

# CI/CD



## 1. Objetivo do Tutorial

Este tutorial visa criar um ambiente local para simulação de uma esteira de desenvolvimento com CI/CD (Integração Contínua e Entrega Contínua).

## 2. Pré-requisitos para realizar esse tutorial

Antes de começar, verifique se Java e Maven estão instalados e configurados corretamente, siga as instruções abaixo para configurar as variáveis de ambiente e demais ferramentas.

- **Guia Docker** [Instalação Docker Compose no Ubuntu 20.04](#).
- **Guia DBeaver** [Instalação DBeaver](#).
- **Guia Git** [Instalação Git](#).
- **Guia Jenkins** [Instalação Jenkins](#).

## 3. Instalando Ferramentas

### 3.1 Instalando o Git

- Execute os seguintes comandos para instalar

```
sudo add-apt-repository ppa:git-core/ppa  
sudo add-apt-repository ppa:git-core/ppa  
sudo apt-get update && sudo apt-get -y install git
```

- Execute o comando para visualizar a versão instalada

```
git --version
```

- Execute o comando para configurar usuário e e-mail global

```
git config --global user.name "Firstname Lastname"  
git config --global user.email firstname.lastname@mail.com
```

- Execute o comando para validar as configurações realizadas

```
cat .gitconfig
```

### 3.2 Configurando chave SSH

- Execute o seguinte comando para gerar a chave SSH

```
ssh-keygen -t rsa -b 4096 -C "seu email git"
```

Vá pressionando ENTER até finalizar

```
thiago@admin-pc:~/Documentos/WORKSPACE/SPRINGBOOT_JENKINS$ ssh-keygen -t rsa -b 4096 -C "thiago.henrique.25@hotmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/thiago/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/thiago/.ssh/id_rsa
Your public key has been saved in /home/thiago/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:ZuPEiOU+Cu0nq5V8Q4a7vvMonwba9zchYIDrXmcmdZ0 @thiago.henrique.25@hotmail.com
The key's randomart image is:
+---[RSA 4096]----+
| . |
| .. |
| . . . |
| .. += . E |
| .. ooSo |
| o+o&++. |
| + ox=* . |
| . +.+0 .o |
| .B0=o..o |
+---[SHA256]----+
```

- Execute o seguinte comando para obter a chave SSH

```
cat ~/.ssh/id_rsa.pub
```

```
thiago@admin-pc:~/Documentos/WORKSPACE/SPRINGBOOT_JENKINS$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQACQDRg2bJpfLgs6VwgvYvX5X/k0rrj7644l1R5ogS98VjQ0B4D9jpX
HeXqSPzg/gxjX8otKyffzXLgj8SIRvy6/NAgpDGhnAH+KrhYb9kfuYJ4NOVq30zbRCP+xXI6036uKFEq5NBVSy/xI
3uBjZC2Nm4fg6xVTCCLCq0KZMPRVw8+AHIwm607sNqYqw3bxja7TI+y2XotPyIM0tM29AfQvN1Rc5te81fELQH
p12uLBWsfxZ2yuNmI/bWACNTNekVYdvbs4T7e50gMMcvp2Q1oLL+NqI68iZf4xa6USnyFUVNYsfE+p67DYeQgKVfTP2ks
```

- Configurando a chave SSH no GIT

Cole a chave SSH obtida anteriormente

- Testando comunicação do git instalado na sua máquina com sua conta criada no github

```
thiago@admin-pc:~/Documentos/WORKSPACE/SPRINGBOOT_JENKINS$ ssh -T git@github.com
Hi thiago-jv! You've successfully authenticated, but GitHub does not provide shell access.
```

### 3.3 Instalando o DBeaver

- Execute os seguintes comandos para instalar

```
echo "deb https://dbeaver.io/debs/dbeaver-ce /" | sudo tee
/etc/apt/sources.list.d/dbeaver.list
wget -O - https://dbeaver.io/debs/dbeaver.gpg.key | sudo apt-key add -
sudo apt-get update
sudo apt-get install dbeaver-ce
```

- Execute o comando para visualizar a versão instalada

```
dbeaver --version
```

- Execute o comando para executar o aplicativo

```
dbeaver
```

### 3.4 Instalando o Java OpenJDK 17

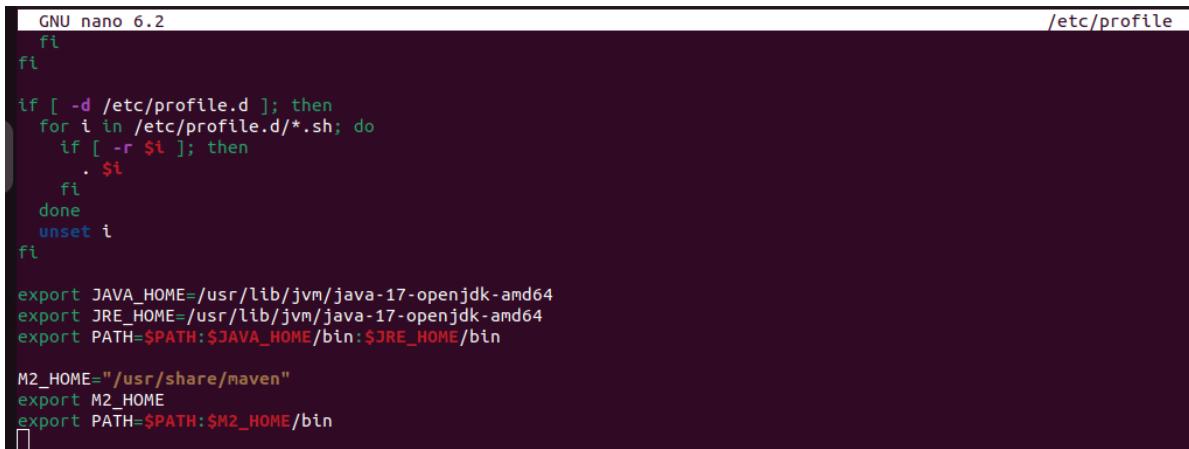
- Execute o comando para instalar o java

```
sudo apt-get install openjdk-17 -yes
```

### 3.5 Configurando Variável de ambiente Java Home

- Execute o comando

```
sudo nano /etc/profile
```



```
GNU nano 6.2
/etc/profile
if [ -d /etc/profile.d ]; then
    for i in /etc/profile.d/*.sh; do
        if [ -r $i ]; then
            . $i
        fi
    done
    unset i
fi

export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
export JRE_HOME=/usr/lib/jvm/java-17-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin

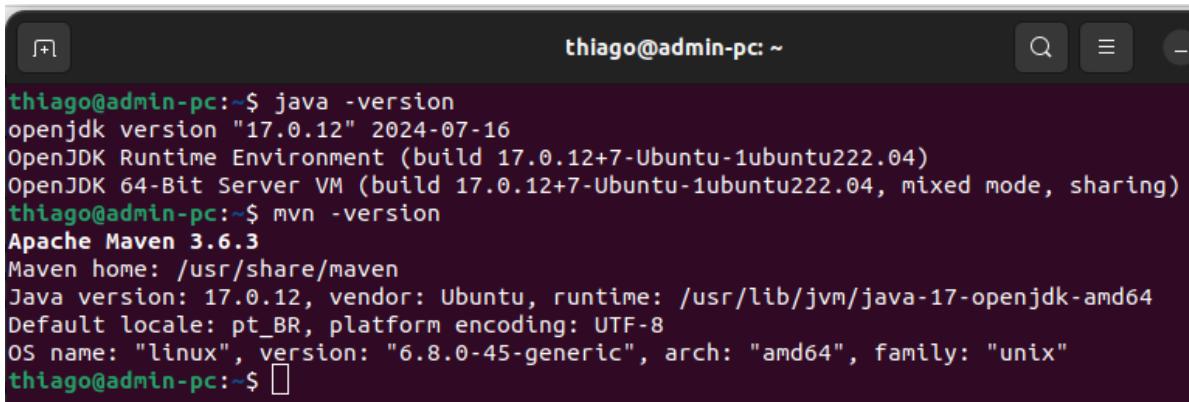
M2_HOME="/usr/share/maven"
export M2_HOME
export PATH=$PATH:$M2_HOME/bin
[]
```

- Cole o conteúdo abaixo

```
export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
export JRE_HOME=/usr/lib/jvm/java-17-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
```

- Verifique a instalação do Java instalada

```
java -version
```



```
thiago@admin-pc:~$ java -version
openjdk version "17.0.12" 2024-07-16
OpenJDK Runtime Environment (build 17.0.12+7-Ubuntu-1ubuntu222.04)
OpenJDK 64-Bit Server VM (build 17.0.12+7-Ubuntu-1ubuntu222.04, mixed mode, sharing)
thiago@admin-pc:~$ mvn -version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 17.0.12, vendor: Ubuntu, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: pt_BR, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-45-generic", arch: "amd64", family: "unix"
thiago@admin-pc:~$ []
```

### 3.6 Instalando o Maven

- Execute o comando

```
sudo apt install -y maven
```

### 3.7 Configurando variável de ambiente Maven Home

- Execute o comando

```
sudo nano /etc/profile
```

```
GNU nano 6.2                                     /etc/profile

if [ -d /etc/profile.d ]; then
    for i in /etc/profile.d/*.sh; do
        if [ -r $i ]; then
            . $i
        fi
    done
    unset i
fi

export JAVA_HOME=/usr/lib/jvm/java-17-openjdk-amd64
export JRE_HOME=/usr/lib/jvm/java-17-openjdk-amd64
export PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin

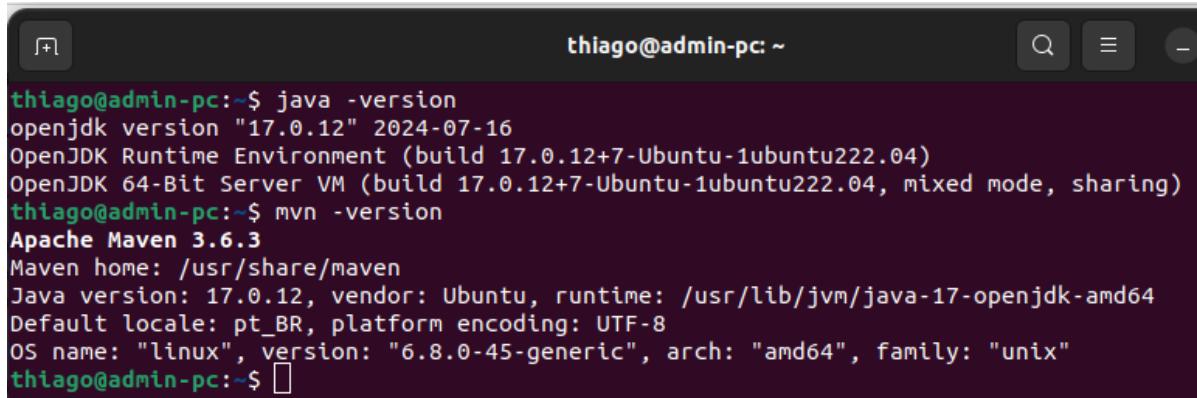
M2_HOME="/usr/share/maven"
export M2_HOME
export PATH=$PATH:$M2_HOME/bin
```

