Modernizando aplicações financeiras usando orquestração e coreografia de microserviços

Paulo Aragão, *Senior Solutions Architect, AWS*

“Se a sua aplicação é *cloud-native*, de larga escala, ou distribuída, e não inclui um componente de mensageria, isso é um bug.”

~ Tim Bray

formerly AWS Messaging, Workflow Management

aws

Potencial problema de sistemas síncronos

Sistemas síncronos são altamente acoplados

Um problema em uma dependência no fluxo tem impacto imediato nas funções que a chamaram

Novas tentativas dessas funções podem facilmente amplificar o problema

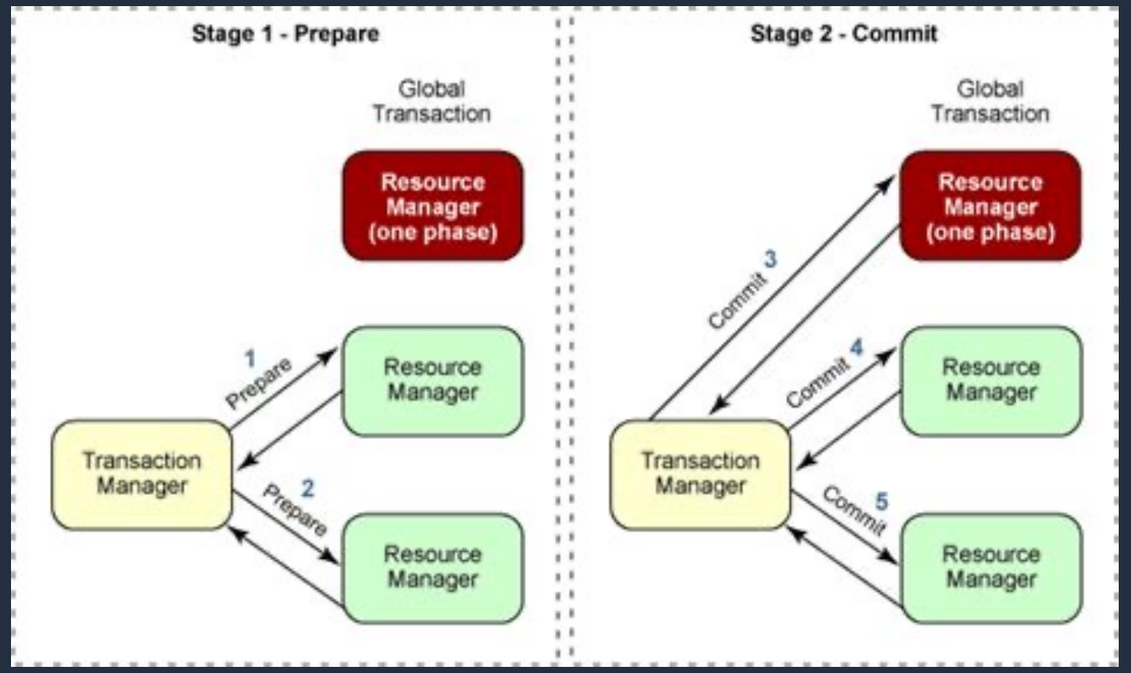
Photo: Dirk Fröhner

O que é o "*Global Transaction*" ou "*2-Phase Commit*"?

