



2023
MANUAL



LAMBDA AWS
CONFIGURAÇÃO

Sumário

1.Configuração.....	03
2.Teste Endpoint.....	11
3.Testes Unitários.....	15



01

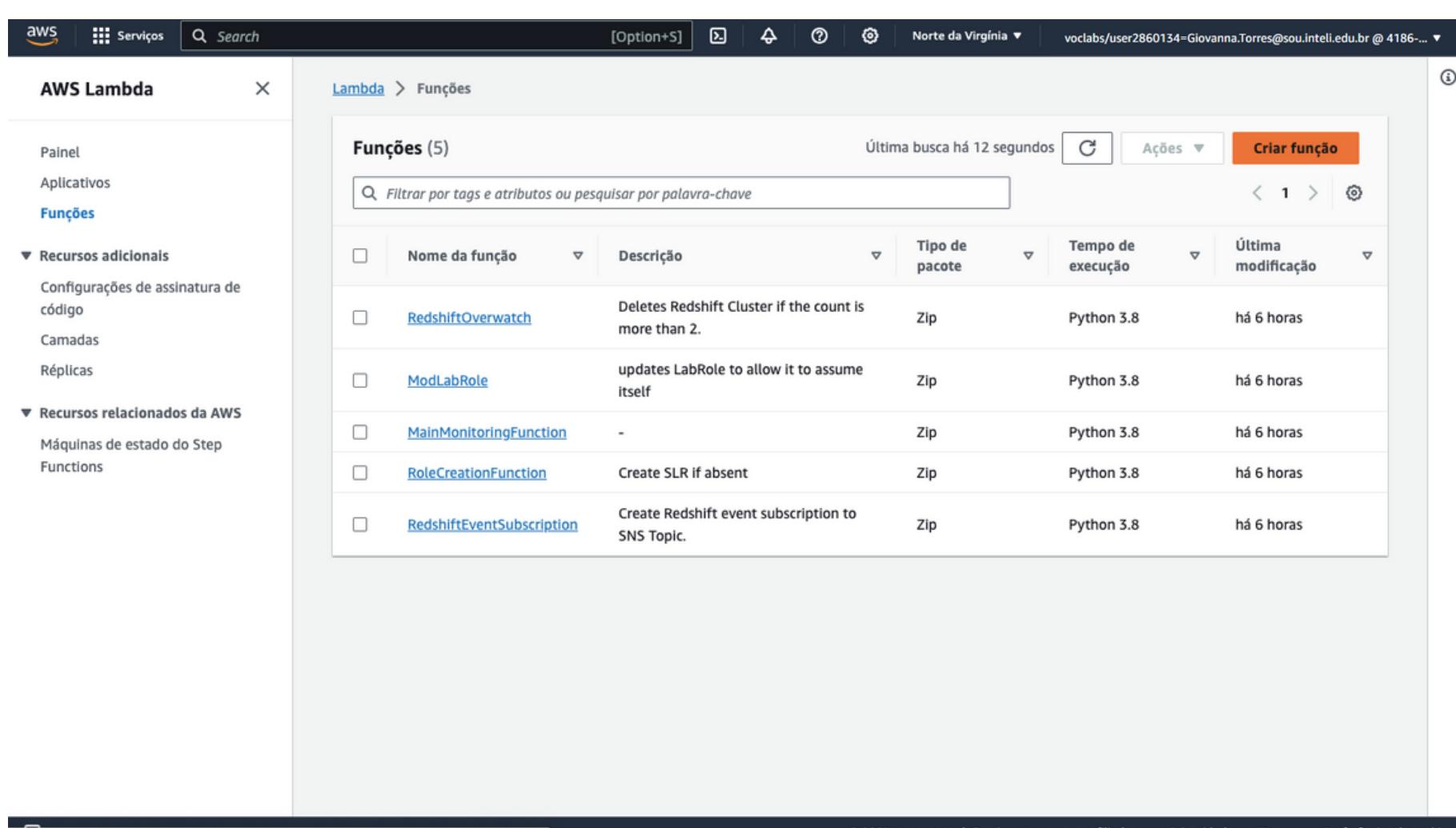
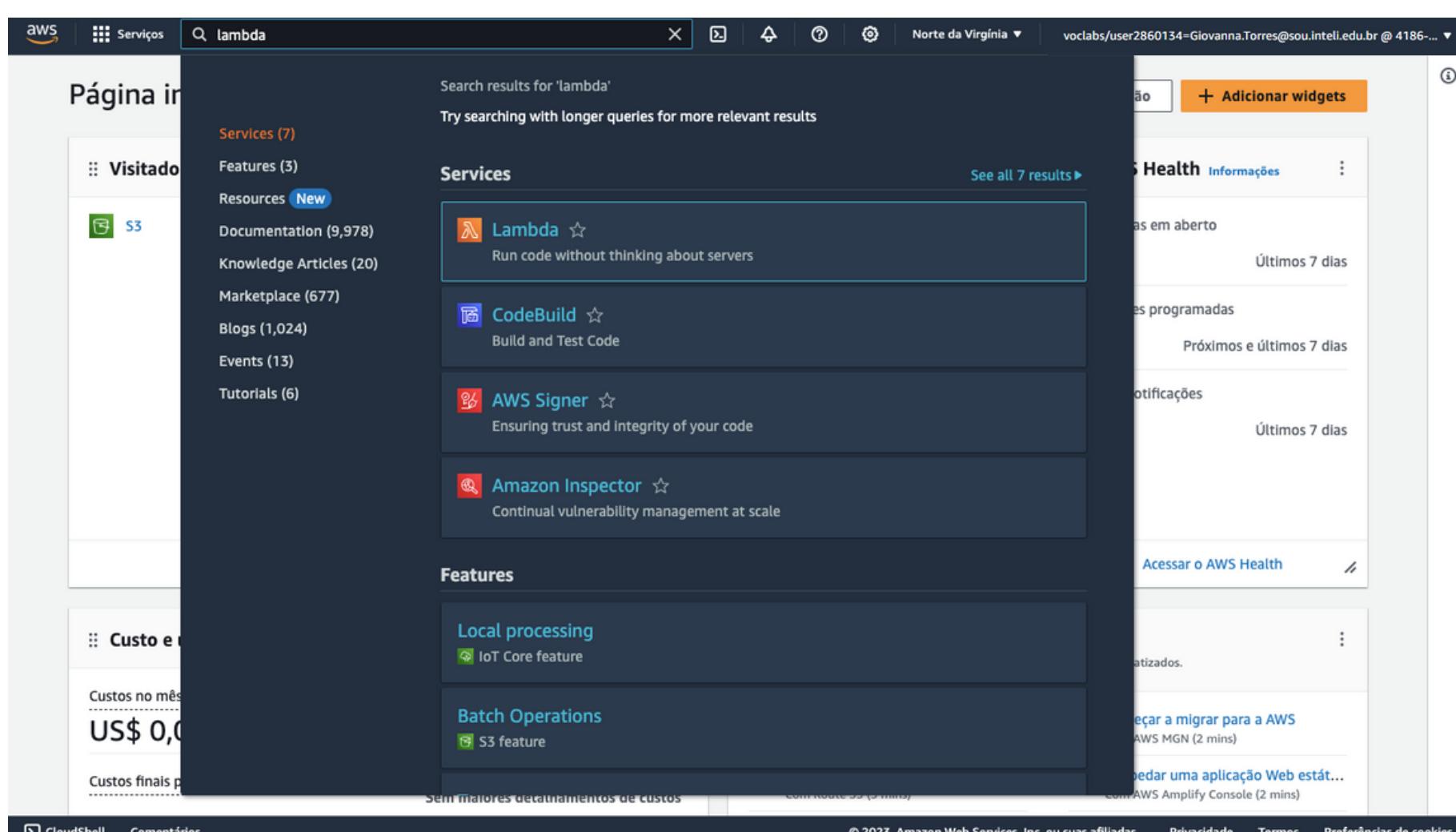
CONFIGURAÇÃO

CONFIGURAÇÃO

Neste manual, você aprenderá a criar um serviço na AWS, utilizando AWS Lambda e o Amazon API Gateway, com a linguagem de programação Python. Siga as etapas detalhadas a seguir para começar a construir o serviço na AWS.

PASSO 1

No painel de serviços da AWS, clique em "Serviços" e depois em "Lambda". Clique em "Criar função".



