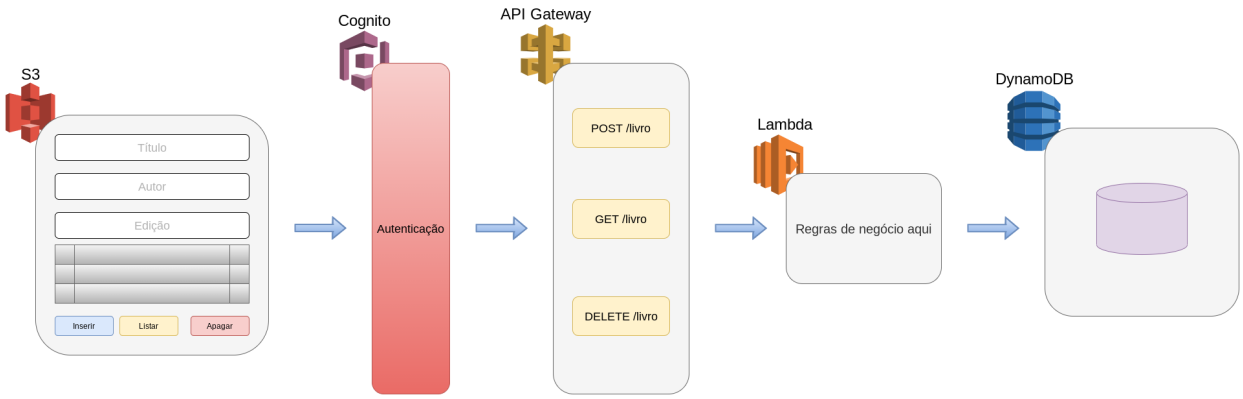
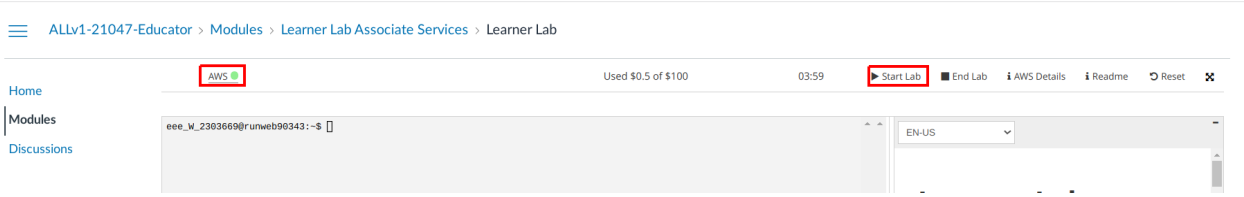
1 Introdução

Prosseguimos com a implementação da aplicação retratada a seguir.

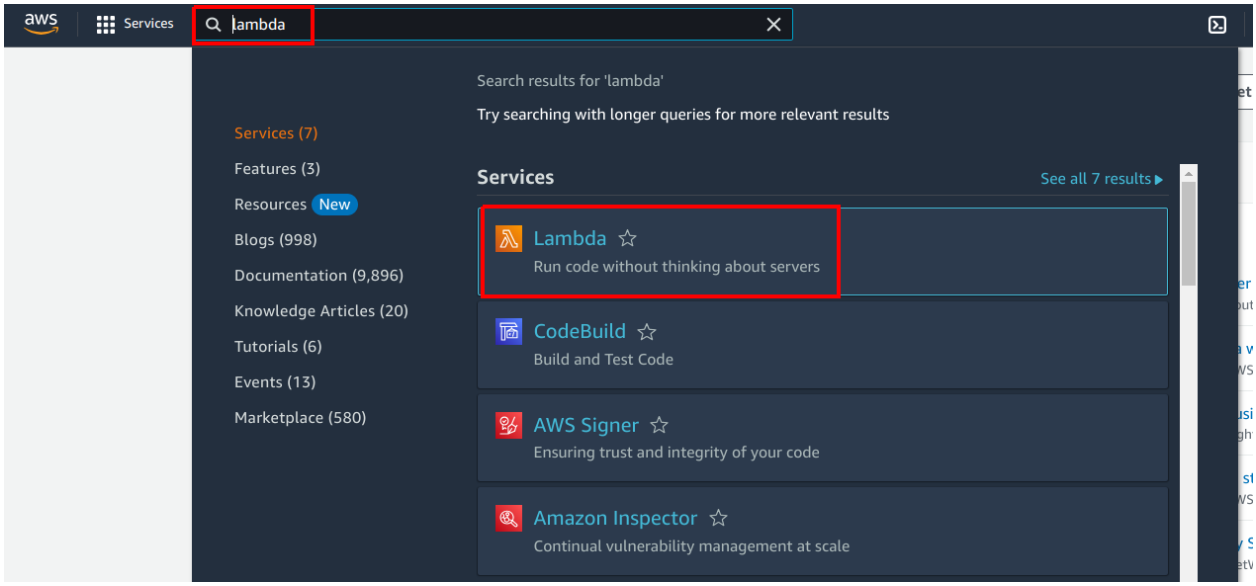


(Criando uma função Lambda) O próximo passo é criar uma função Lambda que nos servirá de Back End. Sua existência nos permitirá implementar nossas regras de negócio. Visite seu Learner Lab, inicie o Lab e clique em AWS quando a bolinha ficar verde.

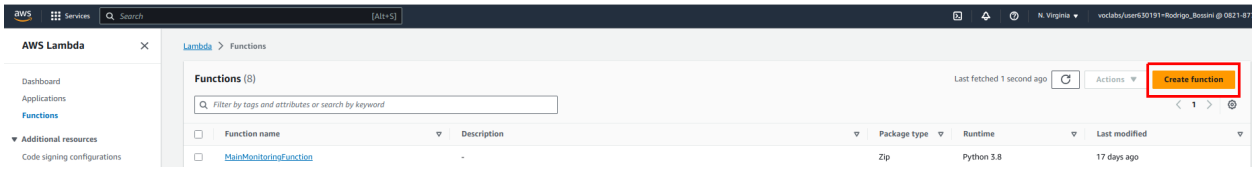


No console AWS, busque pelo serviço Lambda e clique em seu ícone.

