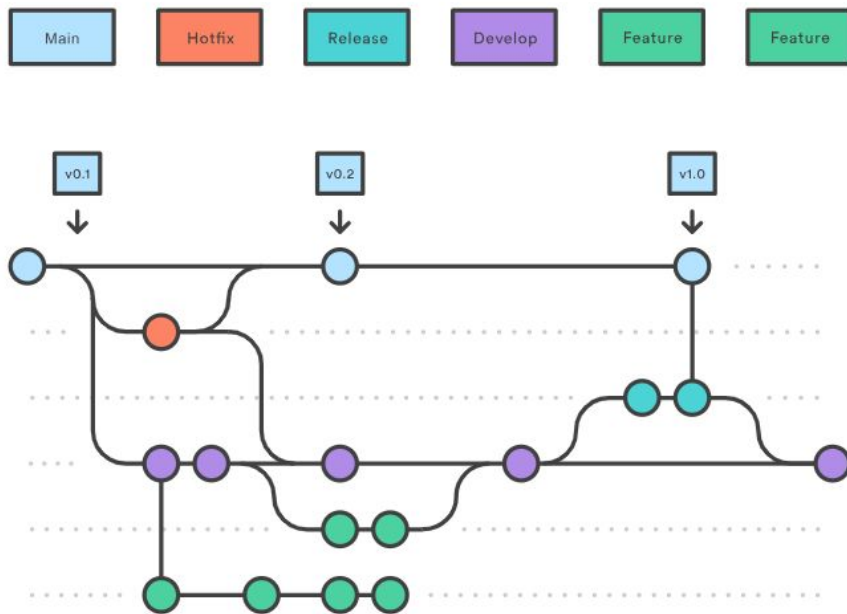
- Release

- Indica el inicio de un ciclo de release (no se admiten nuevas features)
- Solución de bugs puntuales, documentación... etc
- Finalización:
 - Merge a develop
 - Merge a main + tag
 - Se elimina la rama



Flujos de trabajo con GIT - GitFlow

- Correcciones (HotFix)
 - Las utilizamos para reparar releases rápidamente
 - Nacen de main
 - Finalización:
 - Merge a main + tag nueva versión
 - Merge a develop o release
 - Se elimina la rama



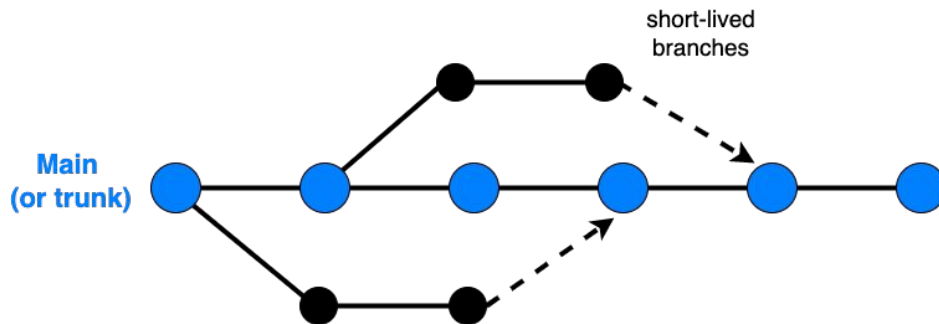
Flujos de trabajo con GIT - TBD

- Problemas de GitFlow

- Según el caso de uso puede generar un overhead de procesos
- Si las ramas de features divergen mucho, se complica la integración