- Adicione as seguintes linhas ao final do arquivo

```
M2_HOME="/usr/share/maven"
export M2_HOME
export PATH=$PATH:$M2_HOME/bin
```

- Verifique a instalação do Maven

```
mvn --version
```



```
thiago@admin-pc:~$ java -version
openjdk version "17.0.12" 2024-07-16
OpenJDK Runtime Environment (build 17.0.12+7-Ubuntu-1ubuntu222.04)
OpenJDK 64-Bit Server VM (build 17.0.12+7-Ubuntu-1ubuntu222.04, mixed mode, sharing)
thiago@admin-pc:~$ mvn -version
Apache Maven 3.6.3
Maven home: /usr/share/maven
Java version: 17.0.12, vendor: Ubuntu, runtime: /usr/lib/jvm/java-17-openjdk-amd64
Default locale: pt_BR, platform encoding: UTF-8
OS name: "linux", version: "6.8.0-45-generic", arch: "amd64", family: "unix"
thiago@admin-pc:~$
```

## 4. Introdução as Ferramentas

### 4.1 A Base da Casa

- **Ubuntu:** É o terreno onde a casa será construída. Ele fornece a base sólida e estável para todas as outras tecnologias. Imagine-o como o alicerce da sua casa.
- **Docker:** São os contêineres que você usa para transportar os materiais de construção (seu código) para o local da obra (o ambiente de execução). Cada contêiner é como um caminhão que carrega tudo o que você precisa para construir um cômodo específico da casa.

#### *4.2 Projetando a Casa*

- **Java:** É a linguagem de programação que você usa para desenhar os projetos da casa. É como o arquiteto que cria os blueprints.
- **Spring:** É um framework que facilita a construção da casa. Ele fornece ferramentas e bibliotecas prontas para uso, assim como um construtor que te ajuda a montar as paredes e o telhado de forma eficiente.

#### *4.3 Gerenciando o Projeto e a Qualidade*

- **Git:** É o sistema de controle de versão que te permite acompanhar todas as mudanças feitas na casa durante a construção. É como um álbum de fotos que registra cada etapa da obra.
- **Maven:** É o gerente de projetos que organiza todos os materiais de construção (bibliotecas, dependências) e garante que tudo esteja no lugar certo.
- Jenkins: É o supervisor da obra. Ele automatiza tarefas repetitivas, como construir a casa, realizar testes e garantir que tudo esteja funcionando corretamente.
- **SonarQube:** É o inspetor de qualidade. Ele verifica se a casa está sendo construída de acordo com as normas e padrões de qualidade.

#### *4.4 Mobiliando a Casa: Banco de Dados*

- **PostgreSQL:** É o depósito onde você guarda todos os móveis e objetos da casa. Ele armazena os dados da sua aplicação, como informações de usuários, produtos, etc.
- **DBeaver:** É a ferramenta que você usa para organizar e gerenciar os móveis e objetos dentro do depósito.

#### *4.5 As Ferramentas do Construtor*

- **IntelliJ IDEA:** É a caixa de ferramentas do construtor. Ela fornece todas as ferramentas necessárias para construir a casa, como uma serra, um martelo e uma chave de fenda.

### *5. Configuração do Docker Compose*

- O arquivo docker-compose.yml com as configurações das imagens a serem baixadas do SonarQube e PostgreSQL encontram-se no projeto conforme imagem abaixo.

The screenshot shows a code editor interface with a dark theme. On the left is a file tree for a Jenkins project named 'jenkins [app]'. The 'extras' directory contains a file named 'docker-compose.yml', which is highlighted with a red box. The right pane displays the contents of this file:

```
1 services:
2   db-postgresql:
3     ports:
4       - "5432:5432"
5     volumes:
6       - pgdata:/var/lib/postgresql/data
7     networks:
8       - network-rede-local
9
10  sonarqube:
11    container_name: sonar
12    image: sonarqube:9.0-community
13    ports:
14      - "9000:9000"
15    networks:
16      - network-rede-local
17    environment:
18      - POSTGRES_USER=admin
19      - POSTGRES_PASSWORD=admin
20      - POSTGRES_DB=bdsoma
21    depends_on:
22      - db-postgresql
23    volumes:
24      - sonarqube_conf:/opt/sonarqube/conf
25      - sonarqube_data:/opt/sonarqube/data
26      - sonarqube_extensions:/opt/sonarqube/extensions
27      - sonarqube_bundled-plugins:/opt/sonarqube/lib/bundled-plugins
28
29  volumes:
30    sonarqube_conf:
31    sonarqube_data:
32    sonarqube_extensions:
33    sonarqube_bundled-plugins:
34    pgdata:
35
36  networks:
37    network-rede-local:
38      driver: bridge
```

- Execute o comando abaixo para criação dos containers dentro do diretório extras no projeto.

```
docker compose up
```

- O resultado será os containers criados conforme imagem abaixo.

The screenshot shows a terminal window with a dark theme. The command 'docker ps' was run, and the output is displayed:

```
thiago@admin-pc:~$ docker ps
CONTAINER ID        IMAGE               COMMAND
a1ed9749a80c        sonarqube:9.0-community   "bin/run.sh bin/sona..."
24a0c44c11ee        postgres             "docker-entrypoint.s..."
```

## 6. Acessando Ferramentas

### 6.1 Jenkins

- Abra um navegador e acesse: <http://localhost:8080>.
- Senha Inicial: Você será solicitado a inserir uma senha inicial. Essa senha pode ser encontrada nos logs do container.

#### Executar os Comandos para Obter a Senha

- Execute os seguintes comandos

```
sudo systemctl start jenkins
```

```
sudo systemctl status jenkins
```

```
thiago@admin-pc: $ sudo systemctl start jenkins
thiago@admin-pc: $ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: Loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
  Active: active (running) since Fri 2024-10-11 08:45:45 -04; 36s ago
    Main PID: 23670 (java)
      Tasks: 58 (limit: 18841)
        Memory: 672.2M
          CPU: 16.390s
        CGroup: /system.slice/jenkins.service
               └─23670 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

out 11 08:45:14 admin-pc jenkins[23670]: 24f70155b5c44c409927efc6a08add58
out 11 08:45:14 admin-pc jenkins[23670]: hints may also be found at: /var/tcb/jenkins/secrets/initialAdminPassword
out 11 08:45:14 admin-pc jenkins[23670]: ****
out 11 08:45:14 admin-pc jenkins[23670]: ****
```

- Anote a Senha: Use a senha exibida para acessar o painel do Jenkins.
- Acesse o Jenkins Novamente: Abra um navegador e vá para <http://localhost:8080>.

#### Começando

## Abrir o Jenkins

Para garantir que o Jenkins está configurado de forma segura pelo administrador, uma senha foi escrita no arquivo de registro ([não sabe onde encontrar?](#)) e neste arquivo no servidor:

`/var/jenkins_home/secrets/initialAdminPassword`

Por favor copie a senha de qualquer uma das localizações e cole abaixo.

Senha do administrador

Continuar

- Instalar os Plugins Sugeridos são obrigatórios.

Getting Started

# Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

**Install suggested plugins**

Install plugins the Jenkins community finds most useful.

**Select plugins to install**

Select and install plugins most suitable for your needs.



Getting Started

# Getting Started

<input type="radio"/> Folders Plugin	<input type="radio"/> OWASP Markup Formatter Plugin	<input type="radio"/> Build Timeout	<input type="radio"/> Credentials Binding Plugin	** Ionicons API
<input type="radio"/> Timestamper	<input type="radio"/> Workspace Cleanup	<input type="radio"/> Ant	<input type="radio"/> Gradle	
<input type="radio"/> Pipeline	<input type="radio"/> GitHub Branch Source Plugin	<input type="radio"/> Pipeline: GitHub Groovy Libraries	<input type="radio"/> Pipeline: Stage View	
<input type="radio"/> Git plugin	<input type="radio"/> SSH Build Agents	<input type="radio"/> Matrix Authorization Strategy Plugin	<input type="radio"/> PAM Authentication	
<input type="radio"/> LDAP	<input type="radio"/> Email Extension	<input type="radio"/> Mailer Plugin		

- Criando usuário de acesso ao Jenkins.

Getting Started

## Criar o primeiro usuário administrativo

Nome de usuário:

Senha:

Confirmar a senha:

Nome completo:

Endereço de e-mail:

Jenkins 2.346.3

[Skip and continue as admin](#)

[Save and Continue](#)

- Nome de usuário: seu usuário de preferência.
- Senha: sua senha de preferência.
- Confirmar a senha: confirme a senha.
- Nome completo: seu nome de usuário completo de preferência.
- Endereço de e-mail: seu endereço de e-mail.

- Porta de acesso a instancia do Jenkins.
- 

## Getting Started

# Configuração da instancia

URL do Jenkins:

`http://localhost:8080/`

A URL do Jenkins é usada para prover a URL raiz para links absolutos para vários recursos do Jenkins. Isto significa que este valor é requerido para a operação apropriada de muitas funcionalidades do Jenkins incluindo notificações por e-mail, atualização de estado de PR e a variável de ambiente `BUILD_URL` provida pelos passos de construção.

O valor proposto padrão mostrado é **ainda não salvo** e é gerado da solicitação atual, se possível. A melhor prática é configurar este valor para a URL que espera-se que os usuários utilizem. Isto evita confusão quando compartilhando ou vendendo links.

## Getting Started

# Jenkins is almost ready!

Your Jenkins setup is complete, but some plugins require Jenkins to be restarted.

[Restart](#)

- Login de acesso ao Jenkins.

**Sign in to Jenkins**

Nome do usuário  
kubedev

Senha  
.....

Mantenha-me conectado

Entrar

- Página inicial do Jenkins.

