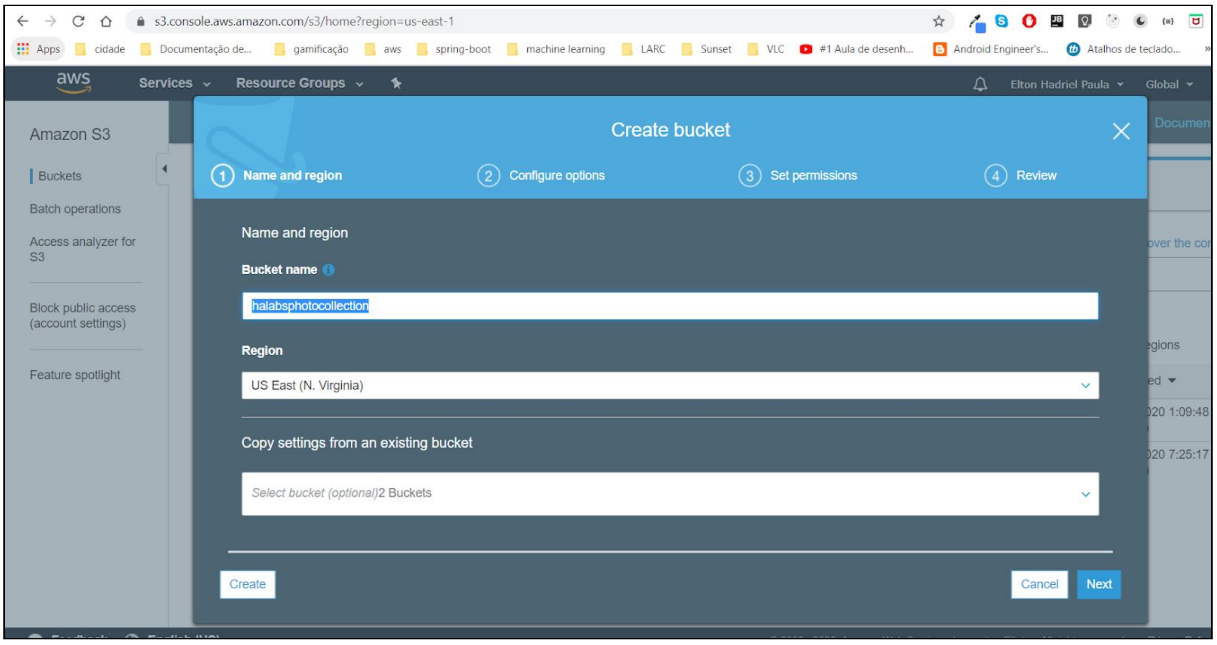
Nomes	Versão
Elton H. Paula	1

Pré Configuração

Crie um bucket que corresponde um serviço de armazenamento para que API Gateway com o objetivo de disparar uma requisição que tenha como função criar um arquivo neste serviço de armazenamento.

Para fazer isto procure pelo serviço Amazon S3 em seguida clique em create bucket e coloque no primeiro input o nome `halabsphotocollection`.



bucket

Em seguinte clique no botão “Next”.

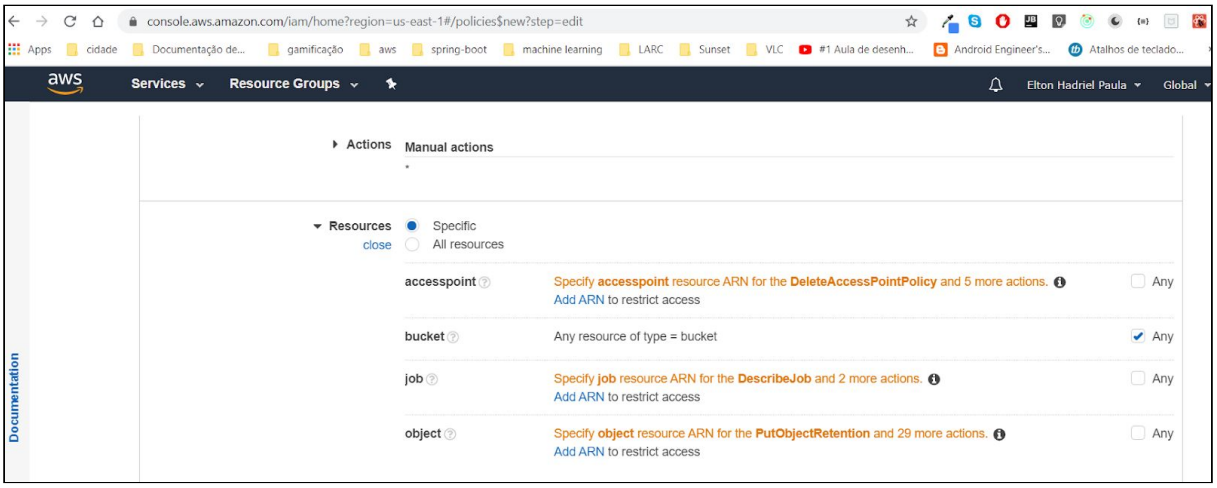
API Gateway

Serve para proteger e controlar acesso aos serviços interno que compõe a aplicação.

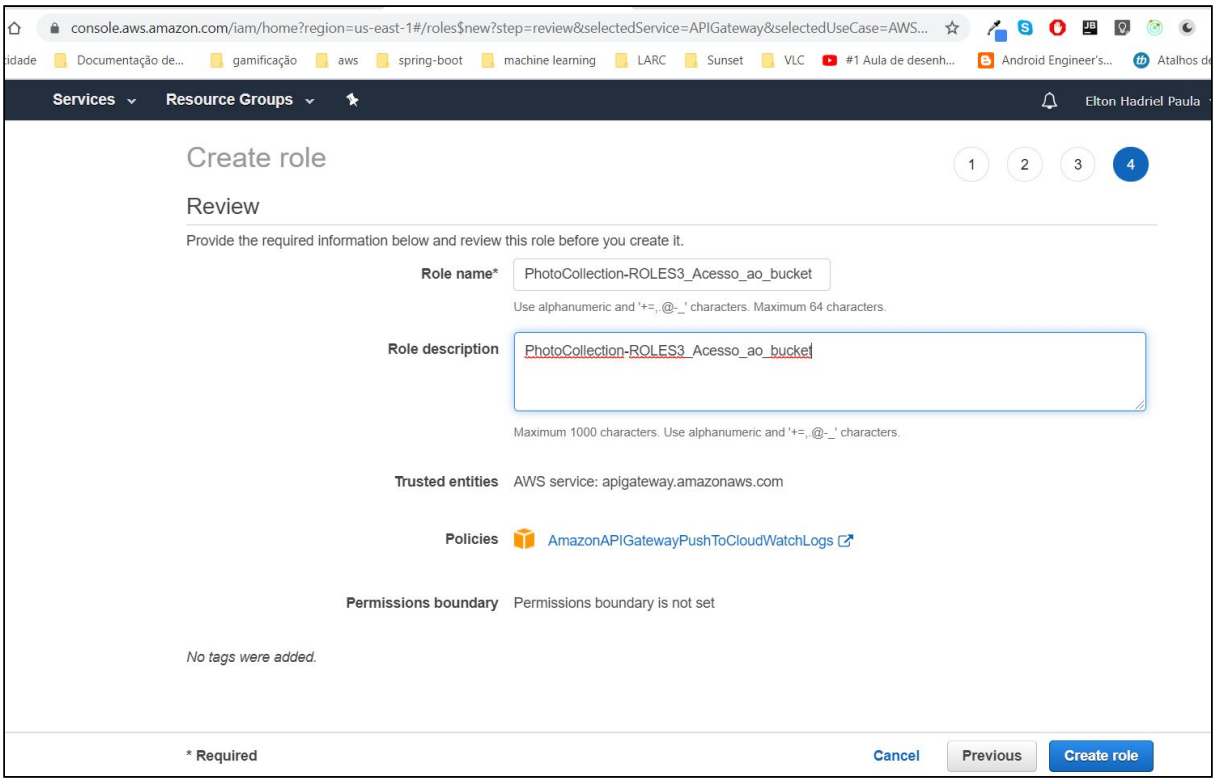
Política e roles

Objetivo: Front-end precisa publicar, apagar arquivos, inserir arquivos.