PASSO 2

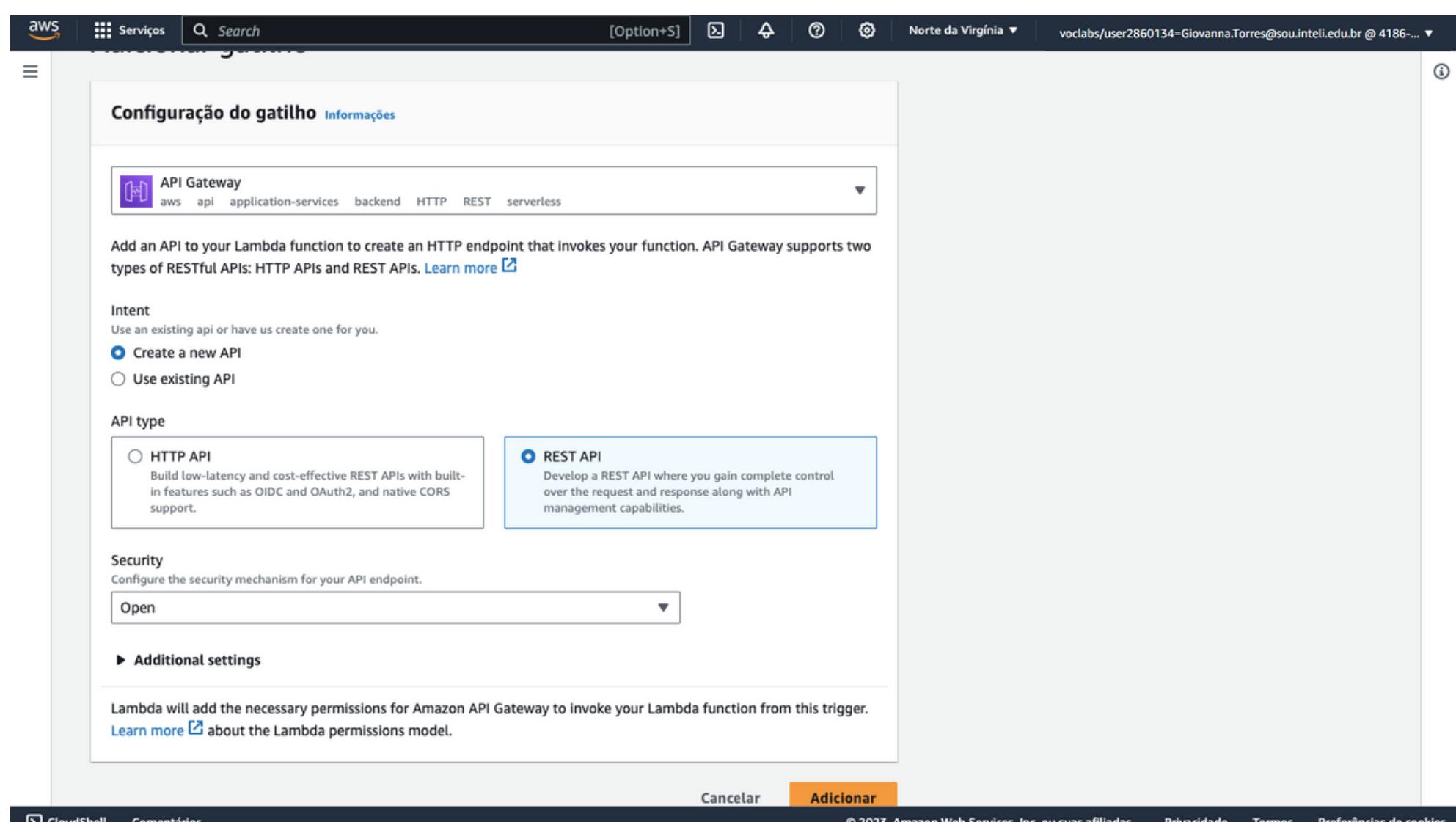
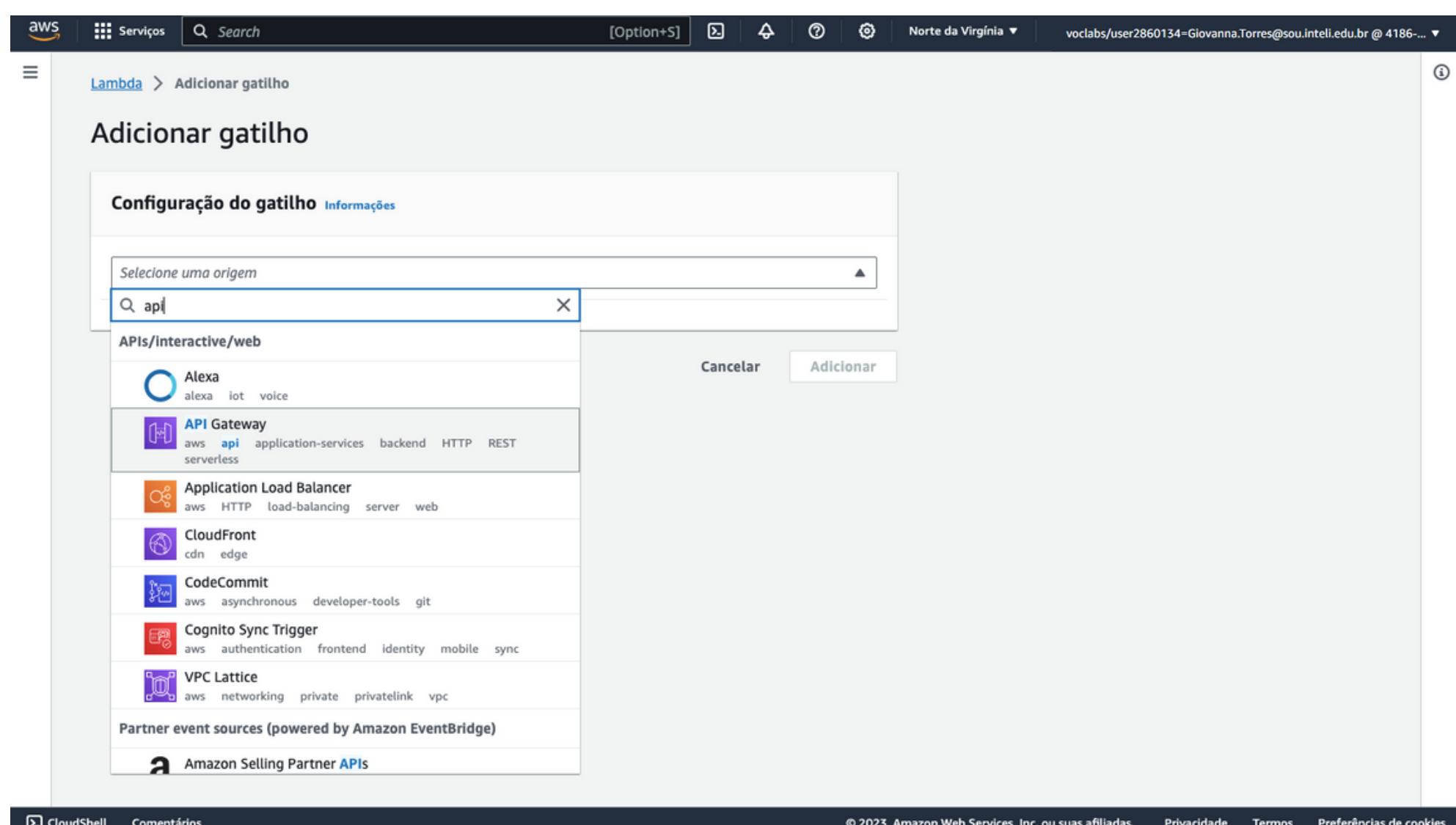
Configure o nome da função, selecione o tempo de execução e, considerando que estamos em um ambiente de laboratório, optaremos por utilizar uma função já existente. No campo "Função existente", escolha "LabRole".

The screenshot shows the AWS Lambda function creation interface. In the 'Nome da função' field, 'lambda-flask-function' is entered. Under 'Tempo de execução', 'Python 3.11' is selected. In the 'Arquitetura' section, 'x86_64' is chosen. Under 'Permissões', 'LabRole' is selected from the dropdown. The 'Papel de execução' section shows 'Usar uma função existente' is selected. The 'Função existente' dropdown also contains 'LabRole'. At the bottom, there are buttons for 'CloudShell', 'Comentários', and links to '© 2023, Amazon Web Services, Inc. ou suas afiliadas.', 'Privacidade', 'Termos', and 'Preferências de cookies'.

The screenshot shows the 'Visão geral da função' (Function Overview) page for 'lambda-flask-function'. It displays the function name, a description box, and sections for triggers and destinations. Below this is the 'Origem do código' (Code Source) section, which includes tabs for 'Código', 'Testar', 'Monitor', 'Configuração', 'Aliases', and 'Versões'. The 'Código' tab is active. A 'Fazer upload de' button is visible. The bottom navigation bar includes 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test', 'Deploy', and links to 'CloudShell', 'Comentários', and '© 2023, Amazon Web Services, Inc. ou suas afiliadas.', 'Privacidade', 'Termos', and 'Preferências de cookies'.

