

**Objetivos:**

- Usar o pacote Express para criar um servidor;
- Definir rotas;
- Restrição de acesso usando JWT (JSON Web Token);
- Uso do ORM Sequelize (Object-Relational Mapper) para persistir os dados no BD;
- Banco de dados PostgreSQL na cloud ElephantSQL;
- Deploy da aplicação Node no Heroku.

**Forma de entrega:** criar um repositório privado no GitHub para manter o projeto e incluir arleysouza como colaborador. Subir a aplicação no Heroku e enviar a URL de acesso para arleysouza no Teams.

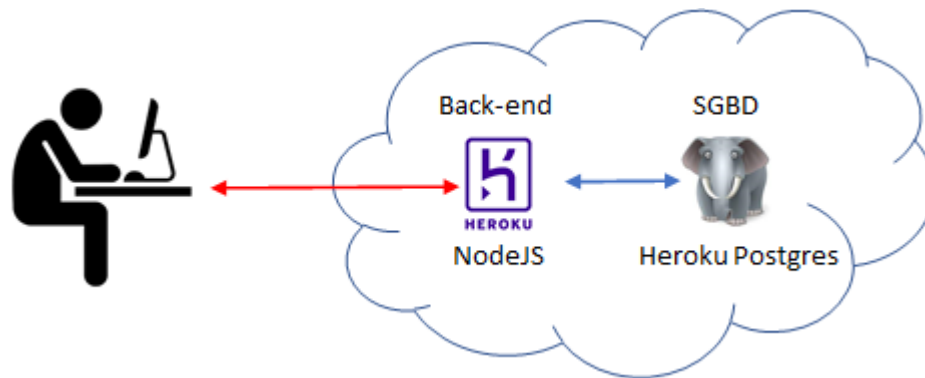
**Data de entrega:** 09/set.

**Descrição da atividade:** fazer uma aplicação Node para manter o cadastro de usuários e seus registros de vacina.

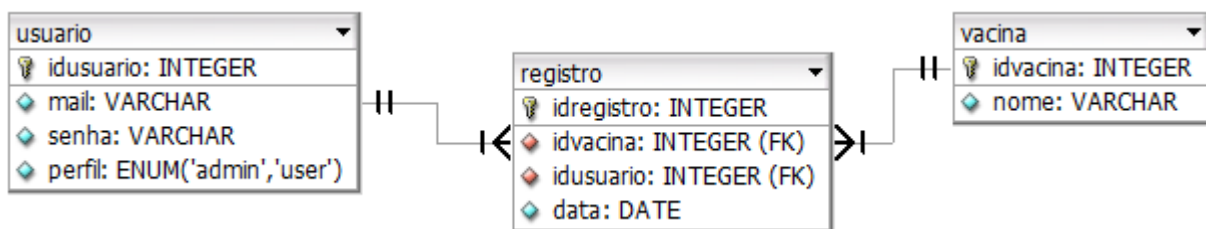
**Requisitos:**

- 1 - O usuário efetua o seu próprio cadastro;
- 2 - O usuário efetua login;
- 3 - O usuário altera mail e senha;
- 4 - O e-mail é único;
- 5 - O usuário admin possui a função de cadastrar, editar, excluir e listar as vacinas;
- 6 - Somente o usuário admin pode mudar o perfil de acesso de outros usuários;
- 7 - O usuário comum não pode cadastrar vacinas;
- 8 - O usuário registra que foi vacinado fornecendo a identificação da vacina e a data no formato YYYY-MM-DD;
- 9 - O usuário pode editar e excluir os seus registros de vacinação;
- 10 - O usuário pode listar os registros de vacina em ordem decrescente de data;
- 11 - O usuário possui acesso a somente os seus próprios registros de vacinação;
- 12 - Todas as operações requerem login;
- 13 - Os dados precisam ser persistidos no SGBD PostgreSQL da cloud ElephantSQL;
- 14 - Fazer deploy da aplicação na cloud do Heroku.

**Representação da estrutura da aplicação:** o BD no ElephantSQL e o servidor Node no Heroku.



**Modelo de dados:** considere a seguinte estrutura de tabelas.



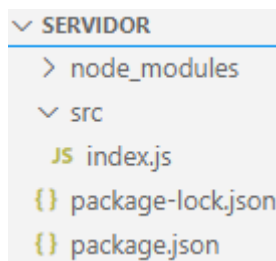
**Instruções para criar o projeto:** para codificar a aplicação você precisará do Visual Studio Code e ter instalado o Node e npm.

