

# DevOps Módulo 9 - IT Talent

Escrito por: [Gabriel Oliveira dos Santos](#)

Github: <https://github.com/Hypothesis>

## Jenkins



# Jenkins

Jenkins é um site que ajuda na atumação do nosso código fonte, do nosso projeto. Ele ajudar a fazermos o CI/CD, assim automatizando as partes do desenvolvimento de software relacionados a construção, teste e implementação.






## Soma Hash e Jenkins

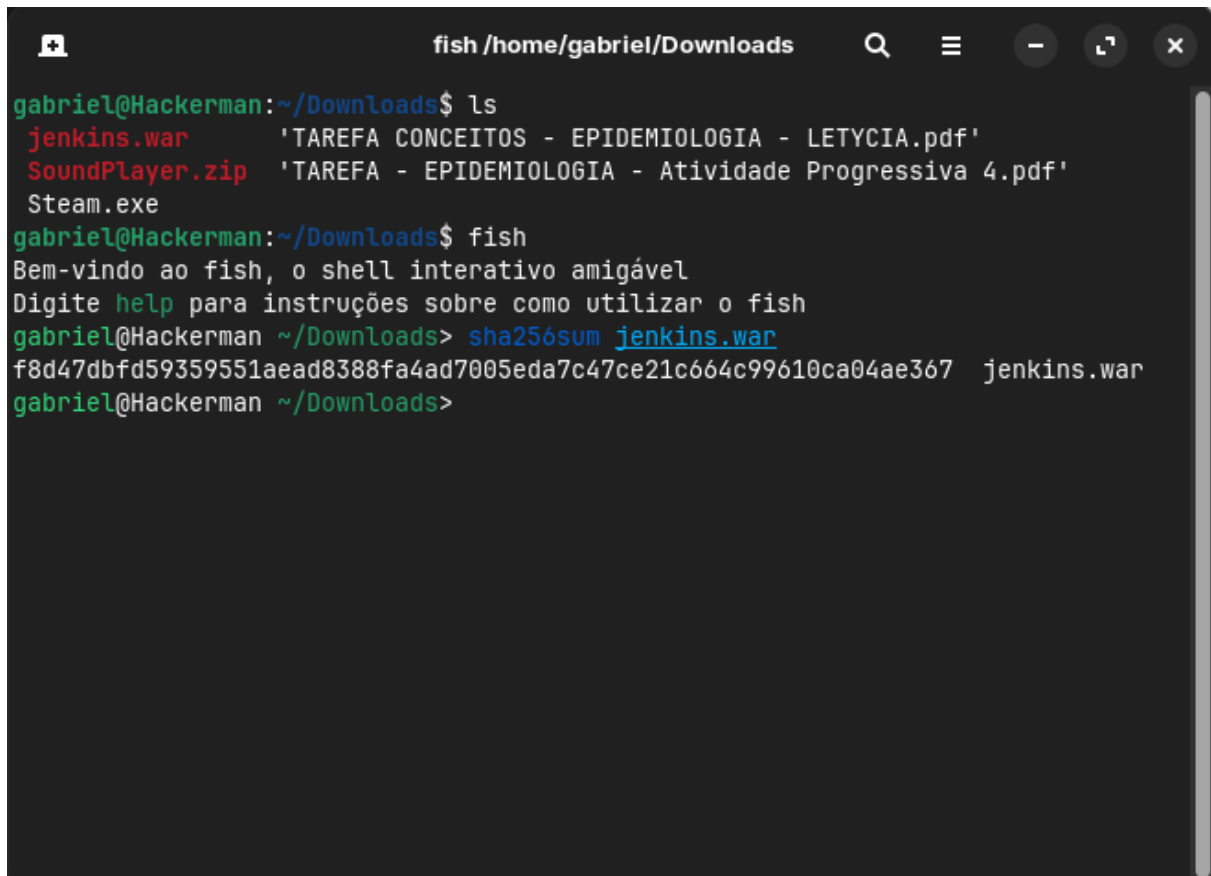
# Downloading Jenkins

Jenkins is distributed as WAR files, native packages, installers, and Docker images. Follow the steps below to download Jenkins.

1. Before downloading, please take a moment to review the [Hardware and Software requirements](#).
2. Select one of the packages below and follow the download instructions.
3. Once a Jenkins package has been downloaded, proceed to the [Installing Jenkins](#) section.
4. You may also want to verify the package you downloaded. [Learn more about verifying Jenkins](#)

Download Jenkins 2.440.3 LTS for:

	<b>Generic Java package (.war)</b> SHA-256: f8d47dbfd59359551aead8388fa4ad7005eda7c47ce21c664c99610ca04ae367	
	<b>Docker</b>	
	<b>Kubernetes</b>	
	<b>Ubuntu/Debian</b>	
	<b>Red Hat/Fedora/Alma/Rocky/CentOS</b>	
	<b>Windows</b>	

A terminal window titled 'fish /home/gabriel/Downloads' with standard window controls. The user 'gabriel@Hackerman' is in the directory '~/Downloads'. They run 'ls', showing files: 'jenkins.war', 'SoundPlayer.zip', and 'Steam.exe'. Then they run 'fish', which shows a welcome message and instructions. Finally, they run 'sha256sum jenkins.war', which outputs a long hash followed by 'jenkins.war'.

```
gabriel@Hackerman:~/Downloads$ ls
jenkins.war      'TAREFA CONCEITOS - EPIDEMIOLOGIA - LETYCIA.pdf'
SoundPlayer.zip  'TAREFA - EPIDEMIOLOGIA - Atividade Progressiva 4.pdf'
Steam.exe

gabriel@Hackerman:~/Downloads$ fish
Bem-vindo ao fish, o shell interativo amigável
Digite help para instruções sobre como utilizar o fish
gabriel@Hackerman ~/Downloads> sha256sum jenkins.war
f8d47dbfd59359551aead8388fa4ad7005eda7c47ce21c664c99610ca04ae367  jenkins.war
gabriel@Hackerman ~/Downloads>
```

A soma hash serve para a integridade do nosso arquivo .war que instalamos do site jenkins para nosso computador, assim é possível aplicar um hash de entrada e obter um hash de saída, se o hash é o mesmo que está no site, temos um jenkins verídico. A soma hash serve muito mais para testarmos nossa aplicação para integridade.

Pela imagem do terminal, vimos que com o sha256sum (soma hash), da a mesma saída do hash que aparece no site, assim garantindo nossa que nossa aplicação instalada não está corrompido.

E depois do hash temos uma variavel JENKINS\_HOME que tem os arquivos de configurações do jenkins.

A instação do Jenkins estão abaixo na imagem, que está presente no site.