Autoria: Professor Rodrigo Bossini<professorbossini@gmail.com>
Produção: agosto de 2023



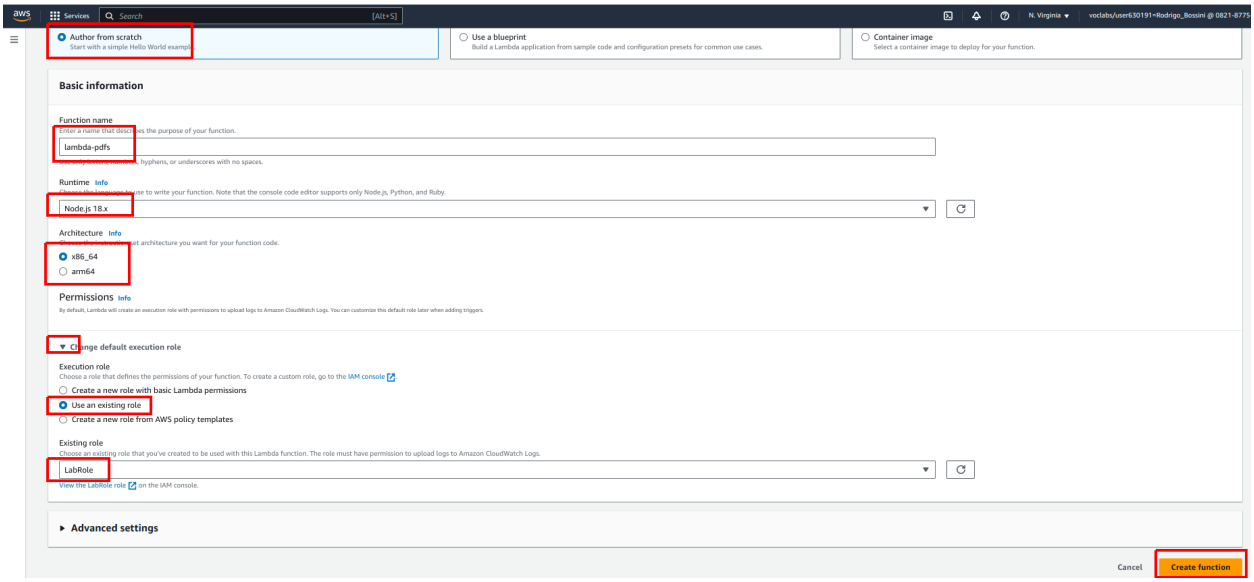
Clique em **Create Function.**



Preencha os campos como a seguir e clique em **Create Function**.

> **Nota**. **LabRole** é uma Role previamente criada no ambiente Learner Lab. As permissões associadas a ela são suficientes para aquilo que pretendemos implementar.

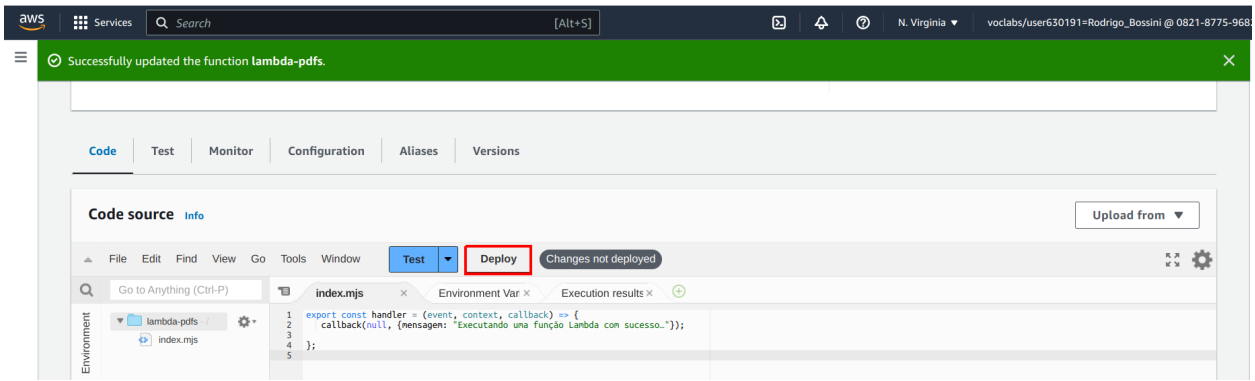
Autoria: Professor Rodrigo Bossini<professorbossini@gmail.com>
Produção: agosto de 2023



Na tela resultante, ajuste o código inicial como a seguir. **callback** é o nome de uma função que é colocada em execução automaticamente quando a função lambda é chamada. Nossa implementação será feita nesta função. Aprenderemos mais sobre os outros objetos em breve.

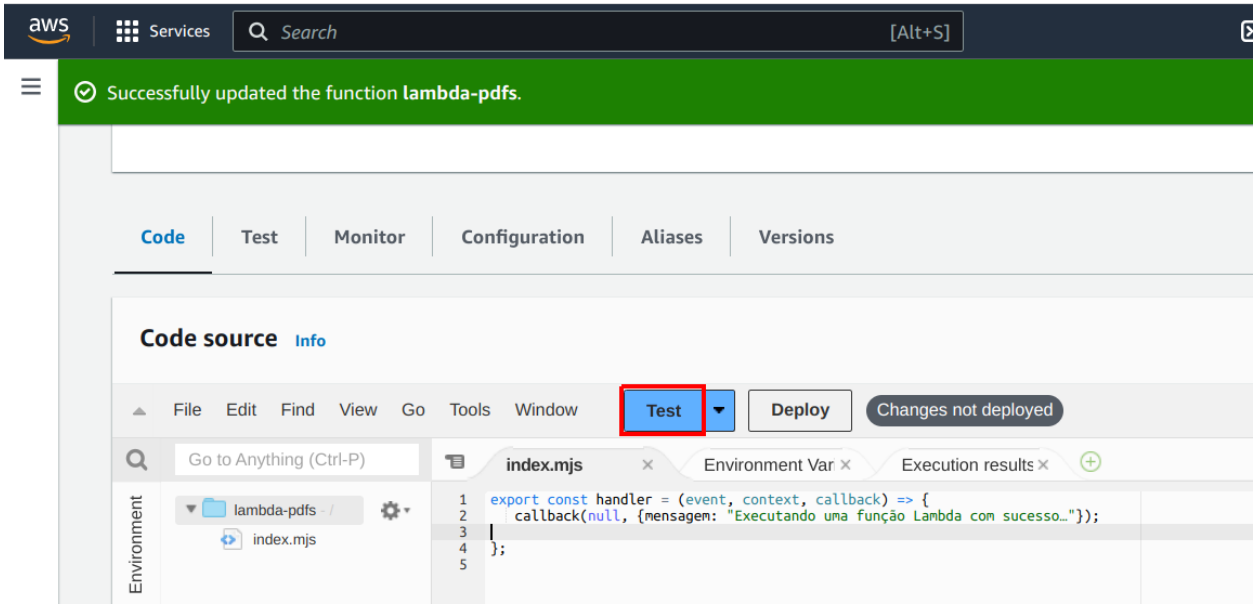
```
export const handler = (event, context, callback) => {
  callback(null, {mensagem: "Executando uma função Lambda com sucesso…"});

};
```

Clique em **Deploy**.



