

# LAB AUTOMAÇÃO DE BUILD E DEPLOY COM JENKINS E GITLAB

PROF MESTRE DANIEL LEMESZENSKI

# LAB DE CI E CD – GITLAB E JENKINS

## SUMÁRIO

LAB DE CI E CD – GITLAB E JENKINS	1
1. Introdução	2
2. Docker Compose	2
3. Subindo o ambiente	2
4. Iniciando os serviços	3
useradd jenkins	3
passwd jenkins	3
ssh-keygen -t rsa	3
ssh-copy-id jenkins@ip-externo	3
5. Configurando Jenkins	3
6. Configurando GitLab	6
7. Criação do repositório no gitlab	6
8. Criando access token no gitlab	8
9. JENKINS – INSTALANDO GITLAB PLUGIN	9
10. JENKINS - Configurando a conexão COM Gitlab	10
11. Configurar Maven em Jenkins>Global tool Configuration	11
12. Criação do Job no Jenkins	12
13. Criando Webhook NO gitlab	16
14. Testando o pipeline (selecione test e push)	17

## 1. Introdução

Nesse lab vamos aprender a instalar e configurar o gitlab e o Jenkins. Além disso vamos criar um job de build no Jenkins disparado quando temos um evento em um repositório do gitlab.

O deploy será um pacote jar do spring boot e que será construído com maven e publicado com scp.

## 2. Docker Compose

Para isso, usar o docker-compose.yml existente no projeto do git abaixo:

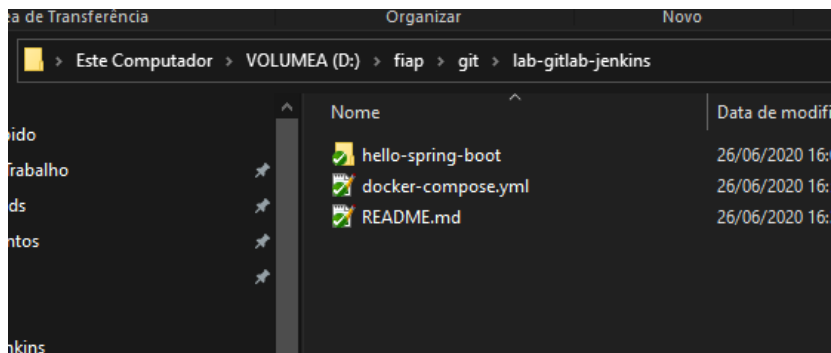
<https://github.com/daniboy82/lab-gitlab-jenkins.git>

```
version: '2'
services:
  jenkins:
    image: jenkins/jenkins
    container_name: jenkins
    hostname: jenkins
    network_mode: bridge
    ports:
      - "85:8080"
    #volumes:
    # - ~/jenkins_home:/var/jenkins_home
  gitlab:
    image: gitlab/gitlab-ce
    container_name: gitlab
    hostname: gitlab
    restart: always
    network_mode: bridge
    ports:
      - "80:80"
    # volumes:
    # - ~/gitlab/config:/etc/gitlab
    # - ~/gitlab/logs:/var/log/gitlab
    # - ~/gitlab/data:/var/opt/gitlab
```

## 3. Subindo o ambiente

Crie um fork do projeto:

<https://github.com/daniboy82/lab-gitlab-jenkins>



Clone o branch criado através do fork  
entre no diretório lab-gitlab-jenkins

#### 4. INICIANDO OS SERVIÇOS

No diretório em que você criou o arquivo docker-compose.yml, execute o comando:

```
docker-compose up -d
```

Criando usuário Jenkins na máquina EC2

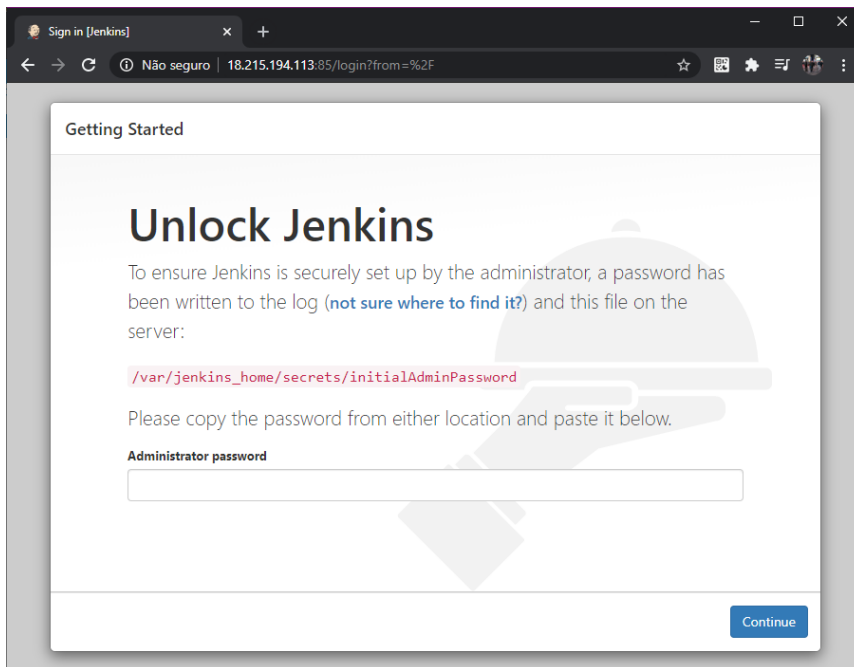
```
useradd jenkins  
passwd jenkins
```

Entrar do container Jenkins e enviar chave para máquina EC2, digitar a senha Jenkins:

```
docker exec -it Jenkins bash  
  
ssh-keygen -t rsa  
ssh-copy-id jenkins@ip-externo
```

#### 5. CONFIGURANDO JENKINS

Após a inicialização dos serviços com o docker-compose, vamos fazer a configuração inicial do Jenkins. Acesse a url <http://ip-externo:85/> e você será direcionado para a página inicial da ferramenta.