## Run the WAR file

The Jenkins Web application ARchive (WAR) file can be started from the command line like this:

1. Download the [latest Jenkins WAR file](#) to an appropriate directory on your machine
2. Open up a terminal/command prompt window to the download directory
3. Run the command `java -jar jenkins.war`
4. Browse to `http://localhost:8080` and wait until the **Unlock Jenkins** page appears
5. Continue on with the [Post-installation setup wizard](#) below

### Notes:

- This process does not automatically install any specific plugins. They need to be installed separately via the [Manage Jenkins](#) > [Plugins](#) page in Jenkins.
- You can change the port by specifying the `--httpPort` option when you run the `java -jar jenkins.war` command. For example, to make Jenkins accessible through port 9090, then run Jenkins using the command:  
`java -jar jenkins.war --httpPort=9090`
- You can change the directory where Jenkins stores its configuration with the `JENKINS_HOME` environment variable. For example, to place the Jenkins configuration files in a subdirectory named `my-jenkins-config`, define `JENKINS_HOME=my-jenkins-config` before running the `java -jar jenkins.war` command. Use the Windows commands:

#### Windows

```
C:\Temp > set JENKINS_HOME=my-jenkins-config
C:\Temp > java -jar jenkins.war
```

or the Unix command:

#### Unix

```
JENKINS_HOME=my-jenkins-config java -jar jenkins.war
```

depois de instalar, acessar o link abaixo e colocar o código que aparece no terminal da nossa instalação, vamos instalar plugins e depois acessar uma página para a iniciação do nosso jenkins, em que vamos colocar nosso nome de usuário, senha e entre outros, como outra imagem abaixo.

