INTRODUCCIÓN AL CURSO

PRUEBAS AUTOMÁTICAS CON SELENIUM Y JAVA EN UN ENTORNO ÁGIL DE INTEGRACIÓN CONTINUA

Marco Antonio Corona Ruiz

Maestro en Ciencias de la Computación.

Tengo más de 10 año de experiencia en planeación, desarrollo e implementación de software.

Desarrollo Ágil con DevOps.

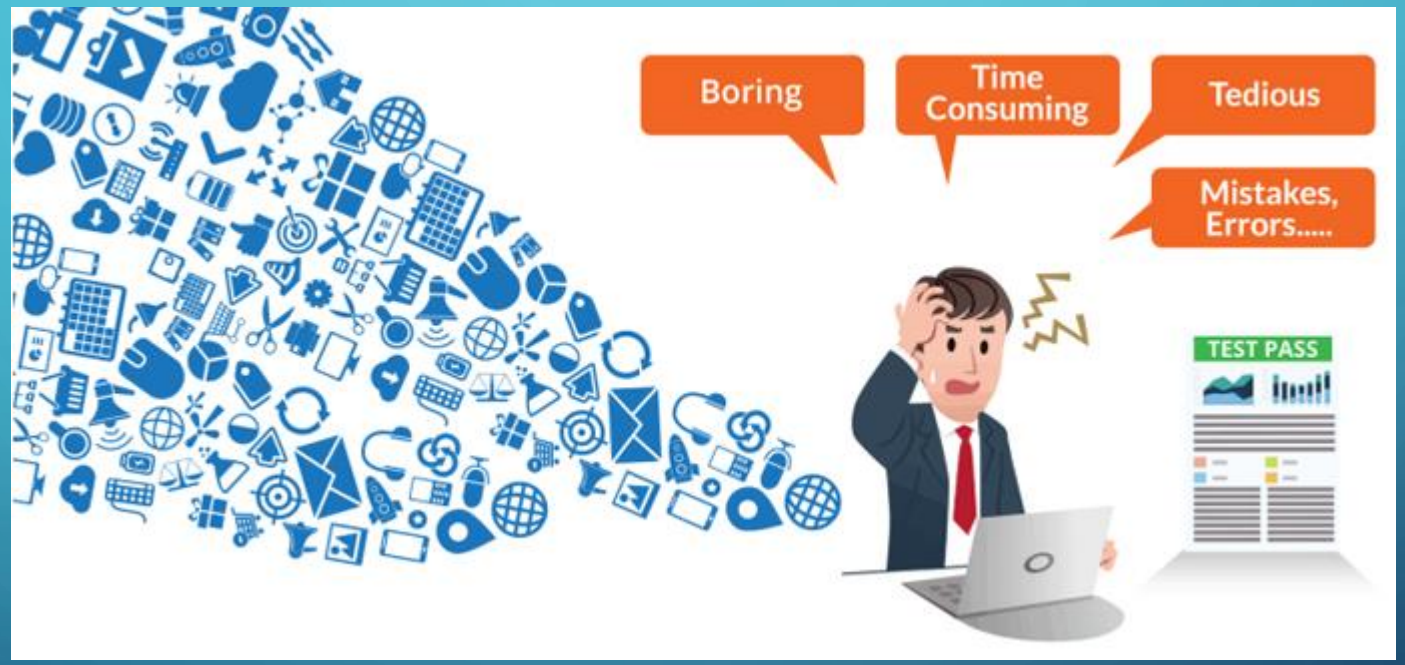
Certificaciones:

- SeU Certified Selenium Enginner (CSE)
- DevOps Essentials Professional Certificate (DEPC)
- Scrum Foundation Professinal Certificate (SFPC)

HISTORIA

- Introducción a Selenium y sus características principales.
- Pruebas automáticas de aceptación con Selenium y BDD (Behaviour-Driven Development).
- Ejecución de pruebas automáticas en un entorno de integración continua.

