
API Gateway

— Prof. Raphael Rodrigues Pereira —

O que é API Gateway

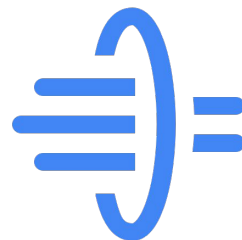
Uma ponte entre as requisições e os serviços



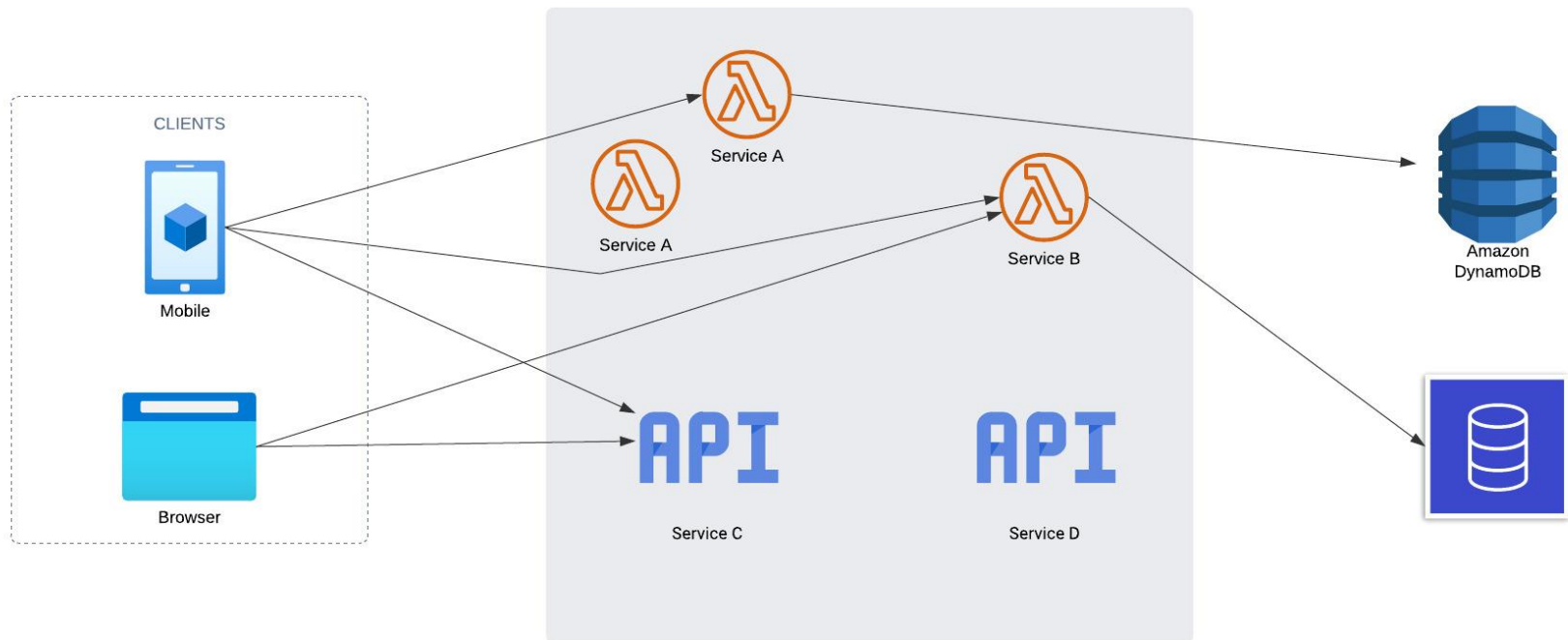
Azure API Management



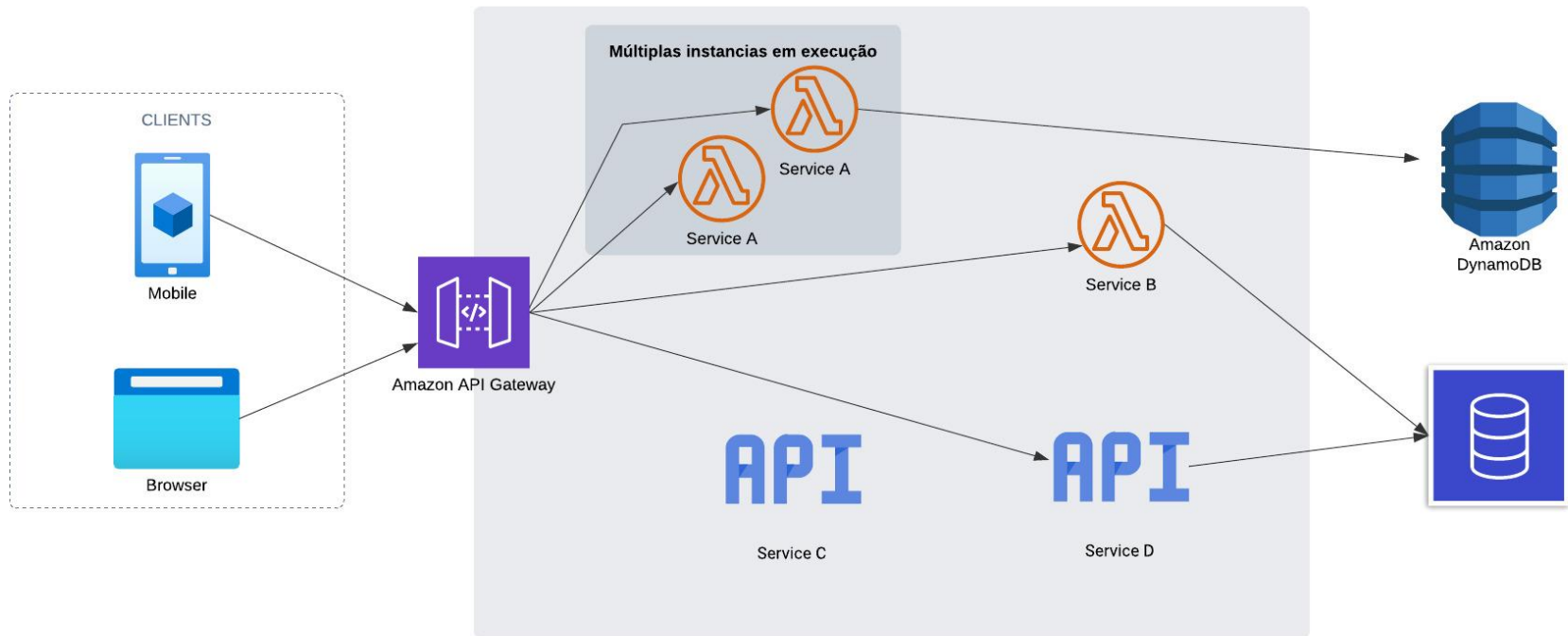
Google Cloud
API Gateway



Arquitetura sem API Gateway

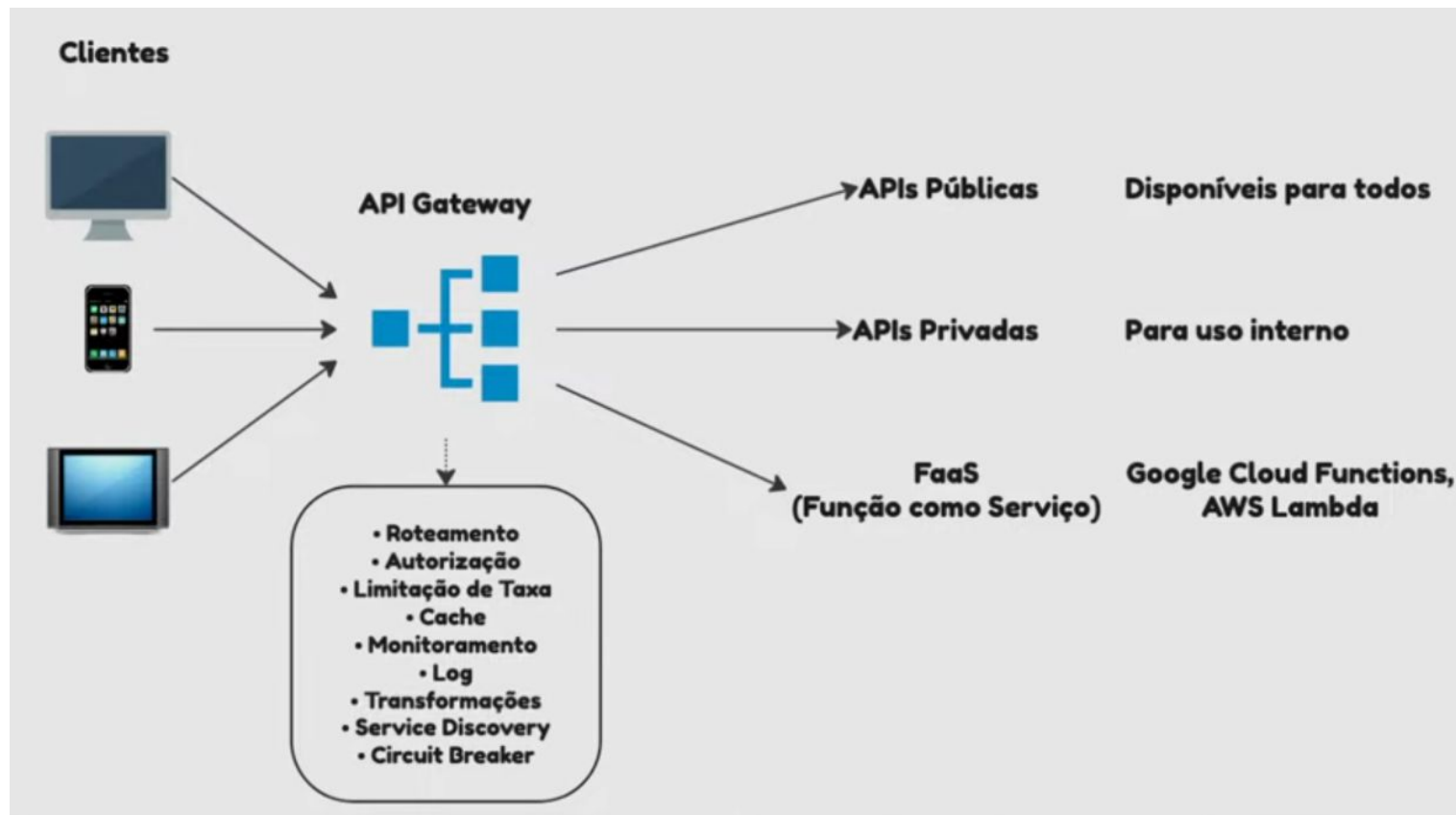


Arquitetura com API Gateway



Definição de API Gateway





Funções do API Gateway

- **Roteamento:** ponto central de chamada para serviços e o API encaminha para o serviço específico.
- **Limitação de Taxa:** O acesso aos micro serviços pode ser limitado - x requisições por período de tempo - (exemplo: Proteção DDoS)
- **Cache:** pode ser uma camada de cache - armazenar algumas respostas recebidas anteriormente evitando acionar novamente o serviço

