

Integración Continua(CI)

Curso de Ingeniería de Software
Primer Semestre 2022



UNIVERSIDAD DE
COSTA RICA

SR-CIE

Carrera de
Informática Empresarial
Sedes Regionales

- Integración Continua (CI)
- Github Actions
- Demo

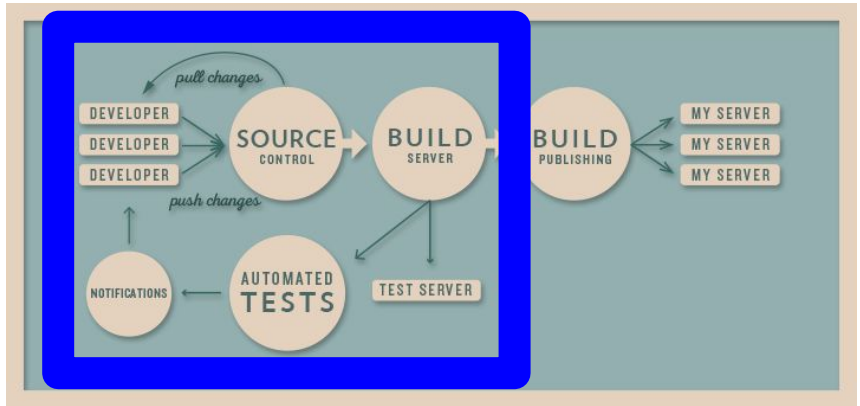
Integración Continua (CI)

Integración Continua (CI)

- La integración continua es el proceso de aplicar técnicas de **revisión** de manera **frecuente** y en pasos “cortos”.
- Se intenta mejorar la calidad del software y **reducir los tiempos de entrega del producto**.

Tipos de Testing (los más comunes)

■

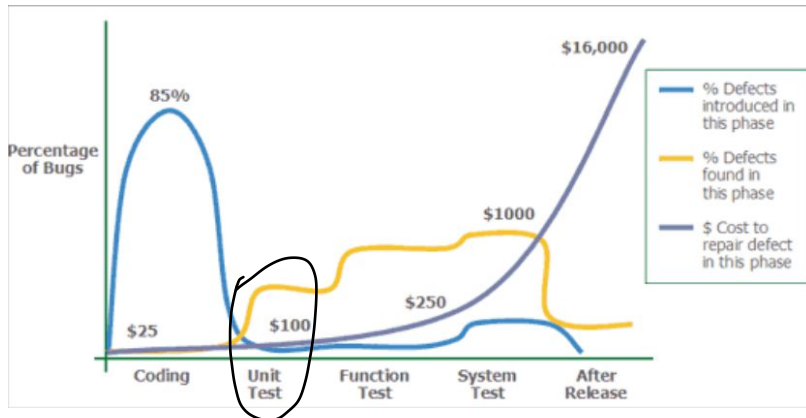


- La CI nace con el paradigma de Programación Extrema (Extreme Programming)
- Fowler and Beck escribieron acerca de este tema en 1999.
- *“Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly.”*

- Martin Fowler

En otras palabras

- Tratamos de fallar lo más rápido posible.
- Lo más tarde que se detecta un error, lo más costoso de recuperarlo



Evolución

1999

Primero todos los test se corrían localmente (el dev era responsable)

