

Deploy de uma aplicação

Plataforma utilizada

Temos algumas opções conhecidas e que podemos utilizar para fazer o deploy com plano gratuito para uma aplicação com Flask e Python. O Render, PythonAnywhere, Railway e algumas outras.



Deploy Vendure to



Deploy Vendure to



Render

Render é uma plataforma de hospedagem em nuvem que simplifica o processo de deployment e gerenciamento de aplicações web, APIs e serviços. Com uma interface fácil de usar, Render permite que desenvolvedores implementem suas aplicações rapidamente, sem a necessidade de gerenciar a infraestrutura subjacente.

Render é uma ótima opção para desenvolvedores que desejam uma solução de hospedagem ágil e eficiente, com menos complexidade na configuração e manutenção.

PythonAnywhere

PythonAnywhere é um ambiente de desenvolvimento integrado e serviço de hospedagem web baseado na linguagem de programação Python. Ele fornece acesso, por meio de navegador web, ao Python baseado em servidor e interfaces de linha de comando Bash, juntamente com um editor de código com realce de sintaxe.

Ele funciona como um SaaS. O modelo SaaS permite às empresas usar softwares hospedados na nuvem na modalidade de assinatura, evitando custos com hardware e atualizações.

Railway

Railway é uma plataforma moderna de desenvolvimento e implantação que torna o processo de criar, gerenciar e escalar aplicações mais acessível. Com uma interface intuitiva, permite que desenvolvedores configurem seus projetos rapidamente, sem complicações. A implantação é facilitada por meio de comandos simples e integração com repositórios Git, o que otimiza o fluxo de trabalho.

Embora seja popular para aplicações Node.js, Railway também suporta Python, Ruby, Go e outras linguagens, tornando-se uma opção versátil. A plataforma oferece suporte integrado para bancos de dados como PostgreSQL e MySQL, permitindo a configuração e gestão direta dos bancos.

Código e configurações necessárias para o deploy com Render

Clone o repositório no link https://github.com/andrelbribeiro/sge_py.git.

Após isso apague o ambiente virtual que está no repositório (/venv), e crie um novo ambiente virtual: `python -m venv venv`

Ative o ambiente virtual : `.\venv\Scripts\activate`

Ou faça usando alguma outra biblioteca

E instale o gunicorn: `pip install gunicorn`

O gunicorn será responsável por gerenciar seu aplicativo em produção.

Instale também o Flask, SQLAlchemy e Flask-SQLAlchemy nesse ambiente caso tenha feito o clone: `pip install Flask SQLAlchemy Flask-SQLAlchemy`

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

(venv) PS D:\fap\deploy-flask\sge_py> pip install gunicorn
Collecting gunicorn
  Downloading gunicorn-23.0.0-py3-none-any.whl.metadata (4.4 kB)
Collecting packaging (from gunicorn)
  Downloading packaging-24.1-py3-none-any.whl.metadata (3.2 kB)
Downloading gunicorn-23.0.0-py3-none-any.whl (85 kB)
Downloading packaging-24.1-py3-none-any.whl (53 kB)
Installing collected packages: packaging, gunicorn
Successfully installed gunicorn-23.0.0 packaging-24.1
(venv) PS D:\fap\deploy-flask\sge_py> 
```


Agora você precisará criar um arquivo com todas as bibliotecas instaladas em seu ambiente de desenvolvimento para que o Render possa utilizá-las no ambiente de produção.

1. Crie um arquivo chamado “[requirements.txt](#)” na raiz do projeto.

2. Em seu ambiente rode o comando: [pip freeze](#)

Você pode copiar e colar todos as dependências que foram mostradas com o comando anterior ou rodar o comando: [pip freeze > requirements.txt](#)

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
(venv) PS D:\fap\deploy-flask\sge_py> pip freeze
blinker==1.8.2
click==8.1.7
colorama==0.4.6
Flask==3.0.3
Flask-SQLAlchemy==3.1.1
greenlet==3.1.1
gunicorn==23.0.0
itsdangerous==2.2.0
Jinja2==3.1.4
MarkupSafe==3.0.2
packaging==24.1
SQLAlchemy==2.0.36
typing_extensions==4.12.2
Werkzeug==3.0.6
(venv) PS D:\fap\deploy-flask\sge_py> pip freeze > requirements.txt
(venv) PS D:\fap\deploy-flask\sge_py>
```

```
requirements.txt  U  X
requirements.txt
1  blinker==1.8.2
2  click==8.1.7
3  colorama==0.4.6
4  Flask==3.0.3
5  Flask-SQLAlchemy==3.1.1
6  greenlet==3.1.1
7  gunicorn==23.0.0
8  itsdangerous==2.2.0
9  Jinja2==3.1.4
10 MarkupSafe==3.0.2
11 packaging==24.1
12 SQLAlchemy==2.0.36
13 typing_extensions==4.12.2
14 Werkzeug==3.0.6
15
```