`http://localhost:8080`

← → ↺

localhost:8080

🔑 ☆ 🔊 🔥 📄 📁 ⬇️ 🌐 ⋮

Iniciando

# Criar o primeiro usuário administrativo

Nome de usuário

Senha

Confirmar a senha

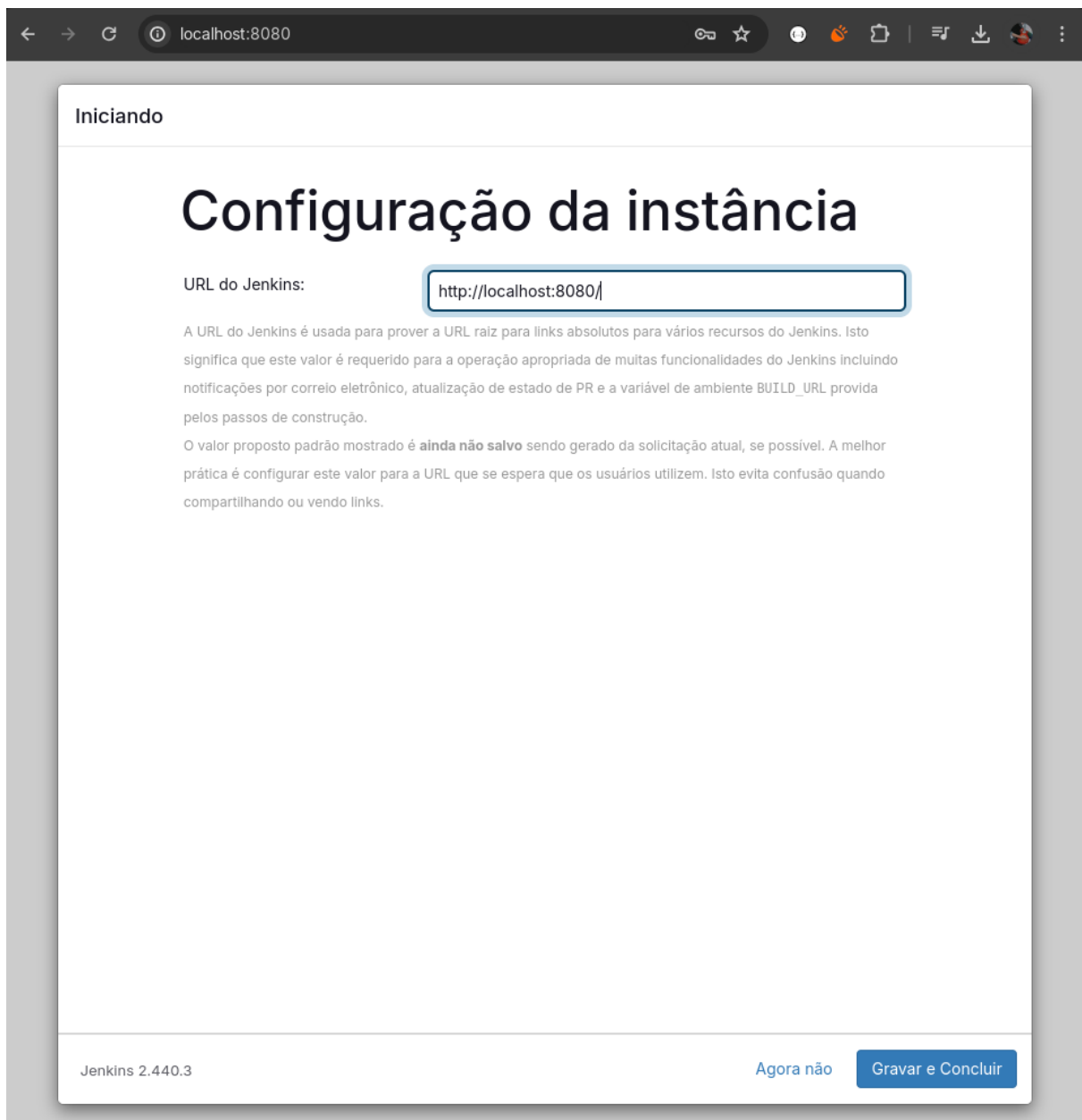
Nome completo

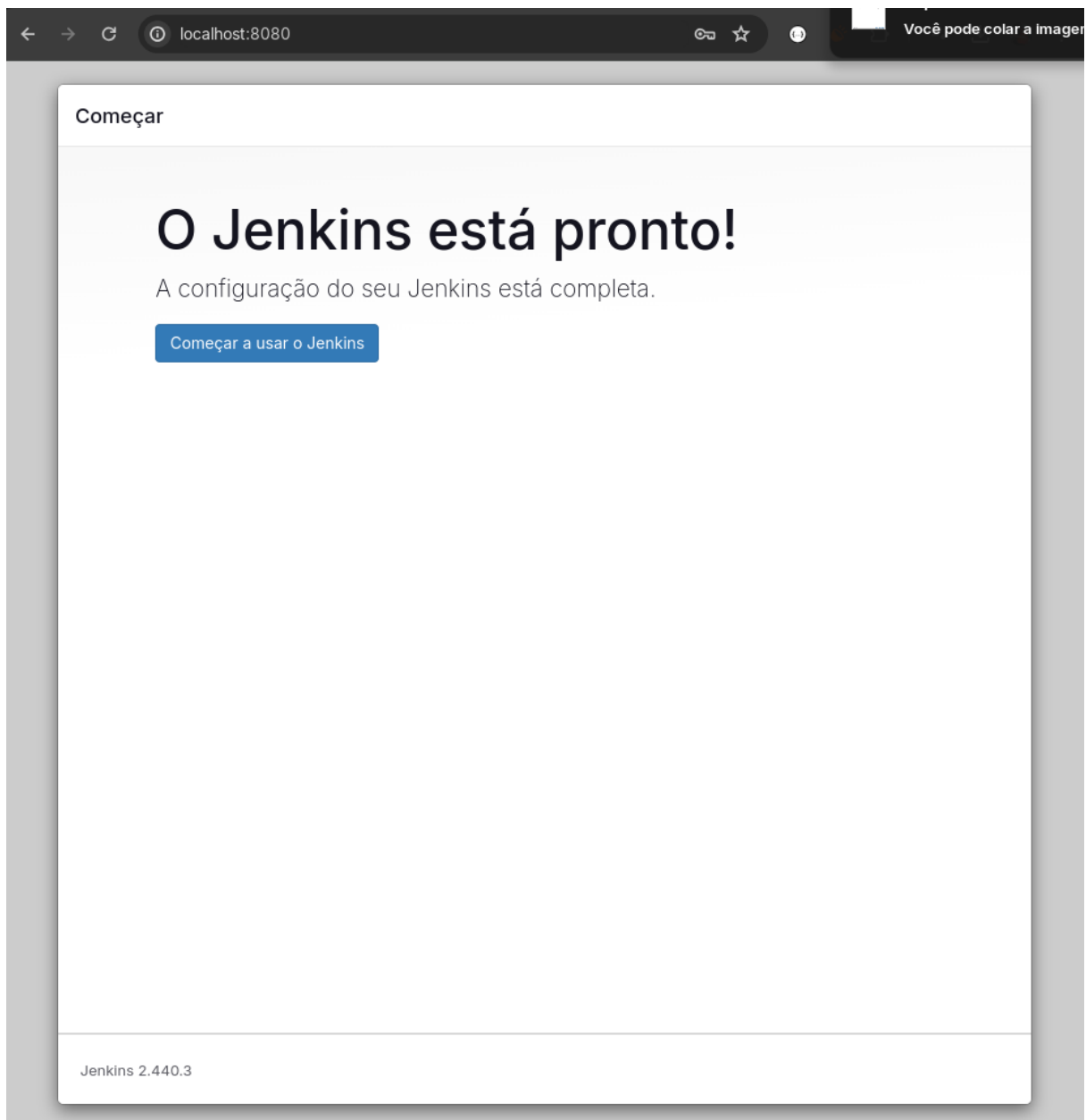
Endereço de e-mail

Jenkins 2.440.3

[Pular e continuar como administrador](#)

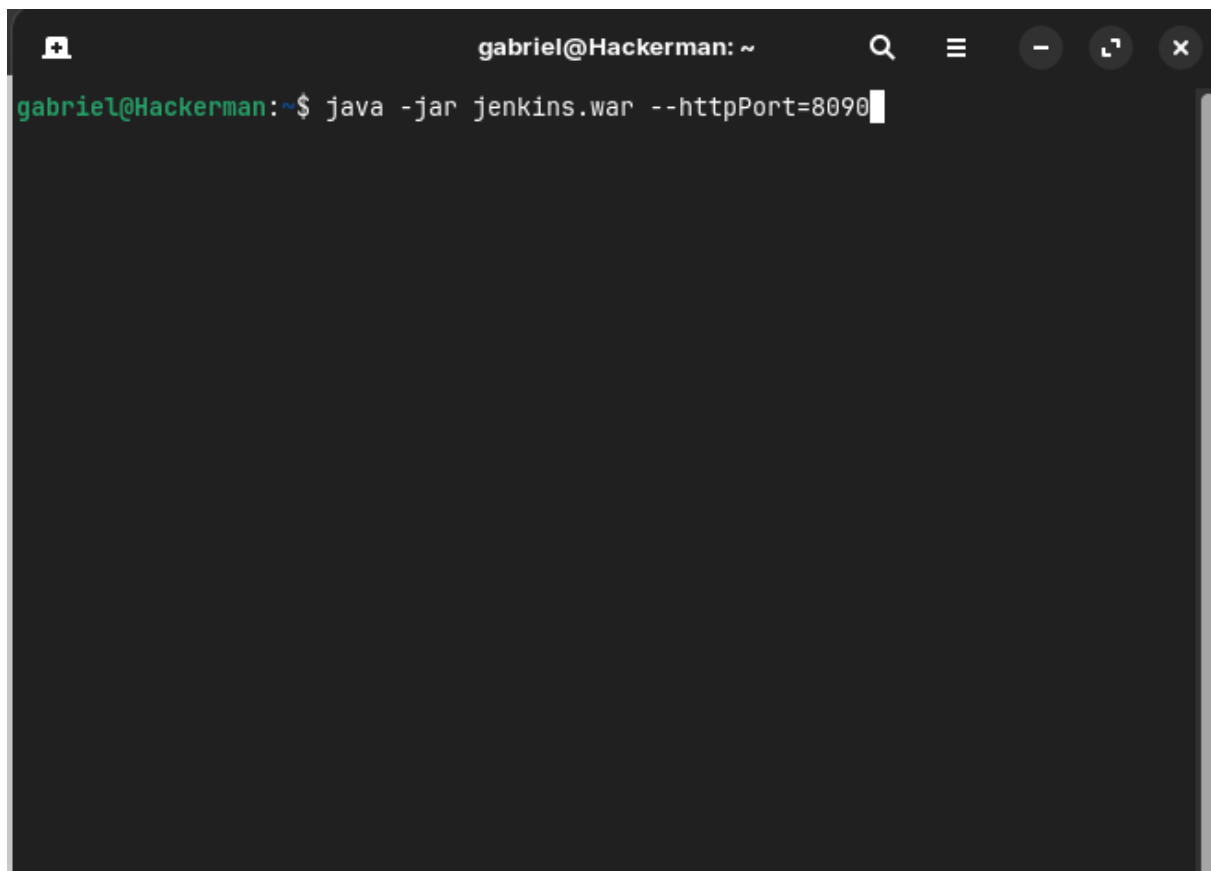
Salvar e continuar





Assim para fechar o jenkins é so apertar Crtl+C no terminal, interrompendo o processo do jenkins, já para iniciar de novo, é so executar o arquivo .war jenkins.

Já a imagem abaixo é so um bonus, em que você pode escolher a porta em que o nosso jenkins vai rodar, se não executarmos esse comando ele vai rodar na porta padrão que foi escolhida na instalação, que normalmente é a porta 8080, se caso você não tenha mudado a porta durante a instalação.