Para configurá-lo, vamos inserir a chave que ele gerou no momento da instalação. O jeito mais simples é digitar no terminal:

*docker logs -f Jenkins*

Ele vai exibir assim:

```
ubuntu@ip-172-31-76-219: ~/lab-gitlab-jenkins
]}; root of context hierarchy
2020-06-26 19:52:42.540+0000 [id=25] INFO o.s.c.s.AbstractApplicationContext#obtainFreshBeanFactory: Bean factory for application
context [org.springframework.web.context.support.StaticWebApplicationContext@7bf339d]: org.springframework.beans.factory.support.Default
ListableBeanFactory@59e418b2
2020-06-26 19:52:42.549+0000 [id=25] INFO o.s.b.f.s.DefaultListableBeanFactory#preInstantiateSingletons: Pre-instantiating singlet
ons in org.springframework.beans.factory.support.DefaultListableBeanFactory@59e418b2: defining beans [filter,legacy]; root of factory hi
erarchy
2020-06-26 19:52:43.216+0000 [id=25] INFO jenkins.install.SetupWizard#init:

*****
*****
*****

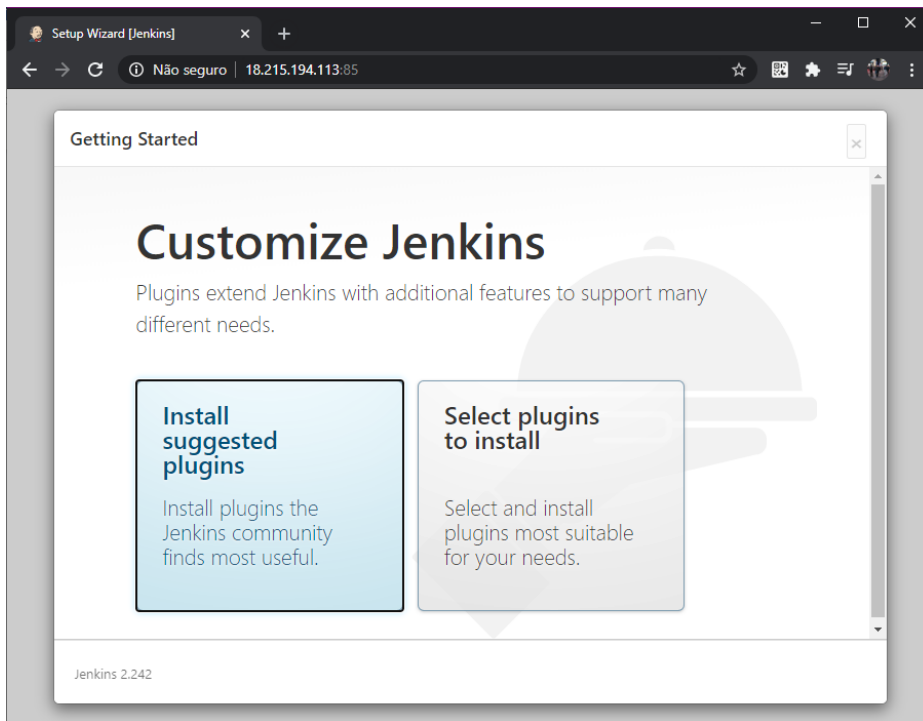
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

25de6841138a4bd7b55841e236517609

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
*****

2020-06-26 19:52:48.255+0000 [id=40] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.M
aven.MavenInstaller
2020-06-26 19:52:48.262+0000 [id=40] INFO hudson.util.Retrier#start: Performed the action check updates server successfully at the
attempt #1
2020-06-26 19:52:48.265+0000 [id=40] INFO hudson.model.AsyncPeriodicWork#lambda$doRun$0: Finished Download metadata. 8,938 ms
2020-06-26 19:52:48.683+0000 [id=26] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
2020-06-26 19:52:48.694+0000 [id=19] INFO hudson.WebAppMain$3#run: Jenkins is fully up and running
```

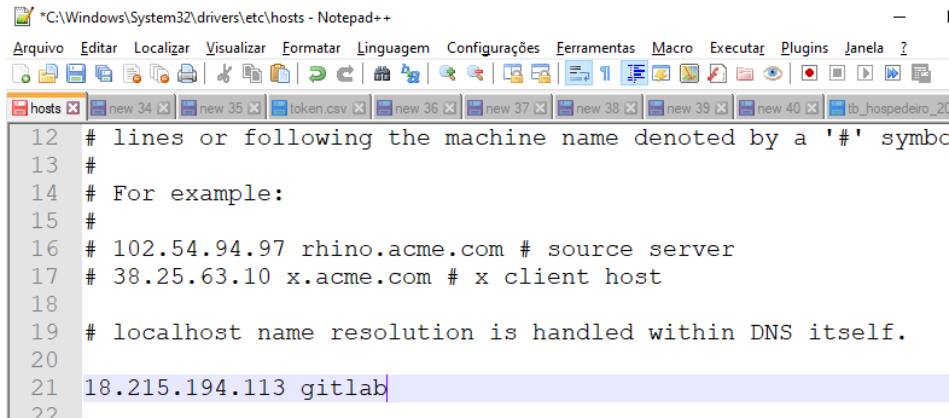


Pronto, o Jenkins já está pronto para ser utilizado!

The screenshot shows the 'Create First Admin User' screen. It contains five input fields: 'Nome de usuário:' with the value 'daniel', 'Senha:' with masked characters, 'Confirmar a senha:' with masked characters, 'Nome completo:' with the value 'daniel lemes', and 'Endereço de e-mail:' with the value 'daniel.lemes@gmail.com'. At the bottom, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'. The Jenkins version '2.242' is shown at the bottom left.The screenshot shows the 'Instance Configuration' screen. It features a 'Jenkins URL:' field with the value 'http://18.215.194.113:85/'. Below this, there is explanatory text about the Jenkins URL. At the bottom, there are two buttons: 'Not now' and 'Save and Finish'. The Jenkins version '2.242' is shown at the bottom left.