**Welcome to Jenkins!**

Painel de controle >

Novo tarefa

Histórico de compilações

Gerenciar Jenkins

Minhas visões

Open Blue Ocean

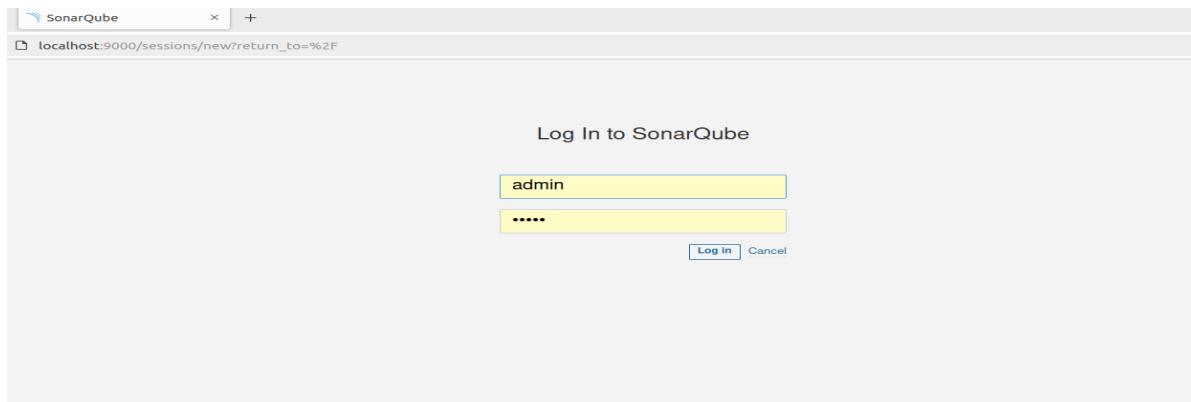
pesquisar (CTRL+)

- Instalar o plugin Pepiline Maven Integration conforme caminho na imagem abaixo.

The screenshot shows the Jenkins 'Plugins' page. In the search bar at the top right, the text 'pesi' is typed. Below the search bar, there is a sidebar with links: 'Atualizações', 'Extensões disponíveis' (which is highlighted in grey), 'Extensões instaladas', and 'Configurações avançadas'. The main content area has a heading 'Pipeline Maven' with a search icon. Below it, there is a table with one row. The row contains a checkbox (which is checked), the name 'Pipeline Maven Integration 1457.vf7a\_de13b\_c0d4', and a link 'Instalar'. A small note below the table says: 'This plugin provides integration with Pipeline, configures maven environment to use within a pipeline job by calling sh mvn or bat mvn. The selected maven installation will be configured and prepended to the path.' There is also a link 'Pipeline Maven Integration 1457.vf7a\_de13b\_c0d4' at the bottom of the table row.

## 6.2 SonarQube

- Abra um navegador e acesse: <http://localhost:9000>.
- Dados de Acesso:
  - Usuário: admin
  - Senha: admin



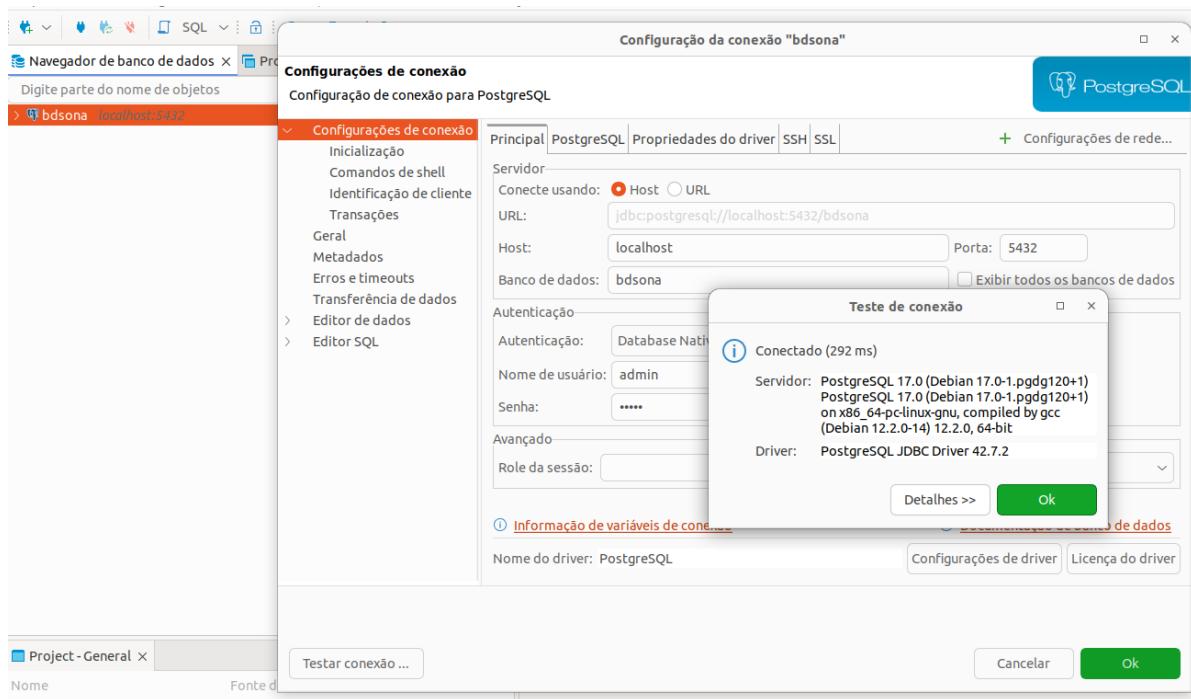
- Página inicial do SonarQube.

The screenshot shows the SonarQube interface. At the top, there is a navigation bar with links: 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', 'Administration', and a search bar 'Search for projects...'. Below the navigation bar, there is a section titled 'How do you want to create your project?'. It contains a message: 'Are you just testing or have an advanced use-case? Create a project manually. Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.' Below this message, there is a note: 'We recommend setting up a DevOps platform configuration so you and your team can benefit from more SonarQube features.' There are five cards for creating a project:
 

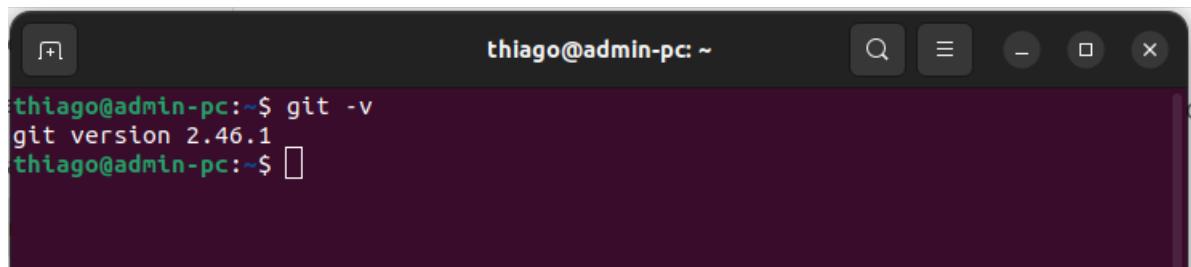
- 'From Azure DevOps' with a blue icon of a cloud and arrow.
- 'From Bitbucket' with a blue icon of a square with a white 'B'.
- 'From GitHub' with a black icon of a GitHub logo.
- 'From GitLab' with a red and orange icon of a cat-like head.
- 'Manually' with a blue icon of two arrows pointing left and right.

 Each card has a note below it: 'Global configuration not set'.

### 6.3 PostgreSQL com DBeaver



### 6.4 Git



## 7. Configuração do Projeto Spring Boot

### 7.1 docker-compose.yml

- Este arquivo é usado para definir e executar serviços Docker.

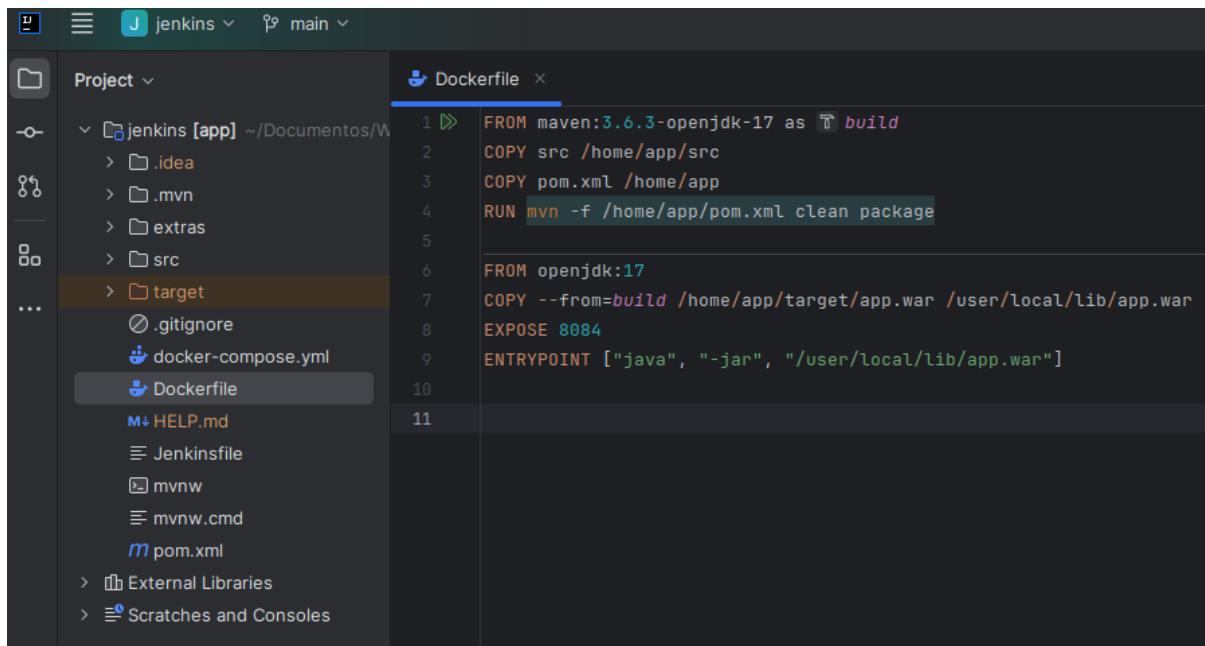
The screenshot shows a code editor interface with a dark theme. On the left is a file tree for a project named "jenkins [app]". The "target" folder is selected. Inside "target", there are files: .gitignore, docker-compose.yml (which is currently selected), Dockerfile, HELP.md, Jenkinsfile, mvnw, mvnw.cmd, pom.xml, External Libraries, and Scratches and Consoles. The right pane displays the contents of the docker-compose.yml file:

```
1 services:
2   backend:
3     container_name: backend
4     restart: always
5     build: .
6     ports:
7       - "8084:8084"
8     networks:
9       - network-rede-local
10
11 networks:
12   network-rede-local:
13     driver: bridge
14
```

- **services:** Define os serviços que serão executados. Neste caso, temos apenas um serviço chamado backend.
- **container\_name:** Nome do contêiner que será criado.
- **restart:** Define a política de reinício do contêiner. always significa que ele será reiniciado se falhar.
- **build:** Indica que o contêiner deve ser construído a partir do Dockerfile na pasta atual (.).
- **ports:** Mapeia a porta 8084 do contêiner para a porta 8084 da máquina host.
- **networks:** Define a rede na qual o contêiner operará.