- TBD

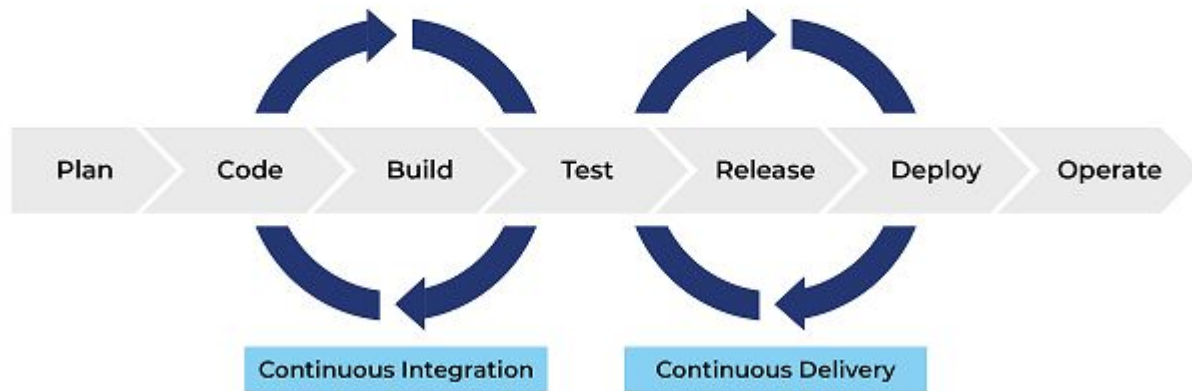
- Trunk Based Development
- Se trabaja con una única rama master (trunk) y features
- Integraciones frecuentes entre feature / trunk



--> merging is done more frequently and more easily
for shorter branches

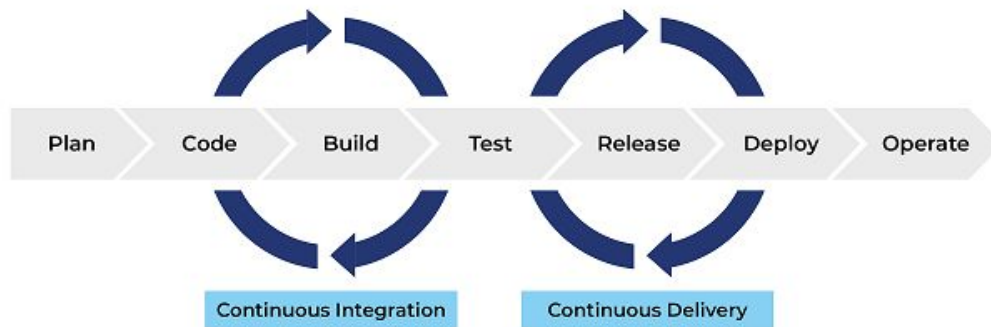