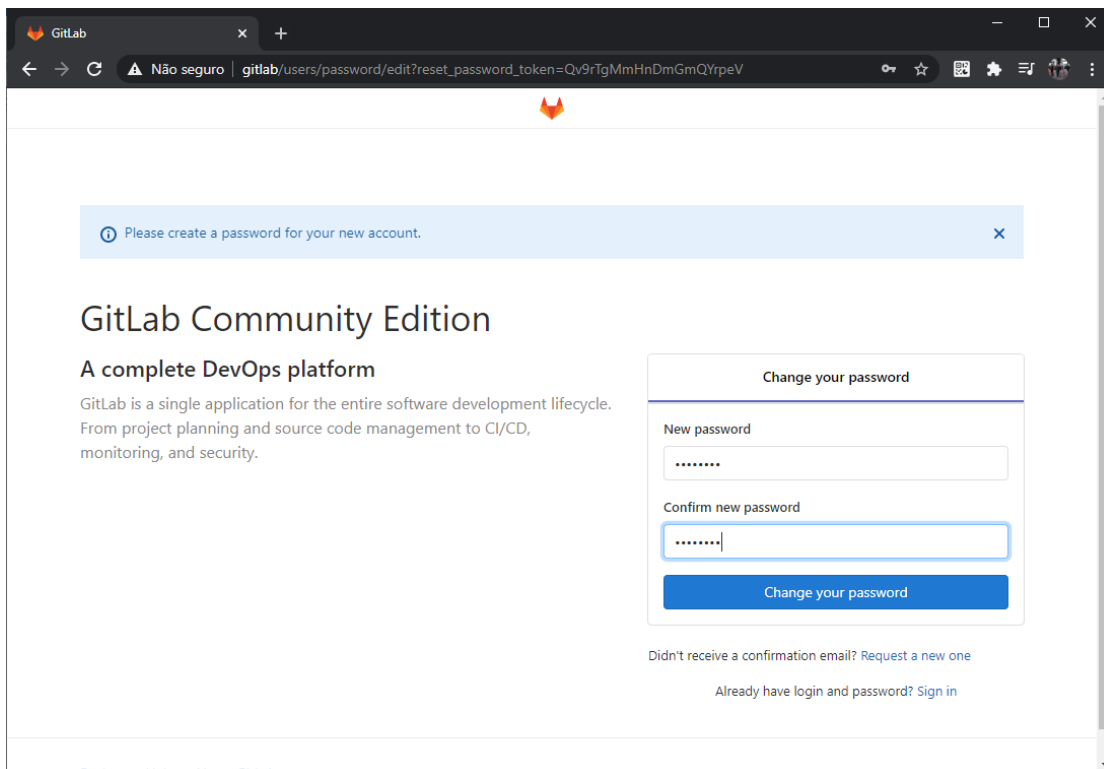
## 6. CONFIGURANDO GITLAB

Configurar o etc/hosts com o ip-externo da máquina ec2:



```
*C:\Windows\System32\drivers\etc\hosts - Notepad++
Arquivo Editar Localizar Visualizar Formatar Linguagem Configurações Ferramentas Macro Executar Plugins Janela ?
hosts new 34 new 35 token.csv new 36 new 37 new 38 new 39 new 40 tb_hospedeiro_2
12 # lines or following the machine name denoted by a '#' symbol
13 #
14 # For example:
15 #
16 # 102.54.94.97 rhino.acme.com # source server
17 # 38.25.63.10 x.acme.com # x client host
18
19 # localhost name resolution is handled within DNS itself.
20
21 18.215.194.113 gitlab
22
```

Com o GitLab, o processo é bem mais simples. É só acessar a página inicial dele <http://gitlab/> e colocar uma senha com no mínimo oito caracteres.



Agora estamos com o ambiente pronto e podemos começar a configuração da integração. Segura, que o filho é seu!

## 7. CRIAÇÃO DO REPOSITÓRIO NO GITLAB

Vamos criar um repositório no Gitlab para armazenar nosso projeto.

Crie o grupo devops:

**New group**

Groups allow you to manage and collaborate across multiple projects. Members of a group have access to all of its projects.

Groups can also be nested by creating subgroups.

Projects that belong to a group are prefixed with the group namespace. Existing projects may be moved into a group.

**Create group**

Group name:

Group URL:

Group description (optional):

Group avatar:  No file chosen  
The maximum file size allowed is 200KB.

Visibility level: Who will be able to see this group? [View the documentation](#)

☒ Private  
The group and its projects can only be viewed by members.

☐ Internal  
The group and any internal projects can be viewed by any logged in user.

- a - Crie o projeto spring-boot-hello no gitlab
- b - Copie os arquivos do github existentes em <https://github.com/daniboy82/lab-gitlab-jenkins/tree/master/hello-spring-boot> nesse projeto;
- c-realize o commit e push desses fontes:

**daniboy82 / lab-gitlab-jenkins**

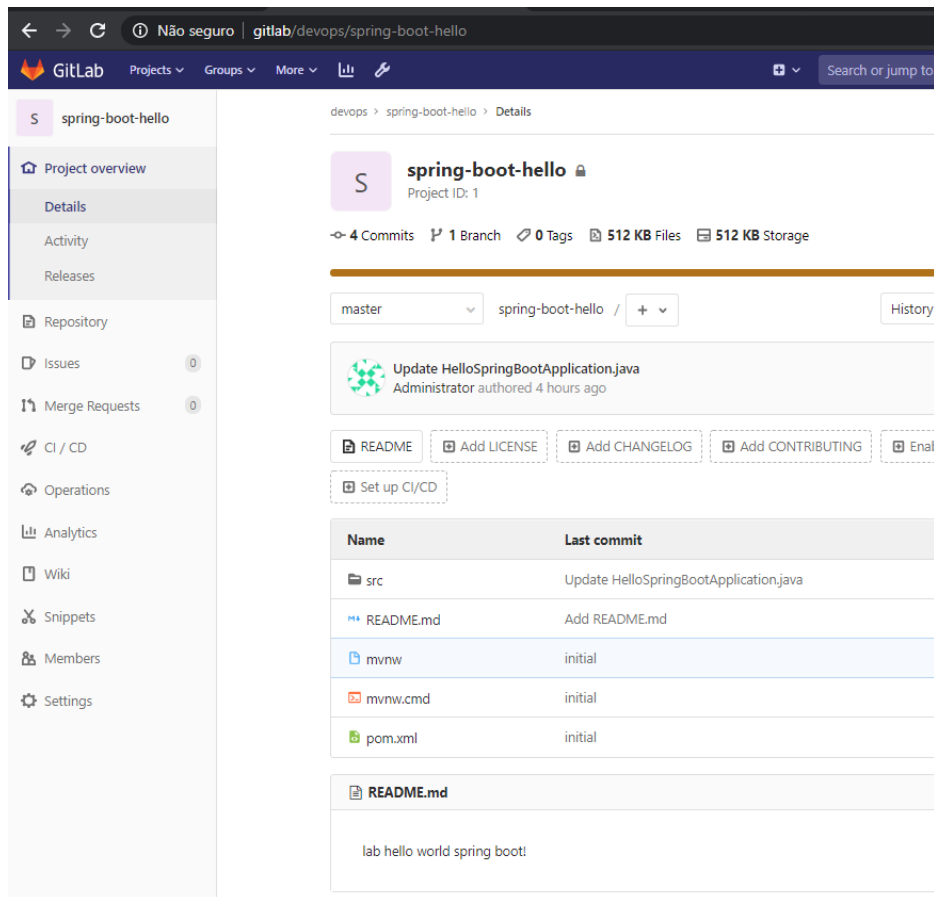
<> Code    ⓘ Issues    🔗 Pull requests    ▶ Actions    📁 Projects

Branch: master    **lab-gitlab-jenkins / hello-spring-boot /**

**daniboy82** committed addb5bf 8 hours ago

..	
..mvn/wrapper	first commit
src	first commit
.gitignore	first commit
mvnw	first commit
mvnw.cmd	first commit
pom.xml	first commit