Fase 1: Preparar

Microserviços enviam uma "previa" da sua chamada

Depende de um coordenador da transação



Fase 2: Commit

Sucesso: transação é completada

Falha: coordenador notifica todos microserviços a fazer rollback

aws

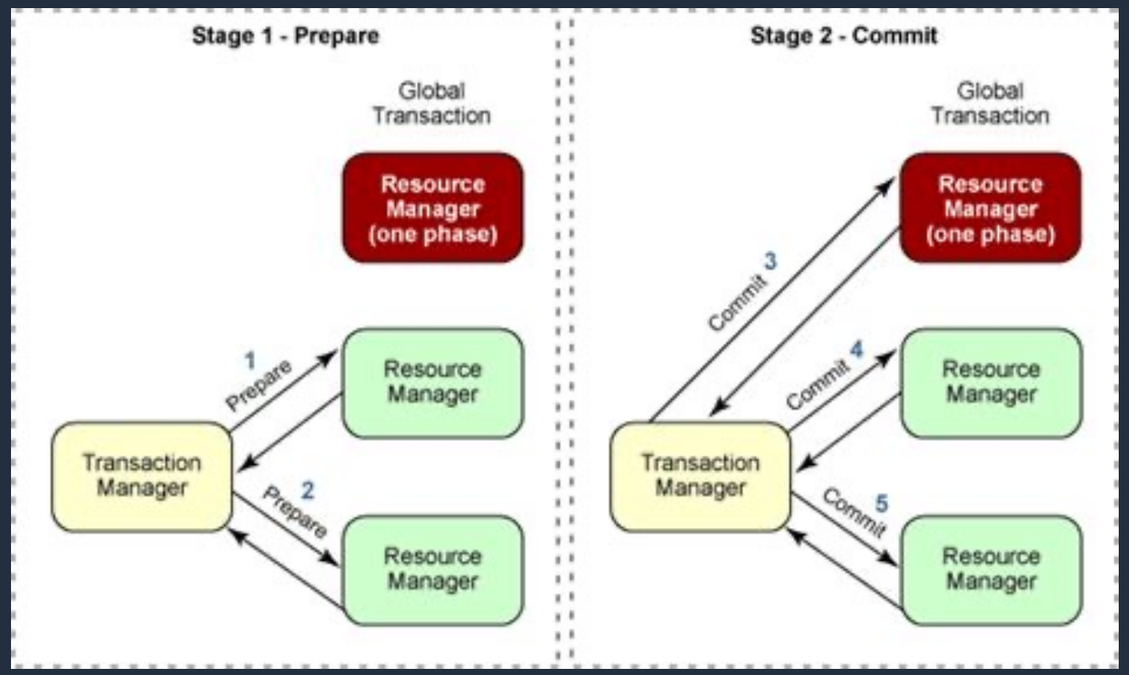
O que é o "*Global Transaction*" ou "*2-Phase Commit*"?

Vantagens:

ACID

- Atomicidade
- Consistência
- Isolamento
- Durabilidade

Transação global



Desvantagens:

