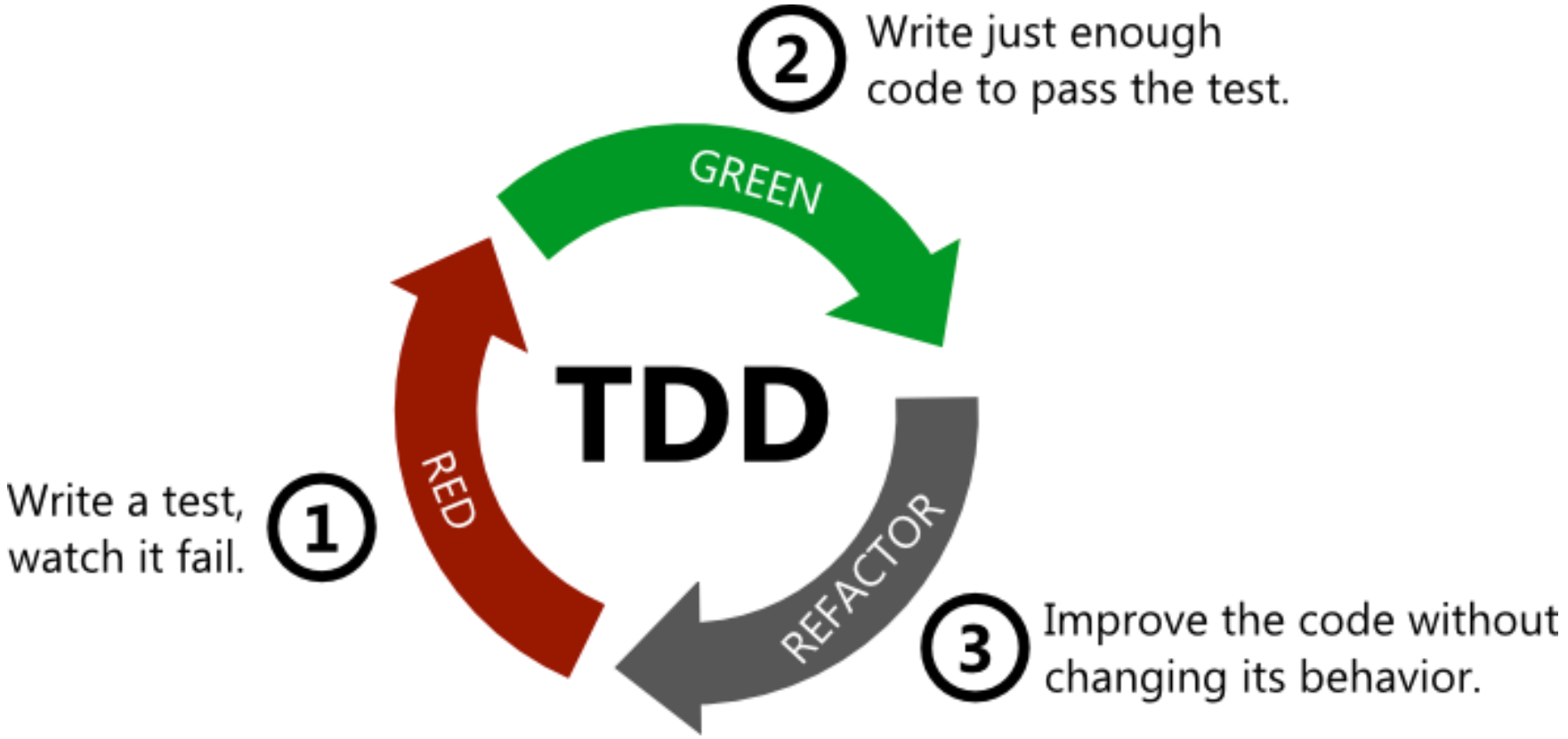
# Ciclo de Vida dos Testes no JUnit 5



# Java Continuous Integration-Delivery c. AWS e Github Actions

## Theory

# 0202 O que é Integração e Entrega Contínua?

## Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

# O que é Integração e Entrega Contínua?

São metodologias que auxiliam as equipes de DevOps, melhorando o ciclo de desenvolvimento de software;

Com elas é possível resolver os problemas causados pela integração de novos códigos e aumentar a frequência na entrega de novas funcionalidades;

É indispensável para empresas que querem acompanhar as demandas e entregar novas funcionalidades com maior rapidez.

