

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
5. criando chave rsa entre jenkins e host	3
6. Configurando Jenkins	4
7. Configurando GitLab	6
8. Criação do repositório no gitlab	7
9. Criando access token no gitlab	8
10. JENKINS – INSTALANDO GITLAB PLUGIN	9
11. JENKINS - Configurando a conexão COM Gitlab	11
12. Configurar Maven em Jenkins>Global tool Configuration	11
13. Criação do Job no Jenkins	12
14. Criando Webhook NO gitlab	16
15. 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:

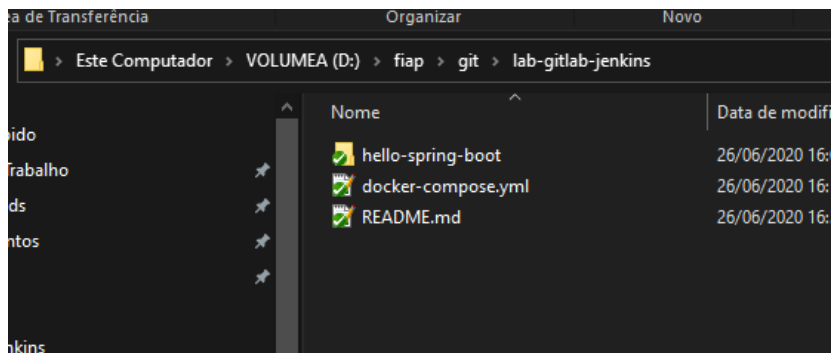
<https://github.com/daniboy82/lab-gitlab-jenkins.git>

```
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
```

5. CRIANDO CHAVE RSA ENTRE JENKINS E HOST

A - Entrar no container Jenkins e criar chave rsa:

```
docker exec -it jenkins bash
```

(criar chave sem pass frase, enter, enter, enter...)

```
ssh-keygen -t rsa
```

B - Criando usuário Jenkins na máquina EC2

```
useradd -m jenkins
```

```
passwd jenkins
```

```
ssh-keygen -t rsa
```

(enter, enter, enter....)

```
sudo docker cp jenkins:/var/jenkins_home/.ssh/id_rsa.pub /home/jenkins/
```

```
cat /home/jenkins/id_rsa.pub >> /home/jenkins/.ssh/authorized_keys
```

```
chmod 644 /home/jenkins/.ssh/authorized_keys
```

C- testar chave

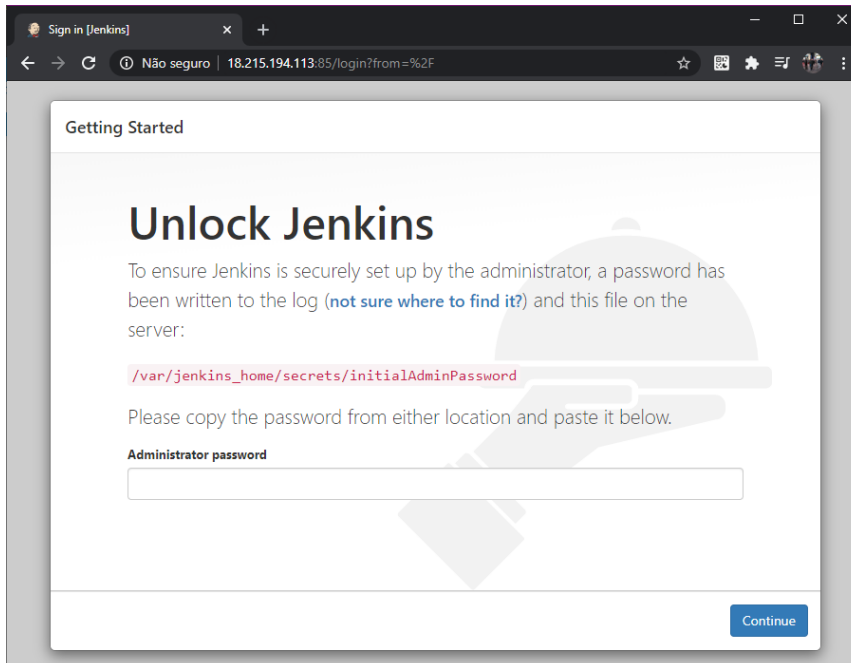
Entrar no container jenkins e executar

```
ssh jenkins@ip-elastico
```

Se conectar sem pedir senha significa que funcionou!

6. 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 singletons
in org.springframework.beans.factory.support.DefaultListableBeanFactory@59e418b2: defining beans [filter,legacy]; root of factory hierarchy
2020-06-26 19:52:43.216+0000 [id=25] INFO jenkins.install.SetupWizard#init:

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

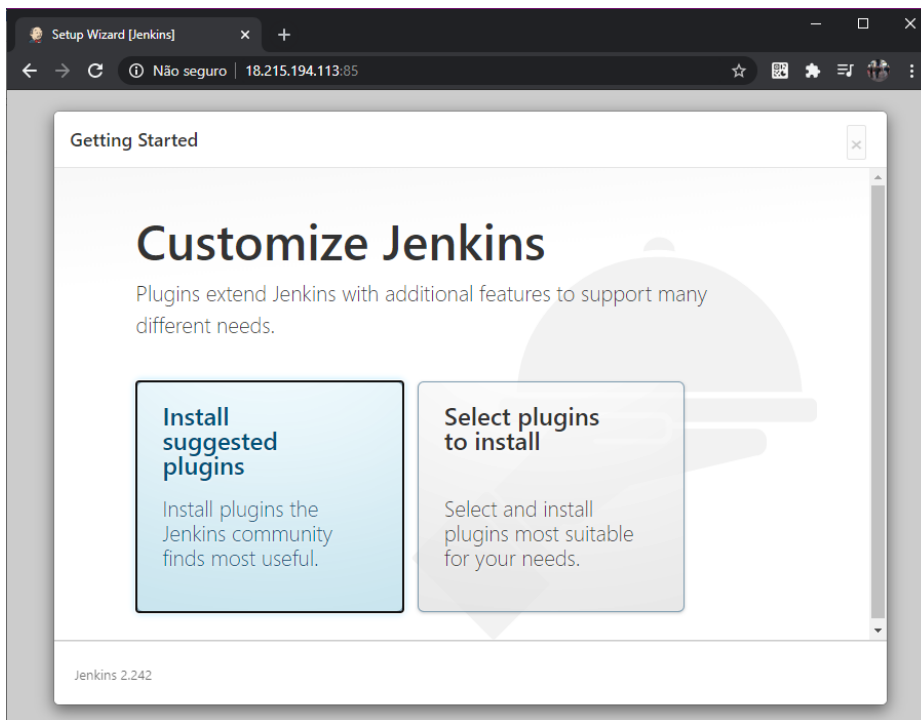
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

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$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
2020-06-26 19:52:48.262+0000 [id=40] INFO hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
2020-06-26 19:52:48.265+0000 [id=40] INFO hudson.model.AsyncPeriodicWork$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!

