

Criar Job Maven e Docker build no Jenkins

Crie um repositório no [GitHub](#):

Create a new repository



A repository contains all project files, including the [Import a repository](#).

Owner * Repository name *

 nidioldolfini / docker-spring-jenkins

Great repository names are short and memorable. ↗

Description (optional)

- ☒  **Public**
Anyone on the internet can see this repository. Y
- ☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

- ☐ **Add a README file**
This is where you can write a long description for your project.
- ☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more](#)
- ☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more](#)

Create repository

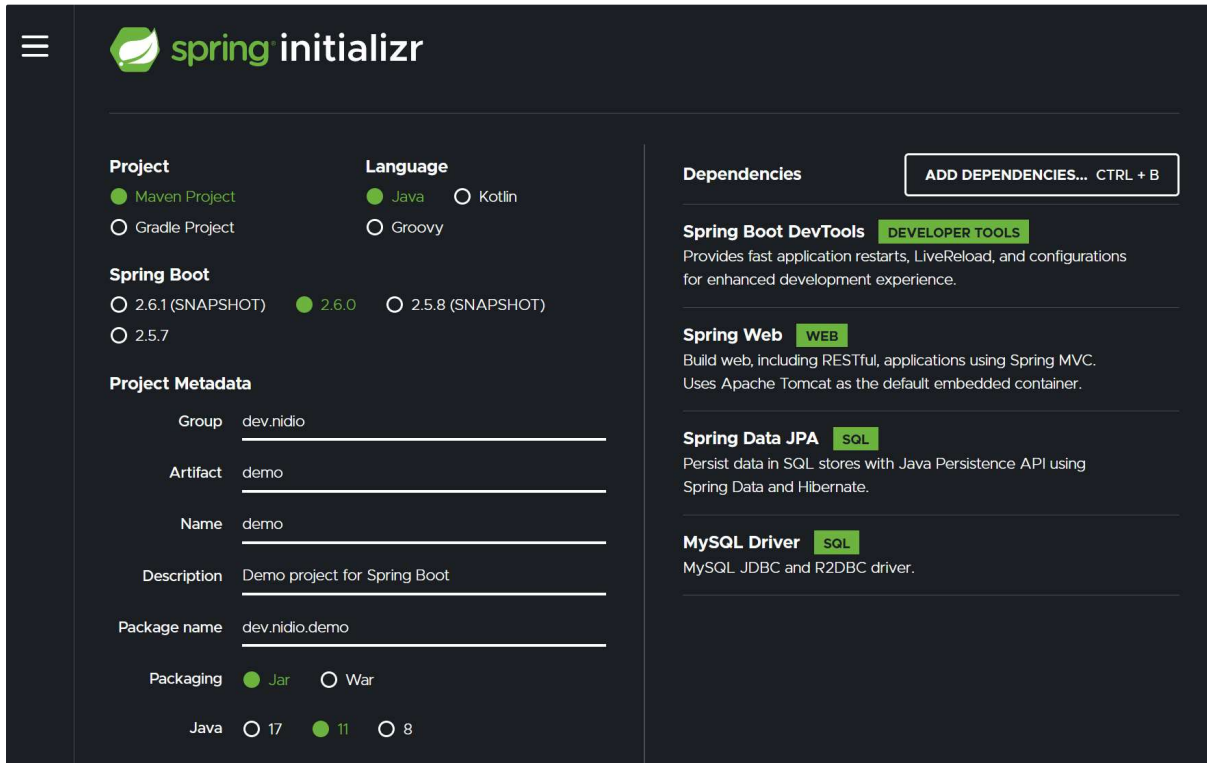
Crie um projeto no [start.spring.io](#) com dependências:

Spring Boot DevTools

Spring Web

Spring Data JPA

MySQL Driver



The image shows the Spring Initializr web interface. It has a dark theme with a sidebar on the left containing a hamburger menu icon and the 'spring initializr' logo. The main content area is divided into several sections: 'Project' with radio buttons for 'Maven Project' (selected) and 'Gradle Project'; 'Language' with radio buttons for 'Java' (selected), 'Kotlin', and 'Groovy'; 'Spring Boot' with radio buttons for '2.6.1 (SNAPSHOT)', '2.6.0' (selected), '2.5.8 (SNAPSHOT)', and '2.5.7'; 'Project Metadata' with input fields for 'Group' (dev.nidio), 'Artifact' (demo), 'Name' (demo), 'Description' (Demo project for Spring Boot), and 'Package name' (dev.nidio.demo); 'Packaging' with radio buttons for 'Jar' (selected) and 'War'; and 'Java' with radio buttons for '17', '11' (selected), and '8'. On the right side, there is a 'Dependencies' section with a button 'ADD DEPENDENCIES... CTRL + B'. Below this, there are four dependency cards: 'Spring Boot DevTools' (DEVELOPER TOOLS) with a description, 'Spring Web' (WEB) with a description, 'Spring Data JPA' (SQL) with a description, and 'MySQL Driver' (SQL) with a description.

Abra o projeto no [Intellij](#)

Inicialize um repositório vazio na pasta do nosso projeto com o comando: `git init`

```
PS C:\Users\nidio\Documents\GitHub\docker-spring-jenkins\docker-spring-jenkins> git init
Initialized empty Git repository in C:/Users/nidio/Documents/GitHub/docker-spring-jenkins/docker-spring-jenkins/.git/
PS C:\Users\nidio\Documents\GitHub\docker-spring-jenkins\docker-spring-jenkins>
```

Crie um Dockerfile

```
FROM openjdk:17-jdk-alpine3.14 WORKDIR /diretorioprincipal EXPOSE 8080 COPY target/mysqlspringdocker-0.0.1-SNAPSHOT.jar /diretorioprincipal/app.jar ENTRYPOINT ["java", "-jar", "app.jar"]
```

Abra o **application.properties** e altere colocando informações do seu banco de dados:

```
spring.datasource.username=root spring.datasource.password=senhadobanco spring.datasource.url=jdbc:mysql://localhost:3306/alterar?useTimezone=true &serverTimezone=America/Sao_Paulo
```

execute o comando **git add .** para adicionar os arquivos ao stage

execute o comando **git commit -m "criado projeto"** para commitar os arquivos do stage.

git branch -M main para trocar a **branch** de **master** para **main**

execute o comando **git remote add origin https://github.com/nidiodolfini/docker-spring-jenkins.git** para adicionar o repositório remoto

`git push -u origin main` com este comando é feito push dos arquivos para o GitHub

Clique em [Novo job:](#)



Pode ser digitado qualquer nome neste campo, desde que seja único no seu servidor Jenkins (recomendo a utilizar o mesmo nome do repositório no GitHub), depois selecione *Construir um projeto de software free-style*

Dashboard

Tudo

Enter an item name

» Required field



Construir um projeto de software free-style

Esta é a central de funcionalidades do Jenkins. Ele constrói qualquer SCM com qualquer sistema de builds, e ele até lida com diferentes tipos de builds de software.



Pipeline

Orchestrates long-running activities that can span multiple pipelines (formerly known as workflows) and/or organizing free-style job type.



Construir projeto de múltiplas configurações

Apropriado para projetos que necessitam de grande número de testes em múltiplos ambientes, builds para plataformas e...



Folder

Creates a container that stores nested items in it. Useful for organizing jobs which is just a filter, a folder creates a separate namespace with the same name as long as they are in different folders.



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches.

ation Folder



Selecione *GitHub Project* e cole a [url](#) do repositório no GitHub

A screenshot of the Jenkins 'General' tab for a new job configuration. The tab is selected, and the 'Ações de pós-build' section is visible. The 'Descrição' field is empty. Below it, there is a list of checkboxes for post-build actions: 'Commit agent's Docker container', 'Define a Docker template', 'Descartar builds antigos', 'Este build é parametrizado', and 'GitHub project'. The 'GitHub project' checkbox is checked. Below this, there is a 'Project url' field with the value 'https://github.com/nidiodolfini/docker-spring-jenkins'. A 'Visualizar' link is visible next to the '[HTML escapado]' text. At the bottom right, there is an 'Avançado...' button.