Continuous Integration (CI)

- Facilita la integración del código
- Verifica que el código cumple con
 - Las reglas de negocio
 - Testing - Unitarios / Integración / E2E
 - Los criterios de calidad
 - Seguridad
 - Buenas prácticas (SAST)



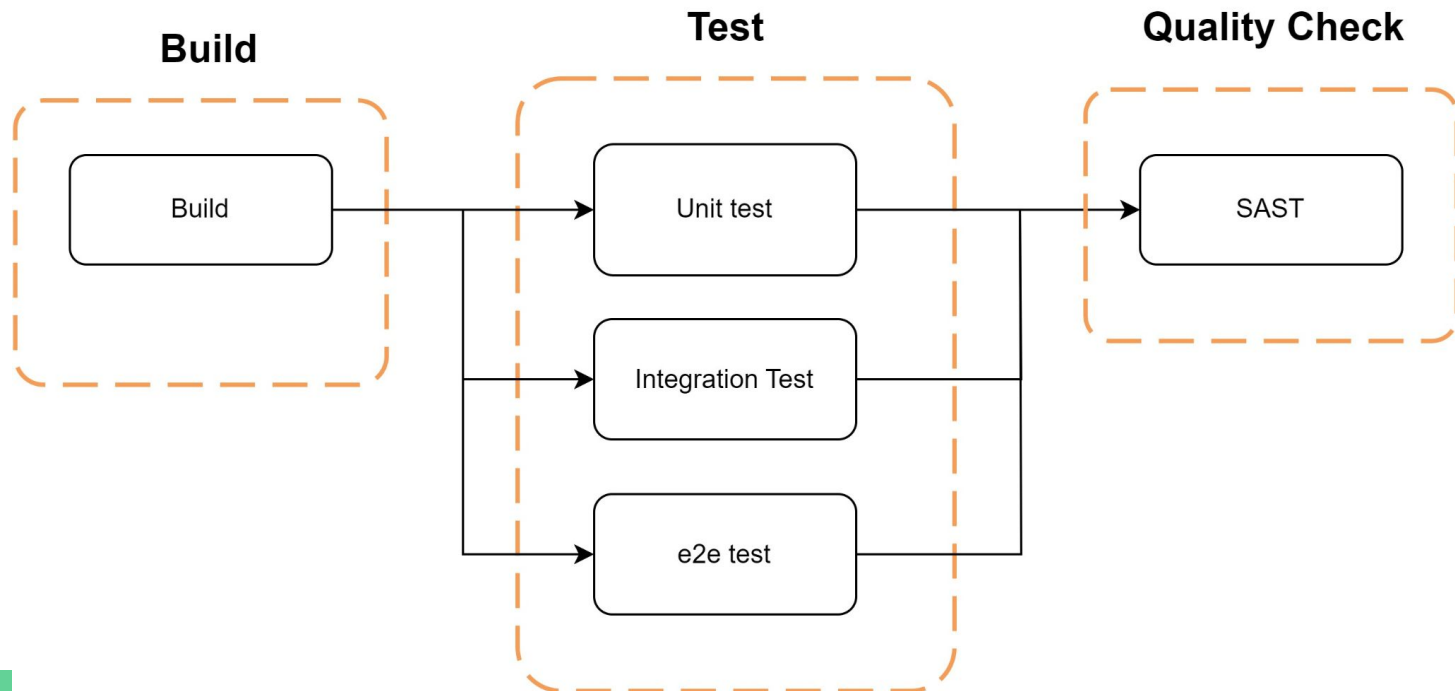
Continuous Delivery (CD)