A terminal window with a dark background. The title bar at the top shows a window icon, the text 'gabriel@Hackerman: ~', and standard window controls (search, menu, zoom, and close). The terminal content shows the prompt 'gabriel@Hackerman:~\$' followed by the command 'java -jar jenkins.war --httpPort=8090' and a cursor at the end of the line.

```
gabriel@Hackerman: ~  
gabriel@Hackerman:~$ java -jar jenkins.war --httpPort=8090
```

## Fluxo de criação para um trabalho simples

No dashboard da parte inicial do nosso jenkins, podemos criar um trabalho, e ao clicar podemos escrever o nome do nosso trabalho e clicamos na primeira opção (Construir um Projeto de Software de Estilo Livre), escrever uma descrição e selecionar os itens de acordo com a nossa necessidade.



 Adicionar descrição

# Welcome to Jenkins!

Esta página é onde o seus trabalhos do Jenkins serão exibidos. Para começar, você pode configurar construções distribuídas ou começar a construir um projeto de software.

## Comece a construir seu projeto de software

Criar um trabalho



## Configure uma construção distribuída

Configure um agente



Configurar uma nuvem



Aprenda mais sobre construções distribuídas



## Entre com um nome de item

Trabalho de Build 1

» Campo requerido



### **Construir um projeto de software de estilo livre.**

Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.



### **Pipeline**

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



### **Construir projeto de múltiplas configurações**

Apropriado para projetos que necessitam de grande número de diferentes configurações, como teste em múltiplos ambientes, builds para plataformas específicas, etc.



### **Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.



### **Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.



### **Organization Folder**

Creates a set of multibranch project subfolders by scanning for repositories.

## Geral

Habilitado



### Descrição

Trabalho de build simples para teste

HTML escapado [Visualizar](#)

- ☐ Descartar construções antigas ?
- ☐ Esta construção é parametrizada ?
- ☐ GitHub project
- ☐ Throttle builds ?
- ☐ Execute as construções se necessário ?

Avançado ▼

## Gerenciamento de código fonte

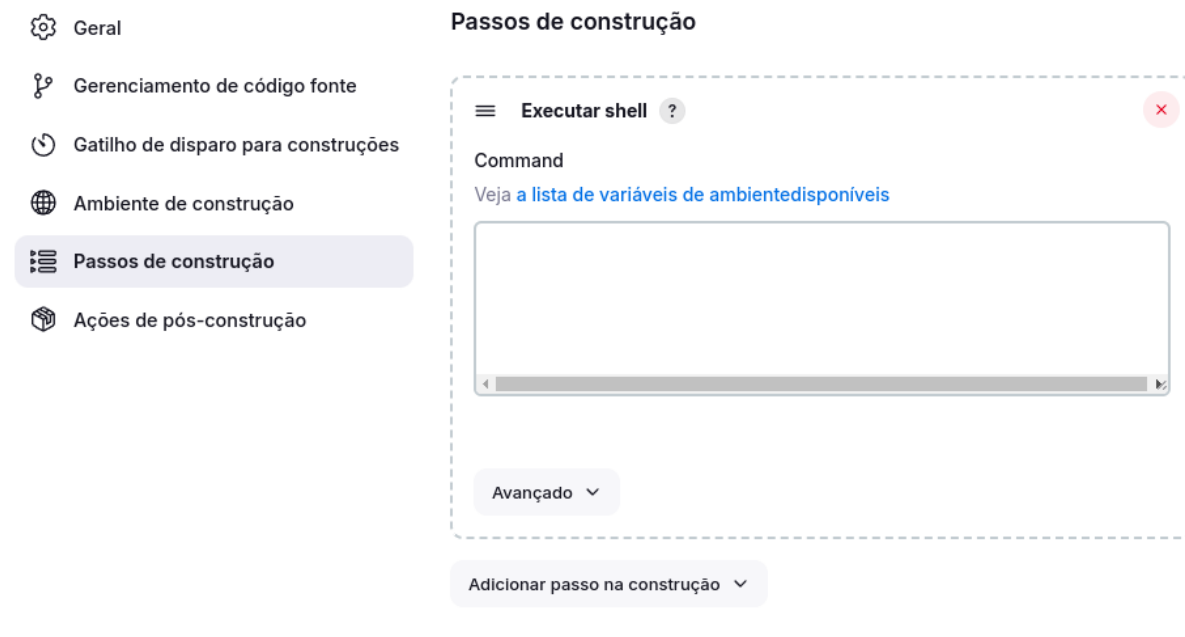
☒ Nenhum

☐ Git ?

