



Montando um Ambiente Local CI/CD



1. Objetivo do Tutorial

- Criar um ambiente local para simulação de esteira de desenvolvimento com CI/CD.

2. Pré-requisitos

- Java e Maven estejam instalados. Siga as instruções abaixo para configurar as variáveis de ambiente.
- Para instalar o Docker, você pode seguir este guia: [Como instalar e usar o Docker Compose no Ubuntu 20.04](#).
- Guia DBeaver:
<https://snapcraft.io/install/dbeaver-ce/ubuntu>
- Guia Git:
<https://git-scm.com/downloads/linux>
- Para Instalar o Jenkins, você pode seguir este guia:
<https://pkg.jenkins.io/debian-stable/>

3. Instalando Ferramentas

3.1 Instalando o Git

- Execute os seguintes comandos:

```
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

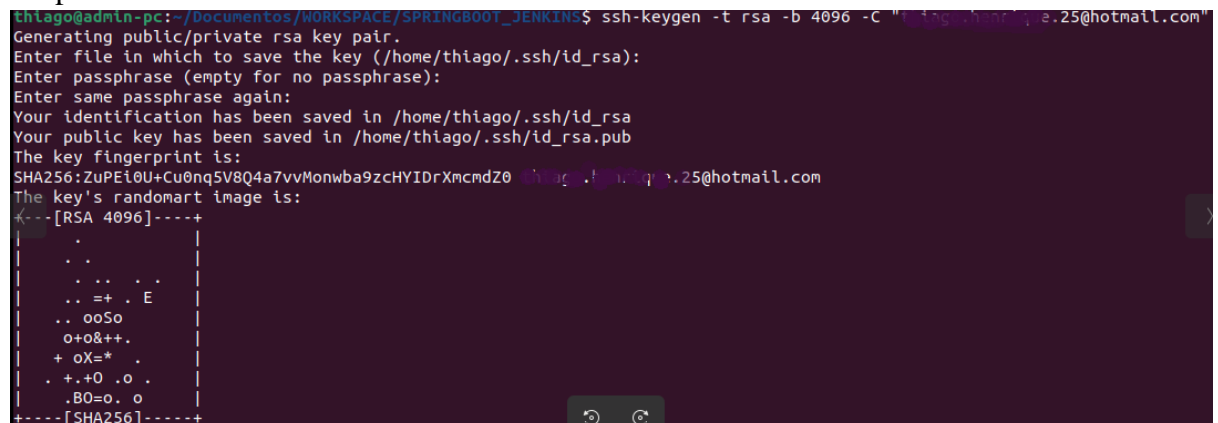
cat .gitconfig

3.2 Configurando chave SSH

- Execute o seguinte comando para gerar a chave SSH:

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

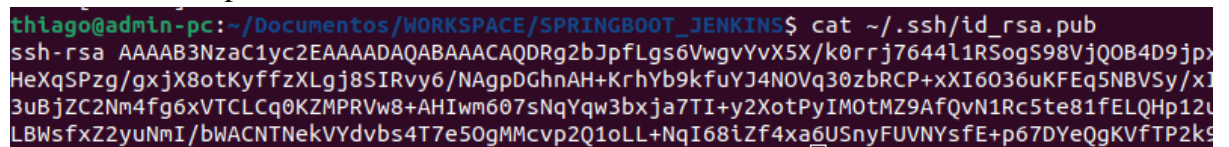
Vá pressionando ENTER até finalizar



```
thiago@admin-pc:~/Documentos/WORKSPACE/SPRINGBOOT_JENKINS$ ssh-keygen -t rsa -b 4096 -C "thiago.bonfatti.e.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:ZuPEi0U+Cu0nq5V8Q4a7vvMonwba9zcHVIDrXmcmdZ0 thiago.bonfatti.e.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:

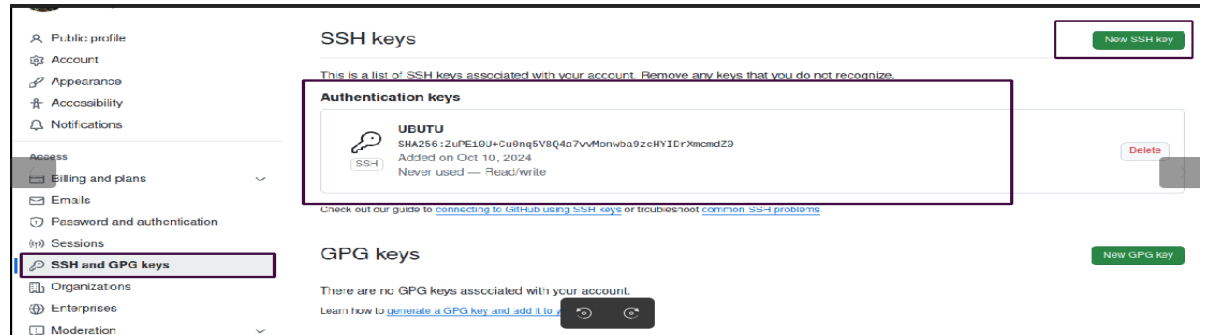
cat ~/.ssh/id_rsa.pub



```
thiago@admin-pc:~/Documentos/WORKSPACE/SPRINGBOOT_JENKINS$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDRg2bJpfLgs6VwgvYvX5X/k0rrj7644l1RSogS98VjQ0B4D9jpx
HeXqSPzg/gxjX8otKyffzXLgj8SIRvy6/NAGpDGhnAH+KrhYb9kfuYJ4NOVq30zbRCP+xxI6036uKFEq5NBVSy/xI
3uBjZC2Nm4fg6xVTCLCqQKZMPRVw8+AHIm607sNqYqw3bxja7TI+y2XotPyIM0tMZ9AfQvN1Rc5te81fELQHp12U
LBWsfxZ2yuNmI/bWACNTNekVYdvbs4T7e50gMMcvp2Q1oLL+NqI68iZf4xa6USnyFUVNysfE+p67DyeQgKVfTP2k9
```

- Configurando a chave SSH no seu GIT

Cole a chave SSH obtida anteriormente



- Testando a conexão

```
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:

```
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

- Execute o comando:

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

3.5 Setando Java Home

- Execute o comando:

sudo nano /etc/profile

```
GNU nano 6.2 /etc/profile
fi
fi
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:

java -version

```
thiago@admin-pc: ~
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 Setando Maven Home

- Execute o comando:

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

```
GNU nano 6.2 /etc/profile
fi
fi
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: ~
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 a algumas Ferramentas

4.1 Jenkins

Jenkins é uma poderosa ferramenta de automação de código aberto, amplamente utilizada no desenvolvimento de software. Ele atua como um servidor que automatiza tarefas repetitivas no ciclo de desenvolvimento, desde a construção do código até sua implantação.

Principais Funcionalidades do Jenkins

- Integração Contínua (CI): Automatiza a construção e os testes do código a cada nova alteração, garantindo que o software esteja sempre em um estado funcional.

- Entrega Contínua (CD): Facilita a entrega do software para ambientes de teste e produção, tornando o processo mais rápido e confiável.
- Automação de Tarefas: Permite a automação de diversas atividades, como geração de relatórios, documentação e execução de scripts.

Benefícios do Jenkins

- Aumento da Eficiência: Libera os desenvolvedores de tarefas repetitivas, permitindo que se concentrem em atividades de maior valor.
- Melhoria da Qualidade: Testes frequentes ajudam a identificar e corrigir bugs rapidamente.
- Redução do Tempo de Lançamento: A entrega do software se torna mais rápida e previsível.
- Maior Colaboração: Facilita o acompanhamento do status das builds e testes entre os membros da equipe.
- Flexibilidade: Altamente configurável, o Jenkins pode ser integrado a diversas ferramentas e tecnologias.

4.2 SonarQube

SonarQube é uma ferramenta essencial para garantir a qualidade do código. Ele realiza uma análise automática do código-fonte, identificando:

- Bugs: Erros que podem causar falhas no software.
- Vulnerabilidades: Fraquezas que podem ser exploradas por atacantes.
- Code Smells: Práticas de programação que dificultam a manutenção.
- Duplicidade de Código: Trechos de código repetidos que podem gerar problemas.
- Cobertura de Testes: Avaliação da qualidade dos testes unitários.

Benefícios do SonarQube

- Melhoria da Qualidade do Software: Identifica e corrige problemas antes que se tornem críticos.
- Aumento da Produtividade: Automatiza a análise de código, permitindo que os desenvolvedores foquem em outras tarefas.
- Redução de Custos: Evita problemas futuros que podem ser caros para corrigir.
- Promoção da Colaboração: Compartilha os resultados da análise com toda a equipe.

4.3 PostgreSQL

PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional, conhecido por sua robustez, confiabilidade e extensibilidade. Ele é amplamente utilizado em aplicações que exigem uma estrutura de dados complexa e integridade referencial.

Principais Características do PostgreSQL

- Suporte a ACID: Garante transações seguras e consistentes.
- Extensibilidade: Permite a criação de tipos de dados personalizados e funções.
- Suporte a JSON: Facilita o armazenamento e a consulta de dados não estruturados.
- Alta Disponibilidade: Oferece recursos como replicação e failover.

4.4 Docker

Docker é uma plataforma de virtualização que permite empacotar aplicações e suas dependências em contêineres leves e portáteis. Com o Docker, você pode garantir que sua aplicação funcione de maneira consistente em diferentes ambientes, desde o desenvolvimento até a produção.

Vantagens do Docker

- Isolamento: Cada contêiner opera de forma isolada, evitando conflitos de dependências.
- Portabilidade: Contêineres podem ser executados em qualquer ambiente que suporte Docker.
- Escalabilidade: Facilita a escalabilidade de aplicações ao permitir a criação de múltiplas instâncias de contêineres.
- Consistência: Garante que a aplicação funcione da mesma maneira, independentemente do ambiente.

4.5 DBeaver

DBeaver é uma ferramenta de gerenciamento de banco de dados universal, que suporta uma ampla variedade de bancos de dados, incluindo PostgreSQL. É especialmente útil para desenvolvedores e administradores de banco de dados, pois oferece uma interface gráfica intuitiva para realizar consultas SQL, gerenciar esquemas de banco de dados e visualizar dados.

Benefícios do DBeaver

- Interface Amigável: Facilita a navegação e a manipulação de dados em diferentes bancos de dados.
- Suporte a Vários Bancos de Dados: Compatível com muitos sistemas de gerenciamento de banco de dados, o que a torna uma ferramenta versátil.

- Ferramentas de Visualização: Oferece gráficos e visualizações para ajudar na análise de dados.
- Recursos Avançados: Inclui recursos como edição de dados em formato de tabela, exportação de dados e execução de scripts SQL.

4.6 Git

O Git é como um álbum de fotos para o seu código. Ele registra todas as mudanças que você faz, permitindo que você:

- Volte no tempo: Recuperar versões anteriores do seu projeto.
- Colabore: Trabalhar com outras pessoas no mesmo projeto sem se atrapalhar.
- Experimentar: Criar novas versões sem afetar o projeto original.

Conceitos básicos:

- Repositório: Onde tudo é armazenado.
- Commit: Uma "foto" do seu projeto em um determinado momento.
- Branch: Uma cópia do seu projeto para trabalhar em novas funcionalidades.
- Merge: Combinar as mudanças de diferentes branches.

Comandos essenciais:

- `git init`: Cria um novo repositório.
- `git clone`: Copia um repositório existente.
- `git add`: Marca os arquivos para serem adicionados ao próximo commit.
- `git commit`: Cria um novo commit.
- `git push`: Envia as suas alterações para um repositório remoto.
- `git pull`: Baixa as últimas alterações de um repositório remoto.

5. Configuração do Docker Compose

- `docker-compose.yml`: Configuração para Jenkins, SonarQube e PostgreSQL.

5.1 Configuração do Docker Compose

Aqui está o nosso `docker-compose.yml` de configuração do Docker Compose, SonarQube e PostgreSQL:

- Execute o comando após criação do `docker-compose.yml`