- Continuous delivery
 - Generación del *artifact*
 - Despliegue a producción sujeto a aprobación manual
- Continuous deployment
 - Generación del *artifact*
 - Despliegue a producción automáticamente
- Proporciona un mecanismo automático para desplegar código a producción
- Simplifica el paso a producción
- Hace que el proceso sea reproducible y menos propenso a errores humanos



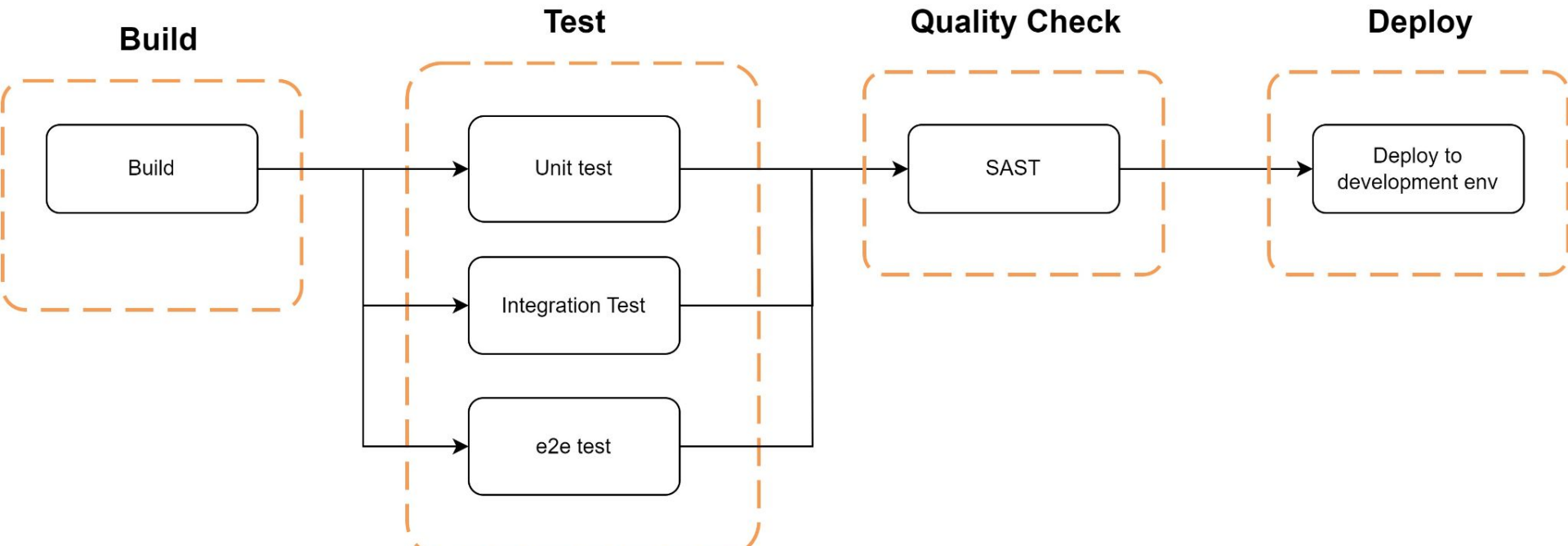
Pipelines CI/CD

- Feature branch



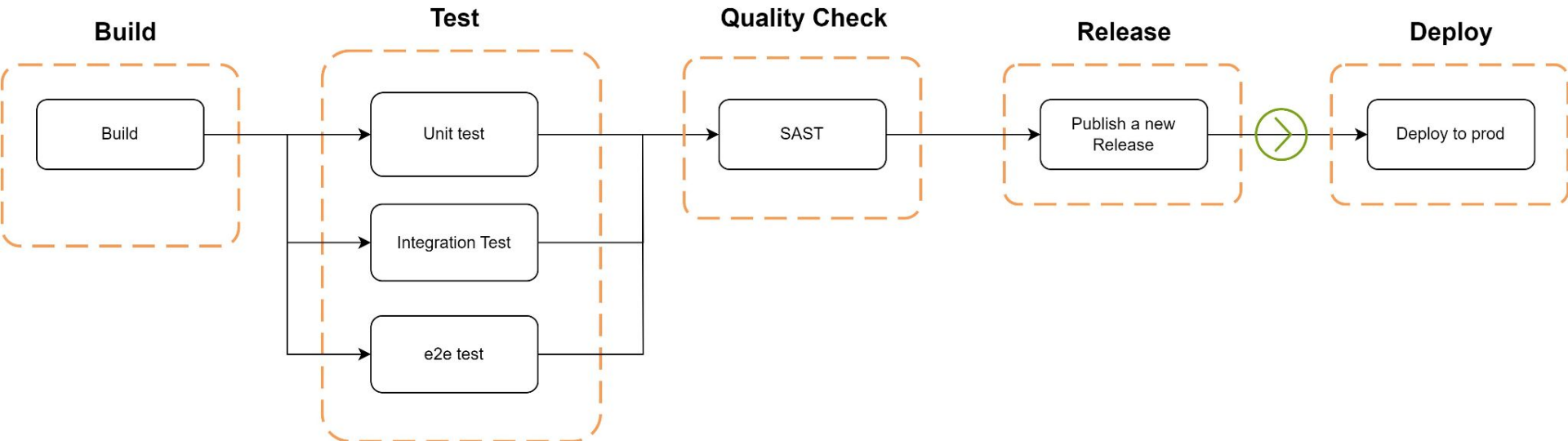
Pipelines CI/CD

- Development branch



Pipelines CI/CD

- Nueva release
- Despliegue a producción



Plataformas de CI/CD

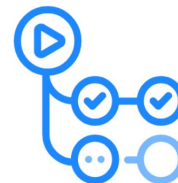
- Jenkins
- GitLab CI/CD
- **Github Actions**
- AWS
 - CodePipeline
 - Gestión del pipeline ci/cd
 - CodeCommit
 - Repositorio de código (git)
 - CodeBuild
 - Servicio de CI (compila / ejecuta tests)
 - CodeDeploy
 - Servicio de CD (despliegue)