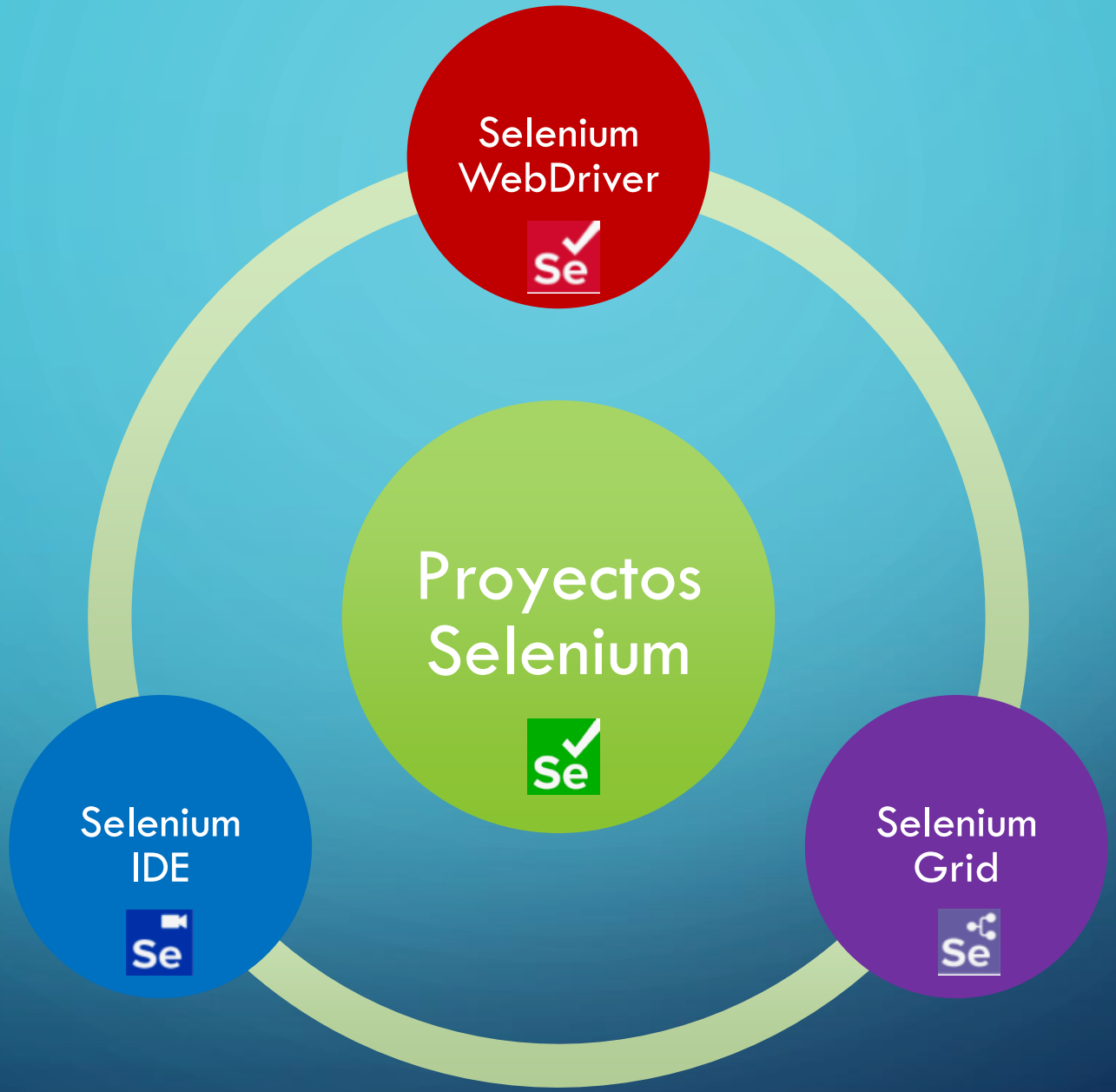
ETAPA DE PRUEBAS



https://www.mytechlogy.com/

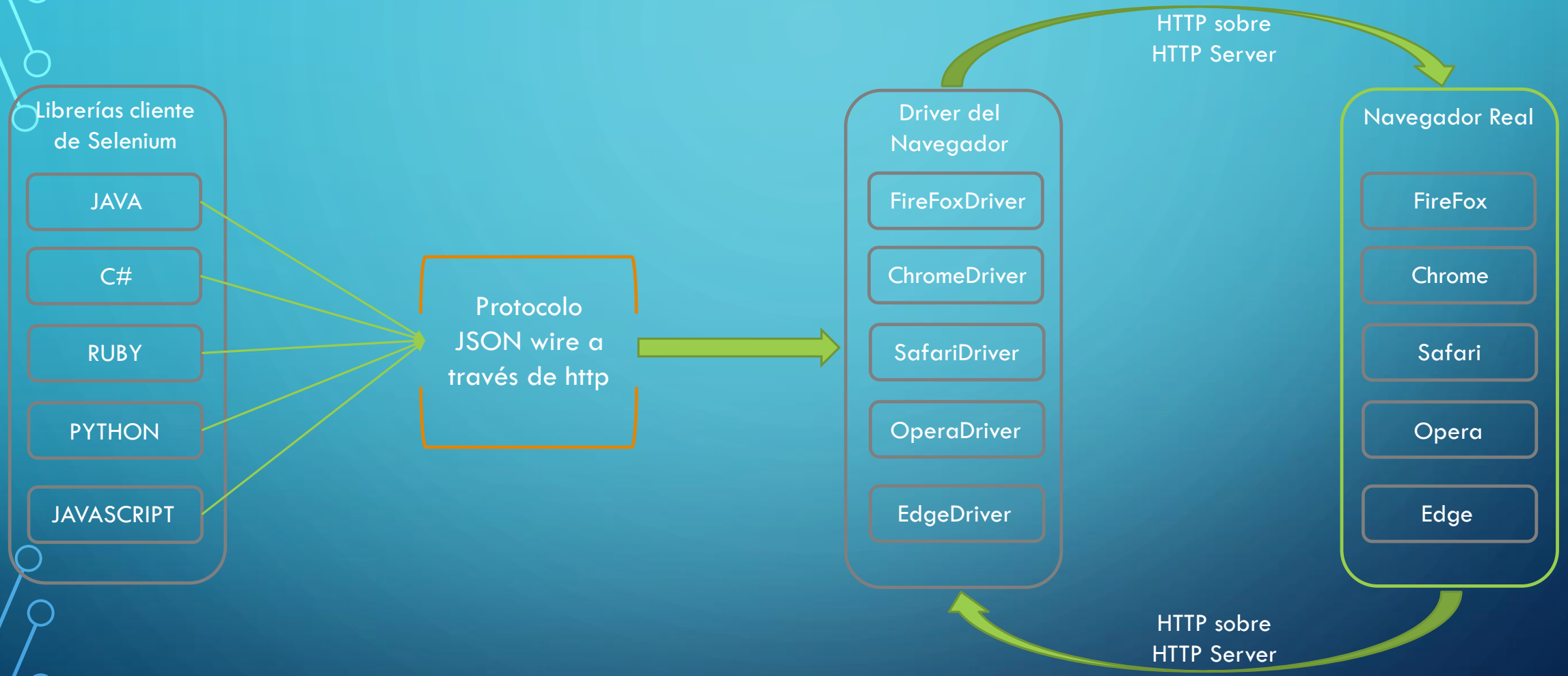
¿QUÉ ES SELENIUM?