Funções do API Gateway

- **Autenticação e Autorização:** ponto central de verificação de identidade do usuário - junto com serviço IAM
- **Balanceamento de Carga:** a nível de serviço o API Gateway pode tratar a distribuição de carga
- **Monitoramento:** métricas, informações sobre solicitações e respostas. Facilita detecção e tratativas de problemas.

Funções do API Gateway

- **Transformação:** possibilidade de pegar informações de um micro serviço, tratar a informação e enviar como resposta (Ex. transformar XML em JSON)
- **Agregação:** Requisições que precisam acionar mais de um serviço, podem ser encadeadas e o API Gateway pode realizar as chamadas, tratar as respostas e enviar a resposta tratada para o cliente.
- **Validação:** Validação de requisição e de resposta. Verificar se os dados da requisição estão completos, o próprio API Gateway já barra a requisição e evita execução de dados incorretas no serviço.

Funções do API Gateway

- **Service Discovery:** Integração com serviços de descoberta. API Gateway saber que determinado serviço subiu no servidor X. Todo serviço que sobe é registrado no service discovery e o API Gateway consulta este serviço para encaminhar as requisições.
- **Circuit Breaker:** Garante que a falha de um serviço não pare o restante do sistema. Um serviço que começa a funcionar de forma estranha o circuit braker já passa a chamar outra instância do serviço.
- **Versionamento:** fazer a gestão de versões da API para entregar a versão correta para determinado cliente.

