



Integração Contínua – Prática - Etapa B

Prof. Jean Carlo Rossa Hauck
Prof. Osmar de Oliveira Braz Junior
Prof. Richard Henrique de Souza

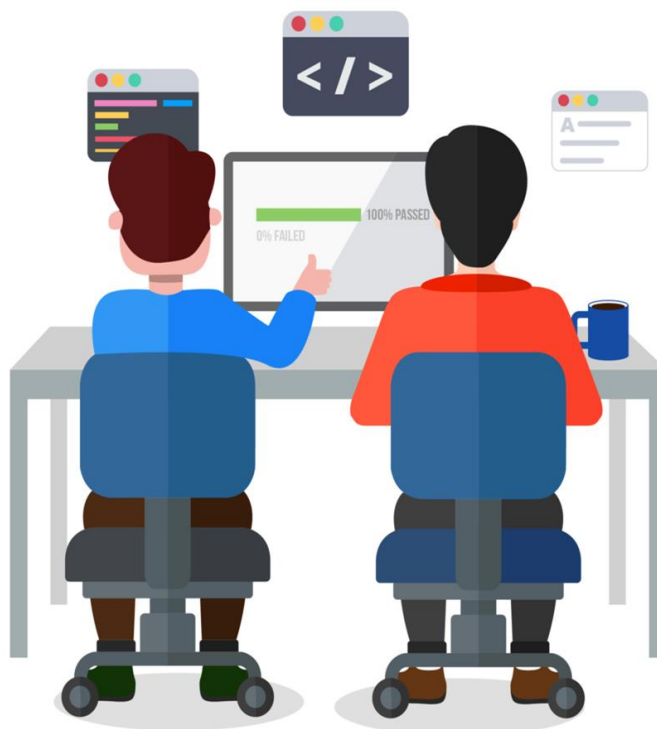
Objetivos

- Realizar um exemplo de **integração contínua** da **análise** até **cobertura** do código.
- A **integração** continua será realizada em 3 ambientes distintos com tarefas distintas.
- A **análise** irá considerar diversas métricas de qualidade de software como confiabilidade, manutibilidade, segurança, **cobertura** e duplicação de código.
- Nesta etapa criaremos a integração continua do projeto no Github Actions.

Atividade em Grupo

Para esta atividade crie grupos de 2 alunos, para desenvolver a atividade segundo ***Pair Programming***.