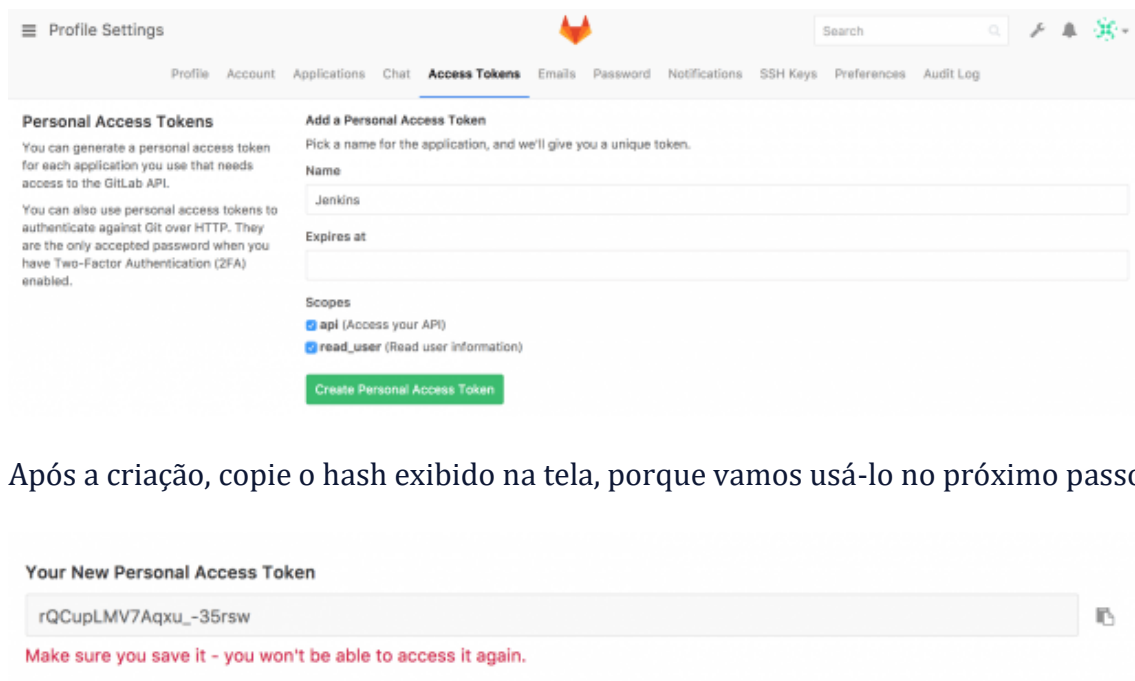




## 8. CRIANDO ACCESS TOKEN NO GITLAB

Clique no círculo com sua imagem de perfil, que fica no canto superior direito, e vá em *Profile Settings*.

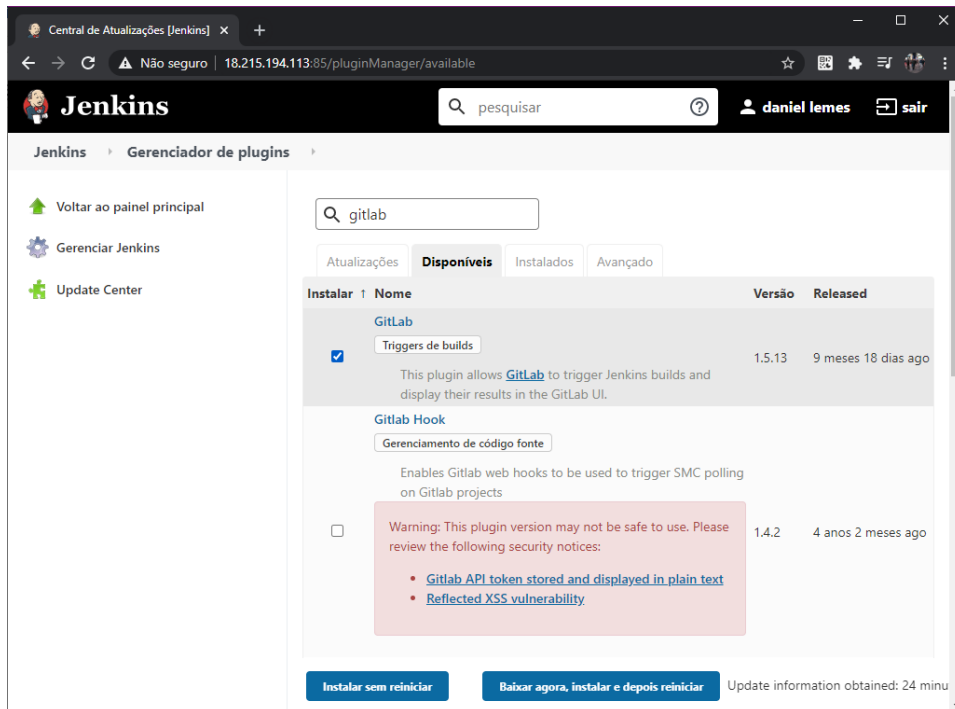
Clique na aba *Access Tokens*. Digite um nome fácil de identificar para seu token e clique em *Create Personal Access Token*.



O token desaparece após atualizar a página, então, guarde em um lugar seguro.

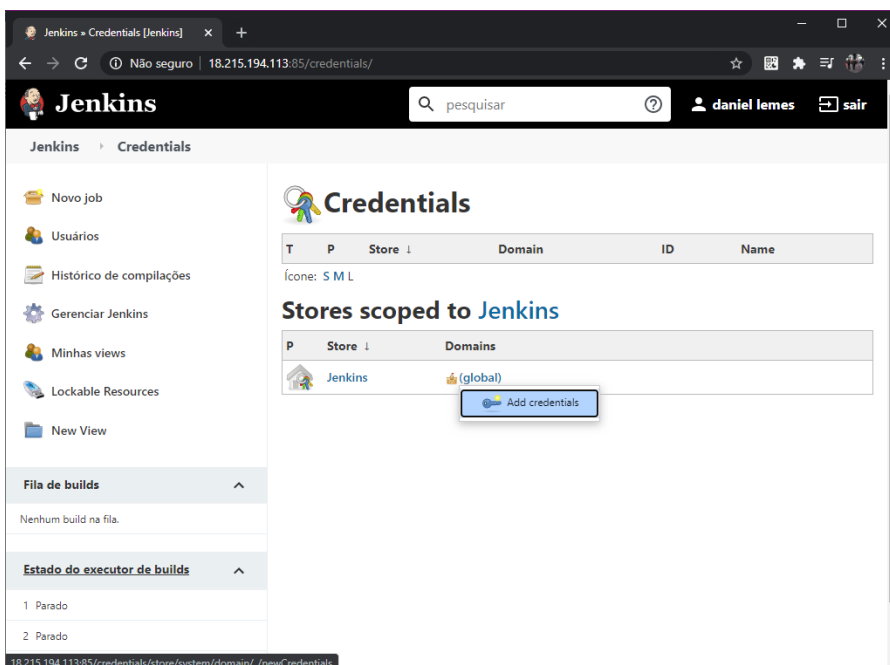
## 9. JENKINS – INSTALANDO GITLAB PLUGIN

Vamos precisar do [GitLab Plugin](#) para nos conectarmos ao repositório do projeto (lembre-se: são só dois arquivos, mas vamos manter a positividade). Para instalar, vá até [Gerenciar Jenkins]->[Gerenciar Plugins] e clique na aba *Disponíveis*.