Entre no IAM clique em políticas, crie uma política com acessos. No momento que estiver criando uma política clique em *resource* e especifique qual `bucket` S3 você está direcionando está política.

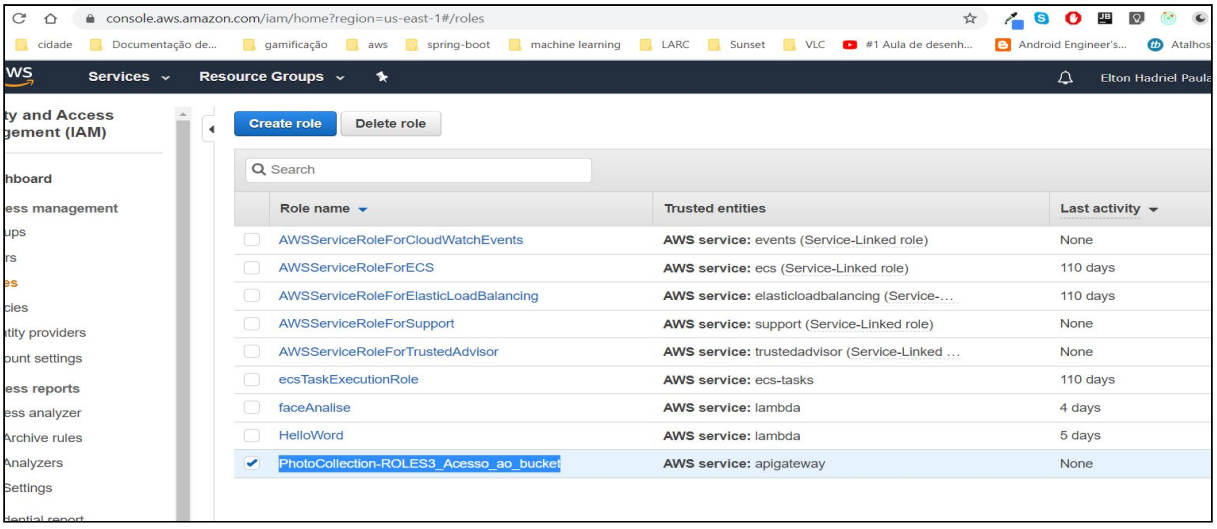


Na imagem exibe que o `bucket` que está sendo direcionando é halabsphotocollection isto ocorreu após clicar em “Add ARM”. Marque o “Any” para que possa trabalhar com qualquer objeto dentro do `bucket`, se não pode ocorrer permissão negada.
Avançamos para o *review*, escolhe o nome “PhotoCollection-S3_Acesso_bucket”. A descrição pode ser a mesma coisa. Crie a política.
Em seguida entre em roles, criar a nova role e nesta etapa deve ser definido qual o serviço que deve usar esta role, no nosso caso o serviço será a API Gateway.



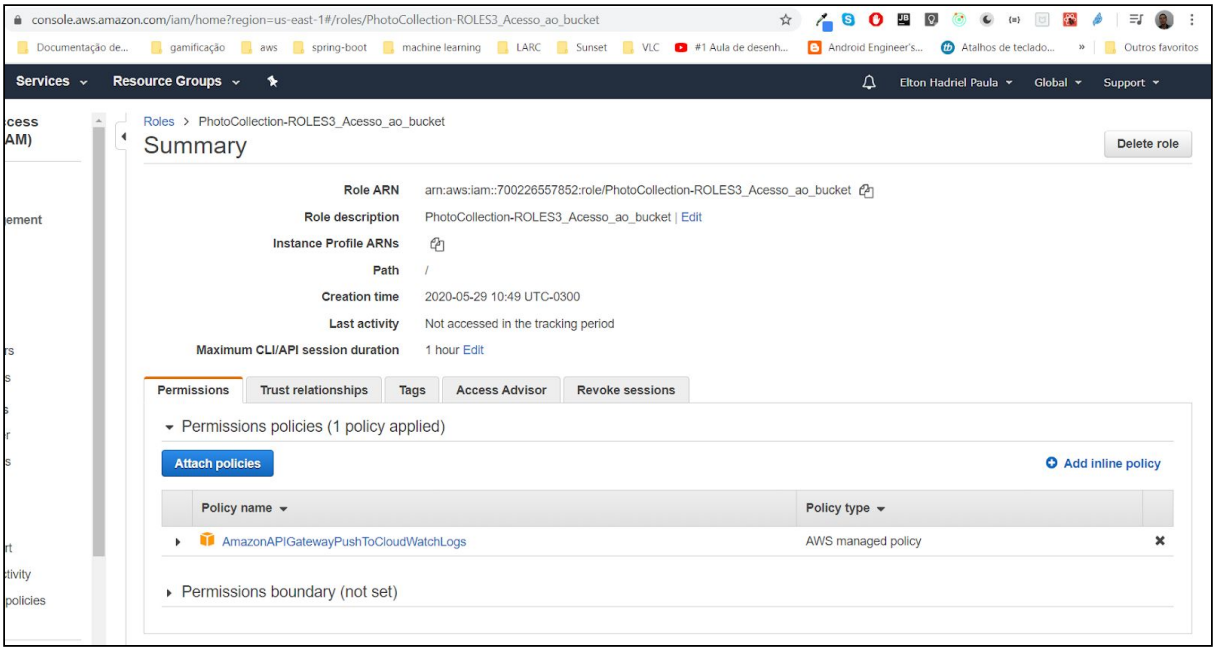
Criação da role

Após definir a role name clique em “create role”.
A diferença entre políticas e roles corresponde que a política é o que pode ser e o que não pode ser feito em um serviço. E a *role* corresponde qual o serviço que irá acessar a política. A *role* é qual o serviço que consome essa política. Ou seja, quem vai acessar essas regras, que no caso é o API gateway.