Modifique o arquivo app.py para que ele crie a aplicação com a função criar_app() e exponha a instância da aplicação. Faça isso removendo a linha app.run(debug=True) de dentro da função criar_app e criando uma instância app fora do if __name__ == '__main__'

```
app.py > ...
1  from flask import Flask      Import "flask" could not be resolved
2  from models import db
3  from config import Config
4  from controllers.usuario_controller import usuario_bp
5  from controllers.produto_controller import produto_bp
6  from controllers.cliente_controller import cliente_bp
7  from controllers.pedido_controller import pedido_bp
8  from controllers.detalhepedido_controller import detalhePedido_bp
9
10 # Função para criar a aplicação
11 def criar_app():
12     # Instância do Flask
13     app = Flask(__name__)
14     app.config.from_object(Config)
15     db.init_app(app)
16
17     with app.app_context():
18         db.create_all()
19
20     app.register_blueprint(usuario_bp)
21     app.register_blueprint(produto_bp)
22     app.register_blueprint(cliente_bp)
23     app.register_blueprint(pedido_bp)
24     app.register_blueprint(detalhePedido_bp)
25
26     return app
27
28 # Instância da aplicação
29 app = criar_app()
30
31 if __name__ == '__main__':
32     app.run(debug=True)
33
```

Adicionamos o arquivo que criamos para commit com: `git add requirements.txt app.py`
E fazemos o commit: `git commit -m "add requirements.txt and changed app.py"`
Se nosso projeto ainda não estiver versionado podemos criar um novo repositório no Github antes de subir

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
(venv) PS D:\fap\deploy-flask\sge_py> pip freeze > requirements.txt
(venv) PS D:\fap\deploy-flask\sge_py> git add requirements.txt app.py
(venv) PS D:\fap\deploy-flask\sge_py> git commit -m "add requirements.txt and changed app.py"
(venv) PS D:\fap\deploy-flask\sge_py>
```

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner * Eduardo-J-S / **Repository name *** flask-deploy

✔ flask-deploy is available.

Great repository names are short and memorable. Need inspiration? How about [symmetrical-guide](#) ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Remova o link para o repositório remoto original (para evitar acidentalmente fazer commits no repositório original): `git remote remove origin`

Definimos a branch: `git branch -M main`

Adicione o link do seu novo repositório como origin: `git remote add origin <url-do-seu-repositorio>`

Suba o código para o seu repositório: `git push -u origin main`

```
Quick setup — if you've done this kind of thing before
[ ] Set up in Desktop or HTTPS SSH https://github.com/Eduardo-J-S/flask-deploy.git
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and other files.

