



SÃO  
PAULO  
TECH  
SCHOOL

# Arquitetura de soluções em nuvem

**EDA e AWS SNS**

**Eduardo Verri**

[eduardo.verri@sptech.school](mailto:eduardo.verri@sptech.school)

# Events-Driven Architecture

## O que é EDA?

A arquitetura orientada a eventos ou **EDA (Events-Driven Architecture)** é um modelo de arquitetura de software para o design de aplicações. Um sistema orientado a eventos é feito para capturar, comunicar e processar eventos entre serviços desacoplados. Isso significa que os sistemas podem permanecer assíncronos enquanto compartilham informações e realizam tarefas.

## O que é EDA?

É possível criar aplicações orientadas a eventos em qualquer linguagem porque esse tipo de arquitetura é uma abordagem de programação. A arquitetura orientada a eventos permite um pouco de acoplamento, o que a torna uma boa opção para arquiteturas de aplicações distribuídas e modernas.

Esse tipo de arquitetura é **levemente acoplado** porque os produtores de eventos não sabem que os consumidores estão detectando um evento. Além disso, esse evento não sabe as consequências da sua ocorrência.

## O que é um evento?

Um evento atua como um registro de qualquer ocorrência significativa ou mudança de estado do hardware, ou software do sistema. Um evento e uma notificação de evento não são a mesma coisa. Este último é uma mensagem ou notificação enviada pelo sistema para avisar outra parte do sistema que um evento ocorreu.

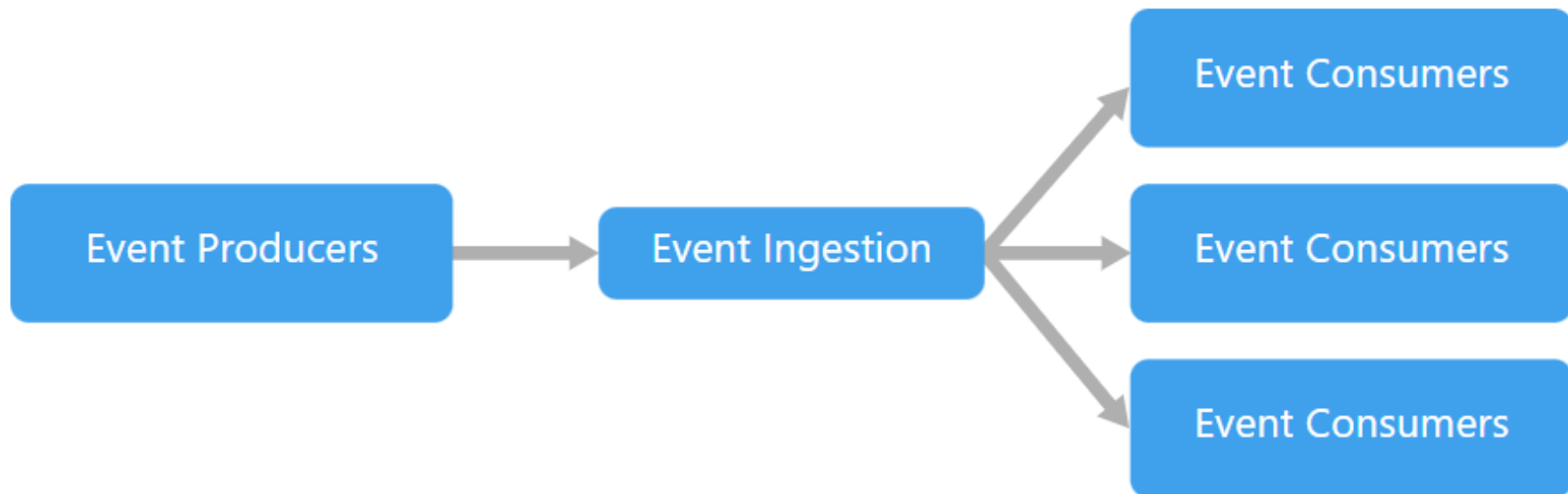
A origem de um evento pode ser a partir de entradas internas ou externas. Os eventos podem ser gerados por um usuário, como clicar no mouse ou apertar uma tecla, uma fonte externa, como uma saída de sensor, ou vir do sistema, como carregar um programa.

## Como funciona a EDA?

- ❑ A arquitetura orientada a eventos é composta de produtores e consumidores de eventos (clientes). Um produtor de eventos detecta ou percebe um evento e o representa como uma mensagem. Devido ao desacoplamento, ele não conhece o consumidor nem o resultado do evento.
- ❑ Após um evento ser detectado, ele é transmitido do produtor para os consumidores por meio de canais, onde uma plataforma de processamento de eventos processa o evento de maneira assíncrona.
- ❑ A plataforma de processamento de eventos executará a resposta correta a um evento e enviará a atividade downstream para os consumidores certos. É nessa atividade downstream que o resultado de um evento é visto.

## Estilo de EDA

Uma arquitetura orientada a eventos consiste em produtores de eventos que geram um fluxo de eventos, consumidores de eventos que escutam esses eventos e canais de eventos que transferem eventos de produtores para consumidores.





## Estilo de EDA

Os eventos são entregues quase em tempo real, para que os consumidores possam responder imediatamente conforme os eventos ocorrem. Os produtores são separados dos consumidores. Um produtor não sabe quais consumidores estão ouvindo. Os consumidores também são separados uns dos outros; e cada consumidor vê todos os eventos. Isso difere do padrão de Consumidores Concorrentes, onde os consumidores removem mensagens de uma fila, e uma mensagem é processada apenas uma vez (presumindo que não haja erros). Em alguns sistemas, como IoT, os eventos devem ser incluídos em volumes muito altos.

# Variações no lado do consumidor

- ❑ **Processamento de eventos simples:** Um evento dispara imediatamente uma ação no consumidor. Por exemplo, você pode usar as Azure Functions com um gatilho de Barramento de Serviço para que uma função seja executada sempre que uma mensagem é publicada em um tópico do Barramento de Serviço.
- ❑ **Correlação básica de eventos:** Um consumidor precisa processar um pequeno número de eventos de negócios discretos, geralmente correlacionados por algum identificador, persistindo algumas informações de eventos anteriores para usar ao processar eventos posteriores.