Es una librería de código abierto utilizada para realizar pruebas automáticas en aplicaciones web en diferentes navegadores y plataformas. También puede ser utilizada en la automatización de tareas repetitivas o de regresión.



https://www.selenium.dev

ARQUITECTURA SELENIUM WEBDRIVER



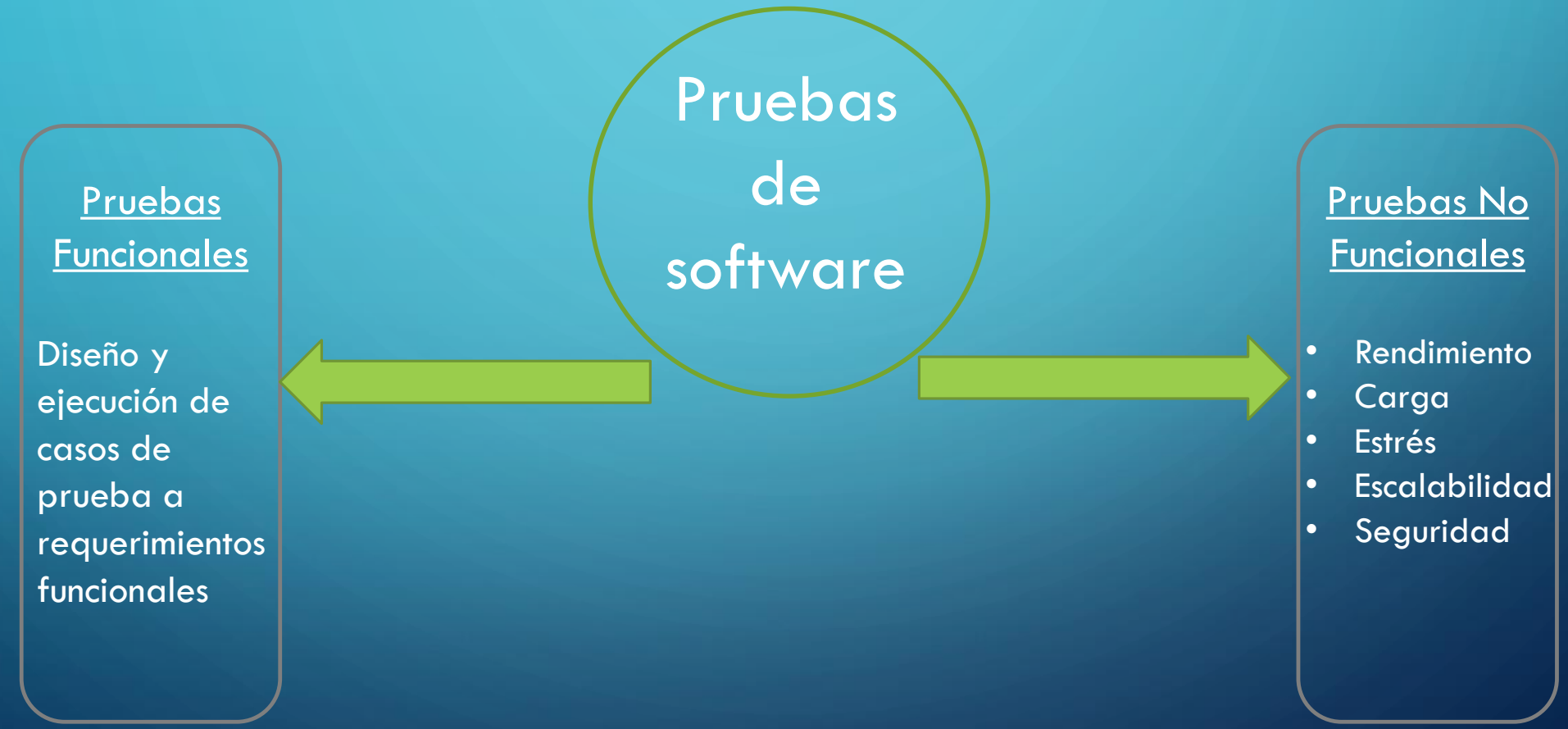
VENTAJAS DE LAS PRUEBAS CON SELENIUM

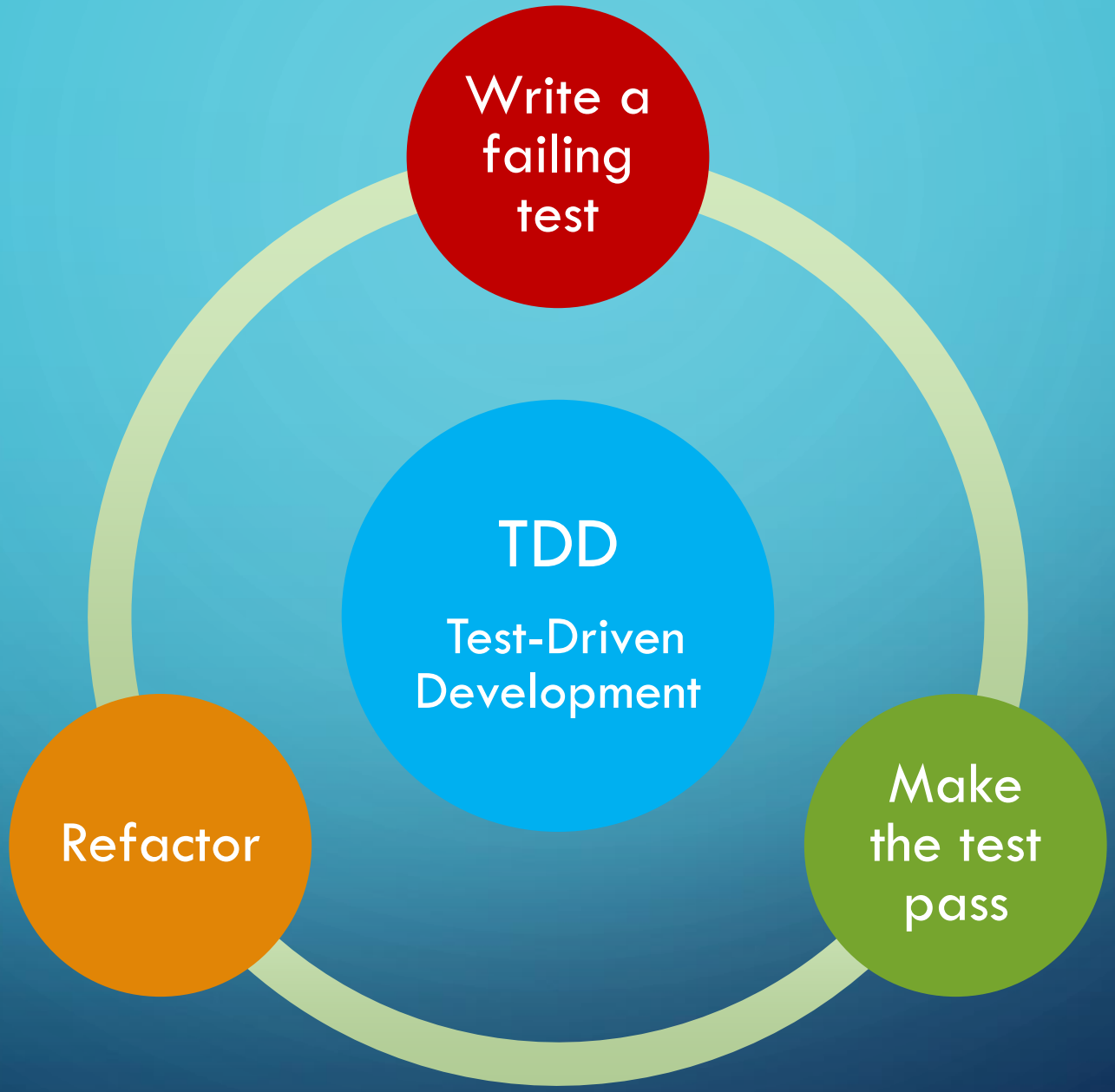
- Selenium es muy confiable, ya que ofrece resultados precisos en las pruebas.
- Ahorro de tiempo en la ejecución de pruebas en aplicaciones web.
- Es multilenguaje (Java, Python, Ruby, etc.).
- Es multinavegador (Chrome, Edge, Safari, FireFox, Opera)
- Es multiplataforma (Windows, Linux, MAC).