```
docker compose up
```

services:

db-postgresql:

container_name: db-postgresql
image: postgres
restart: always
environment:
- POSTGRES_USER=admin
- POSTGRES_PASSWORD=admin
- POSTGRES_DB=bdsona
ports:
- "5432:5432"
volumes:
- pgdata:/var/lib/postgresql/data
networks:
- network-rede-local

sonarqube:
container_name: sonar
image: sonarqube:9.0-community
ports:
- "9000:9000"
networks:
- network-rede-local
environment:
- POSTGRES_USER=admin
- POSTGRES_PASSWORD=admin
- POSTGRES_DB=bdsona
depends_on:
- db-postgresql
volumes:
- sonarqube_conf:/opt/sonarqube/conf
- sonarqube_data:/opt/sonarqube/data
- sonarqube_extensions:/opt/sonarqube/extensions
- sonarqube_bundled-plugins:/opt/sonarqube/lib/bundled-plugins

volumes:
sonarqube_conf:
sonarqube_data:
sonarqube_extensions:
sonarqube_bundled-plugins:
pgdata:

networks:
network-rede-local:

driver:

bridge

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]: This may also be found at: /var/lib/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>.

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

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	** Icons API
<input type="radio"/> Timestampers	<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		

Getting Started

Criar o primeiro usuário administrativo

Nome de usuário:

kubedev

Senha:

Confirmar a senha:

Nome completo:

kubedev

Endereço de e-mail:

thiago.henrique.25@hotmail.com

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

Getting Started

Configuração da instancia

URL do Jenkins:

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.

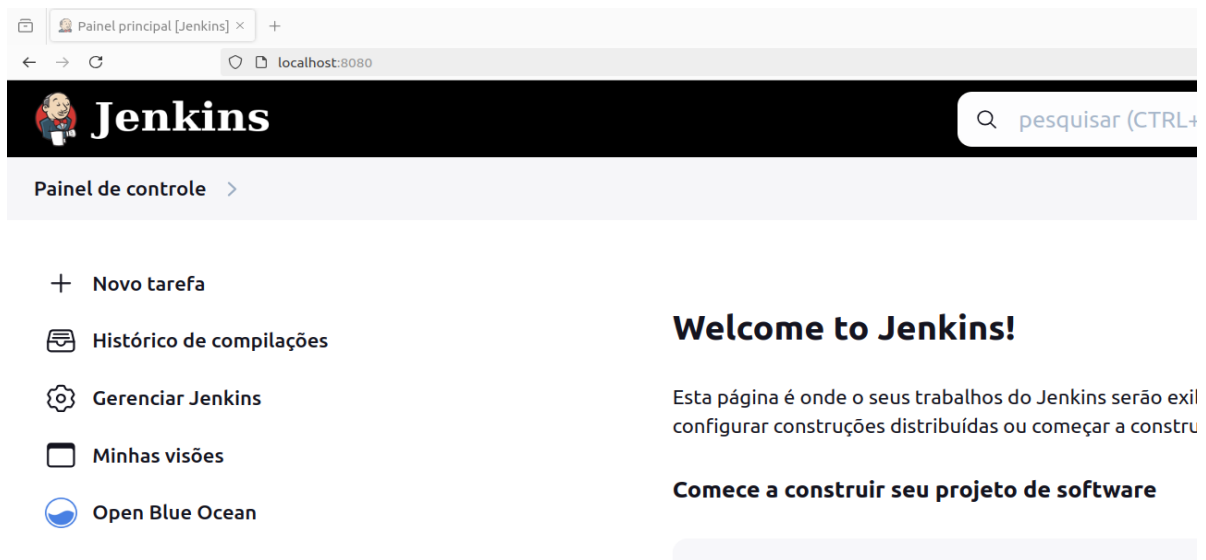
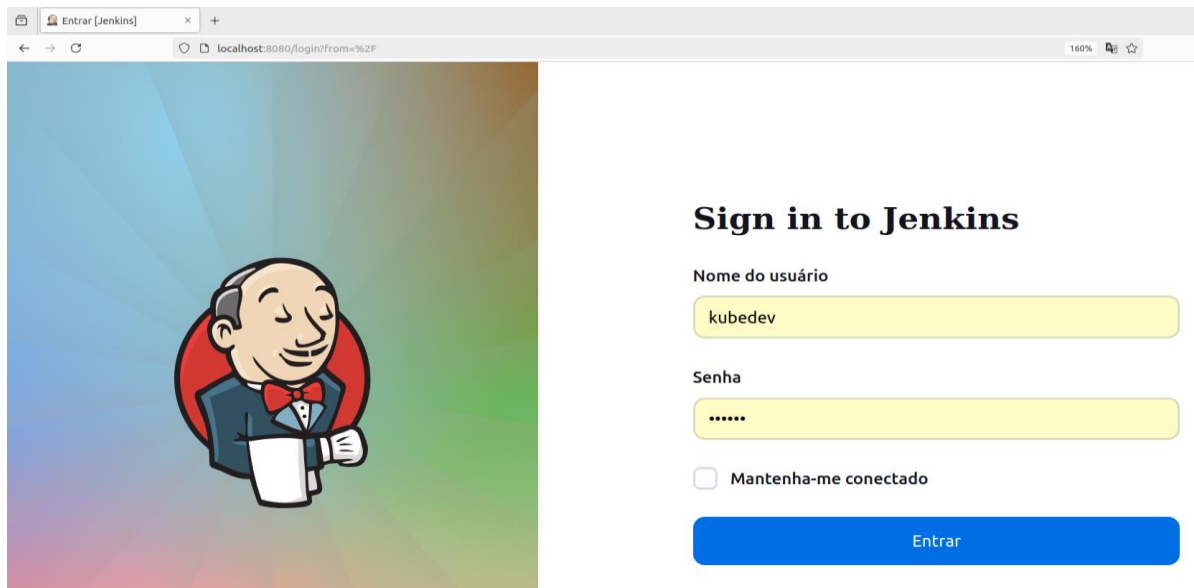
O valor proposto padrão mostrado é **ainda não salvo** e é gerado da solicitação atual, se possível. A melhor prática é configurar este valor para a URL que espera-se que os usuários utilizem. Isto evita confusão quando compartilhando ou vendo links.

Getting Started

Jenkins is almost ready!

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

[Restart](#)

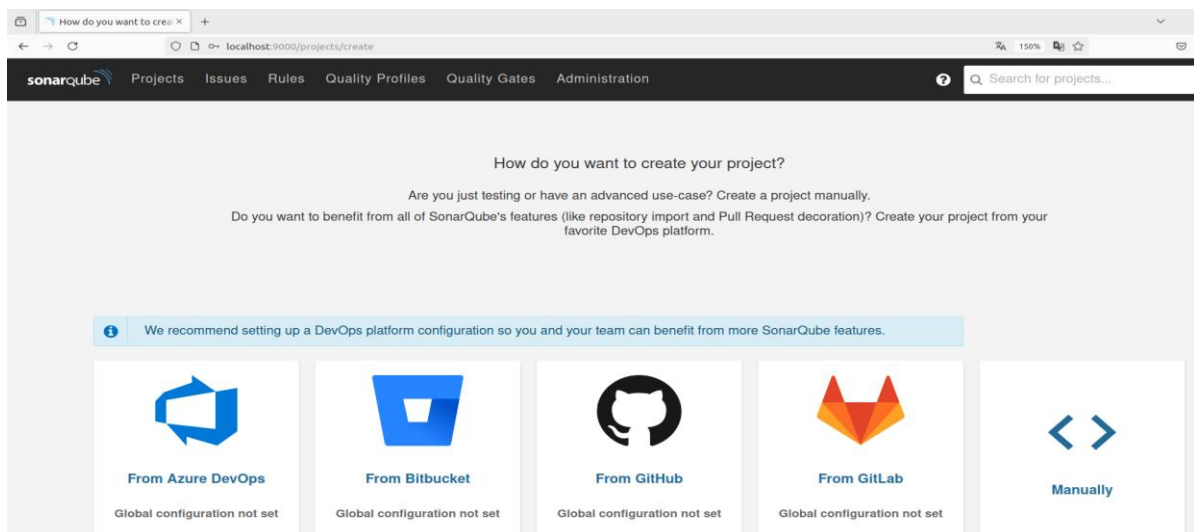
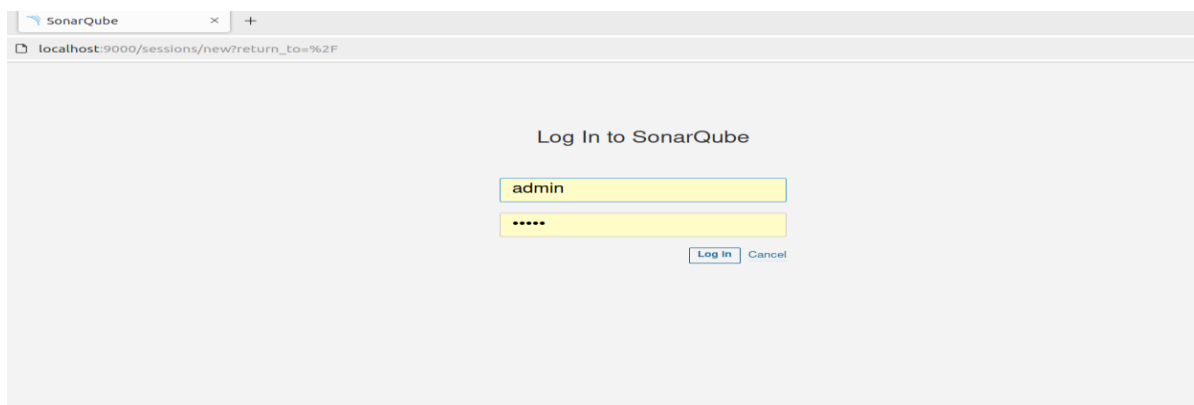


- Instalar o plugin Pipeline Maven Integration

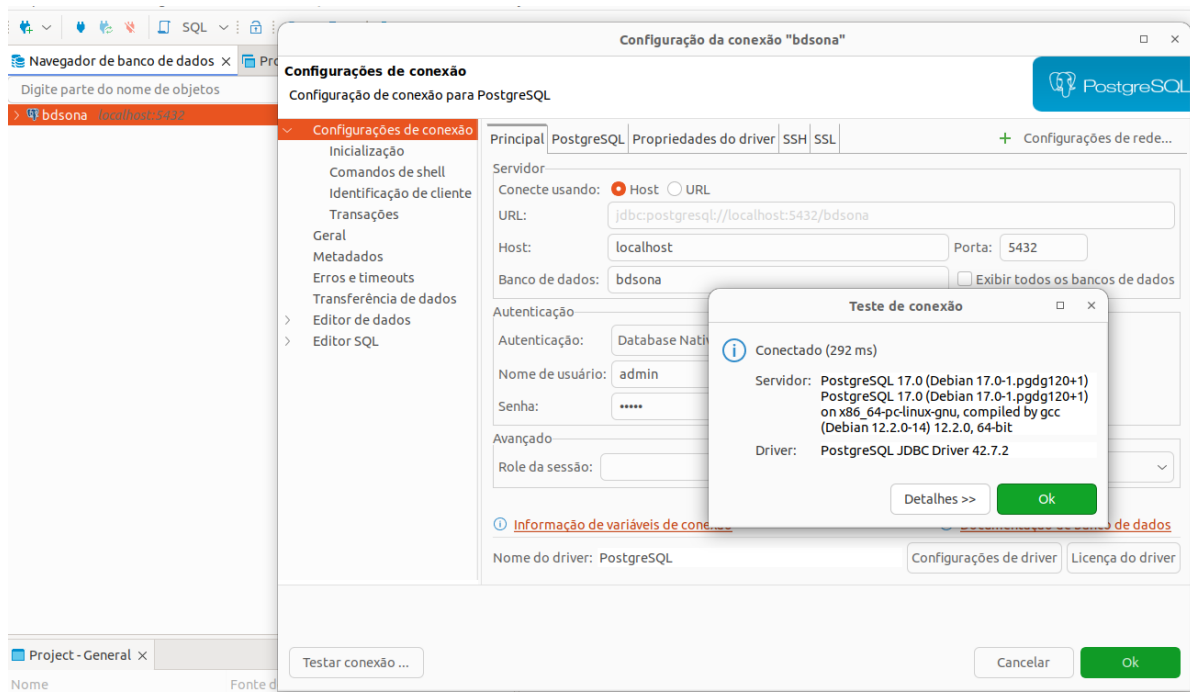


6.2 SonarQube

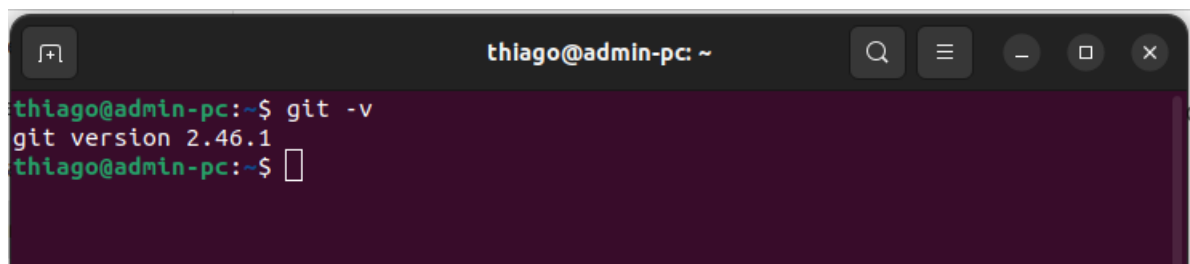
- Abra um navegador e acesse: <http://localhost:9000>.
- Dados de Acesso:
 - Usuário: admin
 - Senha: admin



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. Aqui está o que ele contém:

version: '3.8'

services:

backend:

container_name: backend

restart: always