Jenkins



GitLab



GitHub Actions



AWS
CodeCommit



AWS
CodeBuild



AWS
CodeDeploy

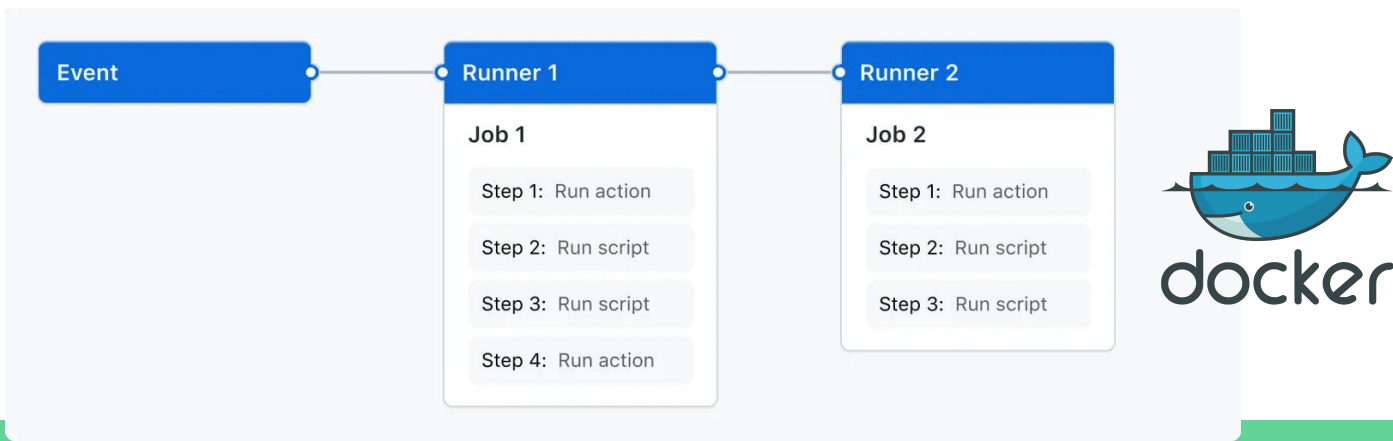


AWS
CodePipeline

Plataformas de CI/CD - GitHub Actions

- Workflow

- El equivalente a un pipeline
- Incluye uno más *Jobs*, que pueden ser ejecutados en *runners* diferentes
- Se ejecutan mediante un evento
 - Acción en el repositorio (push, merge request, release, tag...)
 - Acciones externas via API (*repository_dispatch*)
 - Programaciones con cron
 - Manual



Plataformas de CI/CD - GitHub Actions

- Runners

- Servidor donde se ejecutan las tareas (jobs) de los workflows
- Cada runner puede ejecutar una tarea a la vez
- Github ofrece runners autogestionados
 - Múltiples entornos
 - ubuntu-latest (x1)
 - windows-latest (x2)
 - macos-latest (x10)
 - Gratuito hasta 2000 minutos (por mes) y/o 500MB (artifacts, logs)
- Permite configurar runners propios
- Los jobs pueden ejecutarse en contenedores Docker dentro del runner

<https://docs.github.com/es/actions/using-github-hosted-runners/about-github-hosted-runners/about-github-hosted-runners>

Plataformas de CI/CD - GitHub Actions

- Workflow
 - Formato YAML
 - Definidos en el directorio *.github/workflows*

learn-actions.yml



```
1 name: learn-github-actions
2 run-name: ${github.actor} is learning GitHub Actions
3 on: [push]
4 jobs:
5   check-bats-version:
6     runs-on: ubuntu-latest
7     steps:
8       - uses: actions/checkout@v4
9       - uses: actions/setup-node@v3
10        with:
11          node-version: '14'
12       - run: npm install -g bats
13       - run: bats -v
14
```

Event

on: push

Runner: ubuntu-latest

Job: check-bats-version

Step 1: Check out repository

- uses: actions/checkout@v3

Step 2: Install Node.js

- uses: actions/setup-node@v3

Step 3: Install bats


- run: npm install -g bats

Step 4: Run bats

- run: bats -v

Plataformas de CI/CD - GitHub Actions

- Actions
 - Aplicación personalizada de la plataforma
 - Ejecuta acciones complejas y repetitivas
 - Reduce el código repetido en los workflows
 - Existe un Marketplace de aplicaciones donde los desarrolladores pueden publicar sus aplicaciones