Lentidão

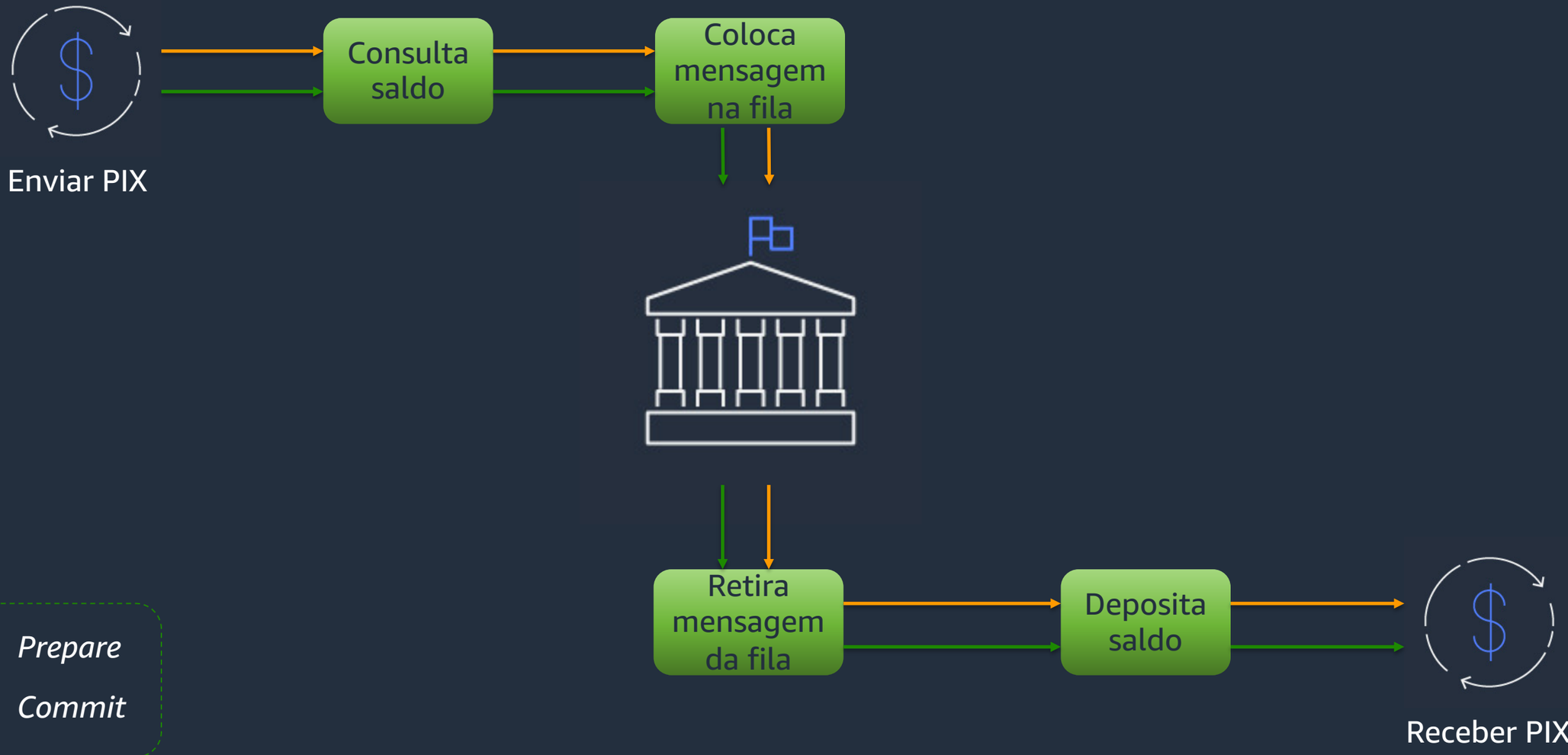
Altamente acoplado ao coordenador

Lock de transação no banco de dados

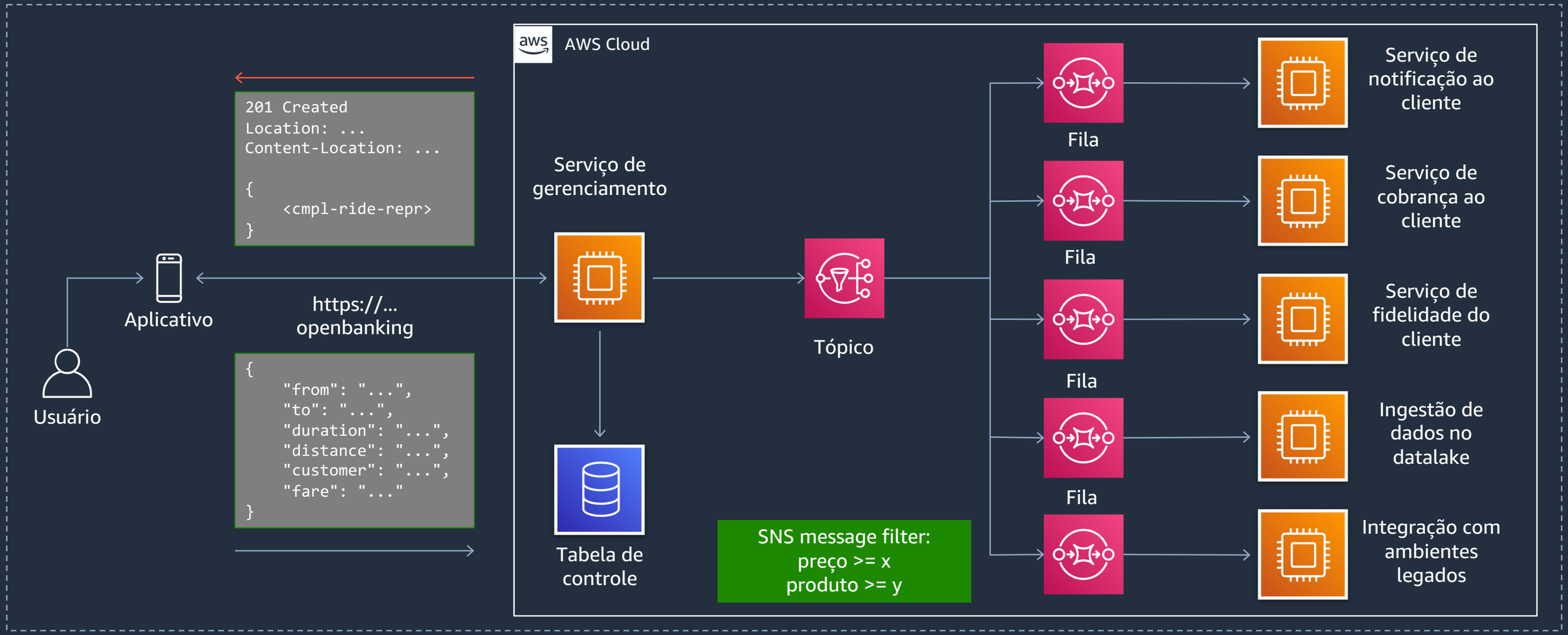
Dificuldade para escalar grandes volumes