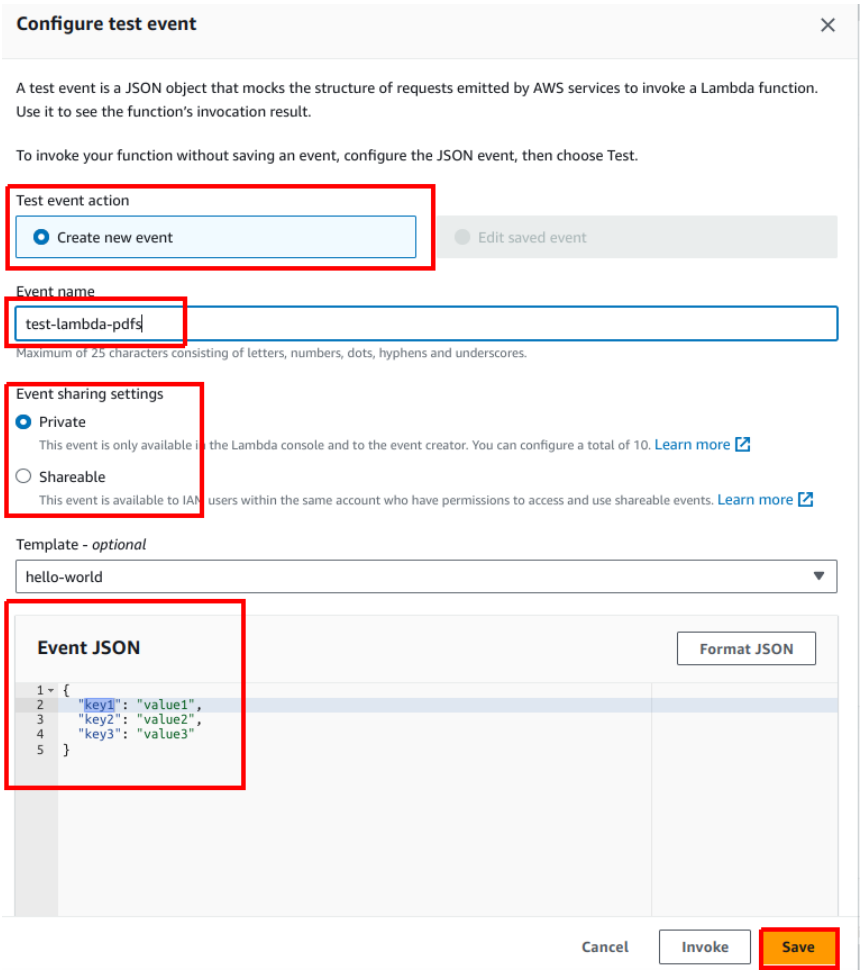
Autoria: Professor Rodrigo Bossini<professorbossini@gmail.com>
Produção: agosto de 2023

Podemos testar a função Lambda criando um evento de teste “Mock”. Para tal, clique em **Test**.



Escolha a opção **Create new Event**. Dê um nome para o seu teste. Mantenha as demais opções com seu valor padrão e clique em **Save**.

Autoria: Professor Rodrigo Bossini<professorbossini@gmail.com>
Produção: agosto de 2023



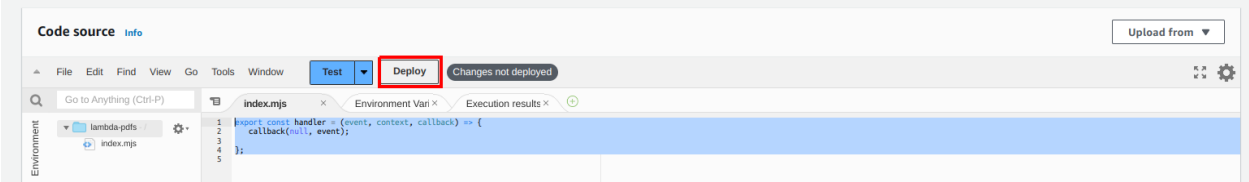
Clique em **Test** para obter o resultado a seguir.

Autoria: Professor Rodrigo Bossini<professorbossini@gmail.com>
Produção: agosto de 2023

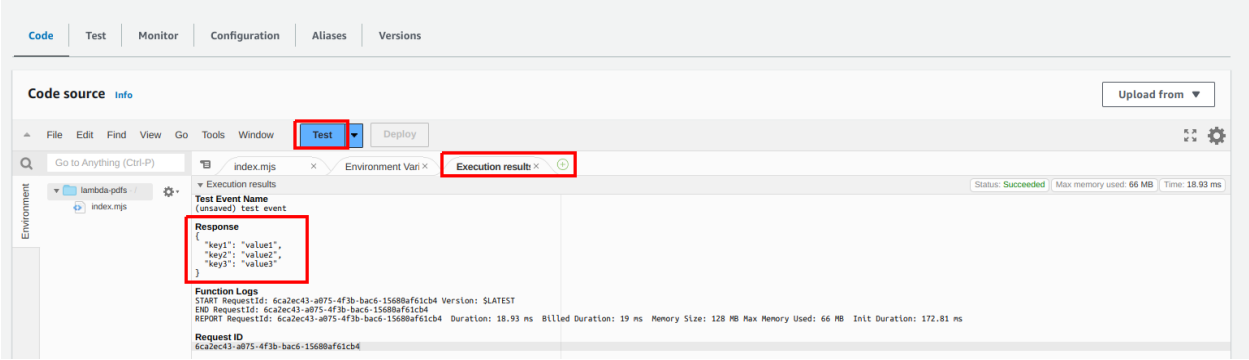
Para visualizar o objeto de teste especificado (o json com três pares/chave valor key: value), volte ao arquivo **index.mjs** e ajuste o código como a seguir. Observe que a nossa função agora devolve o objeto **event**.

```
export const handler = (event, context, callback) => {
  callback(null, event);

};
```

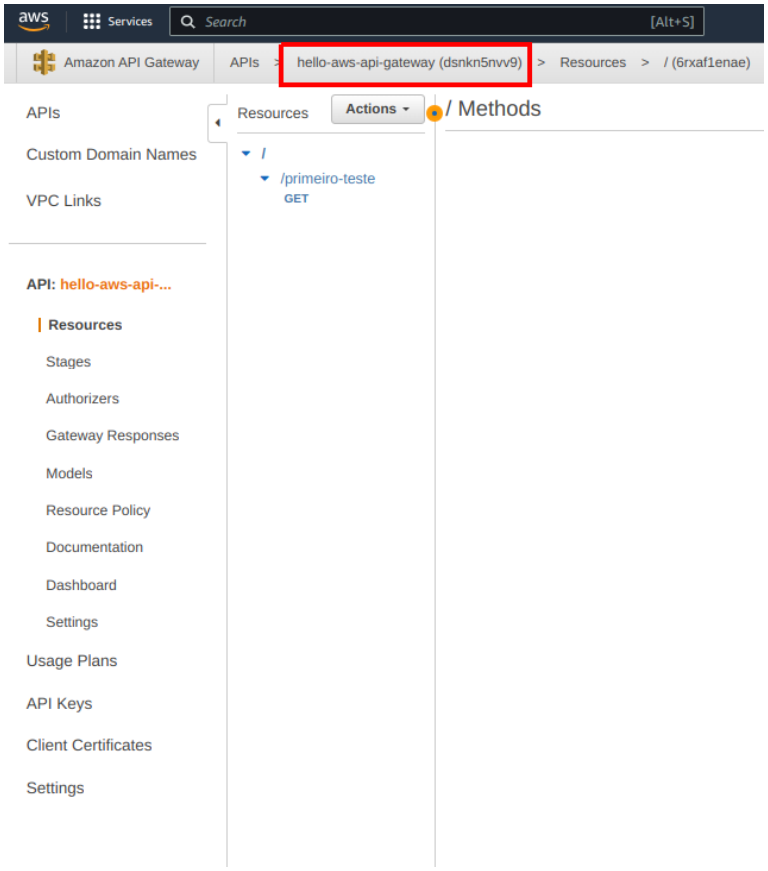
Clique em **Deploy** a seguir.