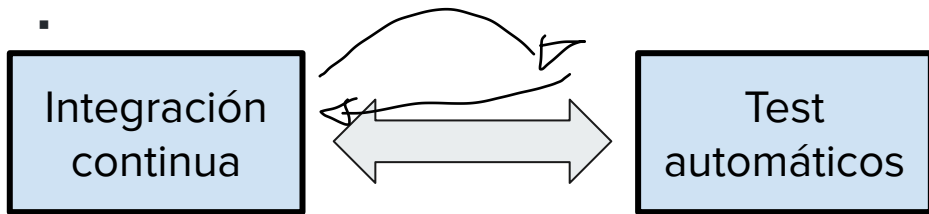
2010

Después se corría en un Jenkins o algo con un click / periódicamente

2010+

GIT notifica al pipeline tool y se corre con cada push

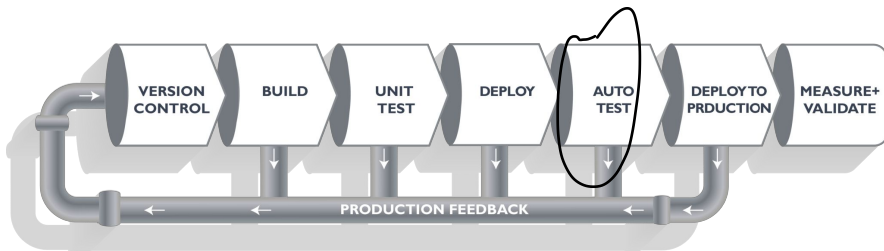
- Según la definición de Kent Beck y Martin Fowler, no podemos hablar de Integración Continua si no tenemos test automáticos. Independientemente de si corren en el pipeline o los ejecuta cada uno en su máquina antes de subir el código



-
- Igualmente hoy día se utiliza el término CI o CI/CD para referirse al pipeline de tareas automáticas.

Pipeline CI/CD

- Un Pipeline de CI/CD es la espina dorsal de un ambiente de DevOps moderno. Es el puente entre desarrollo y operaciones automatizando el build, testing y deployment de aplicaciones.



<https://dzone.com/articles/learn-how-to-setup-a-cicd-pipeline-from-scratch>

Ambiente de Integración Continua

Existen muchas aplicaciones que nos permiten configurar un ambiente integración continua.

De los más conocidos:

- Jenkins
- Codeship.io
- Gitlab-CI
- Travis-CI
- Shippable
- Circle-CI
- Y más...



