

Deploy do Backend no Heroku

O que é Deploy?

O verbo **deploy**, em inglês, significa **implantar**.

Em programação, seu sentido está intimamente relacionado à sua tradução literal: fazer um deploy, em termos práticos, significa colocar no ar alguma aplicação que teve seu desenvolvimento concluído.

Quando um site é finalizado por um desenvolvedor e, após seus testes, é finalmente hospedado na nuvem e colocado no ar, ele passa pelo processo de deploy.

De mesmo modo, quando um sistema sofre alguma melhoria ou alteração em seu código-fonte, implementar essa alteração ao sistema que está no ar também é um tipo de deploy.

O que veremos por aqui?

Esse documento é um passo a passo para você enviar a sua aplicação SPRING, gratuitamente para a nuvem (Deploy). Este processo irá gerar um link de acesso a sua aplicação, que poderá ser acessado de qualquer lugar, a partir de qualquer dispositivo com acesso a Internet.

Para efetuar o Deploy vamos precisar fazer algumas modificações em nosso projeto, que serão detalhadas nas próximas páginas.

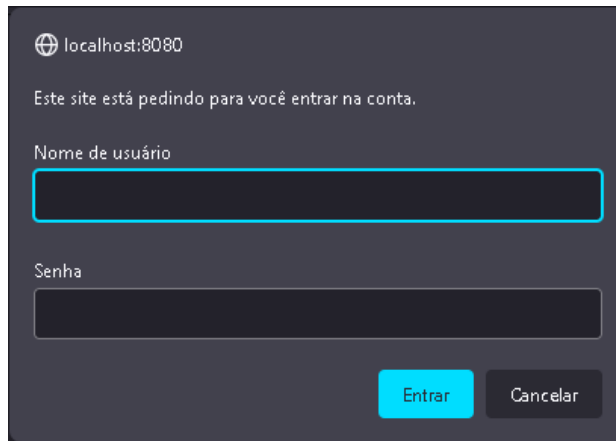


Passo 01 - Criar a Documentação da API

Para criar a Documentação da API no Swagger, utilize o **Guia de Configuração do SPringdoc**.

Passo 02 - Testar a API no seu computador

1. Execute a sua aplicação localmente pelo STS
2. Abra o endereço: <http://localhost:8080/> no seu navegador
3. A sua aplicação deverá exibir a tela de **Login (Usuário e Senha)**. Utilize o teste o **usuário: root** e a **Senha: root**, que foram criados em memória na **Classe BasicSecurityConfig**, na Camada Security.



4. Caso a aplicação **não** solicite o **Usuário** e a **Senha**, feche todas as janelas abertas do seu Navegador da Internet, abra novamente e acesse o endereço acima. Se o problema persistir, verifique a sua configuração do Swagger.
5. Verifique se após o login, o **Swagger** está inicializando automaticamente.
6. Caso você não tenha testado no **Insomnia**, execute os testes e verifique se tudo está funcionando.
7. Em especial, verifique se o Método **logar** está devolvendo o **Token**.
8. Antes de continuar a configuração do projeto para efetuar o Deploy, não esqueça de **parar a execução do Projeto no STS**.



IMPORTANTE: Não altere a senha do usuário root. Os instrutores da sua turma utilizarão este usuário para abrir, testar e corrigir a sua aplicação.

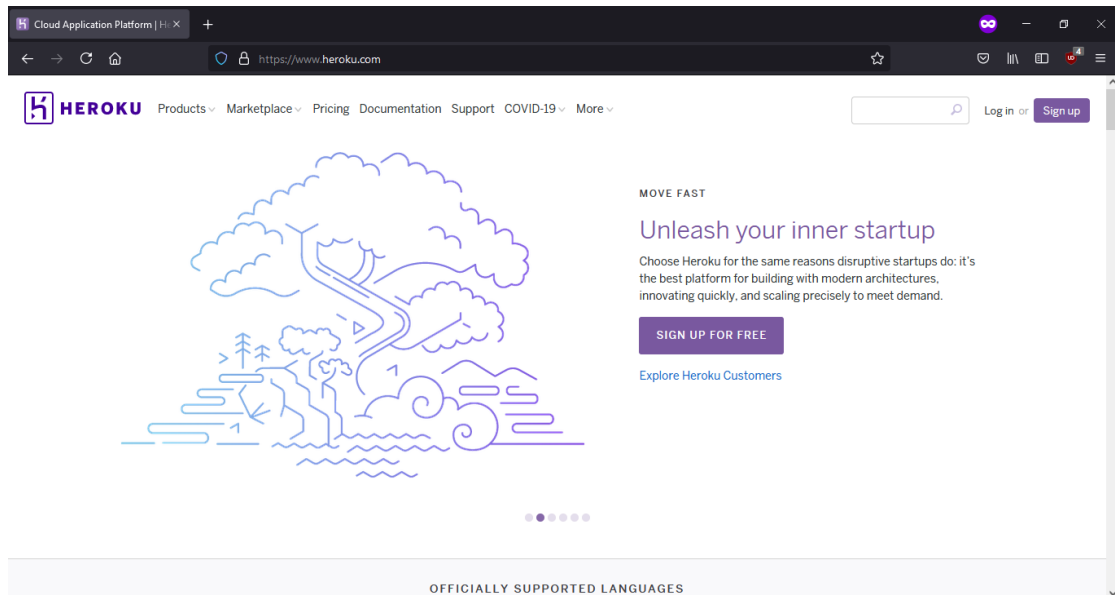


ATENÇÃO: Lembre-se que antes de fazer o Deploy é fundamental que a API esteja rodando e sem erros. Não faça os testes via Swagger porque o usuário root (em memória, não utiliza todos os recursos da Spring Security, em especial o Token).




Passo 03 - Criar uma conta grátis no Heroku

1. Acesse o endereço: <https://www.heroku.com>




2. Crie a sua conta grátis no Heroku clicando no botão **SIGN UP FOR FREE**.

3. Preencha os dados do formulário e clique no botão **CREATE FREE ACCOUNT**.




Free account

Create apps, connect databases and add-on services, and collaborate on your apps, for free.



Your app platform

A platform for apps, with app management & instant scaling, for development and production.



Deploy now

Go from code to running app in minutes. Deploy, scale, and deliver your app to the world.

First name *

Last name *

Email address *

Company name

Role *

Role


Country *

Brazil

Primary development language *

Select a language

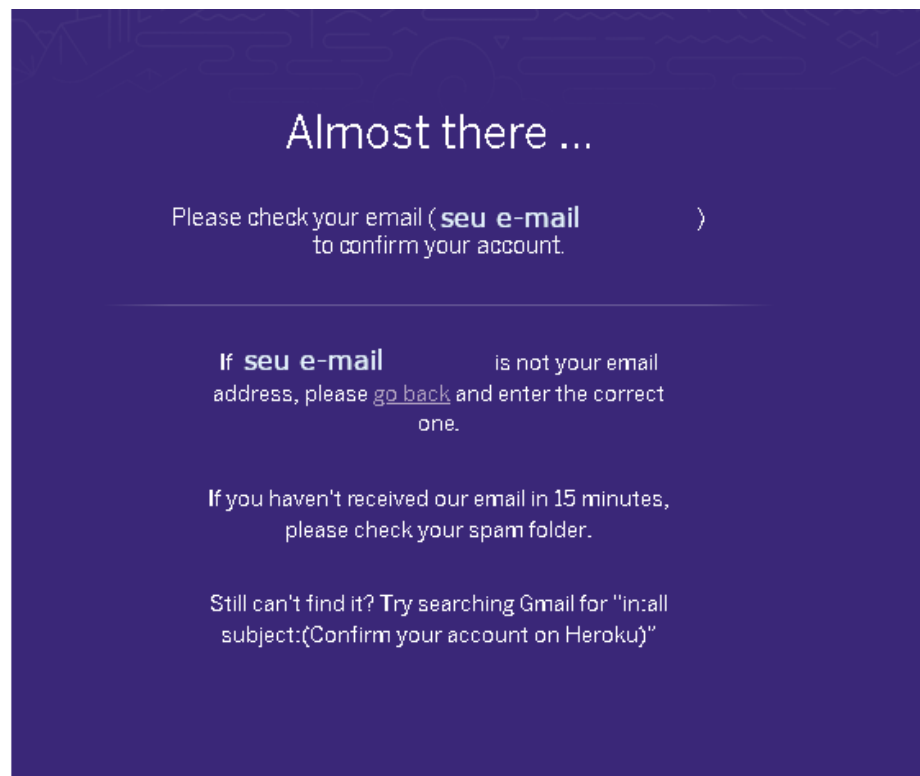
☐ I'm not a robot


reCAPTCHA
Privacy - Terms

CREATE FREE ACCOUNT

Signing up signifies that you have read and agree to the [Terms of Service](#) and our [Privacy Policy](#).
[Cookie Preferences](#).