Navegador



Piloto

Pair Programming

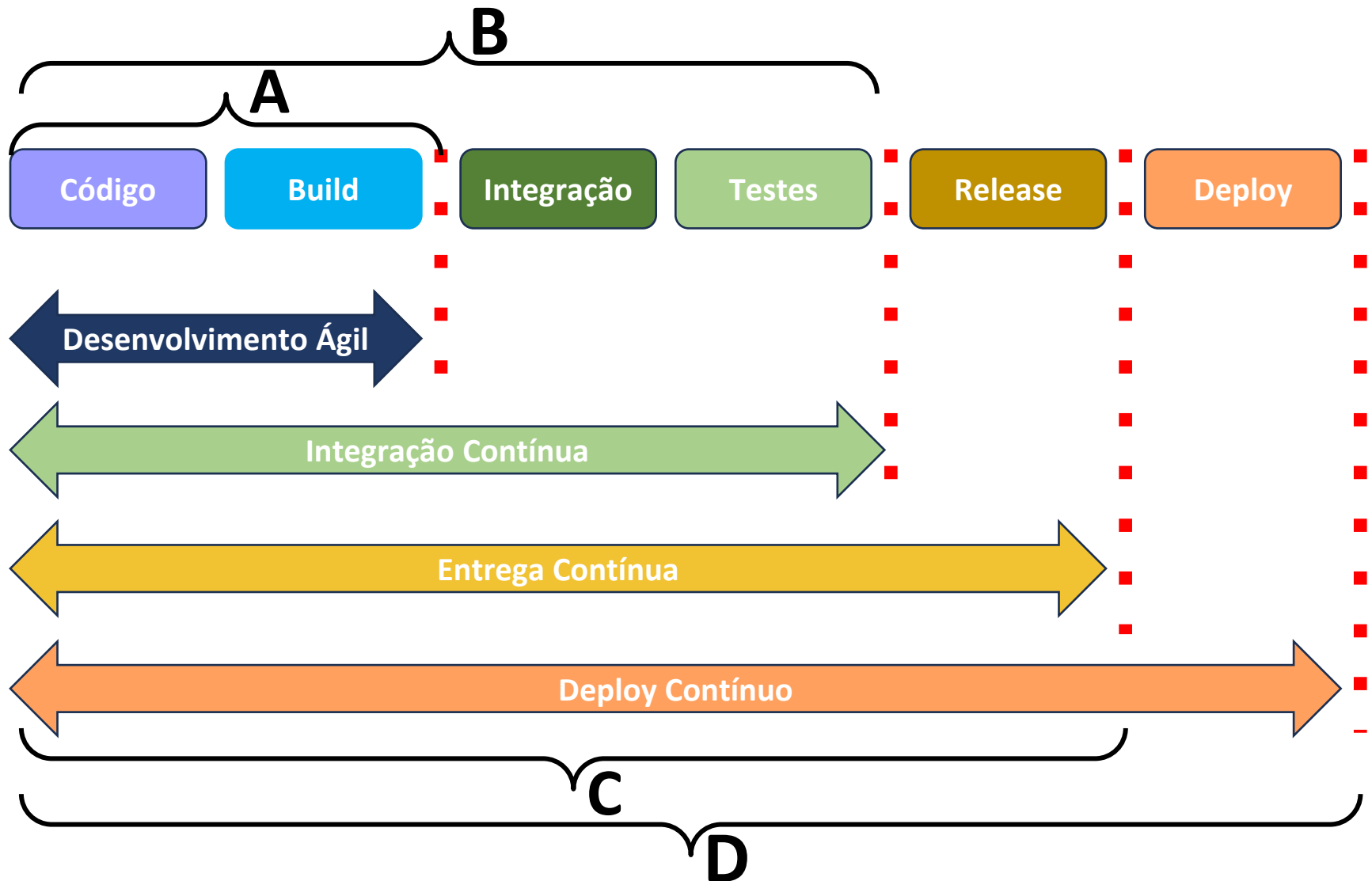
- Um é o **piloto**, responsável por escrever o código, o outro o navegador, acompanha a escrita de código e verificar se está de acordo com os **padrões do projeto** e de encontro à solução necessária.
- A intenção desta técnica é **evitar** erros de lógica, e ter um código mais confiável e melhor estruturado, utilizando-se para isso a máxima de que “**duas cabeças pensam melhor do que uma**”.

Integração, Entrega e Implantação Contínua

■ Abstração do Pipeline



Integração, Entrega e Implantação Contínua



Ferramentas utilizadas

- IDE com suporte
 - Apache Maven
 - JUnit 4
 - Github
- Github
 - Github Actions
 - Sonarcloud
 - JaCoCo

maven

JUnit



sonarcloud 

JACO
Java Code Coverage

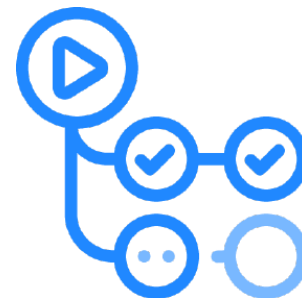
Atividades práticas

- A - Criação de Projeto e testes unitários
 - Criar projeto na IDE
 - Automatizado com Apache Maven
 - Criar testes unitários com JUnit 4
 - Armazenar projeto no Github
- B - Integração Contínua
 - Github Actions
 - JUnit 4
- C - Análise do Código
 - Sonarcloud
 - Integração com Github Actions
- D - Cobertura do código
 - Jacoco
 - Integração com Github Actions
 - Integração com Maven e Sonarcloud

B – Integração Contínua

- Github Actions
- JUnit 4

JUnit





Passo 1

Vamos criar o fluxo de trabalho para integração contínua no **Github Action** com **Java e Maven**.