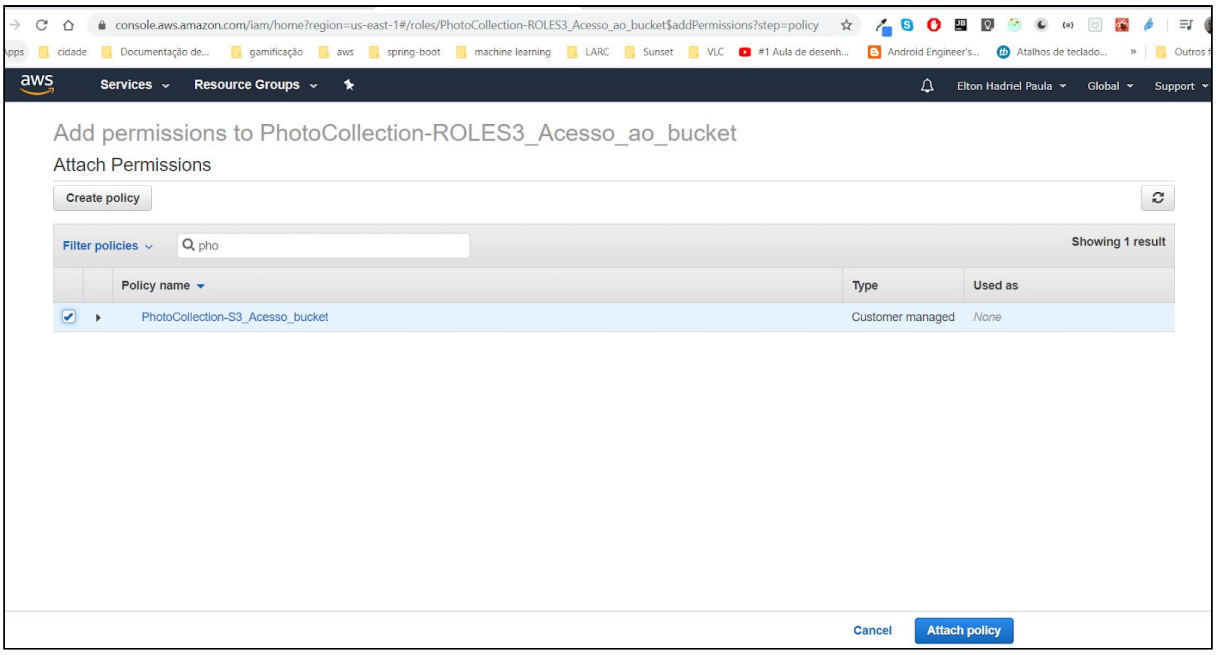
Role criada

Além de criar a role falta anexar a política, pois a role criada permite somente fazer o envio de mensagem para o *Cloud Watch* conforme a imagem abaixo.



A api gateway com a politica de envio de push para cloud watch

Clique em “Attach policies” em seguida procure a política criada e anexe a política conforme a imagem abaixo.



Anexar política a regra

Resumindo por padrão o acesso entre os serviços da AWS são feitos policies e roles, no qual a policies define o que pode ser utilizado dentro de um serviço, exemplo inserir um objeto dentro de um `bucket` e as roles serve para definir quais são os conjuntos de políticas estão vinculados ao serviço.

Usar uma API customizada na AWS

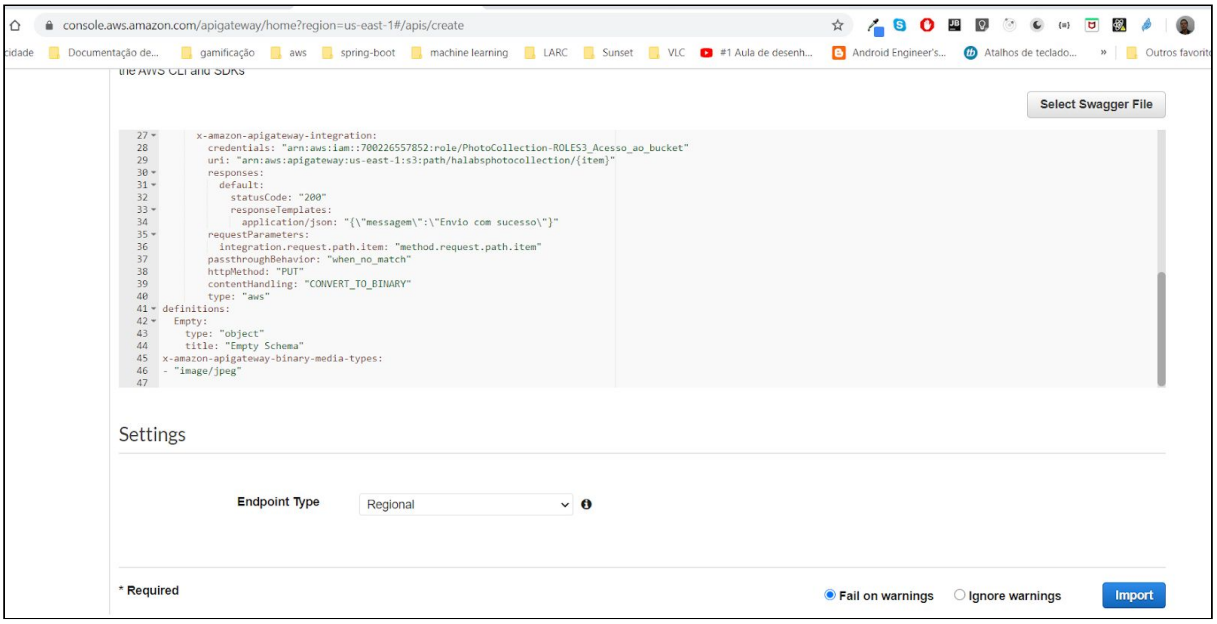
Salve esse código em um arquivo de extensão yml que corresponde um arquivo já criado que têm o método POST com o objetivo de inserir um arquivo através do *endpoint* /bucket/{item}, onde o item é o nome do arquivo a ser criado no bucket.

```
---
swagger: "2.0"
info:
  version: "2019-04-17T18:59:57Z"
  title: "PhotoCollection"
schemes:
- "https"
paths:
  /bucket/{item}:
    post:
      produces:
      - "application/json"
      parameters:
      - name: "Content-Type"
        in: "header"
        required: true
        type: "string"
      - name: "item"
        in: "path"
        required: true
        type: "string"
      responses:
        200:
          description: "200 response"
          schema:
            $ref: "#/definitions/Empty"
      x-amazon-apigateway-integration:
        credentials:
"arn:aws:iam::700226557852:role/PhotoCollection-ROLES3_Acesso_ao_bucket"
        uri: "arn:aws:apigateway:us-east-1:s3:path/halabsphotocollection/{item}"
        responses:
          default:
            statusCode: "200"
            responseTemplates:
              application/json: "{\"messagem\":\"Envio com sucesso\"}"
        requestParameters:
          integration.request.path.item: "method.request.path.item"
        passthroughBehavior: "when_no_match"
        httpMethod: "PUT"
        contentHandling: "CONVERT_TO_BINARY"
        type: "aws"
definitions:
  Empty:
    type: "object"
    title: "Empty Schema"
x-amazon-apigateway-binary-media-types:
- "image/jpeg"
```

arquivo yml

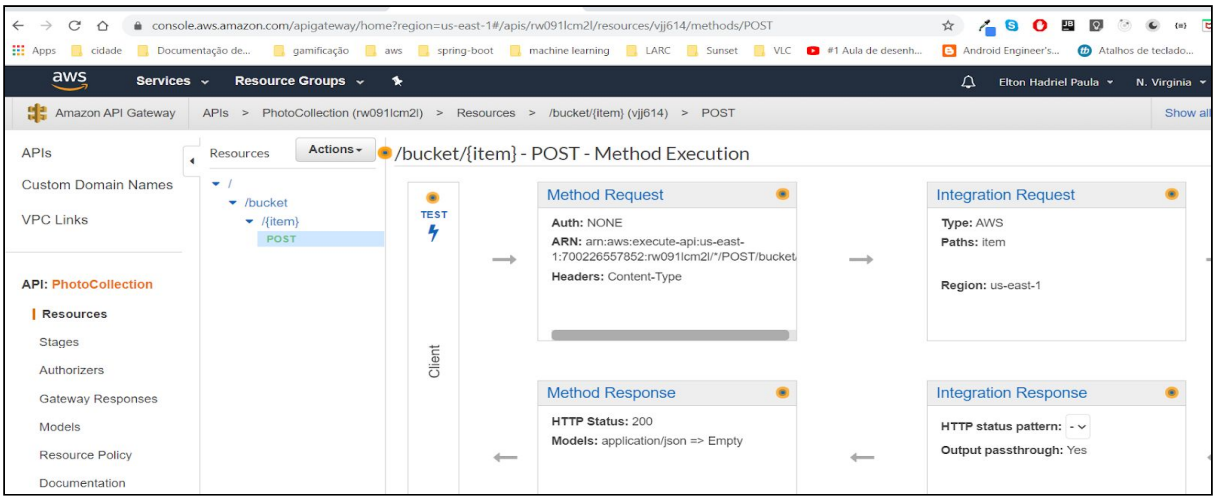
No atributo `credentials` e `uri`, certifique se o nome da regra criada `arn:aws:iam::700226557852:role/PhotoCollection-ROLES3_Acesso_ao_bucket` esteja no atributo credentials e a uri aponte para o bucket `halabsphotocollection` criado no início deste artigo.