General Gerenciamento de código fonte Trigger de builds Ambiente de build Build

Ações de pós-build

Descrição

[HTML escapado] [Visualizar](#)

- ☐ Commit agent's Docker container
- ☐ Define a Docker template
- ☐ Descartar builds antigos
- ☐ Este build é parametrizado
- ☒ GitHub project

Project url

<https://github.com/nidiodolfini/docker-spring-jenkins>

[Avançado...](#)

☐ This build requires lockable resources

Cole a [url](#) do repositório junto com o *.git*

Gerenciamento de código fonte

☐ Nenhum

☒ Git

Repositories

Repository URL

Credentials

altere a **branch** para **main**

Branches to build

Branch Specifier (blank for 'any')

Navegar no repositório

Selecione consultar periodicamente o SCM

digite: * * * * * , para verificar o repositório a todo minuto

Trigger de builds

- ☐ Dispare builds remotamente (exemplo, a partir dos scripts) ?
- ☐ Construir após a construção de outros projetos ?
- ☐ Construir periodicamente ?
- ☒ Consultar periodicamente o SCM ?

Agenda ?

⚠️ Você realmente quis dizer "a todo minuto" quando diz "***"? Talvez você quisesse dizer "H ***" para verificar uma vez por hora**

Deveria ter executado em quarta-feira, 17 de novembro de 2021 19:32:23 Horário Padrão de Brasília; deverá executar novamente em quarta-feira, 17 de novembro de 2021 19:32:23 Horário Padrão de Brasília.

- ☐ Ignorar ações de pós-commit ?
- ☐ GitHub hook trigger for GITScm polling ?

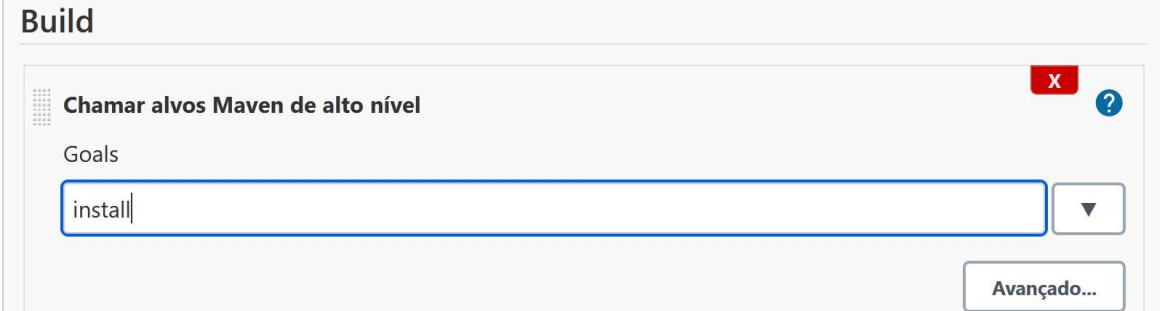
Selecione **chamar alvos Maven de alto nível**

Build

Adicionar passo no build ▲

- Add a new template to all docker clouds
- Build / Publish Docker Image
- Chamar alvos Maven de alto nível**
- Docker Build and Publish

Digite **install** para chamar o comando mvn install dentro do Jenkins, para gerar nosso pacote .jar



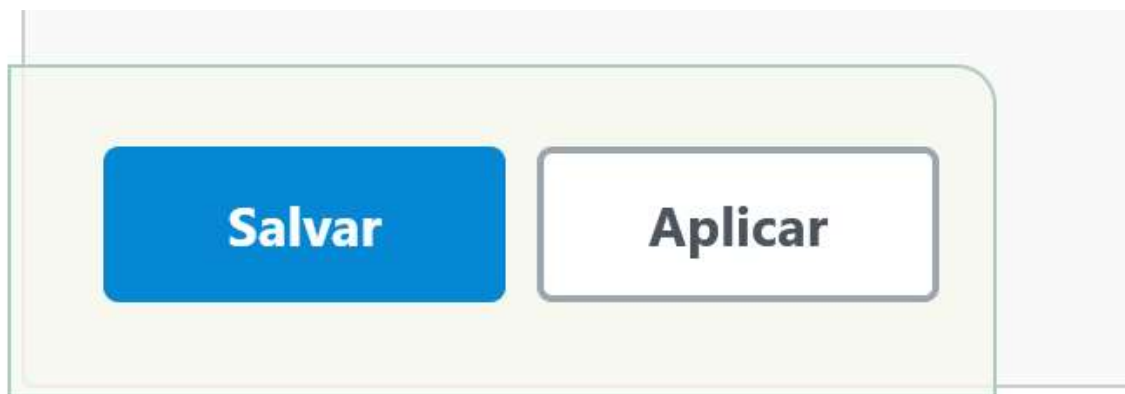
The image shows the 'Build' configuration page in Jenkins. At the top, there's a section titled 'Chamar alvos Maven de alto nível' with a red 'X' icon and a help icon. Below this, there's a 'Goals' field with the text 'install' and a dropdown arrow. At the bottom right, there's a button labeled 'Avançado...'.