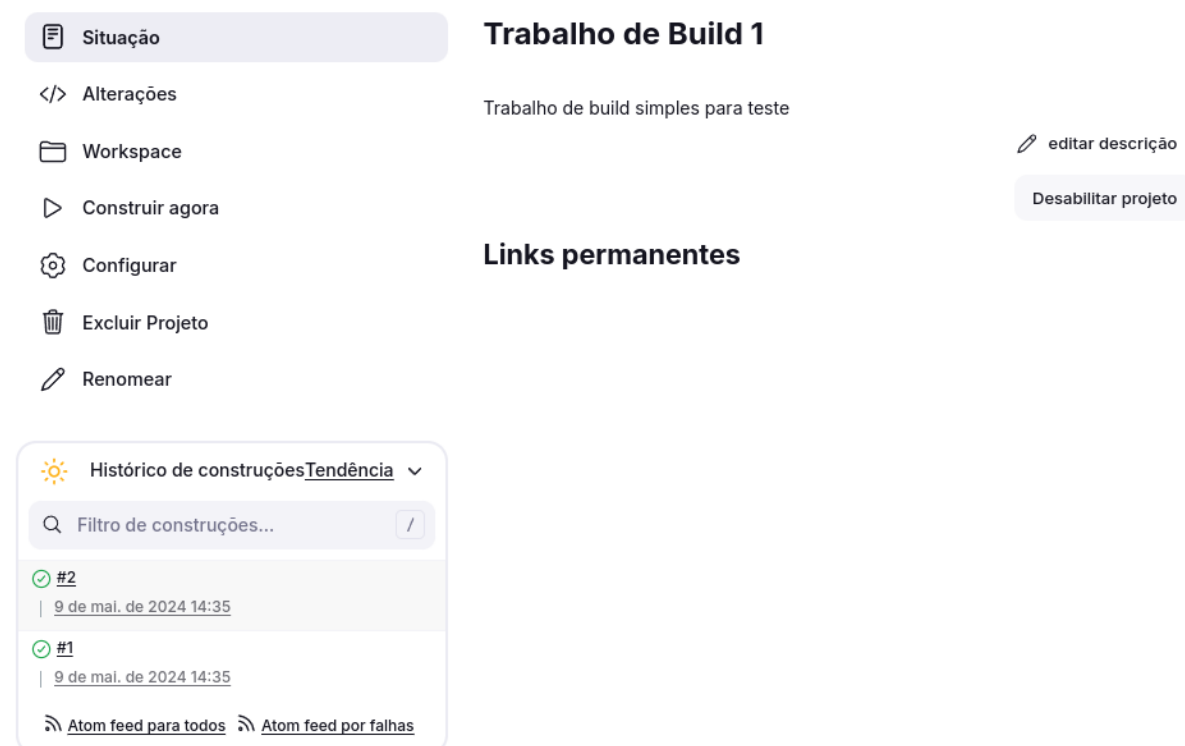
Salvar

Aplicar

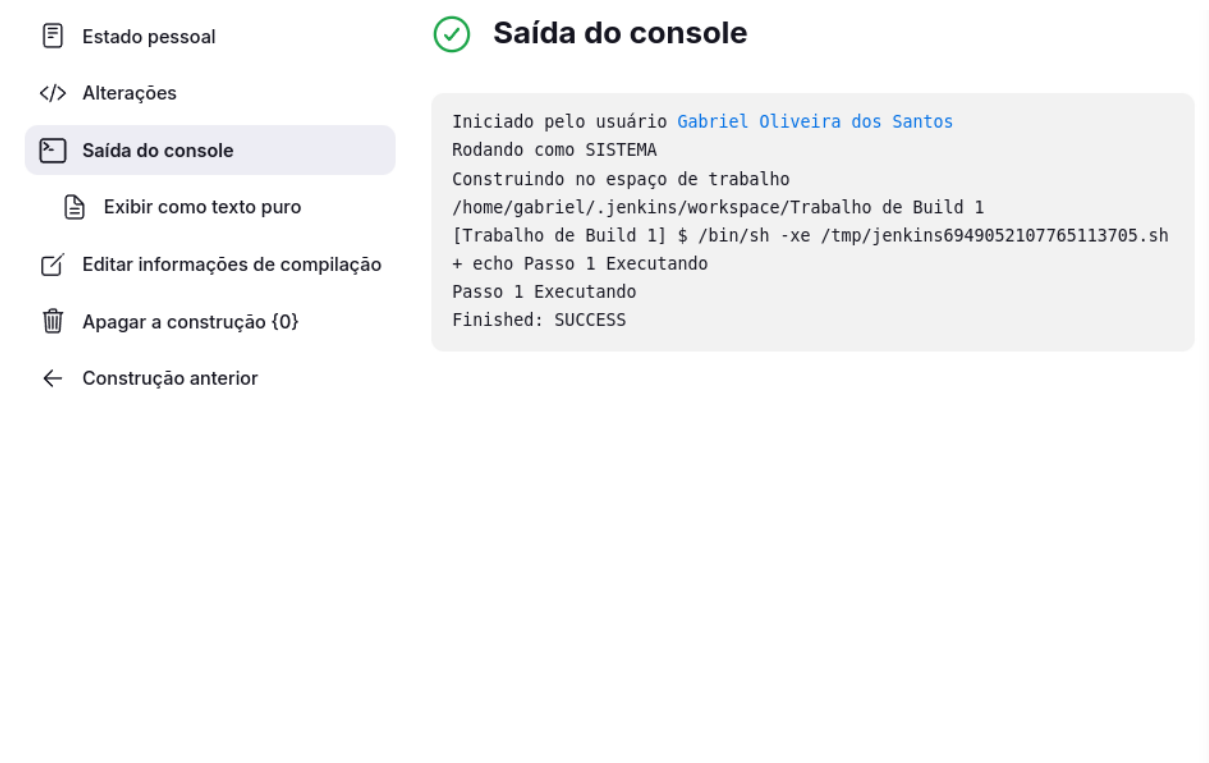
Podemos criar passos de construção, assim podemos fazer um shell para ser executado no passo de construção. Podemos escrever o que quisermos, assim cada "shell" é agendado, assim tem seu tempo de execução e quando vai para construção.



Assim podemos construir nossa aplicação, e essas aplicações são agendadas.



E na saída do nosso console, ao clicarmos na nossa aplicação, onde fica o Histórico de construções. Essa saída do nosso console é como se fosse o Hello World da programação, assim podemos ver se a nossa construção está rodando e vendo no console a saída e execução do nosso script.



The screenshot displays the Jenkins web interface for a build. On the left, a sidebar contains several menu items: 'Estado pessoal' (Personal state), 'Alterações' (Changes), 'Saída do console' (Console output), 'Exibir como texto puro' (Show as plain text), 'Editar informações de compilação' (Edit compilation information), 'Apagar a construção {0}' (Delete build {0}), and 'Construção anterior' (Previous build). The 'Saída do console' item is selected and highlighted. The main area on the right, titled 'Saída do console' with a green checkmark icon, shows the build's output. The output text indicates the build was initiated by 'Gabriel Oliveira dos Santos', ran as 'SISTEMA', and was executed in the workspace '/home/gabriel/.jenkins/workspace/Trabalho de Build 1'. It shows a shell command being run: '[Trabalho de Build 1] \$ /bin/sh -xe /tmp/jenkins6949052107765113705.sh', followed by 'Passo 1 Executando' and 'Finished: SUCCESS'.

Estado pessoal

</> Alterações

**Saída do console**

Exibir como texto puro

Editar informações de compilação

Apagar a construção {0}


← Construção anterior


**Saída do console**

```
Iniciado pelo usuário Gabriel Oliveira dos Santos
Rodando como SISTEMA
Construindo no espaço de trabalho
/home/gabriel/.jenkins/workspace/Trabalho de Build 1
[Trabalho de Build 1] $ /bin/sh -xe /tmp/jenkins6949052107765113705.sh
+ echo Passo 1 Executando
Passo 1 Executando
Finished: SUCCESS
```

Existem uma lista de variáveis de ambientes disponíveis pelo Jenkins para nós rodarmos, ao clicar nesse link somos direcionados para uma página com essa lista.

## Passos de construção

 Executar shell 



Command

Veja [a lista de variáveis de ambiente disponíveis](#)

```
echo "Passo 1 Executando"
```

Avançado ▾

localhost:8090/env-vars.html/

Jenkins

pesquisar (CTRL+K)

Painel de controle >

As seguintes variáveis estão disponíveis para passos de construção de shell e batch:

**BRANCH\_NAME**  
For a multibranch project, this will be set to the name of the branch being built, for example in case you wish to deploy to production from master but not from feature branches; if corresponding to some kind of change request, the name is generally arbitrary (refer to `CHANGE_ID` and `CHANGE_TARGET`).