Funções do API Gateway

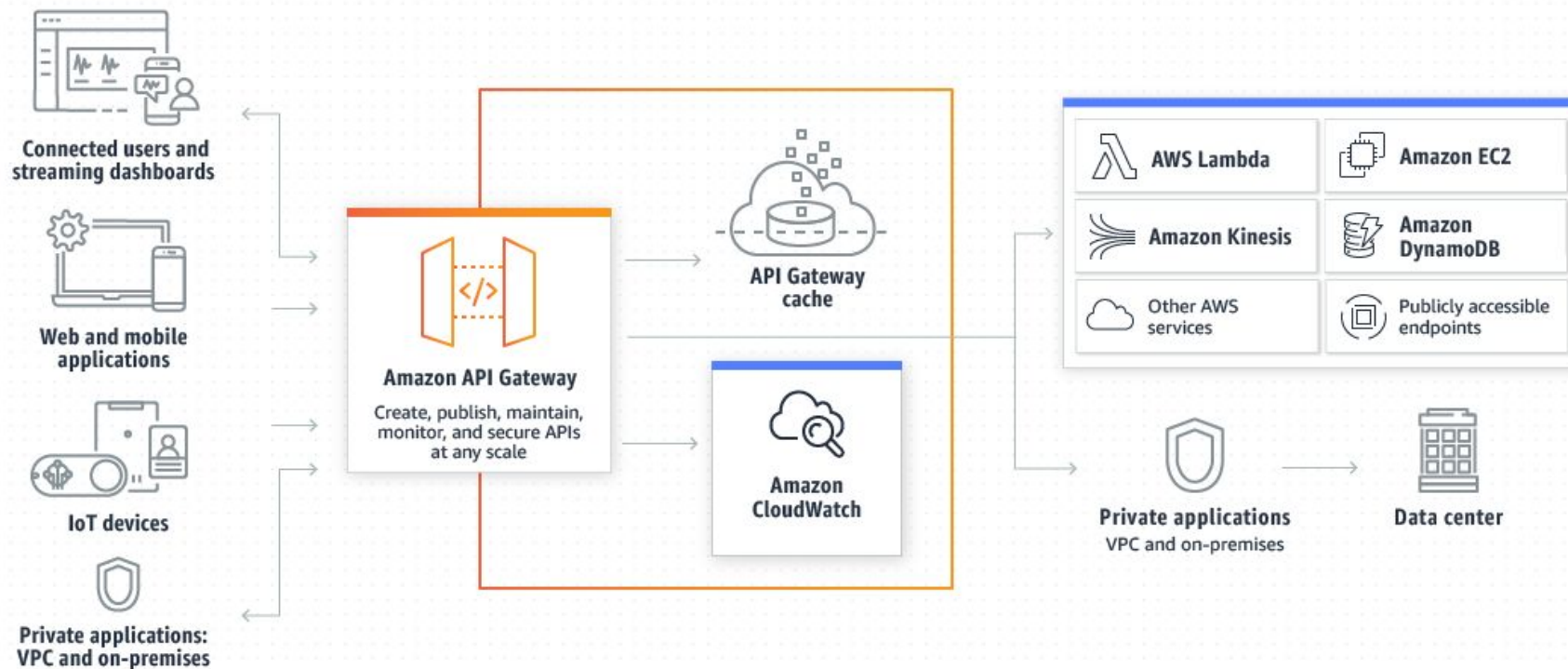
- **Tratamento de Erros:** pode ser o componente que tratará erros e retornará resposta para cliente. Por exemplo, um micro serviço está down.
- **WAF:** Web Application Firewall - proteção contra SQL Injection, XSS, DDoS, etc.
- **Documentação:** API Gateway gera e fornece documentação facilmente.

Benefícios

- **Desempenho:** Cache, limitação de taxa → Reduz latência
- **Simplificação:** Centralização do ponto de chamada para as APIs.
- **Segurança aprimorada:** Política de autenticação e autorização fica a cargo do API Gateway
- **Escalabilidade aprimorada:** Divide solicitação entre vários micro serviços.
- **Monitoramento e Visibilidade:**
- **Integração Simplificada:**

Desvantagens

- **Complexidade adicional:** Novo componente na arquitetura do sistema aumenta a complexidade de gestão do mesmo.
- **Ponto único de falha:** se o API Gateway parar o software todo para, portanto, atenção para **redundância** deste item.
- **Latência:** aumenta o tempo de resposta se comparar à chamada direto para um serviço. Tenta compensar isso com cache por exemplo.
- **Lock in:** preso ao provedor de serviços dependendo do que for utilizar do API Gateway
- **Custo:** o custo pode aumentar por ser um item a mais na arquitetura do software.
- **Complexidade de configuração:** quanto mais funcionalidades for utilizar mais complexa a configuração.



Fonte: https://docs.aws.amazon.com/pt_br/apigateway/latest/developerguide/welcome.html

Exemplo Prático AWS

API Gateway comunicando com Lambda Function e DynamoDB

ATENÇÃO! Os próximos slides servem apenas como guia informativo. Não é necessário que os alunos sigam esses passos durante a aula. No entanto, para quem desejar reproduzir as etapas, será necessário criar uma conta na AWS e cadastrar um cartão de crédito.

A AWS oferece recursos gratuitos através do Free Tier, mas é importante ficar atento aos limites de uso para evitar cobranças indesejadas. Recomendamos também que, ao final dos experimentos, todos os recursos criados na AWS sejam excluídos para evitar custos adicionais futuros.

Passos para execução na AWS

1- Criar função Lambda do zero.

- Nome da function criada neste exemplo: **ContactFormFunction**
- Foi selecionada a linguagem Python e todas as configurações padrão.