```
build: .  
ports:  
  - "8084:8084"  
networks:  
  - network-rede-local
```

```
networks:  
  network-rede-local:  
    driver: bridge
```

- version: Especifica a versão do Docker Compose que você está usando.
- 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. Aqui está a estrutura:

```
FROM maven:3.6.3-openjdk-17 AS build  
COPY src /home/app/src  
COPY pom.xml /home/app  
RUN mvn -f /home/app/pom.xml clean package
```

```
FROM openjdk:17  
COPY --from=build /home/app/target/app.war /usr/local/lib/app.war  
EXPOSE 8084
```

```
ENTRYPOINT ["java", "-jar", "/usr/local/lib/app.war"]
```

- 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.
- stage('Deploy'): Esta etapa utiliza o Docker Compose para subir o contêiner com a aplicação.

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  
    http://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>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<project.reporting.outputEncoding>
    UTF-8
</project.reporting.outputEncoding>
<java.version>17</java.version>
</properties>
```

```
<dependencies>
<dependency>
<groupId>org.springdoc</groupId>
<artifactId>springdoc-openapi-ui</artifactId>
<version>1.6.14</version>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-tomcat</artifactId>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>
```

```
<build>
<finalName>app</finalName>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
<executions>
<execution>
<goals>
<goal>repackage</goal>
</goals>
</execution>
</executions>
</plugin>
</plugins>
</build>
<profiles>
<profile>
<id>docker</id>
<build>
<plugins>
<plugin>
<groupId>com.spotify</groupId>
<artifactId>
```

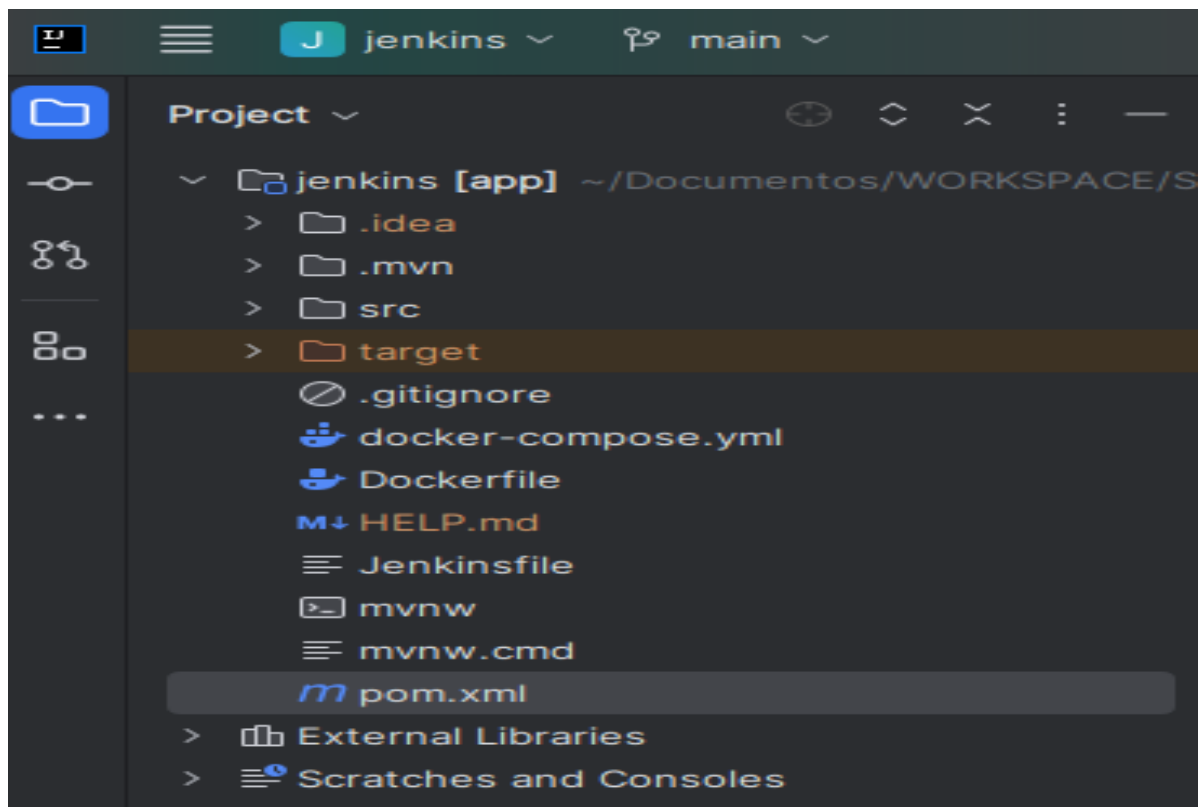
```
dockerfile-maven-plugin
</artifactId>
```

```
<version>1.4.13</version>
<executions>
```

```
<execution>
<id>default</id>
<goals>
<goal>build</goal>
<goal>push</goal>
</goals>
</execution>
</executions>
<configuration>
<repository>jenkins-api</repository>
<buildArgs>
<JAR_FILE>
app.war
</JAR_FILE>
</buildArgs>
</configuration>
</plugin>
</plugins>
</build>
</profile>
</profiles>

</project>
```