Setup Wizard [Jenkins]

Não seguro | 18.215.194.113:85

Getting Started

Create First Admin User

Nome de usuário:

Senha:

Confirmar a senha:

Nome completo:

Endereço de e-mail:

Jenkins 2.242

[Skip and continue as admin](#) [Save and Continue](#)

Setup Wizard [Jenkins]

Não seguro | 18.215.194.113:85

Getting Started

Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.242

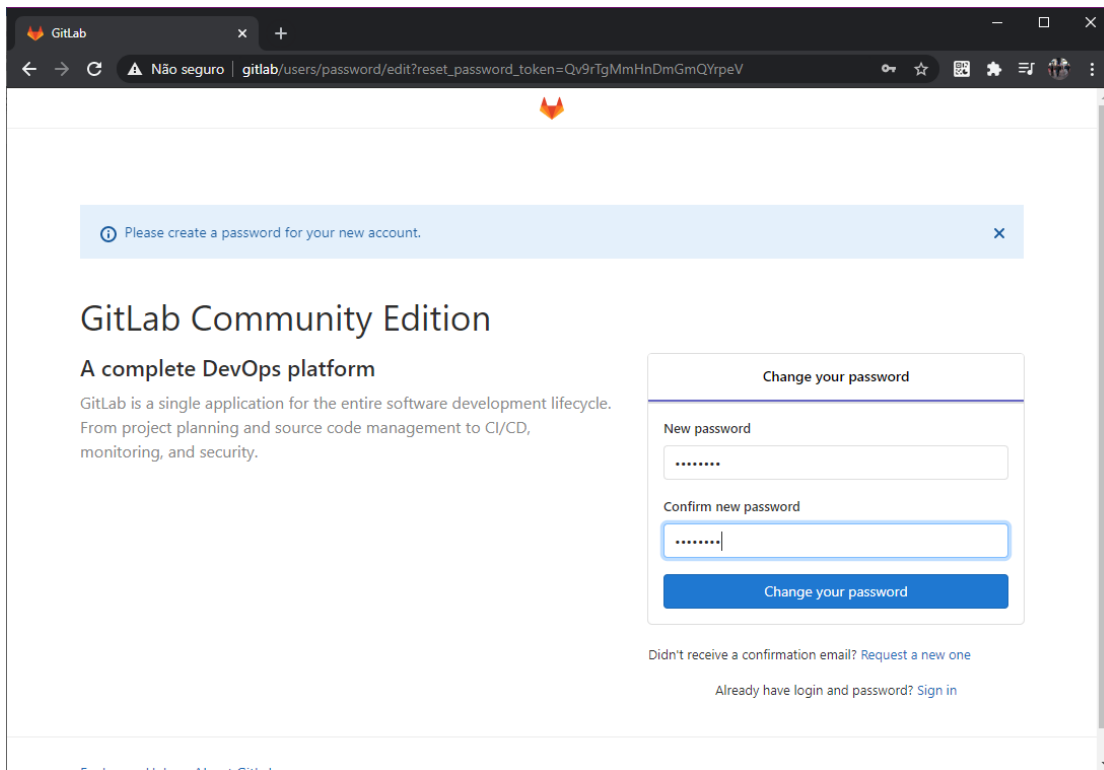
[Not now](#) [Save and Finish](#)