1. Crie uma pasta de nome **servidor** no local de sua preferência do computador;
2. Abra a pasta servidor no VS Code;
3. Acesse o terminal do VS Code e digite o comando **npm init -y**. Esse comando criará o arquivo **package.json**;
4. Digite **npm i bcrypt cors dotenv express jsonwebtoken pg sequelize** para instalar os pacotes necessários para o desenvolvimento da aplicação:
  - Utilizaremos o **bcrypt** para codificar a senha do usuário no BD (<https://www.npmjs.com/package/bcrypt>);
  - Utilizaremos o **cors** (Cross-origin Resource Sharing) para configurar o servidor para aceitar requisições de outros domínios (<https://www.npmjs.com/package/cors>);
  - Utilizaremos **dotenv** para ter acesso as variáveis de ambiente declaradas no arquivo **.env** (<https://www.npmjs.com/package/dotenv>);
  - Utilizaremos **express** para facilitar a codificação do servidor (<https://www.npmjs.com/package/express>);
  - Utilizaremos **jsonwebtoken** para criar a autorização de acesso do usuário as rotas e serviços no servidor (<https://www.npmjs.com/package/jsonwebtoken>);

- Utilizaremos `pg` para criar a interface com o SGBD PostgreSQL (<https://www.npmjs.com/package/pg>);
  - Utilizaremos `sequelize` como ORM para o PostgreSQL (<https://www.npmjs.com/package/sequelize>).
5. Digite o comando `npm i nodemon --save-dev` para adicionar o pacote `nodemon` como dependência de desenvolvimento, ou seja, essa dependência não será incluída no deploy do projeto. Esse comando adicionará a propriedade `devDependencies` no arquivo `package.json`:

```
"devDependencies": {
  "nodemon": "^2.0.12"
}
```

6. Crie uma pasta de nome `src` dentro da pasta `servidor` para melhorar a organização do código do projeto. Manteremos o nosso código dentro da pasta `src`;
7. Crie o arquivo `index.js` dentro da pasta `src`. Esse arquivo será chamado na inicialização do projeto. Ao final deste passo a estrutura de arquivos do projeto estará da seguinte forma:



8. Para o arquivo `src/index.js` ser executado ao inicializar o projeto teremos de incluir as propriedades `dev` e `start` na propriedade `scripts` do arquivo `package.json`. Ao final desses passos o arquivo `package.json` terá a seguinte estrutura:

```
{
  "name": "servidor",
  "version": "1.0.0",
  "description": "Rrotas, uso do ORM Sequelize e PostgreSQL Elephant",
  "main": "index.js",
  "scripts": {
    "start": "node ./src",
    "dev": "nodemon ./src"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.0.1",
    "cors": "^2.8.5",
    "dotenv": "^10.0.0",
    "express": "^4.17.1",
    "jsonwebtoken": "^8.5.1",
    "pg": "^8.7.1",

```

```
    "sequelize": "^6.6.5"
  },
  "devDependencies": {
    "nodemon": "^2.0.12"
  }
}
```

9. Crie o arquivo `.env` na raiz do projeto. Nesse arquivo colocaremos as variáveis de ambiente do projeto. Como exemplo a URL de acesso ao BD no ElephantSQL;
10. Crie o arquivo `.gitignore` e coloque somente o comando para ignorar a pasta `/node_modules` ao fazer o commit do projeto;

11. Como o objetivo é criar um projeto com a estrutura MVC (Model-View-Controller), então adicione as pastas `models`, `controllers` e `routes` na pasta `src`, assim como é mostrado ao lado.

Na pasta `models` colocaremos os arquivos para definir as instâncias usuário, vacina e registro.

Na pasta `controllers` colocaremos os arquivos com as operações de CRUD nas instâncias de usuário, vacina e registro.

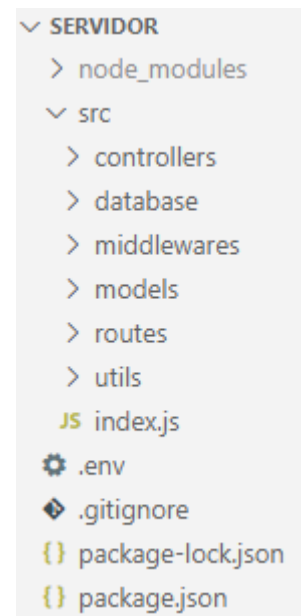
Na pasta `routes` colocaremos as rotas para os CRUDs.