# Variações no lado do consumidor

- ❑ **Processamento de eventos complexos:** Um consumidor processa uma série de eventos, procurando padrões nos dados de eventos usando uma tecnologia, como o Azure Stream Analytics ou o Apache Storm.
- ❑ **Processamento de fluxo de eventos:** Use uma plataforma de fluxo de dados, como o Hub IoT do Azure ou o Apache Kafka, como um pipeline para ingestão de eventos e os encaminhe para os processadores de fluxo. Os processadores de fluxo agem para processar ou transformar o fluxo. Pode haver vários processadores de fluxo para subsistemas diferentes do aplicativo. Esta abordagem é uma boa opção para cargas de trabalho de IoT.
- ❑ [Estilo de arquitetura controlada por evento - Azure Architecture Center | Microsoft Learn](#)

## Apache Kafka, por exemplo...

O **Apache Kafka** é uma plataforma distribuída de transmissão de dados muito usada para processamento de eventos. Ele consegue lidar com publicação, assinatura, armazenamento e processamento de fluxos de eventos em tempo real.

Há outros gerenciadores de eventos de middleware disponíveis que podem servir de plataforma de processamento de eventos. Como serviços das próprias nuvens. Como o barramento de serviços da Azure...

# Modelos de EDA

A arquitetura orientada a eventos pode ser baseada em um modelo de publicação/subscrição (pub/sub) ou de transmissão de eventos.

- **Modelo pub/sub:** Essa é uma infraestrutura de mensageria baseada em subscrições de um fluxo de eventos. Com esse modelo, após um evento acontecer ou ser publicado, ele é enviado aos clientes que precisam ser informados.
- **Modelo de fluxo de eventos:** Nesse modelo, os eventos são gravados em um log. Consumidores de eventos não se inscrevem em um fluxo de eventos. Em vez disso, eles podem ler o log a partir de qualquer parte do fluxo e ingressar nele a qualquer momento.

## Benefícios da EDA

- **Escalabilidade e falha de maneira independentemente:** Com o desacoplamento dos serviços, eles reconhecem apenas o roteador de eventos, e não uns aos outros. Isso significa que, mesmo que sejam interoperáveis, se um serviço apresentar uma falha, os serviços restantes continuarão em execução. O roteador de eventos atua como um buffer elástico que acomoda picos em workloads.
- **Desenvolvimento com agilidade:** Você não precisa mais escrever código personalizado para pesquisar, filtrar e rotear eventos: o roteador de eventos filtrará e enviará eventos automaticamente aos consumidores. O roteador também elimina a necessidade de coordenação pesada entre serviços produtores e consumidores, acelerando assim o seu processo de desenvolvimento.

## Benefícios da EDA

- **Auditoria com facilidade:** Um roteador de eventos atua como um local centralizado para auditar sua aplicação e definir políticas. Essas políticas podem restringir quem é capaz de publicar e assinar um roteador e controlam quais usuários e recursos têm permissão para acessar seus dados. Você também pode criptografar seus eventos em trânsito e em repouso.
- **Redução dos custos:** Arquiteturas orientadas por eventos são baseadas em push e, portanto, tudo acontece sob demanda à medida que o evento se apresenta no roteador. Dessa forma, você não paga por pesquisas contínuas para verificar a presença de um evento. Isso significa menor consumo de largura de banda da rede, menos utilização de CPU, menos capacidade ociosa da frota e menos handshakes Secure Sockets Layer (SSL)/Transport Layer Security (TLS).

## Quando usar EDA

- **Replicação de dados entre contas e regiões:** Você pode usar uma arquitetura orientada por eventos para coordenar sistemas entre equipes que operam e implantam em diferentes contas e regiões. Ao usar um roteador de eventos para transferir dados entre sistemas, você pode desenvolver, escalar e implantar serviços independentemente de outras equipe.
- **Distribuição e processamento paralelo:** Se você tem muitos sistemas que precisam operar em resposta a um evento, pode usar uma arquitetura orientada por eventos para distribuir o evento sem precisar escrever um código personalizado para envio por push a cada consumidor. O roteador enviará o evento por push para os sistemas, e cada um poderá processar esse evento em paralelo com uma finalidade diferente.



## Quando usar EDA

- **Monitoramento e geração de alertas de estados de recursos:** Em vez de conferir continuamente seus recursos, você pode usar uma arquitetura orientada por eventos para monitorar e receber alertas sobre anomalias, alterações e atualizações. Esses recursos podem incluir buckets de armazenamento, tabelas de banco de dados, funções sem servidor, nós de computação e muito mais.
- **Integração de sistemas heterogêneos:** Se você tem sistemas em execução em pilhas diferentes, pode usar uma arquitetura orientada por eventos para compartilhar informações entre eles sem acoplamento. O roteador de eventos estabelece caminhos indiretos e interoperabilidade entre os sistemas, para que eles possam trocar mensagens e dados ao mesmo tempo que permanecem independentes.

# Casos comuns de EDA –

## Comunicação de microsserviços para back-ends móveis

- Geralmente, sites de varejo ou de mídia e entretenimento precisam aumentar a escala verticalmente para lidar com tráfego variável. Os clientes visitam um site de comércio eletrônico e realizam um pedido. O evento de pedido é enviado a um roteador de eventos. Todos os microsserviços subsequentes podem coletar o evento de pedido para processá-lo. Algumas ações podem incluir: envio do pedido, autorização do pagamento e transmissão dos detalhes do pedido para uma transportadora.
- Como cada um dos microsserviços pode ter escalabilidade e apresentar falhas de maneira independente, o processo pode aumentar a escala verticalmente durante os períodos de pico de pedidos sem apresentar pontos únicos de falha.

# Casos comuns de EDA –

## Automação do fluxo de trabalho empresarial

- Muitos fluxos de trabalho empresariais, como transações de serviços financeiros, requerem a repetição das mesmas etapas. Você pode instaurar e automatizar essas etapas com a arquitetura orientada a eventos (EDA).
- Por exemplo, quando um cliente solicita abertura de uma nova conta em um banco, o banco deve executar algumas verificações de dados (como: identificação, endereço etc.). Algumas contas também exigirão um estágio de aprovação humana. Você pode orquestrar todas essas etapas por meio de um serviço de fluxo de trabalho que executa as etapas automaticamente quando solicitações de aberturas de contas são recebidas.
- Também é possível adicionar um fluxo de trabalho para processar dados da solicitação do cliente de forma assíncrona utilizando machine learning para extração de dados relevantes, o que economizará horas de coleta e validação manual de dados.