<> Code Issues Pull requests **Actions** Projects Wiki Security Insights Settings

Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

Skip this and [set up a workflow yourself](#) →

Continuous integration

Java with Maven

By GitHub Actions

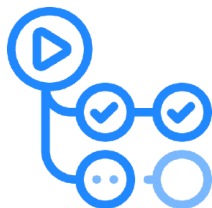
Build and test a Java project with Apache Maven.

Configure

Java

Pesquise por “**Java with Maven**”

Passo 2



Cria automaticamente o arquivo *maven.yml* na pasta */.github/workflows/* do seu repositório.

Actions

<https://github.com/actions/setup-java>

Versão java

Temurin

<https://adoptium.net/>

Nome do *workflow*.
Vamos mudar depois,
Para "Integração
continua de Java com
Maven"

Evento de de
disparo do
workflow.

Versão
do SO

Versão
do Java

Executa o tarefa do
pom.xml do projeto.

```
name: Java CI with Maven

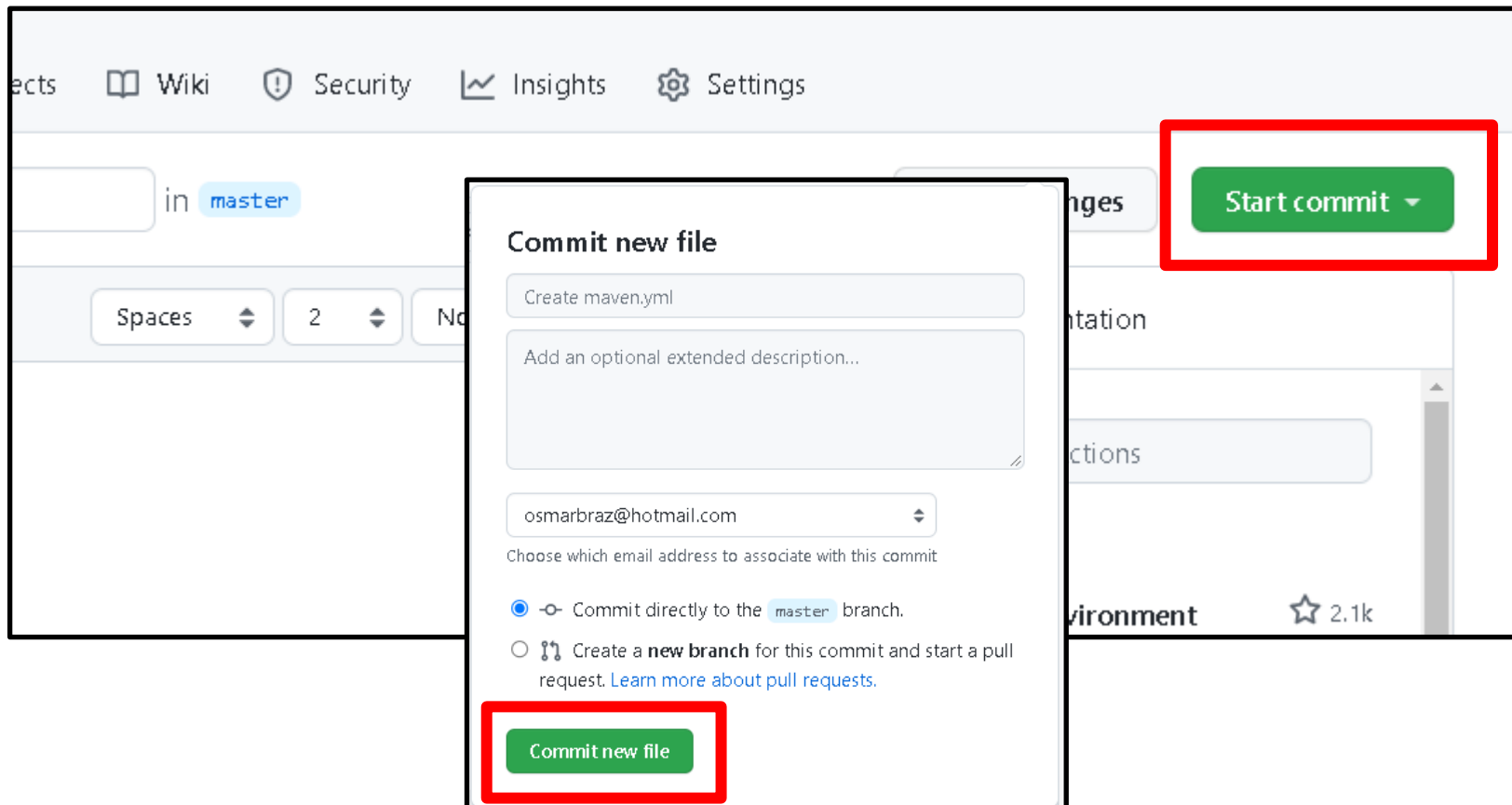
on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 11
        uses: actions/setup-java@v3
        with:
          java-version: '11'
          distribution: 'temurin'
          cache: maven
      - name: Build with Maven
        run: mvn -B package --file pom.xml
```

