DevOps:
Introducción y procesos para desarrollo

Integración Continua y Entrega Continua

CI / CD

Integración Continua (CI)

Es la práctica de automatizar la integración de los cambios de código de múltiples contribuciones hacia un proyecto individual.

Herramienta básicas: control de versión de código fuente (repositorio).

Entrega Continua (CD)

Es una extensión de la integración continua que asegura que se puedan liberar nuevos cambios en el código de una manera sostenida.

Clave: automatización

Entrega Continua (CD)

1. Desarrollo

1. Control de calidad del código

3. QA (calidad): automatizada / manual

3. Liberación (release)

Desarrollo

Procesos de DevOps para Desarrollo

1. Cumplimiento de estándares de código y estructuración de los paquetes.
2. Inclusión de librerías y dependencias.
3. Disponibilidad de ambientes similares a producción para desarrollo.

Estándares de código

- Python: PEP https://www.python.org/dev/peps
- C#: Coding Conventions https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions
- PHP: PSRs https://www.php-fig.org/psr/

SemVer (Versionamiento)

semver.org

Versión x.y.z

z = Hotfixes

y = Nuevas funcionalidades

x = Rompimiento de compatibilidad de requisitos

Construcción de artefactos

1. Servidor de construcción (Jenkins, GitLab CI, etc)
2. Repositorio de artefactos
3. Manejadores de paquetes (apt, composer, npm, etc)

Sistemas de construcción

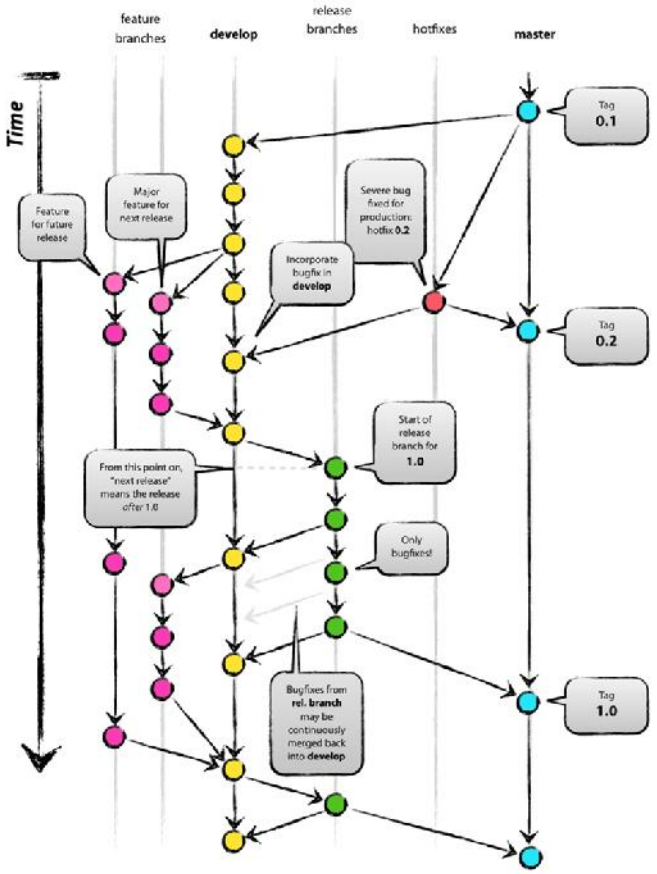
- Java: Maven, Gradle, Ant
- C/C++: Make
- Javascript: Gulp
- Ruby: Rake
- PHP: Phing
- Shell (bash, zsh, etc)

Manejo de repositorio git

- Modelo de manejo de ramas: https://nvie.com/posts/a-successful-git-branching-model/

- 1 rama master
- 1 rama develop
- n ramas feature hacia develop
- n ramas hotfix hacia master y develop
- Uso de merge requests

Modelo de manejo de ramas git



Tips para manejo de git y construcciones

1. Uso de directorios “dist” para construcción de artefactos
2. Uso de directorios “src” para manejo de código a construir
3. Exclusiones en .gitignore para configuraciones y artefactos

Software de integración continua (CI)

- Jenkins
- Travis
- GitLab CI

DevOps:
procesos para análisis de estándares y control de calidad de código

Bases del Quality Assurance

Concepto de pruebas

- En ciclos de liberación rápida, asegurar que lo que se entrega está libre de errores es vital.
- Debe establecerse un sistema que permita:
  - Cómo hacer las pruebas manuales más fácilmente y más libres de errores.
  - Involucrar varios tipos de sistemas de pruebas.
  - Automatizar las pruebas.
- Nota importante: el software de pruebas es un software en sí mismo y puede contener también errores.

Desarrollo orientado a pruebas (TDD)

Test-Driven Development

- Popular en cortos ciclos de desarrollo
- Enfoque en crear funcionalidades basadas en “asserts” (pruebas)
- Unidades pequeñas de desarrollo que puedan ser creadas a partir de las pruebas definidas.