# Casos comuns de EDA –

## Integração de aplicações SaaS

- Um dos principais desafios para ambientes de software como serviço (SaaS) é a falta de visibilidade da atividade e dos dados do usuário. Para liberar dados em silos, as arquiteturas orientadas a eventos podem ingerir eventos de aplicações SaaS ou enviar eventos para suas aplicações SaaS. Por exemplo, você pode criar um middleware para ingerir dados de pedidos de parceiros que são recebidos e enviar os pedidos diretamente para uma aplicação interna de processamento de pedidos.

# Casos comuns de EDA –

## Automação de infraestrutura

- Ao executar workloads com uso intensivo de computação (como análises financeiras, pesquisas genômicas ou transcódificações de mídia), você pode precisar atender aos recursos de computação aumentando o processamento em paralelo e, então, reduzindo-o após a conclusão do trabalho.
- Por exemplo, em setores altamente regulamentados, as empresas com EDA podem ativar recursos de postura de segurança em resposta a um incidente ou tomar medidas de correção sempre que uma política de segurança enviar um evento de alerta.

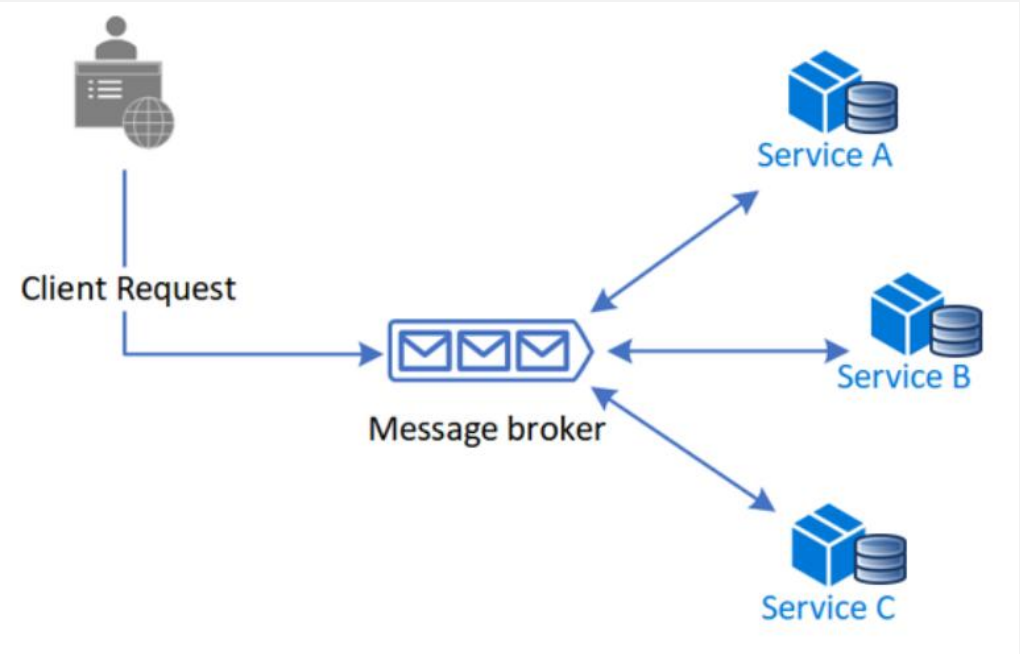
# Desafios da EDA

- **Entrega garantida:** Em alguns sistemas, especialmente em situações de IoT, é crucial garantir que os eventos sejam entregues.
- **Eventos processados em ordem ou exatamente uma vez:** Normalmente, cada tipo de consumidor é executado em várias instâncias de resiliência e escalabilidade. Isso pode criar um desafio se os eventos precisarem ser processados em ordem (dentro de um tipo de consumidor) ou se a lógica de processamento de mensagens idempotentes não estiver implementada.
- **Coordenar mensagens entre serviços:** Os processos empresariais geralmente envolvem vários serviços que publicam e assinam mensagens para obter um resultado consistente em toda uma carga de trabalho. Padrões de fluxo de trabalho, como o padrão **Coreografia** e a **Orquestração de Saga**, podem ser usados para gerenciar de maneira confiável os fluxos de mensagens em vários serviços.

# Coreografia

Na abordagem coreográfica, os serviços trocam eventos sem um controlador centralizado. Com a coreografia, cada transação local publica eventos de domínio que disparam transações locais em outros serviços.

[Padrão de coreografia - Azure Architecture Center | Microsoft Learn](#)

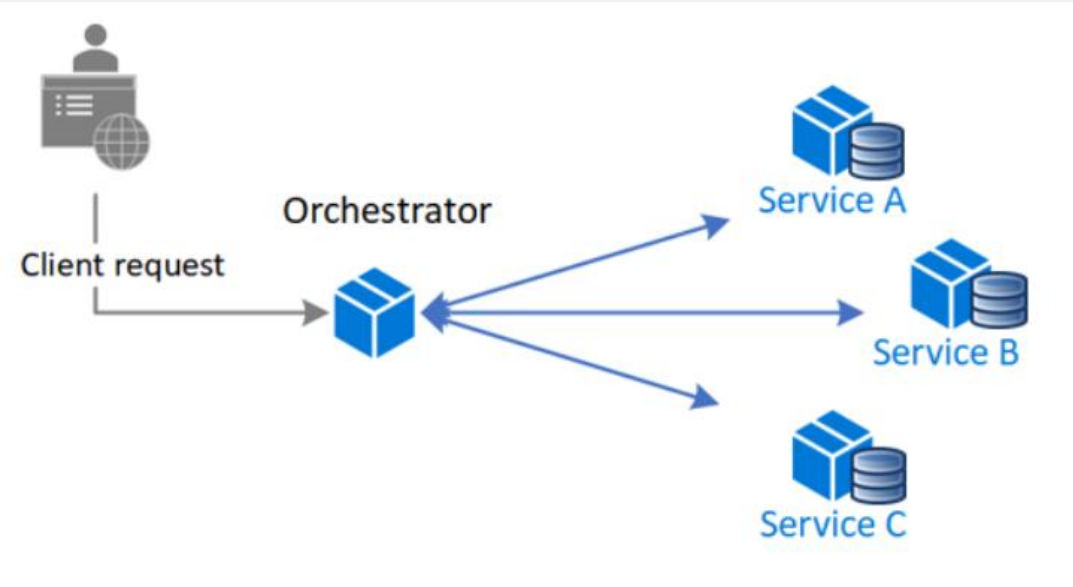


Benefícios da coreografia	Desvantagens da coreografia
Bom para fluxos de trabalho simples que têm poucos serviços e não precisam de uma lógica de coordenação.	O fluxo de trabalho pode ser confuso quando você adiciona novas etapas. É difícil rastrear a quais comandos cada participante da saga responde.
Nenhum outro serviço é necessário para coordenação.	Há um risco de dependência cíclica entre os participantes da saga porque eles têm que consumir os comandos uns dos outros.
Não introduz um único ponto de falha porque as responsabilidades são distribuídas entre os participantes da saga.	O teste de integração é difícil porque todos os serviços devem ser executados para simular uma transação.