2- Adaptar o código para mostrar os dados enviados no corpo da requisição na tela:

```
import json

def lambda_handler(event, context):

    data = event['name'] + ' ' + event['email'] + ' ' + event['phone'] + ' ' + event['message']

    return {
        'statusCode': 200,
        'body': json.dumps('Dados recebidos: ' + data)
    }
```


3- Criar API Gateway

- Acesse a página principal da API Gateway
- Localize **API REST** e clique em Compilar
- Selecione a opção **API Nova**
- Defina um nome para sua API Gateway
- Escolha uma das opções de Tipo de Endpoint (neste caso selecionamos a opção: Regional)
- Clique em **Criar API**

Criar API REST

Detalhes da API

☒ **API nova**
Crie uma nova API REST.

☐ **Clonar a API existente**
Crie uma cópia de uma API nessa conta da AWS.

☐ **Importar API**
Importe uma API de uma definição de OpenAPI.

☐ **API de exemplo**
Saiba mais sobre o API Gateway com uma API de exemplo.

Nome da API

MyAPI

Descrição: *opcional*

Tipo de endpoint de API

As APIs regionais são implantadas na região atual da AWS. As APIs otimizadas para a borda direcionam as solicitações para o ponto de presença do CloudFront mais próximo. As APIs privadas só podem ser acessadas de VPCs.

Regional

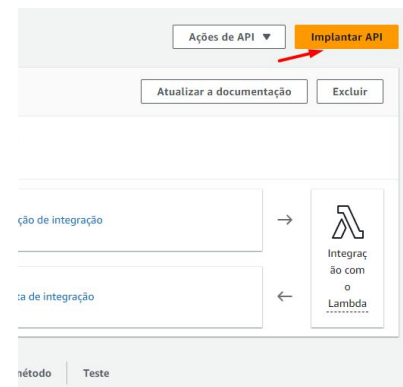
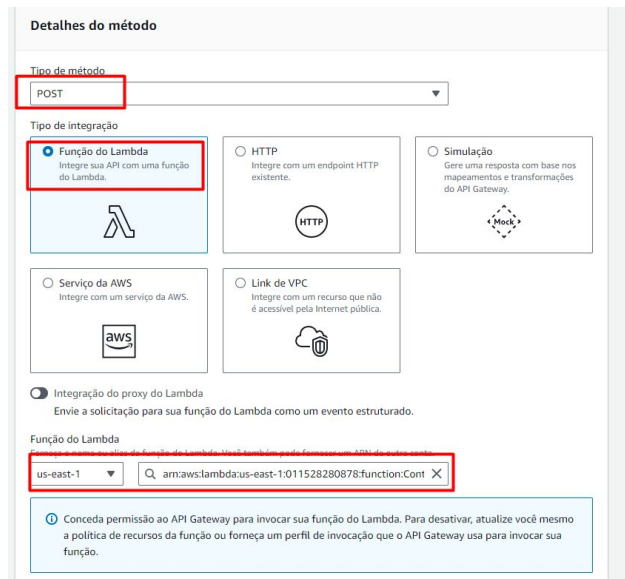
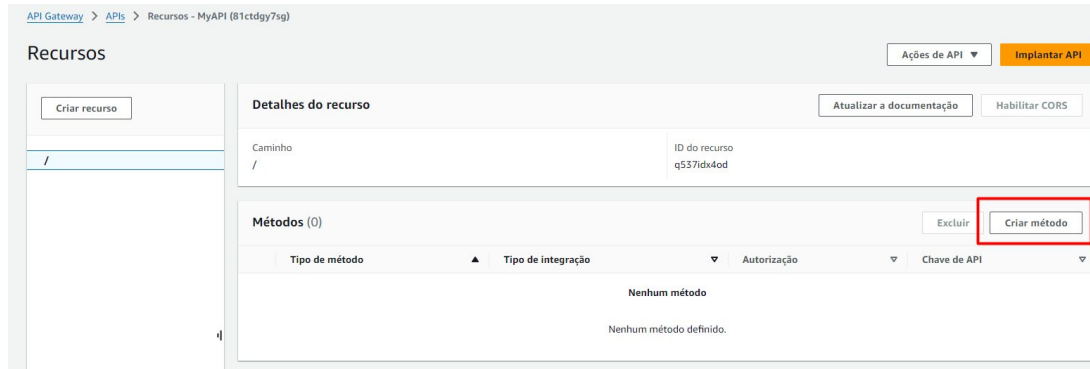
Cancelar

Criar API

4- Criar método POST


- Localize e clique no botão **Criar Método**
- Selecione o **método** tipo **POST**
- Em **Tipo de Integração**, selecione a opção **Função Lambda**
- Selecione a região e o nome da função lambda criada no primeiro passo.
- Clique em **Criar Método**
- Clique em **Implantar API**

*Será solicitado que selecione vincule essa implantação a um estágio - crie um estágio, por exemplo **dev***