**BRANCH\_IS\_PRIMARY**  
For a multibranch project, if the SCM source reports that the branch being built is a primary branch, this will be set to "true"; else unset. Some SCM sources may report more than one branch as a primary branch while others may not supply this information.

**CHANGE\_ID**  
For a multibranch project corresponding to some kind of change request, this will be set to the change ID, such as a pull request number, if supported; else unset.

**CHANGE\_URL**  
For a multibranch project corresponding to some kind of change request, this will be set to the change URL, if supported; else unset.

**CHANGE\_TITLE**  
For a multibranch project corresponding to some kind of change request, this will be set to the title of the change, if supported; else unset.

**CHANGE\_AUTHOR**  
For a multibranch project corresponding to some kind of change request, this will be set to the username of the author of the proposed change, if supported; else unset.

**CHANGE\_AUTHOR\_DISPLAY\_NAME**  
For a multibranch project corresponding to some kind of change request, this will be set to the human name of the author, if supported; else unset.

**CHANGE\_AUTHOR\_EMAIL**  
For a multibranch project corresponding to some kind of change request, this will be set to the email address of the author, if supported; else unset.

**CHANGE\_TARGET**  
For a multibranch project corresponding to some kind of change request, this will be set to the target or base branch to which the change could be merged, if supported; else unset.

**CHANGE\_BRANCH**  
For a multibranch project corresponding to some kind of change request, this will be set to the name of the actual head on the source control system which may or may not be different from `BRANCH_NAME`. For example in GitHub or Bitbucket this would have the name of the origin branch whereas `BRANCH_NAME` would be something like PR-24.

**CHANGE\_FORK**  
For a multibranch project corresponding to some kind of change request, this will be set to the name of the forked repo if the change originates from one; else unset.

**TAG\_NAME**

## Passos de construção

≡ Executar shell ?

×

Command

Veja [a lista de variáveis de ambiente disponíveis](#)

```
echo "Passo 1 Executando"
echo "Número do build: "$BUILD_NUMBER
```

Avançado ▾

≡ Executar shell ?

×

Command

Veja [a lista de variáveis de ambiente disponíveis](#)

```
echo "ID do build: "$BUILD_ID
```

Avançado ▾

E na saída do console podemos obter o que aconteceu com a nossa aplicação e construção da mesma.





## Saída do console

```
Iniciado pelo usuário Gabriel Oliveira dos Santos
Rodando como SISTEMA
Construindo no espaço de trabalho
/home/gabriel/.jenkins/workspace/Trabalho de Build 1
[Trabalho de Build 1] $ /bin/sh -xe /tmp/jenkins7923382960977970452.sh
+ echo Passo 1 Executando
Passo 1 Executando
+ echo Número do build: 3
Número do build: 3
[Trabalho de Build 1] $ /bin/sh -xe /tmp/jenkins883875746458044011.sh
+ echo ID do build: 3
ID do build: 3
Finished: SUCCESS
```

## Workspace

Temos um shell que obtém o diretório do nosso workspace e armazena-lo em um arquivo, que fica no nosso workspace, como são descritos nas imagens abaixo.

Executar shell ?

Command

Veja [a lista de variáveis de ambiente disponíveis](#)

```
echo "ID do build: "$WORKSPACE
cat >$WORKSPACE/arquivo1<<EOL
Arquivo de Log
EOL
```





Avançado ▾

```
Iniciado pelo usuário Gabriel Oliveira dos Santos
Rodando como SISTEMA
Construindo no espaço de trabalho
/home/gabriel/.jenkins/workspace/Trabalho de Build 1
[Trabalho de Build 1] $ /bin/sh -xe /tmp/jenkins8503481052522272915.sh
+ echo Passo 1 Executando
Passo 1 Executando
+ echo Número do build: 7
Número do build: 7
[Trabalho de Build 1] $ /bin/sh -xe /tmp/jenkins9309111505805402036.sh
+ echo ID do build: /home/gabriel/.jenkins/workspace/Trabalho de Build
1
ID do build: /home/gabriel/.jenkins/workspace/Trabalho de Build 1
+ cat
Finished: SUCCESS
```

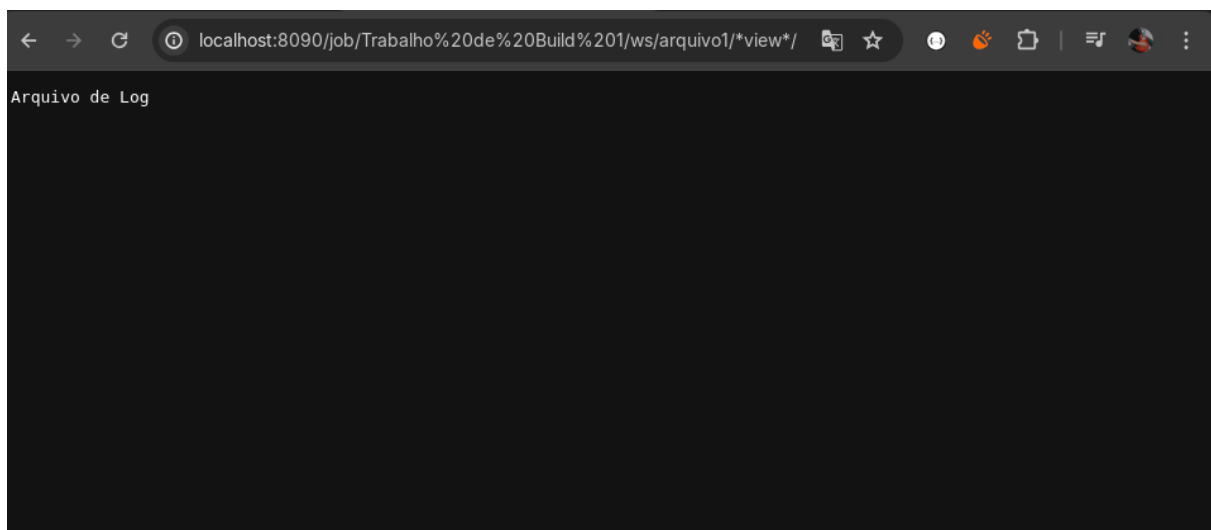
## Workspace Trabalho de Build 1 em mestre