Com o plugin instalado, vamos adicionar uma conexão com o GitLab, seguir alguns passos simples.

Na tela inicial do Jenkins, vá até Credentials. Clique na seta do lado de global e selecione *Add Credentials*.





Na janela a seguir, selecione *GitLab API Token* e coloque o hash gerado no GitLab no campo API token.

The screenshot shows the Jenkins 'New Credentials' page. The breadcrumb trail is 'Jenkins > Credentials > System > Global credentials (unrestricted)'. On the left, there are links for 'Back to credential domains' and 'Add Credentials'. The main form has the following fields: 'Kind' (set to 'GitLab API token'), 'Scope' (set to 'Global (Jenkins, nodes, items, all child items, etc)'), 'API token' (a masked field with dots), 'ID' (set to 'gitlab'), and 'Description' (set to 'gitlab'). An 'OK' button is at the bottom left. The footer shows 'Página gerada: 26/06/2020 20h23min30s UTC', 'REST API', and 'Jenkins 2.242'.

Após clicar em OK, você verá sua credencial criada.

The screenshot shows the 'Global credentials (unrestricted)' page. It includes a header with a house icon and the title 'Global credentials (unrestricted)'. Below the header, it says 'Credentials that should be available irrespective of domain specification to requirements matching.' There is a table with the following data:

Name	Kind	Description
 <a href="#">GitLab API token (Melhor Gitlab)</a>	GitLab API token	Melhor Gitlab 

Below the table, it says 'Ícone: S M L'.

## 10. JENKINS - CONFIGURANDO A CONEXÃO COM GITLAB

Com a nossa credencial criada, vamos em [Gerenciar Jenkins]->[Configurar o sistema]. Desça até a Gitlab e preencha as informações conforme a imagem abaixo. Ao terminar, clique em *Test Connection*. Se tudo estiver certo, clique em *Salvar*.

Desmarcar opção *enable authentication for '/project' end-point*

Configurar o sistema [Jenkins] x +

Não seguro | 18.215.194.113:85/configure

Jenkins > configuração

Adicionar ▾

List of default health metrics for Folders

**Gitlab**

Enable authentication for '/project' end-point ☐

GitLab connections

Connection name

Gitlab host URL

The complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com)

Credentials  Add ▾

API-Token for Gitlab access required

API Token for accessing Gitlab

API-Level

API Level for accessing Gitlab

Ignore SSL Certificate Errors ☐

Connection timeout (in seconds)

The time to wait for establishing the connection

Read timeout (in seconds)

Salvar Apply

Marcar ignore ssl certficate erros:

Configurar o sistema [Jenkins] x +

Não seguro | 18.215.194.113:85/configure

Jenkins > configuração

Adicionar ▾

Gitlab host URL

A name for the connection

The complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com)

Credentials  Add ▾

API-Token for Gitlab access required

API Token for accessing Gitlab

API-Level

API Level for accessing Gitlab

Ignore SSL Certificate Errors ☒

Connection timeout (in seconds)

The time to wait for establishing the connection

Read timeout (in seconds)

The time to wait while receiving the response

Test Connection

Excluir

Adicionar

Timestamp

Salvar Apply

## 11. CONFIGURAR MAVEN EM JENKINS>GLOBAL TOOL CONFIGURATION

Global Tool Configuration [Jenkins] x +

Não seguro | 18.215.194.113:85/configure/tools/

Jenkins > Global Tool Configuration

Add Git ▾

**Gradle**

Gradle instalações

Adicionar Gradle

Lista de Gradle instalações nesse sistema

**Ant**

Ant instalações

Adicionar Ant

Lista de Ant instalações nesse sistema

**Maven**

Maven instalações

Adicionar Maven

Maven

Nome

Obrigatório

Instalar automaticamente

Instalar a partir do Apache

Versão

Excluir instalador

Excluir Maven

Adicionar instalador ▾

Adicionar Maven

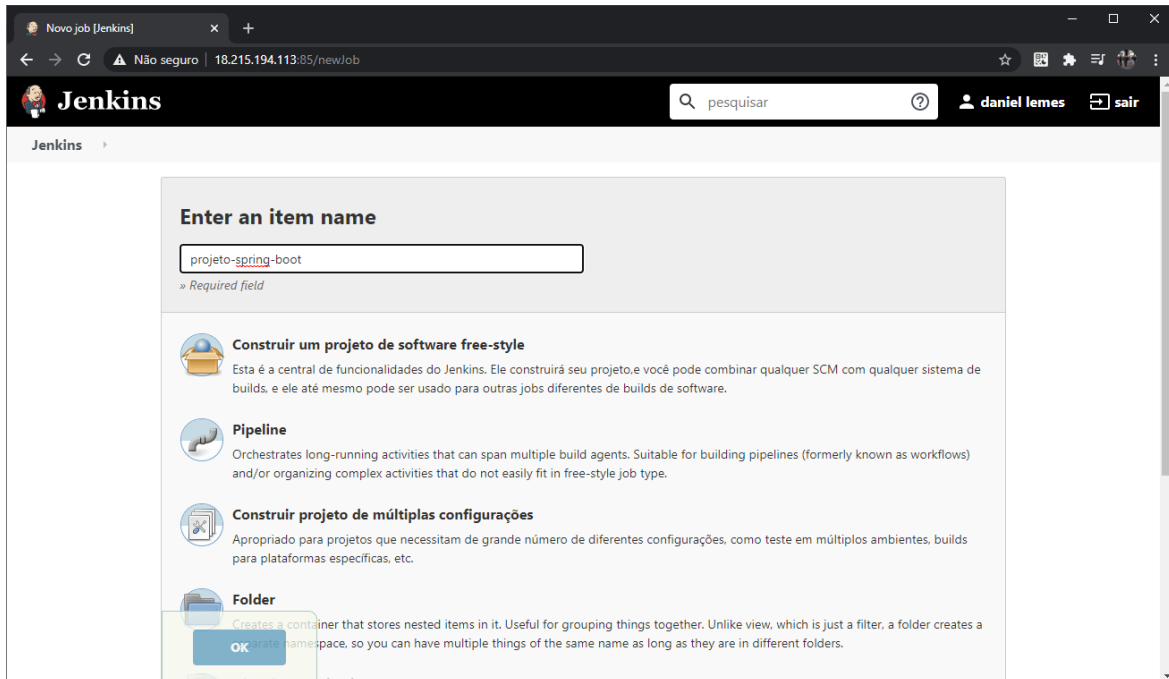
Lista de Maven instalações nesse sistema