# Orquestração de saga

O padrão Saga gerencia transações dividindo-as em uma sequência de transações locais .

[Padrão de design de saga - Azure Architecture Center | Microsoft Learn](#)  
[Workflows • NServiceBus • Particular Docs](#)



Benefícios da orquestração	Desvantagens da orquestração
Mais adequado para fluxos de trabalho complexos ou quando você adiciona novos serviços.	Outra complexidade de design requer uma implementação de uma lógica de coordenação.
Evita dependências cíclicas porque o orquestrador gerencia o fluxo.	Apresenta um ponto de falha porque o orquestrador gerencia o fluxo de trabalho completo.
A separação clara de responsabilidades simplifica a lógica do serviço.	



# Desafios da EDA

- **Tratamento de erro:** A arquitetura orientada por eventos usa principalmente a comunicação assíncrona. O desafio da comunicação assíncrona é o tratamento de erros. Uma maneira de resolver esse problema é usar um processador de tratamento de erros separado. Assim, quando o consumidor de eventos apresenta um erro, ele envia imediatamente, e de forma assíncrona, o evento incorreto para o processador de tratamento de erros e segue em frente. O processador do manipulador de erros tenta corrigir o erro e envia o evento de volta ao canal de ingestão original. Mas se o processador de tratamento de erros falhar, ele poderá enviar o evento incorreto a um administrador para inspeção adicional. Se você usar um processador de tratamento de erros, os eventos incorretos serão processados fora de sequência quando forem reenviados.

# Desafios da EDA

- **Perda de dados:** Se algum dos componentes falhar antes de processar e entregar com êxito o evento ao próximo componente, o evento será descartado e nunca chegará ao destino final. Para minimizar a chance de perda de dados, persista os eventos em trânsito e remova ou remova os eventos da fila somente quando o próximo componente confirmar o recebimento do evento. Esses recursos geralmente são conhecidos como modo de confirmação do cliente e suporte ao último participante.
- **Implementando um padrão tradicional de solicitação-resposta:** Às vezes, o produtor do evento exige uma resposta imediata do consumidor do evento, como obter a qualificação de um cliente antes de prosseguir com um pedido. Na arquitetura orientada a eventos, a comunicação síncrona pode ser obtida por meio de mensagens de solicitação-resposta. Esse padrão geralmente é implementado utilizando várias filas - uma fila de solicitação e uma fila de resposta.

# Desafios da EDA

- **Manter o número apropriado de eventos:** A geração de um número excessivo de eventos refinados pode saturar e sobrecarregar o sistema, dificultando a análise eficaz do fluxo geral de eventos. Esse problema é exacerbado quando as alterações precisam ser revertidas. Por outro lado, a consolidação excessiva de eventos também pode criar problemas, resultando em processamento e respostas desnecessários dos consumidores de eventos.

**AWS SNS**

## O que é o AWS SNS?

O Amazon Simple Notification Service (Amazon SNS) é um serviço totalmente gerenciado que fornece entrega de mensagens de editores (produtores) para assinantes (consumidores). Os publicadores se comunicam de maneira assíncrona com os assinantes produzindo e enviando mensagens para um tópico, que é um canal de comunicação e um ponto de acesso lógico.

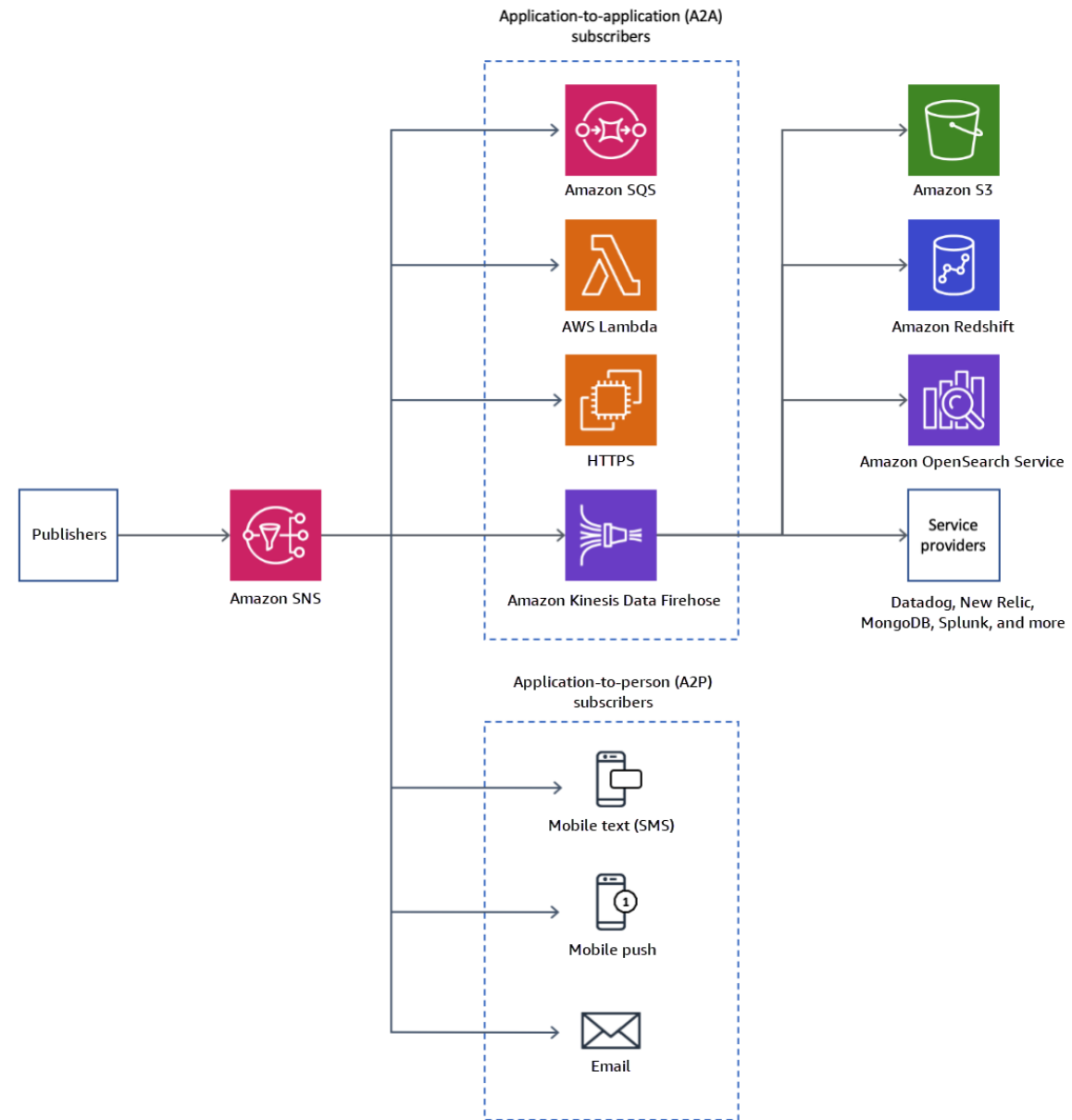
## Como funciona o AWS SNS?