4. Será exibida a mensagem abaixo informando que você receberá uma mensagem no seu e-mail para ativar a sua conta no Heroku. Acesse o seu e-mail e ative a sua conta.



5. O e-mail que você receberá será semelhante a imagem abaixo. Clique no link indicado em vermelho para ativar a sua nova conta



Thanks for signing up with Heroku! You must follow this link within 30 days of registration to activate your account:

<https://id.heroku.com/account/accept/10514805/556bc551c22058e09cc4986c24caaaf6>

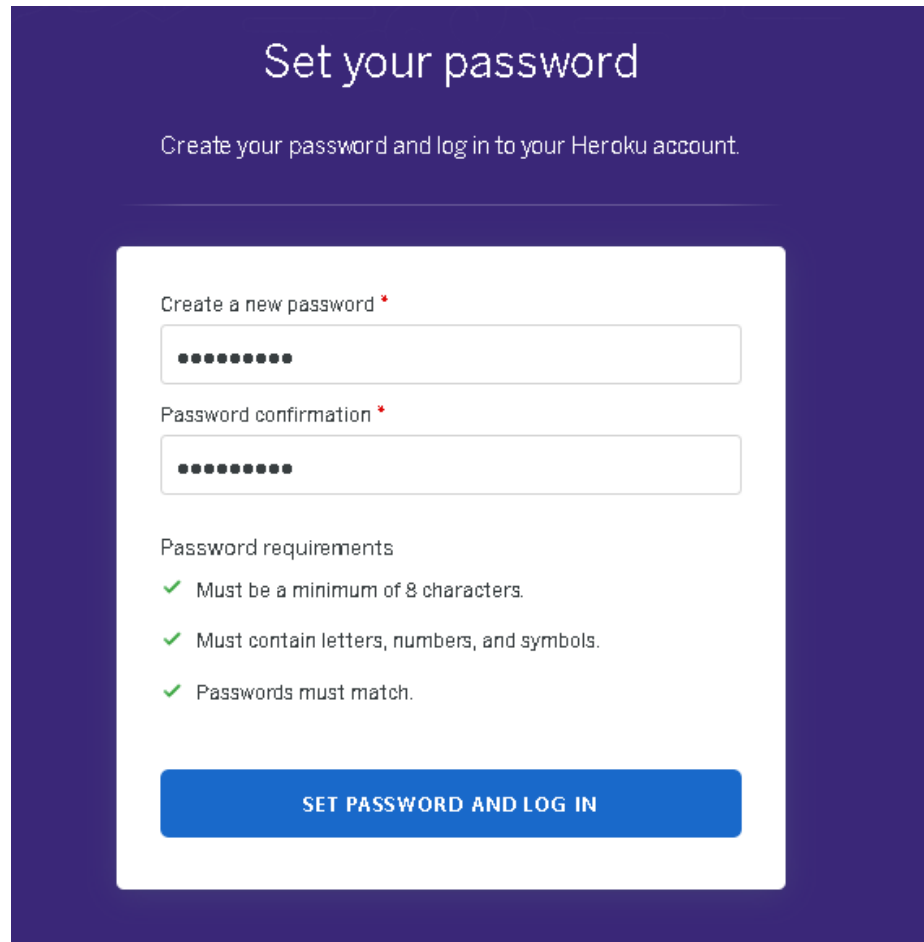
Have fun, and don't hesitate to contact us with your feedback.

The Heroku Team
<https://heroku.com>

Heroku is the cloud platform for rapid deployment and scaling of web applications. Get up and running in minutes, then deploy instantly via Git.

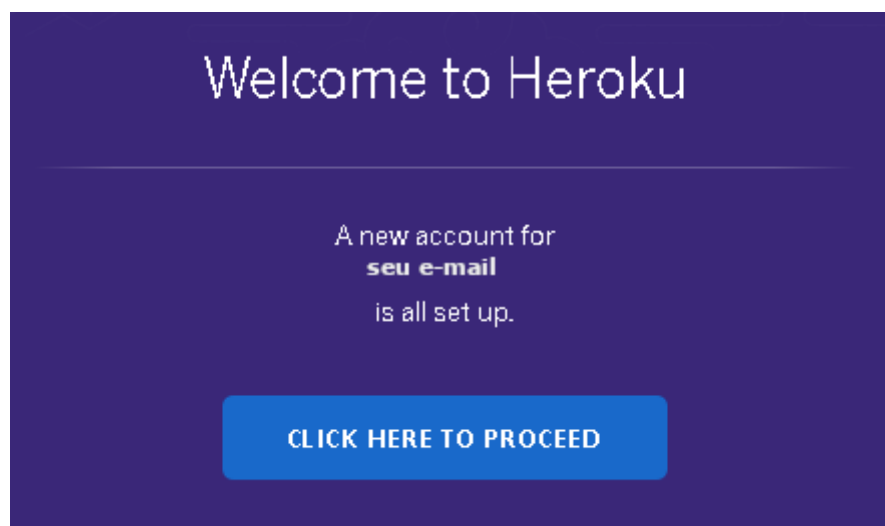
To learn more about Heroku and all its features, check out the Dev Center:
<https://devcenter.heroku.com/articles/quickstart>

6. Será aberta a janela abaixo para criar a senha da sua conta. Crie uma senha e clique no botão **SET PASSWORD AND LOGIN**.



ATENÇÃO: A senha deve ter no mínimo 8 caracteres e pelo menos 1 letra maiúscula, 1 caracter especial e 1 numero.

7. Será exibida a tela de Boas Vindas. Clique no botão **CLICK HERE TO PROCEED**.



8. Na próxima tela, concorde com os termos de uso da plataforma clicando no botão **Accept**.

Terms of Service

All Customers

Effective October 1, 2020, you agree that your use of the Heroku Services is governed by the [Salesforce Master Subscription Agreement](#), unless (except for free customers of the Heroku Services) you have a written master subscription agreement executed by salesforce.com for such Heroku Services as referenced in the Documentation, in which case such written salesforce.com master subscription agreement will govern. You further agree that your use or purchase of Heroku Add-ons, Buildpacks or Buttons that are not Heroku Services ("Heroku Elements") are governed by the [Heroku Elements Terms of Use \(Default\)](#), unless the Heroku Elements Marketplace provider has furnished to Heroku a separate terms of use, in which case such provider's terms of use govern the use or purchase of the applicable Heroku Elements. [Additional Terms](#) apply to credit card customers of the Heroku Services.

Heroku Marketplace Providers

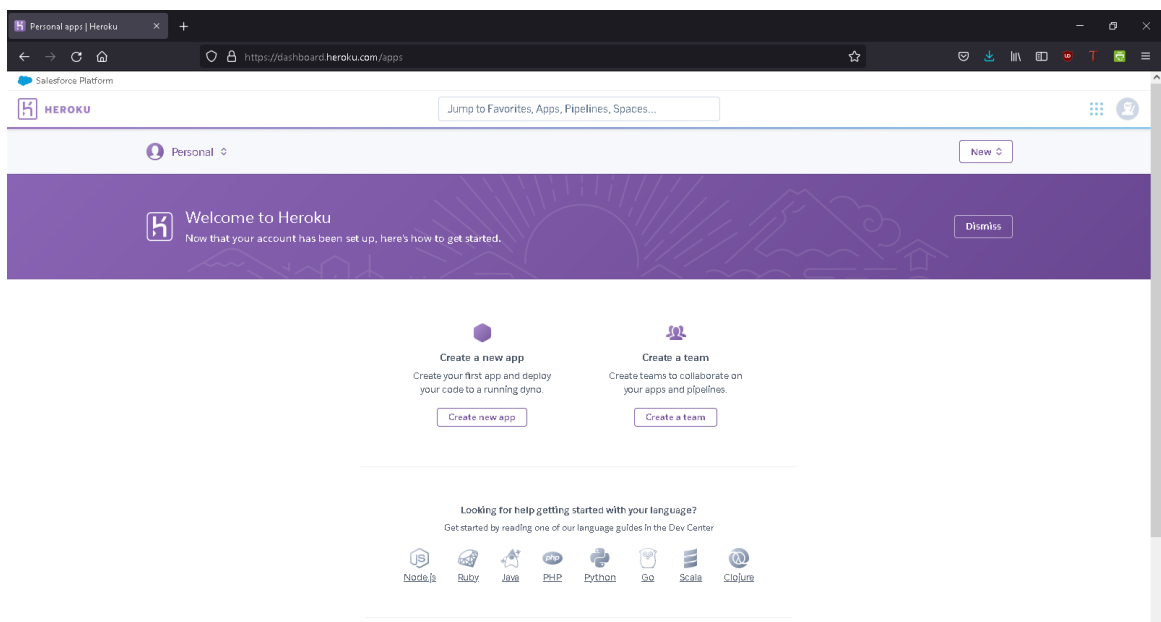
If you are or become a Heroku Elements Marketplace Provider, you further agree that your participation in the Heroku Elements Marketplace is governed by the [Salesforce License and Distribution Agreement for the Heroku Elements Marketplace](#).