Passo 3

Submeta (*commit*) as alterações no repositório **github**.



Commit new file

Create maven.yml

Add an optional extended description...

osmarbraz@hotmail.com

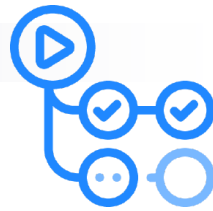
Choose which email address to associate with this commit

☒ Commit directly to the `master` branch.

☐ Create a **new branch** for this commit and start a pull request. [Learn more about pull requests.](#)

Commit new file

Start commit



Passo 4

Ao submeter a alteração do fluxo de trabalho **"Java CI with Maven"** este será executado.

osmarbraz / calculadora Public

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests **Actions** Projects Wiki Security Insights Settings

Workflows [New workflow](#)

All workflows

Java CI with Maven

1 workflow run

✓ Create maven.yml
Java CI with Maven #2: Commit fed269a pushed by osmarbraz

Event Status Branch Actor

36 seconds ago 19s



Passo 4 - Continuação

Os **jobs de *build*** do fluxo são executados pelo **Github Actions** utilizando o automatização provida pelo **Apache Maven**.

The screenshot shows a GitHub Actions workflow run. The workflow is titled "Create maven.yml Java CI with Maven #2" and has a green checkmark icon. In the top right corner, there are buttons for "Re-run all jobs" and a menu icon. The left sidebar shows a "Summary" tab and a list of jobs, with the "build" job highlighted. The main content area shows the details of the "build" job, which is a "maven.yml" job triggered on a push. The job status is "Success" and it took 8 seconds to complete. A red rectangle highlights the "build" job entry, and a blue arrow points to it.

✓ Create maven.yml Java CI with Maven #2 Re-run all jobs ...

Summary

Jobs

✓ build

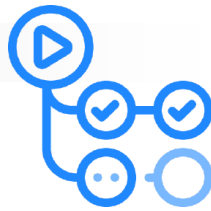
Triggered via push 2 minutes ago

osmarbraz pushed - fed269a master

Status	Total duration	Artifacts
Success	19s	—

maven.yml
on: push

✓ build 8s



Passo 4 - Continuação

Passos do job build.

✓ Create maven.yml Java CI with Maven #2 Re-run all jobs ...