Depois, clique em **Test**. Veja o resultado esperado.



(Mais funcionalidades do API Gateway e nossa nova API) Como vimos, o serviço AWS API Gateway oferece diferentes funcionalidades. Nesta seção estudaremos as principais. Depois disso, criaremos a API da solução computacional que estamos implementando. Certifique-se de que você está na página inicial da sua API. Para tal, você pode clicar no nome dela.



(API Keys) Essa feature nos permite criar chaves de acesso para nossa API. Tratam-se de chaves que poderíamos entregar para nossos clientes e restringir o acesso aos serviços somente àqueles que enviem uma chave válida. Além disso, elas podem ser usadas para garantir que um usuário não ultrapasse os limites de uso impostos pelo plano que tiver escolhido. À esquerda, clique **API Keys** e então clique **Actions >> Create API Key**. Dê um nome qualquer para a chave, talvez **teste**, e escolha **Auto Generate**. Na tela a seguir, clique **Show** para ver a chave gerada.

- **(Custom Domain Name)** Como o nome sugere, com esta opção podemos escolher um domínio personalizado para nossa api, como **api.meudominio.com**.

- **(Client Certificates**) Aqui podemos criar um certificado que pode ser enviado para outros serviços a fim de garantir a autenticidade das requisições. Ou seja, para garantir que elas estão sendo, de fato, originadas a partir do API Gateway.

- **Authorizers** Aqui podemos utilizar um serviço de autenticação (Cognito ou mesmo uma função lambda) para restringir à API.

- **Models**: Aqui podemos definir a estrutura esperada dos dados recebidos na requisição.

- **Documentation**: Permite que especifiquemos documentações para os endpoints de nossa API.

- Dashboard: Informações sobre uso.

(Manipulação de requisição e resposta) Como vimos, ao especificar um método para um endpoint, temos quatro quadros. Veja a finalidade de cada um a seguir. Para visualizar os quadros, certifique-se de selecionar o método (GET, neste exemplo que criamos) no menu Resources.

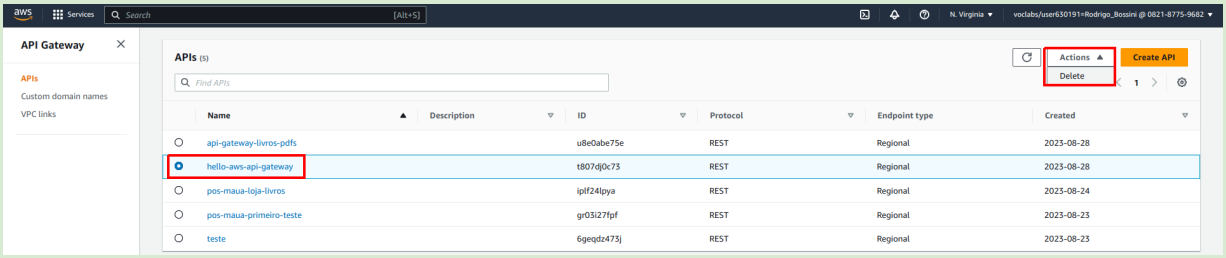
- Method Request: Essa fase do fluxo pode filtrar requisições e bloquear aquelas que não estiverem de acordo com critérios pré definidos. Por exemplo, aqui especificamos se uma chave de API é requerida, se parâmetros na URL são requeridos (e se sim, quais são) entre outras coisas.

- Integration Request: Caso a requisição passe pela fase **Method Request**, ela é entregue para a fase Integration Request. Nesta fase, podemos realizar transformações em geral envolvendo os dados recebidos na requisição para entregar para o componente responsável por executar nossas regras de negócio. Por exemplo, podemos extrair os parâmetros de uma requisição GET e montar um objeto JSON para que ele possa ser entregue para uma função Lambda.

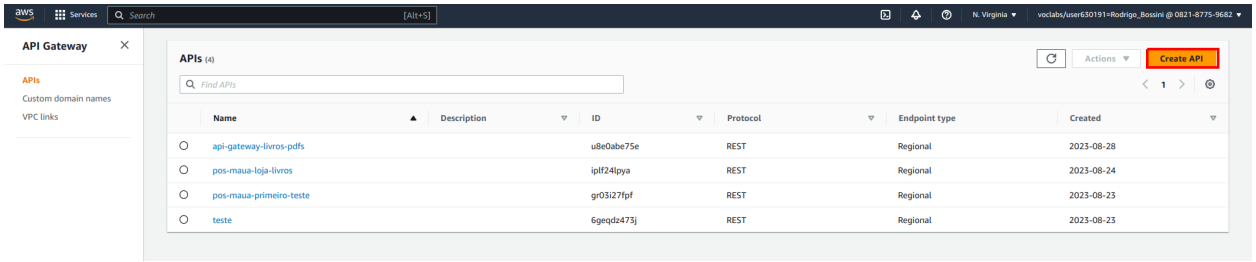
- Integration Response Esta fase entra em cena uma vez que a requisição tenha sido processada. Aqui podemos, por exemplo, **mapear códigos de status vindos do Back End para código que desejamos entregar para o cliente**. Por exemplo, verificar se o código de resposta do back end está de acordo com algum padrão (2xx, por exemplo) e mapeá-lo para um código que faça sentido para a nossa API. **Também podemos adicionar pares chave/valor ao cabeçalho da resposta.** Também é possível aplicar transformações aos dados recebidos do back end antes de entregá-los ao cliente.

- Method Response: Esta fase define o que é público para o cliente. Por exemplo, quais headers, quais códigos, quais MIME types o endpoint pode devolver. É importante entender que nesta fase são feitas as **definições das características da resposta. Somente seu formato**. O preenchimento com os dados de fato ocorre na fase **Integration Response**.