Abra o serviço API Gateway e clique no botão “create API” em seguida em REST API clique em “Build”. Em seguida clique na opção que têm o swagger e insira o arquivo yml salvo anteriormente conforme a imagem.

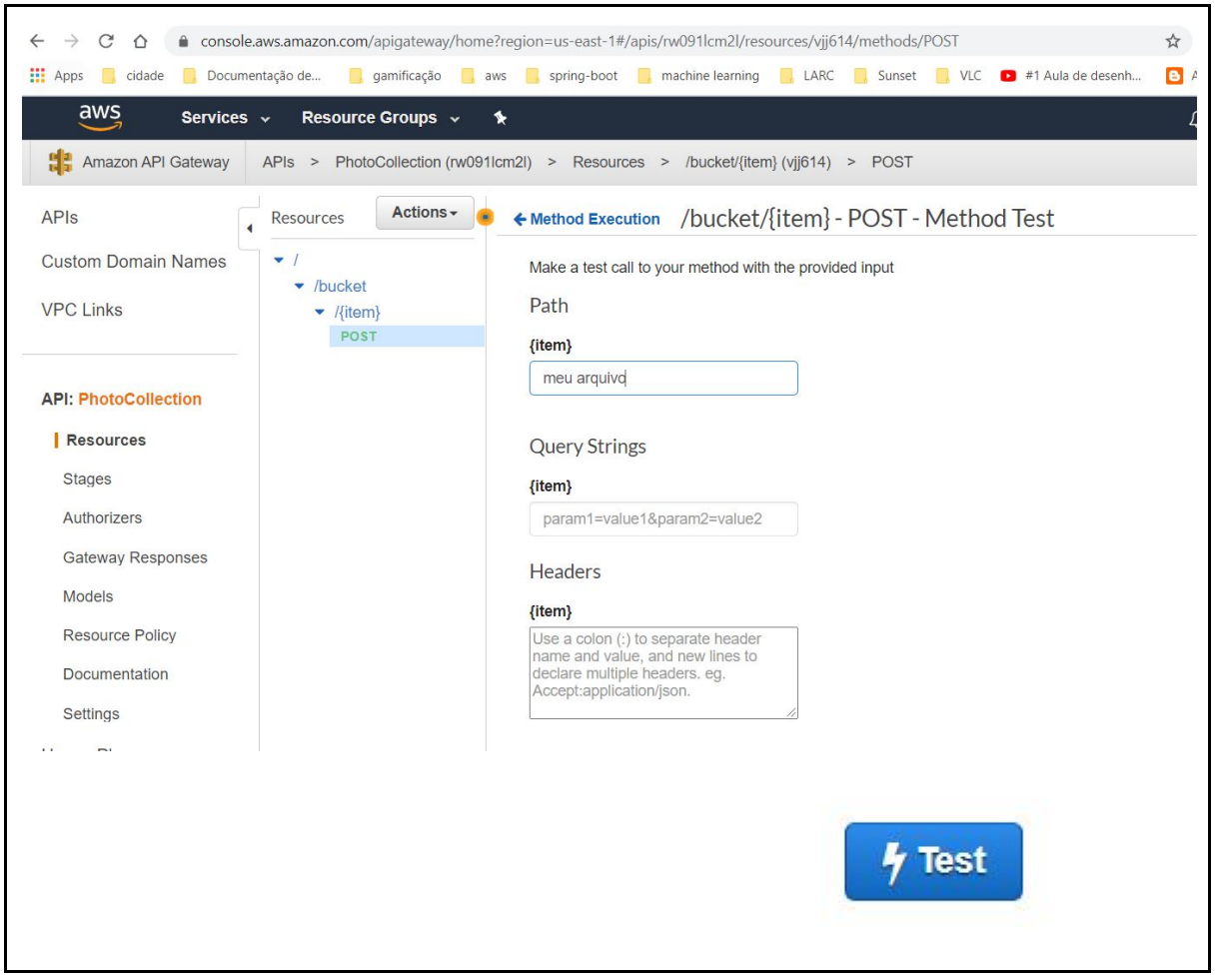


Serviço API Gateway

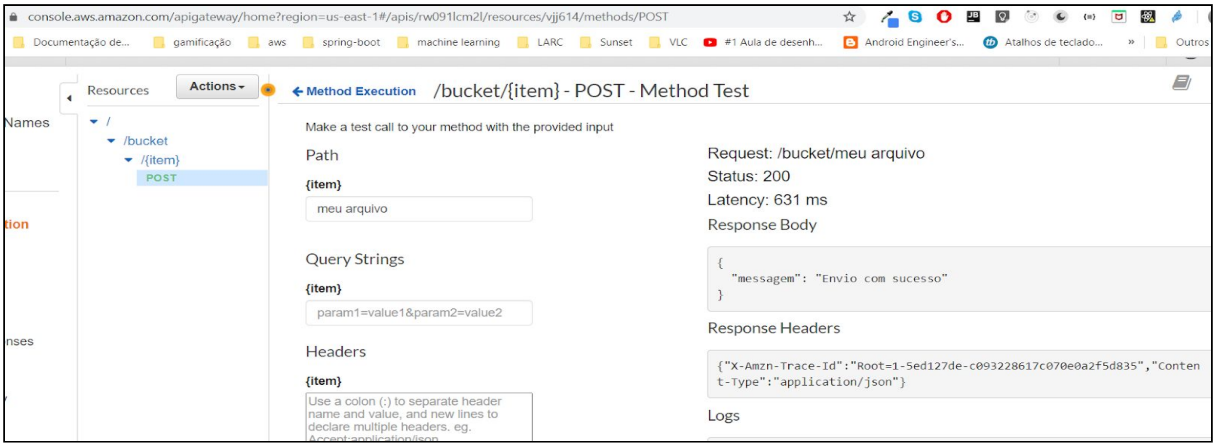
Clique no botão *import* com isto o arquivo JSON será inserido na API Gateway.
Faça o teste para verificar se um arquivo será criado para isto abra o serviço API Gateway escolha a API PhotoCollection e clique em POST logo em seguida em *test* conforme a imagem abaixo.



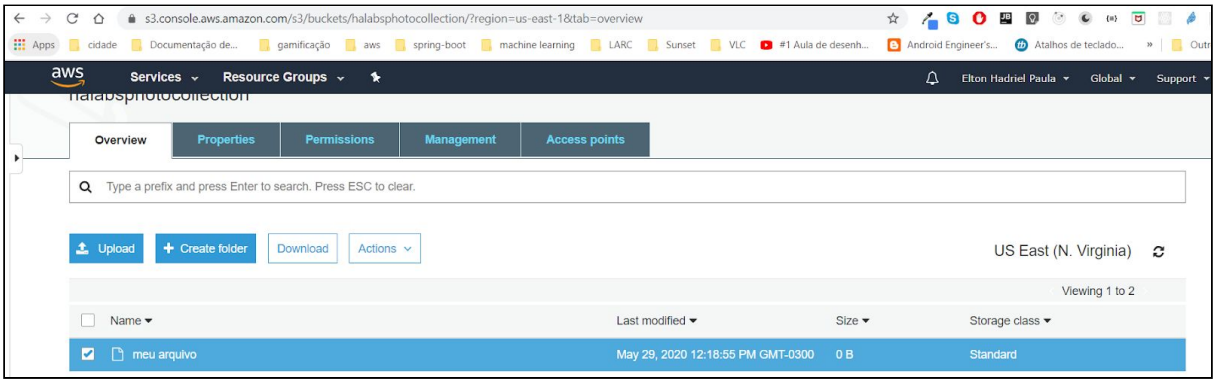
No primeiro `input` coloque um nome, neste caso foi inserido “meu arquivo” e clique no botão “test”.