7. 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 ?  
# lines or following the machine name denoted by a '#' symbol  
#  
# For example:  
#  
# 102.54.94.97 rhino.acme.com # source server  
# 38.25.63.10 x.acme.com # x client host  
# localhost name resolution is handled within DNS itself.  
18.215.194.113 gitlab
```

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.

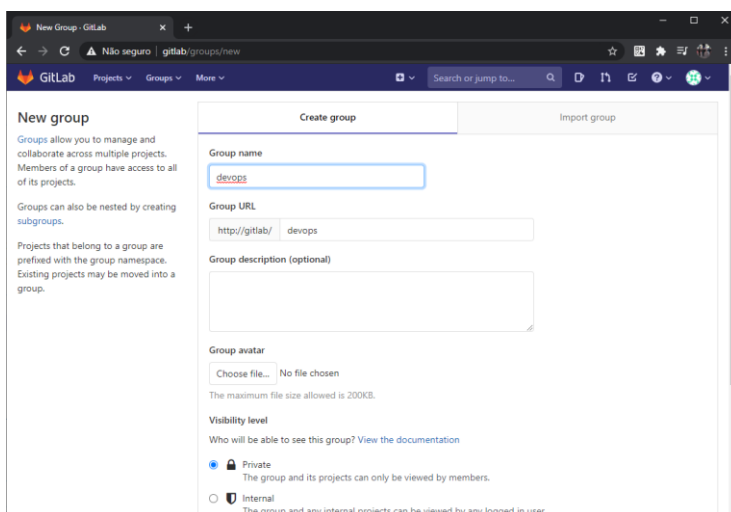


Agora estamos com o ambiente pronto e podemos começar a configuração da integração. Segura, que o filho é seu!

8. CRIAÇÃO DO REPOSITÓRIO NO GITLAB

Vamos criar um repositório no Gitlab para armazenar nosso projeto.

Crie o grupo devops:



- 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

← → ↻ ⓘ Não seguro | gitlab/devops/spring-boot-hello

GitLab Projects Groups More 🔍 Search or jump to...

spring-boot-hello

Project overview
Details
Activity
Releases

Repository
Issues 0
Merge Requests 0
CI / CD
Operations
Analytics
Wiki
Snippets
Members
Settings

devops > spring-boot-hello > Details

spring-boot-hello