Salvar Apply

Página gerada: 2019-03-20 10:18:00 UTC Jenkins 2.242

## 12.CRIAÇÃO DO JOB NO JENKINS

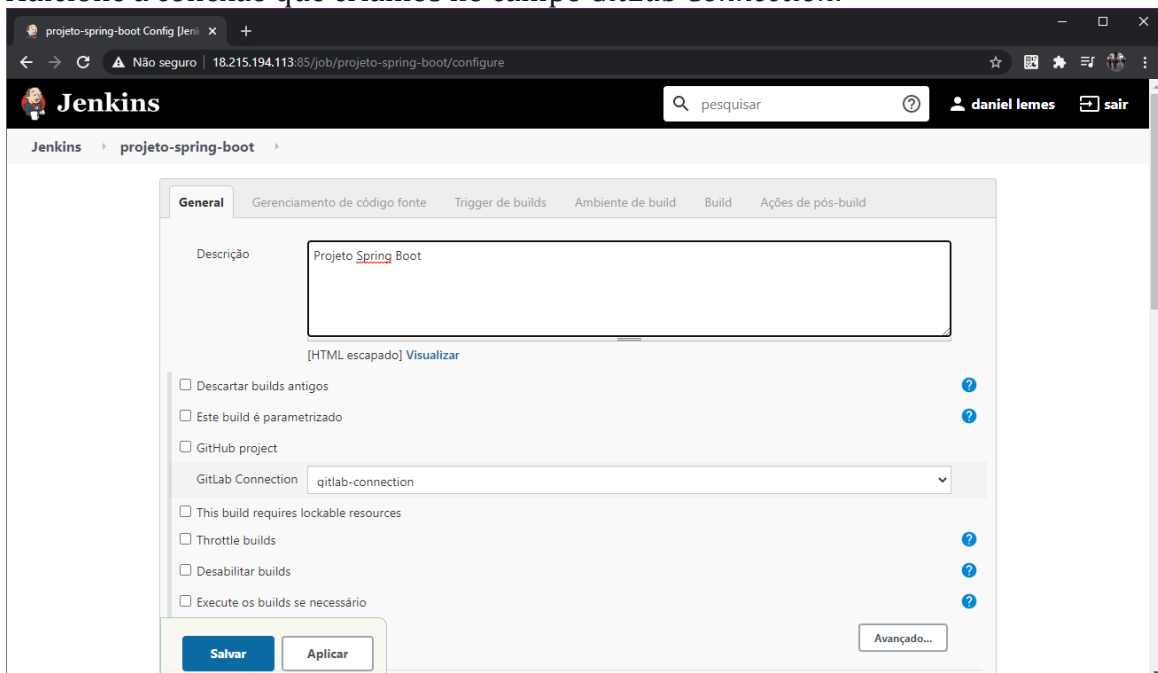
No canto esquerdo, clique em *Novo Job*, escolha *Projeto Freestyle*, digite um nome para o Job e clique em *OK*.



Com o Job criado, vamos às configurações.

### General

Adicione a conexão que criamos no campo *GitLab Connection*.



### Gerenciamento de código fonte

Selecione a opção Git e adicione as informações do repositório criado no GitLab.

The screenshot shows the Jenkins web interface. The browser address bar indicates the URL is `18.215.194.113:85/job/projeto-spring-boot/configure`. The page title is 'projeto-spring-boot Config [Jen]'. The main configuration area is titled 'Gerenciamento de código fonte' and has several tabs: 'General', 'Gerenciamento de código fonte' (selected), 'Trigger de builds', 'Ambiente de build', 'Build', and 'Ações de pós-build'. Under 'Gerenciamento de código fonte', there are three radio buttons: 'Nenhum', 'Git' (selected), and 'Subversion'. Below these, there are sections for 'Repositories' and 'Branches to build'. The 'Repositories' section has a 'Repository URL' field with the value `http://18.215.194.113:80/devops/spring-boot-hello.git` and a 'Credentials' dropdown menu showing 'root/\*\*\*\*\*' with an 'Add' button. There are also 'Avançado...' and 'Add Repository' buttons. The 'Branches to build' section has a 'Branch Specifier (blank for \'any\')' field with the value `*/master` and an 'Add Branch' button. At the bottom, there is a 'Navegar no repositório' dropdown menu set to '(Auto)' and an 'Adicionar' button. At the very bottom, there are 'Salvar' and 'Aplicar' buttons.

Para adicionar as credencias válidas para o repositório, clique em [Add]->[Jenkins] e adicione um usuário e senha válidos para conexão.

The screenshot shows the 'Add Credentials' dialog box in Jenkins. The dialog has a title bar 'Jenkins Credentials Provider: Jenkins'. Inside, there is a section 'Add Credentials' with several fields: 'Domain' (Global credentials (unrestricted)), 'Kind' (Username with password), 'Scope' (Global (Jenkins, nodes, items, all child items, etc)), 'Username' (root), 'Password' (masked with dots), 'ID' (empty), and 'Description' (empty). At the bottom, there are 'Add' and 'Cancel' buttons. The dialog is overlaid on the same Jenkins configuration page as the previous screenshot.

## Trigger de Builds

Esse é o primeiro passo para nossa integração entre os dois serviços. Estamos quase lá!

Marque a opção *Build when a change is pushed to GitLab* e selecione quais ações executadas no GitLab vão acionar o Job no Jenkins. Copie a URL logo após a frase *GitLab CI Service URL*, pois vamos usá-la mais pra frente.

The screenshot shows the 'Trigger de builds' configuration page in Jenkins. It has a title 'Trigger de builds' and a list of triggers. The first trigger, 'Build when a change is pushed to GitLab', is selected with a blue checkmark. Below it, there's a section 'Enabled GitLab triggers' with several options: 'Push Events' (checked), 'Opened Merge Request Events' (checked), 'Accepted Merge Request Events' (unchecked), 'Closed Merge Request Events' (unchecked), 'Rebuild open Merge Requests' (set to 'Never'), 'Approved Merge Requests (EE-only)' (checked), 'Comments' (checked), and 'Comment (regex) for triggering a build' (set to 'Jenkins please retry a build'). There's also a 'GitHub hook trigger for GITScm polling' option which is unchecked. At the bottom, there are 'Salvar' and 'Aplicar' buttons.

Marque delete workspace before build starts:

## Ambiente de Build

The screenshot shows the 'Ambiente de build' configuration page in Jenkins. It has a title 'Ambiente de build' and a list of options. The first option, 'Delete workspace before build starts', is selected with a blue checkmark. Below it, there's a section 'Ambiente de build' with several options: 'Use secret text(s) or file(s)' (unchecked), 'Abort the build if it's stuck' (unchecked), 'Add timestamps to the Console Output' (unchecked), 'Inspect build log for published Gradle build scans' (unchecked), and 'With Ant' (unchecked). At the bottom, there's an 'Avançado...' button.