- Es fácil de implementar y no se requiere conocimientos profundos de la herramienta.
- Facilita la ejecución de las pruebas en aplicaciones web.

LIMITACIONES DE SELENIUM

- No puede probar aplicaciones móviles o de escritorio.
- Tiene soporte limitado para gestión de pruebas, por eso se integra con Junit y TestNG.
- Requieres conocimiento de lenguajes de programación para utilizar Selenium.

ETAPA DE PRUEBAS




Write a failing test
TDD
Test-Driven Development
Make the test pass
Refactor

BDD (BEHAVIOUR-DRIVEN DEVELOPMENT)

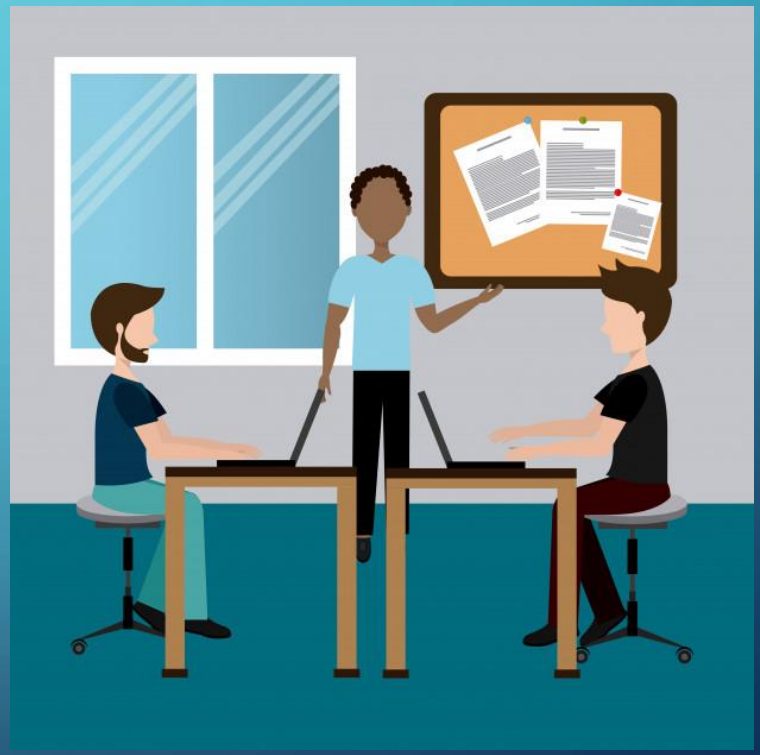
- Es un enfoque de colaboración para el desarrollo de software que cierra la brecha de comunicación entre los expertos del negocio y el equipo de TI.