# O que é Integração e Entrega Contínua?

*Continuous Integration* (CI) e *Continuous Delivery* (CD) são práticas complementares;

Sua principal finalidade é automatizar o processo de desenvolvimento de software, tornando-o mais seguro para suportar entregas frequentes.

# O que é *Continuous Integration* (Integração Contínua)?

Processos que permitem aos desenvolvedores realizar testes automatizados sempre que o código é enviado para o repositório de código;

Os diferentes módulos que formam a aplicação são testados. E, caso hajam erros na execução dos testes, a CI ajuda a encontra-los e corrigi-los de forma rápida, frequente e ágil;

Garante que o código desenvolvido por diferentes programadores funciona e assegura a integridade do sistema.

# O que é *Continuous Delivery* (Entrega Contínua)?

Depois da integração, as alterações feitas no código passam pela Continuous Delivery (CD) ou Entrega Contínua;

Graças à automação e pouca intervenção humana, é possível fazer entregas contínuas de novas *features*;

A CD direciona o merge do código alterado para o repositório ideal. O seu foco é fazer o release do código para o local em que deve ser armazenado;

# O que é *Continuous Delivery* (Entrega Contínua)?

A entrega é relizada para o repositório em que foram feitas as mudanças;

Fica mais fácil prevenir problemas com outras partes do sistema, sobretudo durante a implementação de aplicações mais complexas;

CD também pode se referir à Implantação Contínua, ou seja o *deploy* automático das alterações à produção;

Todas essas abordagens formam o pipeline de entrega

# CI e CD são partes complementares

*Continuous Integration* e *Continuous Delivery* são métodos que promovem um desenvolvimento e entrega mais ágeis;

É importante ressaltar que a CD só funciona se a CI for implementada adequadamente;

Ambas complementam uma mesma estratégia e são essenciais para as empresas que desejam implantar uma cultura *DevOps*.

# Benefícios da *Continuous Integration* e da *Continuous Delivery*

Além de melhorar o processo de integração,  a CI/CD ajuda a minimizar os custos de desenvolvimento, identificando os erros logo no início do ciclo de vida do software.

Trazem outros benefícios como:

- Rápida identicação de problemas;
- Melhorias de usabilidade;
- Agilidade;
- Compartilhamento da visão;
- Flexibilidade e segurança;
- Qualidade.


## 1. Developer

Developer pushes code to Github.

## 2. Github

Github hook kicks off build in Github Actions.