## Configurar build:

The screenshot shows the 'Build' configuration page in Jenkins. It has a title 'Build' and a list of options. The first option, 'Add timestamps to the Console Output', is unchecked. Below it, there's a section 'Build' with several options: 'Inspect build log for published Gradle build scans' (unchecked), 'With Ant' (unchecked), and 'Chamar alvos Maven de alto nível' (checked). The 'Chamar alvos Maven de alto nível' section has a 'Versão do Maven' dropdown set to 'mvn-3' and a 'Goals' text box set to 'package'. At the bottom, there's an 'Avançado...' button.

Executar shell

Comando

scp ./target/hello-spring-boot-0.0.1-SNAPSHOT.jar jenkins@18.215.194.113:/tmp

Veja a lista de variáveis de ambiente disponíveis

Avançado...

Executar shell

Comando

ssh jenkins@18.215.194.113 <<'ENDSSH'  
echo "java -jar /tmp/hello-spring-boot-0.0.1-SNAPSHOT.jar" | at now + 1 min  
ENDSSH

Veja a lista de variáveis de ambiente disponíveis

Avançado...

### Ações de pós-build

Selecione a opção *Publish build status to GitLab commit*. Agora o Jenkins colocará o feedback de cada build nos commits/merges que acionaram o Job.

Ações de pós-build

Publish build status to GitLab commit (GitLab 8.1+ required)

Avançado...

Adicionar ação de pós-build ▼

Clique em Salvar e o Job está finalizado.

Depois de salvar o job, entrar novamente nas configurações do job, em trigger de builds e clicar em avançado, generate token (salve a hash gerada no notepad ou área de transferência) e clique em salvar.



→ ↻ ⓘ Não seguro | 52.70.157.53:8080/job/job-gitlab/configure

Jenkins » job-gitlab »

General Gerenciamento de código fonte **Trigger de builds** Ambiente de build Build Ações de pós-build

Set build description to build cause (eg. Merge request or Git Push ) ☒

Build on successful pipeline events ☐

Pending build name for pipeline  ?

Cancel pending merge request builds on update ☐

Allowed branches

- ☒ Allow all branches to trigger this job ?
- ☐ Filter branches by name ?
- ☐ Filter branches by regex ?
- ☐ Filter merge request by label

Secret token  ?

☐ GitHub hook trigger for GITScm polling ?

☐ Consultar periodicamente o SCM ?

**Ambiente de build**

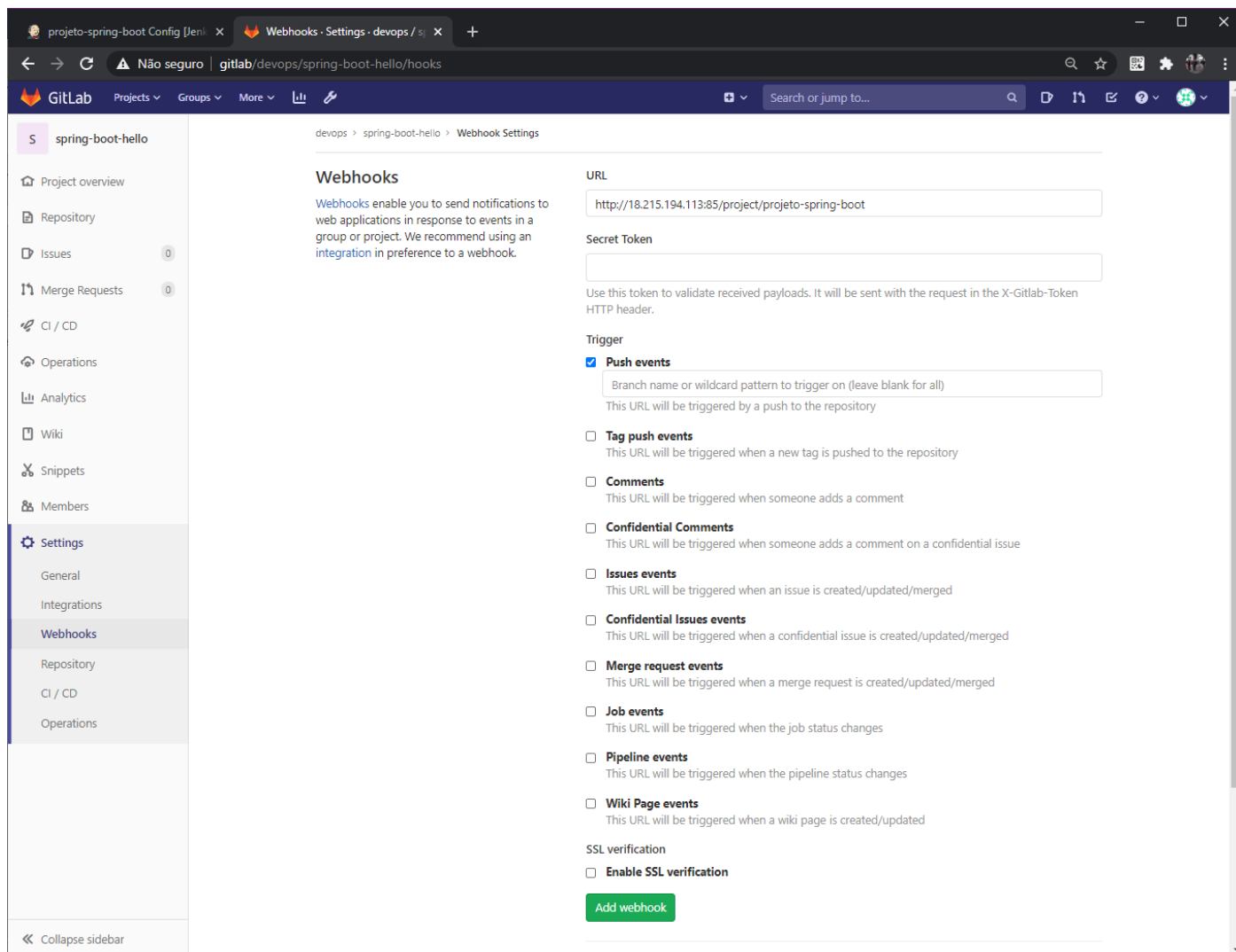
☐ Delete workspace before build starts

☐ Use secret text(s) or file(s) ?