## 7.2 Dockerfile

- Este arquivo contém as instruções para criar uma imagem Docker personalizada.



The screenshot shows a code editor interface with a sidebar and a main panel. The sidebar on the left lists files and folders: .idea, .mvn, extras, src, target, .gitignore, docker-compose.yml, Dockerfile (which is selected), HELP.md, Jenkinsfile, mvnw, mvnw.cmd, pom.xml, External Libraries, and Scratches and Consoles. The main panel displays the content of the Dockerfile:

```
1 FROM maven:3.6.3-openjdk-17 as build
2 COPY src /home/app/src
3 COPY pom.xml /home/app
4 RUN mvn -f /home/app/pom.xml clean package
5
6 FROM openjdk:17
7 COPY --from=build /home/app/target/app.war /user/local/lib/app.war
8 EXPOSE 8084
9 ENTRYPOINT ["java", "-jar", "/user/local/lib/app.war"]
10
11
```

- FROM:** Define a imagem base. Aqui usamos uma imagem do Maven com OpenJDK 17 para a fase de construção.
- COPY:** Copia arquivos e diretórios do sistema de arquivos do host para a imagem Docker.
- RUN:** Executa comandos na imagem. Aqui, usamos o Maven para compilar o aplicativo e gerar um arquivo .war.
- EXPOSE:** Informa ao Docker que o contêiner escutará na porta 8084.
- ENTRYPOINT:** Define o comando que será executado quando o contêiner iniciar. Neste caso, estamos executando a aplicação Java.

## 7.3 Pipeline Jenkins

Este é o script que define o pipeline de CI/CD no Jenkins.

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                dir('jenkins') {
                    sh 'mvn clean package'
```

```

    }
}

}

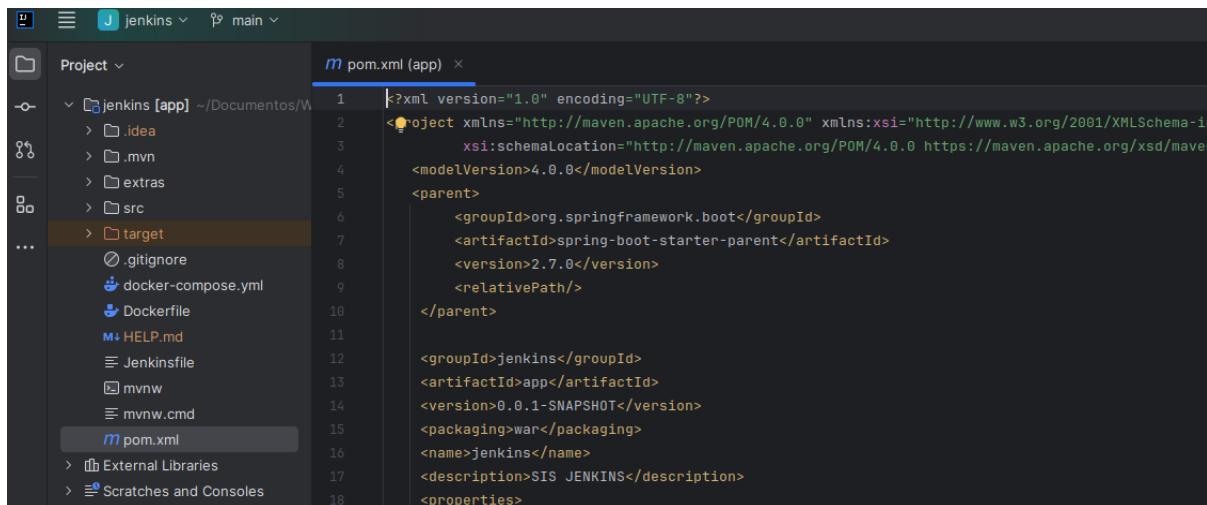
stage('Deploy') {
    steps {
        dir('jenkins') {
            sh 'docker-compose up -d'
        }
    }
}
}

```

- **pipeline**: Define que estamos criando um pipeline.
- **agent any**: Indica que o pipeline pode ser executado em qualquer agente disponível.
- **stages**: Define as etapas do pipeline. Temos duas: **Build** e **Deploy**.
- **stage('Build')**: Esta etapa executa o comando Maven para construir o projeto e gerar o arquivo .war.
- **steps**: Define as ações a serem executadas nesta etapa.
- **dir('jenkins')**: Muda o diretório de trabalho para a pasta jenkins.
- **sh 'mvn clean package'**: Executa um comando shell que chama o Maven para limpar a construção anterior e empacotar o projeto.
- **stage('Deploy')**: Esta etapa utiliza o Docker Compose para subir o contêiner com a aplicação.
- **steps**: Define as ações a serem executadas nesta etapa.
- **dir('jenkins')**: Novamente muda o diretório de trabalho para jenkins.
- **sh 'docker-compose up -d'**: Executa um comando shell para iniciar os contêineres definidos no arquivo docker-compose.yml em modo destacado (-d). Isso significa que os contêineres serão executados em segundo plano, permitindo que o Jenkins não fique bloqueado e continue o fluxo da pipeline sem interrupções.

#### *7.4 pom.xml*

- Este é o arquivo de configuração do Maven que define as dependências e configurações do seu projeto.



```

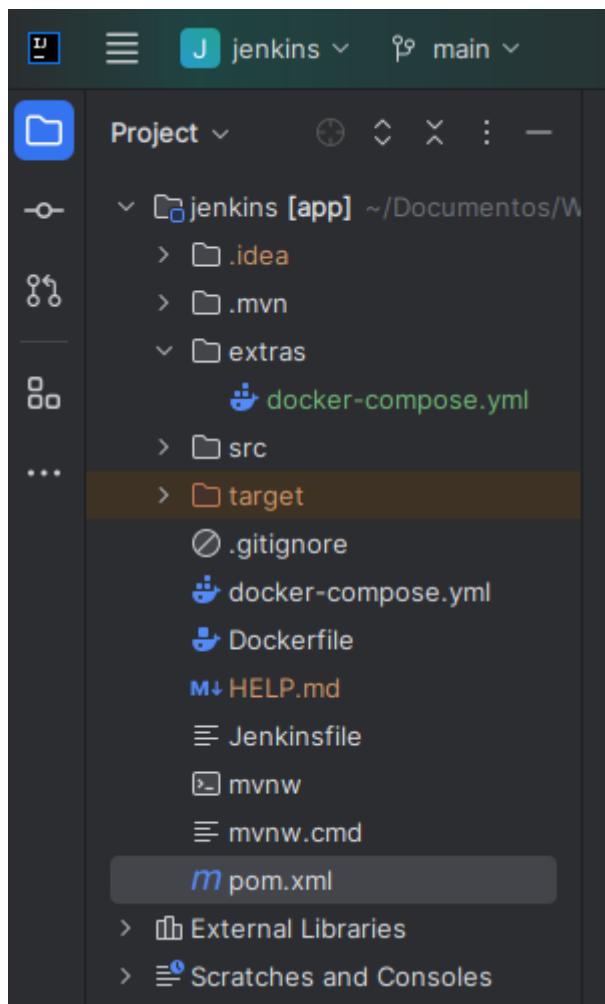
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven_4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.0</version>
        <relativePath/>
    </parent>

    <groupId>jenkins</groupId>
    <artifactId>app</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>jenkins</name>
    <description>SIS JENKINS</description>
    <properties>

```

- **Declaração XML:** Indica que é um documento XML e define a versão e a codificação.
- **Projeto:** O elemento raiz que contém todas as informações sobre o projeto.
- **Versão do Modelo:** Especifica a versão do modelo POM que está sendo utilizado.
- **Projeto Pai:** Define um projeto pai do qual este projeto herda configurações, facilitando a gestão de dependências e configurações comuns.
- **Coordenadas do Projeto:**
  - **groupId:** Identifica o grupo ou a organização do projeto.
  - **artifactId:** O nome do artefato gerado.
  - **version:** A versão do projeto, com "SNAPSHOT" indicando que é uma versão em desenvolvimento.
  - **packaging:** O tipo de empacotamento do projeto (neste caso, um arquivo WAR).
- **Propriedades do Projeto:** Define propriedades, como codificação de caracteres e a versão do Java utilizada.
- **Dependências:** Lista as bibliotecas necessárias para o projeto, incluindo detalhes como o grupo, o artefato e a versão. Também pode especificar o escopo das dependências (por exemplo, se são necessárias apenas para testes).
- **Configuração de Construção:**
  - Define como o projeto será construído, incluindo o nome final do artefato gerado.
  - Lista plugins utilizados durante a construção, como o plugin do Spring Boot, que ajuda a criar um JAR ou WAR executável.
- **Perfis:** Permite definir configurações específicas que podem ser ativadas conforme necessário. Um perfil pode incluir plugins e configurações específicas, como um plugin para construir imagens Docker.

## 7.5 Estrutura básica do projeto



```
package com.jenkins.controller;

import ...