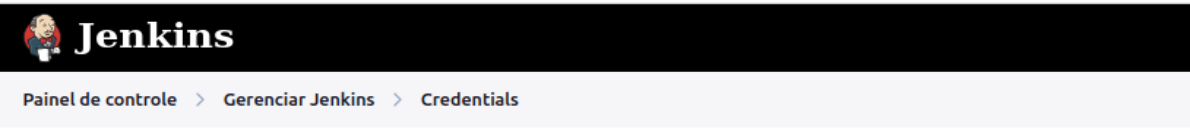
7.5 Estrutura básica do projeto



```
package com.jenkins.controller;  
  
> import ...  
  
@RestController  Thiago Henrique  
@RequestMapping("/api")  
public class HelloJenkins {  
  
    @GetMapping("/hello")  Thiago Henrique  
>    public String hello() { return "Hello, World!"; }  
}
```

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

8.1 Criando credencias de acesso ao github de seu usuário



Credentials

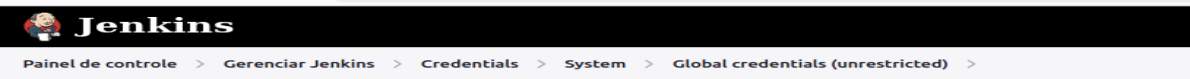
T	P	Store ↓	Domain
---	---	---------	--------

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global) ▼

Add credentials

Ícone: P M G



New credentials

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

thiago-jv

☐ Treat username as secret ?

Password ?

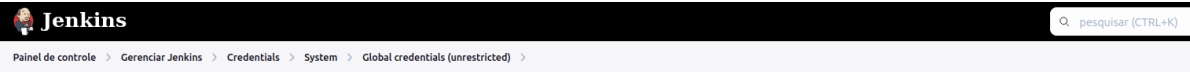
ID ?

GIT

Description ?

Credenciais de Acesso ao GIT

Create




Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
GIT	thiago-jv/***** (Credenciais de Acesso ao GIT)	Username with password	Credenciais de Acesso ao GIT

Ícone: P M G

8.2 Criando nossa pipeline

 **Jenkins**


Painel de controle > Tudo > Novo tarefa


Novo tarefa


Entre com um nome de item


projeto_primeira_pipeline


Select an item type


**Construir um projeto de software de estilo livre.**
Classic, general-purpose job type that checks out from up to one SCM build steps like archiving artifacts and sending email notifications.

**Pipeline**
Orchestrates long-running activities that can span multiple build agents as workflows) and/or organizing complex activities that do not easily

**Construir projeto de múltiplas configurações**
Apropriado para projetos que necessitam de grande número de diferentes ambientes, builds para plataformas específicas, etc.

**Folder**
Creates a container that stores nested items in it. Useful for grouping folder creates a separate namespace, so you can have multiple thing folders.

**Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in

**Organization Folder**
Creates a set of multibranch project subfolders by scanning for repo

Tudo certo

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

Script Path: local onde o jenkins irá buscar nosso arquivo para rodar a pipeline criado, nome_projeto/aquivo_jenkinsfile

Branch: especifica qual qual será a branch a ser baixada para ser utilizada

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

Painel de controle > Gerenciar Jenkins > Plugins

Plugins

Atualizações

Extensões disponíveis

Extensões instaladas

Configurações avançadas

Pipeline: Stage View Plugin Versão

Nome ↓

Pipeline: Stage View Plugin 2.34

Pipeline Stage View Plugin.

Relatar um problema com esta extensão.

8.4 Rodando nossa pipeline

Painel de controle > projeto_primeira_pipeline >

Status

Changes

Construir agora

Configurar

Excluir Pipeline

Full Stage View

Stages

Renomear

Pipeline Syntax

projeto_primeira_pipeline

Stage View

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

	Declarative: Checkout SCM	Build	Deploy
#4 out. 11 09:57 No Changes	15s	7s	369ms
#3 out. 11 09:52 No Changes	15s	5s	350ms
#2 out. 11 09:20 1 commit	15s	5s	340ms

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

Histórico de construções Tendência

Filtro de construções...

#4 11 de out. de 2024 09:57

#3 11 de out. de 2024 09:52

#2 11 de out. de 2024 09:20

#1 11 de out. de 2024 09:04

Atom feed para todos

Atom feed por falhas

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

The image shows two side-by-side browser windows. The left window displays the Jenkins web interface for a pipeline named 'projeto_primeira_pipeline'. The right window displays the Swagger UI for the Jenkins API.

Jenkins Interface (Left Window):

- URL: `localhost:8080/job/projeto_primeira_pipeline/`
- Page Title: **projeto_primeira_pipeline**
- Left Sidebar: Contains navigation links like 'Status', 'Changes', 'Construir agora', 'Configurar', 'Excluir Pipeline', 'Full Stage View', 'Stages', 'Renomear', 'Pipeline Syntax', and 'Histórico de construções'.
- Main Content: 'Stage View' table showing pipeline execution details.