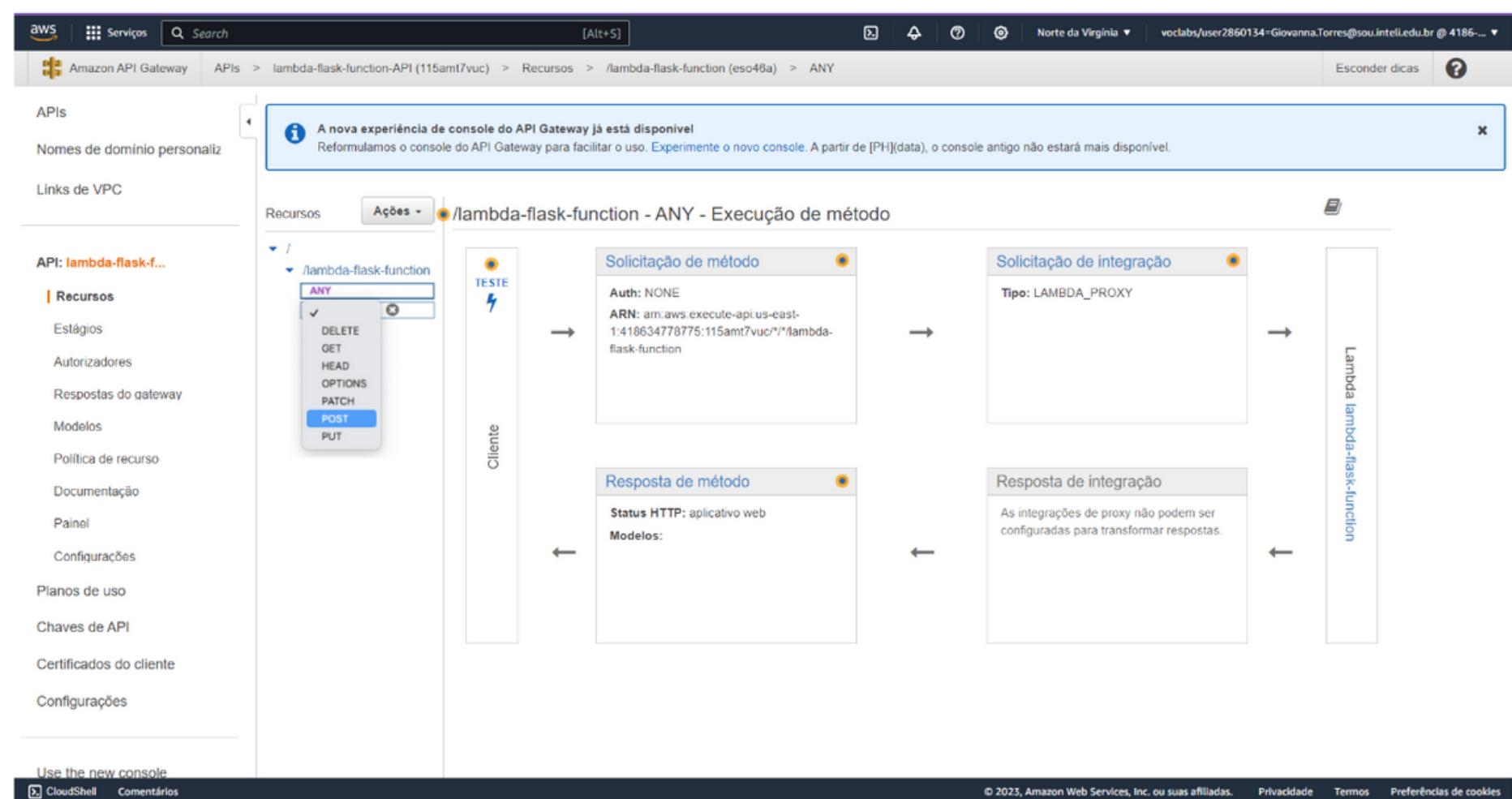
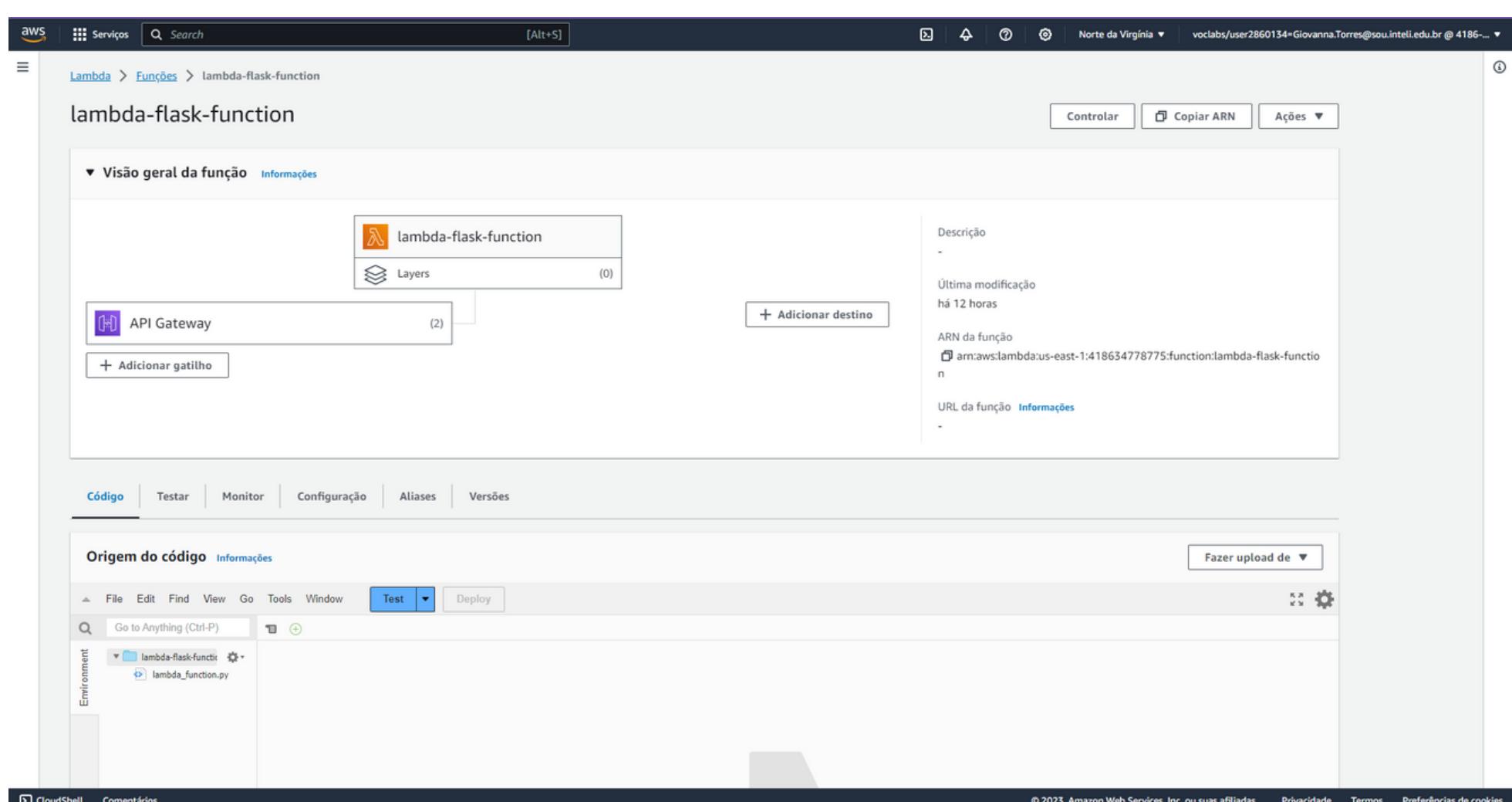
PASSO 3

Configure o nome da função, selecione o tempo de execução e, considerando que estamos em um ambiente de laboratório, optaremos por utilizar uma função já existente. No campo "Função existente", escolha "LabRole".