...or create a new repository on the command line

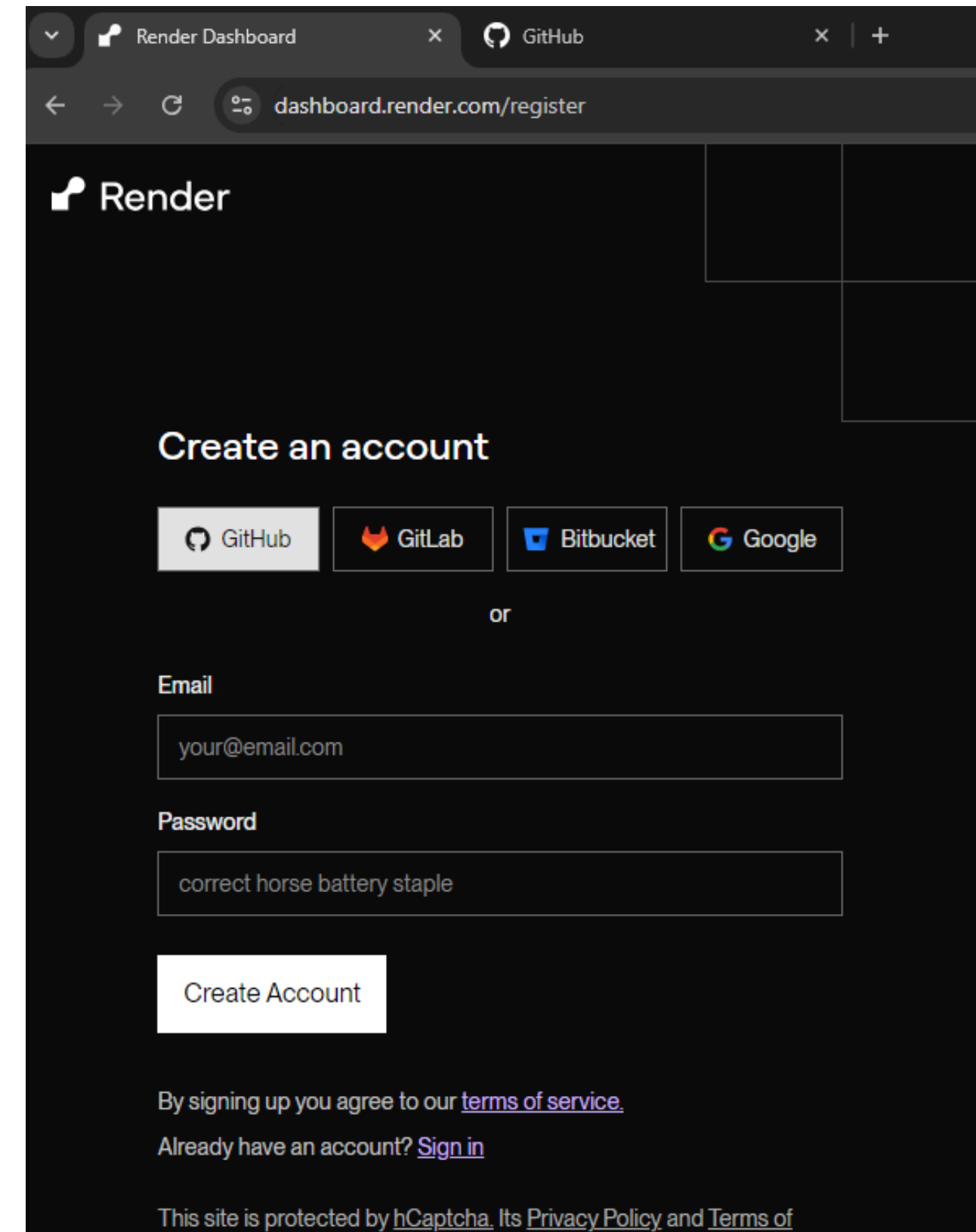
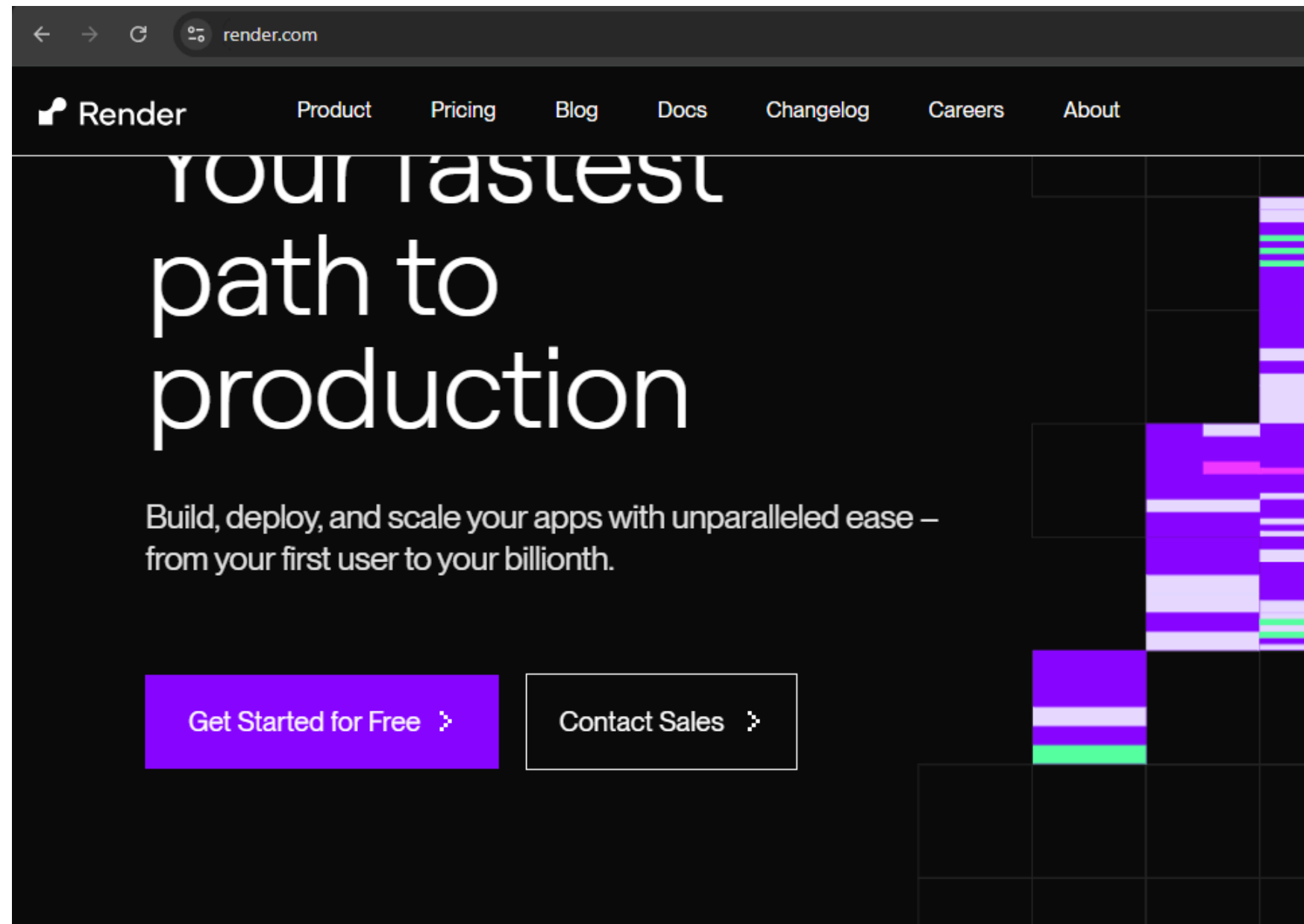
echo "# flask-deploy" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Eduardo-J-S/flask-deploy.git
git push -u origin main