	Declarative: Checkout SCM	Build	Deploy
Average stage times: (Average full run time: ~40s)	15s	6s	407ms
#9 out. 11 11:29 No Changes	16s	6s	601ms
#8 out. 11 11:22 No Changes	16s	6s	371ms
#7 out. 11 11:19 No Changes	15s	5s	380ms
#6 out. 11 11:18 No Changes	15s	5s	376ms
#5 out. 11 11:15 No Changes	15s	7s	472ms
#4 out. 11 11:13 No Changes	15s	7s	369ms
#3 out. 11 09:57 No Changes	15s	5s	350ms
#2 out. 11 09:52 No Changes	15s	5s	340ms
#1 out. 11 09:04 No Changes	15s	5s	340ms

Swagger UI Interface (Right Window):

- URL: `localhost:8084/jenkinsapi/swagger-ui/`
- Page Title: **OpenAPI definition** (v0.0A53)
- Server: `http://localhost:8084/jenkinsapi - Generated server url`
- API Endpoint: `GET /api/hello`
- Parameters: No parameters
- Execute button: `Execute`
- Responses: `200` with response body `Hello, World!`

8.6 Realizando configuração do plugin do sonar no jenkins

SonarQube Scanner e Sonar Quality Gates

The image shows two screenshots of the Jenkins web interface. The top screenshot shows the 'Plugins' page with a search for 'Sonar'. A red box highlights the 'Instalar' (Install) button for 'SonarQube Scanner' and 'Sonar Quality Gates'. The bottom screenshot shows the 'Andamento do download' (Download progress) page. A red box highlights the 'Sucesso' (Success) status for 'SonarQube Scanner', 'Sonar Quality Gates', and 'Carregando extensões plugáveis' (Loading pluggable extensions). Below the progress bar, there are links to 'Voltar para a página principal' (Return to the main page) and a checkbox to 'Reinicie o Jenkins quando a instalação estiver completa e nenhuma tarefa estiver em execução' (Restart Jenkins when installation is complete and no tasks are running).

Jenkins

Painel de controle > Gerenciar Jenkins > Plugins

Plugins

Atualizações

Extensões disponíveis

Extensões instaladas

Configurações avançadas

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

The screenshot shows the SonarQube Administrator interface. At the top, the user is logged in as 'Administrator'. The 'Security' tab is selected. In the 'Tokens' section, there is a text input field containing 'jenkins' and a 'Generate' button. Below this, a table shows no existing tokens. A second screenshot shows the 'Generate Tokens' page with a text input field and a 'Generate' button. Below the input field, a yellow notification box states: 'New token "jenkins" has been created. Make sure you copy it now, you won't be able to see it again!'. Below the notification, there is a 'Copy' button and the token value: 'eb0361e04fa153c77c65986870e0f4bcafff2577c'.

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 **Generate**

Name	Last use	Created
No tokens		

Generate Tokens

Enter Token Name **Generate**

Copy eb0361e04fa153c77c65986870e0f4bcafff2577c

8.8 Realizando configuração do SonarQube servers no Jenkins

System [Jenkins]

Security - My Account

localhost:8080/manage/configure

Painel de controle > Gerenciar Jenkins > System >

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

Apl

SonarQube servers

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

☐ 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

Tools [Jenkins] Security - My Account

localhost:8080/manage/configureTools/

Painel de controle > Gerenciar Jenkins > Tools

SonarQube Scanner instalações

Adicionar SonarQube Scanner

SonarQube Scanner

Nome

SONAR_SCANNER

☒ Instalar automaticamente ?

Install from Maven Central

Versão

SonarQube Scanner 6.2.1.4610

Adicionar instalador ▾

Adicionar SonarQube Scanner

Ant instalações

Adicionar Ant

Maven instalações

Save Apply ↺ ↻

8.9.2 Realizando criação do projeto no SonarQube

The first screenshot shows the 'How do you want to create your project?' page. It offers five options: From Azure DevOps, From Bitbucket, From GitHub, From GitLab, and Manually. The 'Manually' option is highlighted with a red box. A message above the options suggests setting up a DevOps platform configuration for better features.

The second screenshot shows the 'Create a project' form. The 'Project display name' and 'Project key' fields are both filled with 'BackEnd' and are highlighted with red boxes. Green checkmarks indicate the values are valid. The form also includes a 'Set Up' button and a note about character limits and allowed characters for the project key.

The third screenshot shows the 'How do you want to analyze your repository?' page for the 'BackEnd' project. It offers seven analysis methods: Locally, With Azure Pipelines, With Bitbucket Pipelines, With GitHub Actions, With GitLab CI, With Jenkins, and Other CI. The 'Locally' option is highlighted with a red box.

BackEnd x +

localhost:9000/dashboard?id=BackEnd&selectedTutorial=manual

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

BackEnd ☆ master +

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

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
- 2 Run analysis on your project