No SNS, os editores enviam mensagens para um tópico, que funciona como um canal de comunicação. O tópico atua como um ponto de acesso lógico, garantindo que as mensagens sejam entregues a vários assinantes em diferentes plataformas.

Os assinantes de um tópico do SNS podem receber mensagens por meio de diferentes endpoints, dependendo do caso de uso, como:

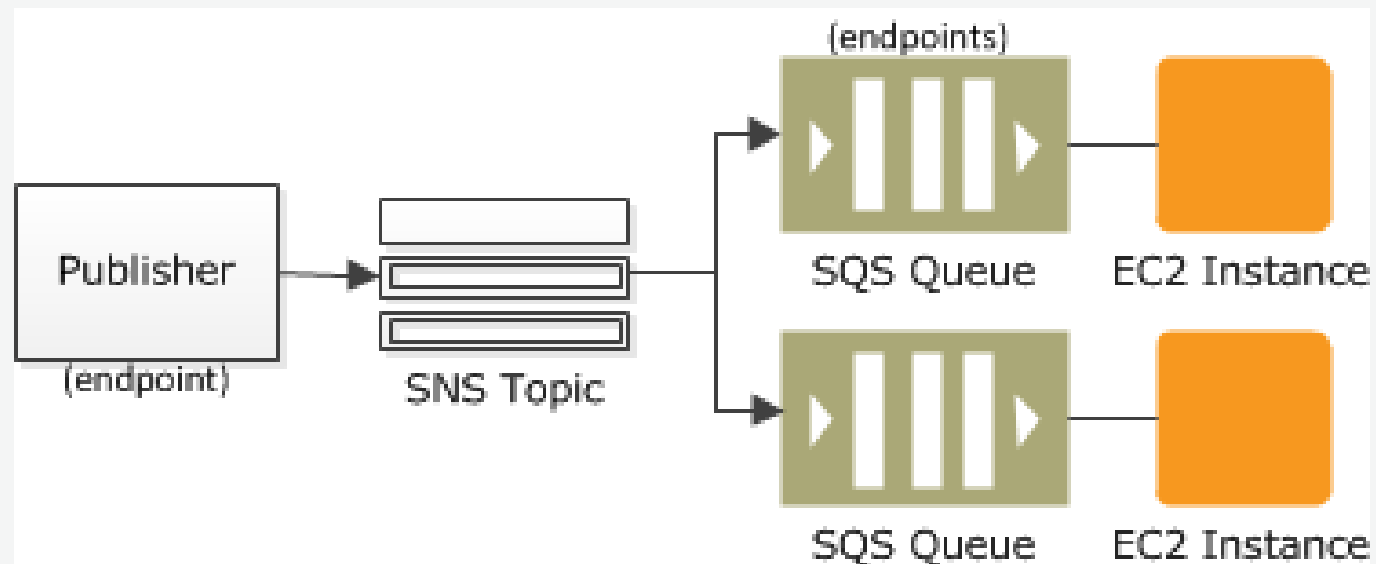
- Amazon SQS

- Lambda
- Pontos de extremidade HTTP(S)
- E-mail
- Notificações por push para dispositivos móveis
- Mensagens de texto móveis (SMS)
- Amazon Data Firehose
- Provedores de serviços (por exemplo, Datadog, MongoDB, Splunk)



## Cenários comuns de uso

- ❑ **Integração de aplicações:** O cenário de *Fanout* é quando uma mensagem publicada em um tópico do SNS é replicada e enviada para vários endpoints, como fluxos de entrega do Firehose, filas do Amazon SQS, endpoints HTTP(S) e funções do Lambda. Isso permite o processamento paralelo assíncrono.





## Cenários comuns de uso

- ❑ **Alertas:** Alertas de aplicações e do sistema são notificações que são acionadas por limites predefinidos. O Amazon SNS pode enviar essas notificações para usuários especificados via SMS e e-mail.
- ❑ **Notificações ao usuário:** O Amazon SNS pode enviar mensagens de e-mail de push e mensagens de texto (mensagens SMS) para indivíduos ou grupos e notificações por push para dispositivos móveis permitem que você envie mensagens diretamente para aplicativos móveis.

**Lab . AWS SNS**

## Criando um tópico

- ❑ Um tópico do Amazon SNS é um ponto de acesso lógico que atua como um canal de comunicação. Um tópico permite agrupar vários endpoints (como Amazon SQS AWS Lambda, HTTP/S ou um endereço de e-mail).
- ❑ Para transmitir as mensagens de um sistema produtor de mensagem (por exemplo, um site de comércio eletrônico) que trabalha com vários outros serviços que exigem suas mensagens (por exemplo, sistemas de cumprimento e checkout), crie um tópico para o sistema produtor.