Summary

Jobs

- ✓ build

build
succeeded 3 minutes ago in 8s

Search logs

> ✓ Set up job	2s
> ✓ Run actions/checkout@v3	1s
> ✓ Set up JDK 11	1s
> ✓ Build with Maven	3s
> ✓ Post Set up JDK 11	0s
> ✓ Post Run actions/checkout@v3	0s
> ✓ Complete job	0s

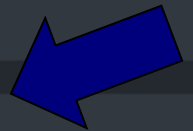


Passo 4 - Continuação

Passo de construção com o **maven**, usando a tarefa **package** do **Apache Maven**.

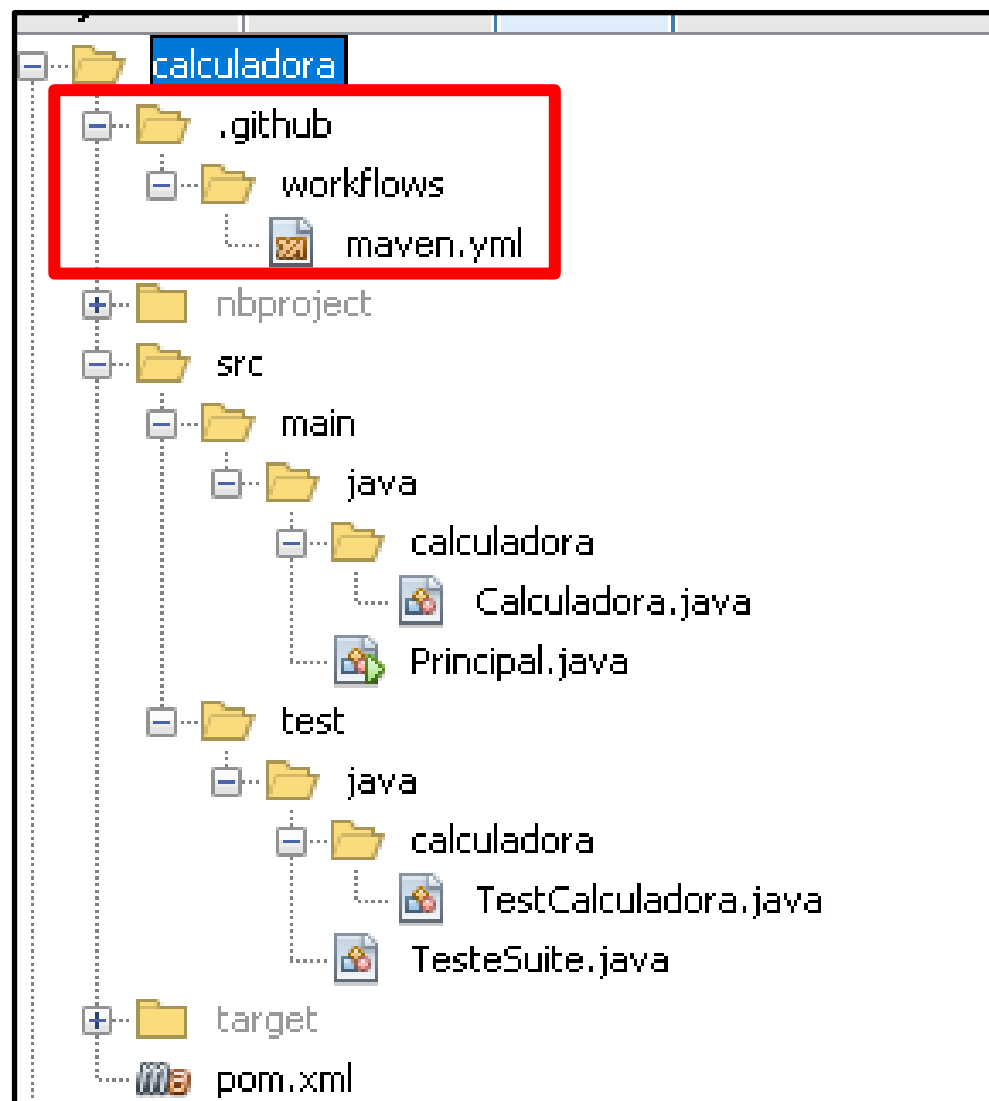
```
build
succeeded 11 minutes ago in 8s

> ✓ Set up job
> ✓ Run actions/checkout@v3
> ✓ Set up JDK 11
v ✓ Build with Maven
  ▶ Run mvn -B package --file pom.xml
  [INFO] Scanning for projects...
```