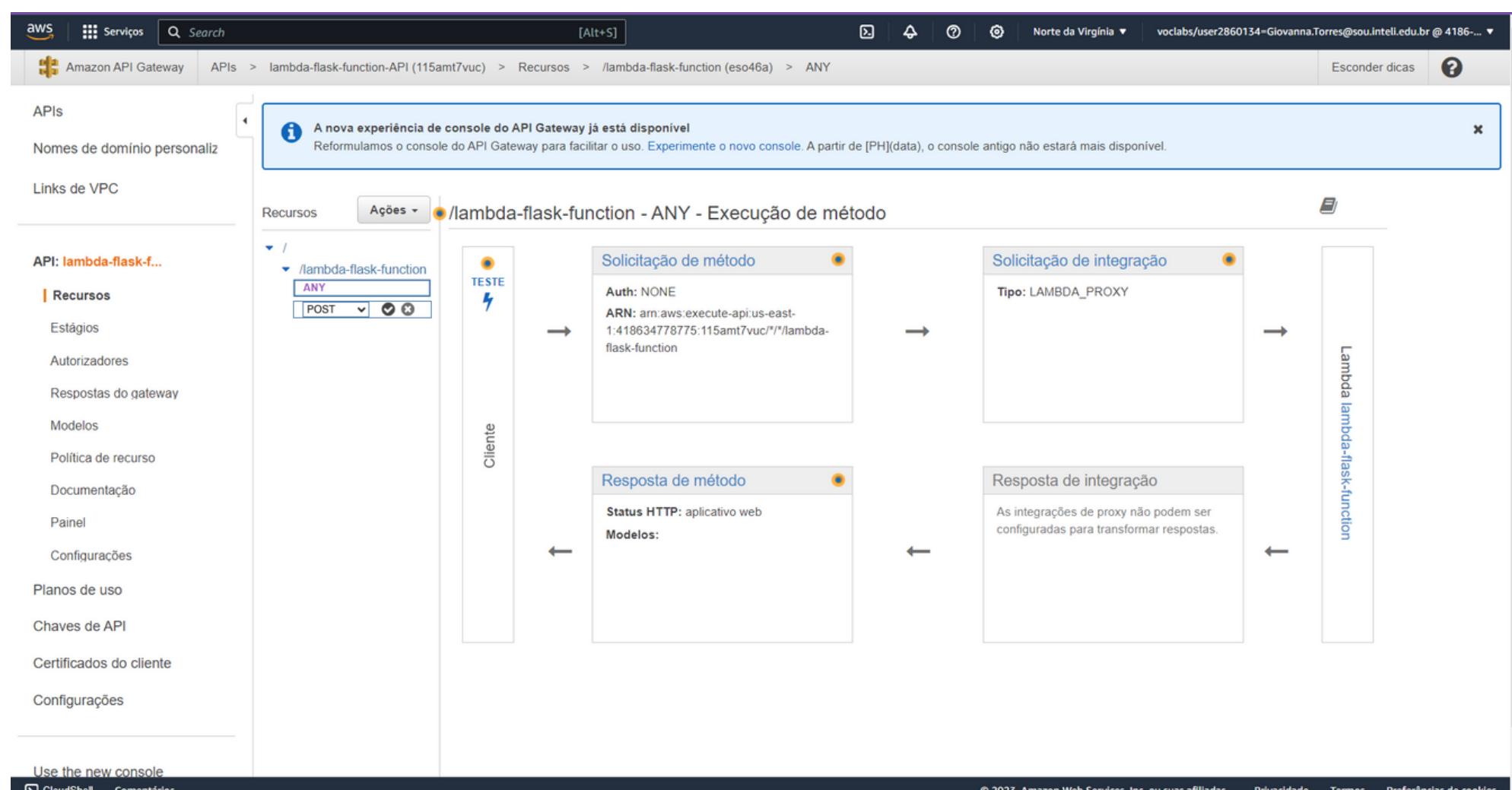
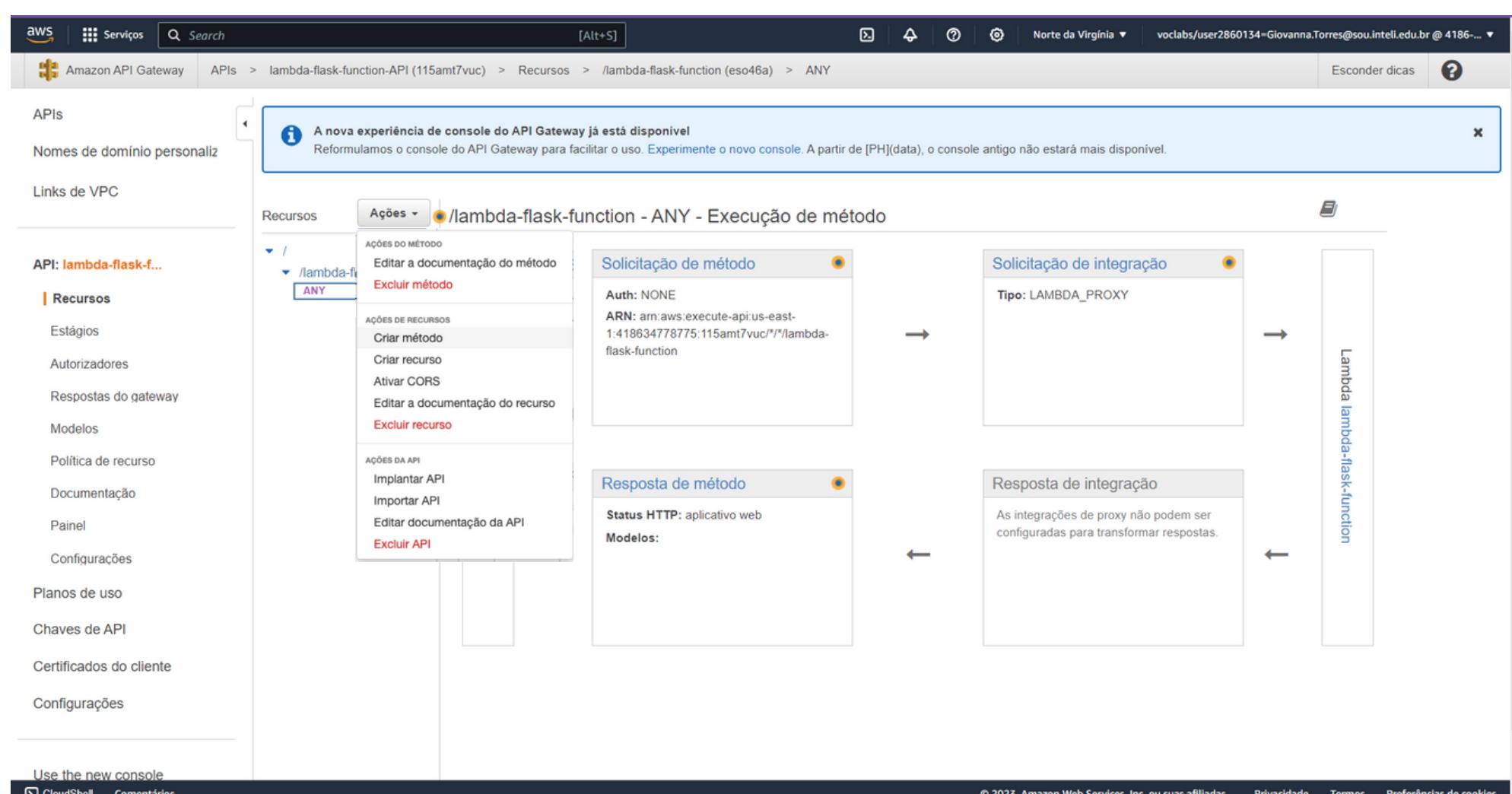
PASSO 4

Com o API Gateway criado, na tela inicial você pode visualizá-lo conectado à função Lambda. Clicando sobre essa conexão, você poderá configurar o método que deseja adicionar ao seu endpoint.



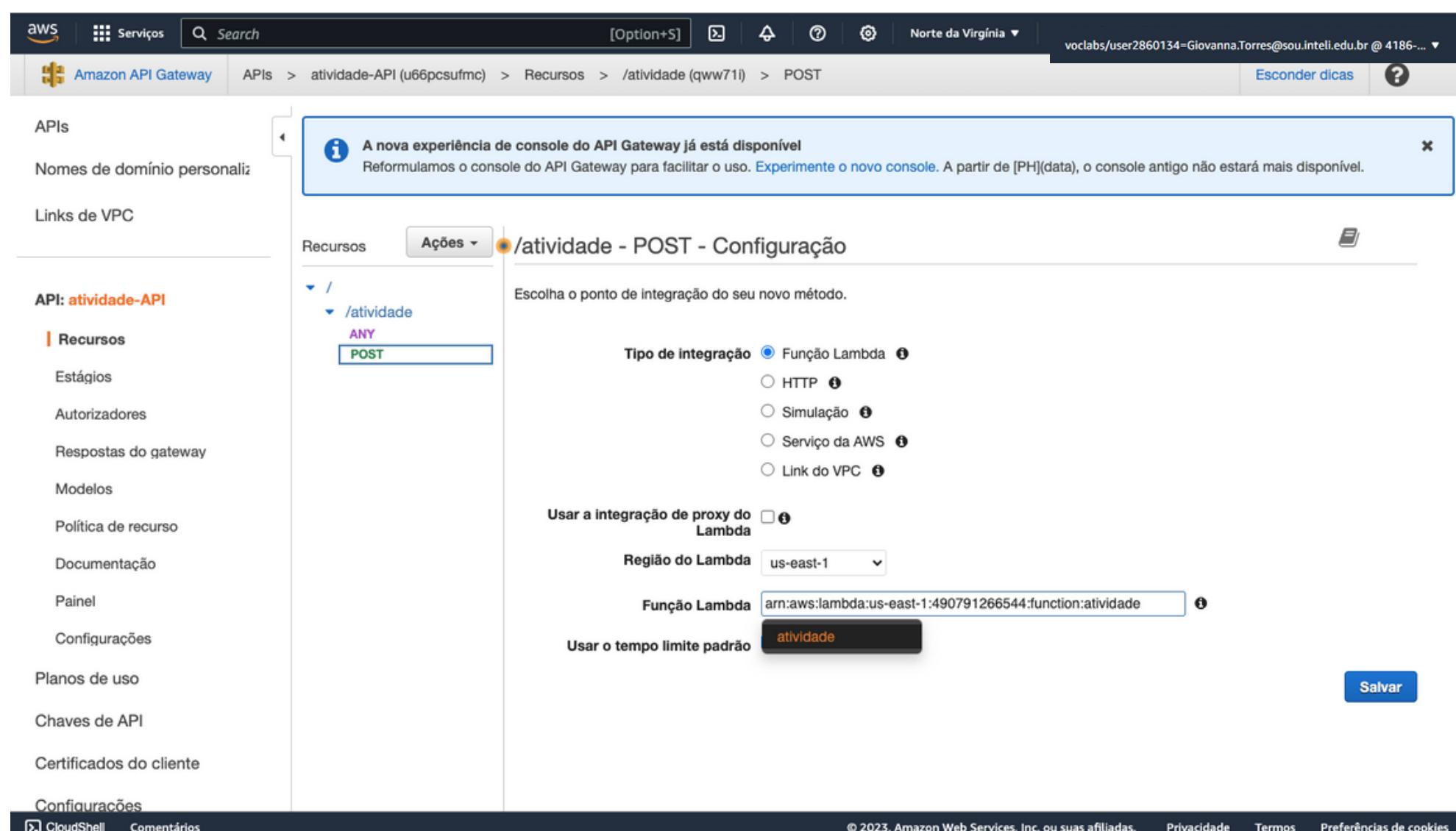
PASSO 5

Com o API Gateway criado, na tela inicial você pode visualizá-lo conectado à função Lambda. Clicando sobre essa conexão, você terá a oportunidade de configurar o método que deseja adicionar ao seu endpoint.