@RestController
@RequestMapping("/api")
public class HelloJenkins {

    @GetMapping("/hello")
    public String hello() { return "Hello, World!"; }
}
```

## 8. Interações, Configurações SonarQube, Quality Gates, Acessos Git e Criação da Pipeline

- Criando credenciais de acesso ao GitHub do seu usuário, conforme o caminho indicado na imagem abaixo.

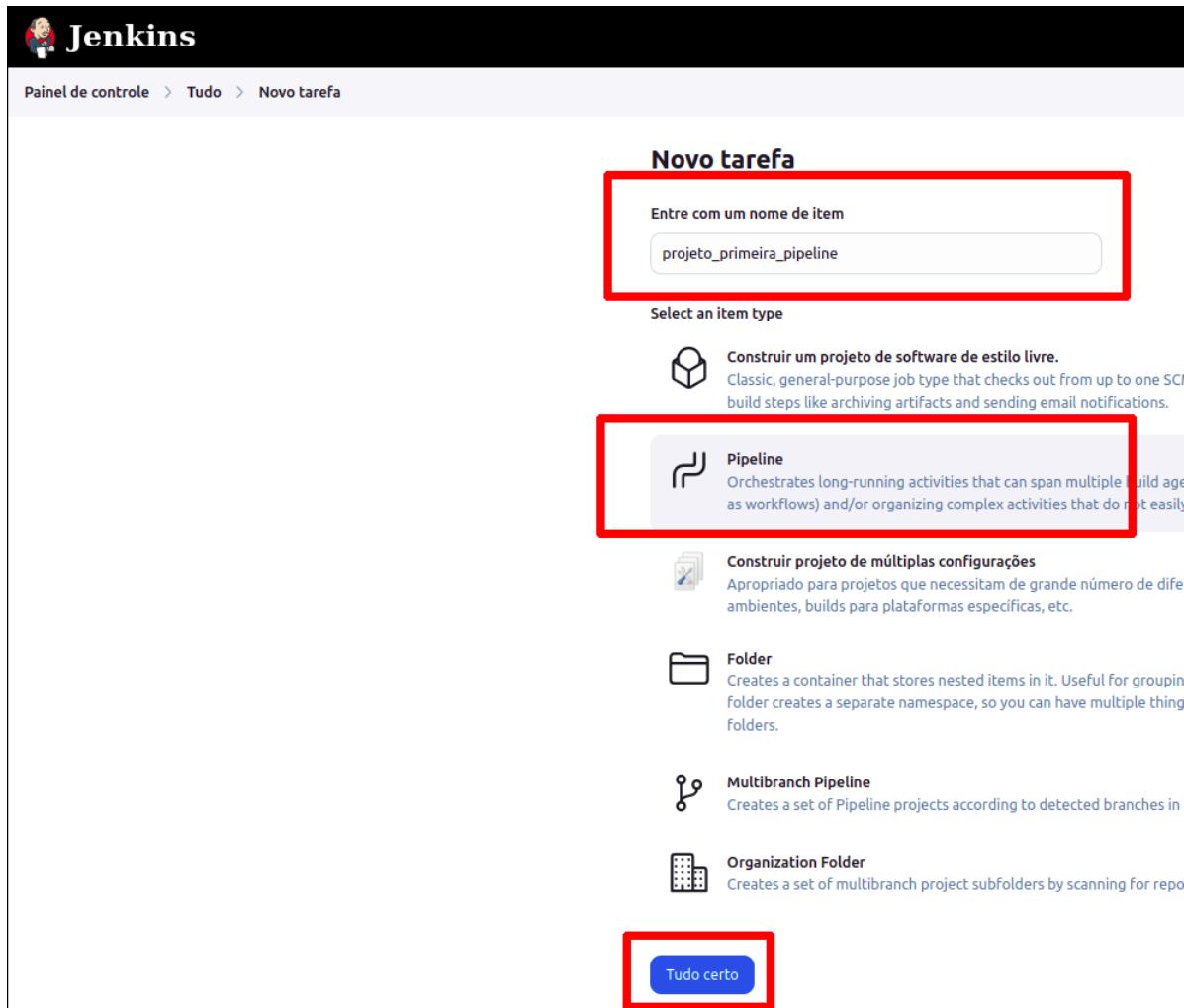
The screenshot shows the Jenkins 'Credentials' page under 'Stores scoped to Jenkins'. It lists a single credential named 'System' of type 'Username with password'. The 'Domain' dropdown is set to '(global)'. A button labeled 'Add credentials' is visible. The top navigation bar includes links for Painel de controle, Gerenciar Jenkins, and Credentials.

The screenshot shows the 'New credentials' page for creating a new credential. The 'Kind' is selected as 'Username with password'. The 'Scope' is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' field contains 'thiago-jv', and the 'Password' field contains a masked value. The 'ID' field is 'GIT' and the 'Description' is 'Credenciais de Acesso ao GIT'. A 'Create' button is at the bottom.

The screenshot shows the 'Global credentials (unrestricted)' page, listing the previously created credential 'thiago-jv/\*\*\*\*\* (Credenciais de Acesso ao GIT)'. The table columns are ID, Name, Kind, and Description. The 'ID' is 'GIT', 'Name' is 'thiago-jv/\*\*\*\*\* (Credenciais de Acesso ao GIT)', 'Kind' is 'Username with password', and 'Description' is 'Credenciais de Acesso ao GIT'. The top navigation bar includes links for Painel de controle, Gerenciar Jenkins, Credentials, System, and Global credentials (unrestricted).

## 8.2 Criando nossa pipeline

- Criando nossa Pepiline conforme o caminho indicado na imagem abaixo.



- Vamos agora realizar as configurações da nossa Pipelne no Jenkins.

The screenshot shows a Jenkins interface for a pipeline named "projeto\_primeira\_pipeline". The top navigation bar includes links for "Painel de controle" and the pipeline name. Below this, a card displays pipeline status information. A red box highlights the "Configurar" button, which corresponds to the "Configure" option in the original text. Other visible options include "Status", "Changes", "Construir agora" (Build now), "Excluir Pipeline" (Delete Pipeline), and "Full Stage View".

Painel de controle > projeto\_primeira\_pipeline >

Status

</> Changes

▷ Construir agora

**⚙️ Configurar**

🗑️ Excluir Pipeline

🔍 Full Stage View

- Procure a sessão “Pepiline” e faça as configurações conforme modelo abaixo.

## Pipeline

Configuração

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

`https://github.com/thiago-jv/SPRINGBOOT_JENKINS`

Credentials ?

`thiago-jv/******** (GIT HUB)`

+ Add

Avançado ▾

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

`*/main`

Add Branch

Navegar no repositório ?

(Auto)

Additional Behaviours

Adicionar ▾

Script Path ?

`jenkins/Jenkinsfile`

- **Pipeline script from SCM:** Essa opção indica que o script do pipeline será obtido diretamente do repositório, e não digitado manualmente na interface do Jenkins.
- **SCM:** Aqui você escolhe o tipo de SCM que será utilizado. No seu caso, foi selecionado o Git.
- **Repositories:** Nesta seção, você define o repositório Git onde se encontra o script do pipeline.
- 
- **Repository URL:** O endereço completo do repositório no GitHub.
- 
- **Credentials:** As credenciais necessárias para acessar o repositório (usuário e senha ou token).
- **Branches to build:** Define quais branches do repositório serão monitorados para executar o pipeline.
- 
- **Branch Specifier:** A expressão que define quais branches serão incluídos. No seu caso, \*/main indica que o pipeline será executado para qualquer branch que comece com "main".
- **Script Path:** O caminho completo do arquivo que contém o script do pipeline dentro do repositório.

8.3 Instale esse plugin para ter uma visão melhor da pipeline rodando

The screenshot shows the Jenkins 'Plugins' page. On the left sidebar, under 'Extensões instaladas', the 'Pipeline: Stage View Plugin' is listed. It has a version of 2.34 and a brief description: 'Pipeline Stage View Plugin.' Below it is a link to 'Relatar um problema com esta extensão'. A search bar at the top right contains the text 'Pipeline: Stage View Plugin Versão'.

8.4 Rodando nossa pipeline

The screenshot shows the Jenkins project page for 'projeto\_primeira\_pipeline'. In the left sidebar, the 'Construir agora' button is highlighted with a red box. The main area is titled 'Stage View' and displays a grid of build stages. The columns are labeled 'Declarative: Checkout SCM', 'Build', and 'Deploy'. The first stage (#4) took 15s and had 'No Changes'. The second stage (#3) took 15s and had 'No Changes'. The third stage (#2) took 15s and had 1 commit. The average stage time is 15s, and the average full run time is approximately 39s. Below the grid, there's a section for 'Links permanentes' with links to the last successful builds and an atom feed.

#	Data/Hora	Declarative: Checkout SCM	Build	Deploy
#4	out. 11 09:57	15s	7s	369ms
#3	out. 11 09:52	15s	5s	350ms
#2	out. 11 09:20	15s	5s	340ms

Average stage times:  
(Average full run time: ~39s)

Links permanentes

- Última construção (#4), 2 min 39 seg atrás
- Última construção estável (#4), 2 min 39 seg atrás
- Última construção bem sucedida (#4), 2 min 39 seg atrás
- Última construção que falhou (#1), 55 min atrás
- Última construção completada. (#4), 2 min 39 seg atrás

## 8.5 Acessando nossa API <http://localhost:8084/jenkinsapi/swagger-ui/index.html#/>

The screenshot displays two browser windows side-by-side. The left window is the Jenkins Pipeline interface for the project 'projeto\_primeira\_pipeline'. It shows a 'Stage View' with several stages: 'Declarative: Checkout SCM' (15s), 'Build' (6s), and 'Deploy' (407ms). The Deploy stage has a red background, indicating it failed. The right window is the Swagger UI for the Jenkins API, specifically the 'hello-jenkins' endpoint. It shows a 'GET /api/hello' request with a 'Parameters' section (empty) and an 'Execute' button. Below it, the 'Responses' section shows a curl command and a successful response body: 'Hello, World!'. The URL in the address bar is `localhost:8084/jenkinsapi/swagger-ui/index.html#/jenkinsapi/v3/api-docs`.

## 8.6 Realizando configuração do plugin do sonar no jenkins

- SonarQube Scanner
- Sonar Quality Gates

The screenshot shows the Jenkins Plugins page. The search bar contains 'Sonar'. Two plugins are listed: 'SonarQube Scanner' (version 2.17.2) and 'Sonar Quality Gates' (version 315.v1f12b\_e61a\_3a\_4). Both have an 'Instalar' (Install) button highlighted with a red box. The SonarQube Scanner plugin was last updated 7 months and 25 days ago. The Sonar Quality Gates plugin was last updated 1 month and 15 days ago. A warning message at the bottom of the SonarQube Scanner card states: 'Aviso: esta versão de extensão pode não ser segura para usar. Por favor revise as seguintes notas de segurança: • Credentials transmitted in plain text'.

**Plugins**

**Andamento do download**

Preparação

- Checando conexão com a Internet
- Verificando conectividade com o centro de atualização
- Sucesso

SonarQube Scanner	Sucesso
Sonar Quality Gates	Sucesso
Carregando extensões plugáveis	Sucesso

→ [Voltar para a página principal](#)  
(você pode começar a usar os plugins instalados imediatamente)

Reinicie o Jenkins quando a instalação estiver completa e nenhuma tarefa estiver em execução

## 8.7 Realizar criação do token no sonar para o Jenkins ter acesso

A My Account

localhost:9000/account

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects... A

Administrator Profile Security Notifications Projects

My Account Log out

Login	admin
Groups	sonar-administrators sonar-users
SCM Accounts	admin

- Vamos gerar um token para referenciar no Jenkins

A Administrator Profile Security Notifications Projects

Tokens

If you want to enforce security by not providing credentials of a real SonarQube user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

Generate Tokens

jenkins

Name	Last use	Created
No tokens		

---

## Generate Tokens

Generate

New token "jenkins" has been created. Make sure you copy it now, you won't be able to see it again!

Copy

eb0361e04fa153c77c65986870e0f4bcaff2577c

Name	Last use	Created
------	----------	---------

## 8.8 Realizando configuração do SonarQube servers no Jenkins

- Siga o caminho conforme imagem e encontre a sessão exibida.
- A configuração abaixo serve para conectar o Jenkins ao SonarQube. Essa integração é fundamental para automatizar a análise de qualidade do código dentro do seu processo de desenvolvimento de software.

The screenshot shows the Jenkins system configuration page under the 'SonarQube servers' section. A red box highlights the 'SonarQube servers' heading. Below it, there's a note about injecting environment variables and a checkbox for 'Environment variables'. A second red box highlights the 'Name' field, which contains 'SONAR\_LOCAL'. A third red box highlights the 'Server URL' field, which contains 'http://localhost:9000'. A fourth red box highlights the 'Server authentication token' section, which shows a dropdown set to '- none -' and a '+ Add' button, with a Jenkins icon and the word 'Jenkins' next to it. At the bottom, there are 'Salvar' and 'Aplicar' buttons.

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as

Environment variables

SonarQube installations

List of SonarQube installations

Name

SONAR\_LOCAL

Server URL

Default is `http://localhost:9000`

`http://localhost:9000`

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

- none -

+ Add

Jenkins

Add SonarQube

Salvar

Aplicar

- Ao clicar em Add Jenkins na imagem anterior essa outra é exibida para informar o token configurado anteriormente no sonar.

The screenshot displays two overlapping web application windows:

- Jenkins Credentials Provider: Jenkins** (Left Window):
  - Add Credentials**
  - Domain**: Global credentials (unrestricted)
  - Kind**: Secret text
  - Scope**: Global (Jenkins, nodes, items, all child items, etc)
  - Secret**: A red box highlights the secret value, which is a long string of asterisks (.....).
  - ID**: TOKEN SONAR
  - Description**: TOKEN SONAR
- Security - My Account** (Right Window):
  - Tokens**
  - A message: "If you want to enforce security by not providing credentials of a real SonarQube user to your code scan or to invoke web services, you can provide a User Token as the user login. This will increase the security of your installation by not letting user's password going through your network."
  - Generate Tokens** button
  - Tokens Table**:
 

Name	Last use	Created
jenkins	Never	October 11, 2024
  - An alert message: "New token "jenkins" has been created. Make sure you copy it now before to see it again!" with a **Copy** button and the token value: eb9362e84fa153c7/c659868/0ef4bcff25/c.
  - Cancel** and **Add** buttons at the bottom right.

## 8.9 Realizando configuração do Quality Gates – SonarQube no Jenkins

- Agora, vamos configurar o Quality Gate para analisar nosso código e identificar a presença de bugs, code smells, duplicações e outros problemas.

The screenshot displays two browser windows side-by-side. The left window is titled 'Quality Gates - Sonarqube' and shows the configuration of a new Quality Gate named 'SONAR\_LOCAL\_QG'. It includes fields for 'SonarQube Server URL' (set to 'http://localhost:9000') and 'SonarQube account token' (set to '\*\*\*\*\*'). The right window is titled 'Security - My Account' and shows the 'Tokens' section. A new token named 'jenkins' has been generated, with the value 'e98361e04fa153c77c5598870e0f4bcff2577c'. This token is highlighted with a red box and an arrow points from it to the 'SonarQube account token' field in the Jenkins configuration window, indicating that the token should be used instead of the user and password.

- Selecione a configuração realizada em um passo anterior.

The screenshot shows the Jenkins System configuration page under the 'SonarQube servers' section. A red box highlights the 'Server authentication token' field, which contains the value 'TOKEN SONAR'. This field is described as mandatory when anonymous access is disabled.

**SonarQube servers**

If checked, job administrators will be able to inject a SonarQube server configuration as an environment variable.

Environment variables

**SonarQube installations**

List of SonarQube installations

Name: SONAR\_LOCAL

Server URL: Default is <http://localhost:9000>  
http://localhost:9000

Server authentication token: SonarQube authentication token. Mandatory when anonymous access is disabled.  
TOKEN SONAR

+ Add

Avançado ▾

Add SonarQube

Salvar Aplicar

### 8.9.1 Realizando configuração do SonarQube Scanner instalações no Jenkins

- Realiza configurações conforme caminho especificado.

The screenshot shows the Jenkins 'Manage Tools' configuration page at `localhost:8080/manage/configureTools/`. The 'SonarQube Scanner instalações' section is highlighted with a red box. Inside this section, the 'Nome' field containing 'SONAR\_SCANNER' and the 'Instalar automaticamente' checkbox are also highlighted with red boxes. Below this section, there's a 'Install from Maven Central' section with a 'Versão' field set to 'SonarQube Scanner 6.2.1.4610'. At the bottom of the 'SonarQube Scanner' section is a 'Save' button, which is also highlighted with a red box. The 'Ant instalações' and 'Maven instalações' sections are partially visible below.

SonarQube Scanner instalações

Nome: SONAR\_SCANNER

Instalar automaticamente

Install from Maven Central

Versão: SonarQube Scanner 6.2.1.4610

Adicionar instalador

Save

Ant instalações

Adicionar Ant

Maven instalações

### 8.9.2 Realizando criação do projeto no SonarQube

- Realizar acesso <http://localhost:9000>.

The screenshot shows the 'How do you want to create your project?' screen. A red box highlights the 'Manually' option, which is the selected path.

Below, the 'Create a project' page is shown with fields for 'Project display name' (BackEnd) and 'Project key' (BackEnd). Both fields have red boxes around them, indicating they are required. A green checkmark icon is visible next to each field.

A 'Set Up' button is at the bottom left.

The screenshot shows the SonarQube dashboard for the 'BackEnd' project. A red box highlights the 'Locally' analysis option under 'How do you want to analyze your repository?'. Other options include 'With Azure Pipelines', 'With Bitbucket Pipelines', 'With GitHub Actions', 'With GitLab CI', 'With Jenkins', and 'Other CI'.

- Siga os passos sugeridos e informa a informação abaixo.

The screenshot shows the SonarQube interface for the 'BackEnd' project. The URL is `localhost:9000/dashboard?id=BackEnd&selectedTutorial=manual`. The top navigation bar includes 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. Below the navigation, it shows 'BackEnd' with a star icon and 'master'. The main menu has tabs for 'Overview', 'Issues', 'Security Hotspots', 'Measures', 'Code', and 'Activity', with 'Overview' being the active tab. A large central box titled 'Analyze your project' contains the instruction 'We initialized your project on SonarQube, now it's up to you to launch analyses!'. Below this, step 1 'Provide a token' is shown, with a text input field containing 'scanner:a301b02dbba1361309fa705159fb41f742d03413' and a red-bordered 'Continue' button. A note below the token explains its purpose and provides a link to the user account for revocation.

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

scanner:a301b02dbba1361309fa705159fb41f742d03413

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point of time in your [user account](#).

Continue

- Siga passos sugeridos conforme imagem abaixo.

The screenshot shows the SonarQube interface for the 'BackEnd' project. The URL is `localhost:9000/dashboard?id=BackEnd&selectedTutorial=manual`. The top navigation bar includes 'sonarqube' logo, 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Admin'. Below the header, the project name 'BackEnd' is displayed with a star icon and a 'master' branch indicator. A blue button labeled 'Provide a token' is visible. The main content area is titled 'Analyze your project' with the sub-instruction 'We initialized your project on SonarQube, now it's up to you to launch analyses!'. Step 1, 'Provide a token', is shown with a grey background. Step 2, 'Run analysis on your project', is shown with a white background. A question 'What option best describes your build?' is followed by four buttons: 'Maven' (highlighted with a red box), 'Gradle', '.NET', and 'Other (for JS, TS, Go, Python, PHP, ...)'. Below this, the section 'Execute the Scanner for Maven' contains the command: `mvn sonar:sonar \ -Dsonar.projectKey=BackEnd \ -Dsonar.host.url=http://localhost:9000 \ -Dsonar.login=a301b02dbba1361309fa705159fb41f742d03413`, which is also highlighted with a red box.

- Aqui, estou mostrando como são feitas as configurações para que o Sonar realize as análises em nosso projeto e onde passo no jenkinsfile.

The screenshot shows a Jenkinsfile editor at the top and a SonarQube web interface below it. Red arrows highlight specific lines of code in the Jenkinsfile that correspond to configuration settings in the SonarQube UI.

**Jenkinsfile (Top Panel):**