- BDD ayuda a los equipos a comunicar y especificar los requerimientos con mayor precisión y a detectar defectos con anticipación para producir software con mayor calidad y fácil de mantener.

BDD – DISCOVERY WORKSHOPS

- **Equipos Agiles que implementan BDD (scrum)**
  - Expertos del negocio y los representantes de TI (desarrolladores y testers)
  - Tres amigos

- **Reuniones periódicas (inicio de cada sprint)**
  - Conversaciones sobre ejemplos concretos que indican las reglas del negocio así como los criterios aceptación.
  - Descubrir ambigüedades y malentendidos entre las personas con diferentes perspectivas.



DEFINICIÓN DE ESCENARIOS

Titulo (describe la historia)

Narrativa:
As a [role]
I want [característica]
So that [beneficio]

Acceptance Criteria: (presentada como escenario)

Scenario 1: titulo
Given [contexto]
And [algo más del contexto]…
When [evento]
Then [el resultado]
And [otro resultado] …

Scenario 2: …

Story: Llenado del formulario y despliegue de una alerta.

Como tester
Quiero probar un sitio web
Para comprobar su buen funcionamiento

Scenario 1: Una vez autenticado en el sistema debo llenar el formulario.

Given Observo el formulario y selecciono el título correcto Mr.
And Coloco mis iniciales MACR
And Coloco mi nombre MARCO
And Coloco mi apellido CORONA
Then Debería ver la alerta final
And Se cierra la aplicación

AUTOMATIZAR LOS CRITERIOS DE ACEPTACIÓN

LlenarFormulario.feature

```
Feature: Llenar el formulario con la información del usuario

  Background: Precondiciones para el scenario
    Given I navigate to the login page http://www.executeautomation.com/demosite/Login.html
    And I enter the username as user and password as Curso2020
    And I click login button

  Scenario: Una vez autenticado en el sistema debo
    llenar el formulario
    Given Observo el formulario y selecciono el titulo correcto Mr.
    And Coloco mis iniciales MACR
    And Coloco mi nombre MARCO
    And Coloco mi apellido CORONA
    Then Debería ver la alerta final
```

cucumber

```
Given("^Observo el formulario y selecciono el titulo correcto ([^\"]*)$", (String titulo) -> {
    System.out.println("titulo del usuario: " + titulo);
    UserFormPage userForm = new UserFormPage(this.base.driver);
    userForm.seleccionarTitulo(titulo);

});

And("^Coloco mis iniciales ([^\"]*)$",(String iniciales) -> {
    System.out.println("Las iniciales: " + iniciales);
    UserFormPage userForm = new UserFormPage(this.base.driver);
    userForm.insertarIniciales(iniciales);
});

And("^Coloco mi nombre ([^\"]*)$",(String nombre) -> {
    System.out.println("El nombre: " + nombre);
    UserFormPage userForm = new UserFormPage(this.base.driver);
    userForm.insertarNombre(nombre);
});

And("^Coloco mi apellido ([^\"]*)$",(String apellido) -> {
    System.out.println("El apellido: " + apellido);
    UserFormPage userForm = new UserFormPage(this.base.driver);
    userForm.insertarApellido(apellido);
    userForm.ClickBoton();
});

Then("^Debería ver la alerta final$",() -> {
    System.out.println("Se mostro la alerta");
});
```

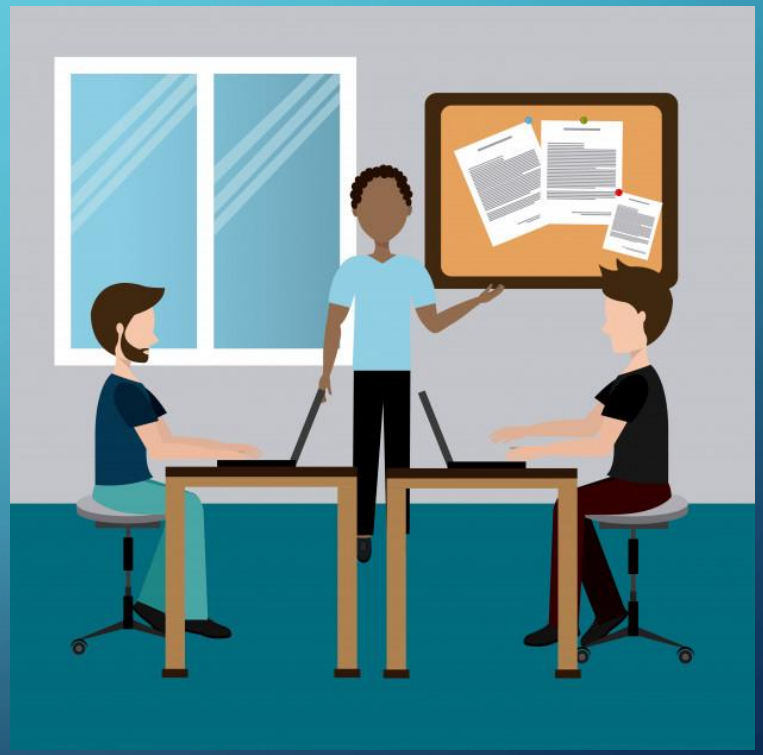
VENTAJAS DE BDD (BEHAVIOUR-DRIVEN DEVELOPMENT)