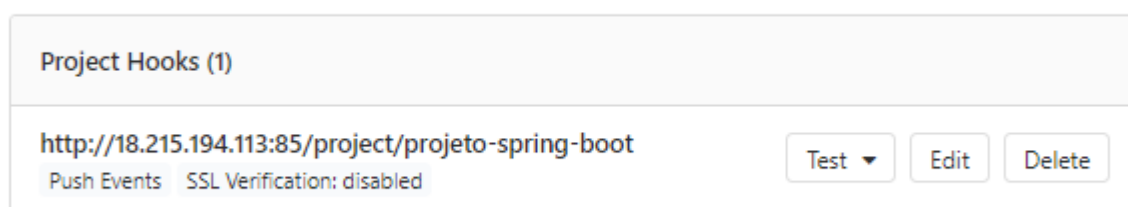
## 13. CRIANDO WEBHOOK NO GITLAB

Como último passo, vamos adicionar o Webhook em nosso repositório no GitLab. Ele vai fornecer as informações para o Jenkins quando houver alguma alteração ou ação no repositório no qual ele foi configurado.

Na página inicial do repositório, clique na engrenagem no canto superior direito e selecione *Webhooks*.

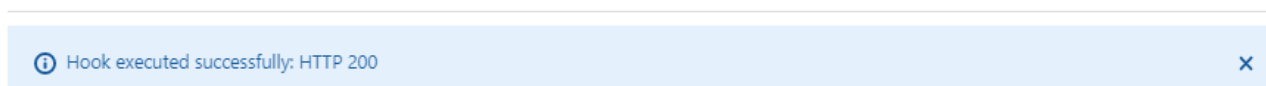


## 14.TESTANDO O PIPELINE (SELECIONE TEST E PUSH)



Deve aparecer:

devops > spring-boot-hello > Webhook Settings



Após tantos passos e configurações, vamos ver nosso projeto rodando. Abra um merge request ou faça um commit/push para o repositório criado do gitlab e aguarde o Job ser iniciado. Ao finalizar, ele vai adicionar ao GitLab o feedback do build realizado com as alterações que foram feitas no código. Com isso seu projeto ganha muito mais agilidade e confiabilidade, com testes e feedbacks rápidos e automatizados.

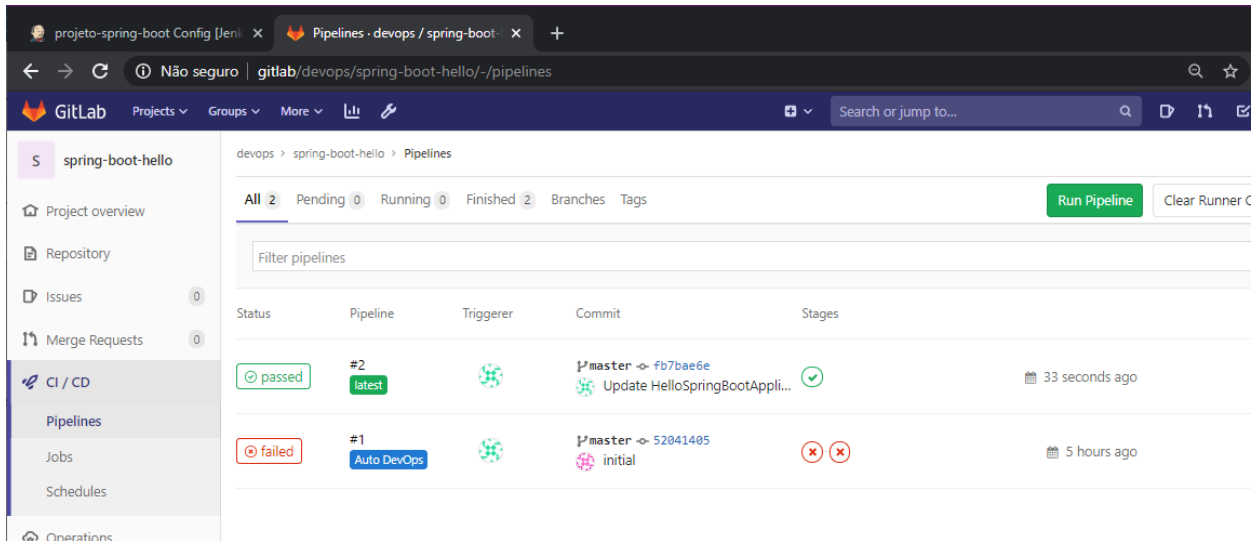
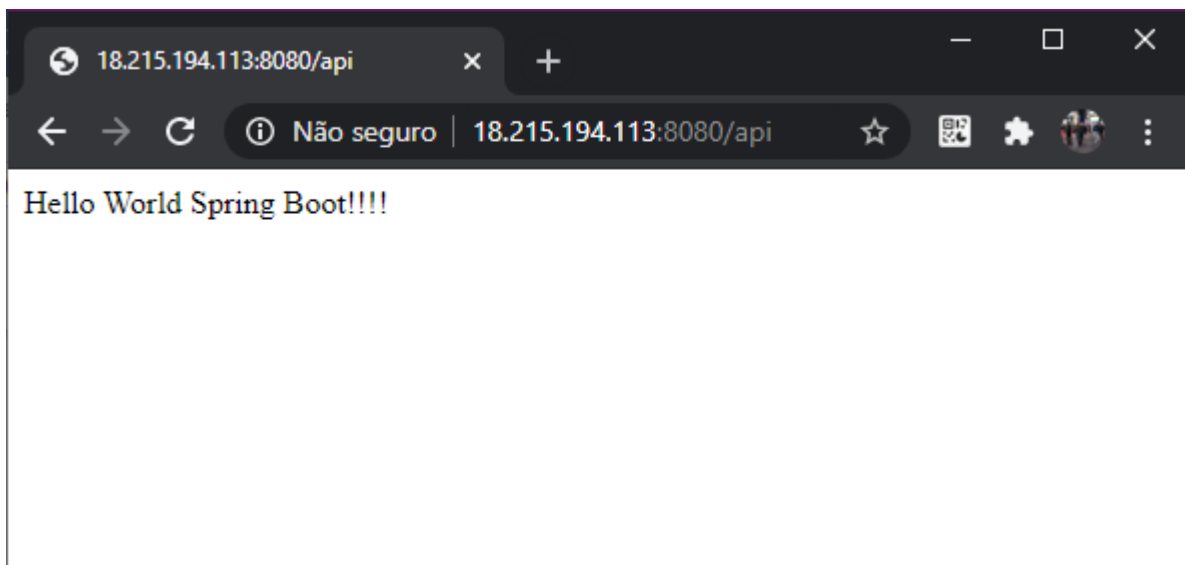


Figura 1 menu pipeline do repositório

Valide que o deploy foi realizado com sucesso acessando <http://ip-ec2:8080>



Referências:

<https://imasters.com.br/devsecops/gitlab-jenkins-uma-integracao-poderosa>