Passo 5

- Sincronize(***pull remote***) o projeto entre todos os membros do grupo.
- Desta forma o arquivo ***maven.yml*** estará disponível no projeto para todos os desenvolvedores.



Passo 6

Modifique o teste da calculadora(**TestCalculadora.java**), adicionando teste unitário para a subtração.

```
package calculadora;
import static org.junit.Assert.assertEquals;
import org.junit.Test;
public class TestCalculadora {

    @Test
    public void testGetAdicao() {
        Calculadora calculadora = new Calculadora(4.0, 2.0);
        double retornoEsperado = 6.0;
        double retornoFeito = calculadora.getAdicao();
        assertEquals(retornoEsperado, retornoFeito, 0);
    }

    @Test
    public void testGetSubtracao() {
        Calculadora calculadora = new Calculadora(4.0, 2.0);
        double retornoEsperado = 2.0;
        double retornoFeito = calculadora.getSubtracao();
        assertEquals(retornoEsperado, retornoFeito, 0);
    }
}
```

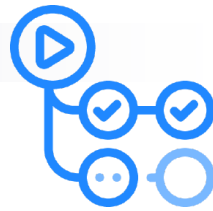
Passo 7

Execute o teste unitário e avalie o resultado.

```
-----  
T E S T S  
-----  
Running calculadora.TestCalculadora  
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.067 sec  
  
Results :  
  
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0  
  
-----  
BUILD SUCCESS  
-----  
  
Total time: 1.638 s  
Finished at: 2022-05-15T11:57:59-03:00  
-----
```

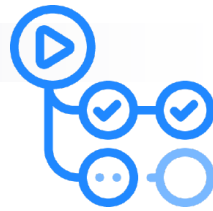
Passo 8

- Submeta ao repositório github o arquivo do teste unitário alterado.
 - Sincronize entre todos os integrantes do grupo.
 - Verifique a execução do ***workflow*** no **Github Actions**.
 - Não deve ter erros



Passo 9

- Vamos criar 3 ambientes (***enviroments***) para homologar o projeto:
 - desenvolvimento(**dev**),
 - Desenvolvimento e compilação do projeto.
 - homologação(**hmg**) e
 - Testes unitários e métricas de software
 - produção(**prd**).
 - Empacotamento do projeto.



Passo 9 - Continuação

- Acesse as configurações(*settings*) do repositório do projeto opção *Enviroments*.

The screenshot shows the GitHub repository settings page for 'osmarbraz / calculadora'. The 'Settings' link in the top navigation bar is highlighted with a red box and a blue arrow. The 'Environments' option in the left sidebar is also highlighted with a red box and a blue arrow.

osmarbraz / calculadora Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Actions

Webhooks

Environments

Repository name

calculadora Rename

☐ Template repository

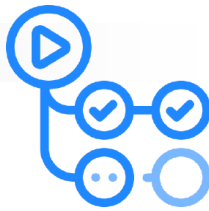
Template repositories let users generate new repositories with the same directory structure and file

Social preview

Upload an image to customize your repository's social media preview.

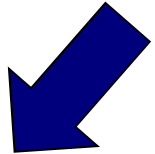
Images should be at least 640×320px (1280×640px for best display).