...or push an existing repository from the command line

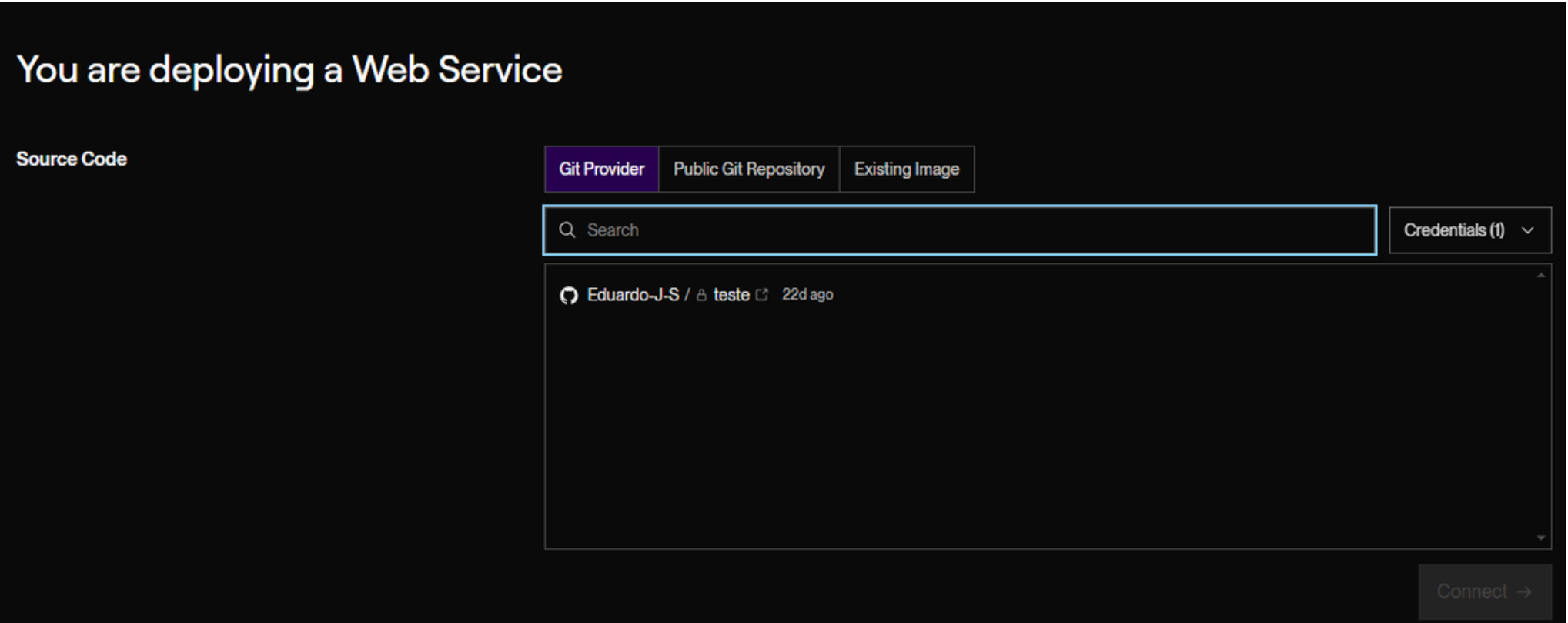
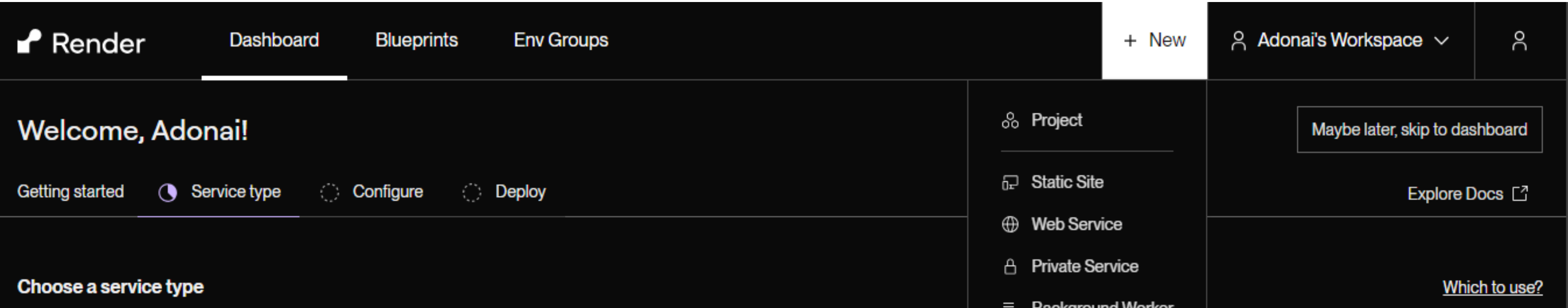
git remote add origin https://github.com/Eduardo-J-S/flask-deploy.git
git branch -M main
git push -u origin main
```

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
(venv) PS D:\fap\deploy-flask\sge_py> git remote remove origin
```