Na pasta `database` colocaremos o código para criar uma instância do Sequelize e conectar com o SGBD no ElephantSQL.

Na pasta `middlewares` colocaremos o código para autenticar o usuário.

Na pasta `utils` colocaremos o código para decodificar o token de autenticação de usuário.

Essa organização do projeto possibilita distribuir os arquivos pela sua função no código.

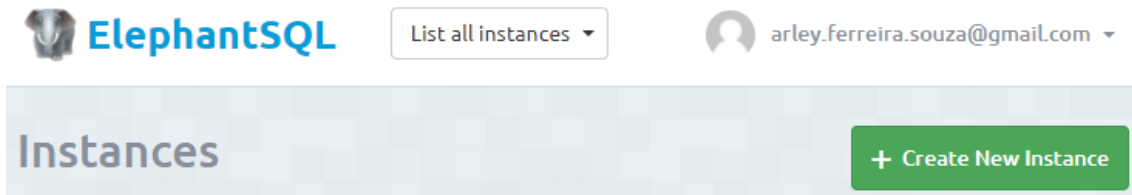


12. Para testar o projeto digite no terminal o comando `npm run dev` para rodar a versão de desenvolvimento e `npm run start` para rodar a versão de produção da aplicação. Esses comandos chamarão, respectivamente, os comandos que estão nas propriedades `start` e `dev` do `package.json`:

```
"scripts": {
  "dev": "nodemon ./src",
  "start": "node ./src"
}
```

**Instruções para criar uma instância no ElephantSQL:** utilizaremos o SGBD PostgreSQL hospedado na cloud.

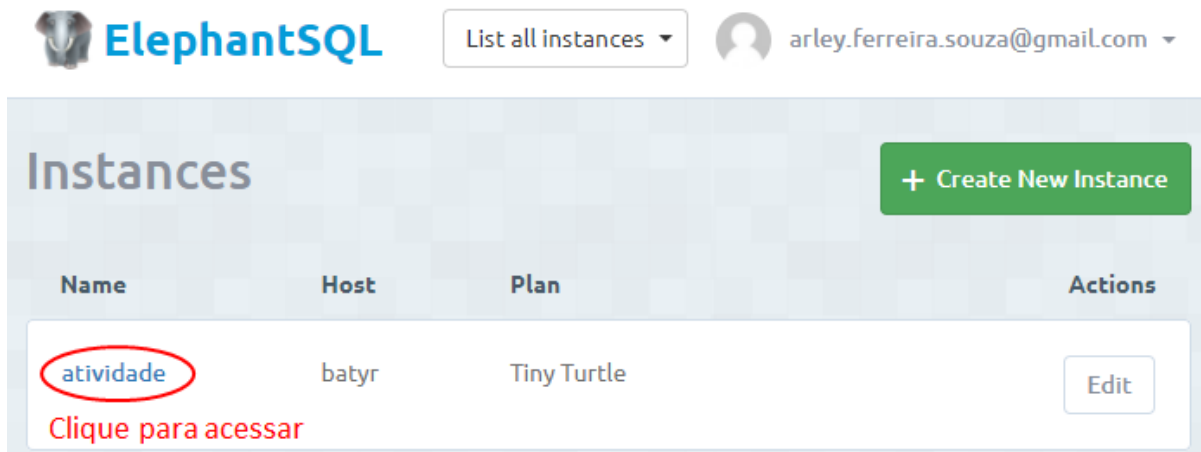
1. Acesso o <https://www.elephantsql.com/> e efetue o login;
2. Clique no botão **Create New Instance** para criar uma instância de BD;