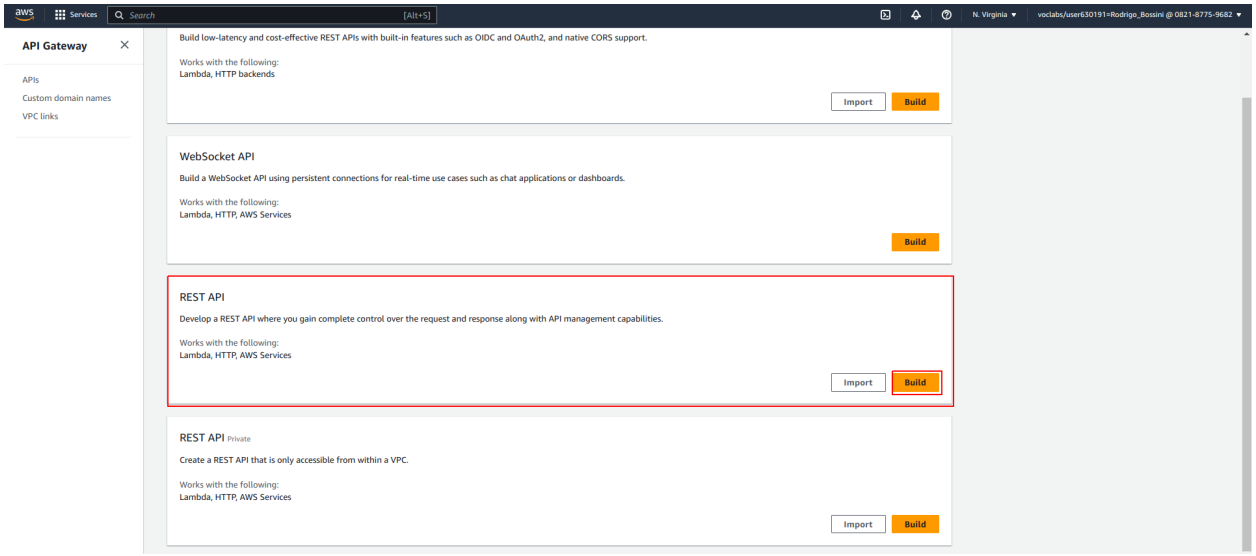
Nota: Agora vamos criar uma nova API. Caso queira apagar a API de teste que criamos, visite a página do API Gateway, selecione a API e clique em **Actions >> Delete**. Para encontrar a página inicial, você pode sempre buscar pelo nome do serviço - API Gateway - no Console AWS.



(Criando uma nova API) Na página principal do API Gateway, clique em **Create API.**



Clique na opção **Build** de **REST API** (não private).



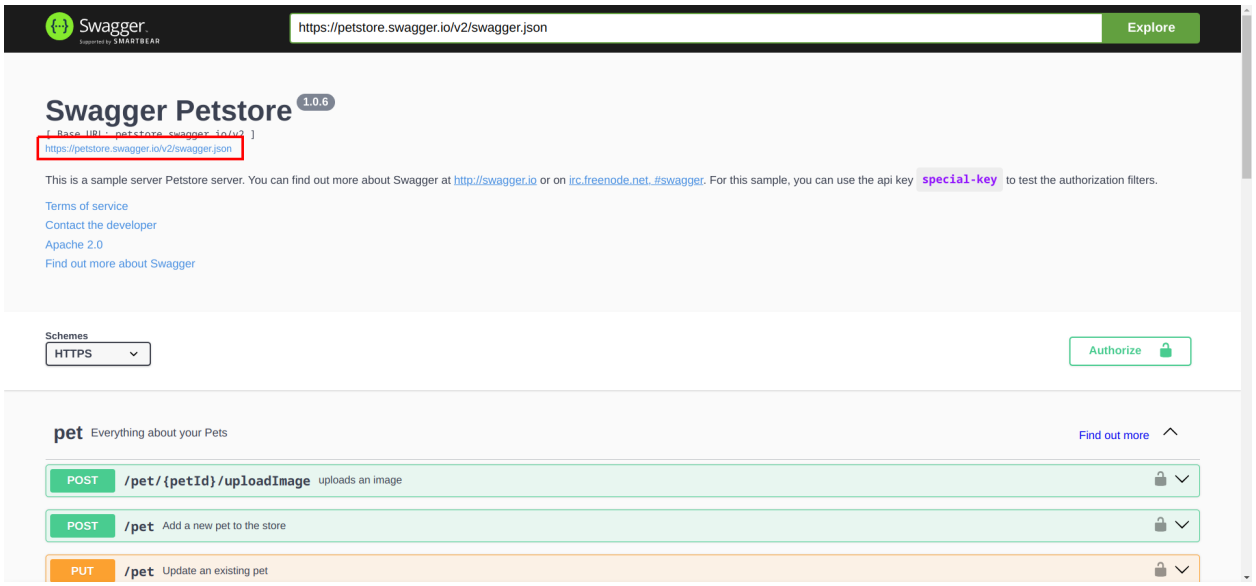
A tela seguinte mostra quatro opções.

- New API: Já utilizamos uma vez e vamos utilizar novamente. Usada para criar uma API "do zero".

- Clone from existing API: Permite fazer uma cópia de uma API existente.

- Import from Swagger: Permite importar um arquivo textual que descreve uma API criada pelo Swagger. Veja um exemplo a seguir. Basta clicar no pequeno link com extensão json nesta página.

https://petstore.swagger.io/



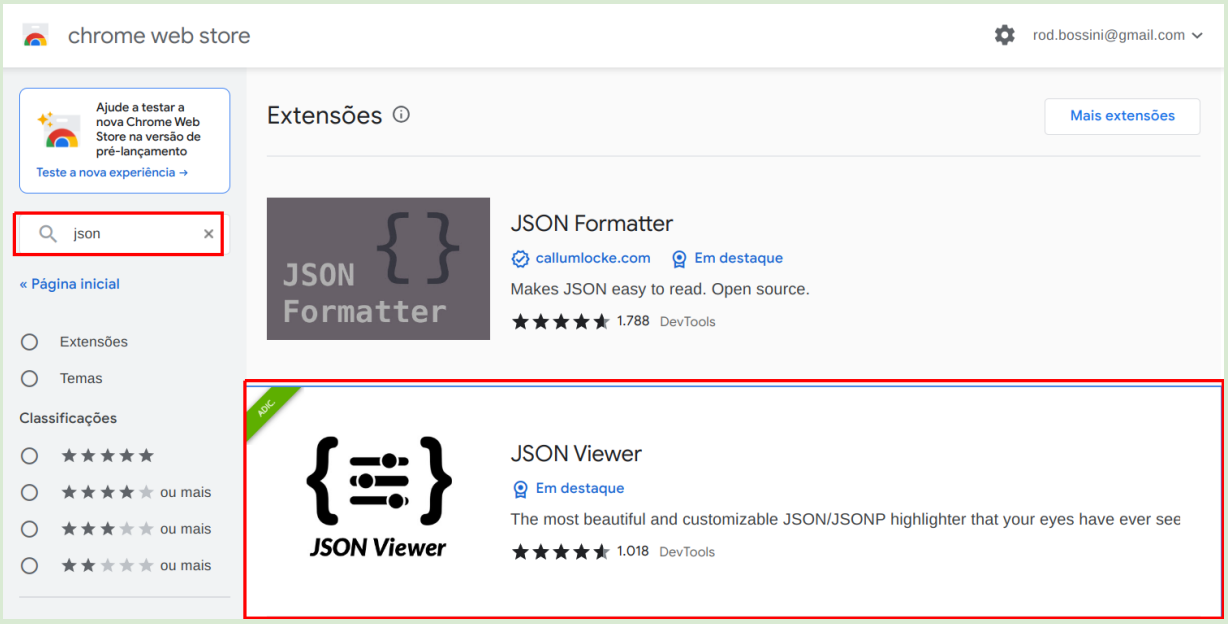
Lembre-se que esse é apenas um exemplo usando Swagger, ferramenta que não estamos utilizando neste material.