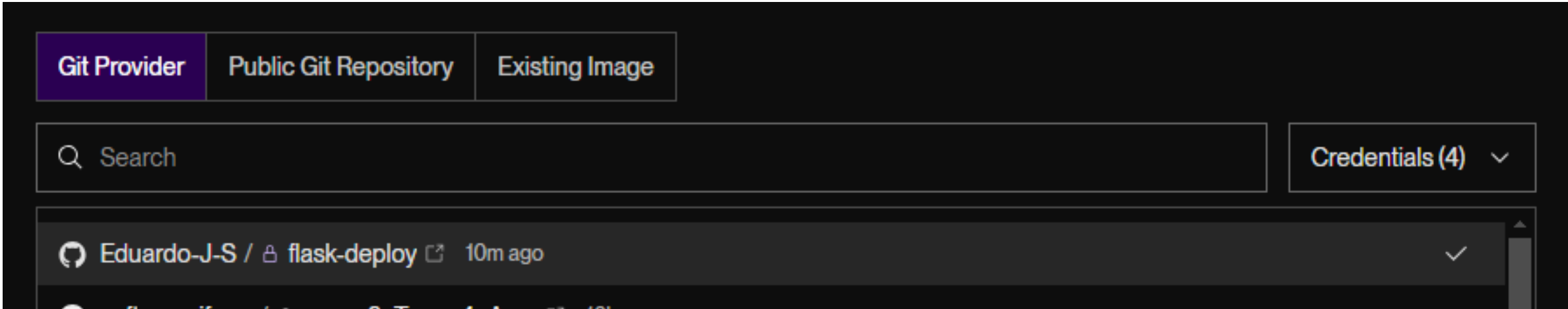
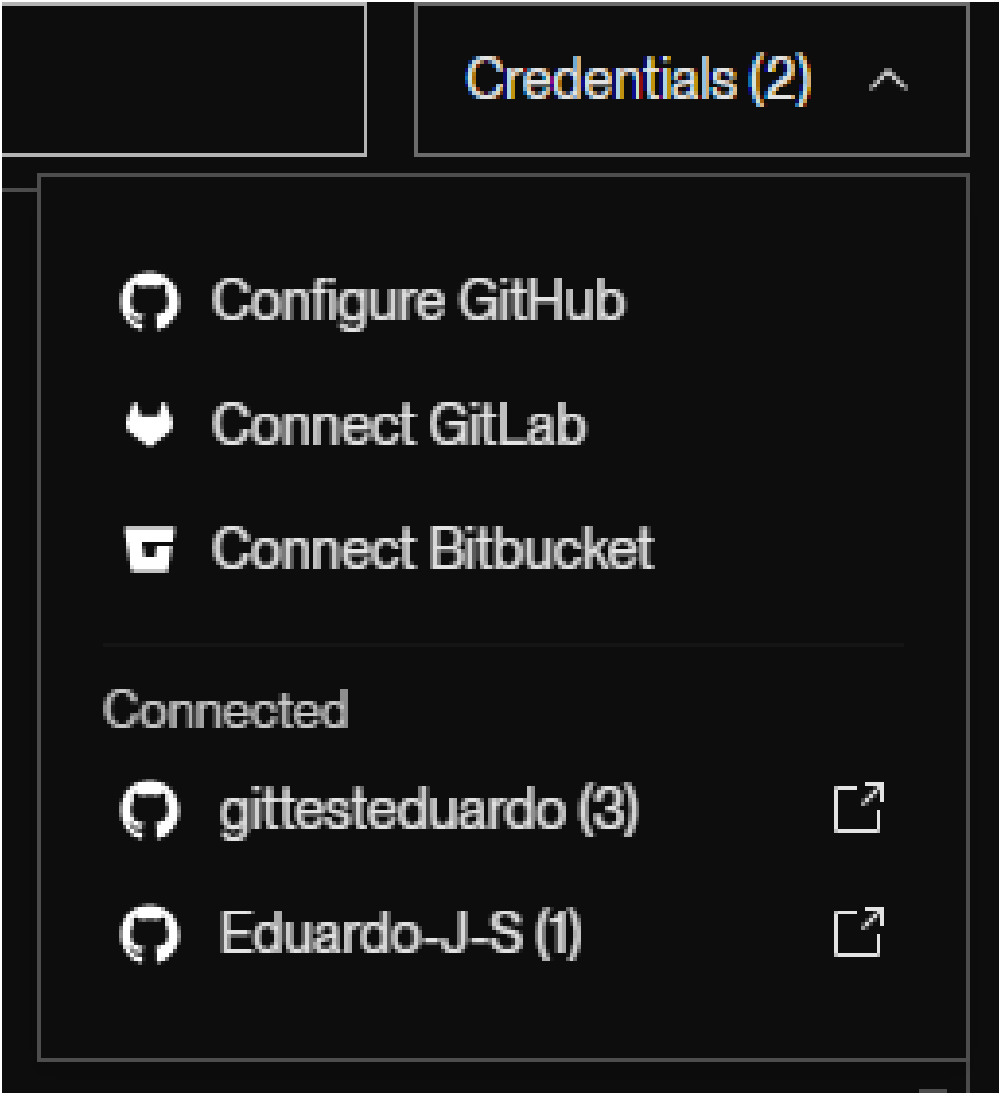
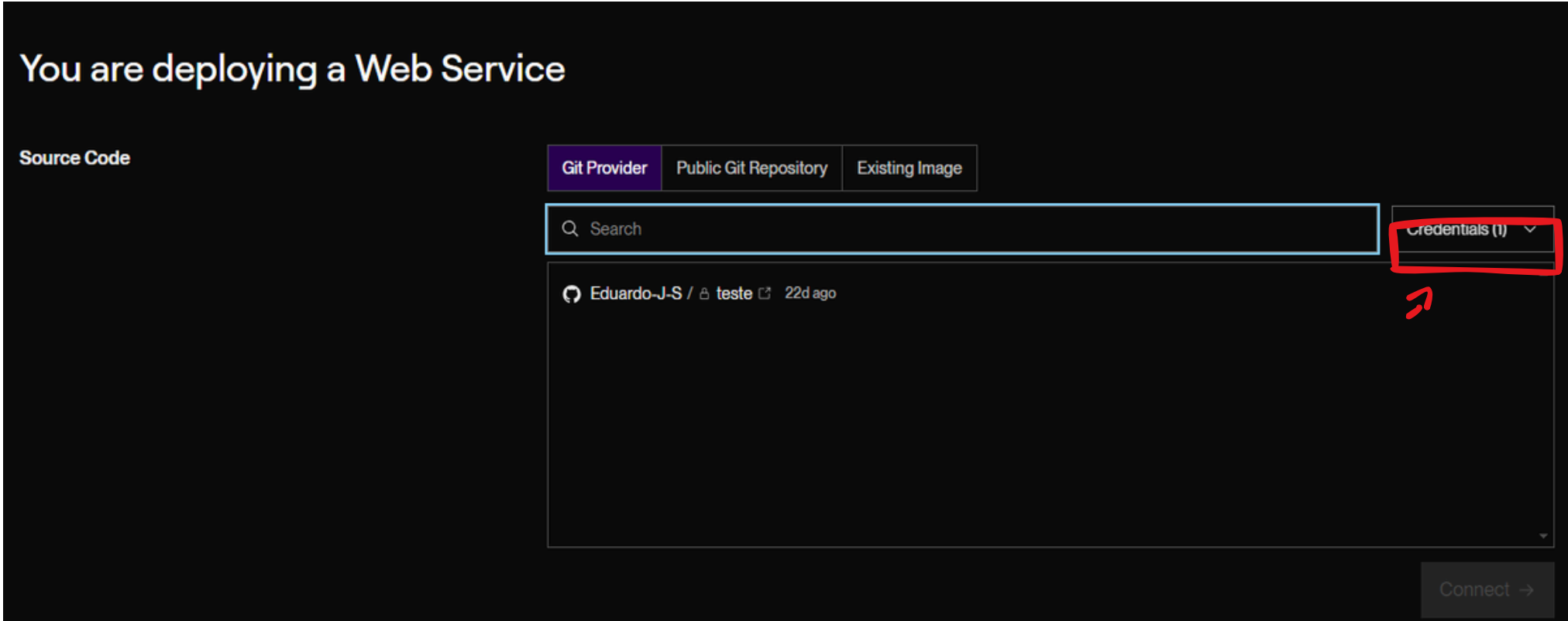
Agora vamos criar uma conta no [Render](#). Primeiro entramos no site do [Render](#) e então podemos criar nossa conta com o próprio GitHub que ele já integra com os nossos repositórios