Trabalho de Build 1 /



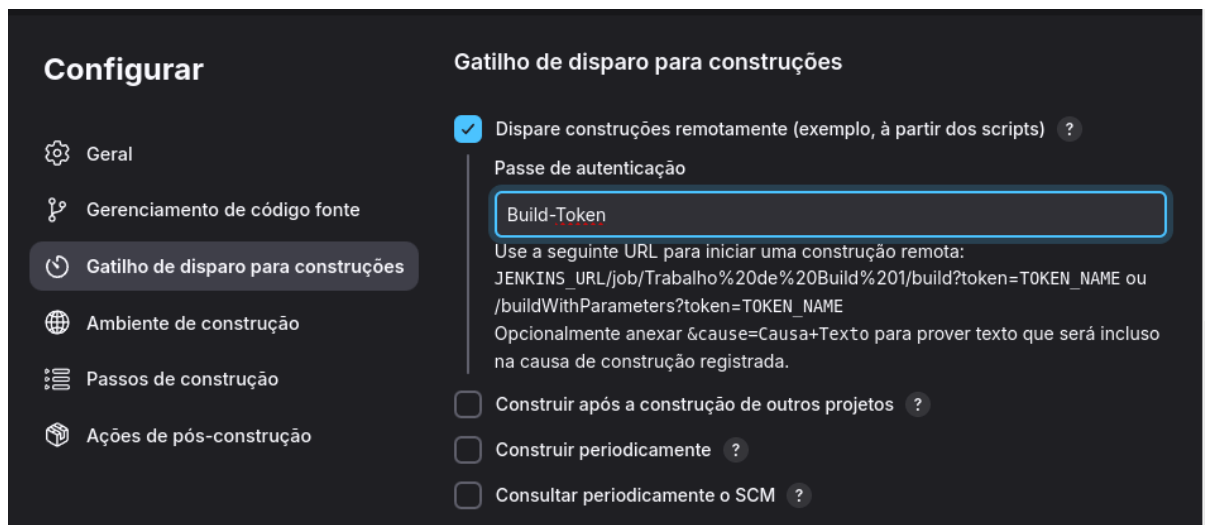
 <b>arquivo1</b>	9 de mai. de 2024 14:56:26	15 B	
 <b>EOL</b>	9 de mai. de 2024 14:55:38	0 B	

 (Todos os arquivos em .zip)



## Gatilhos de disparo

É possível fazer os builds remotamente, através de gatilhos de disparo. em que é possível criar um "token" que ao acessar-lo, faz com que o build seja feito sem ter que clicarmos em um botão, como os passos abaixos que mostram como fazer.



Como está escrito na gatilho, temos que acessar a URL com o nome do nosso token, que no nosso caso fica assim:

```
localhost:8090/job/Trabalho%20de%20Build%201/build?token=Build
```

Gerando um build remoto, que está descrito na imagem abaixo:



## Dependências de trabalhos

As dependências de trabalhos, servem como passos, um build só é executado se caso outro seja executado com sucesso, ou seja, uma forma sequencial de trabalhos a serem executados.

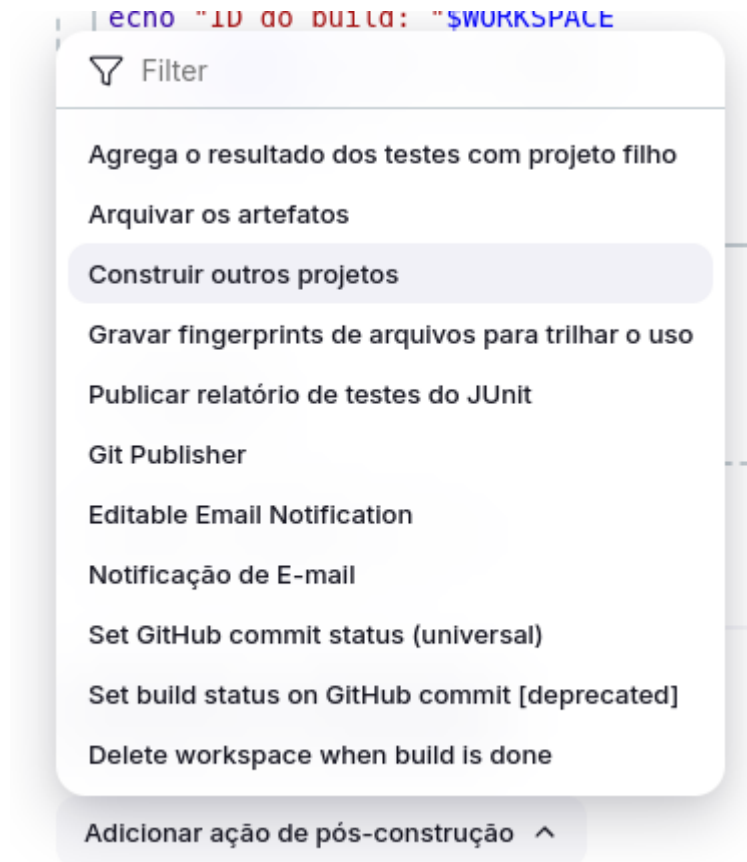
S	W	Nome ↓	Último sucesso	Última falha	Última duração	
		Trabalho de Build 1	4 min 42 seg #8	17 min #6	43 ms	
		Trabalho de Build 2	N/D	N/D	N/D	

Na imagem acima foram feitas 2 tarefas, cada uma serve como build de uma aplicação nossa teste. Executamos nosso build 2, com um simples:

```
echo "Trabalho de Build 2"
```



```
Iniciado pelo usuário Gabriel Oliveira dos Santos
Rodando como SISTEMA
Construindo no espaço de trabalho
/home/gabriel/.jenkins/workspace/Trabalho de Build 2
[Trabalho de Build 2] $ /bin/sh -xe
/tmp/jenkins16049187021075661994.sh
+ echo Trabalho de Build 2
Trabalho de Build 2
Finished: SUCCESS
```


Vamos no nosso primeiro build (build 1), que faremos Ações de pós-construção, e nessa ação iremos executar o build 2, assim encadeando nossa execução de builds, assim sendo sequenciais.



Assim iremos executar nosso Build 2 se caso o Build 1 seja construído estavelmente, que significa que só será executado o build 2, se caso o 1 seja executado sem erros, por isso escolhemos essa opção.

## Ações de pós-construção

 **Construir outros projetos** 




Projetos para construir

Trabalho de Build 2

☒ Disparar apenas se a construção estiver estável

☐ Disparar mesmo se a construção estiver instável

☐ Disparar mesmo se a construção falhar

Adicionar ação de pós-construção 

Voltamos para o painel central e executamos o Build 1, assim o numero de tentativas de construção aumenta no Build 1 e no Build 2.

S	W	Nome ↓	Último sucesso	Última falha	Última duração	
		Trabalho de Build 1	11 min #8	24 min #6	43 ms	
		Trabalho de Build 2	6 min 18 seg #1	N/D	52 ms	

Agora executando o build 1 novamente:

S	W	Nome ↓	Último sucesso	Última falha	Última duração	
		Trabalho de Build 1	14 seg #9	26 min #6	29 ms	
		Trabalho de Build 2	7,3 seg #2	N/D	29 ms	

Assim o último trabalho executado com sucesso no Build 1 , fará com que o trabalho que executa o Build 2 seja executado com sucesso. E se você for no Build 2, aparece para a gente o projeto pai do nosso Build.