## Simple Notification Service

SNS managed message topics for Pub/Sub



Amazon SNS > Tópicos



### Amazon SNS



Painel

**Tópicos**

Assinaturas

#### ▼ Mobile

Notificações por push

Mensagens de texto (SMS)

## Tópicos (2)

Editar

Excluir

Publicar mensagem

Criar tópico

Q Pesquisar

< 1 > ⚙

	Nome	Tipo	ARN
<input type="radio"/>	<a href="#">email-sender</a>	Padrão	arn:aws:sns:us-east-1:344665194032:email-se...
<input type="radio"/>	<a href="#">RedshiftSNS</a>	Padrão	arn:aws:sns:us-east-1:344665194032:Redshift...

# Criar tópico

## Detalhes

### Tipo | [Informações](#)

Não é possível modificar o tipo de tópico após a criação do tópico

☐ FIFO (primeiro a entrar, primeiro a sair)

- Ordenação de mensagens rigorosamente preservada
- Entrega de mensagens exatamente uma vez
- Protocolos de assinatura: SQS

☒ Padrão

- Ordenação de mensagens com melhor esforço
- Entrega de mensagens pelo menos uma vez
- Protocolos de assinatura: SQS, Lambda, Data Firehose, HTTP, SMS, e-mail, endpoints de aplicativos móveis

### Nome

meu-topico-01

Máximo de 256 caracteres. Pode incluir caracteres alfanuméricos, hifens (-) e sublinhados (\_).

### Nome de exibição - *opcional* | [Informações](#)

Para usar esse tópico com inscrições de SMS, insira um nome de exibição. Somente os primeiros 10 caracteres são exibidos em uma mensagem SMS.

meu-topico-01

Máximo de 100 caracteres.

## Criando uma assinatura em um tópico

- ❑ Para receber mensagens publicadas em um tópico, você precisa inscrever um endpoint nesse tópico. Depois de inscrito, o endpoint começa a receber todas as mensagens publicadas no tópico associado.
- ❑ Inscrever um endpoint em um tópico do Amazon SNS permite a entrega de mensagens para o endpoint especificado, garantindo que os sistemas ou usuários certos recebam notificações quando uma mensagem é publicada no tópico. Essa etapa é essencial para vincular o tópico aos consumidores, sejam eles aplicativos, destinatários de e-mail ou outros serviços, permitindo uma comunicação perfeita entre sistemas.



## Amazon SNS

Painel

Tópicos

Assinaturas

### ▼ Mobile

Notificações por push

Mensagens de texto (SMS)

## Assinaturas (2)

Editar

Excluir

Solicitar confirmação

Confirmar assinatura

Criar assinatura

🔍 Pesquisar

< 1 > ⚙️

	ID ▲	Endpoint ▼	Status ▼	Protocolo
<input type="radio"/>	<a href="#">47480e95-890c-4297-bfcf-54...</a>	eduardo.verri@sptech.school	✅ Confirmado	EMAIL
<input type="radio"/>	<a href="#">88ccafb1-d6c6-4ad5-902d-5a...</a>	arn:aws:lambda:us-east-1:344...	✅ Confirmado	LAMBDA

# Criar assinatura

## Detalhes

### ARN do tópico

arn:aws:sns:us-east-1:344665194032:meu-topico-01



### Protocolo

O tipo de endpoint a ser inscrito

E-mail



### Endpoint

Um endereço de e-mail que pode receber notificações do Amazon SNS.

eduardo.verri@sptech.school





meu-topico-01<no-reply@sns.amazonaws.com>

Para: Eduardo Verri



Dom, 13/04/2025 22:09



Esta mensagem está em Inglês

Traduzir para o Português (Brasil)

Nunca traduzir do Inglês

You have chosen to subscribe to the topic:

**arn:aws:sns:us-east-1:344665194032:meu-topico-01**

To confirm this subscription, click or visit the link below (If this was in error no action is necessary):

[Confirm subscription](#)

Please do not reply directly to this email. If you wish to remove yourself from receiving all future SNS subscription confirmation requests please send an email to [sns-opt-out](#)



Responder



Encaminhar

## Assinaturas (1)

Editar

Excluir

Solicitar confirmação

Confirmar assinatura

Criar assinatura

Q Pesquisar

< 1 > ⚙

	ID ▲	Endpoint ▼	Status ▼	Protocolo ▼
<input type="radio"/>	<a href="#">60270278-ffa1-4899-973e-feea36ca...</a>	eduardo.verri@sptech.school	✔ Confirmado	EMAIL

# Publicar uma mensagem no AWS SNS

- ❑ Depois de criar um tópico do Amazon SNS e inscrever um endpoint nele, é possível publicar mensagens no tópico. Quando uma mensagem é publicada, o Amazon SNS tenta entregar a mensagem aos endpoints inscritos.



Amazon SNS > Tópicos > meu-topico-01



## Amazon SNS



Painel

**Tópicos**

Assinaturas

### ▼ Mobile

Notificações por push

Mensagens de texto (SMS)

## meu-topico-01

Editar

Excluir

Publicar mensagem

### Detalhes

#### Nome

meu-topico-01

#### Nome de exibição

meu-topico-01

#### ARN

arn:aws:sns:us-east-1:344665194032:meu-topico-01

#### Proprietário do tópico

344665194032

#### Tipo

Padrão



**Assinaturas**

Política de acesso

Política de proteção de dados

Política de entrega (HTTP/S)

Reg



## Publicar mensagem no tópico

### Detalhes da mensagem

**ARN do tópico**  
arn:aws:sns:us-east-1:344665194032:meu-topico-01

**Assunto - *opcional***

Teste AWS SNS

Máximo de 100 caracteres ASCII imprimíveis

**Vida útil (TTL) - *opcional*** | [Informações](#)  
Essa configuração se aplica apenas aos endpoints de aplicativos móveis. Ela se refere ao número de segundos que o serviço de notificação por push tem para entregar a mensagem para o endpoint.

### Corpo da mensagem

**Estrutura da mensagem**

☒ **Carga idêntica para todos os protocolos de entrega.**  
A mesma carga é enviada a endpoints inscritos no tópico, independentemente de seus protocolos de entrega.

☐ **Carga personalizada para cada protocolo de entrega.**  
Diferentes cargas são enviadas a endpoints inscritos no tópico, com base em seus protocolos de entrega.