Italian Customers

Are you domiciled in Italy?
☐ No.

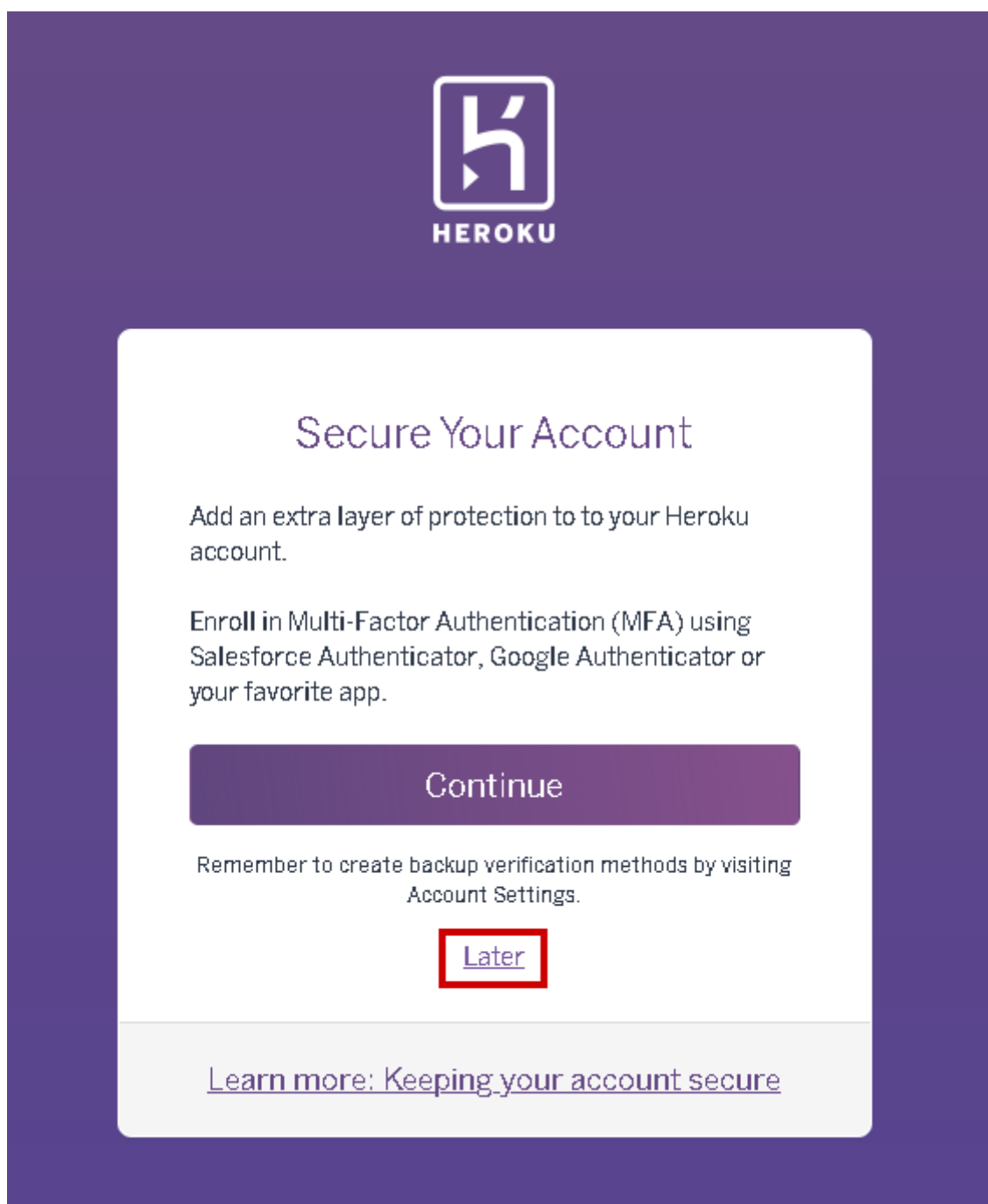
Accept

9. Você será redirecionado para o **Dashboard do Heroku**. Agora você está pronto para criar as suas aplicações na Nuvem do Heroku.



ATENÇÃO: Conclua todas etapas do processo de criação da conta no Heroku antes de avançar para o próximo passo do Deploy.

10. Caso o Heroku exiba a mensagem abaixo, solicitando a ativação do **MFA (Multi-Factor Authentication)**, não habilite esta opção. Clique no link **Later**, como mostra a figura abaixo, no item marcado em vermelho.



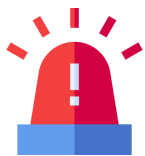
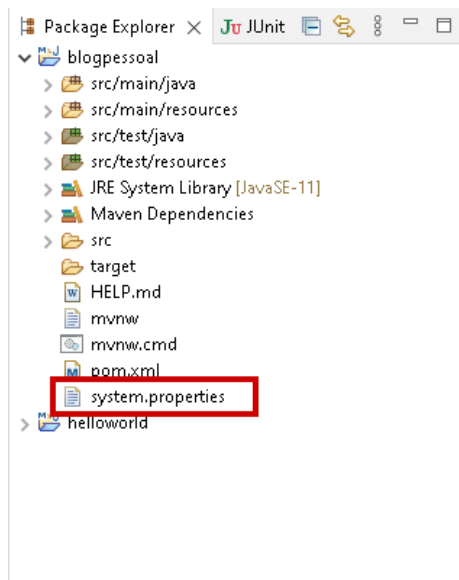
ATENÇÃO: Não habilite em sua conta no Heroku a opção MFA (Multi-Factor Authentication), ou seja, o login em 2 etapas. Em alguns servidores não é possível efetuar login via Heroku Client com o MFA habilitado.



Passo 04 - Criar o arquivo system.properties

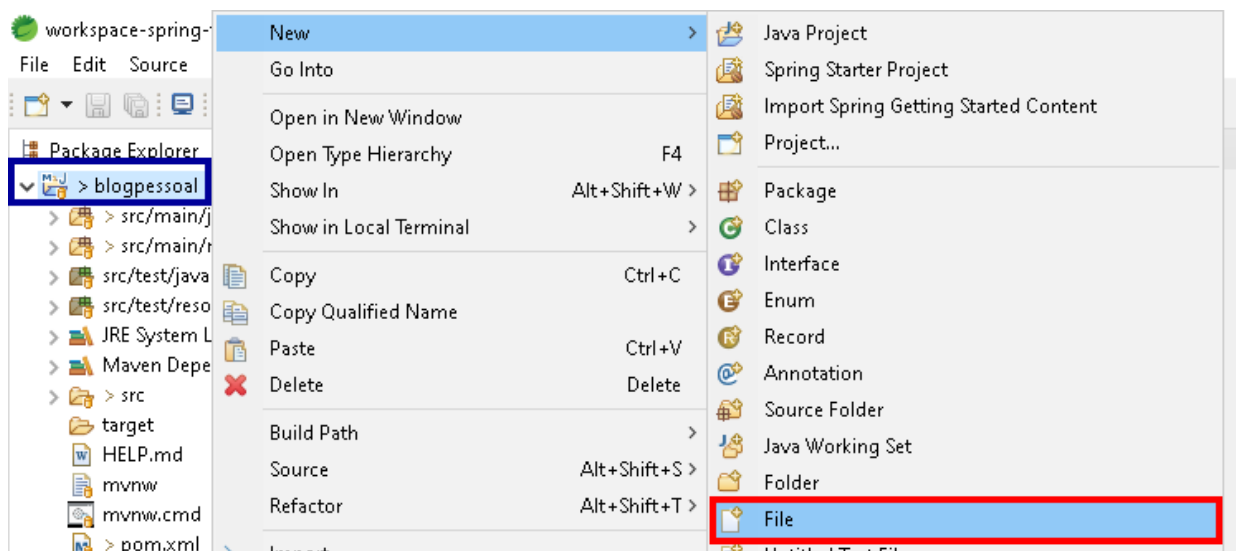
O arquivo **system.properties**, tem o objetivo de informar ao Heroku qual a versão a do Java ele deve utilizar para gerar o seu projeto e criar o arquivo executável (.jar).