Teste a API

- Clique na guia Testes
- Informe o body da requisição no formato json
- Clique em Testar
- Será exibido o log.
- Veja se no campo Corpo da Mensagem aparecerá a mensagem de retorno da requisição conforme configurado na Lambda Function.
- Veja também no monitoramento da Lambda Function se o log do CloudWatch foi atualizado com a requisição. (Se tiver dúvidas, consulte slides da aula sobre lambda)

 / - Resultados do teste de método POST

Solicitação	Latência ms	Status
/	373	200

Corpo da resposta

```
{ "statusCode": 200, "body": "\"Dados recebidos: Raphael R. Pereira raphael.rodrigues@imepac.edu.br 34 3249-3900 Teste j\\u00e1 na API Gateway + Lambda \"" }
```

Cabeçalhos de resposta

aws/lambda/ContactFormFunction

Ações

Visualizar no Logs Insights

Começar a seguir

Pesquisar grupo de logs

▼ Detalhes do grupo de logs

Classe de log

Informações

Padrão

ARN

amawslogs-us-east-1:011528280878log-group:/aws/lambda/ContactFormFunction*

Hora de criação

36 minutos atrás

Retenção

Nunca expirar

Bytes armazenados

-

Filtros de métrica

0

Filtros de assinatura

0

Regras do Contributor Insights

-

ID da chave do KMS

-

Detecção de anomalias

Configurar

Proteção de dados

-

Contagem de dados confidenciais

-

Streams de log

Tags

Detecção de anomalias

Filtros de métrica

Filtros de assinatura

Contributor Insights

Proteção de dados

Streams de log (4)

☐ Correspondência exata ☐ Mostrar expirados

Informações

< 1 > ⓘ

Fluxo de logs	Hora do último evento
<input checked="" type="checkbox"/> 2024/10/10/[\$LATEST]0e53cd5e984542d49cafb6b6f9dcda1c	2024-10-10 18:04:17 (UTC)
<input type="checkbox"/> 2024/10/10/[\$LATEST]af3076f9f08149a4b05826fce12e63b3	2024-10-10 17:55:42 (UTC)
<input type="checkbox"/> 2024/10/10/[\$LATEST]0eaf8df94ddd43f4ae89304684a78b8d	2024-10-10 17:30:29 (UTC)
<input type="checkbox"/> 2024/10/10/[\$LATEST]bcf956cd2119455da923c880d6fca58	2024-10-10 17:30:02 (UTC)

5- Criar tabela DynamoDB

- **Acesse** o painel do **DynamoDB** e clique em **Criar Tabela**
- Informe um nome para a tabela (neste exemplo foi **ContactFormDataBase**)
- Informe um nome para a chave de partição (neste caso, ID do tipo String)
- Clique em **Criar Tabela**
- Clique sobre o nome da tabela, localize e clique em **Informações Adicionais** e copie o **ARN da tabela**

Criar tabela

Detalhes da tabela [informações](#)

O DynamoDB é um banco de dados sem esquema que requer apenas um nome de tabela e uma chave primária quando você cria a tabela.

Nome da tabela
Isso será usado para identificar sua tabela.

Entre 3 e 255 caracteres, contendo apenas letras, números, sublinhados (_), hífens (-) e pontos (.).

Chave de partição
A chave de partição faz parte da chave primária da tabela. É um valor de hash usado para recuperar itens de sua tabela e alocar dados entre hosts para escalabilidade e disponibilidade.
 String
De 1 a 255 caracteres e diferencia maiúsculas de minúsculas.

Chave de classificação - opcional
É possível usar uma chave de classificação como s ou pesquisar entre todos os itens que compartilham

De 1 a 255 caracteres e diferencia maiúsculas de minúsculas.

Configurações da tabela

- Configurações padrão
A maneira mais rápida de criar a tabela. Você pode modificar essas configurações agora ou depois de criar a tabela.

ContactFormDataBase [Ações](#) [Explorar itens da tabela](#)

[Visão geral](#) [Índices](#) [Monitorar](#) [Tabelas globais](#) [Backups](#) [Exportações e streams](#) [Permissões](#) [Configurações adicionais](#)

Proteja sua tabela do DynamoDB contra gravações e exclusões acidentais
Quando você ativa a recuperação a um ponto anterior no tempo (PITR), o DynamoDB faz backup dos dados da tabela automaticamente para que você possa restaurar a qualquer segundo nos últimos 35 dias. Cobranças adicionais se aplicam. [Saiba mais](#)

Informações gerais [informações](#)