- Mejora la comunicación entre los expertos del negocio y el equipo de TI.
- Los requerimientos de usuario serán entendidos completamente por las partes involucradas y definidas en historias de usuario.
- Permite detectar defectos con anticipación para producir software con mayor calidad y fácil de mantener.
- Ahorro en el tiempo de entrega a producción.
- Pruebas automáticas de alto nivel (regresión, funcionales y de aceptación).

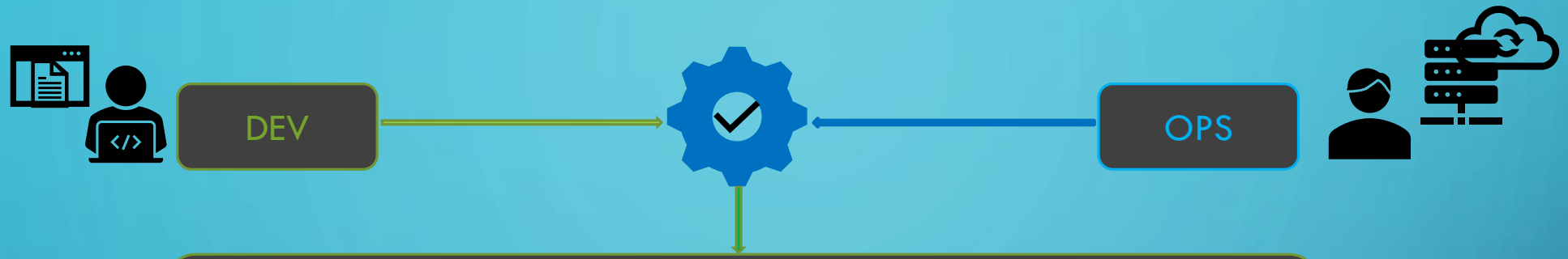
LIMITACIONES DE BDD (BEHAVIOUR-DRIVEN DEVELOPMENT)

- Requiere la especificación de los requerimientos antes del desarrollo.
- Depende de la retroalimentación constante.

SELENIUM Y BDD (BEHAVIOUR-DRIVEN DEVELOPMENT)



DEVOPS



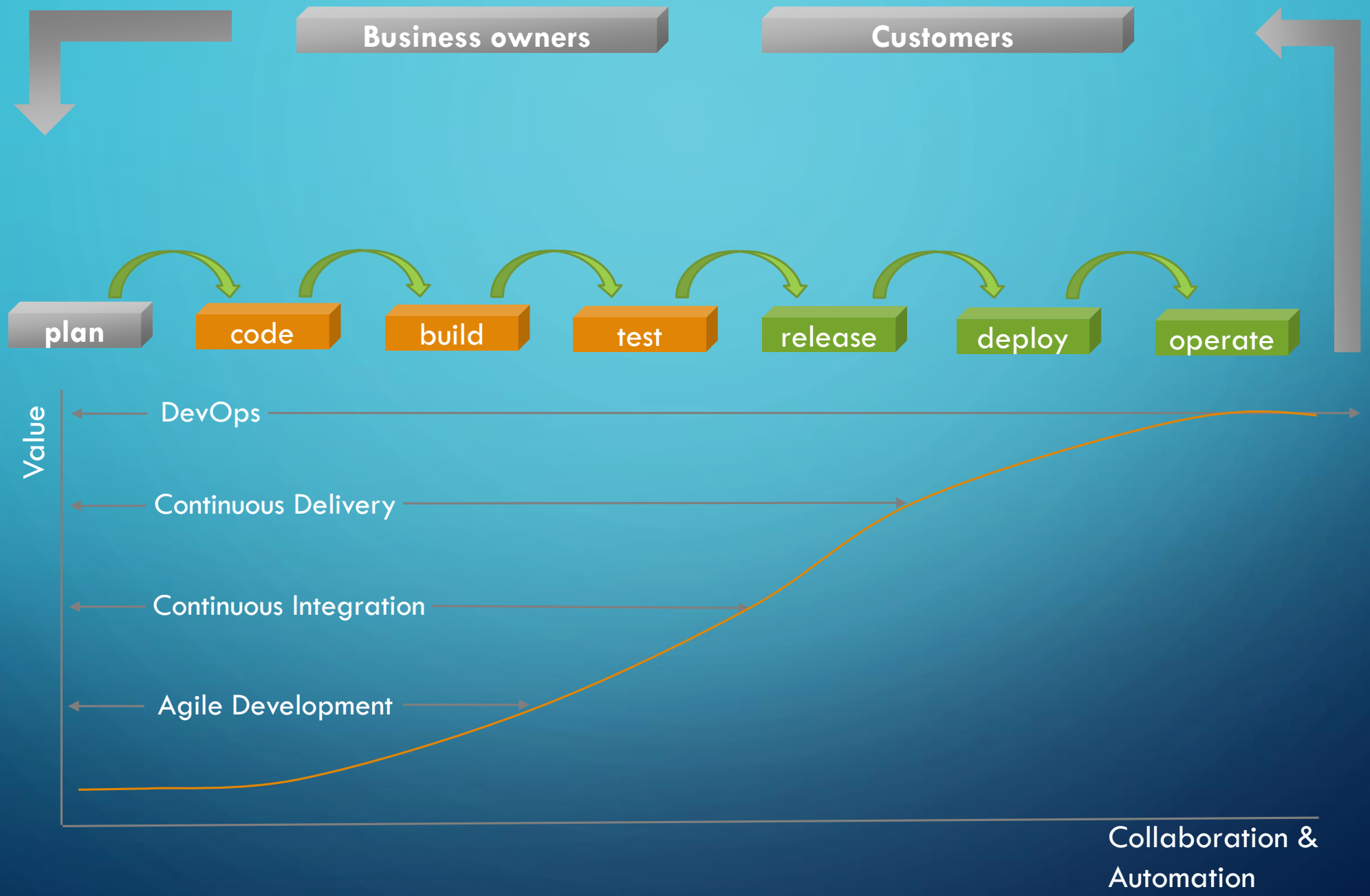
Es una cultura de buenas practicas que integra las tareas de desarrollo y operaciones, asegurando la entrega de nueva funcionalidad y la estabilidad del proyecto de forma continua y rápida