Próximo passo é clicar em “New” no topo da página, em seguida em “web service”. Se o GitHub estiver conectado ele vai reconhecer os repositórios. Então é só selecionar o que acabamos de subir.



Se os seus repositórios não estiverem aparecendo automaticamente, você irá precisar adicionar as credencias. Clique onde está marcado, depois selecione a opção “Configure GitHub”, selecione a credencial onde está seu repositório e depois clique em “install”



Configurações básicas para o deploy. Defina nome, linguagem, branch e região.

Name A unique name for your web service.	flask-deploy
Language	Python 3
Branch The Git branch to build and deploy.	main
Region Your services in the same region can communicate over a private network .	Ohio (US East)

Este primeiro campo é basicamente para instalar as dependências e preparar o ambiente da aplicação para rodar. No caso da aplicação, vamos instalar as bibliotecas especificadas no arquivo requirements.txt.

Esse comando instalará todas as dependências necessárias para que o projeto funcione corretamente.

No campo "Start Command", configuramos o comando para iniciar o servidor em produção. Usamos o gunicorn como servidor WSGI para rodar a aplicação de maneira eficiente. Aqui, app:app indica o caminho do módulo (app) e a instância da aplicação Flask (app).

Build Command Render runs this command to build your app before each deploy.	<code>\$ pip install -r requirements.txt</code>
Start Command Render runs this command to start your app with each deploy.	<code>\$ gunicorn app:app</code>

Selecionamos o plano FREE

Instance Type

For hobby projects

Free

\$0 / month

512 MB (RAM)

0.1 CPU

For professional use

For more power and to get the most out of Render, we recommend using one of our paid instance types. All paid instances support:

▪ Zero Downtime

▪ SSH Access

⚠ Upgrade to enable more features

Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.

<div>Starter</div> <div>\$7 / month</div> <div>512 MB (RAM)</div> <div>0.5 CPU</div>	<div>Standard</div> <div>\$25 / month</div> <div>2 GB (RAM)</div> <div>1 CPU</div>
<div>Pro</div> <div>\$75 / month</div> <div>4 GB (RAM)</div> <div>0.5 CPU</div>	<div>Pro Plus</div> <div>\$175 / month</div> <div>8 GB (RAM)</div> <div>1 CPU</div>

E criamos o deploy

Advanced

Deploy Web Service

Log de quando ele subir a aplicação será assim.

```
October 26, 2024 at 3:29 PM ✓ Live  
f32fe65 add requirements.txt and changed app.py  
  
All logs Search Live tail GMT-3  
er, werkzeug, SQLAlchemy, Jinja2, gunicorn, Flask, Flask-SQLAlchemy  
Oct 26 03:30:04 PM ⓘ Successfully installed Flask-3.0.3 Flask-SQLAlchemy-3.1.1 Jinja2-3.1.4 MarkupSafe-3.0.2 SQLAlchemy-2.0.36 Werkzeug-3.0.6 blinker-1.8.2 click-8.1.7 colorama-0.4.6 greenlet-3.1.1 gunicorn-23.0.0 itsdangerous-2.2.0 packaging-24.1 typing_extensions-4.12.2  
Oct 26 03:30:05 PM ⓘ  
Oct 26 03:30:05 PM ⓘ [notice] A new release of pip is available: 24.0 -> 24.2  
Oct 26 03:30:05 PM ⓘ [notice] To update, run: pip install --upgrade pip  
Oct 26 03:30:10 PM ⓘ ==> Uploading build...  
Oct 26 03:30:18 PM ⓘ ==> Build uploaded in 8s  
Oct 26 03:30:18 PM ⓘ ==> Build successful 🎉  
Oct 26 03:30:19 PM ⓘ ==> Deploying...  
Oct 26 03:31:14 PM ⓘ ==> Running 'gunicorn app:app'  
Oct 26 03:31:21 PM ⓘ [2024-10-26 18:31:21 +0000] [93] [INFO] Starting gunicorn 23.0.0  
Oct 26 03:31:21 PM ⓘ [2024-10-26 18:31:21 +0000] [93] [INFO] Listening at: http://0.0.0.0:10000 (93)  
Oct 26 03:31:21 PM ⓘ [2024-10-26 18:31:21 +0000] [93] [INFO] Using worker: sync  
Oct 26 03:31:21 PM ⓘ [2024-10-26 18:31:21 +0000] [94] [INFO] Booting worker with pid: 94  
Oct 26 03:31:21 PM ⓘ 127.0.0.1 - - [26/Oct/2024:18:31:21 +0000] "HEAD / HTTP/1.1" 404 0 "-" "Go-http-client/1.1"  
Oct 26 03:31:30 PM ⓘ ==> Your service is live 🎉
```