1. Na **raiz do seu projeto**, na pasta **blogpessoal** (como mostra a figura abaixo), crie o arquivo **system.properties**.

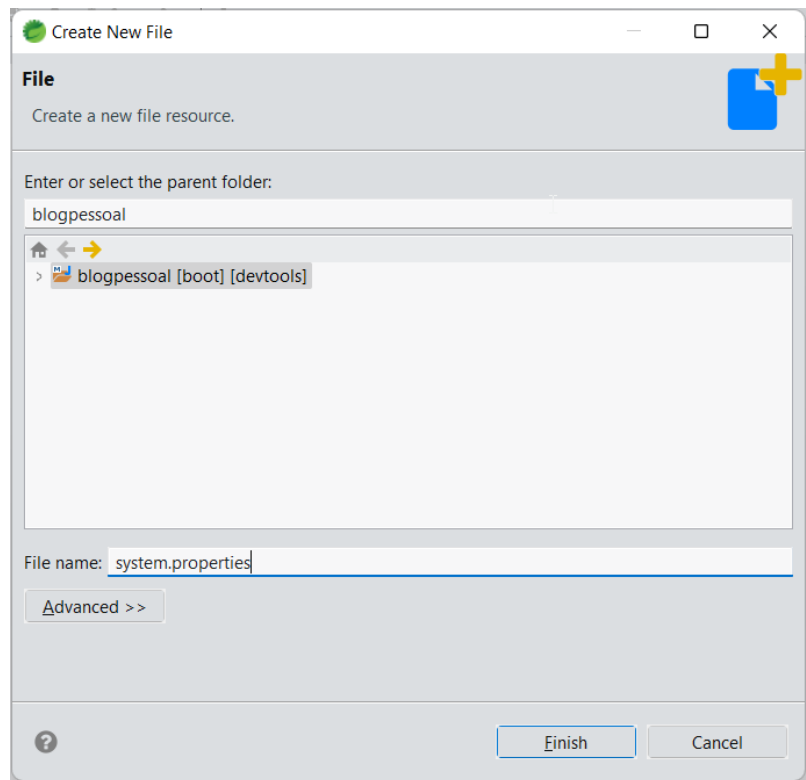


ALERTA DE BSM: Mantenha a atenção aos detalhes ao criar o arquivo system.properties. Um erro muito comum é não criar o arquivo na pasta raiz do projeto. Outro erro comum é digitar o nome do arquivo errado.

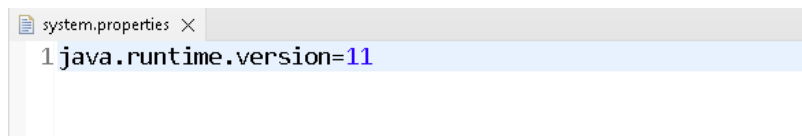
2. Na Guia **Package explorer**, clique com o botão direito do mouse sobre a pasta do projeto (indicada em azul) e clique na opção **New** → **File**.



3. Em **File name**, digite: **system.properties** e clique no botão **Finish**.



4. No arquivo **system.properties** indique a versão do Java que será utilizada pelo Heroku através da linha abaixo:



```
java.runtime.version=11
```



ATENÇÃO: A versão do Java informada no arquivo `system.properties` deve ser a mesma informada no arquivo `pom.xml`.



DICA: Durante o Deploy, caso o Heroku não reconheça a versão correta do Java (Exemplo: informei a versão 11 e o Heroku reconheceu a versão 1.8), apague o arquivo `system.properties`, recrie o arquivo na raiz do projeto e tente fazer o Deploy novamente.

Passo 05 - Adicionar a Dependência do PostgreSQL no pom.xml

O Heroku, na sua versão gratuita, utiliza o **PostgreSQL** como **SGBD** (Sistema Gerenciador de Banco de dados).

No Bloco 02 estamos utilizando o **MySQL** para desenvolver o Blog Pessoal. Ambos são Banco de dados Relacionais e graças ao **Spring Data JPA**, não será necessário realizar nenhuma alteração no código da nossa aplicação. A única mudança necessária, além de adicionar a **Dependência do PostgreSQL no pom.xml**, será necessário configurar a conexão com o Banco de dados PostgreSQL na nuvem.



[Site Oficial: PostgreSQL](https://www.postgresql.org/)

No arquivo, **pom.xml**, vamos adicionar as linhas abaixo, com a dependência do **PostgreSQL**:

```
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
</dependency>
```

Passo 06 - Configurar o Banco de Dados na Nuvem

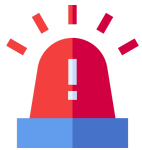
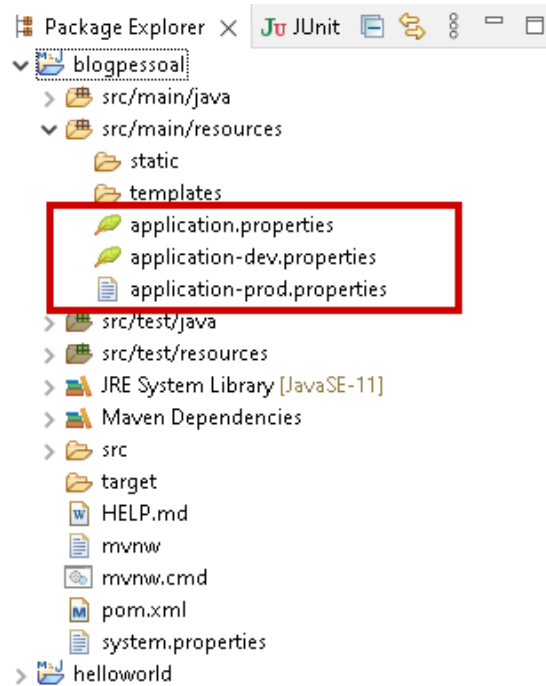
A Configuração do Banco de dados Local é diferente da configuração que será utilizada no Heroku.

No passo anterior, adicionamos a Dependência do PostgreSQL no arquivo pom.xml, neste passo vamos configurar a aplicação para acessar o Banco de dados remoto no Heroku.

Para simplificar o processo, vamos utilizar um recurso do Spring chamado **Profiles** (perfis), que nada mais é do que criar um modelo de configuração para cada situação, ou seja, uma configuração para usar localmente (**application-dev.properties**) e outra para usar na nuvem (**application-prod.properties**).

O grande benefício dos Profiles é simplificar a troca entre a configuração Local (**MySQL**) e a configuração Remota do Heroku (**PostgreSQL**).