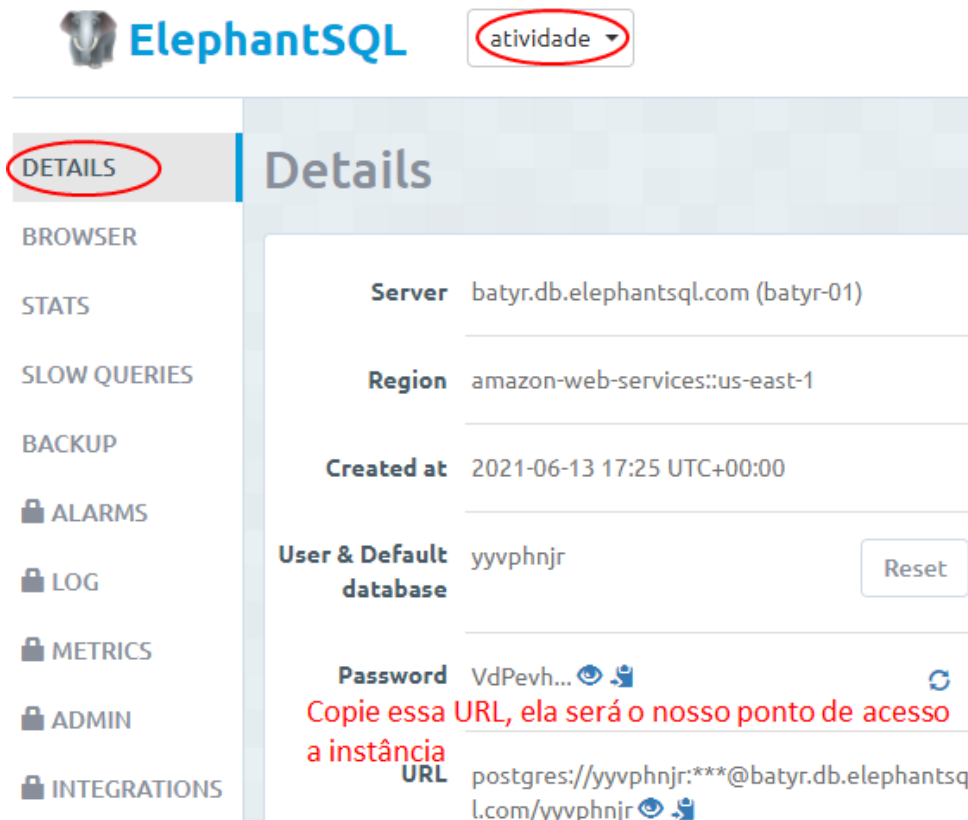
Na tela que será exibida dê o nome de **atividade** para a instância.

This is a 'Step 1 of 4' form titled 'Select a plan and name'. It contains three input fields: 'Name' with the value 'atividade', 'Plan' with a dropdown menu currently showing 'Tiny Turtle (Free)', and 'Tags' which is currently empty.

Na tela seguinte escolha o provedor (Amazon Web Services) e a região (US-East-1 (Northern Virginia)) de acordo com a sua vontade. Ao final teremos a instância **atividade** criada.



Na aba **Browser** você precisará copiar a URL de acesso a instância. Ela será utilizada ao criar uma instância do Sequelize para instanciar uma conexão com o SGBD.



3. Para criar uma instância do Sequelize utilizando a URL de acesso ao ElephantSQL use a dica disponível em <https://sequelize.org/master/manual/getting-started.html>;

**Instruções para criar um aplicativo no Heroku:** faremos o deploy da aplicação na cloud Heroku após finalizar o projeto.

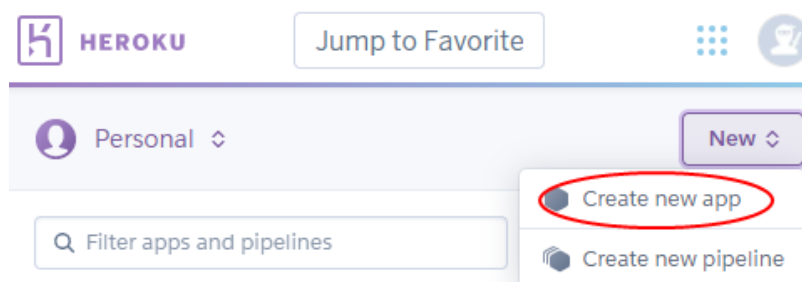
1. Antes de prosseguir será necessário instalar o Heroku CLI caso você não o tenha (<https://devcenter.heroku.com/articles/heroku-cli>). Apesar de ser possível instalar o Heroku CLI usando npm, o artigo sugere alguma outra forma.

```
npm install -g heroku
```