Project ID: 1

4 Commits 1 Branch 0 Tags 512 KB Files 512 KB Storage

master spring-boot-hello / + History

Update HelloSpringBootApplication.java
Administrator authored 4 hours ago

📄 README 📄 Add LICENSE 📄 Add CHANGELOG 📄 Add CONTRIBUTING 📄 Enable

📄 Set up CI/CD

Name	Last commit
📁 src	Update HelloSpringBootApplication.java
📄 README.md	Add README.md
📄 mvnw	initial
📄 mvnw.cmd	initial
📄 pom.xml	initial

📄 README.md

lab hello world spring boot!

9. 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*.

The screenshot shows the 'Profile Settings' page in GitLab, specifically the 'Access Tokens' tab. On the left, there's a section titled 'Personal Access Tokens' explaining their use. The main area is 'Add a Personal Access Token', where a name 'Jenkins' has been entered in the 'Name' field. The 'Expires at' field is empty. Under 'Scopes', 'api (Access your API)' and 'read_user (Read user information)' are selected. A green 'Create Personal Access Token' button is at the bottom.

Após a criação, copie o hash exibido na tela, porque vamos usá-lo no próximo passo.

This screenshot shows the 'Your New Personal Access Token' section. A text box displays the token 'rQCupLMV7Aqxu_-35rsw'. Below it, a red warning message states: 'Make sure you save it - you won't be able to access it again.'

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

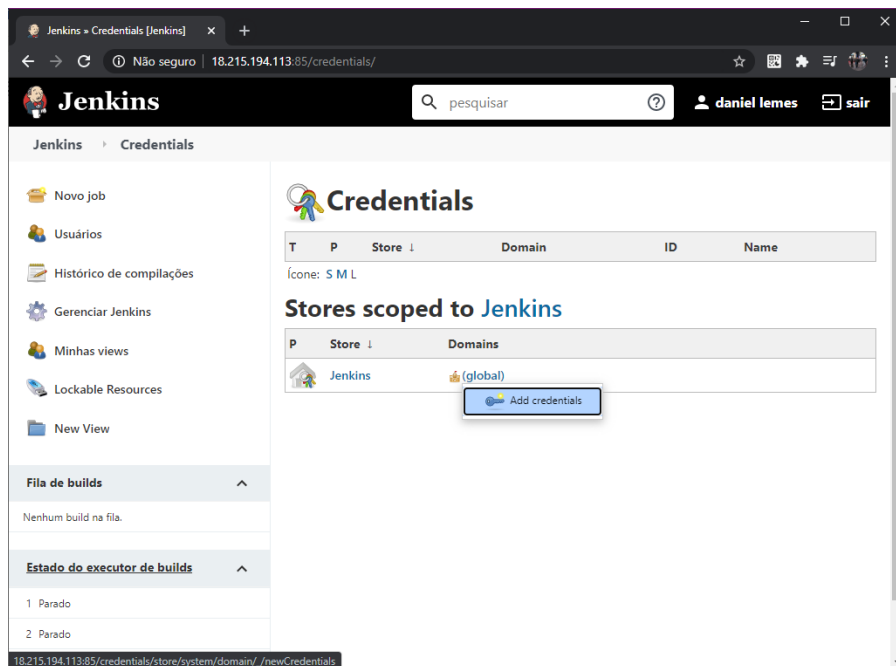
10.JENKINS – INSTALANDO GITLAB PLUGIN

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*.

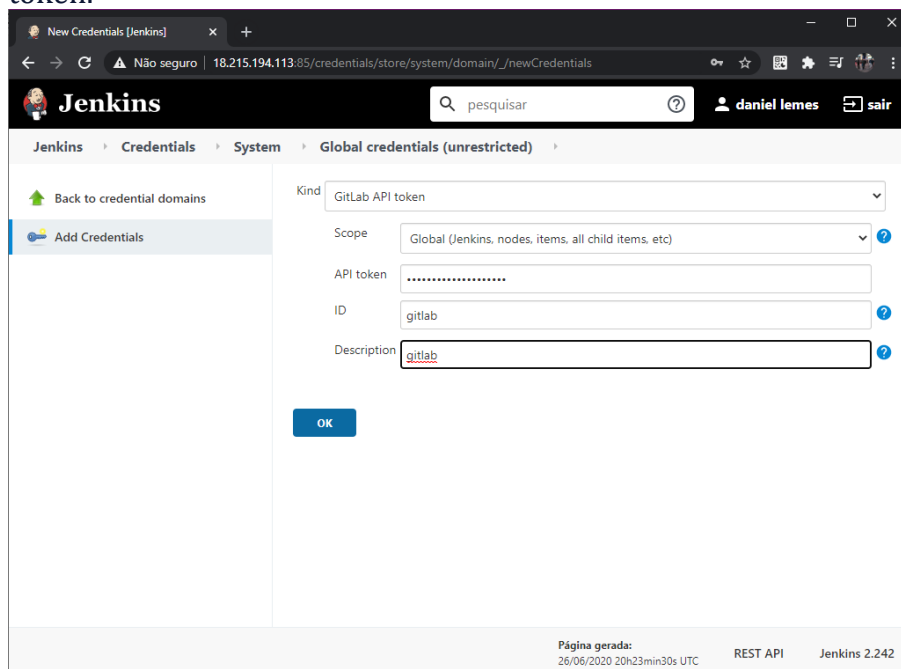
The screenshot shows the Jenkins 'Gerenciador de plugins' (Plugin Manager) page. The 'Disponíveis' (Available) tab is selected. A search bar contains 'gitlab'. Two plugins are listed: 'GitLab Triggers de builds' (version 1.5.13, released 9 months ago) and 'GitLab Hook Gerenciamento de código fonte' (version 1.4.2, released 4 years ago). The 'GitLab Hook' plugin has a red warning box stating it may not be safe to use due to security issues: 'GitLab API token stored and displayed in plain text' and 'Reflected XSS vulnerability'. At the bottom, there are buttons for 'Instalar sem reiniciar' and 'Baixar agora, instalar e depois reiniciar'.