## Trabalho de Build 2

Trabalho de Build simples 2

 editar descrição

Desabilitar projeto

### Projetos pai

 Trabalho de Build 1

## Build com Github

Agora vamos fazer um build de um repositório de forma remota, com acesso ao Github.

Vamos utilizar um exemplo de repositório do nosso professor Moises do IT Talent, com o link do repositório abaixo.


```
https://github.com/moisesAlc/ReactBasic.git
```




Ao criar um nova tarefa, vamos em configurações e Gerenciamento de Código Fonte selecionar git e adicionar a URL

 Git 

Repositories 

Repository URL 


https://github.com/moisesAlc/ReactBasic.git


Credentials 

- none -

+ Add ▾

Avançado ▾

Branches to build 

Branch Specifier (blank for 'any') 

\*/main

Add Branch

Navegar no repositório 

(Auto)

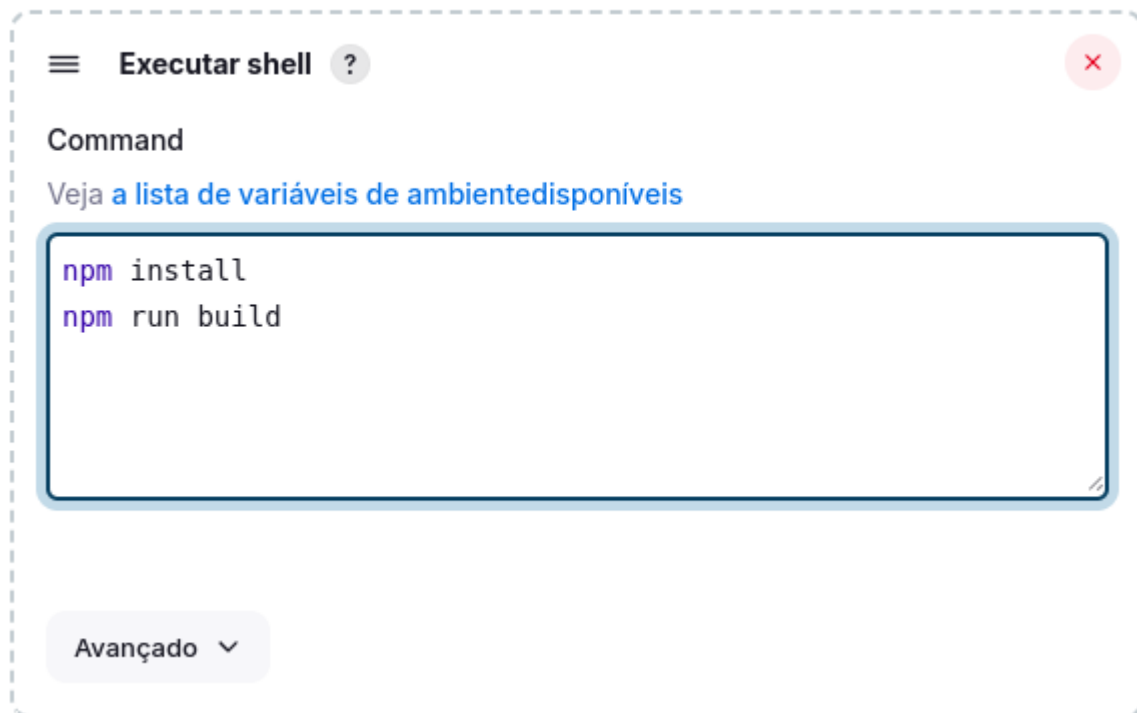
Additional Behaviours

Adicionar ▾

Para rodar a aplicação é necessário ter o npm na maquina virtual, entao no shell iremos digitar

```
npm install  
npm run build
```

## Passos de construção



Ao clicar no progresso podemos ver o que está rodando na maquina virtual, e depois ver o que aconteceu no console.



## Saída do console

```
Iniciado pelo usuário Gabriel Oliveira dos Santos
Rodando como SISTEMA
Construindo no espaço de trabalho
/home/gabriel/.jenkins/workspace/Build Github Teste
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir
/home/gabriel/.jenkins/workspace/Build Github Teste/.git #
timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url
https://github.com/moisesAlc/ReactBasic.git # timeout=10
Fetching upstream changes from
https://github.com/moisesAlc/ReactBasic.git
> git --version # timeout=10
> git --version # 'git version 2.34.1'
> git fetch --tags --force --progress --
https://github.com/moisesAlc/ReactBasic.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision e146903011cf8ec77ed5691f86f005d61c4d1cae
(refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f e146903011cf8ec77ed5691f86f005d61c4d1cae #
timeout=10
Commit message: "updated print img from readme"
> git rev-list --no-walk e146903011cf8ec77ed5691f86f005d61c4d1cae #
timeout=10
[Build Github Teste] $ /bin/sh -xe
/tmp/jenkins4412334152575963968.sh
+ npm install
```

Estado pessoal

</> Alterações

Saída do console

Adicionar descrição

Editar informações de compilação

Apagar a construção {0}

Git Build Data

Construção anterior

#2 (9 de mai. de 2024 15:52:02)

Deixar essa construção como permanente

Iniciado 1 min 13 seg atrás

Levou 13 seg

No changes.

Iniciado pelo(a) usuário(a) [Gabriel Oliveira dos Santos](#)

Revision: e146903011cf8ec77ed5691f86f005d61c4d1cae

Repository: <https://github.com/moisesAlc/ReactBasic.git>

refs/remotes/origin/main

Se voltarmos no workspace, podemos ver o que foi instalado no build.

## Workspace Build Github Teste em mestre

Build Github Teste /

→

.git

.github

build

node\_modules

public

src

.gitignore

9 de mai. de 2024 15:50:52

310 B

👁

package.json

9 de mai. de 2024 15:50:52

812 B

👁

package-lock.json

9 de mai. de 2024 15:52:06

701,03 KiB

👁

README.md

9 de mai. de 2024 15:50:52

3,30 KiB

👁

📄

(Todos os arquivos em .zip)

Podemos instalar tudo como zip, descompactar e rodar um http-server, se caso nao tenha rode o comando abaixo, assim voce pode usar o ip e a porta para acessar a aplicação na web.

```
npm install -g http-server
```

colocar o ip e a porta e rodar app na web

```
Connection Timeout: 120 seconds
Directory Listings: visible
AutoIndex: visible
Serve GZIP Files: false
Serve Brotli Files: false
Default File Extension: none

Available on:
  http://127.0.0.1:8080
  http://172.21.67.7:8080
Hit CTRL-C to stop the server
```



∞ Módulo de Jenkins do IT TALENT! 🚀