aws

O que é o "*Global Transaction*" ou "*2-Phase Commit*"?

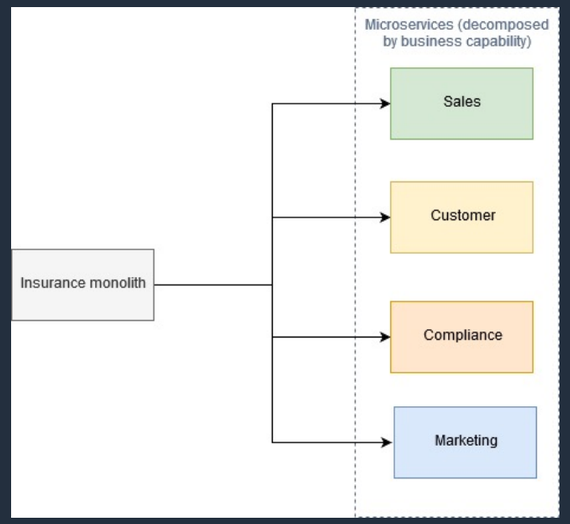


Aplicativo ~~completo~~ de finanças

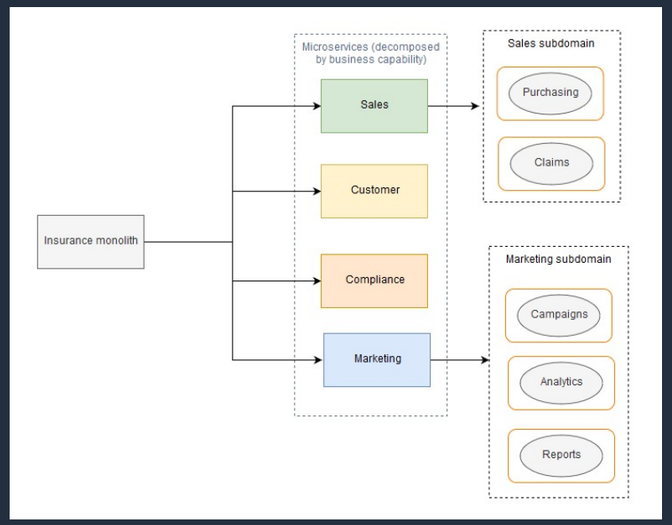


Como modernizar essas aplicações?