[Download template](#)

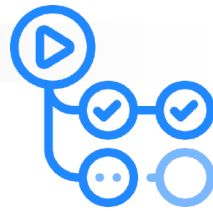


Passo 9

- Clique em “*New Enviroment*” e crie **dev**, **hmg** e **prd**.



The screenshot displays the GitHub 'Environments' page for a repository. On the left is a sidebar with navigation options: General, Access, Collaborators, Moderation options, Code and automation, Branches, Tags, Actions, Webhooks, Environments (selected), and Pages. The main content area is titled 'Environments' and contains a message: 'There are no environments for this repository. Environments are used by your workflows for deployments.' A 'New environment' button is located in the top right corner of this area and is highlighted with a red rectangular box. A large blue arrow points from the text above to this button. Below the main content area, an inset window shows the 'Environments / Add' form. This form has a 'Name' label followed by a text input field. At the bottom of the form is a green button labeled 'Configure environment'.



Passo 9

Ambientes criados.

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Actions

Webhooks

Environments

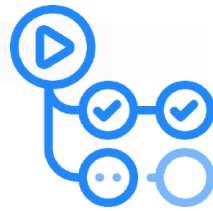
Pages

Environments

New environment

You can configure environments with protection rules and secrets. [Learn more.](#)

prd	
hmg	
dev	



Passo 10

Configure o **pom.xml** para que empacote o projeto com as dependências (**jar-with-dependencies**).

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com</groupId>
  <artifactId>calculadora</artifactId>
  <version>1</version>
  <packaging>jar</packaging>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.13.2</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
```

Codificação dos caracteres
do Código fonte.

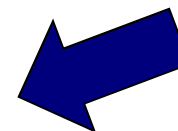
Versão do java de origem e
destino da compilação Java.

Continua...

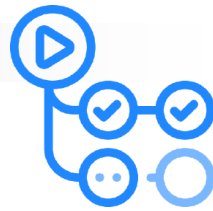


Passo 10 - Continuação

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-assembly-plugin</artifactId>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>single</goal>
          </goals>
          <configuration>
            <archive>
              <manifest>
                <mainClass>
                  Principal
                </mainClass>
              </manifest>
            </archive>
            <descriptorRefs>
              <descriptorRef>jar-with-dependencies</descriptorRef>
            </descriptorRefs>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
</project>
```



Gera um *java archive (jar)* com todas as dependências incluídas.



Passo 11

Ajustaremos o *maven.yml* para executar as tarefas em cada um dos ambientes.

```
name: Integração contínua de Java com Maven
```

```
on:
```

```
push:
```

```
  branches: [ master ]
```

```
pull_request:
```

```
  branches: [ master ]
```

```
jobs:
```

```
  # Jobs do ambiente de desenvolvimento
```

```
  build-dev:
```

```
    runs-on: ubuntu-latest
```

```
    environment:
```

```
      name: dev
```

```
    steps:
```

```
      - name: Realiza o checkout do repositório
```

```
        uses: actions/checkout@v3
```

```
      - name: Configura o JDK 18
```

```
        uses: actions/setup-java@v3
```

```
        with:
```

```
          java-version: '18'
```

```
          distribution: 'temurin'
```

```
          cache: maven
```

```
      - name: Compila o projeto com Maven
```

```
        run: mvn -B compile --file pom.xml
```

Jobs do ambiente dev

Step do maven do ambiente dev

Continua...



Passo 11 - Continuação

Jobs do ambiente hmg

Jobs do ambiente de homologação
build-hmg:

runs-on: ubuntu-latest

environment:

name: **hmg**

needs: **build-dev**

steps:

- name: Realiza o checkout do repositório

uses: actions/checkout@v3

with:

fetch-depth: 0

- name: Configura o JDK 18

uses: actions/setup-java@v3

with:

java-version: '18'

distribution: 'temurin'

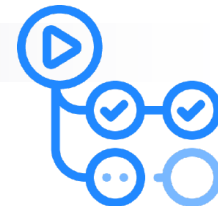
cache: maven

- name: Testa o projeto com Maven

run: mvn -B **test** --file pom.xml

Step do maven do ambiente hmg

Continua...