Digite o seu ID do Docker / e o nome que você quer dar na sua imagem em **Repository Name**.



The image shows the 'Docker Build and Publish' configuration page in Jenkins. It includes several fields: 'Repository Name' (filled with 'nidio/mysqspringdocker'), 'Tag', 'Docker Host URI', 'Server credentials' (set to '- none -' with an 'Add' button), 'Docker registry URL', and 'Registry credentials' (set to 'nidio/*' with an 'Add' button). At the bottom left, there's a button 'Adicionar passo no build' and at the bottom right, a button 'Avançado...'.

Clique em **Salvar**



Agora deve ter um projeto no Dashboard

Tudo	+				
S	W	Nome ↓	Último sucesso	Última falha	Última duração
		docker-spring-jenkins	N/D	N/D	N/D
Ícone:					
S M L		Legenda		Atom feed de tudo	Atom feed das falhas

Agora a cada commit o Jenkins vai gerar um build usando o Maven.

Clicando no nome do job e depois no ultimo histórico de build:



Voltar para o Dashboard



Situação



Alterações



Workspace



Construir agora



Configurar



Excluir Projeto



Git Log de consulta periódica



GitHub



Rename



Histórico de builds



#2

17 de nov de 2021 19:40



17 de nov de 2021 19:37

Depois em Saída do console.



Saída do console

Podemos ver todo o log do console.

```
#9 exporting to image
#9 sha256:e8c613e07b0b7ff33893b694f7759a10d42e180f2b4dc349fb57dc6b71dcab00
#9 exporting layers done
#9 writing image sha256:f66311a950a0c6bd0cfff58eac1c15b2306a0dc7f16f679f5f5ad7efae038f95 done
#9 naming to docker.io/nidio/mysqlspringdocker done
#9 DONE 0.0s
WARNING: Support for the legacy ~/.dockercfg configuration file and file-format is deprecated and will be removed in an upcoming release

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[mysqspringdocker] $ docker push nidio/mysqlspringdocker
WARNING: Support for the legacy ~/.dockercfg configuration file and file-format is deprecated and will be removed in an upcoming release
WARNING: Support for the legacy ~/.dockercfg configuration file and file-format is deprecated and will be removed in an upcoming release
Using default tag: latest
The push refers to repository [docker.io/nidio/mysqlspringdocker]
dccc82c59b30: Preparing
7dcc5337c6e1: Preparing
34f7184834b2: Preparing
5836ece05bfd: Preparing
72e830a4dff5: Preparing
34f7184834b2: Layer already exists
7dcc5337c6e1: Layer already exists
dccc82c59b30: Layer already exists
5836ece05bfd: Layer already exists
72e830a4dff5: Layer already exists
latest: digest: sha256:62df649a67257092a0ba32aed09241a23bef3f317662a556979ba6a725f260a8 size: 1370
```

Crie um **docker-compose.yml** na raiz do projeto

```
version: "3.7" services: springweb: image: nidio/mysqlspringdocker container_name: springserver ports: - "8081:8080" networks: - servers depends_on: - mysql_db environment: - SPRING_DATASOURCE_URL=jdbc:mysql://mysql_db:3306/alterar?allowPublicKeyRetrieval=true&useUnicode=true&characterEncoding=utf8&useSSL=false mysql_db: image: "mysql:8.0" container_name: mysqlserver ports: - "3307:3306" environment: MYSQL_DATABASE: alterar MYSQL_USER: root MYSQL_ROOT_PASSWORD: senhadobanco volumes: - db_mysql:/var/lib/mysql networks: - servers volumes: db_mysql: networks: servers: driver: bridge
```