CI (Continuous Integration)

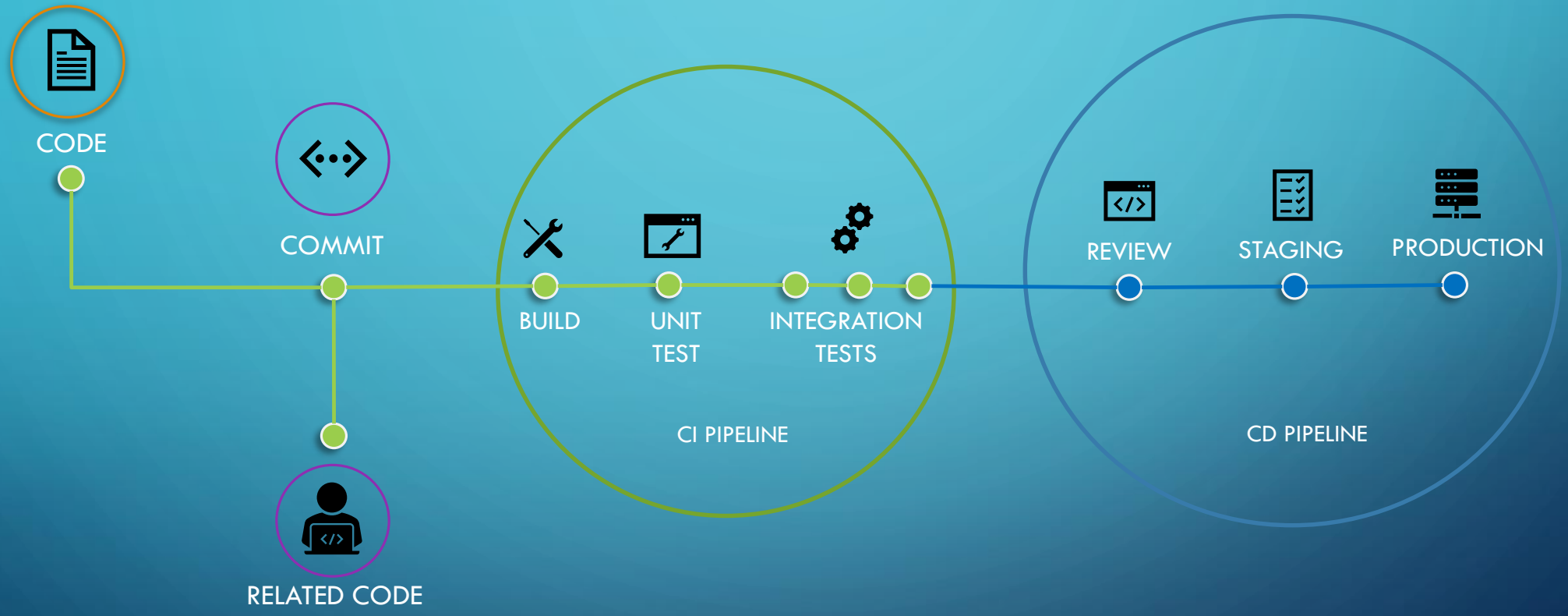
Es la práctica de integrar de manera continua los cambios de código de un proyecto para que sean probados con la mayor frecuencia posible.

CD (Continuous Delivery)