Passo 11 - Continuação

Jobs do ambiente de produção

build-prd:

runs-on: ubuntu-latest

environment:

name: **prd**

needs: **build-hmg**

steps:

- name: Realiza o checkout do repositório

uses: actions/checkout@v3

- name: Configura o JDK 18

uses: actions/setup-java@v3

with:

java-version: '18'

distribution: 'temurin'

cache: maven

Executa o empacotamento do projeto com o Maven.

O parâmetro -DskipTests pula os testes pois já foram executados no ambiente anterior.

- name: Empacota o projeto com Maven

run: mvn -B **package** --file pom.xml -DskipTests

Cria o artefato de download

- name: Crie uma pasta temporária de downloads de artefatos

run: mkdir staging

- name: Copia os artefatos a serem publicados na pasta temporária

run: cp target/*jar-with-dependencies.jar staging

- name: Usa Upload Artifact do GitHub Action

uses: actions/upload-artifact@v3

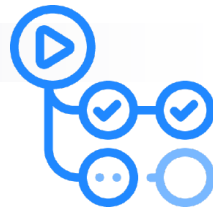
with:

name: Package

path: staging

Jobs do ambiente prd

Step do maven do ambiente prd



Passo 11

É possível adicionar revisores(*reviewers*) entre os ambientes.

The screenshot shows the GitHub interface for the repository 'osmarbraz / calculadora' (Public). The 'Settings' tab is selected in the top navigation bar. On the left sidebar, the 'Environments' section is highlighted. The main content area is titled 'Environments / Configure prd'. Under the 'Environment protection rules' section, which includes the subtitle 'Can be used to configure manual approvals and timeouts.', there are two checkboxes. The first checkbox, 'Required reviewers', is highlighted with a red rectangular box. Its description is 'Specify people or teams that may approve workflow runs when they access this environment.' The second checkbox is 'Wait timer', with the description 'Set an amount of time to wait before allowing deployments to proceed.' At the bottom of the configuration area is a green button labeled 'Save protection rules'.

Passo 12

Submeta ao repositório **github** as alterações dos arquivos **pom.xml** e **maven.yml**



Passo 13

Verifique se o **workflow** da integração continua foi executada com sucesso.

The screenshot shows the GitHub Actions interface for a workflow named "Início do projeto Integração continua de Java com Maven #4". The workflow status is "Success". The summary section shows it was triggered via push 2 minutes ago by osmarbraz pushed ee5b664 on the master branch. The total duration is 1m 25s, and there is 1 artifact.

The jobs section lists three jobs: build-dev, build-hmg, and build-prd, all of which are successful. A red box highlights the job progress bar, which shows the following durations: build-dev (18s), build-hmg (13s), and build-prd (27s). A green box with the text "Verde! Execução sem erros" and a blue arrow points to the job progress bar.

The artifacts section shows a single artifact named "Package" with a size of 2.12 KB. A red box highlights the artifact name, and a blue arrow points to it with the text "Arquivo do projeto empacotado."

Job	Status	Duration
build-dev	Success	18s
build-hmg	Success	13s
build-prd	Success	27s

Name	Size
Package	2.12 KB

Conclusão

- A Integração Continua é um processo essencial a qualquer software que deseja manter vivo por um período de tempo mais longo.
- Conhecer e dominar as ferramentas é um ponto crítico para garantir agilidade no processo de distribuição do software.
- Nesta etapa criamos a integração continua com o Github Actions.

Referências

- PRESSMAN, Roger; MAXIM, Bruce. Engenharia de software: uma abordagem profissional. 8.ed. Bookman, 2016.
- SOMMERVILLE, Ian. Engenharia de software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.
- LARMAN, Craig. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e desenvolvimento iterativo. 3. ed Porto Alegre: Bookman, 2007



Fim