**O corpo da mensagem a ser enviada ao endpoint**

1

Olá SNS

**ARN do tópico**  
arn:aws:sns:us-east-1:344665194032:meu-topico-01

**Assunto - *opcional***

## Teste AWS SNS

Máximo de 100 caracteres ASCII imprimíveis

Vida útil (TTL) - *opcional* | Informações

Essa configuração se aplica apenas aos endpoints de aplicativos móveis. Ela se refere ao número de segundos que o serviço de notificação por push tem para entregar a mensagem para o endpoint.

10/10

Carga idêntica para todos os protocolos de entrega.  
A mesma carga é enviada a endpoints inscritos no tópico, independentemente de seus protocolos de entrega.

Carga personalizada para cada protocolo de entrega.  
Diferentes cargas são enviadas a endpoints inscritos no tópico, com base em seus protocolos de entrega.

O corpo da mensagem a ser enviada ao endpoint

1

Olá SNS

## Estrutura da mensagem

- ☒ **Carga idêntica para todos os protocolos de entrega.**  
A mesma carga é enviada a endpoints inscritos no tópico, independentemente de seus protocolos de entrega.
  - ☐ **Carga personalizada para cada protocolo de entrega.**  
Diferentes cargas são enviadas a endpoints inscritos no tópico, com base em seus protocolos de entrega.

### O corpo da mensagem a ser enviada ao endpoint

1 Olá SNS



## Teste AWS SNS



meu-topico-01<no-reply@sns.amazonaws.com>

Para: ○ Eduardo Verri



Dom, 13/04/2025 22:15

Olá SNS

--

If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:

<https://sns.us-east-1.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:us-east-1:344665194032:meu-topico-01:60270278-ffa1-4899-973e-fee36caf6a8&Endpoint=eduardo.verri@sptech.school>

Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at

<https://aws.amazon.com/support>



Responder



Encaminhar


# Associar o lambda ao SNS


## etl-function-20250405


▼ Visão geral da função [Informações](#)

Diagrama


Modelo

 etl-function-20250405

 Layers (1)

 S3

+ Adicionar gatilho



+ Adicionar destino

## Adicionar destino

### Configuração de destino [Informações](#)

Configure um destino para receber registros de invocação. O Lambda pode enviar registros quando sua função é invocada de maneira assíncrona ou quando sua função processa registros de um mapeamento de origem de eventos.

#### Origem

Escolha o tipo de invocação ao qual o Lambda envia registros.

- ☒ Invocação assíncrona
- ☐ Invocação do mapeamento da fonte de eventos

#### Condição

Escolha se deseja enviar registros de invocação para falhas no processamento de eventos ou para invocações bem-sucedidas.

- ☐ Em caso de falha
- ☒ Em caso de sucesso

#### Tipo de destino

Escolha o tipo de destino ao qual o Lambda envia registros de invocação.

Tópico do SNS ▼

#### Destino

Escolha o ARN do destino ou insira o ARN manualmente.

🔍



**Agora só testar o trigger!**



## AWS Notification Message




meu-topico-01<no-reply@sns.amazonaws.com>

Para: Eduardo Verri



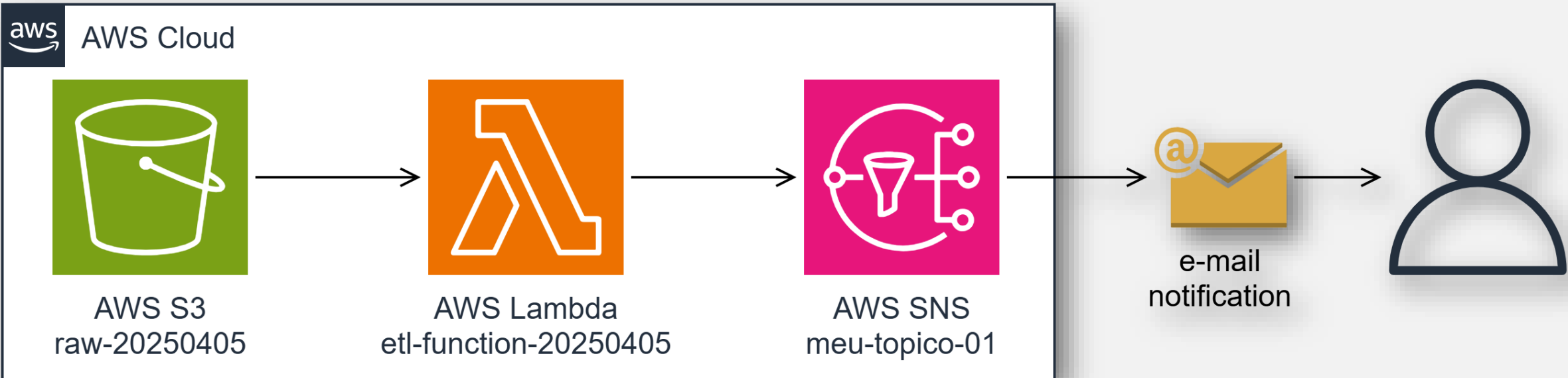
Dom, 13/04/2025 22:23

 Esta mensagem está em Inglês

[Traduzir para o Português \(Brasil\)](#)

[Nunca traduzir do Inglês](#)

```
{"version":"1.0","timestamp":"2025-04-14T01:22:57.052Z","requestContext":{"requestId":"1d524057-1bfe-416e-a910-b9164cb9b168","functionArn":"arn:aws:lambda:us-east-1:344665194032:function:etl-function-20250405:$LATEST","condition":"Success","approximateInvokeCount":1},"requestPayload":{"Records":[{"eventVersion":"2.1","eventSource":"aws:s3","awsRegion":"us-east-1","eventTime":"2025-04-14T01:22:50.021Z","eventName":"ObjectCreated:Put","userIdentity":{"principalId":"AWS:AROAVAP5FPYYFVFO6PB4V:user3877795=Testar_aluno"},"requestParameters":{"sourceIPAddress":"201.95.218.187"},"responseElements":{"x-amz-request-id":"SYGK61Y9YD9FQE3C","x-amz-id-2":"PawKD4RGbzXPliA7IbJEmPq+3FGDzAQXTCMO/yeLNKA EaA4NQSN4ESEBooavknzSQR e/P/zxa6zT36cBlvjxxsSR0feg0PX1"},"s3":{"s3SchemaVersion":"1.0","configurationId":"b67f7660-d2f4-4a82-9b3c-eedce112e6fd","bucket":{"name":"raw-20250405","ownerIdentity":{"principalId":"A13B6LSC3L65HX"},"arn":"arn:aws:s3:::raw-20250405"},"object":{"key":"base01.csv","size":24769,"eTag":"4a600f6c287b3b7acb7eaf58fa3d10d7","sequencer":"0067FC6369F17108B6"}}}]}},"responseContext":{"statusCode":200,"executedVersion":"$LATEST"},"responsePayload":null}
```