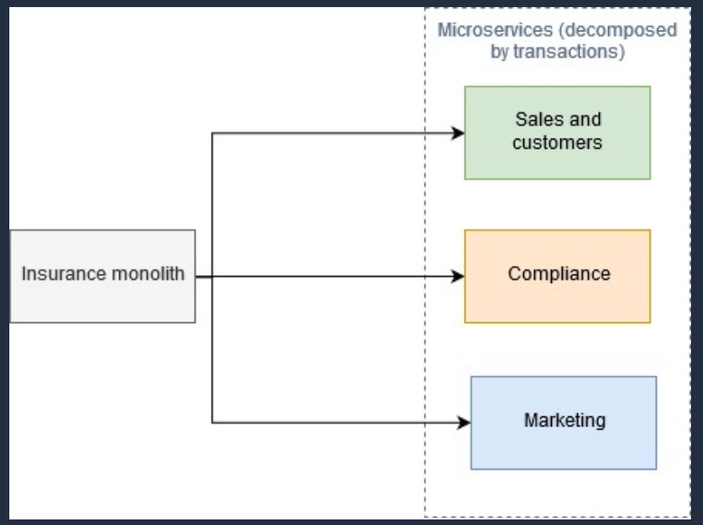
Decomposição de microserviços



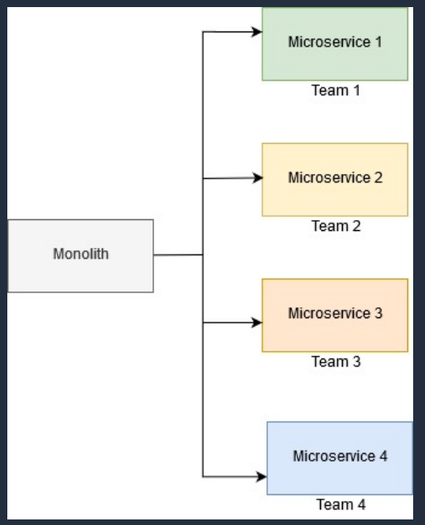
Capacidades do negócio



Subdomínios



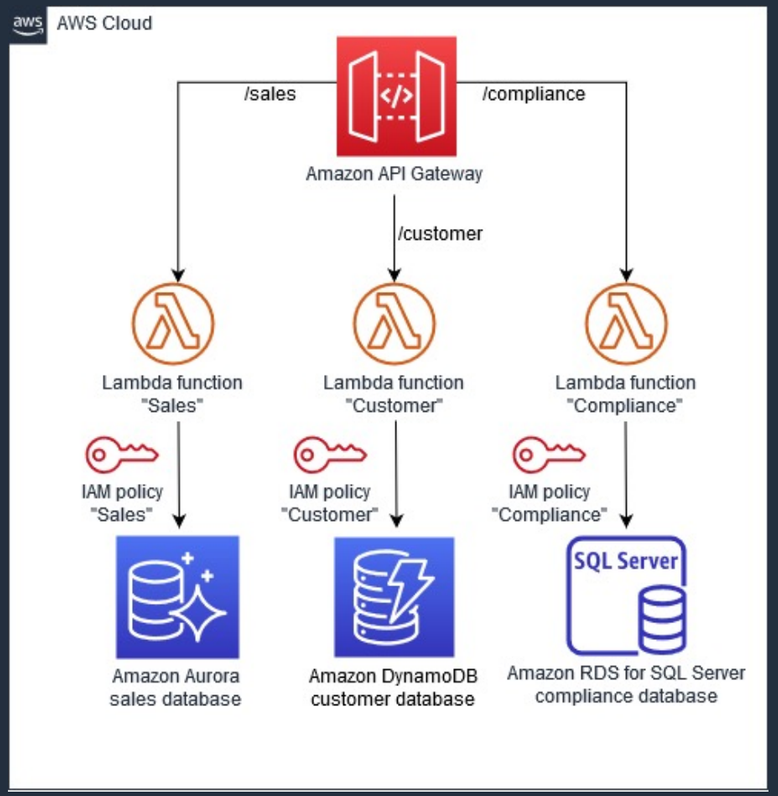