Chave de partição ID (String)	Chave de classificação -	Modo de capacidade Provisionada	Status da tabela Ativo
Alarmes Nenhum alarme ativo	Recuperação em um ponto anterior no tempo (PITR) Desativado	Política baseada em recursos Não ativo	
Informações adicionais			
Classe da tabela DynamoDB Standard	Índices 0 globais, 0 locais	Stream do DynamoDB Desativado	Tempo de vida (TTL) Desativado
Regiões de replicação 0 Regiões	Criptografia De propriedade da Amazon	Data de criação outubro 10, 2024, 15:12:58 (UTC-03:00)	Proteção contra exclusão Desativado
Integrações informações			
Nome do recurso da Amazon (ARN) arn:aws:dynamodb:us-east-1:123456789012:table/ContactFormDataBase			

6- Atualizar política de acesso da Lambda Function para que possa acessar a tabela DynamoDB

Acesse o painel Lambda e abra a função **ContactFormFunction**

Clique em **Configurações > Permissões** e clique sobre a regra desta função.

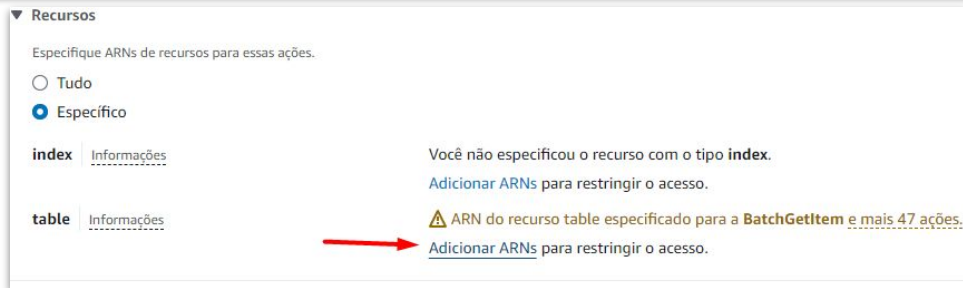
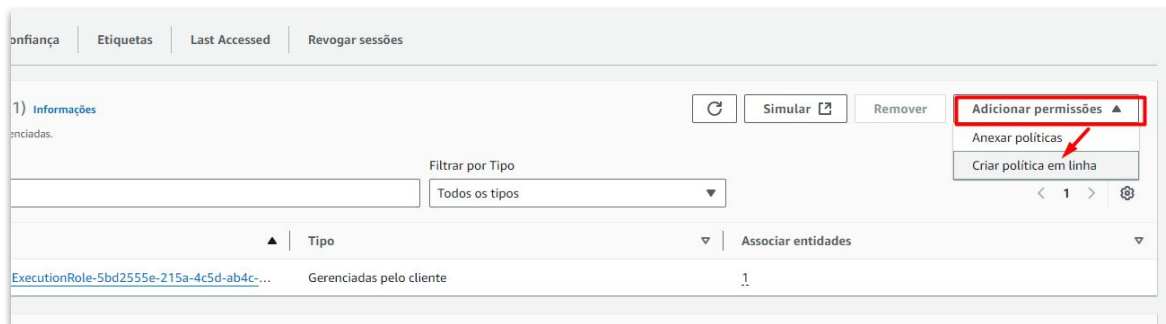
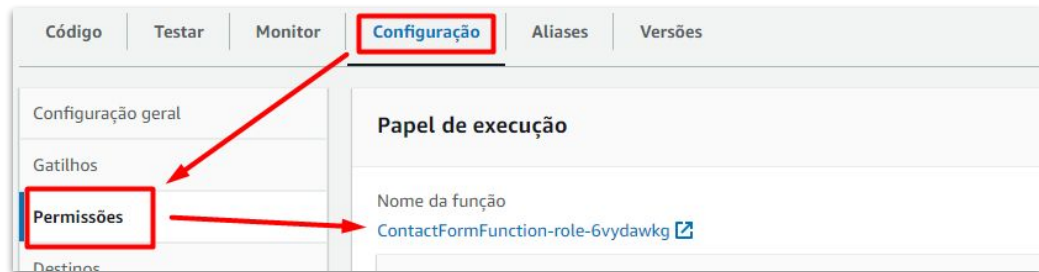
Clique em Adicionar Permissões > Criar Política em Linha

Em serviço selecione **DynamoDB**
Selecione as seguintes ações:
Leitura: **GetItem, Scan, Query**
Gravação: **PutItem, DeleteItem, UpdateItem**

Adicione o ARN da tabela DynamoDB para que essas permissões sejam atribuídas especificamente nesta tabela.
(Utilize o modo texto para colar o ARN)

Clique em **Próximo**

Informe um nome para a política:
Lambda-DynamoDB-ReadWrite e clique em **Criar Política**



6- Atualizar código da lambda function para salvar no DynamoDB

Atualize o código da Lambda Function para [este](#).

Clique em **Deploy** para publicar a versão atualizada

Já é possível testar na própria Lambda Function e também no API Gateway. Em ambos os testes os dados enviados devem ter sido inseridos no DynamoDB.