BackEnd x +

localhost:9000/dashboard?id=BackEnd&selectedTutorial=manual

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

BackEnd ☆ master +

Overview Issues Security Hotspots Measures Code Activity

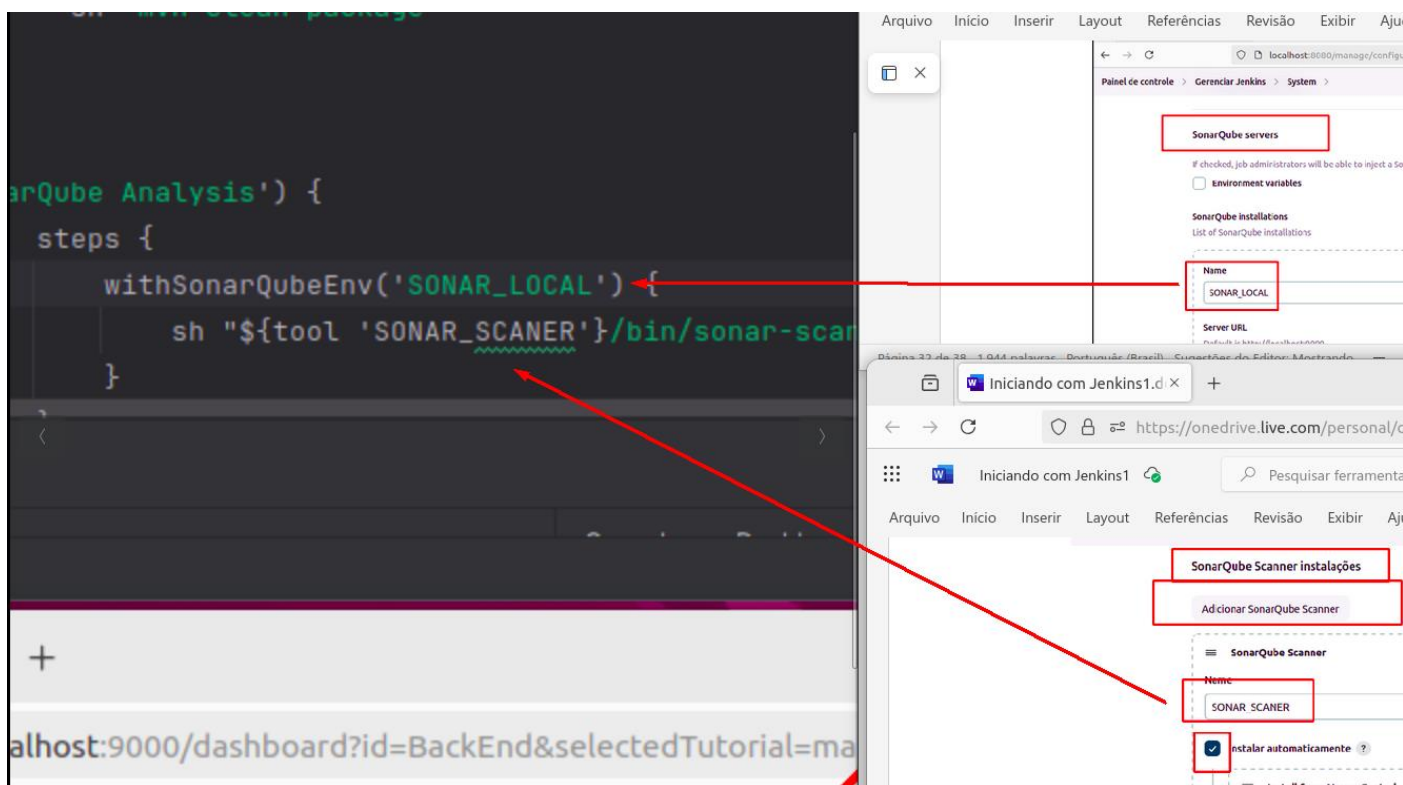
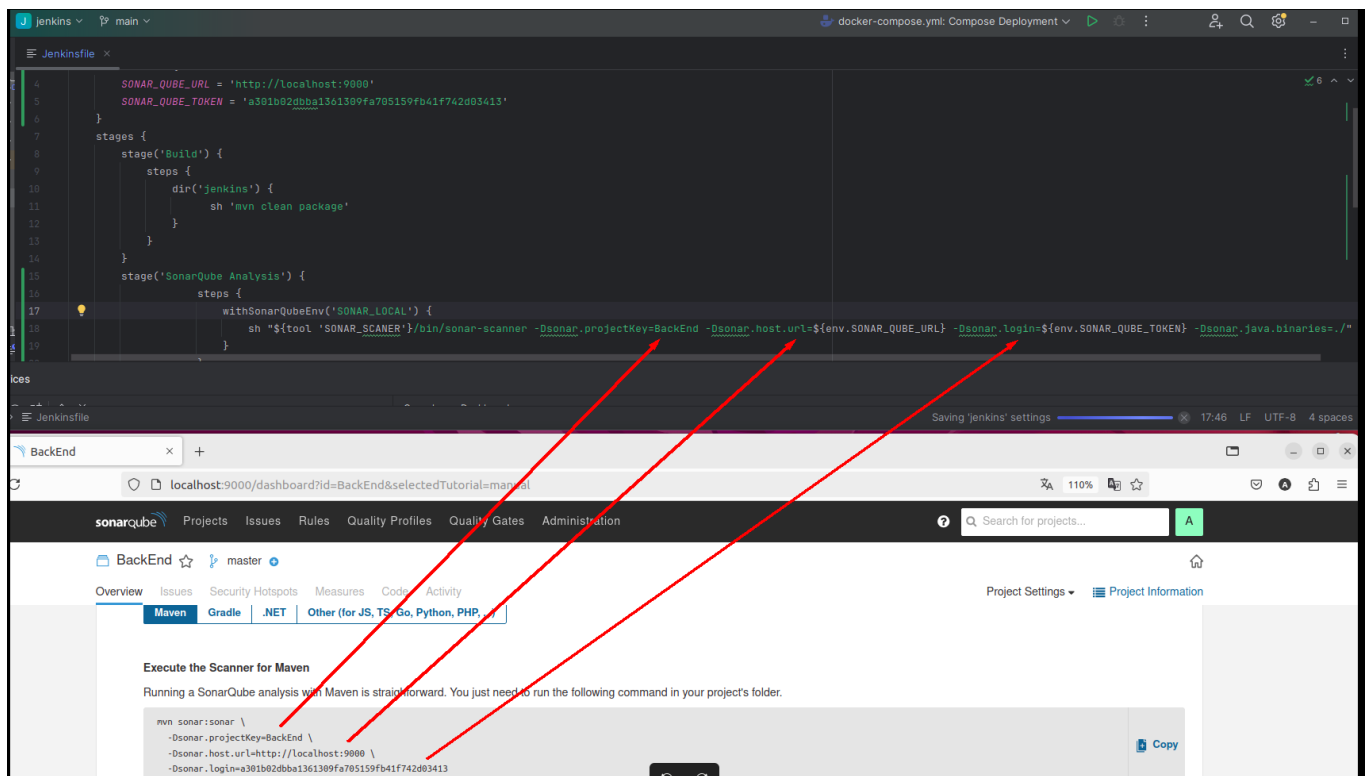
Analyze your project

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

- 1 Provide a token
- 2 Run analysis on your project
What option best describes your build?
Maven Gradle .NET Other (for JS, TS, Go, Python, PHP, ...)
Execute the Scanner for Maven
Running a SonarQube analysis with Maven is straightforward. You just need to run the :

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


Please visit the [official documentation](#) of the Scanner for Maven for more details.



8.9.3 Pipeline atualizada

```

pipeline {
    agent any
    environment {
        SONAR_QUBE_URL = 'http://localhost:9000'
        SONAR_QUBE_TOKEN = 'a301b02dbbq1361309fa705159fb41f742d03413'
    }
    stages {
        stage('Build') {
            steps {
                dir('jenkins') {
                    sh 'mvn clean package'
                }
            }
        }
        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('SONAR_LOCAL') {
                    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="
                }
            }
        }
        stage('Quality Gate') {
            steps {
                sleep(10)
                timeout(time: 1, unit: 'MINUTES') {
                    waitForQualityGate abortPipeline: true
                }
            }
        }
        stage('Deploy') {
            steps {
                dir('jenkins') {
                    sh 'docker compose up -d'
                }
            }
        }
    }
}

```

9. Rodando a Pipeline no Jenkins

projeto_primeira_pipeline x +

localhost:8080/job/projeto_primeira_pipeline/

Jenkins

pesquisar (CTRL+K)

Painel de controle > projeto_primeira_pipeline >

Status

Changes

Construir agora

Configurar

Excluir Pipeline

Full Stage View

SonarQube

Stages

Renomear

Pipeline Syntax

projeto_primeira_pipeline

Stage View

Average stage times:
(Average full run time: ~2min 33s)

Declarative: Checkout SCM	Build	SonarQube Analysis	Quality Gate	Deploy
16s	7s	1min 40s	10s	353ms

SonarQube Quality Gate

BackEnd **Passed**

server-side processing: **Success**

Links permanentes

Historico de construções Tendência

Filtro de construções...

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code

Overall Code

0 Bugs

Reliability **A**

0 Vulnerabilities

Security **A**

0 Security Hotspots

Reviewed

Security Review **A**

5min Debt

1 Code Smells

Maintainability **A**

0.0% Coverage on 4 Lines to cover

Unit Tests

0.0% Duplications on 118 Lines

Duplicated Blocks

9.1 Criando regra para cobertura de código de teste.

Sonar way - Quality Gates

localhost:9000/quality_gates/show/AZJ8ofKu_dOmpnfQPriLn

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministration

Search for projects...

A

Quality Gates

Create

Sonar wayBUILT-IN

Sonar wayDEFAULTBUILT-IN

Sonar wayBUILT-IN

Copy

Conditions

Conditions on New Code

Metric	Operator	Value
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

Projects

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

Copy Quality Gate

All fields marked with * are required

Name *

Sonar Rules

CopyCancel

Sonar Rules - Quality Gat: X

localhost:9000/quality_gates/show/AZJ9DL4B_dOmpnfQPwCp

sonarqube

ProjectsIssuesRulesQuality ProfilesQuality GatesAdministration

Search for projects...