Logo após uma mensagem de sucesso deverá ocorre e o arquivo será inserido no bucket s3 `halabsphotocollection` criado, isto se a política e regra estiver configurado conforme o capítulo anterior.



Mensagem de sucesso

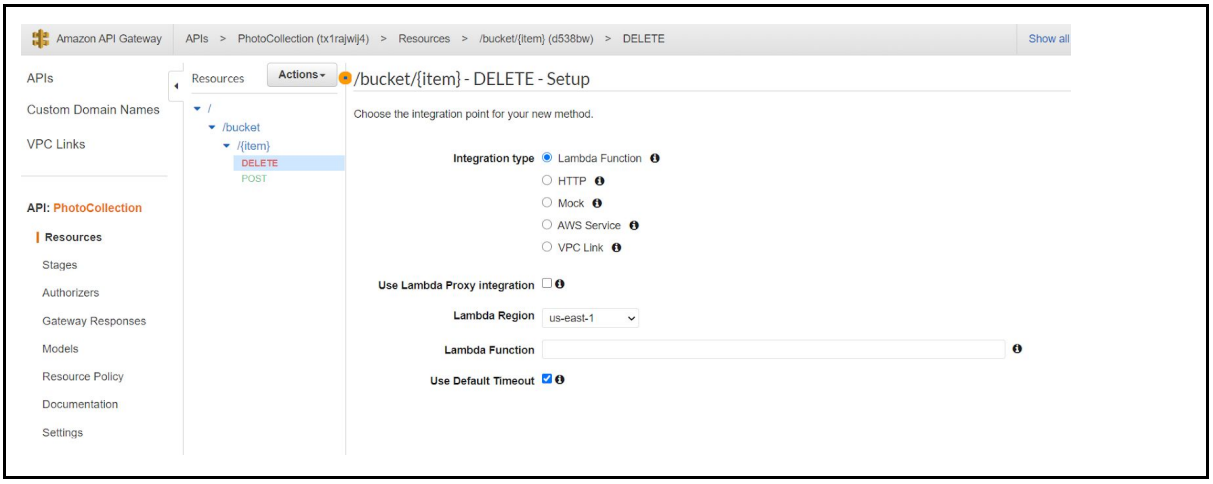


Arquivo “meu arquivo” inserido no S3 pela API Gateway

Conforme a imagem o arquivo “meu arquivo” foi inserido no bucket S3 halabsphotocollection.

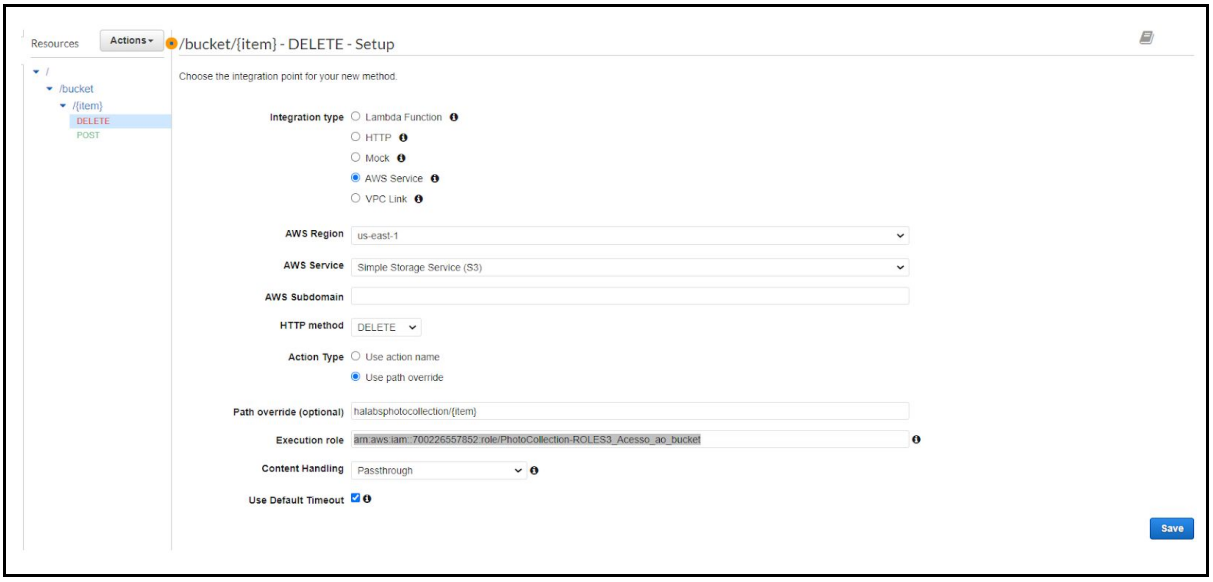
Criando um novo endPoint

Com a api gateway PhotoCollection aberta clique em Actions em seguida “**create Method**” e insira o nome delete conforme a imagem.



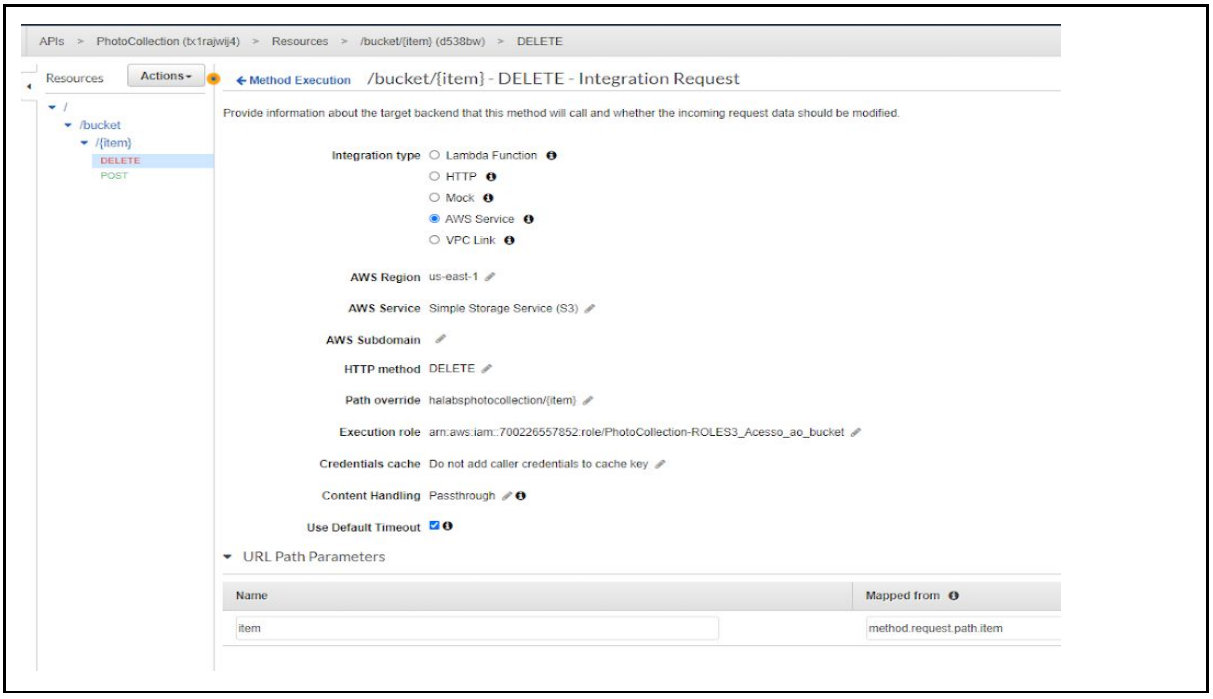
novo endpoint

No lado direito escolha AWS Service, a região us-east-1 que foi configurado a API, em seguida no campo AWS Service escolha S3, em HTTP method descreva o método DELETE em action type escolha a opção **Use path override**. Logo após insera a descrição halabsphotocollection/{item} no campo **Path override (optional)** e no campo Execute Role descreva a regra arn:aws:iam::700226557852:role/PhotoCollection-ROLES3_Acesso_ao_bucket conforme a imagem abaixo:



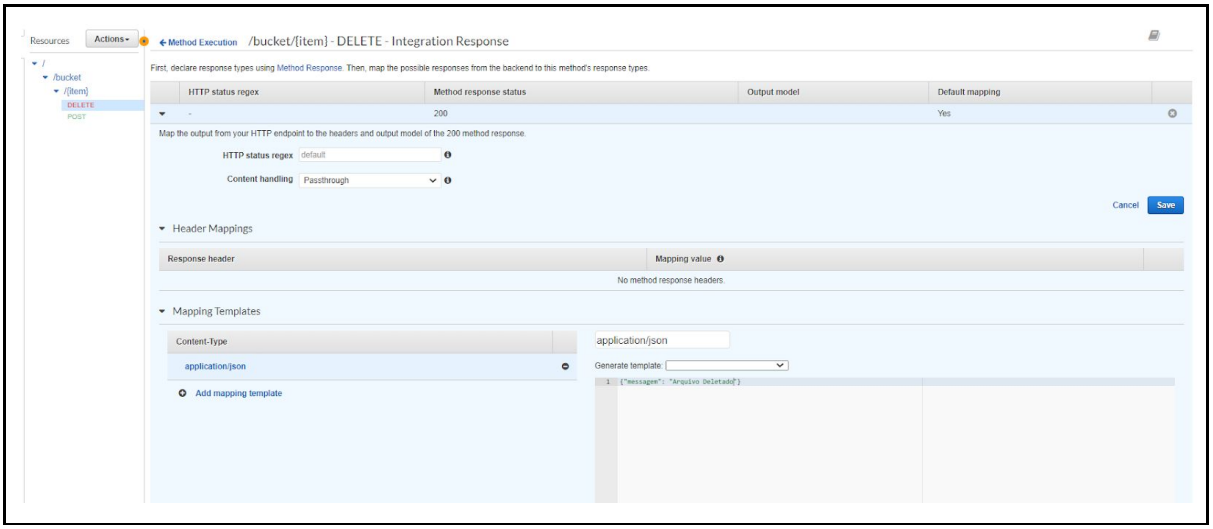
Configuração do endpoint DELETE

Em seguida clique em no botão save. Em integration Request clique em URL Path Paramenters e faça o mapeamento do atributo item conforme o desenho.



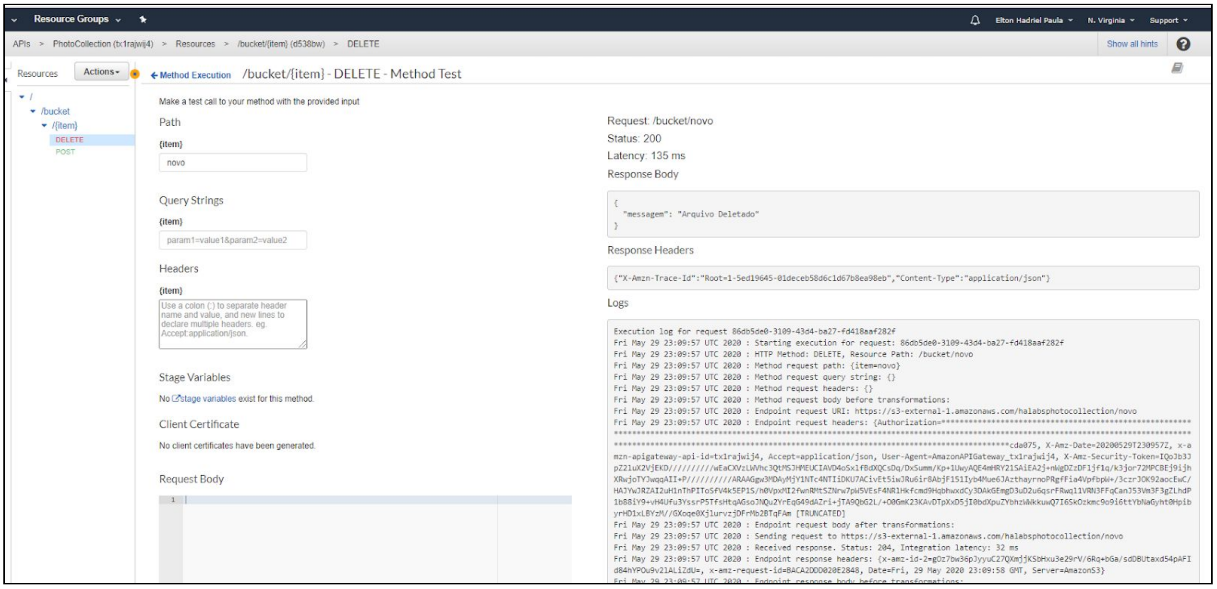
mapeamento do item

Depois disto vá em integration response e configure o método de retorno com a mensagem JSON {“message”:”arquivo deletado”}.



configuração do retorno de mensagem do servidor

E faça o teste inserido no primeiro input o nome do arquivo para verificar se o endpoint está deletando o arquivo dentro do bucket através do acesso do Method Test.