```

4  SONAR_QUBE_URL = 'http://localhost:9000'
5  SONAR_QUBE_TOKEN = 'a301b02dbba1361309fa705159fb41f742d03413'
6
7  stages {
8      stage('Build') {
9          steps {
10             dir('jenkins') {
11                 sh 'mvn clean package'
12             }
13         }
14     }
15     stage('SonarQube Analysis') {
16         steps {
17             withSonarQubeEnv('SONAR_LOCAL') {
18                 sh "${tool 'SONAR_SCANNER'}/bin/sonar-scanner -Dsonar.projectKey=BackEnd -Dsonar.host.url=${env.SONAR_QUBE_URL} -Dsonar.login=${env.SONAR_QUBE_TOKEN} -Dsonar.java.binaries=."
19             }
20         }
21     }
22 }

```

**SonarQube Web Interface (Bottom Panel):**

The SonarQube interface shows a project named "BackEnd". In the "Maven" tab, there is a section titled "Execute the Scanner for Maven" with the following command:

```
mvn sonar:sonar \
-Dsonar.projectKey=BackEnd \
-Dsonar.host.url=http://localhost:9000 \
-Dsonar.login=a301b02dbba1361309fa705159fb41f742d03413
```

Red arrows point from the Jenkinsfile code to the "sonar:sonar" command and the "-Dsonar.host.url" parameter in the SonarQube UI command line.

- Neste trecho, explico a importância de cada configuração feita anteriormente, bem como onde elas são utilizadas e configuradas no Jenkinsfile.

The screenshot shows a Jenkins pipeline editor with a Jenkinsfile snippet. A red arrow points from the 'SONAR\_LOCAL' environment variable reference in the Jenkinsfile to a configuration screen in a browser window titled 'SonarQube servers'. Another red arrow points from the 'SONAR\_SCANNER' tool reference in the Jenkinsfile to another configuration screen in a browser window titled 'SonarQube Scanner instalações'.