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.



Após clicar em OK, você verá sua credencial criada.



Global credentials (unrestricted)

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

Name	Kind	Description
 GitLab API token (Melhor Gitlab)	GitLab API token	Melhor Gitlab 

Ícone: [S](#) [M](#) [L](#)

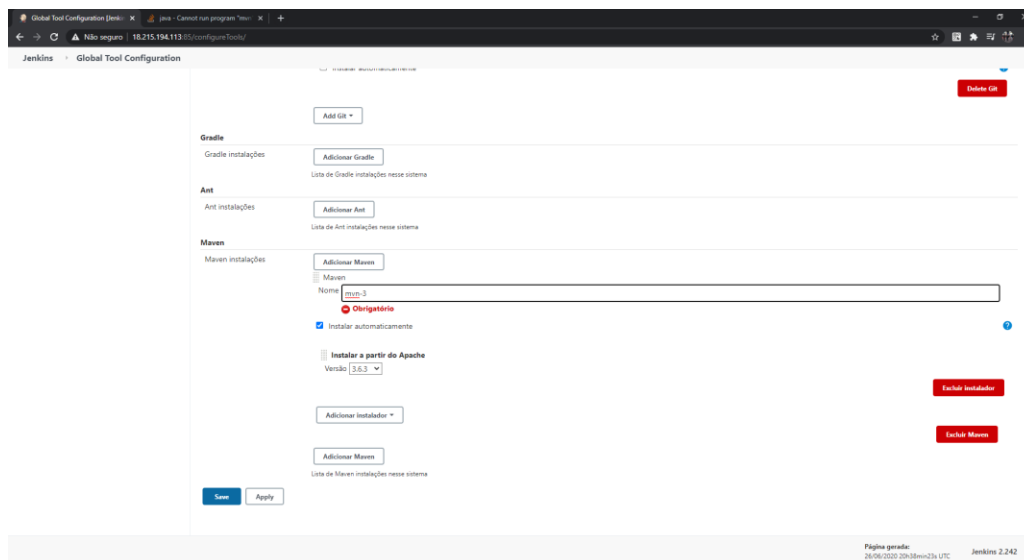
11. JENKINS - CONFIGURANDO A CONEXÃO COM GITLAB

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

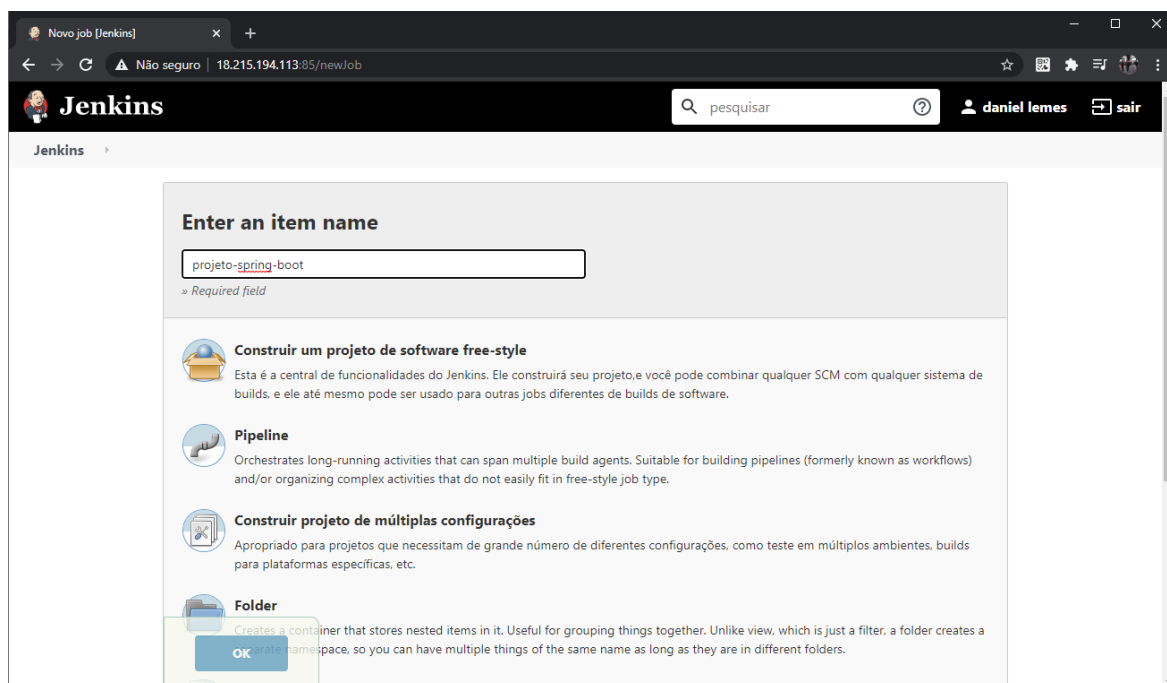
Marcar ignore ssl certificate errors:

12. CONFIGURAR MAVEN EM JENKINS>GLOBAL TOOL CONFIGURATION



13.CRIAÇÃO DO JOB NO JENKINS

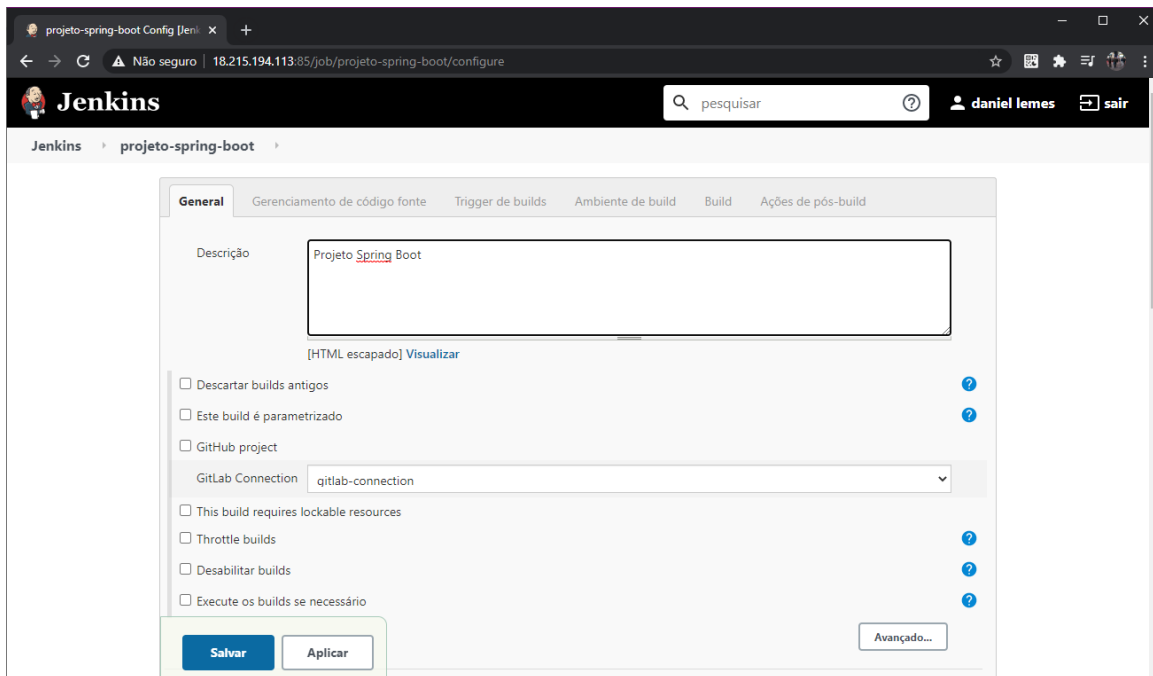
No canto esquerdo, clique em *Novo Job*, escolha *Projeto Freestyle*, digite um nome para o Job e clique em *OK*.



Com o Job criado, vamos às configurações.

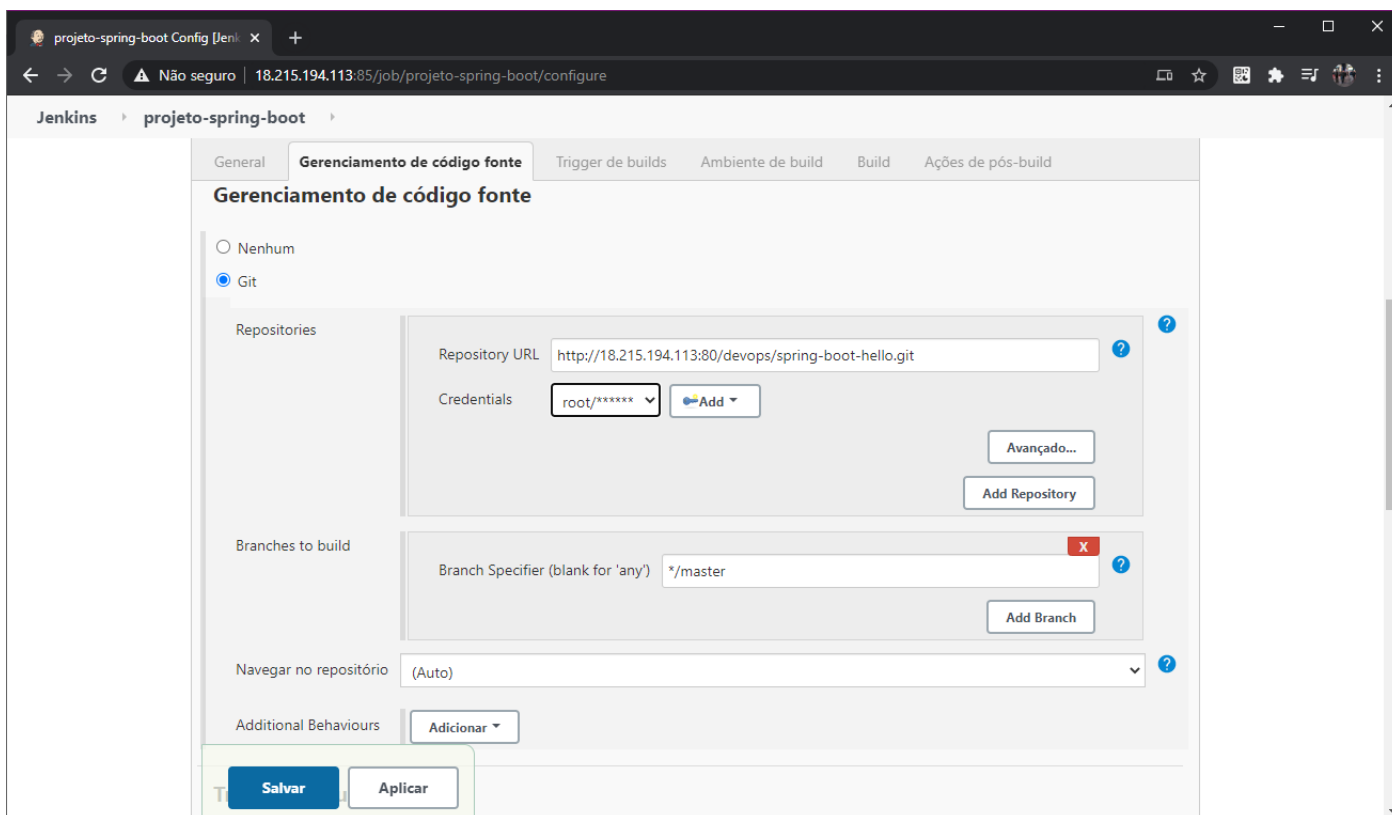
General

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

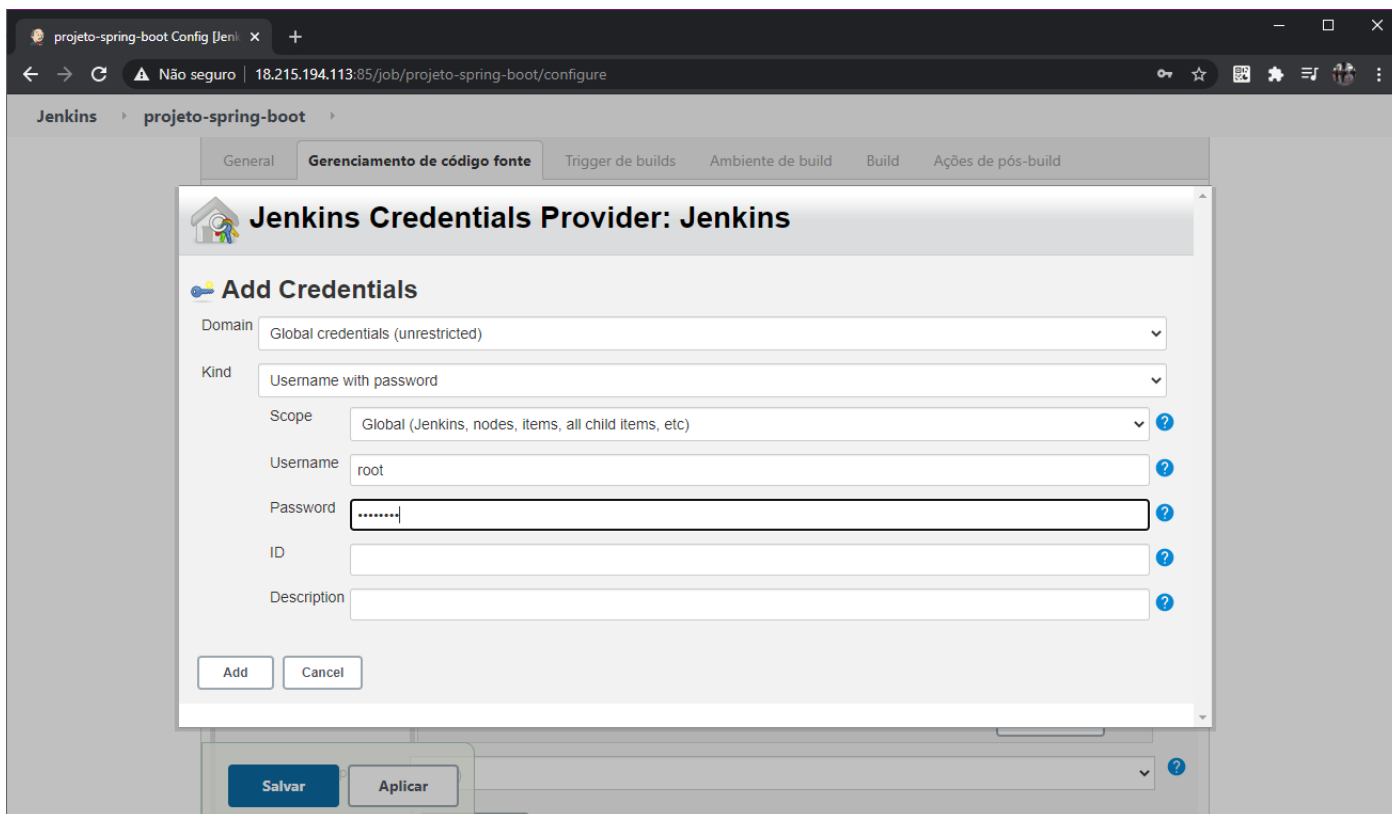