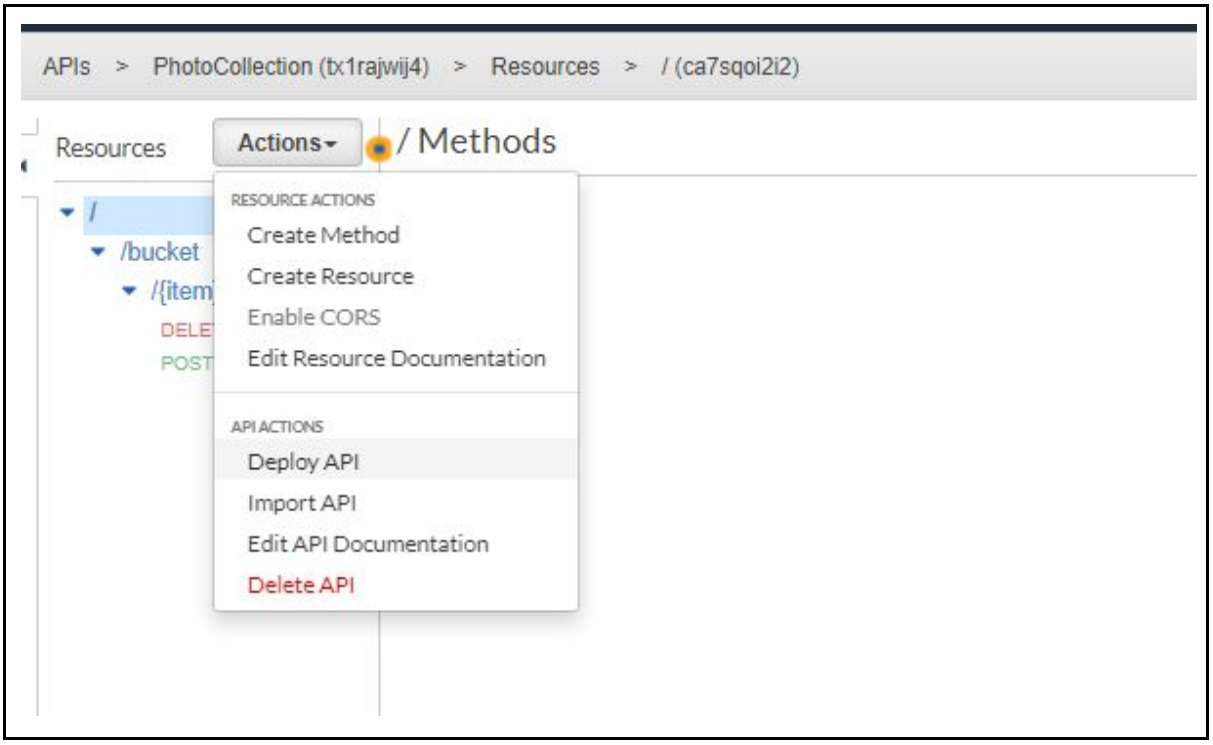
teste do endpoint DELETE

Caso a mensagem seja arquivo deletado, então o teste ocorreu com sucesso.

Na criação da API utilizamos 2 métodos: Um no lado cliente (REST) e outro para acionar os serviços internos da AWS. Desta forma a configuração correta para fazer um upload corresponde esses passos: Cliente → POST → API Gateway → PUT → S3 Bucket.

Deploy da API

A API pode ser publicada externamente para ser utlizado como um endpoint externo. Para isto deve clicar em deploy API, a AWS irá gerar um link externo com esse link pode ser colocado no DNS externo para ficar um nome mais elegante ou usar o DNS Router53 da AWS. Neste arquivo vamos usar o próprio link que a AWS gerou clique no endpoint POST copie este endpoint e abra um front-end de envio JSON POST-MAN ou de preferência.

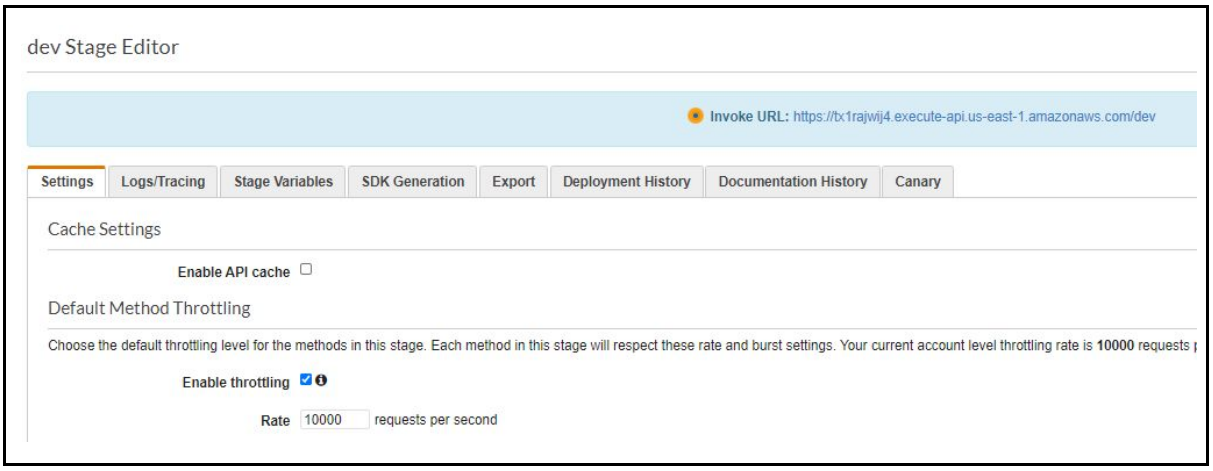


deploy API

Após o clique deve ser configurar o nome do estado, neste caso pode ser dev que significa que estado de desenvolvimento.

stage

Em seguida clique em create.
Será criado a url conforme a imagem abaio.



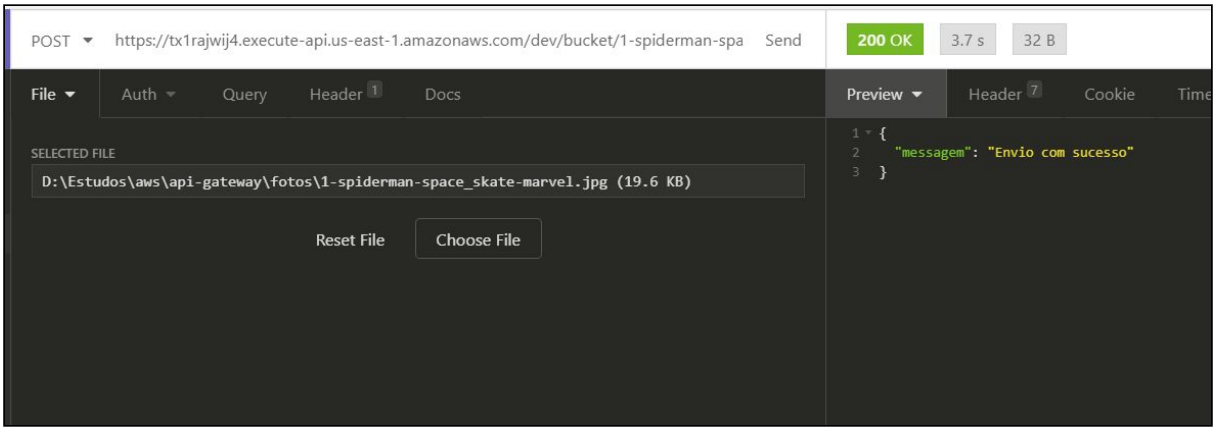
url-gerada-aws

Para exemplificar crie os arquivos de extensão jpg com os seguintes nomes:

- 1-spiderman-space_skate-marvel.jpg
- 2-ironman-aprendendo_a_voar-marvel.jpg
- 3-avengers-infinity_war-herois.jpg
- 4-batman-lego_batman-dc.jpg
- 5-batman-lego_movie-dc.jpg
- 6-jl-lego_cartoon-herois.jpg

Vamos imaginar que foi acordado com a equipe de desenvolvimento os seguinte formato para nome dos arquivos de extensão jpg que são: codigo, assunto, descrição e coleção tal que cada parte mencionada foi separado por “-”.

Abra a ferramenta JSON de preferencia e configura o POST para a URL <url-gerada-aws>/{item}, substituindo a url-gerada-aws e o item para o primeiro nome do arquivo da foto. Em Header configure o Content-type para image/jpeg e em Body escolha a opção binária e aponte para o caminho do arquivo, em seguida clique em Send.



endpoint /dev/bucket/{item}

Conforme a imagem ao chamar a url da aws gerada para a URI /dev/bucket/1-spiderman-space_skate-marvel.jpg é inserido o arquivo e a mensagem de retorno com código 200 e mensagem json com a descrição “Envio com sucesso” é retornada mostrando que o arquivo foi inserido com sucesso.

Serviço DynamoDB - Database Nosql

Neste capítulo vamos mostrar um resumo técnico do serviço DynamoDB que corresponde o banco de dados NOSQL da AWS.