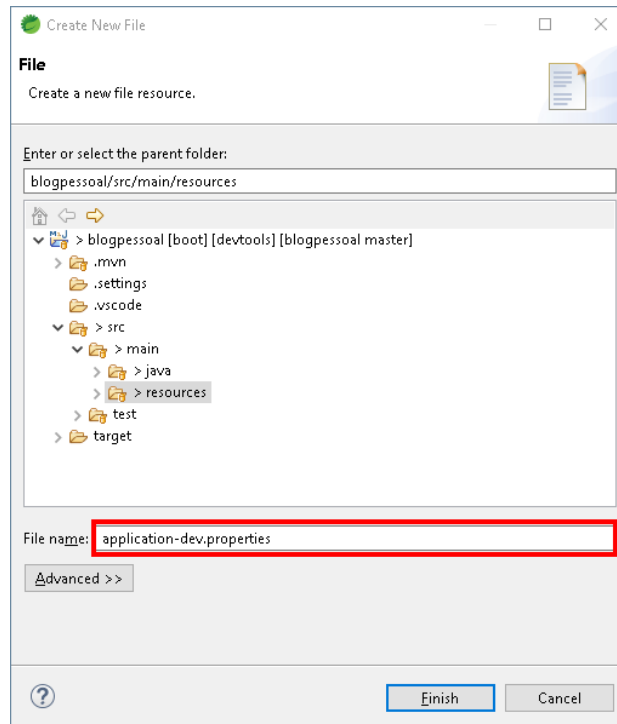
1. Na Source Folder **src/main/resources**, crie os arquivos **application-dev.properties** (Configuração do Banco de dados local) e **application-prod.properties** (Configuração do Banco de dados na nuvem).



ALERTA DE BSM: *Mantenha a atenção aos detalhes ao criar os arquivos **application-dev.properties** e **application-prod.properties**. Cuidado para não se equivocar ao nomear os arquivos ou criar em um pacote diferente.*

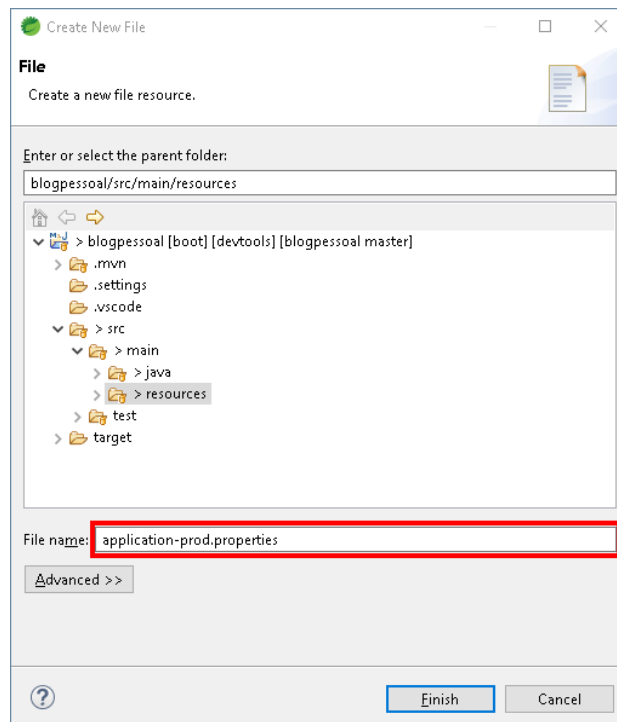
2. Vamos criar o primeiro arquivo. No lado esquerdo superior, na Guia **Package explorer**, na Source Folder **src/main/resources**, clique com o botão direito do mouse e clique na opção **New** → **File**.

3. Em **File name**, digite o nome do primeiro arquivo (**application-dev.properties**) e clique no botão **Finish**.



4. Vamos criar o segundo arquivo da mesma forma que criamos o primeiro.

5. Em **File name**, digite o nome do segundo arquivo (**application-prod.properties**) e clique no botão **Finish**.



Agora vamos configurar os 3 arquivos:

6.1 Configuração do arquivo application.properties

1. Abra o arquivo **application.properties**, apague todo o conteúdo do arquivo e substitua pelas linhas abaixo e salve o arquivo. O arquivo application.properties ficará com o seguinte conteúdo:

```
spring.profiles.active=prod

springdoc.api-docs.path=/v3/api-docs
springdoc.swagger-ui.path=/swagger-ui.html
springdoc.swagger-ui.operationsSorter=method
springdoc.swagger-ui.disable-swagger-default-url=true
springdoc.swagger-ui.use-root-path=true
springdoc.packagesToScan=com.generation.blogpessoal.controller
```



ALERTA DE BSM: *Mantenha a atenção aos detalhes ao configurar o arquivo application.properties. Cuidado para não apagar a configuração do Swagger (SpringDoc).*

6.2 Configuração do arquivo application-dev.properties

1. Abra o arquivo **application-dev.properties**, insira as linhas abaixo (Configuração original do **application.properties**) e salve o arquivo. **Não esqueça de alterar a senha do usuário root caso a sua senha do MySQL não seja root.**

```
1 spring.jpa.hibernate.ddl-auto=update
2 spring.jpa.database=mysql
3 spring.datasource.url=jdbc:mysql://localhost/db_blogpessoal?
  createDatabaseIfNotExist=true&serverTimezone=America/Sao_Paulo&useSSL=false
4 spring.datasource.username=root
5 spring.datasource.password=root
6 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL8Dialect
7
8 spring.jpa.show-sql=true
9
10 spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
11 spring.jackson.time-zone=Brazil/East
```

6.3 Configuração do arquivo application-prod.properties

1. No arquivo, **application-prod.properties**, insira as linhas abaixo e salve o arquivo:

```
spring.jpa.generate-ddl=true
spring.datasource.url=${JDBC_DATASOURCE_URL}
spring.jpa.show-sql=true

spring.jackson.date-format=yyyy-MM-dd HH:mm:ss
spring.jackson.time-zone=Brazil/East
```



ATENÇÃO: Depois de finalizar as configurações dos 3 arquivos, recomendamos executar o comando **Update Project** para atualizar as configurações do projeto.

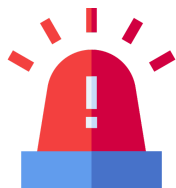
6.4 Alternando entre os perfis no arquivo application.properties

1. Para alternar entre as configurações Local e Remota, abra o arquivo **application.properties** e utilize uma das 2 opções abaixo:

spring.profiles.active=dev → O Spring executará a aplicação com a configuração do Banco de dados local (MySQL)

spring.profiles.active=prod → O Spring executará a aplicação com a configuração do Banco de dados na nuvem (Heroku)

Para o Deploy, devemos deixar a linha **spring.profiles.active** configurada com a opção **prod**.



ALERTA DE BSM: Mantenha a atenção aos detalhes ao criar os perfis do Banco de Dados. Um erro muito comum é tentar executar o seu projeto no STS com o Perfil prod habilitado no arquivo **application.properties**. Com o perfil prod habilitado, o projeto não será inicializado.



[Código fonte: Projeto finalizado](#)

Passo 07 - Atualizar o repositório do projeto no Github

1. Envie as atualizações do seu projeto para o repositório do Github, através do **Git Bash**, utilizando os comandos abaixo:

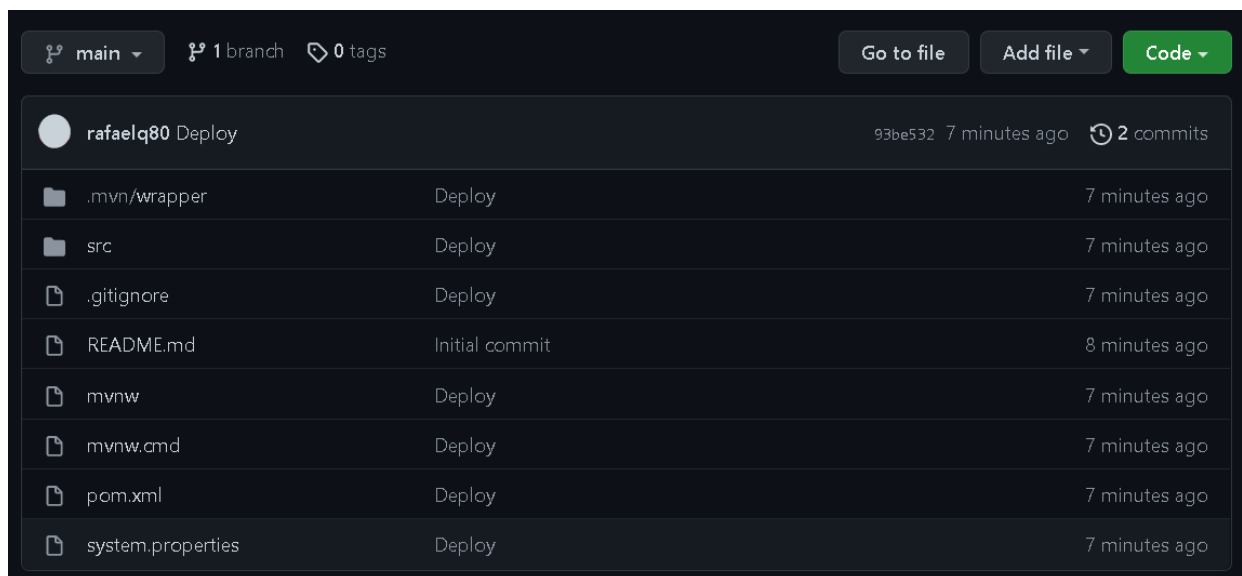
```
git add .
```

```
git commit -m "Deploy do Projeto Blog Pessoal"
```

```
git push origin main
```