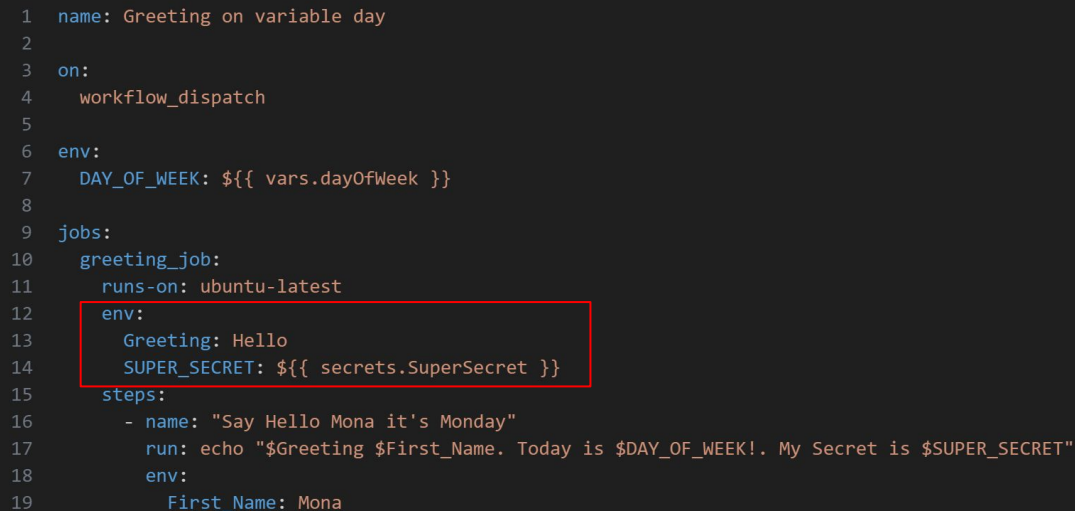


```
1 name: learn-github-actions
2 run-name: ${github.actor} is learning GitHub Actions
3 on: [push]
4 jobs:
5   check-bats-version:
6     runs-on: ubuntu-latest
7     steps:
8       - uses: actions/checkout@v4
9       - uses: actions/setup-node@v3
10         with:
11           node-version: '14'
12       - run: npm install -g bats
13       - run: bats -v
14
```

Plataformas de CI/CD - GitHub Actions

- Variables de entorno

- Permite almacenar y reutilizar información no confidencial en los workflows
- Se puede *hardcodear* en el workflow o definirlo en la configuración del repositorio / organización

A screenshot of a GitHub Actions workflow file. The code is displayed on a dark background with syntax highlighting. A red rectangular box highlights the 'env' section within the 'greeting_job' under the 'jobs' section. The code defines a workflow named 'Greeting on variable day' triggered by 'workflow_dispatch'. It sets an environment variable 'DAY_OF_WEEK' from a variable 'vars.dayOfWeek'. The 'greeting_job' runs on 'ubuntu-latest' and defines two environment variables: 'Greeting: Hello' and 'SUPER_SECRET: \${ secrets.SuperSecret }'. The workflow includes a single step named 'Say Hello Mona it's Monday' that runs 'echo' with a message incorporating these variables.

```
1 name: Greeting on variable day
2
3 on:
4   workflow_dispatch
5
6 env:
7   DAY_OF_WEEK: ${ vars.dayOfWeek }
8
9 jobs:
10  greeting_job:
11    runs-on: ubuntu-latest
12    env:
13      Greeting: Hello
14      SUPER_SECRET: ${ secrets.SuperSecret }
15    steps:
16      - name: "Say Hello Mona it's Monday"
17        run: echo "$Greeting $First_Name. Today is $DAY_OF_WEEK!. My Secret is $SUPER_SECRET"
18        env:
19          First_Name: Mona
```

- Secrets

- Permite almacenar y reutilizar información confidencial
- Proporciona mecanismos para controlar el acceso a estos

GitOps

- Conjunción **GIT + Ops (Operaciones)**
- Gestionar las configuraciones de aplicaciones y infraestructura utilizando GIT como única fuente de verdad
- Utiliza CI/CD para verificar el código
- Que aporta?
 - Un flujo de trabajo estándar para el aprovisionamiento de infraestructura
 - Incremento de confianza gracias al CI/CD
 - Uniformidad entre los diferentes entornos.

GitOps - Repositorios

- Mono-Repo

- Toda la infraestructura en un único repositorio
- Dividimos los módulos en subcarpetas
- Separamos los entornos en carpetas
 - Dominios de negocio en subcarpetas

```
1 > tree my-company-functions
2 └─ modules
3     └─ function
4         ├── main.tf      // contains aws_iam_role, aws_lambda_function
5         ├── outputs.tf
6         └─ variables.tf
7     └─ queue
8         ├── main.tf      // contains aws_sqs_queue
9         ├── outputs.tf
10        └─ variables.tf
11    └─ vpc
12        ├── main.tf      // contains aws_vpc, aws_subnet
13        ├── outputs.tf
14        └─ variables.tf
```

```
1 > tree my-company-functions
2 └─ modules
3 └─ production
4     ├── document-metadata
5     │   └─ main.tf
6     ├── document-translate
7     │   └─ main.tf
8 └─ staging
9     ├── document-metadata
10    │   └─ main.tf
11    ├── document-translate
12    │   └─ main.tf
```