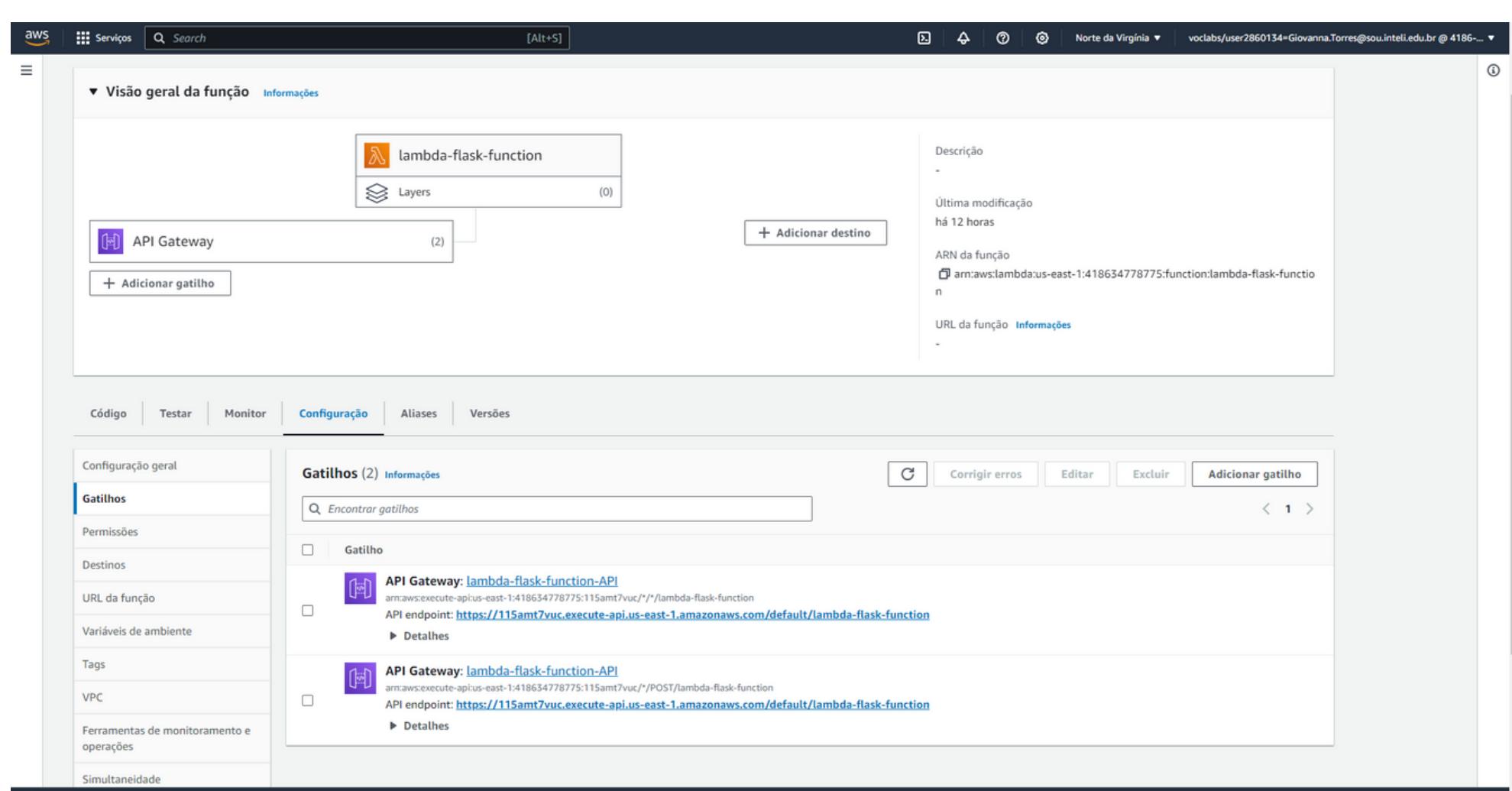
PASSO 6

Após selecionar o método "POST", configure o endpoint. Defina o tipo de integração como "Lambda". Na seção "Função Lambda", adicione a Amazon Resource Name (ARN) da sua função Lambda e clique em "Salvar" para confirmar as configurações.



The screenshot shows the AWS API Gateway configuration interface. On the left, there's a sidebar with various options like APIs, Recursos, Estágios, Autorizadores, etc. The main panel shows a tree structure for the API: / atividade - POST. Under 'Recursos', it lists / and /atividade. Under /atividade, it lists ANY and POST. The 'POST' method is selected. On the right, there's a configuration section for the POST method. It shows the 'Tipo de integração' (Integration Type) is set to 'Função Lambda' (Function Lambda). Below that, there are fields for 'Usar a integração de proxy do Lambda' (Use Lambda proxy integration), 'Região do Lambda' (Lambda Region) set to 'us-east-1', and 'Função Lambda' (Lambda Function) set to 'arn:aws:lambda:us-east-1:490791266544:function:atividade'. There's also a 'Usar o tempo limite padrão' (Use default timeout) checkbox. At the bottom right of the configuration panel is a blue 'Salvar' (Save) button. A blue banner at the top of the page says 'A nova experiência de console do API Gateway já está disponível' (The new API Gateway console experience is now available).

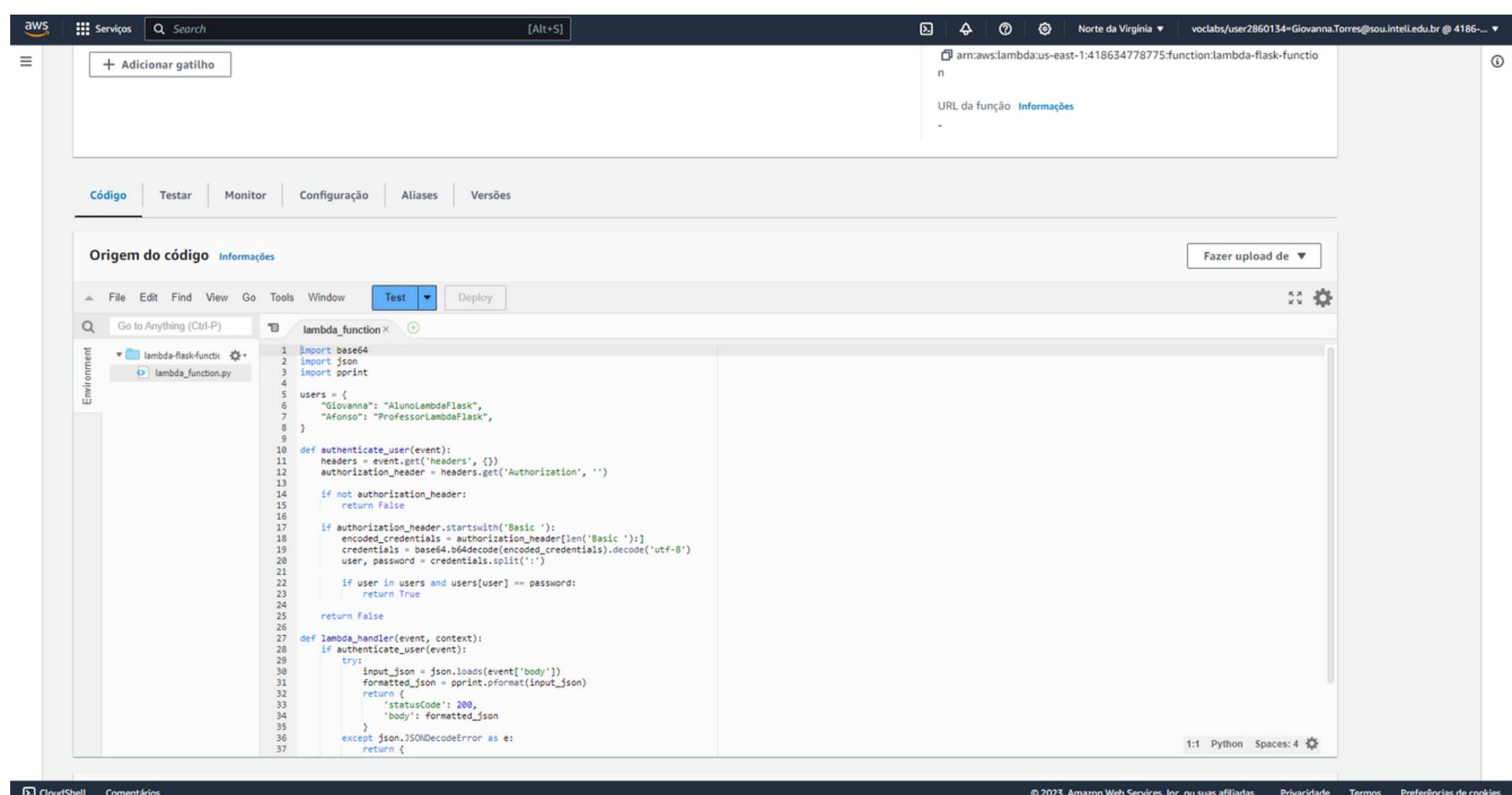
Após salvar as configurações, você poderá visualizar a URL que será usada para receber o JSON esperado. Essa URL é o endpoint que você criou e que agora está configurado para aceitar solicitações POST e acionar a função Lambda.