## 3. Github Actions

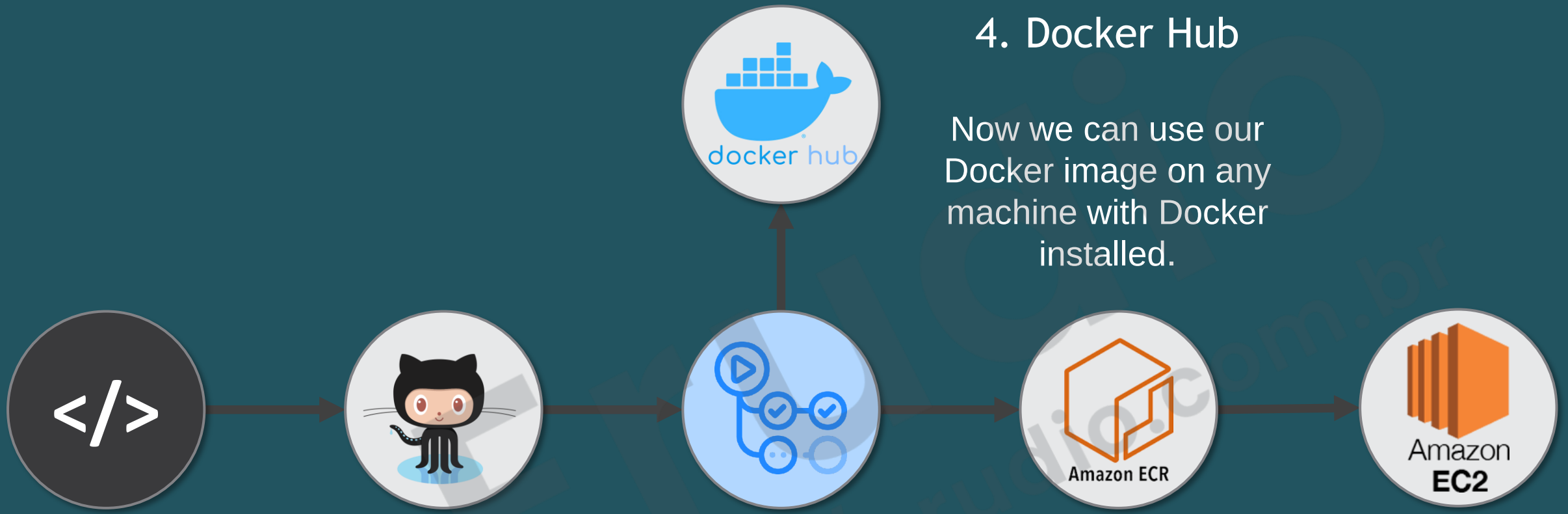
Github Actions builds Docker Images and pushes to Docker Hub.

## 4. Docker Hub

Now we can use our Docker image on any machine with Docker installed.

## 5. Amazon EC2

We can deploy our app in one AMI EC2.




## 1. Developer

Developer pushes code to Github.

## 2. Github

Github hook kicks off build in Github Actions.

## 3. Github Actions

Github Actions builds Docker Images and pushes to Docker Hub.

## 4. Docker Hub

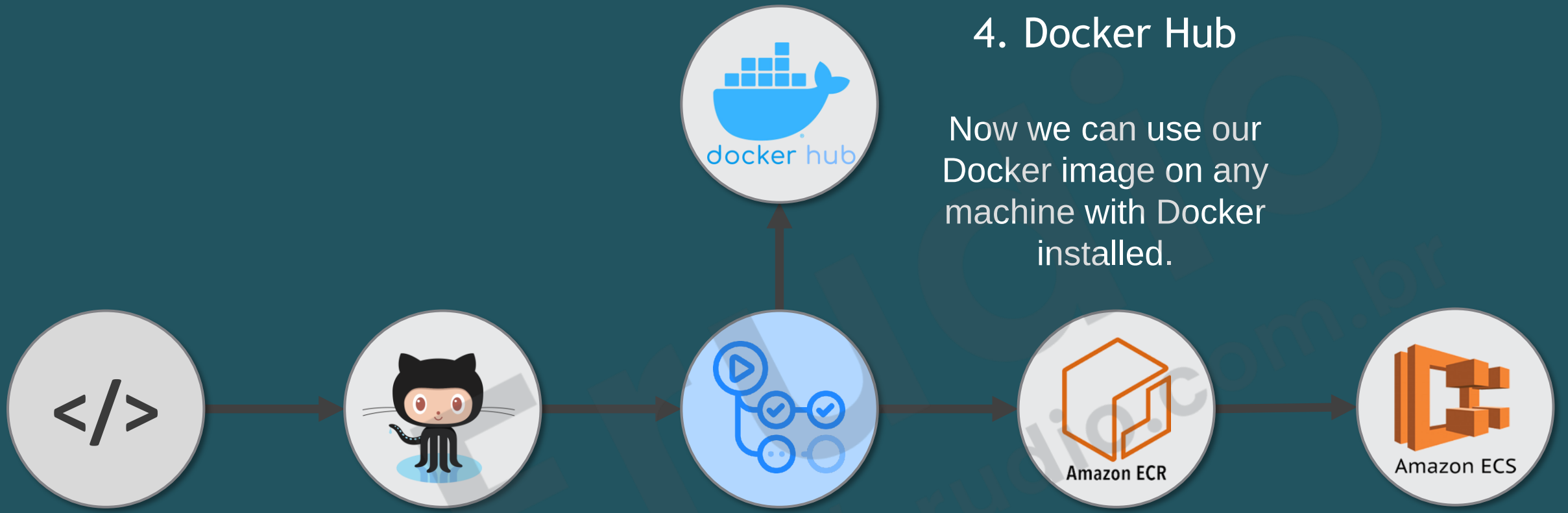
Now we can use our Docker image on any machine with Docker installed.