Autoria: Professor Rodrigo Bossini<professorbossini@gmail.com>
Produção: agosto de 2023

Nota. É possível que o seu objeto JSON não seja exibido formatado como na figura. Para que essa formatação seja realizada, você pode instalar uma extensão no Google Chrome. Comece visitando a Chrome Web Store.

https://chrome.google.com/webstore/

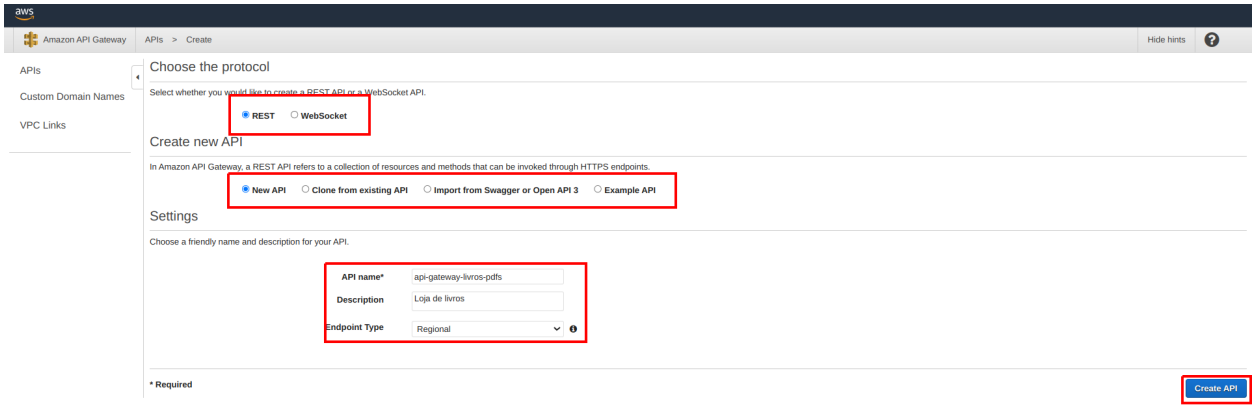
Busque por JSON e instale a extensão JSON Viewer.



- **Example API**: Mostra uma API de exemplo em que você pode se basear.

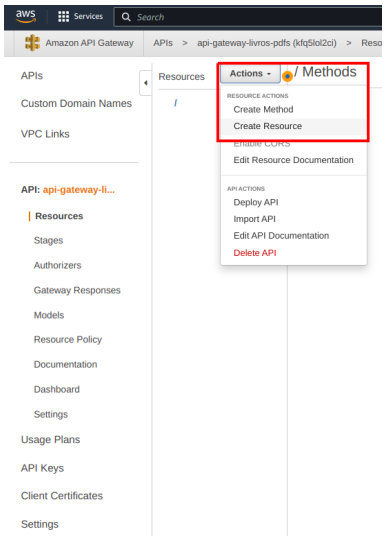
Escolha **New API**. Preencha os campos como a seguir.

API name: loja-livros
Description: Loja de livros
Endpoint Type: Regional (otimizado para funcionar dentro da região escolhida. Edge Optimized faria com que ele fosse distribuído em regiões diferentes usando o CloudFront, visando reduzir a latência).



(Adicionando o recurso /livros, CORS e OPTIONS) O próximo passo é adicionar os recursos da API.

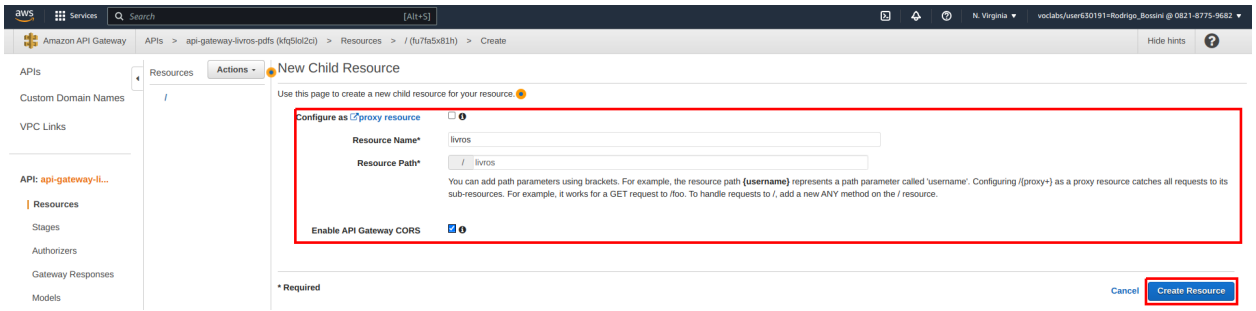
- Clique **Actions >> Create Resource**.



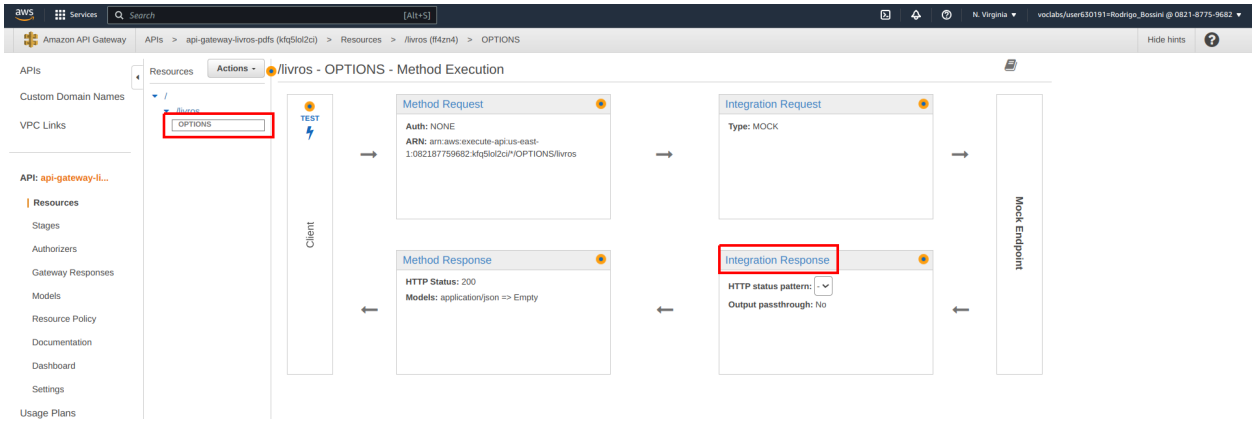
Autoria: Professor Rodrigo Bossini<professorbossini@gmail.com>
Produção: agosto de 2023

Preencha os campos como a seguir e clique em **Create Resource**.