The screenshot shows the AWS Lambda function configuration interface. On the left, there's a sidebar with tabs like Código, Testar, Monitor, and Configuração. The main panel shows a 'Visão geral da função' (Function Overview) section with details about the function: 'lambda-flask-function', 'Layers (0)', 'API Gateway (2)', and a 'Descrição' (Description) field. To the right of this, there's a detailed view of the function's configuration. Under the 'Configuração' tab, the 'Gatilhos' (Triggers) section is selected, showing two entries: 'API Gateway: lambda-flask-function-API' and another identical entry. Both entries show the ARN 'arn:aws:lambda:us-east-1:418634778775:function:lambda-flask-function' and the API endpoint 'https://115amt7vuc.execute-api.us-east-1.amazonaws.com/default/lambda-flask-function'. There are buttons for 'Corrigir erros' (Fix errors), 'Editar' (Edit), 'Excluir' (Delete), and 'Adicionar gatilho' (Add trigger). At the bottom right of the configuration panel is a blue 'Salvar' (Save) button. The footer of the page includes links for 'CloudShell', 'Comentários', '© 2023, Amazon Web Services, Inc. ou suas afiliadas.', 'Privacidade', 'Termos', and 'Preferências de cookies'.

PASSO 7

Finalmente, na aba de código, escreva o código necessário para autenticar o usuário e para a função que receberá o corpo do JSON. Como no exemplo abaixo.



The screenshot shows the AWS Lambda function editor interface. The top navigation bar includes 'aws', 'Serviços', 'Search', and 'Norte da Virginia'. The main area has tabs for 'Código' (selected), 'Testar', 'Monitor', 'Configuração', 'Aliases', and 'Versões'. A sub-tab 'Origem do código' is selected under 'Código'. The code editor displays the following Python script:

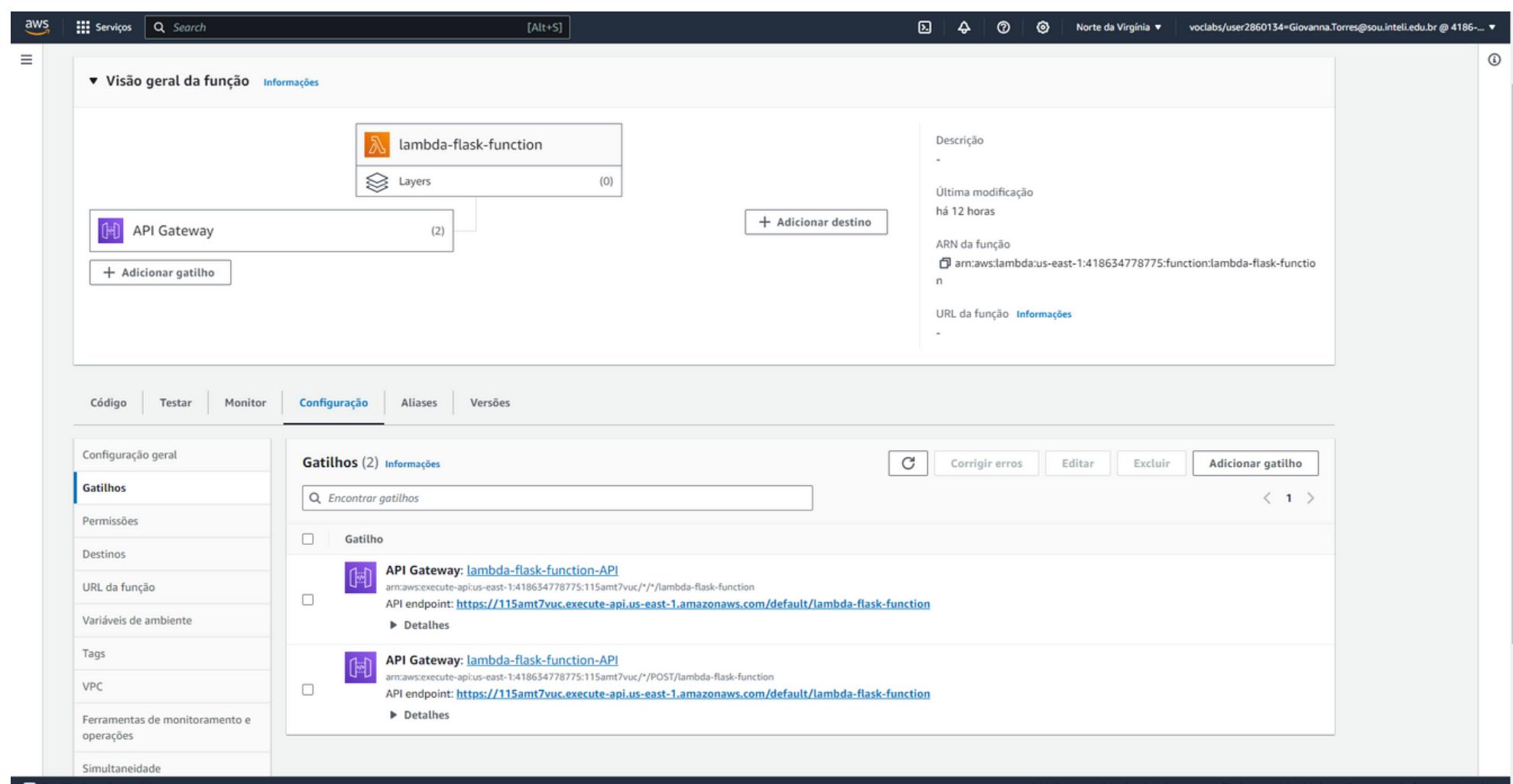
```
1 import base64
2 import json
3 import pprint
4
5 users = [
6     "Giovanna": "AlunoLambdaFlask",
7     "Afonso": "ProfessorLambdaFlask",
8 ]
9
10 def authenticate_user(event):
11     headers = event.get('Headers', {})
12     authorization_header = headers.get('Authorization', '')
13
14     if not authorization_header:
15         return False
16
17     if authorization_header.startswith('Basic '):
18         encoded_credentials = authorization_header[len('Basic '):]
19         credentials = base64.b64decode(encoded_credentials).decode('utf-8')
20         user, password = credentials.split(':')
21
22         if user in users and users[user] == password:
23             return True
24
25     return False
26
27 def lambda_handler(event, context):
28     if authenticate_user(event):
29         try:
30             input_json = json.loads(event['body'])
31             formatted_json = pprint.pformat(input_json)
32             return {
33                 'statusCode': 200,
34                 'body': formatted_json
35             }
36         except json.JSONDecodeError as e:
37             return {
```