Gerenciamento de código fonte

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



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



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.

Marque delete workspace before build starts:

Ambiente de Build

Jenkins > projeto-spring-boot >

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

Ambiente de build

☒ Delete workspace before build starts

Avançado...

☐ Use secret text(s) or file(s)

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Inspect build log for published Gradle build scans

☐ With Ant

Configurar build:

Jenkins > projeto-spring-boot >

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

☐ Add timestamps to the Console Output

☐ Inspect build log for published Gradle build scans

☐ With Ant

Build

Chamar alvos Maven de alto nível

Versão do Maven

Goals

Avançado...

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

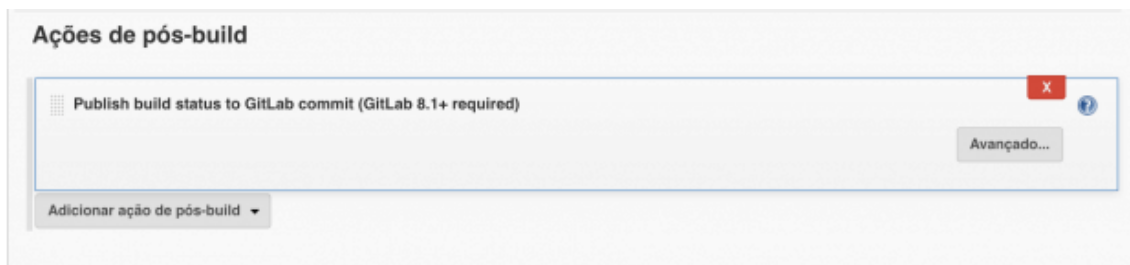
Comando `ssh jenkins@18.215.194.113 <<'ENDSSH'`