## 4. Amazon ECR

Now we can use our Docker image to deploy our app on Amazon AWS.

## 5. Amazon EC2

We can deploy our app in one AMI EC2.




## 1. Developer

Developer pushes code to Github.

## 2. Github

Github hook kicks off build in Github Actions.

## 3. Github Actions

Github Actions builds Docker Images and pushes to Docker Hub.

## 4. Docker Hub

Now we can use our Docker image on any machine with Docker installed.

## 4. Amazon ECR

Now we can use our Docker image to deploy our app on Amazon AWS.

## 5. Amazon ECS

We can deploy our app in one ECS Cluster.

# 0203 Componentes essenciais de uma infraestrutura CI-CD

## Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

# Componentes de uma infraestrutura CI/CD

Uma infraestrutura de Integração Contínua (CI) e Entrega Contínua (CD) envolve diversos componentes que trabalham em conjunto para automatizar o processo de desenvolvimento e entrega de software. Alguns dos componentes essenciais são:

- Controle de Versão;
- Servidor de Build;
- Ambiente de Testes;
- Ferramentas de Automação de Testes;
- Ferramentas de Implantação;
- Orquestração de Pipeline;
- Monitoramento e Logging;
- Infraestrutura como Código.

# 0204 Ferramentas e Tecnologias Comuns para CI-CD

## Leandro Costa

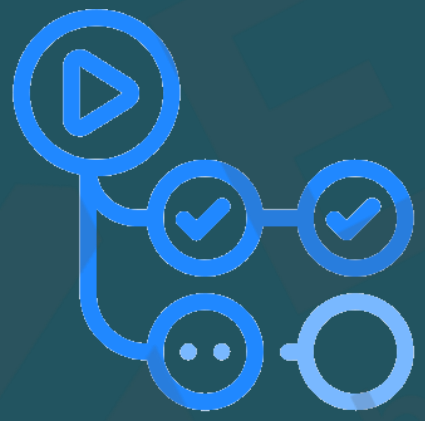
https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

# Ferramentas e Tecnologias Comuns para CI/CD

Existem, várias ferramentas e tecnologias para se trabalhar com CI/CD e elas têm sido amplamente utilizadas para automatizar os workflows de CI/CD. Algumas das mais conhecidas são:




# 0206 O que são Testes Unitários e por que Implementá-los?

## Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

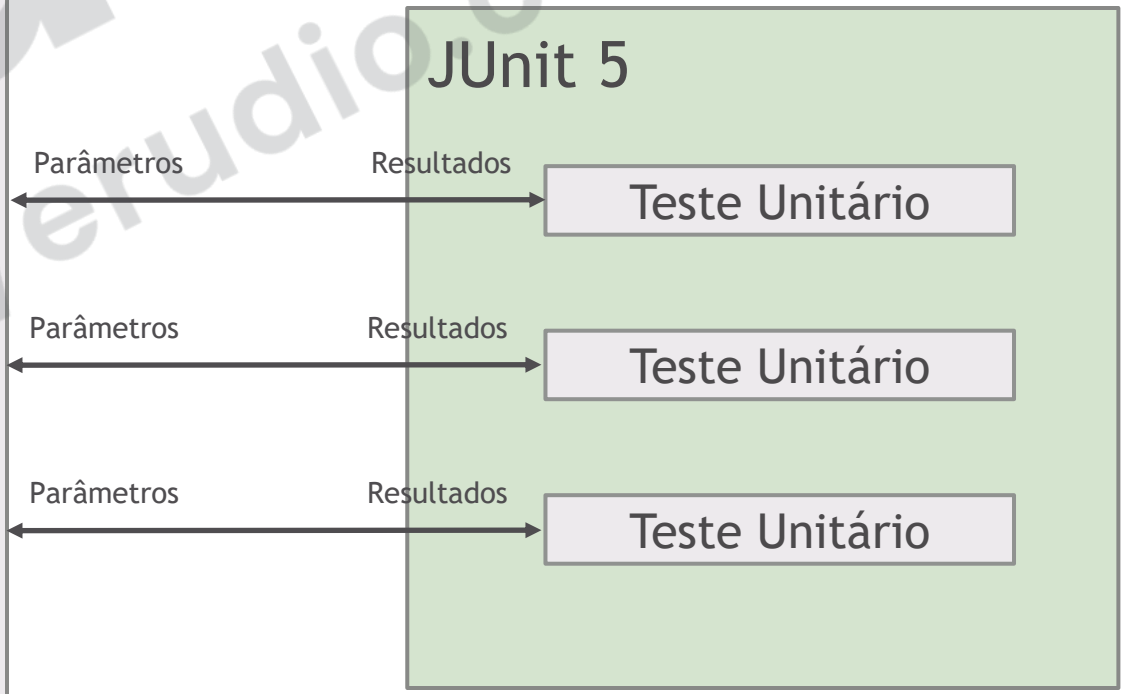
# O que são Testes Unitários?