02

TESTE
ENDPOINT

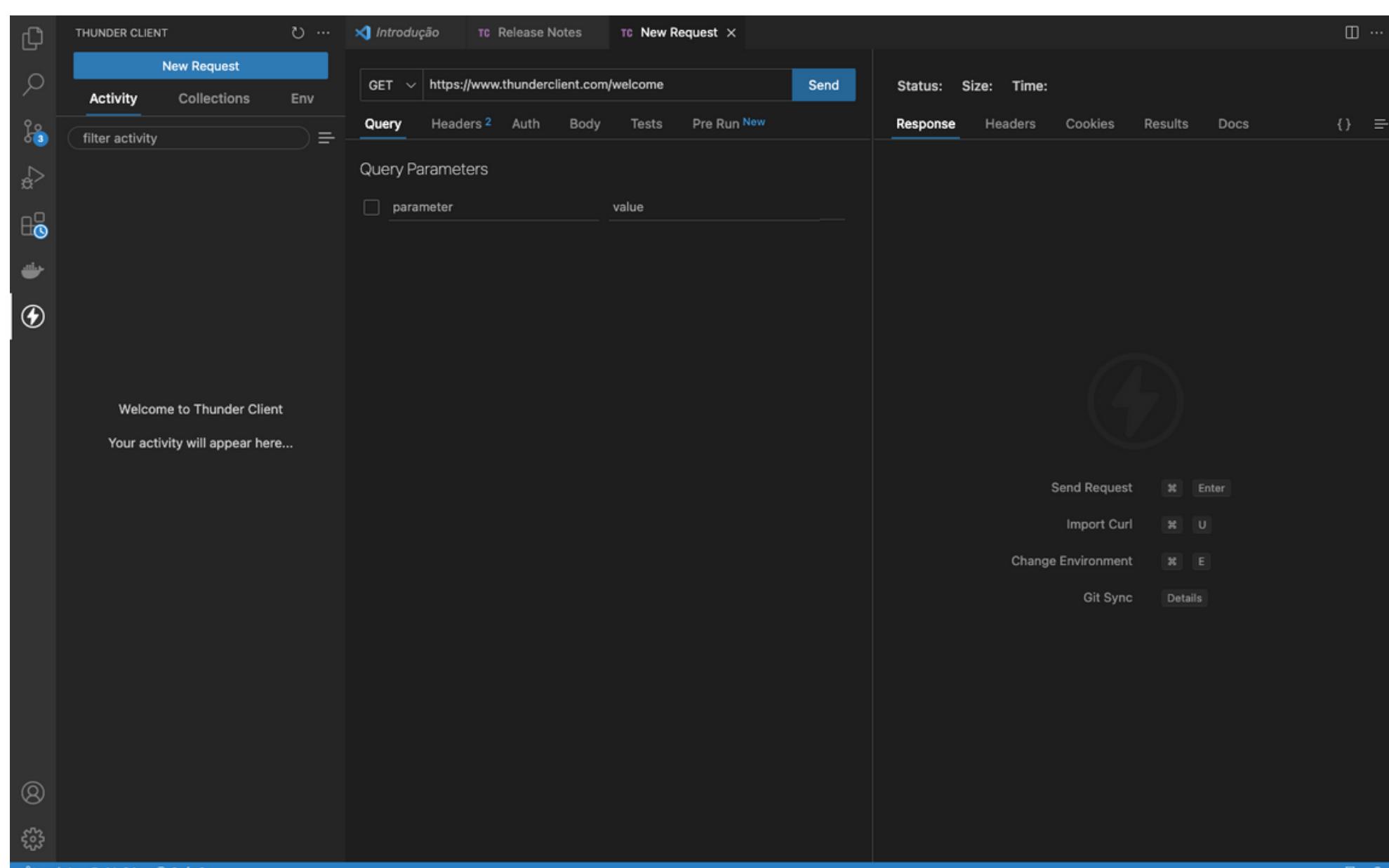
PASSO 1

Agora, para testar o endpoint, copie a URL da API POST. Em seguida, abra a extensão Thunder Client no Visual Studio Code.



The screenshot shows the AWS Lambda function configuration page for a function named "lambda-flask-function". The "Gatilhos" (Triggers) section displays two API Gateway triggers:

- API Gateway: lambda-flask-function-API**
arn:aws:execute-api:us-east-1:418634778775:115am7vuc/*/*/lambda-flask-function
API endpoint: <https://115am7vuc.execute-api.us-east-1.amazonaws.com/default/lambda-flask-function>
- API Gateway: lambda-flask-function-API**
arn:aws:execute-api:us-east-1:418634778775:115am7vuc/*/POST/lambda-flask-function
API endpoint: <https://115am7vuc.execute-api.us-east-1.amazonaws.com/default/lambda-flask-function>



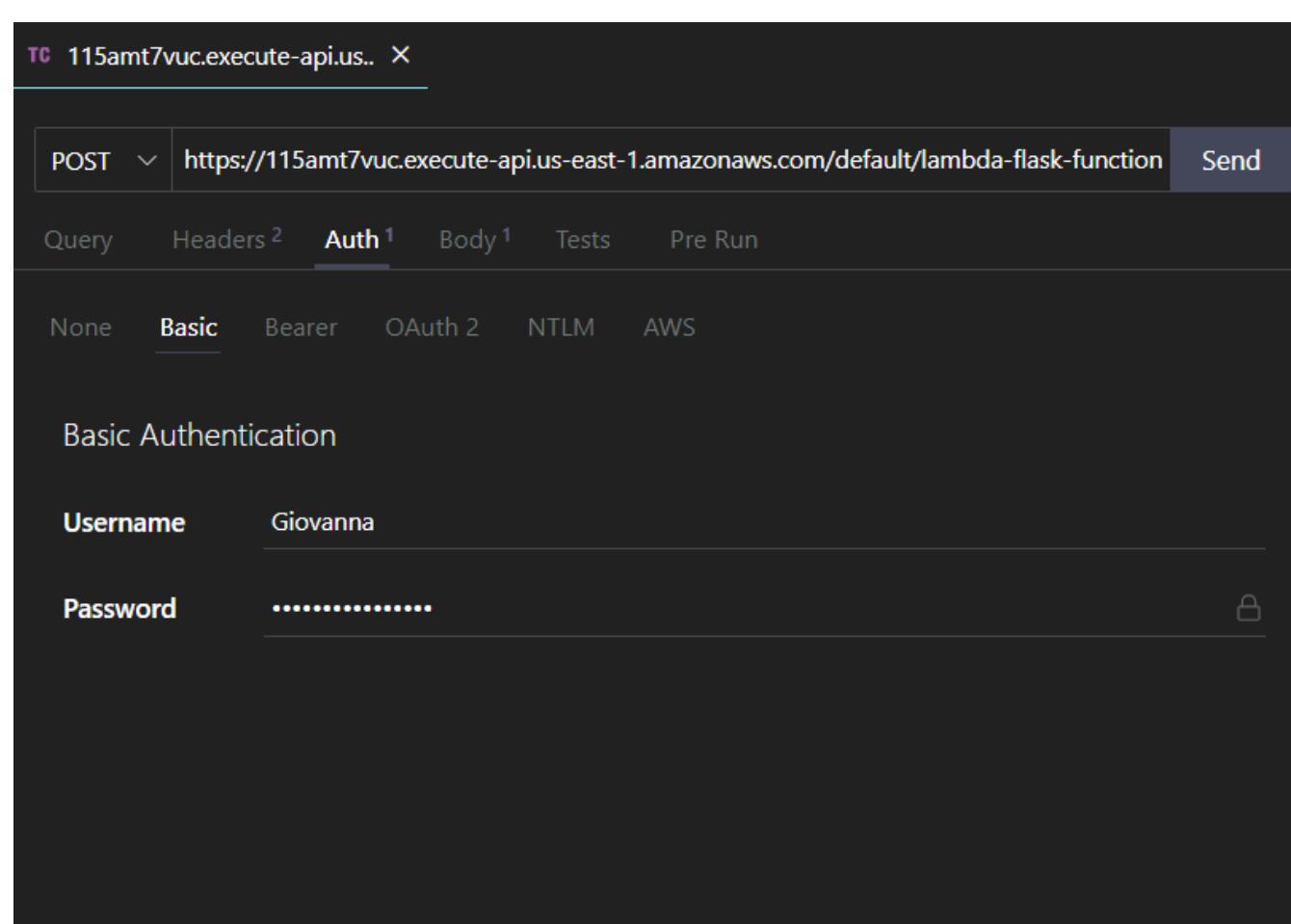
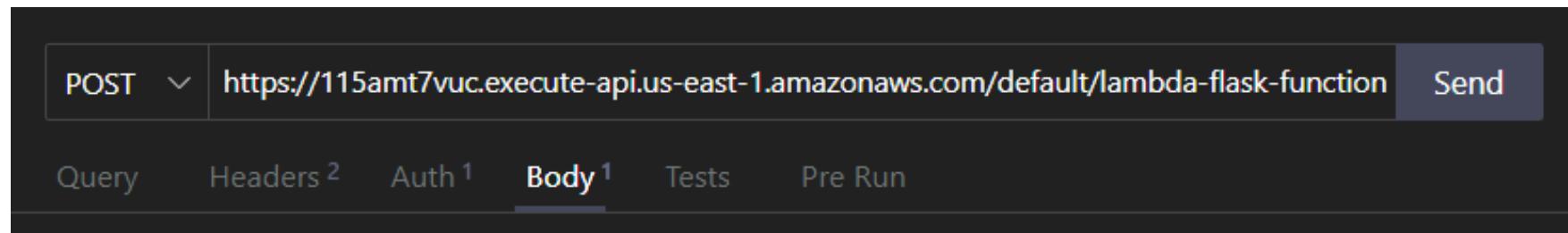
The screenshot shows the Thunder Client extension in Visual Studio Code. A new request is being prepared with the following details:

- Method:** GET
- URL:** <https://www.thunderclient.com/welcome>
- Headers:** 2
- Auth:** None
- Body:** None
- Tests:** None
- Pre Run:** None
- New:** Selected

The "Query Parameters" section is empty. On the right side, there are buttons for "Send Request", "Import Curl", "Change Environment", and "Git Sync".

PASSO 2

No primeiro passo, adicione a URL no campo apropriado e selecione o método POST. Em seguida, no campo "AUTH BASIC" insira o nome de usuário e a senha configurados no seu código Lambda em Python, conforme definido no tópico anterior.