# Lab. AWS SNS – Pompa e circunstância

## ▼ Visão geral da função

Informações

Exportar para o Infrastructure Composer

Fazer download ▼

Diagrama

Modelo



etl-function-  
20250405



Layers

(1)



S3

+ Adicionar gatilho

+ Adicionar destino


### Descrição

-

### Última modificação

há 1 semana

### ARN da função

 arn:aws:lambda:us-east-1:344665194032:function:  
etl-function-20250405

### URL da função

Informações

-

**Retirado o destino via SNS pois  
vamos atualizar no código!**

**Assim podemos formatar a  
mensagem!**

```
sns_client = boto3.client('sns')

def lambda_handler(event, context):
    ''' ◆ Envio de mensagem personalizada ao SNS '''
    resultado = {
        "status": "Sucesso",
        "usuario": "eduardo.verri@sptech.school",
        "ação": "processamento de dados"
    }
    mensagem_formatada = f"""
    Olá! A função Lambda foi executada com sucesso.
    Detalhes:
    - Status: {resultado['status']}
    - Usuário: {resultado['usuario']}
    - Ação executada: {resultado['ação']}

    Att, seu sistema automatizado 😊
    """
    response = sns_client.publish(
        TopicArn='arn:aws:sns:us-east-1:344665194032:meu-topico-01',
        Message=mensagem_formatada,
        Subject='[Notificação Lambda] Sucesso na execução'
    )

    return {
        'statusCode': 200,
        'body': json.dumps('Notificação enviada com sucesso!')
    }
```



meu-topico-01<no-reply@sns.amazonaws.com>

Para: ☒ Eduardo Verri



Esta mensagem está em Inglês

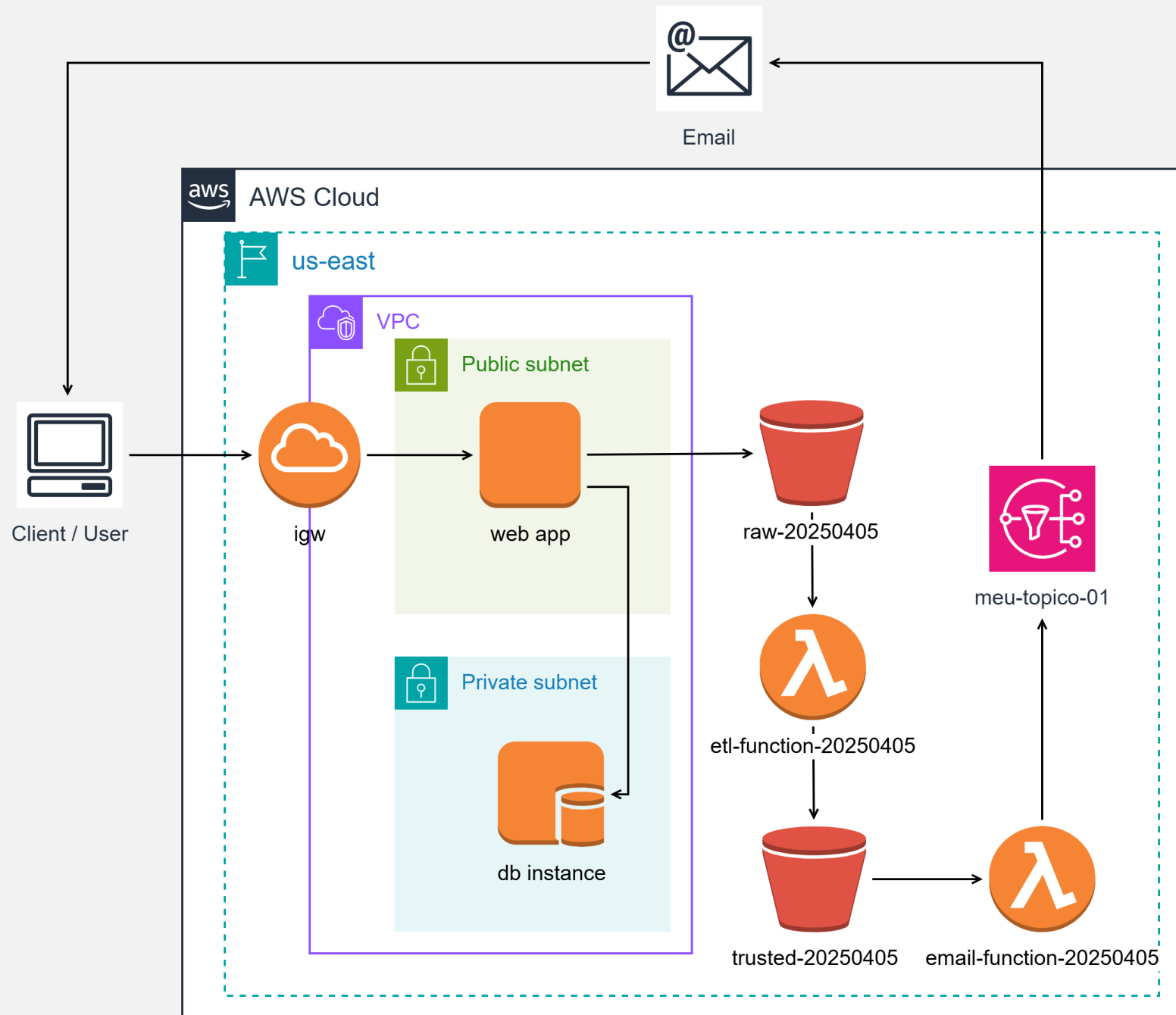
Olá! A função Lambda foi executada com sucesso.

Detalhes:

- Status: Sucesso
- Usuário: eduardo.verri@sptech.school
- Ação executada: processamento de dados

Att, seu sistema automatizado 😊

**Arquitetura da solução – sprint 02**





**Agradeço**  
a sua atenção!



SÃO  
PAULO  
TECH  
SCHOOL