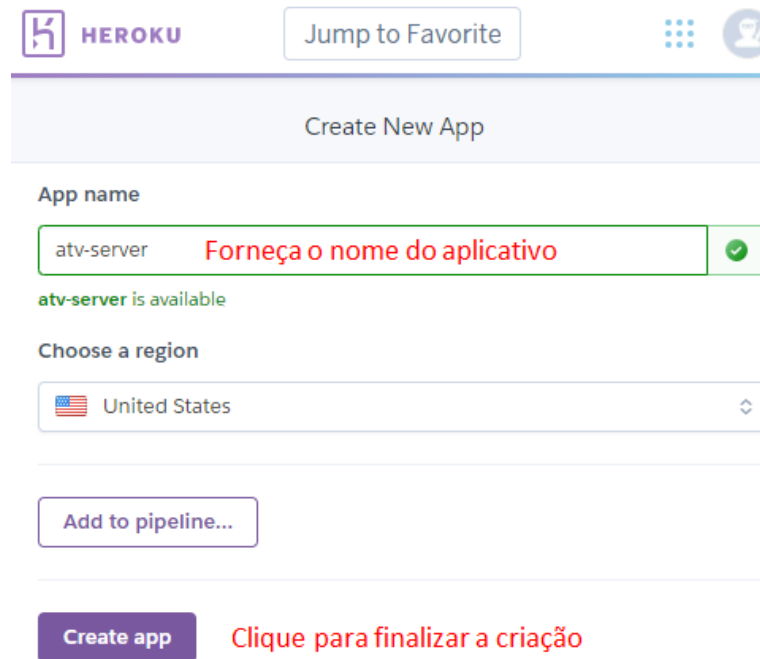
Após instalar o Heroku CLI, retorne para o terminal do VS Code e efetue login no Heroku, se ele não pedir para fazer a autenticação no navegador agora, ele pedirá no futuro:

```
heroku login
```