Na aba "Body", como exemplo, você pode utilizar o JSON de Harry Potter, disponível no seguinte link: <https://github.com/Laboratoria/LIM011-data-lovers/blob/master/src/data/potter/potter.json>. No primeiro passo, adicione a URL no campo apropriado e selecione o método POST. Em seguida, no campo "AUTH BASIC," insira o nome de usuário e a senha configurados no seu código Lambda em Python, conforme definido no tópico anterior.. Em seguida, clique em "Send" para enviar a solicitação e verifique o resultado ao lado, que deve exibir um status 200 de sucesso. Isso confirmará que a integração entre o seu endpoint, a função Lambda e o API Gateway está funcionando corretamente.



lambda-flask-api

POST https://115amt7vuc.execute-api.us-east-1.amazonaws.com/default/lambda-flask-function

Status: 200 OK Size: 11.22 KB Time: 1.39 s

Response Headers Cookies Results Docs

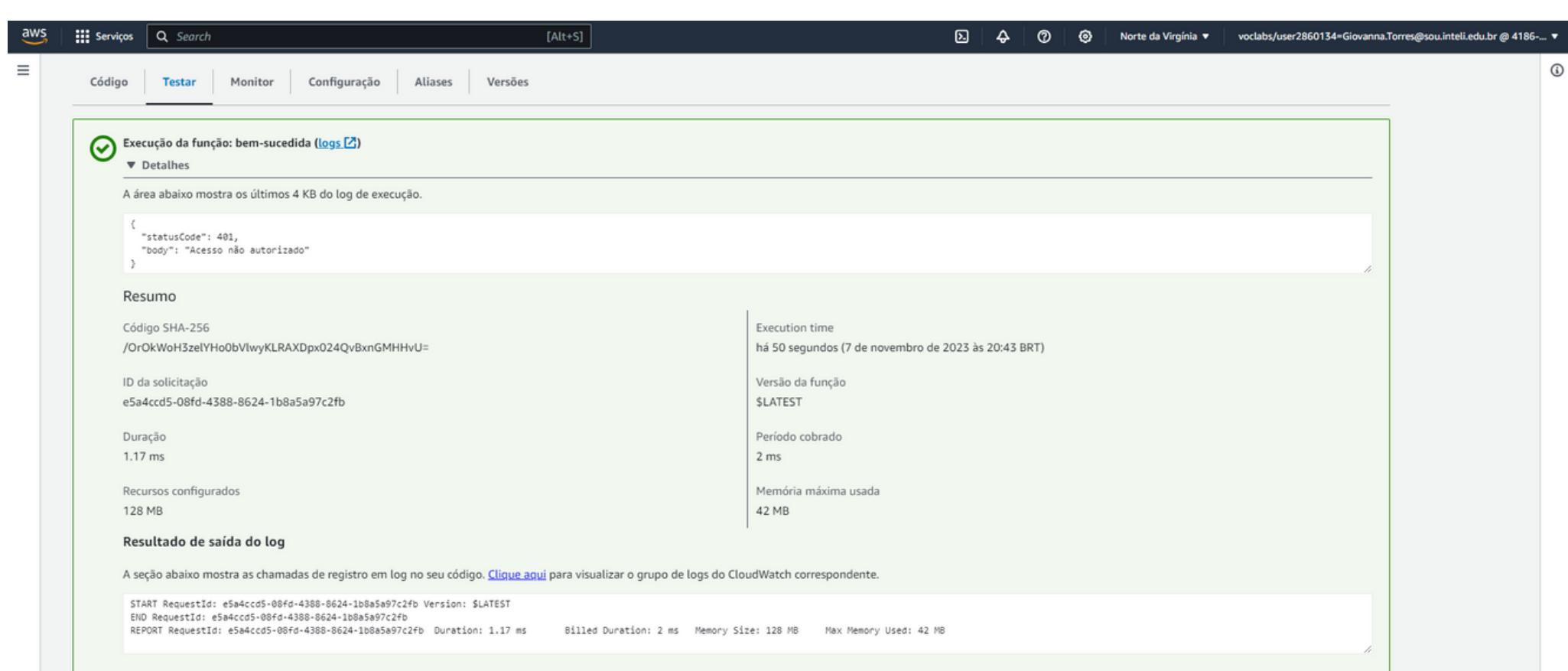
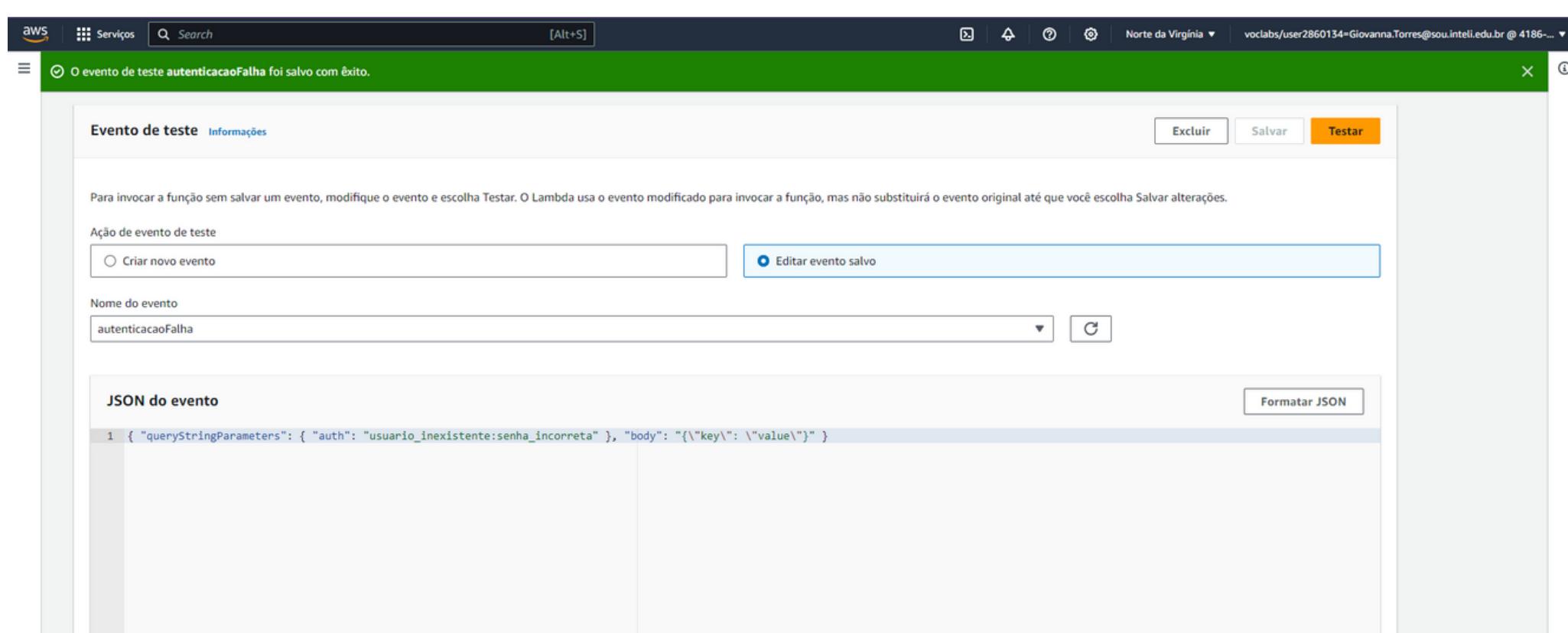
```
[{"actor": "Daniel Radcliffe", "alive": true, "ancestry": "half-blood", "dateOfBirth": "31-07-1980", "eyeColour": "green", "gender": "male", "hairColour": "black", "hogwartsStaff": false, "hogwartsStudent": true, "house": "Gryffindor", "image": "http://hp-api.herokuapp.com/images/harry.jpg", "name": "Harry Potter", "species": "human", "yearOfBirth": 1980, "ancestry": "half-blood", "eyeColour": "green", "hairColour": "black", "wand": {"core": "phoenix feather", "length": 11, "wood": "holly"}, "patronus": "stag", "hogwartsStudent": true, "hogwartsStaff": false, "actor": "Daniel Radcliffe", "alive": true, "image": "http://hp-api.herokuapp.com/images/harry.jpg"}, {"actor": "Emma Watson", "alive": true, "ancestry": "muggleborn", "dateOfBirth": "19-09-1979", "eyeColour": "brown", "gender": "female", "hairColour": "brown", "hogwartsStaff": false, "hogwartsStudent": true, "house": "Gryffindor", "image": "http://hp-api.herokuapp.com/images/hermione.jpeg", "name": "Hermione Granger", "species": "human", "yearOfBirth": 1979, "ancestry": "muggleborn", "eyeColour": "blue", "wand": {"core": "dragon heartstring", "length": 13, "wood": "vine"}, "patronus": "otter", "actor": "Rupert Grint", "alive": true, "image": "http://hp-api.herokuapp.com/images/hermione.jpeg"}]
```

03

TESTE UNITÁRIO

PASSO 1

Para realizar o teste do código Lambda, na guia "Testar", selecione a opção "Criar Novo Evento", dê um nome ao evento e adicione o JSON de teste. Neste primeiro caso, você pode criar um evento que simule uma autenticação inválida, com os dados de autenticação errados. Após preencher todos os campos, clique em "Testar". Se o teste for concluído com sucesso e você recebe um resultado de sucesso, isso indica que a função Lambda está respondendo corretamente à autenticação inválida, impedindo o login do usuário.



PASSO 2

No segundo teste, você está simulando o recebimento de um JSON com a autenticação vazia e o corpo também vazio, o que resulta em acesso negado.

Para criar esse teste, siga o mesmo processo explicado no passo anterior, criando um novo evento de teste com os parâmetros apropriados. Quando você testar essa configuração, espera-se que o resultado seja "Acesso Negado", demonstrando que a função Lambda está tratando a solicitação adequadamente.