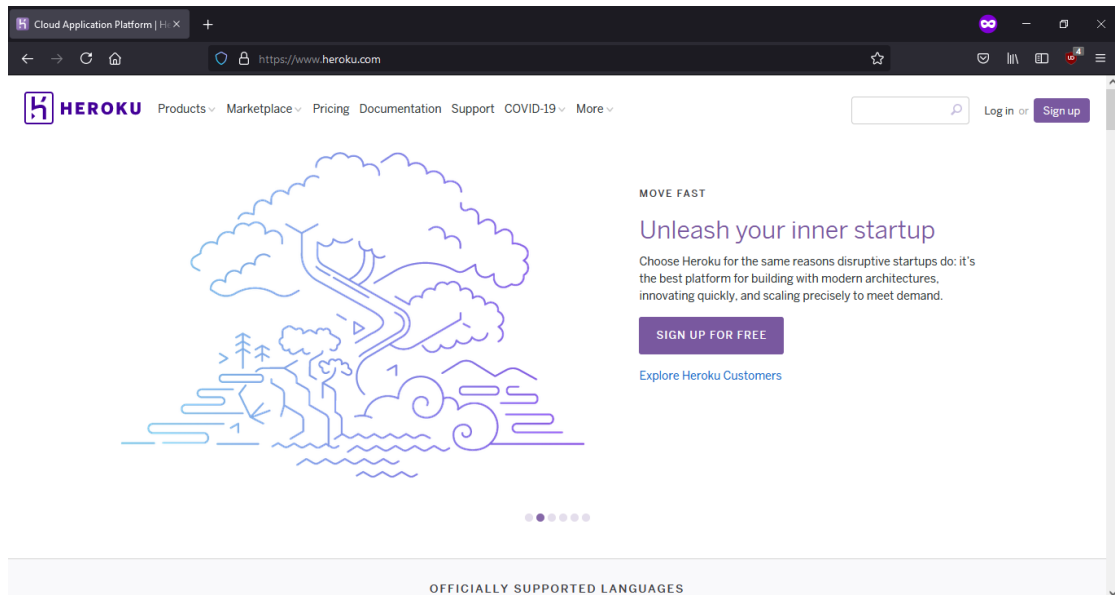
ATENÇÃO: Para efetuar o Deploy, o projeto Spring **OBRIGATORIAMENTE** precisa estar em um Repositório **EXCLUSIVO** e não pode estar **DENTRO DE UMA PASTA**, ou seja, ao abrir o repositório do projeto no Github, o conteúdo exibido será semelhante ao da imagem abaixo. Se estiver diferente da imagem abaixo será necessário refazer o Repositório do Github.



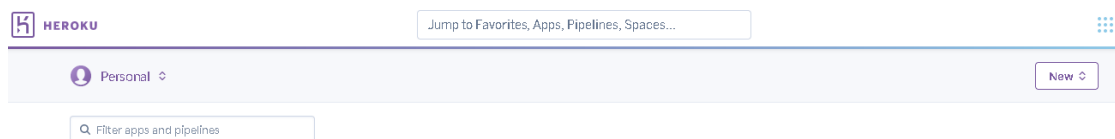


Passo 08 - Criar um novo projeto no Heroku

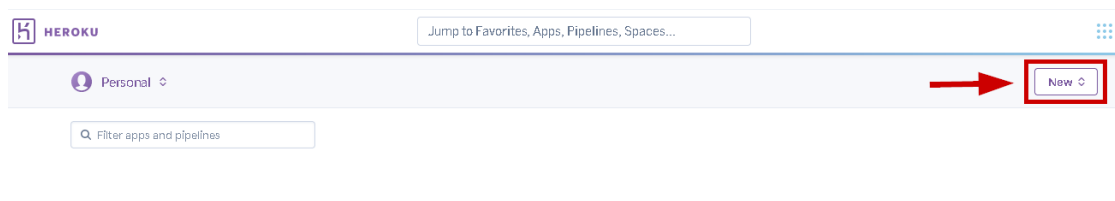
1. Acesse o endereço do Heroku: <https://www.heroku.com>



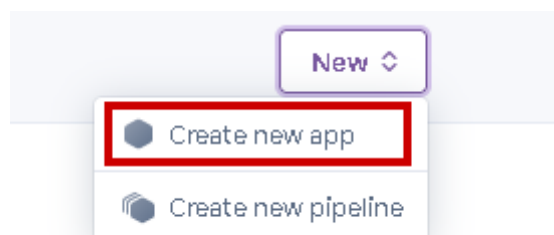
2. Faça o login no Heroku, caso a página Personal do Heroku não seja aberta automaticamente, como mostra a figura abaixo:



3. Clique no botão **New** para criar uma nova aplicação.



4. Clique na opção **Create new app**.



5. Dê um nome para a aplicação e clique no botão **Create app**.

App name

bpspring



bpspring is available



Choose a region



United States



Add to pipeline...

Create app

Caso o nome escolhido esteja disponível, será exibida a mensagem **is available** na cor verde.



ATENÇÃO: O NOME DO PROJETO NÃO PODE CONTER LETRAS MAIUSCULAS, NUMEROS OU CARACTERES ESPECIAIS. ALÉM DISSO ELE PRECISA SER ÚNICO DENTRO DA PLATAFORMA HEROKU.

Caso o nome escolhido não esteja disponível, será exibida a mensagem **is not available** na cor vermelha