`echo "java -jar /tmp/hello-spring-boot-0.0.1-SNAPSHOT.jar" | at now + 1 min`
`ENDSSH`

Veja a lista de variáveis de ambiente disponíveis

Avançado...

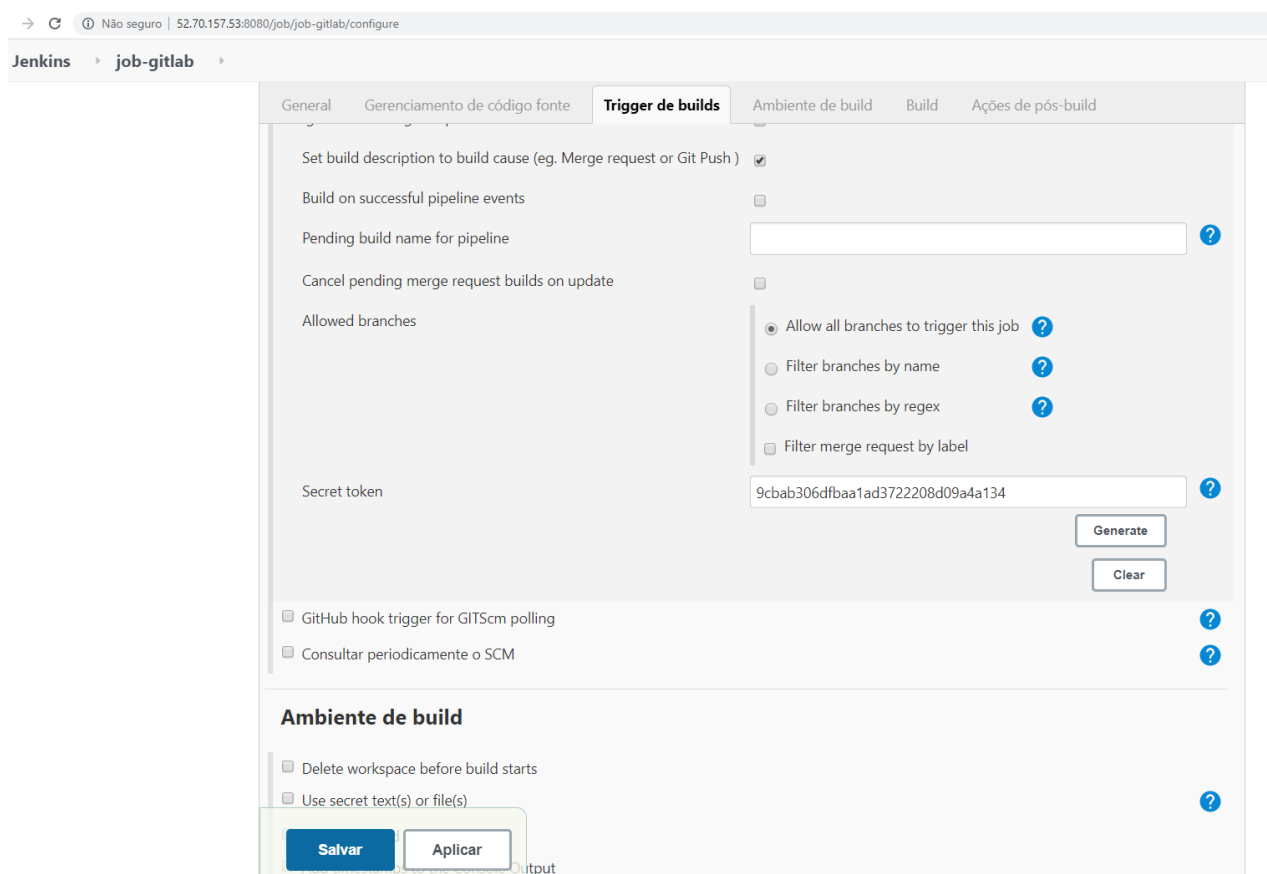
Ações de pós-build

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.



Clique em Salvar e o Job está finalizado.

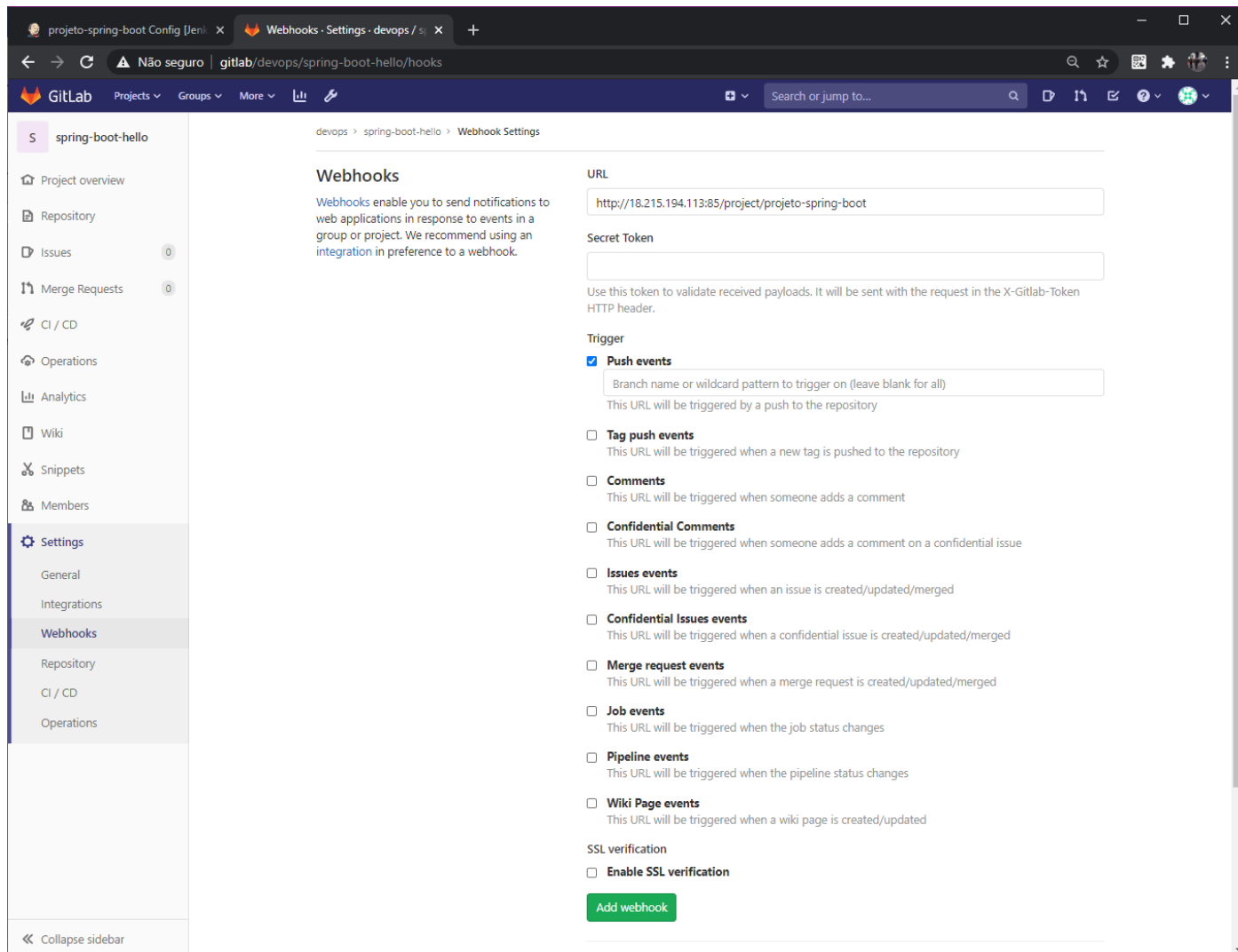
Depois de salvar o job, entrar novamente nas configurações do job, em trigger de builds e clicar em avançado, generate token (salve a hash gerada no notepad ou área de transferência) e clique em salvar.



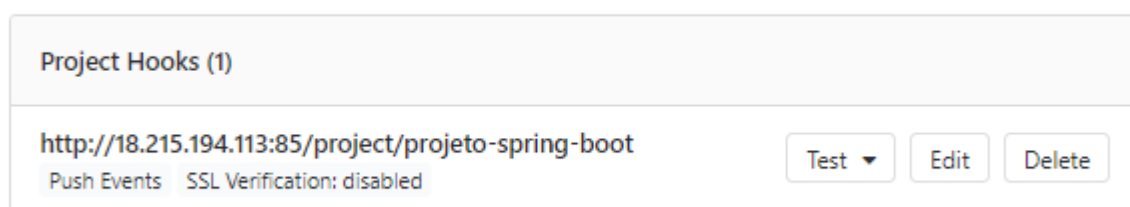
14. CRIANDO WEBHOOK NO GITLAB

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*.

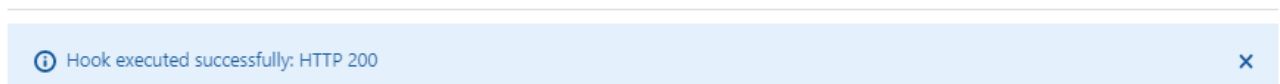


15.TESTANDO O PIPELINE (SELECIONE TEST E PUSH)



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.

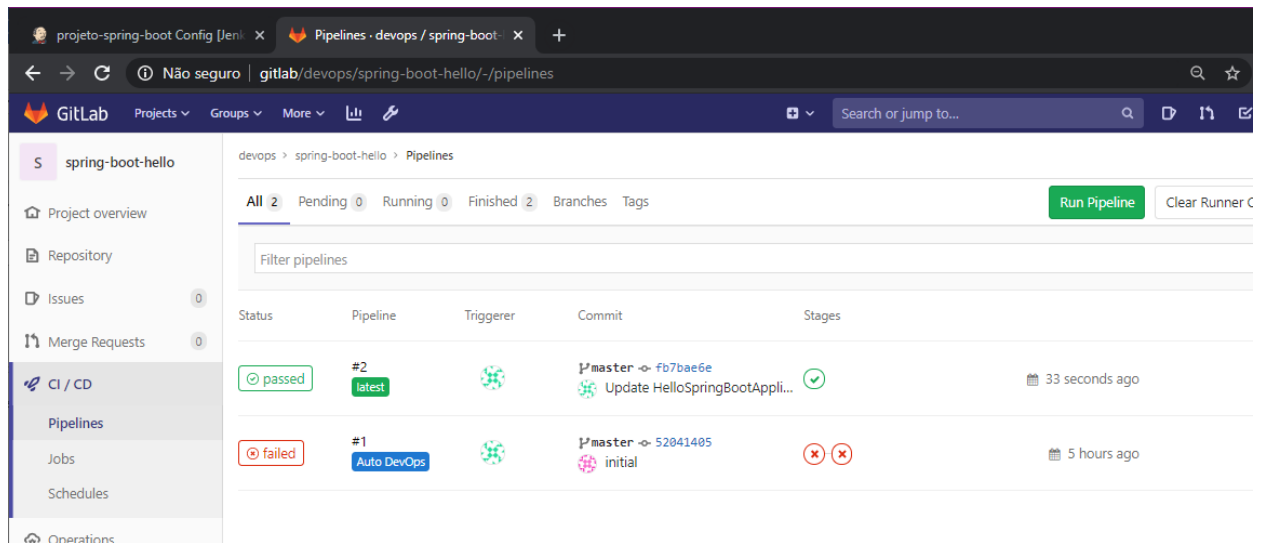
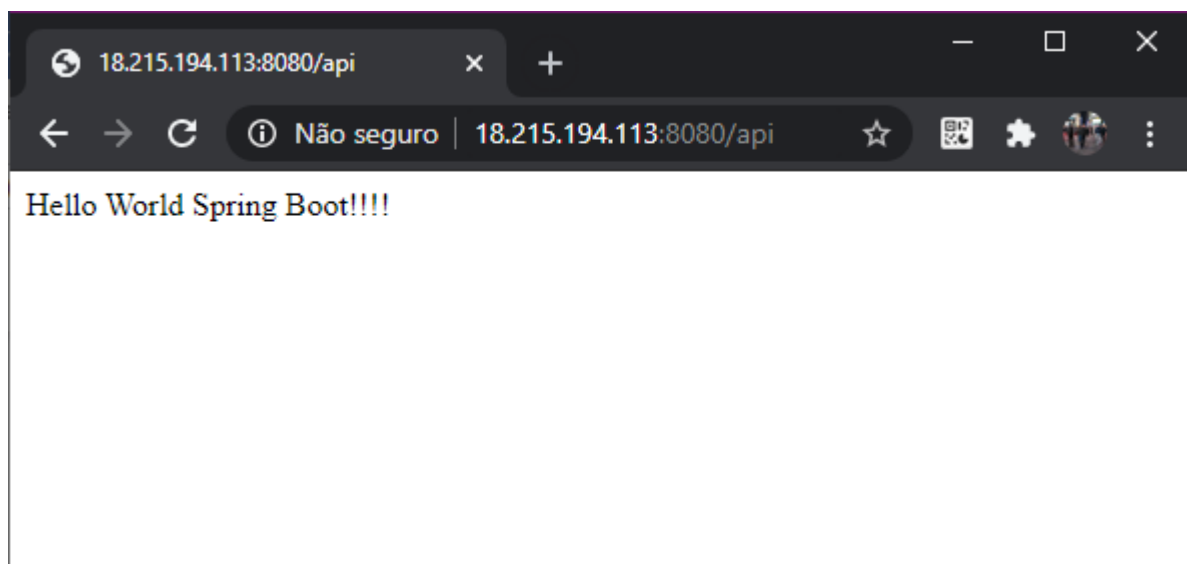


Figura 1 menu pipeline do repositório

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



Referências:

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