Travis CI



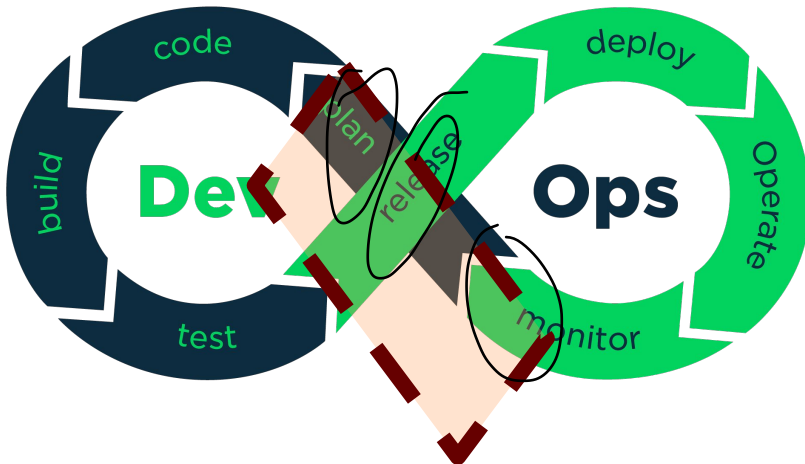
CI/CD

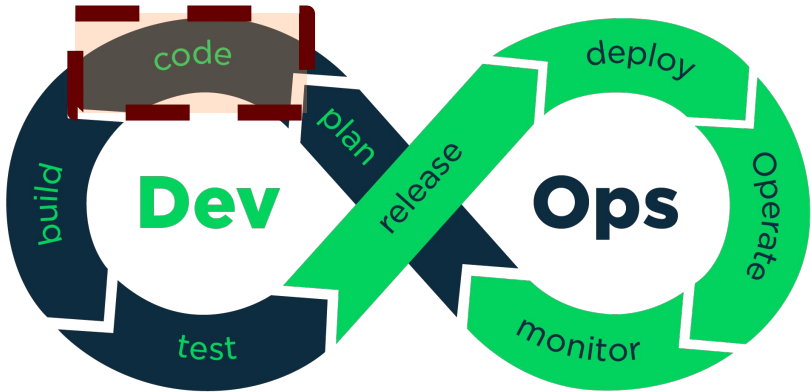


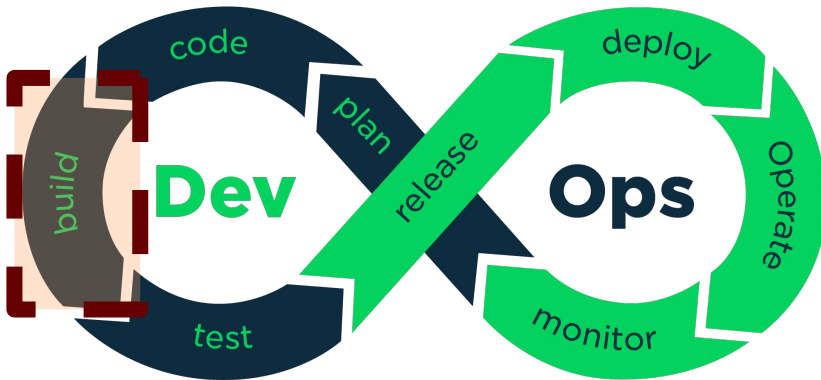
circleci

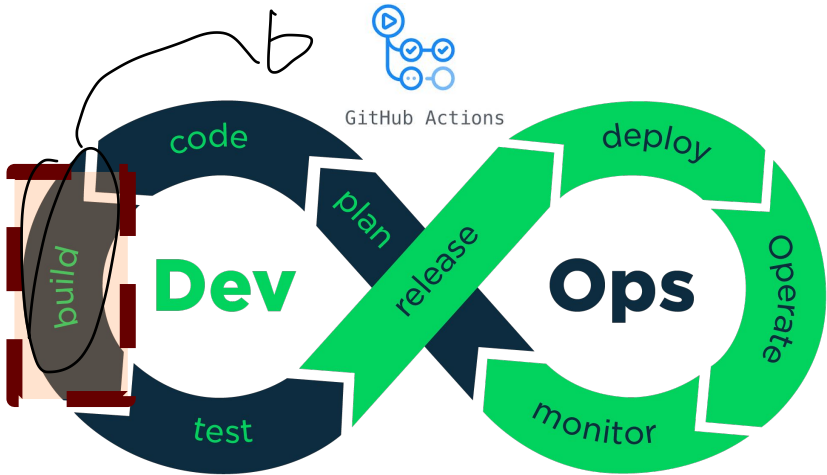


Github Actions









- Versionado
 - Colaborativo
 - Versiones
 - Merging
 - Todo en código
 - documentado?
 - Automatización
-

- YAML:
- YAML
- Ain't
- Markup
- Language
-

`%YAML 1.2`

`YAML: YAML Ain't Markup Language`

`What It Is: YAML is a human friendly data serialization standard for all programming languages.`

`YAML Resources:`

`YAML 1.2 (3rd Edition): http://yaml.org/spec/1.2/spec.html`

`YAML 1.1 (2nd Edition): http://yaml.org/spec/1.1/`

`YAML 1.0 (1st Edition): http://yaml.org/spec/1.0/`

`YAML Issues Page: https://github.com/yaml/yaml/issues`

`YAML Mailing List: yaml-core@lists.sourceforge.net`

`YAML IRC Channel: "#yaml on irc.freenode.net"`

`YAML Cookbook (Ruby): http://yaml4r.sourceforge.net/cookbook/ (local)`

`YAML Reference Parser: http://ben-kiki.org/ypaste/`

`Projects:`

`C/C++ Libraries:`

`- libyaml # "C" Fast YAML 1.1`

`- Syck # (dated) "C" YAML 1.0`

`- yaml-cpp # C++ YAML 1.2 implementation`

`Ruby:`

`- psych # libyaml wrapper (in Ruby core for 1.9.2)`

`- RbYaml # YAML 1.1 (PyYAML Port)`

`- yaml4r # YAML 1.0, standard library syck binding`

`Python:`

`- PyYAML # YAML 1.1, pure python and libyaml binding`

`- ruamel.yaml # YAML 1.2, update of PyYAML with round-tripping of comments`

`- PySyck # YAML 1.0, syck binding`

Demo

Estructura del archivo: Name & Triggers

name: Laravel

Nombre

on:

push:

Cuando

branches: [master]

Que branches

pull_request:

branches: [master]

Estructura del archivo: Jobs

jobs:

laravel-tests:

Nombre del Job

runs-on: ubuntu-latest


Donde va a correr

steps:

- uses: actions/checkout@v2

Usar acciones predeterminadas
(checkout del branch)

- name: Copy .env

run: php -r "file_exists('.env') || copy('_.env.example',
'.env');" 

Acciones custom

Estructura del archivo: variables de entorno

- `name: Execute tests (Unit and Feature tests) via PHPUnit`

`env:`

`DB_CONNECTION: sqlite`

`DB_DATABASE: database/database.sqlite`

`run: vendor/bin/phpunit`

Variables de ambiente del container

- [Testing en Laravel](#)
- [PHP Unit Asserts](#)
- [Faker](#)
- [HTTP Tests Assertions](#)
- [Travis CI](#)
- [Continuous Integration](#)