2. Acesse o Heroku para criarmos um aplicativo (<https://id.heroku.com/login>).

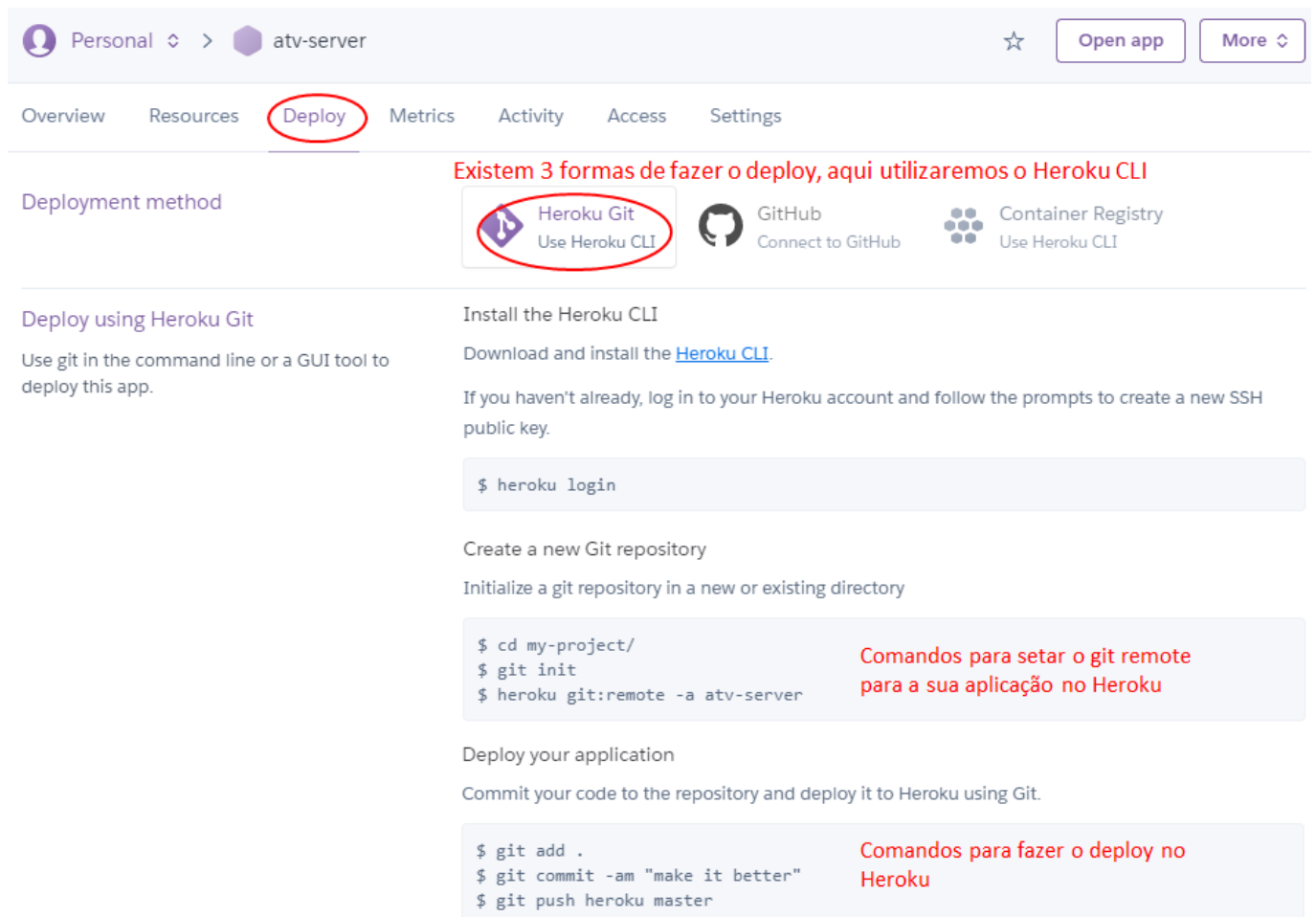


Forneça o nome do aplicativo, esse nome estará na URL de acesso.



The image shows the Heroku 'Create New App' form. At the top, there's a 'HEROKU' logo and a 'Jump to Favorite' button. Below is a 'Create New App' header. The 'App name' field contains 'atv-server' and has a red annotation 'Forneça o nome do aplicativo' with a green checkmark. Below the name field, it says 'atv-server is available'. The 'Choose a region' dropdown is set to 'United States'. There's an 'Add to pipeline...' button. At the bottom, there's a 'Create app' button and a red annotation 'Clique para finalizar a criação'.

3. Após criar o aplicativo acesse a aba [Deploy](#) para obtermos os comandos. Aqui utilizaremos o [Heroku CLI](#) para enviar os arquivos para a cloud.



The image shows the Heroku 'Deploy' page for the 'atv-server' app. The 'Deploy' tab is selected and circled in red. Under 'Deployment method', there are three options: 'Heroku Git Use Heroku CLI' (circled in red), 'GitHub Connect to GitHub', and 'Container Registry Use Heroku CLI'. A red annotation 'Existem 3 formas de fazer o deploy, aqui utilizaremos o Heroku CLI' points to the 'Heroku Git' option. Under 'Deploy using Heroku Git', there's a section 'Install the Heroku CLI' with instructions to download and install the CLI, and a code block with the command '\$ heroku login'. Below that, there's a section 'Create a new Git repository' with instructions to initialize a git repository and a code block with commands '\$ cd my-project/', '\$ git init', and '\$ heroku git:remote -a atv-server'. A red annotation 'Comandos para setar o git remote para a sua aplicação no Heroku' points to the '\$ heroku git:remote' command. Finally, there's a section 'Deploy your application' with instructions to commit code and deploy it, and a code block with commands '\$ git add .', '\$ git commit -am "make it better"', and '\$ git push heroku master'. A red annotation 'Comandos para fazer o deploy no Heroku' points to the '\$ git push heroku master' command.

Primeiramente temos de setar o repositório remoto no seu projeto, veja o exemplo a seguir. Ele poderá pedir para você abrir o navegador para autenticar:

```
PS D:\pessoal\servidor> git init
Initialized empty Git repository in D:/pessoal/servidor/.git/
PS D:\pessoal\servidor> heroku git:remote -a atv-server
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/aa341d1f-2063
jE1M24GAJC6BYh7AWIAAVGA.AmWPkmAucZiaxgx4kcaeChHxZrH0rVrtlwVoBS7t80g
Logging in... done
set git remote heroku to https://git.heroku.com/atv-server.git
PS D:\pessoal\servidor> git remote heroku
```

← Após concluir, verifique se o repositório remoto foi setado para o heroku

4. Utilize os comandos sugeridos na página de deploy para efetuar o deploy. Antes de fazer o git add certifique-se que a pasta node\_modules esteja excluída no arquivo .gitignore:

```
$ git add .
$ git commit -am "make it better"
$ git push heroku master
```

Ao finalizar o deploy ele mostrará a URL de acesso a aplicação, por exemplo:

```
remote: -----> Compressing...
remote:          Done: 36.1M
remote: -----> Launching...
remote:          Released v3
remote:          https://atv-server.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/atv-server.git
* [new branch]      master -> master
```