App name

bp2022



bp2022 is not available



Choose a region



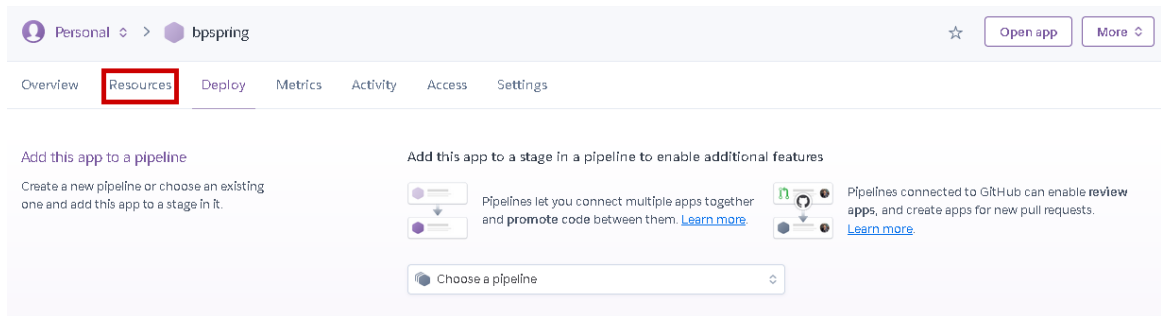
United States



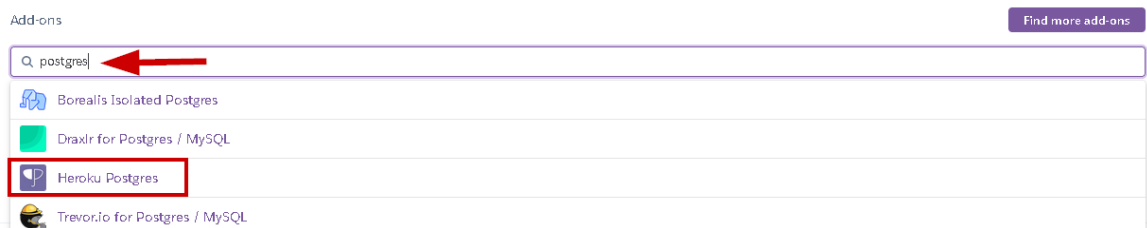


Passo 09 - Adicionar o Banco de dados no Heroku

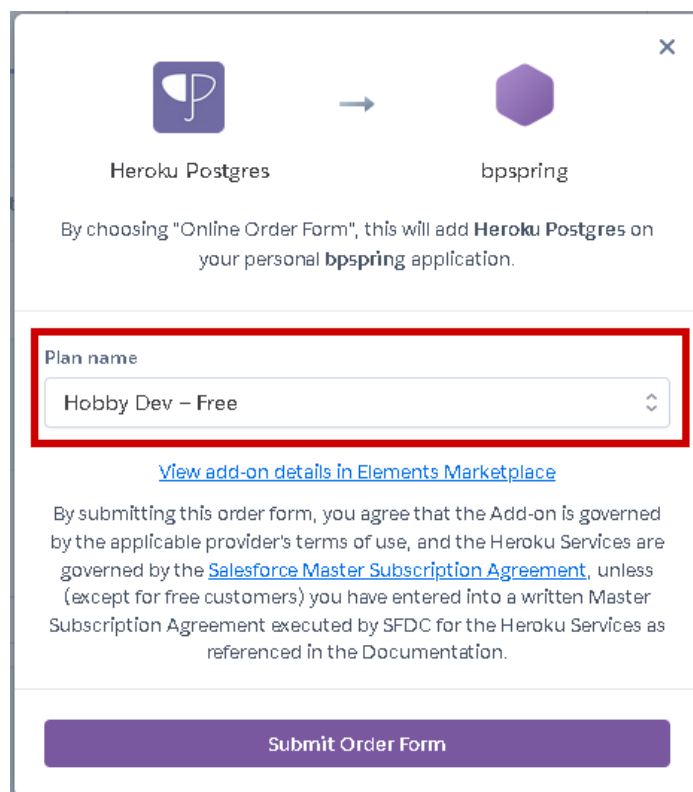
1. Para adicionar um **Banco de Dados PostgreSQL** no seu projeto, clique na guia **Resources** na próxima tela que será aberta após a criação da aplicação.



2. No item **Add-ons**, digite na caixa de pesquisa **“postgres”** e selecione a opção **Heroku Postgres**.



3. Na próxima tela, certifique-se de que o **Plan name** seja o **Hobby Dev - Free**. Clique no botão **Submit Order Form** para concluir.





ATENÇÃO: *Caso seja selecionado um plano diferente, o Heroku exigirá o Cartão de Crédito para emitir a fatura do serviço.

4. Observe na próxima tela que o custo do Plano é gratuito.

Add-ons Find more add-ons

The add-on `heroku-postgresql` has been installed. Check out the documentation in its [Dev Center](#) article to get started.

Quickly add add-ons from Elements

Heroku Postgres ↗	Attached as DATABASE ↕	Hobby Dev	Free ↕
Estimated Monthly Cost			\$0.00



ATENÇÃO: *O processo do Deploy enviará apenas a sua aplicação para a nuvem, logo o Banco de dados que será criado nesta etapa estará vazio.



Passo 10 - Configurar o fuso horário

1. Para configurar o **Fuso Horário** no seu projeto, clique na guia **Settings**.

Personal > bpspring ☆ Open app More

Overview Resources **Deploy** Metrics Activity Access **Settings**

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and promote code between them. [Learn more](#)

Pipelines connected to GitHub can enable review apps, and create apps for new pull requests. [Learn more](#)

Choose a pipeline

2. No item **Config vars**, clique no botão **Reveal Config Vars**.

Config Vars

Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.

Reveal Config Vars

3. Adicione a variável **TZ**, com o valor **America/Sao_Paulo** (como mostra a figura abaixo) e clique no botão **Add** para concluir.

Config Vars

Config vars change the way your app behaves. In addition to creating your own, some addons come with their own.

Config Vars

Hide Config Vars

DATABASE_URL

postgres://omqvamvxdypetx:cea9638368d11

✎ ✕

TZ

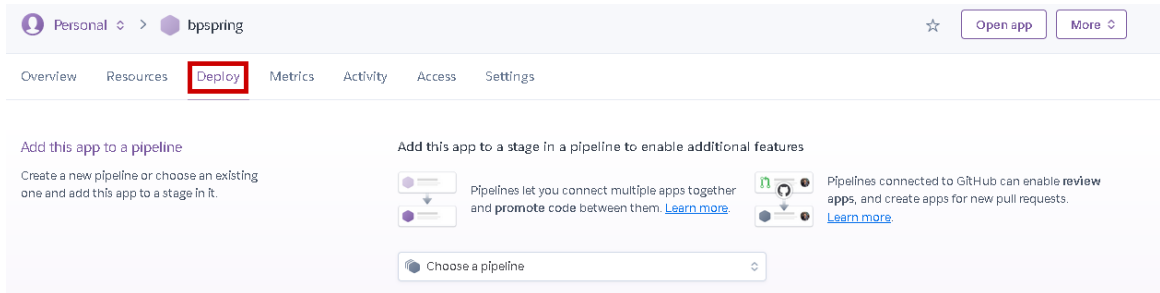
America/Sao_Paulo

✓

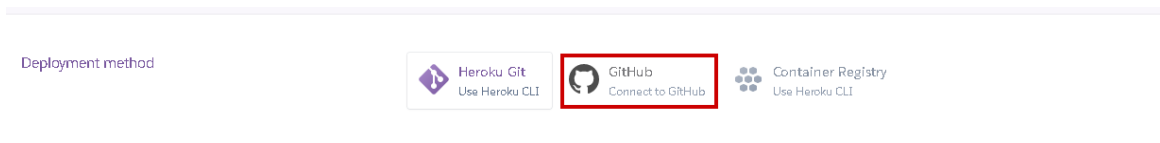
Add

Passo 11 - Deploy

1. Para configurar o **Deploy** do seu projeto, clique na guia **Deploy**.



2. Na sequência, clique no ícone do **Github**



3. No item **App connected to Github**, clique no botão **Connect to Github**

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

View your code diffs on GitHub

Connect your app to a GitHub repository to see commit diffs in the activity log.

Deploy changes with GitHub

Connecting to a repository will allow you to deploy a branch to your app.

Automatic deploys from GitHub

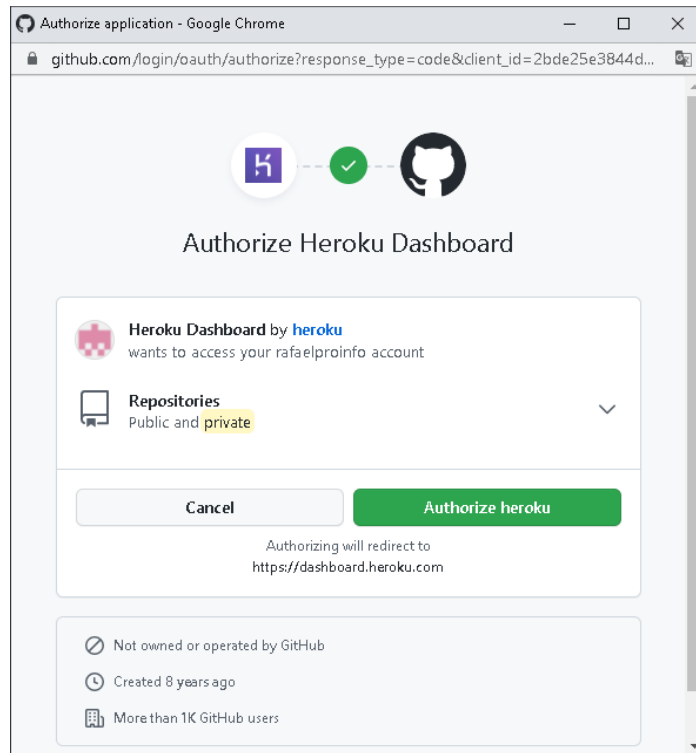
Select a branch to deploy automatically whenever it is pushed to.