Podemos então copiar a URL e fazer as requisições

RenderDashboardBlueprintsEnv Groups

WEB SERVICE

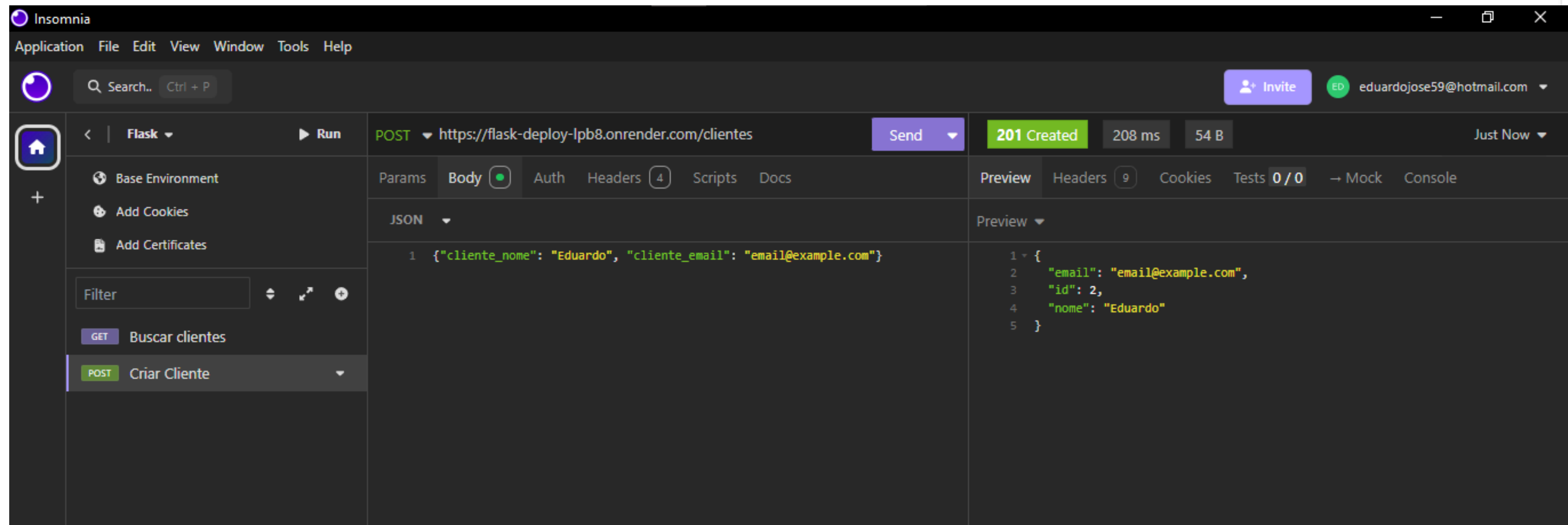
flask-deployPython 3FreeUpgrade your instance →

Eduardo-J-S / flask-deploymain

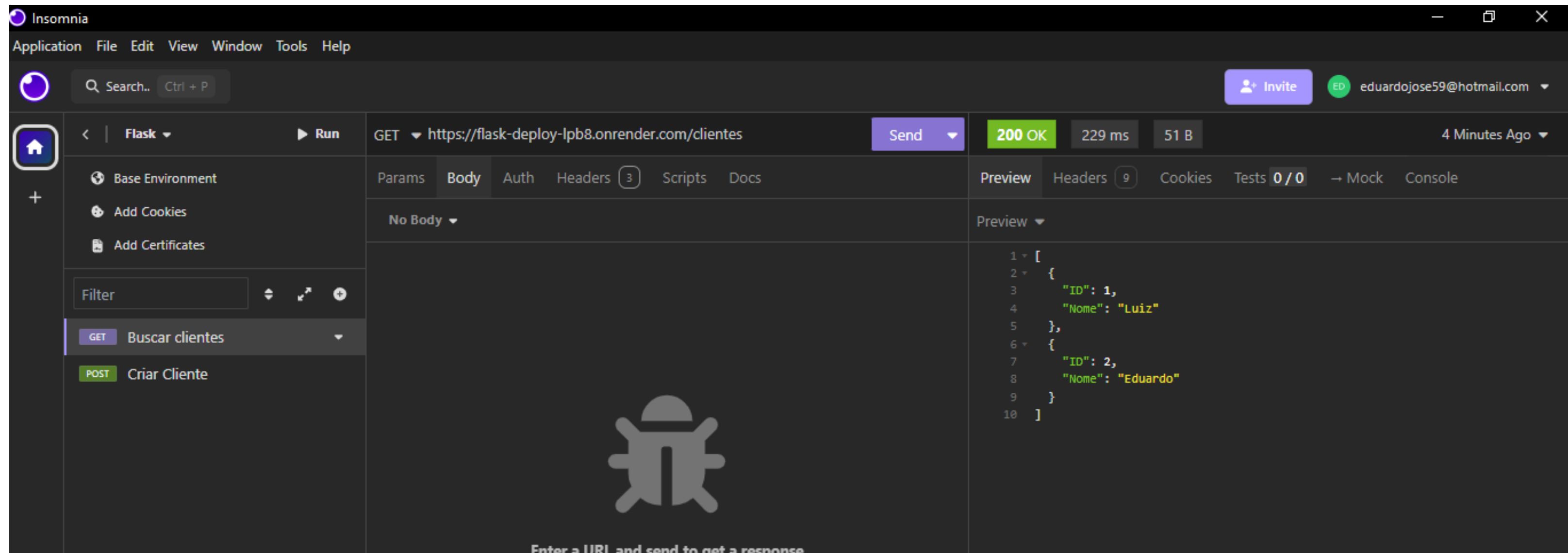
https://flask-deploy-lpb8.onrender.com

Events

ⓘ Your free instance will spin down wi



Exemplo de uma requisição POST de criação de cliente



Exemplo de uma requisição GET para buscar todos os cliente