A

Quality Gates

Create

Sonar Rules

Sonar wayDEFAULTBUILT-IN

Sonar Rules

Sonar wayDEFAULTBUILT-IN

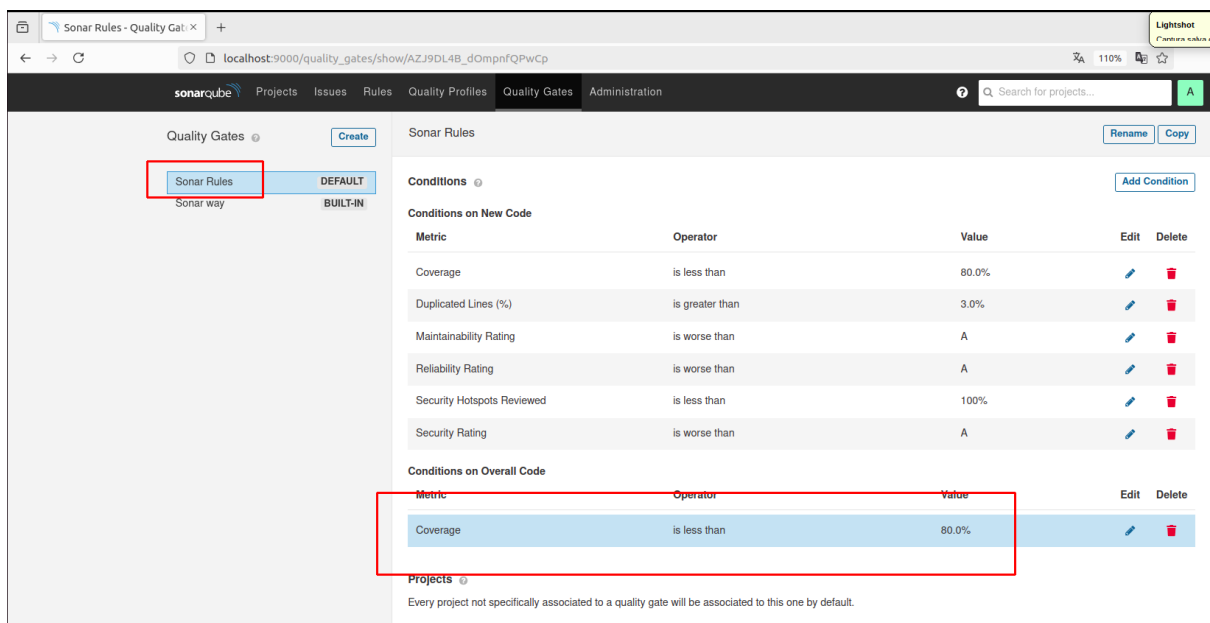
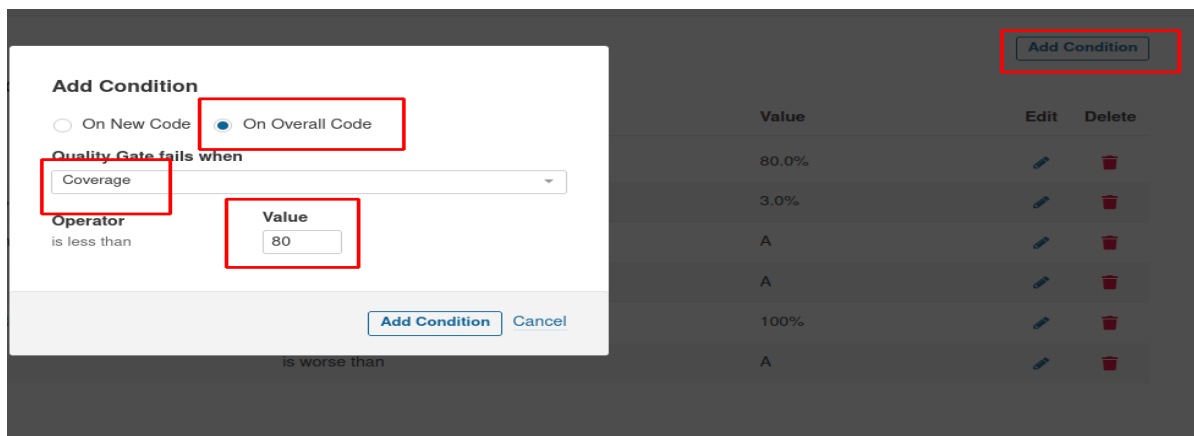
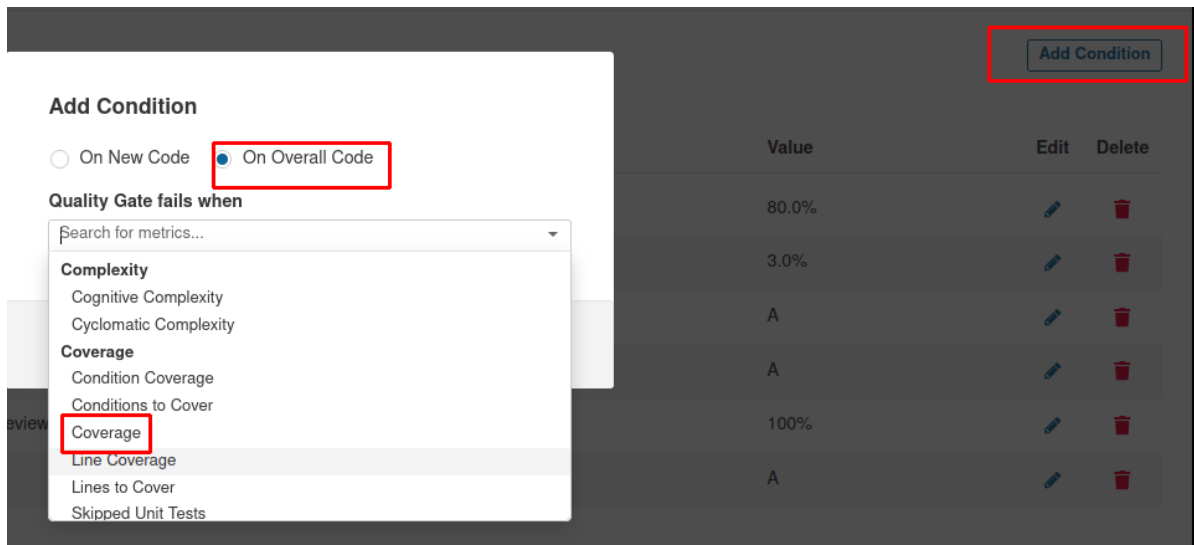
RenameCopySet as DefaultDelete

Conditions

Add Condition

Conditions on New Code

Metric	Operator	Value	Edit	Delete
--------	----------	-------	------	--------



← → ↻ localhost:8080/job/projeto_primeira_pipeline/

Jenkins

Painel de controle > projeto_primeira_pipeline >

Status **projeto_primeira_pipeline**

<> Changes

▶ Construir agora

⚙ Configurar

🗑 Excluir Pipeline

🔍 Full Stage View

🔗 SonarQube

📁 Stages

✎ Renomear

🔍 Pipeline Syntax

Stage View

	Declarative: Checkout SCM	Build	SonarQube Analysis	Quality Gate	Deploy
Average stage times: (Average full run time: ~1min 7s)	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

BackEnd Projects +

← → ↻ localhost:9000/dashboard?id=BackEnd

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects...

Last analysis had 1 warning October 11, 2024 at 3:31 PM Version not provided

Project Settings Project Information

BackEnd master

Overview Issues Security Hotspots Measures Code Activity

QUALITY GATE STATUS

Failed
1 conditions failed

On Overall Code

0.0% Coverage is less than 80.0%

MEASURES

New Code
Since October 11, 2024
Started 35 minutes ago

Overall Code

0 Bugs Reliability A

0 Vulnerabilities Security A

0 Security Hotspots Reviewed Security Review A

5min Debt 1 Code Smells Maintainability A

0.0% Coverage on 4 Lines to cover Unit Tests

0.0% Duplications on 118 Lines Duplicated Blocks 0

FIM