GitOps - Repositorios - Mono Repo

- Pros

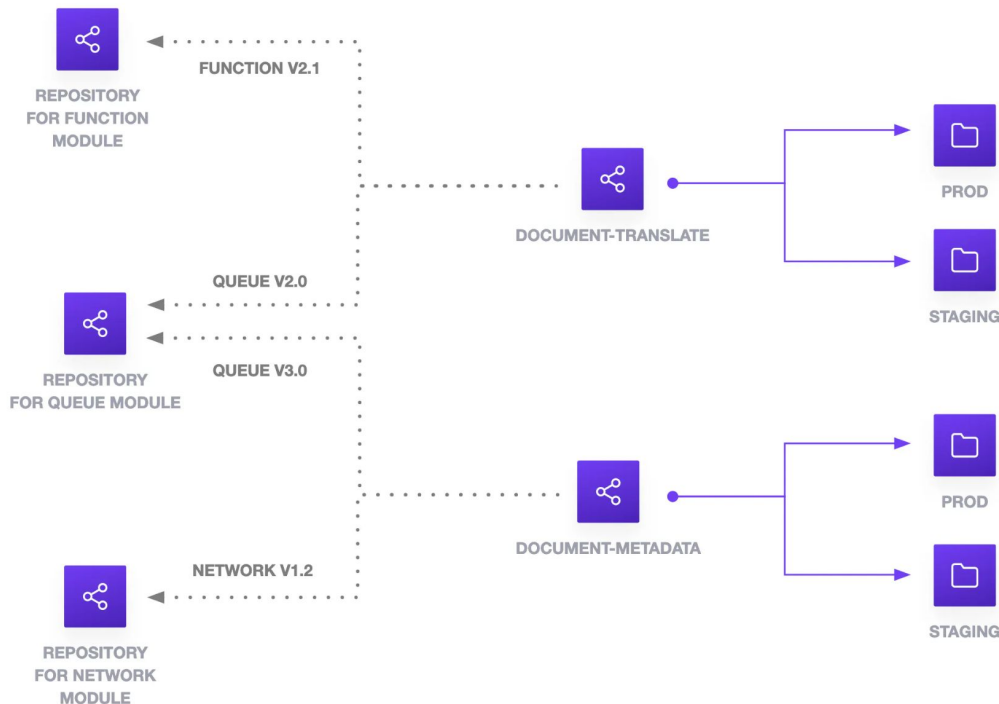
- El repositorio se convierte en la única fuente de verdad de toda la infraestructura
- Simplifica la estandarización de procesos entre equipos
- Facilita el refactor de código

- Contras

- Modificar código puede afectar a muchos componentes y puede dificultar las operaciones de Merge
- Obliga a tener una estrategia de versionado compleja y robusta
- Limita la escalabilidad del pipeline de CI/CD
 - Un cambio en un módulo obliga a verificar todo el repositorio
- Limita el control del acceso a ciertas partes de la infraestructura
 - Dificulta la implantación de políticas de Mínimo privilegio

GitOps - Repositorios

- Multi Repos
 - Separamos módulos de una infraestructura compleja en múltiples repositorios



- Cada módulo tiene su repositorio propio
- Cada dominio de negocio o producto tiene su repositorio
 - módulos como dependencias
 - Definen los diferentes entornos

GitOps - Repositorios - Multi Repos

- Pros
 - Podemos aplicar un protocolo de versionado más simple
 - Permite mayor control del acceso a los módulos de nuestra infraestructura
 - Reduce el acoplamiento
- Contras
 - No tenemos una visión global
 - Dificulta los procesos de refactor
 - Tiempo de inicialización incrementado en caso de tener muchos módulos

GitOps - Lab - Terraform + CI/CD

- Pipeline de infraestructura
- <https://github.com/upcschool-cloud-arch/contenido/tree/main/19-gitops-cicd/labs/lab30-infra-pipeline>

Recursos adicionales

<https://developer.hashicorp.com/terraform/cloud-docs/recommended-practices>