No serviço IAM crie um novo policy e role para acessar DynamoDB:

- Sobre a policy:
    - Define o serviço DynamoDB
    - Use a permissão Read apenas e adicione o ARN da tabela
    - Use o nome PhotoCollection-DynamoDB_Acesso_a_tabela
- Sobre o role:
    - No novo role escolhe o serviço API Gateway
    - Use o nome PhotoCollection-ROLE-DynamoDB_Acesso_a_tabela
    - Ainda no role vincule a policy com role

De volta a API Gateway, na nossa API PhotoCollection, crie novos recursos //photo com sub-recursos /{id}, /assunto e /consulta.

- Na PhotoCollection crie um recurso /photo
    - Adicione um sub-recurso /photo/{id}
    - Use o método GET (Resource Path: /photo/{id})
    - A integração deve ser AWS Service -> DynamoDB
    - Use Método POST na action Scan
    - Também coloque o nome da role criada
    - No Integration Request procure Mapping Templates
        - Escolha a segunda opção (a recomendada)

- Adicione um template com o Content-Type e digite no campo application/json
- Clique no template e adicione o JSON abaixo:

```
{
   "TableName" : "PhotoCollection",
   "FilterExpression" : "id = :v1",
   "ExpressionAttributeValues" :  {
     ":v1" : { "S" : "$input.params('id')" }
   }

 }
```

- Salve e volta a visão do recurso e clique no Teste
    - Teste com uma ID cadastrado no banco

Sub-recurso /assunto

- Adicione mais um sub-recurso /assunto
    - Use o método GET (Resource Path: /photo/assunto)
    - Use as mesmas configurações:
    - AWS Service -> DynamoDB, Method POST com action Scan, cole a role
- Crie um novo JSON template:
- Clique no template e adicione o JSON abaixo:

```
{
 "TableName" : "PhotoCollection",
 "ProjectionExpression" : "id, descricao, colecao",
 "FilterExpression" : "assunto = :v1",
 "ExpressionAttributeValues" :  {
   ":v1" : { "S" : "$input.params('nome')" }
 }}
```

- Salve e volta a visão do recurso e clique no Teste
    - Teste com Query String : nome=batman

Sub-recurso /consulta

- Use o método GET (Resource Path: /photo/consulta)
    - Configurações: AWS Service -> DynamoDB, Method POST com action Scan, cole a role
    - Template:

```
{
  "TableName" : "PhotoCollection",
  "FilterExpression" : "assunto = :v1 AND colecao = v2",
  "ExpressionAttributeValues" :  {
    ":v1" : { "S" : "$input.params('assunto')" },
    ":v2" : { "S" : "$input.params('colecao')" }

  }

 }
```

- Salve e volta a visão do recurso e clique no Teste
    - Teste com Query String : assunto=batman&colecao=dc

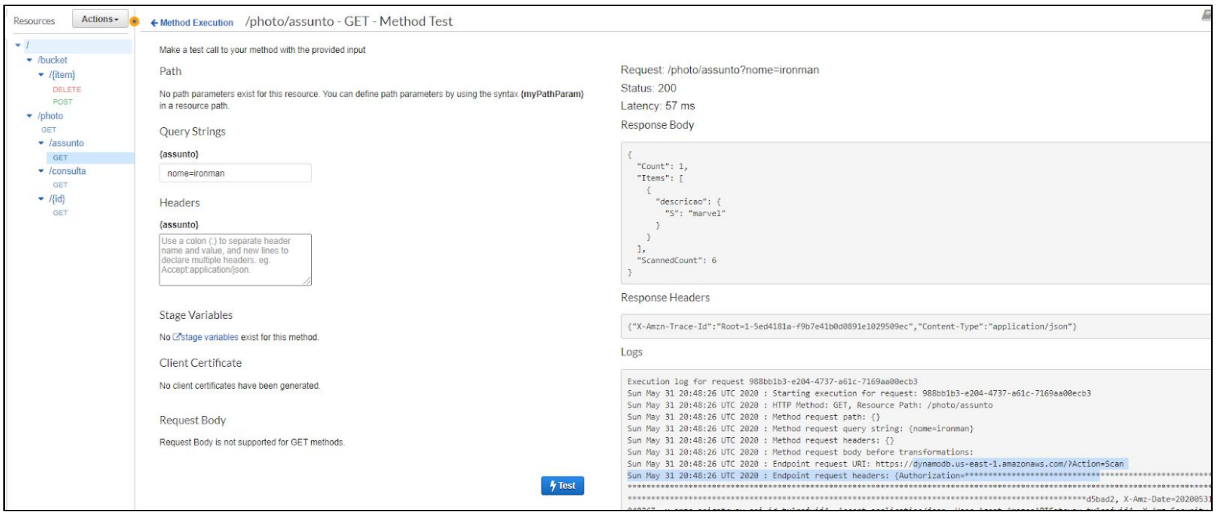
Mock Service /photo

- No Integration Response apague o template que já existente e adicione um novo template HTML com o conteúdo da página HTML abaixo desse vídeo
    - Adicione um header da resposta: Content-Type e 'text/html'

Com isto dentro a API Gateway irá funcionará para os recursos photos dos seguintes endpoints:

endpoint (recursos e sub-recursos)

Com as configurações mencionadas com o recurso photo e sub-recursos /{id}, /assunto e /consulta podemos usar a propria API Gateway para testar se através de uma requisição funcionará a consulta no DynamoDB conforme o exemplo abaixo ao acessar um sub-recurso /consulta pelo filtro nome=ironman.



Exemplo GET endpoint /photo/assunto

Assuntos avançados

Neste capítulo vamos inserir tópicos que deverão ser aprofundados em futuras versões:

Segurança da API Gateway com API Key

1. Criação do novo deploy em produção e configuração da :
    - Escolhe Actions -> Deploy API
        - No formulário escolhe [New Stage] e o nome Stage name deve ser v1
        - Teste a nova URL usando o Postman

- Chave de acesso e plano de uso:
  - Para criar um API Key clique no menu da esquerda no item API Keys
    - No formulário no campo nome coleque photocollection_dev, pode colocar uma descrição e salve
    - Para criar um plano de uso clique no menu da esquerda no item Usage Plan
  - No formulário no campo nome Basic
  - No Throttling coloque a Rate 100 (numero máximo de requisição em 1 segundo) e o Burst 200
  - Na Quota (numero máximo de requisição em 1 mês) coloque 5000 e avance no formulário
    - Escolhe a API PhotoCollection e Stage v1, confirme e avance
    - Associa o API Key com o Usage Plan e clique no Done
- Associação da key com recurso:
  - Volta na APIs -> PhotoCollection -> Resource
    - Habilite a segurança para os método DELETE e POST do recursos /bucket/{item}
  - Não esqueça de gerar um novo deploy (usando o v1)

- Teste:
  - Para testar os recursos protegidas use o Postman e adicione o header x-api-key

2. Resumo
   a. Criar uma API KEY;
   b. Criar um Plano de Uso
   c. Associar a API KEY ao plano de uso;
   d. habilitar a api key nos métodos da API Gateway que precisa de segurança
   e. Na ferramenta REST de preferência em header passamos o x-api-key juntamente com a chave gerada (api key) no API Gateway.

Referência Bibliográfica

[API Gateway] - https://docs.aws.amazon.com/pt_br/apigateway/?id=docs_gateway

[DynamoDB] - https://docs.aws.amazon.com/pt_br/amazondynamodb/latest/APIReference/API_Scan.html#API_Scan_Examples

[DynamoDB em PDF] - https://docs.aws.amazon.com/pt_br/amazondynamodb/latest/APIReference/dynamodb-api.pdf#API_Operations_Amazon_DynamoDB

[S3] - https://docs.aws.amazon.com/pt_br/s3/?id=docs_gateway