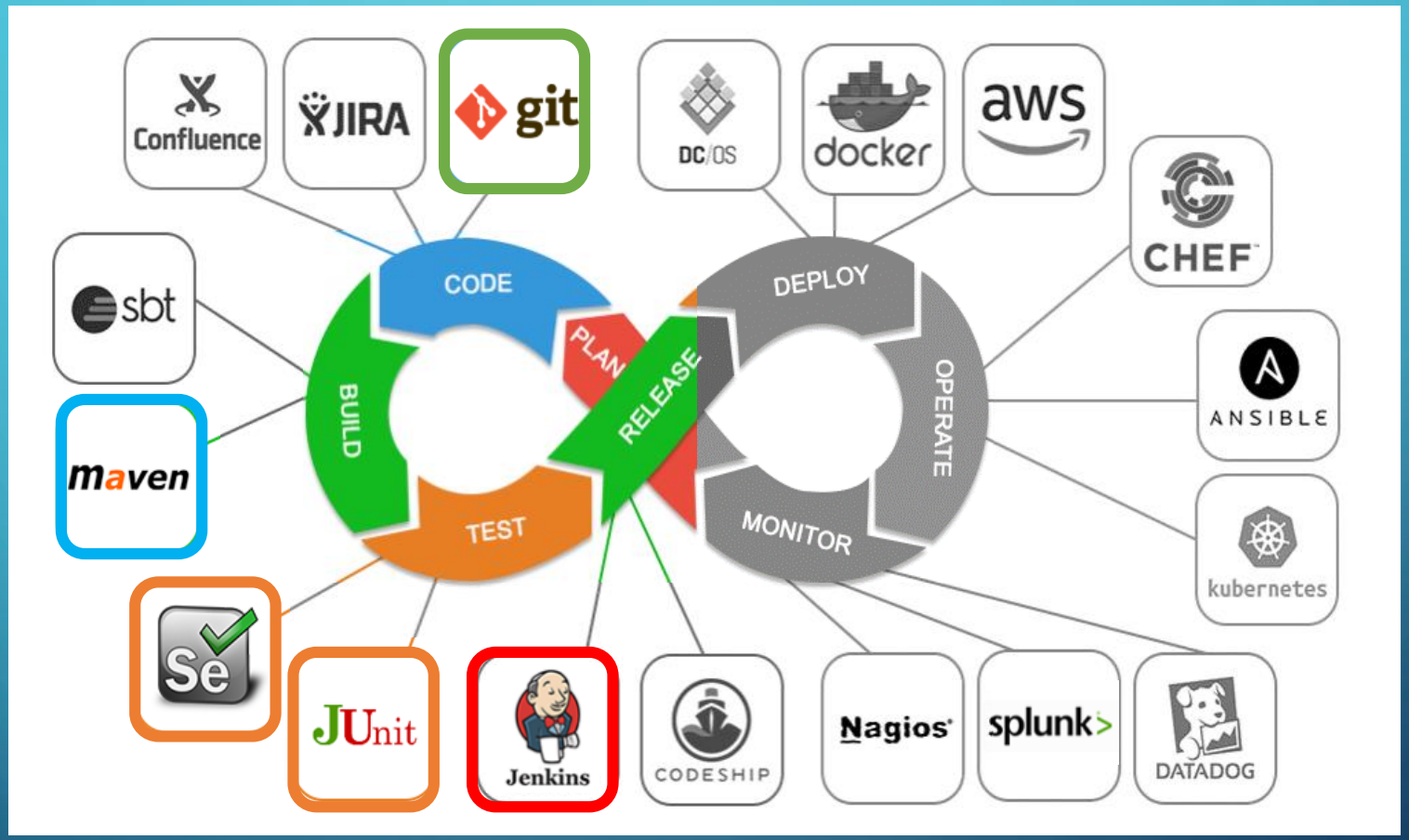
Es la implementación automática y frecuente de nuevas versiones de una aplicación en un entorno de producción.


Business owners
Customers
plan
code
build
test
release
deploy
operate
Value
DevOps
Continuous Delivery
Continuous Integration
Agile Development
Collaboration &
Automation

INTEGRACIÓN CONTINUA



HERRAMIENTAS



VENTAJAS DE CI (CONTINUOUS INTEGRATION)

- Reducir los problemas de integración
- Retroalimentación continua.
- Mejorar la visibilidad del estatus del producto de software
- Acelerar la detección de fallas
- Fomenta el trabajo minucioso
- Asegura que el código compile y funcione como fue diseñado, automatizando su proceso de pruebas (unitarias, funcionales, aceptación, calidad de código, etc.) en un solo flujo.

INCONVENIENTES DE CI (CONTINUOUS INTEGRATION)

- Cambio de la cultura organizacional
- Tiempo en la configuración inicial
- Costos de hardware

CONCLUSIONES

- Pruebas automáticas y reportes detallados sobre el estado de la ejecución de las mismas.
- Proyectos con mayor calidad y fáciles de mantener.
- Retroalimentación más completa y rápida.
- Reducción de incidentes y errores en los sistemas.

CONCLUSIONES

- Ahorro de tiempo en las entregas a producción al prevenir defectos en lugar de encontrarlos.
- Los requerimientos serán entendidos completamente por las partes involucradas.
- Entorno automatizado que asegura que el código compile y funcione como fue diseñado (pipeline).

DATOS DE CONTACTO

- Correo Electrónico: macorona2020@outlook.com
- Twitter: @macorona2020
- Presentación:

GRACIAS

DEVOPS Y SU PROPOSITO

“Valor” en software:

- Nueva Funcionalidad
- Corrección de errores
- Calidad en el servicio
- Resiliencia

Entrega Valor Continuo

DEVOPS Y SU PROPOSITO

sin DevOps

- Errores encontrados en producción
- Deployments dolorosos
- Fragilidad en la infraestructura
- Estado de constante urgencia

con DevOps

- Errores encontrados en desarrollo
- Fallos mínimos en deployments
- Infraestructura automatizada
- Monitoreo y acción proactiva
- Aprendizaje continuo

PRINCIPIOS DEVOPS

Cultura

Automatización

Medición

Colaboración (**S**haring)

Ultimate guide to CAMS Model in DevOps (opengenus.org)