Resource Name: livros
Resource path: livros (name e path não precisam ser iguais mas pode simplificar)
Proxy Resource: não marcar
Enable API Gateway CORS: marcar

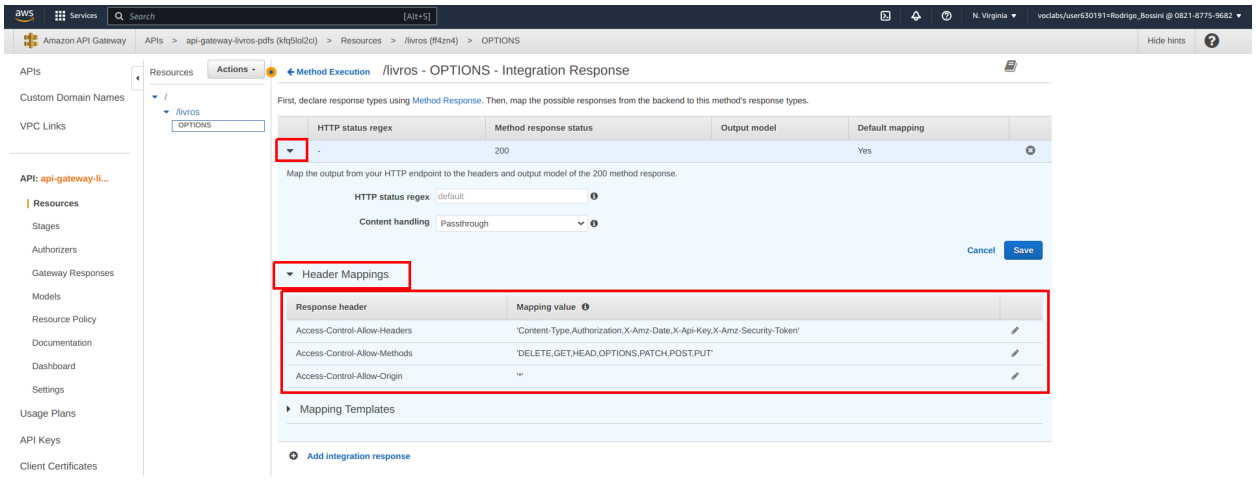


A configuração do mecanismo CORS é feita no cabeçalho da resposta. Podemos verificar mantendo o método OPTIONS selecionado e clicando em **Integration Response**.



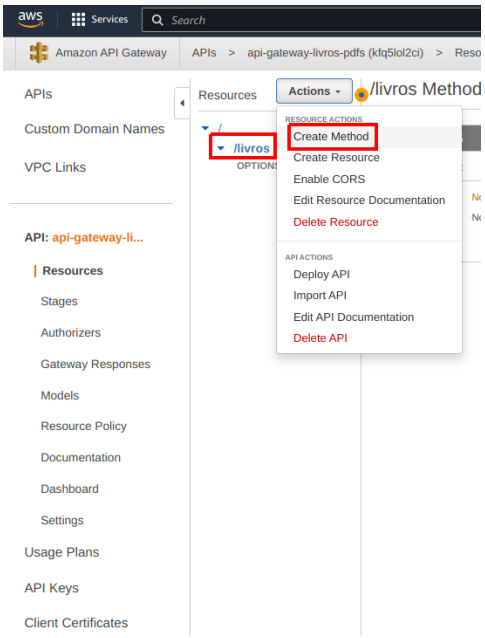
Autoria: Professor Rodrigo Bossini<professorbossini@gmail.com>
Produção: agosto de 2023

Clique na pequena seta para expandir e, então, clique em **Header Mappings**. Veja, ali, os pares chave/valor de configuração do mecanismo CORS.



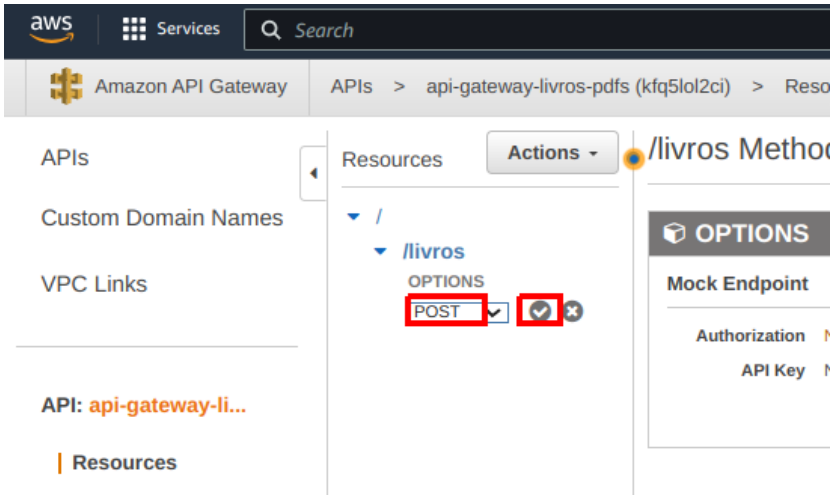
Apenas observe essas configurações. Não há nada a fazer nesta área, por enquanto.

(Armazenamento de um livro com o método POST e uma função Lambda) POST é o método do protocolo HTTP apropriado para realizar operações de criação de conteúdo. Façamos a sua criação, associada ao recurso **livros**, já criado. **Certifique-se de que o endpoint /livros está selecionado** (e não a raiz /) e clique **Actions >> Create Method.**

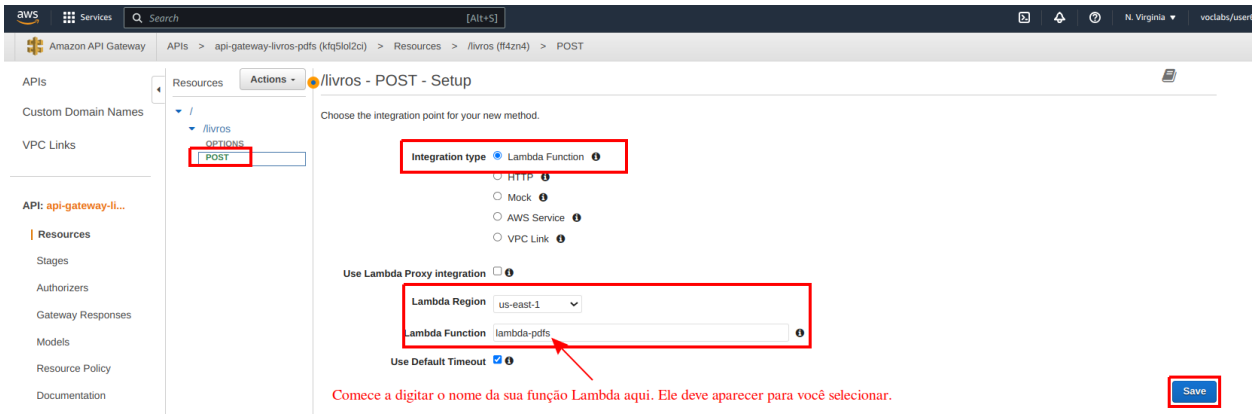


Autoria: Professor Rodrigo Bossini<professorbossini@gmail.com>
Produção: agosto de 2023