```

stage('SonarQube Analysis') {
    steps {
        withSonarQubeEnv('SONAR_LOCAL') {
            sh "${tool 'SONAR_SCANNER'}/bin/sonar-scanner"
        }
    }
}

```

**Jenkinsfile Snippet:**

```

stage('SonarQube Analysis') {
    steps {
        withSonarQubeEnv('SONAR_LOCAL') {
            sh "${tool 'SONAR_SCANNER'}/bin/sonar-scanner"
        }
    }
}

```

**SonarQube servers Configuration (Top Window):**

- checkbox: If checked, job administrators will be able to inject environment variables
- Server URL: localhost:9000
- Name: SONAR\_LOCAL

**SonarQube Scanner installations Configuration (Bottom Window):**

- Add SonarQube Scanner
- Scanner Name: SONAR\_SCANNER
- Install automatically: checked

- stage('SonarQube Analysis')**: Esta é uma etapa do pipeline que executa a análise de código usando o SonarQube. O nome da etapa é "SonarQube Analysis".
- steps**: Dentro da etapa, são definidas as ações que devem ser executadas.
- withSonarQubeEnv('SONAR\_LOCAL')**: Este comando configura o ambiente para a execução do SonarQube. O parâmetro 'SONAR\_LOCAL' refere-se a uma configuração de servidor SonarQube previamente definida no Jenkins, que contém informações como a URL do servidor e as credenciais necessárias.
- sh**: Este comando executa um script shell. Aqui, ele é utilizado para rodar o scanner do SonarQube.
- "\${tool 'SONAR\_SCANNER'}/bin/sonar-scanner"**: Este trecho chama a ferramenta sonar-scanner, que é utilizada para realizar a análise do código. O \${tool 'SONAR\_SCANNER'} busca o caminho da instalação do scanner no Jenkins.
- Dsonar.projectKey=BackEnd**: Define a chave do projeto no SonarQube, que neste caso é "BackEnd". Essa chave é usada para identificar o projeto dentro do SonarQube.
- Dsonar.host.url=\${env.SONAR\_QUBE\_URL}**: Define a URL do servidor SonarQube, onde \${env.SONAR\_QUBE\_URL} é uma variável de ambiente que contém essa URL.
- Dsonar.login=\${env.SONAR\_QUBE\_TOKEN}**: Define o token de login para autenticação no SonarQube, onde \${env.SONAR\_QUBE\_TOKEN} é outra variável de ambiente que armazena esse token.

- **-Dsonar.java.binaries=./**: Especifica o caminho para os arquivos binários do Java que devem ser analisados. O ./ indica que os arquivos binários estão no diretório atual.

```

    stage('Quality Gate') {
        steps {
            sleep(10)
            timeout(time: 1, unit: 'MINUTES') {
                waitForQualityGate abortPipeline: true
            }
        }
    }

```

stage('Quality Gate'):

- **Descrição:** Inicia uma nova etapa no pipeline chamada "Quality Gate" (Portão de Qualidade). Esta etapa é crucial para garantir que o código atenda aos critérios de qualidade estabelecidos.

steps:

- **Descrição:** Indica o início de um bloco onde as etapas específicas desta fase serão definidas.

sleep(10):

- **Descrição:** Faz com que o pipeline aguarde 10 segundos antes de prosseguir para a próxima etapa. Isso pode ser útil para permitir que o SonarQube finalize o processamento dos resultados da análise.

timeout(time: 1, unit: 'MINUTES'):

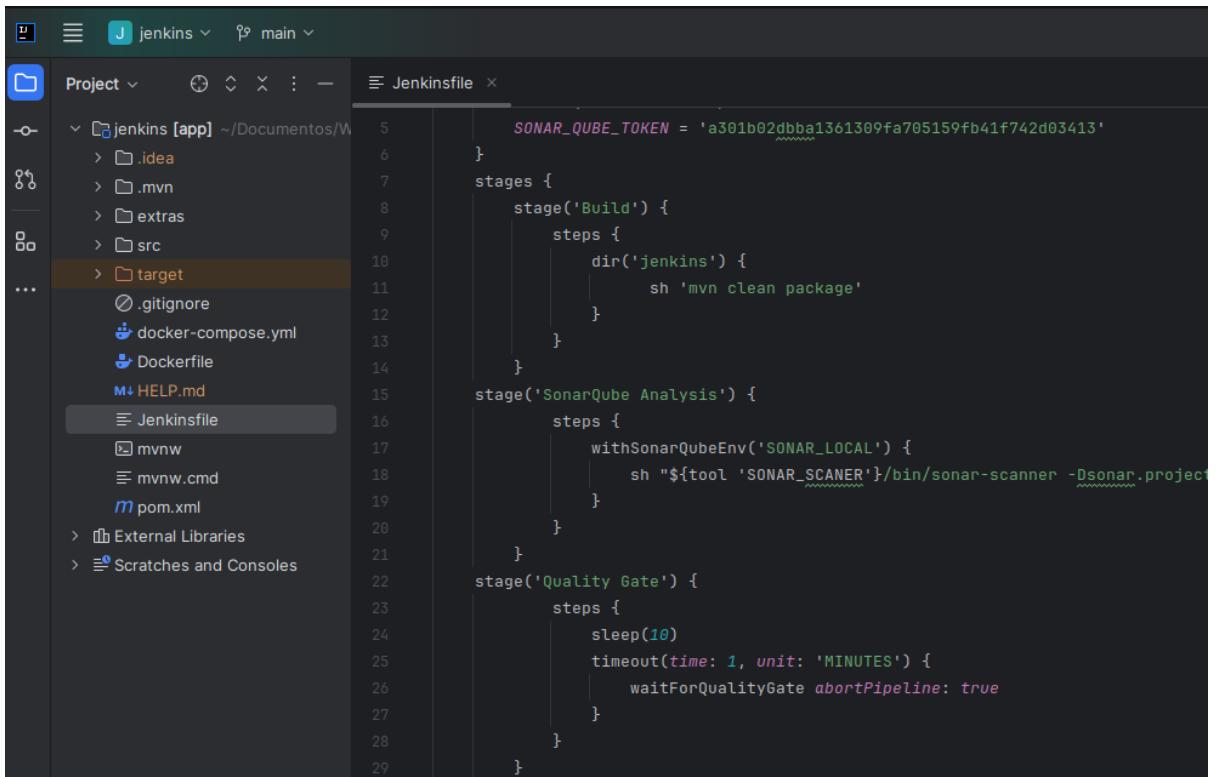
- **Descrição:** Define um tempo limite de 1 minuto para a execução da etapa. Se a etapa não for concluída dentro desse período, o pipeline será interrompido.

waitForQualityGate abortPipeline: true:

- **waitForQualityGate:** Esta instrução faz com que o pipeline aguarde até que o SonarQube retorne o resultado da análise de qualidade.
- **abortPipeline: true:** Se o resultado da análise não atender aos critérios de qualidade definidos no SonarQube, o pipeline será interrompido.

### 8.9.3 Pipeline atualizada

- Agora podemos realizar o resultado final acesso o projeto e verificando arquivo.