Create review apps in pipelines

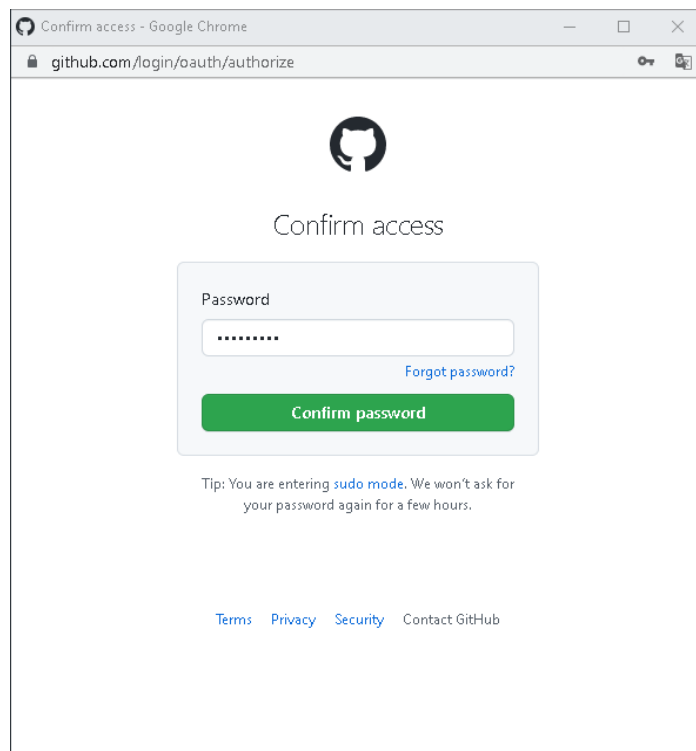
Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. [Learn more](#)

Connect to GitHub

4. Na próxima tela, clique no botão **Authorize Heroku**, para liberar o acesso do Heroku na sua conta do Github.



5. Faça o login na sua conta do Github, para concluir



6. Em seguida, vamos procurar o **Repositório do Projeto Blog Pessoal**
7. No item **Connect to Github**, na caixa de pesquisa **repo-name**, digite o **Nome do Repositório do Projeto Blog Pessoal** e clique no botão **Search**.

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

Search for a repository to connect to

rafaelq80

repo-name

Search

Missing a GitHub organization? [Ensure Heroku Dashboard has team access](#)

8. Será exibido o repositório. Clique no botão **Connect** ao lado do Repositório do Projeto Blog Pessoal.

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

Search for a repository to connect to

rafaelq80

blogpessoal

Search

Missing a GitHub organization? [Ensure Heroku Dashboard has team access](#)

rafaelq80/Testes_blogpessoal	Connect
rafaelq80/swagger_blogpessoal_springfox	Connect
rafaelq80/blogpessoal_T44	Connect
rafaelq80/swagger-blogPessoalSpring	Connect
rafaelq80/blogpessoal_T47	Connect
rafaelq80/blogpessoal	Connect

9. Observe que o Heroku indicará que a aplicação está **Conectada com o Repositório**.

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to [rafaelq80/blogpessoal](#) by [rafaelq80](#) Disconnect...

Releases in the [activity feed](#) link to GitHub to view commit diffs

10. No item **Automatic deploys**, no item **Choose a branch to deploy**, selecione a **Branch** que será usada para fazer o Deploy (main). Na sequência, clique no botão **Enable Automatic Deploys**, para automatizar o processo, ou seja, toda vez que você fizer um **push no Repositório no Github**, o Heroku tentará fazer o Deploy automaticamente.

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the [instructions here](#).

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically**: be sure that this branch is always in a deployable state and any tests have passed before you push. [Learn more](#)

Choose a branch to deploy

main

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Enable Automatic Deploys



ATENÇÃO: *O Deploy automático será concluído, APENAS E SOMENTE SE o código que foi enviado para o Github esteja sem erros e com o perfil PROD habilitado.

11. No item **Manual deploy**, clique no botão **Deploy branch**.

12. Logo abaixo, será exibida a janela do **Console do Heroku**. Acompanhe o processo do Deploy e aguarde a conclusão

13. Se o Deploy foi bem sucedido, será exibida a mensagem **Deploy to Heroku - Your app was successfully deployed**.

14. Clique no botão **View** para abrir a aplicação.



ATENÇÃO: *Caso aconteça algum erro no processo do Deploy será exibida a mensagem: **Build failed!**. Verifique o seu código e tente novamente.



Passo 12 - Abrir o Deploy no Navegador

1. Efetue login com uma conta de usuário antes de exibir a sua documentação no Swagger. Utilize o usuário em memória (root) para fazer o login.

bpspring.herokuapp.com

Este site está pedindo para você entrar na conta.

Nome de usuário

root

Senha

••••

Entrar Cancelar

2. Pronto! A Documentação no Swagger será exibida como a página inicial.

Swagger

Supported by SMARTBEAR

/v3/api-docs

Explore

Projeto Blog Pessoal

v0.0.1 OAS3

/v3/api-docs

Projeto Blog Pessoal - Generation Brasil

[Conteúdo Generation - Website](#)
[Send email to Conteúdo Generation](#)
[Generation Brasil](#)
[Github](#)

Servers

https://bpspring.herokuapp.com - Generated server url

usuario-controller

GET	/usuarios/all
POST	/usuarios/logar

3. Outra forma de abrir a aplicação é digitando o endereço:

<https://nomedoprojeto.herokuapp.com> no navegador.



Passo 13 - Testar o Deploy no Insomnia

1. Abra o Insomnia e acesse a Workspace **Blog Pessoal**.
2. Crie uma pasta chamada **Blog Pessoal** e arraste as 3 pastas (Postagem, Tema e Usuario) para dentro dela.
3. Duplica a pasta **Blog Pessoal**.
4. Na próxima janela, defina o nome da nova pasta como **Blog Pessoal - Heroku**.

Duplicate Folder

New Name

Blog Pessoal - Heroku

Create

5. Abra a requisição Cadastrar Usuário na pasta **Blog Pessoal - Heroku**.
6. Altere o caminho atual: <http://localhost:8080/usuarios/cadastrar>

POST

http://localhost:8080/usuarios/cadastrar

Send

Other Auth Query Header 1 Docs

7. Para o endereço do Heroku:
<https://meuprojeto.herokuapp.com/usuarios/cadastrar> (No exemplo abaixo:
<https://bprfp.herokuapp.com/usuarios/cadastrar>)

POST

http://bprfp.herokuapp.com/usuarios/cadastrar

Send

Other Auth Query Header 1 Docs

8. Execute a requisição e verifique se o Usuário foi criado corretamente.
9. Atualize o caminho de todas requisições da pasta **Blog Pessoal - Heroku**
10. Execute a requisição Login para acessar a API
11. Continue os testes conforme as orientações do **Checklist do Projeto Blog Pessoal**.

Como atualizar o Deploy no Heroku?



ALERTA DE BSM: *Mantenha a atenção aos detalhes e a persistência. Este item você utilizará apenas se você precisar alterar alguma coisa no seu projeto Spring e atualizar a aplicação na nuvem.*

1. Para fazer alterações no código do projeto e executar localmente, volte para o **STS** e altere a primeira linha do arquivo, **application.properties** conforme o código abaixo:

```
spring.profiles.active=dev
```

2. Faça as alterações necessárias no código do seu projeto, execute localmente e verifique se está tudo funcionando **sem erros**.
3. Antes de refazer o Deploy, altere novamente a primeira linha do arquivo, **application.properties** conforme o código abaixo:

```
spring.profiles.active=prod
```

4. Envie as atualizações do seu projeto para o repositório do Github, através do **Git Bash**, utilizando os comandos abaixo:

```
git add .
git commit -m "Atualização do Deploy - Blog Pessoal"
git push origin main
```

5. Ao finalizar o **git push**, o Heroku começará a refazer o Deploy. Acompanhe o processo pelo site do Heroku.

Personal > bpspring

GitHub rafaellq80/blogpessoal main

Overview Resources Deploy Metrics Activity Access Settings

Get a complete visualization of your app in a team-based continuous delivery environment with [Heroku Pipelines](#). [Hide](#) [Create a Heroku Pipeline](#)

Installed add-ons \$0.00/month [Configure Add-ons](#)

Heroku Postgres Hobby Dev postgresql-aerodynamic-95384

Dyno formation \$0.00/month [Configure Dynos](#)

Latest activity [All Activity](#)

rafaelproinfo@gmail.com: Deployed 7f1adb89 Just now - v8 [Compare diff](#)

rafaelproinfo@gmail.com: Build in progress Just now [View build progress](#)

6. Se o Deploy foi bem sucedido, será exibida a mensagem **BUILD SUCCEEDED**.



rafaelpinfo@gmail.com: Build succeeded

Today at 7:10 PM · [View build log](#)

7. Verifique se a Aplicação abre no Navegador e faça os testes no Insomnia.