Especificación de pruebas

I WANT TO GET VIA WEB SERVICES AN EXISTING CONTACT WITH ID=1 AT CONTACTS USING REST

- I am http authenticated 'service_user'," '123456'
- I have http header 'Content-Type'," 'application/x-www-form-urlencoded'
- I send get 'index.php'," ['option' => 'contact'," 'api' => 'Hal'," 'id' => 1]"
- I see response code is "200"
- I see response is json ""
- I see response contains 'id":"1'

Ejemplo de escenario de prueba

10 lines (10 sloc) | 400 Bytes

```php
<?php
$I = new ApiTester($scenario);
$I->wantTo('get a existing Contact with id=1 at com_contacts');
//$I->amHttpAuthenticated('service_user', '123456');
//$I->haveHttpHeader('Content-Type', 'application/x-www-form-urlencoded');
$I->sendGET('index.php', ['option' => 'contact', 'api' => 'Hal', 'id' => 1]);
$I->seeResponseCodeIs(200);
$I->seeResponseIsJson();
$I->seeResponseContains('"id":"1"');
?>
```

QA: pruebas manuales

Concepto de pruebas manuales

- Es el origen de las pruebas de sistema.
- Es base poder probar las cosas al menos una vez manualmente para poder automatizar las pruebas.
- Implica el manejo de:
  - Datos de prueba, primariamente los contenidos de las bases de datos para poder obtener siempre los mismos resultados.
  - Poder hacer despliegues rápidos para poder verificar las correcciones de errores.

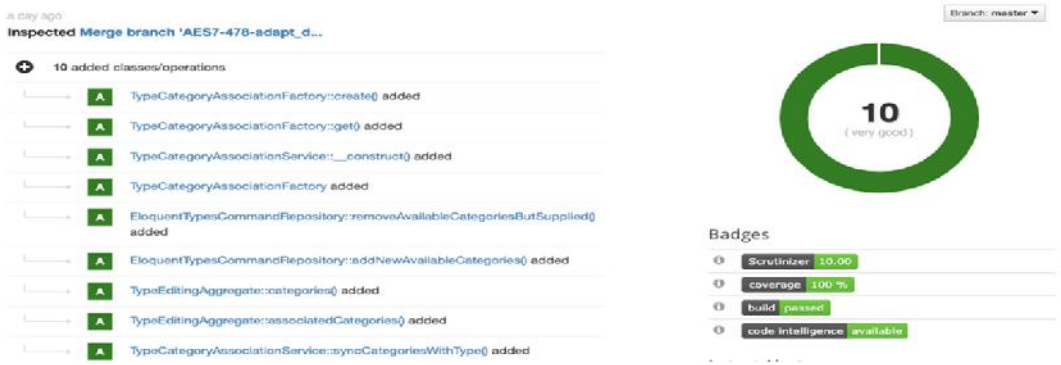
Pruebas unitarias y de integración

Definición de pruebas unitarias

- Son pruebas “de caja negra” que se hacen a las funciones individuales de código.
- Son definidas, creadas y probadas por el programador, quien define qué hace cada función y cuál debe ser su resultado.
- Es clave: escribir funciones bien específicas que arrojen resultados predecibles.

Cobertura de pruebas (Test Coverage)

- Indica el porcentaje de métodos que poseen una prueba unitaria.
- El objetivo es acercarlo al 100%.
- No especifica de ninguna manera la calidad de las pruebas.



Sistemas de pruebas unitarias

- Java: JUnit
- Python: Unittest
- PHP: PHPUnit

Ejemplo de prueba unitaria

```php
public function testListPaginatedResultsLoadSuccessfullyWithDefaults(): void
{
    // Configures returned Collection.
    $collection = $this->mock( class: PaginatedCollection::class);
    $collection->shouldReceive( ...methodNames: 'total')
        ->once()
        ->andReturn( ...args: 2);

    // Configures data retrieval.
    $this->items->shouldReceive( ...methodNames: 'listPaginated')
        ->once()
        ->andReturn($collection);
```

Pruebas de integración

- Son una extensión a las pruebas unitarias, pero que implica una especificación en los almacenes de información.
- Regularmente “integran” una base de datos o cierta persistencia para poder realizar pruebas respecto a esta.

Pruebas de sistema

Concepto general

- Las pruebas del sistema (también llamadas “de aceptación”) sirven para probar la funcionalidad del sistema como tal, tal como lo haría un usuario.
- Se utilizan por lo general herramientas que permitan automatizar el sistema (Selenium / Codeception / Cucumber / Gherkin / CodeceptJS) de tal forma que se repitan las pruebas para encontrar los resultados deseados (aceptación).

Pruebas del sistema (Cucumber)

Feature: Addition

I would like to add numbers with my pocket calculator

Scenario: Integer numbers

* I have entered 4 into the calculator
* I press add
* I have entered 2 into the calculator
* I press equal
* The result should be 6 on the screen

DevOps:
procesos para liberación y entrega

Liberación / Despliegue

Producto versus Proyecto

- Los productos llevan **liberaciones**, utilizando el versionamiento.
- Los proyectos llevan **despliegues**, utilizando ambientes específicos sobre los que serán desplegados.
- Generalmente los proyectos tienen la integración de uno o varios productos, que son desplegados con cierta configuración para dar vida al proyecto.

Integración de todo el Ciclo DevOps

Herramientas para automatización

- Jenkins: construcción de artefactos
- GitLab CI: Auto DevOps
- Netlify: Despliegue de artefactos en web