```

5      SONAR_QUBE_TOKEN = 'a301b02dbba1361309fa705159fb41f742d03413'
6  }
7  stages {
8      stage('Build') {
9          steps {
10             dir('jenkins') {
11                 sh 'mvn clean package'
12             }
13         }
14     }
15     stage('SonarQube Analysis') {
16         steps {
17             withSonarQubeEnv('SONAR_LOCAL') {
18                 sh "${tool 'SONAR_SCANNER'}/bin/sonar-scanner -Dsonar.projectKey=sonar-project -Dsonar.host.url=https://sonarcloud.io"
19             }
20         }
21     }
22     stage('Quality Gate') {
23         steps {
24             sleep(10)
25             timeout(time: 1, unit: 'MINUTES') {
26                 waitForQualityGate abortPipeline: true
27             }
28         }
29     }
}

```

## 9. Rodando a Pipeline no Jenkins

- Acesse o Jenkins através do endereço local (normalmente <http://localhost:8080>) e procure pelo seu projeto. Clique no botão 'Construir agora' para iniciar a execução do pipeline.

The screenshot shows the Jenkins Pipeline execution page for the project 'projeto\_primeira\_pipeline'. On the left, there's a sidebar with options like Status, Changes, Construir agora, Configurar, Excluir Pipeline, Full Stage View, SonarQube (which is highlighted with a red box), Stages, Renomear, and Pipeline Syntax. The main area is titled 'Stage View' and shows the execution of stage #10. It includes a timeline chart with average stage times: Checkout SCM (16s), Build (7s), SonarQube Analysis (1min 40s), Quality Gate (10s), and Deploy (353ms). Below the timeline is a summary card for the SonarQube Quality Gate, which is labeled 'BackEnd Passed' and shows 'server-side processing: Success'. A red box highlights the SonarQube analysis section and the Quality Gate summary.

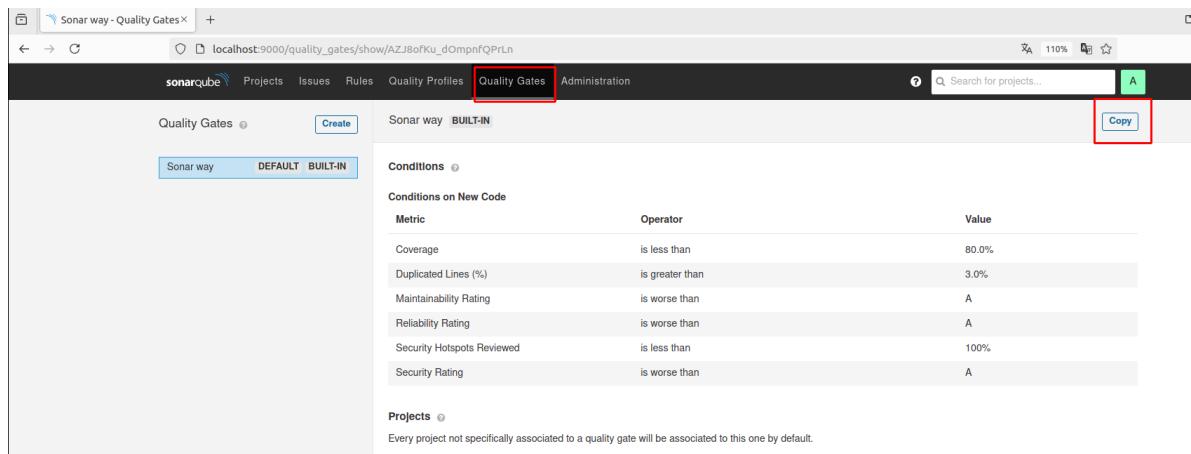
- Acesse o SonarQube, clique no projeto e clique em 'Overall Code' para visualizar o resultado mais detalhado.

The screenshot shows the SonarQube dashboard for the 'BackEnd' project. At the top, it says 'Last analysis had 1 warning' and 'October 11, 2024 at 2:57 PM Version not provided'. The main area is titled 'Overall Code' under 'MEASURES'. It shows the following details:

- Quality Gate Status:** Passed (All conditions passed).
- Measures:**
  - New Code: 0 Bugs (Reliability: A)
  - Overall Code: 0 Vulnerabilities (Security: A)
  - Overall Code: 0 Security Hotspots (Reviewed: —, Security Review: A)
  - Debt: 5min (Reviewed: —, Maintainability: A)
  - Code Smells: 1 (Reviewed: —, Maintainability: A)
  - Coverage: 0.0% (Coverage on 4 Lines to cover, Unit Tests: -)
  - Duplications: 0.0% (Duplications on 118 Lines, Duplicated Blocks: 0)

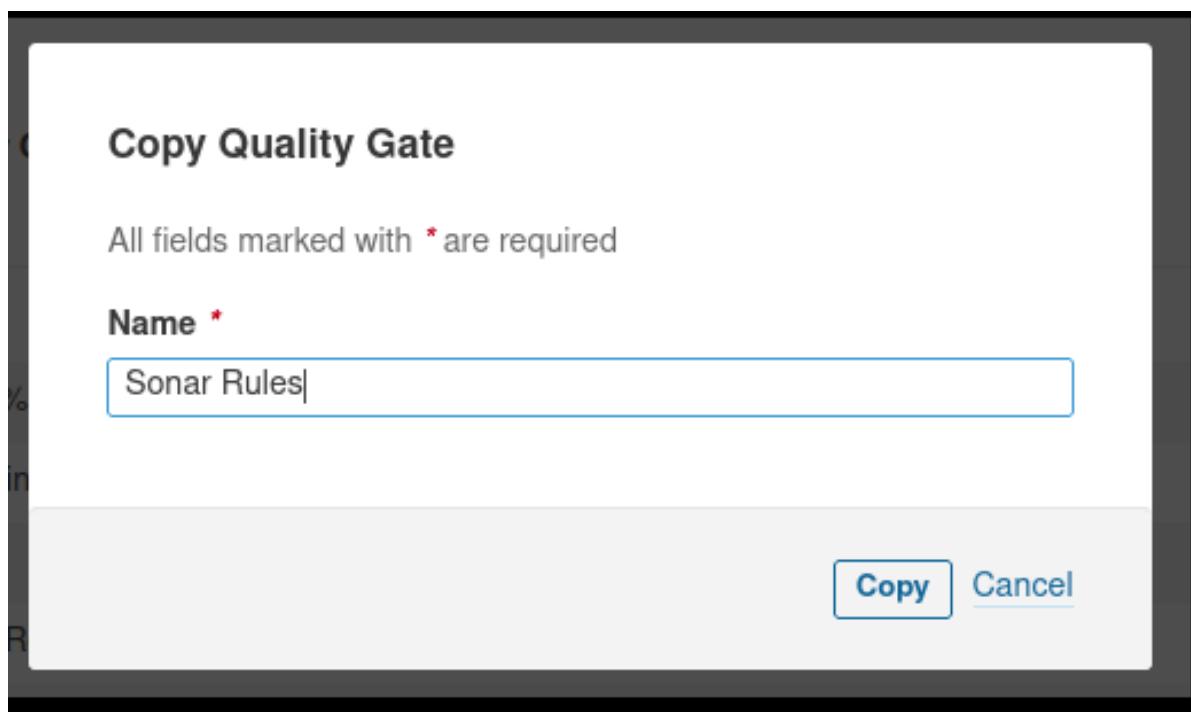
9.1 Agora vamos evoluir um pouco criando a regra para cobertura de código.

- Siga o caminho especificado abaixo para realizar as configurações.
- Vamos criar uma cópia da regra atual já existe.



Screenshot of the SonarQube Quality Gates page. The 'Quality Gates' tab is selected and highlighted with a red box. A 'Copy' button is also highlighted with a red box. The page displays a table of conditions for a 'Sonar way' quality gate, including metrics like Coverage, Duplicated Lines (%), and Reliability Rating, with their operators and values.

- Criando o nome da regra.



Screenshot of the 'Copy Quality Gate' dialog box. It prompts for a name, with 'Sonar Rules' entered in the 'Name \*' field. The 'Copy' button is highlighted with a red box.

- Setando a nova regra com default.

Sonar Rules - Quality Gates

localhost:9000/quality\_gates/show/AZJ9DL4B\_d0mpnfQPwCp

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Quality Gates Sonar Rules Conditions Conditions on New Code Metric Operator Value Edit Delete

**Sonar Rules**

Sonar way DEFAULT BUILT-IN

Rename Copy Set as Default Delete

Add Condition

- Vamos adicionar uma condição na nossa cobertura.

Add Condition

On New Code  On Overall Code

Quality Gate fails when

Search for metrics...

**Coverage**

Condition Coverage  
Conditions to Cover  
**Coverage**  
Line Coverage  
Lines to Cover  
Skipped Unit Tests

Metric	Value	Edit	Delete
Complexity	80.0%		
Cognitive Complexity	3.0%		
Cyclomatic Complexity	A		
Coverage	A		
Condition Coverage	100%		
Conditions to Cover	A		

Add Condition

- Siga os passos e informe que, caso o projeto não tenha 80% de cobertura de teste, ele falhará.

Add Condition

On New Code  On Overall Code

Quality Gate fails when

**Coverage**

Operator is less than

**Value** 80

Add Condition Cancel

is worse than

Metric	Value	Edit	Delete
Complexity	80.0%		
Cognitive Complexity	3.0%		
Cyclomatic Complexity	A		
Coverage	A		
Condition Coverage	100%		
Conditions to Cover	A		

Add Condition

- Podemos ver que nossa regra está definida como padrão e que nossa nova condição foi adicionada.

Metric	Operator	Value	Edit	Delete
Coverage	is less than	80.0%		
Duplicated Lines (%)	is greater than	3.0%		
Maintainability Rating	is worse than	A		
Reliability Rating	is worse than	A		
Security Hotspots Reviewed	is less than	100%		
Security Rating	is worse than	A		

Metric	Operator	Value	Edit	Delete
Coverage	is less than	80.0%		

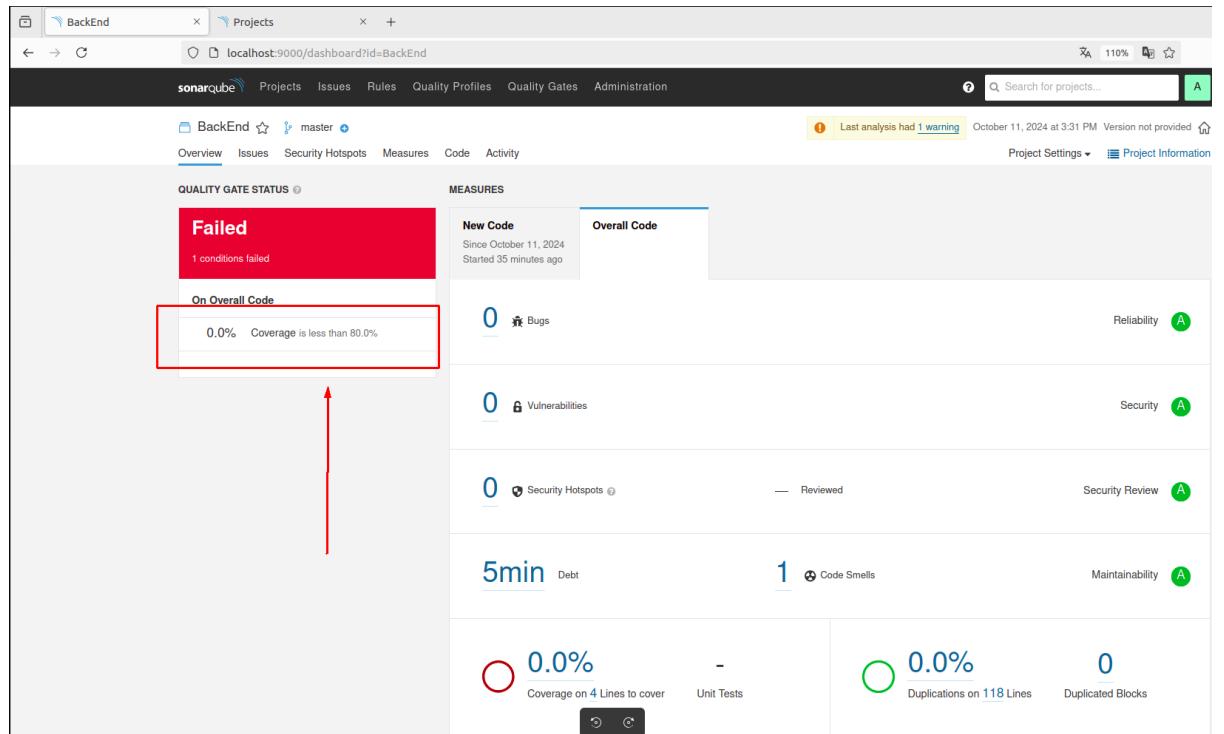
Projects

Every project not specifically associated to a quality gate will be associated to this one by default.

- Vamos construir novamente e visualizar o resultado como uma falha, pois o projeto não possui testes implementados; portanto, a pipeline irá falhar.

Declarative: Checkout SCM	Build	SonarQube Analysis	Quality Gate	Deploy
16s	6s	15s	10s	285ms
#18 out. 11 15:30 No Changes	15s	5s	4s	10s failed 37ms failed
#17 out. 11 15:24 No Changes	17s	5s	4s	10s 357ms

- O Sonar está informando que o projeto não tem cobertura de teste que atenda aos requisitos configurados.



FIM