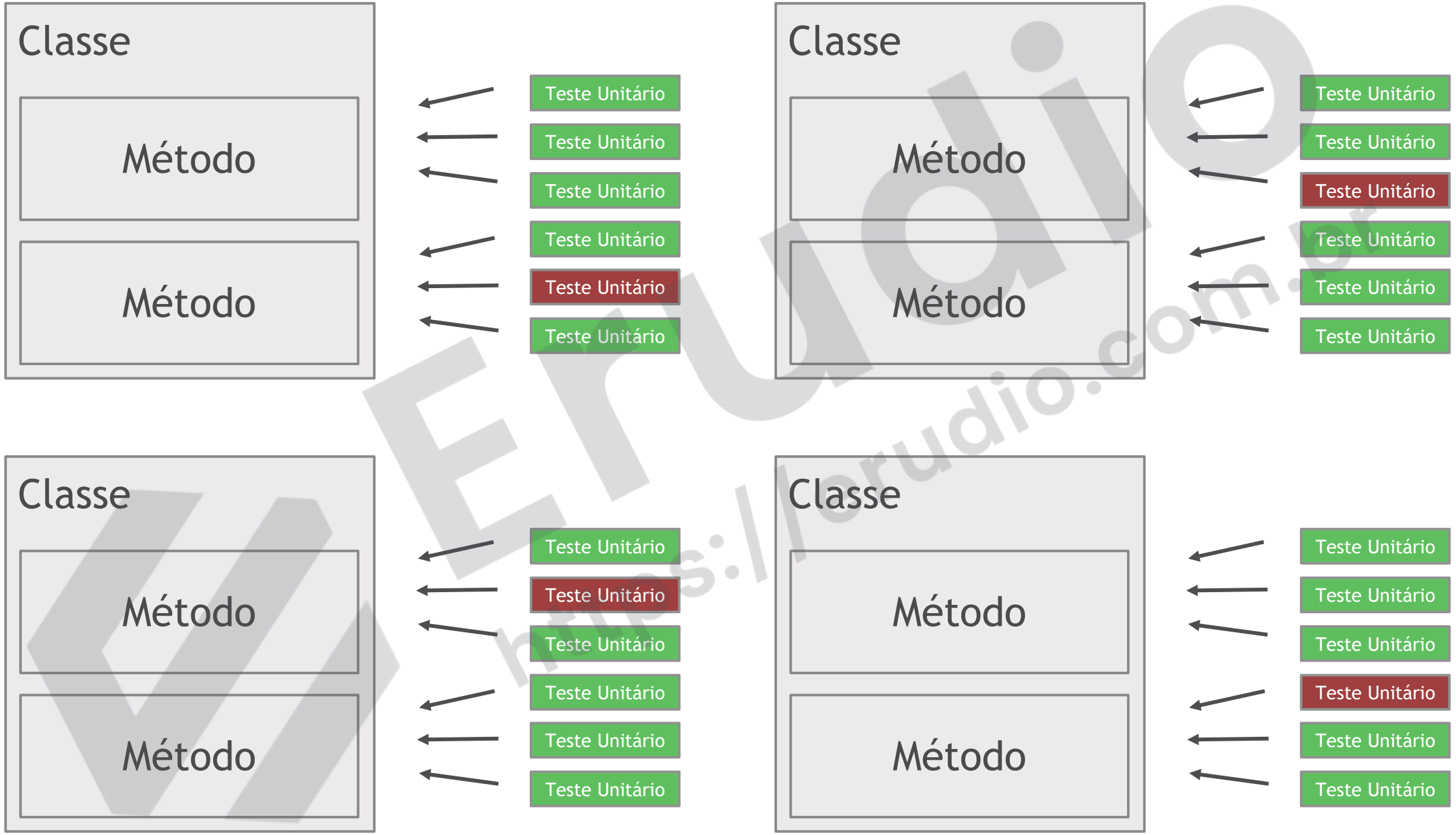
Um **teste unitário** é um pequeno método que você escreve para testar alguma parte do seu código.

```
public String generateCoupon(String couseId) {

    // Code under test

}
```




Classe
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Classe
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Classe
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Classe
Método
Método
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário
Teste Unitário

```java
public class SimpleMath {

    public Double sum(Double firstNumber, Double secondNumber)
    {
        return firstNumber + secondNumber;
    }
}
```

```java
@Test
void testSum_WhenSixDotTwoIsAddedByTwo_ShouldReturnEightDotTwo() {
    // Given / Arrange
    SimpleMath math = new SimpleMath();

    // When / Act
    Double actual = math.sum(6.2D, 2D);

    // Then / Assert
    assertEquals(8.2D, actual, () -> "6.2 + 2 did not produce 8.2!");
}
```

# Por que é tão importante implementar testes unitários?

Existem diversas vantagens em adotar essa prática:

- Garantia da qualidade do código;
- Detecção precoce de falhas;
- Facilita a manutenção e o refactory;
- Melhoria da estrutura do código;
- Documentação viva do código;
- Suporte a práticas de desenvolvimento ágil.

# 0207 Como Implementar o Pipeline CI-CD?


## Leandro Costa

https://www.erudio.com.br/ https://www.youtube.com/c/ErudioTraining

https://www.semeru.com.br/ https://hub.docker.com/u/leandrocgsi/

https://github.com/leandrocgsi/automated-tests-with-java-erudio

# Como implementar um pipeline CI/CD?

- Alinhe os processos;
- Busque entender quais são os elementos de um pipeline de CI/CD;
  - Compilação;
  - Teste;
  - Push;
  - Implantação;
  - Validação
- Limpe os ambientes entre uma implantação e outra;
- Monitore o pipeline;
- Introduza a cultura DevOps entre as equipes.