Itens retornados (3)

<input type="checkbox"/>	ID (String) ▾	email ▾	message ▾	name ▾	phone
<input type="checkbox"/>	add884ae-c88a-40fb-...	<empty>	Teste 2	JOÃO SILVA	34 3249
<input type="checkbox"/>	7df895a0-ad69-4479-...	raphael.rod...	Teste API G...	Raphael Ro...	34 3249
<input type="checkbox"/>	359bbd4d-aa90-4994-...	raphael.rod...	Teste API G...	Raphael Ro...	34 3249

```
import json
import boto3
from time import gmtime, strftime
import uuid

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('ContactFormDataBase')
now = strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())

def lambda_handler(event, context):

    dataText = event['name'] + ' ' + event['email'] + ' ' + event['phone'] + ' ' + event['message']

    id = str(uuid.uuid4())

    response = table.put_item(
        Item={
            'ID' : id,
            'name' : event['name'],
            'email' : event['email'],
            'phone' : event['phone'],
            'message' : event['message'],
            'UpdatedAt' : now
        })

    return {
        'statusCode': 200,
        'body': json.dumps('Dados inseridos: ' + dataText)
    }
```

7- Subir o front-end para o AWS Amplify

Acesse o painel do AWS Amplify e clique em **Deploy an App**

Selecione **Deploy without Git** e clique em Next

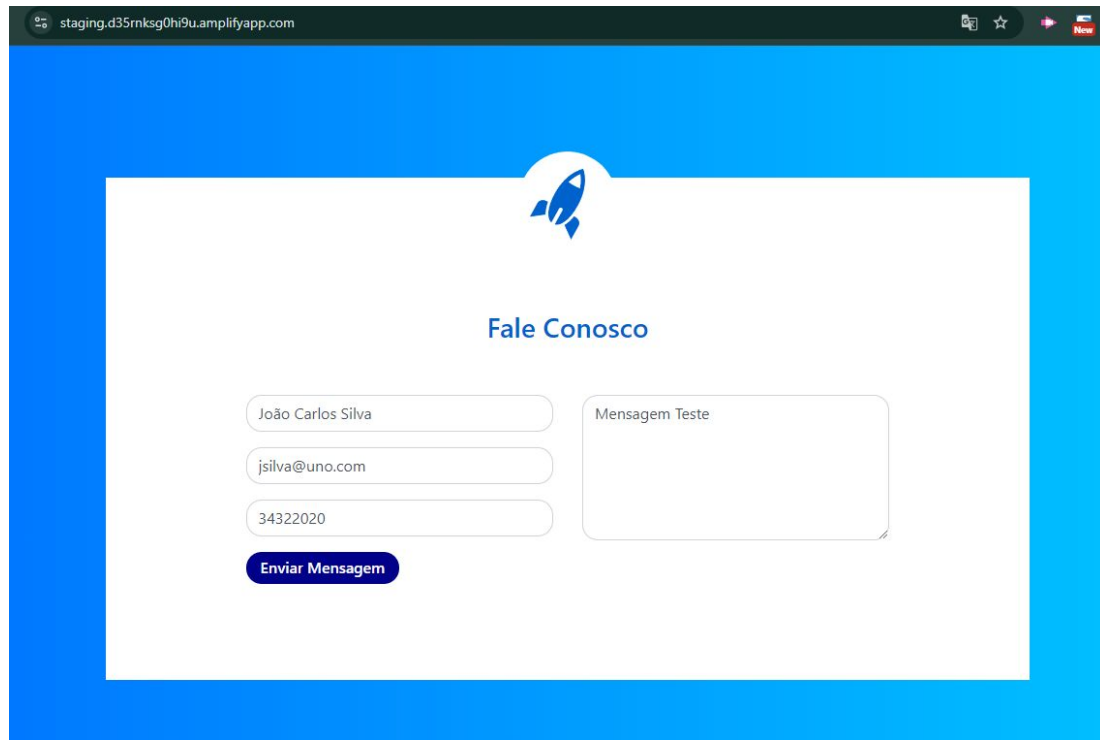
Envie [este arquivo](#) para o Amplify

Clique em **Save and Deploy**

Clique no link fornecido e veja que sua página já está online sem você ter configurado nenhum servidor.

Teste o funcionamento e tente enviar uma mensagem. Se não der resposta, pressione F12 e visualize o log no Console do navegador.

Talvez seja necessário ativar o CORS na API Gateway.



Mais informações sobre API Gateway

- [Entenda API Gateway DO ZERO](#)
- [Guia do Desenvolvedor AWS - API Gateway](#)
- [Configuração do CORS em um API Gateway](#)