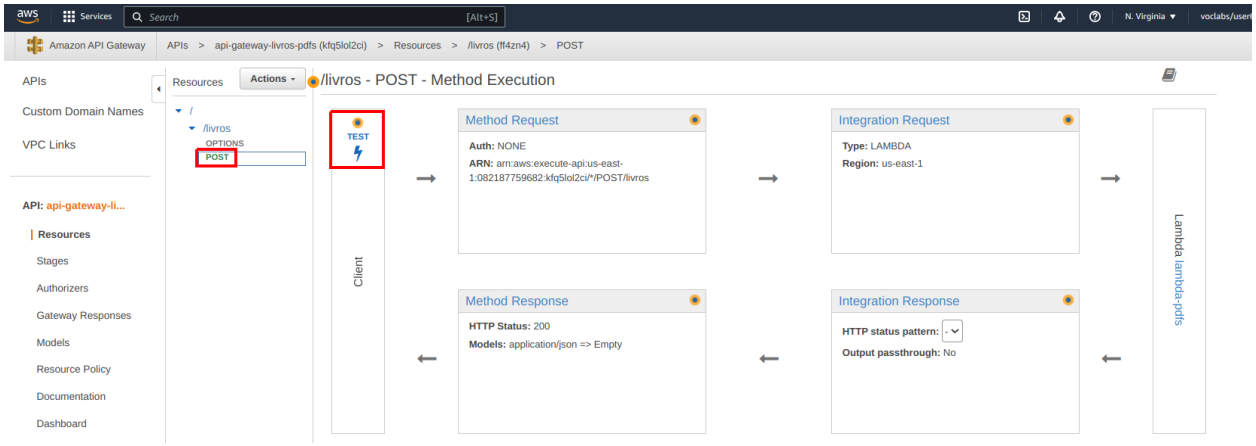
Escolha o método **POST** no pequeno menu que aparecerá e clique no botão logo à frente dele para confirmar.



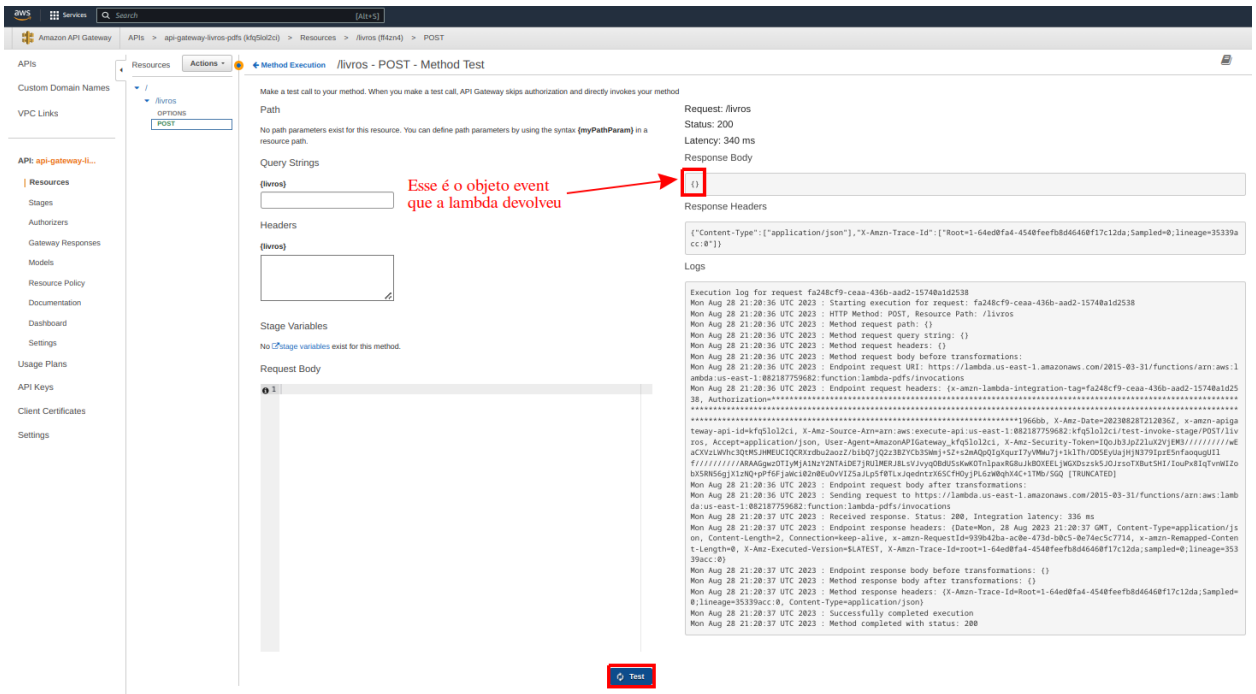
Mantenha o método **POST** selecionado para podermos escolher o **Integration Type**. Ou seja, mediante uma requisição POST neste Endpoint, precisamos especificar o que desejamos fazer. Neste caso, desejamos colocar a função lambda em funcionamento. Faça as escolhas como a seguir. Observe que você precisa começar a digitar o nome de sua função lambda para que ela apareça e possa ser selecionada.



A seguir, mantendo POST selecionado, clique em **Test.**



Clique novamente **Test**, agora na parte inferior da tela e observe o resultado. A função Lambda está devolvendo o objeto **event**. Ele representa aquilo que enviamos no corpo da requisição quando ela foi realizada. Neste momento, não enviamos coisa alguma. Por isso, o objeto está vazio, sem nenhum par chave/valor.



Autoria: Professor Rodrigo Bossini<professorbossini@gmail.com>
Produção: agosto de 2023

Referências

[1] Amazon Web Services (AWS) - Cloud Computing Services. 2023. Disponível em <https://aws.amazon.com/>. Acesso em agosto de 2023.

[2] PiCloud Launches Serverless Computing Platform To The Public | TechCrunch. 2023. Disponível em <https://techcrunch.com/2010/07/19/picloud-launches-serverless-computing-platform-to-the-public/>. Acesso em agosto de 2023.

[3] Serverless Architectures. 2023. Disponível em <https://martinfowler.com/articles/serverless.html>. Acesso em agosto de 2023.

[4]Who coined the term ‘serverless’?. 2023. Disponível em <https://www.quora.com/Who-coined-the-term-serverless>. Acesso em agosto de 2023.