Transações



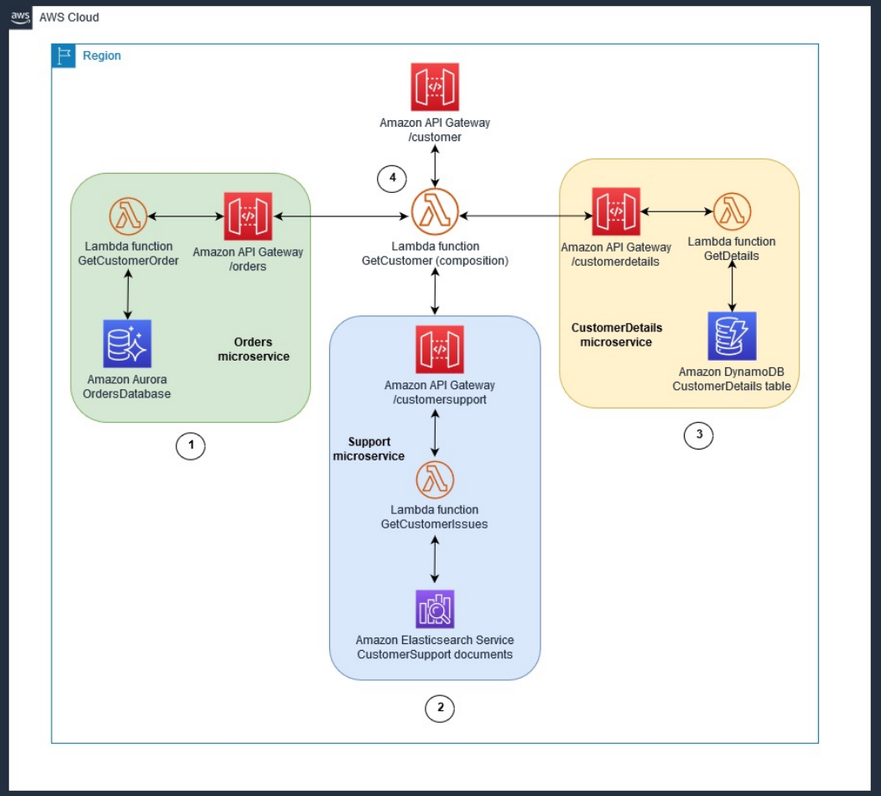
Padrões de times

aws

Persistência de dados em microserviços

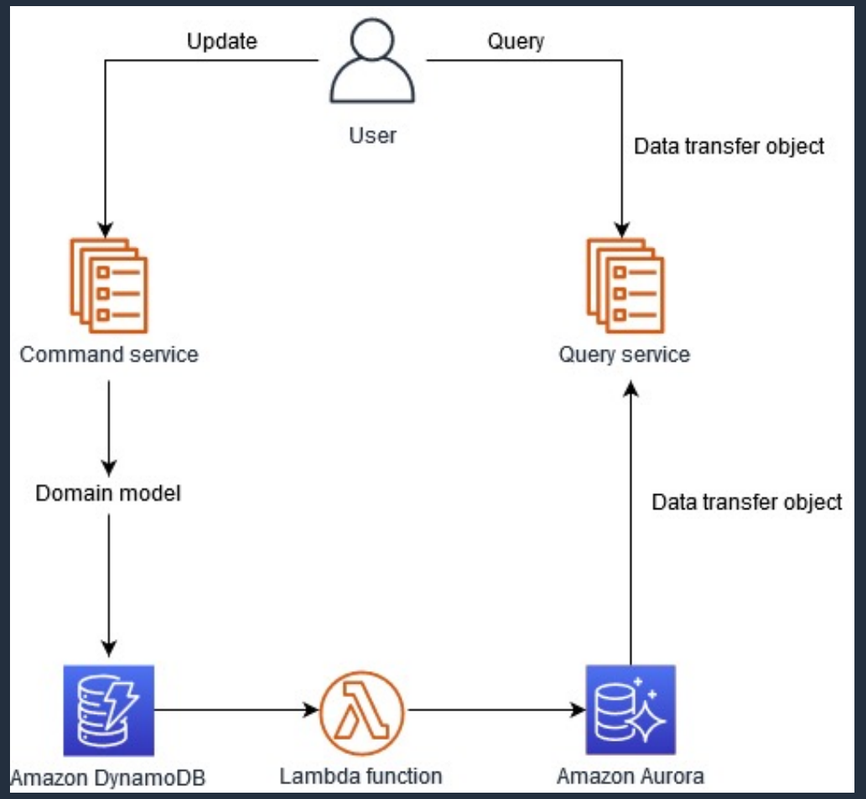


Banco de dados para cada capacidade/área

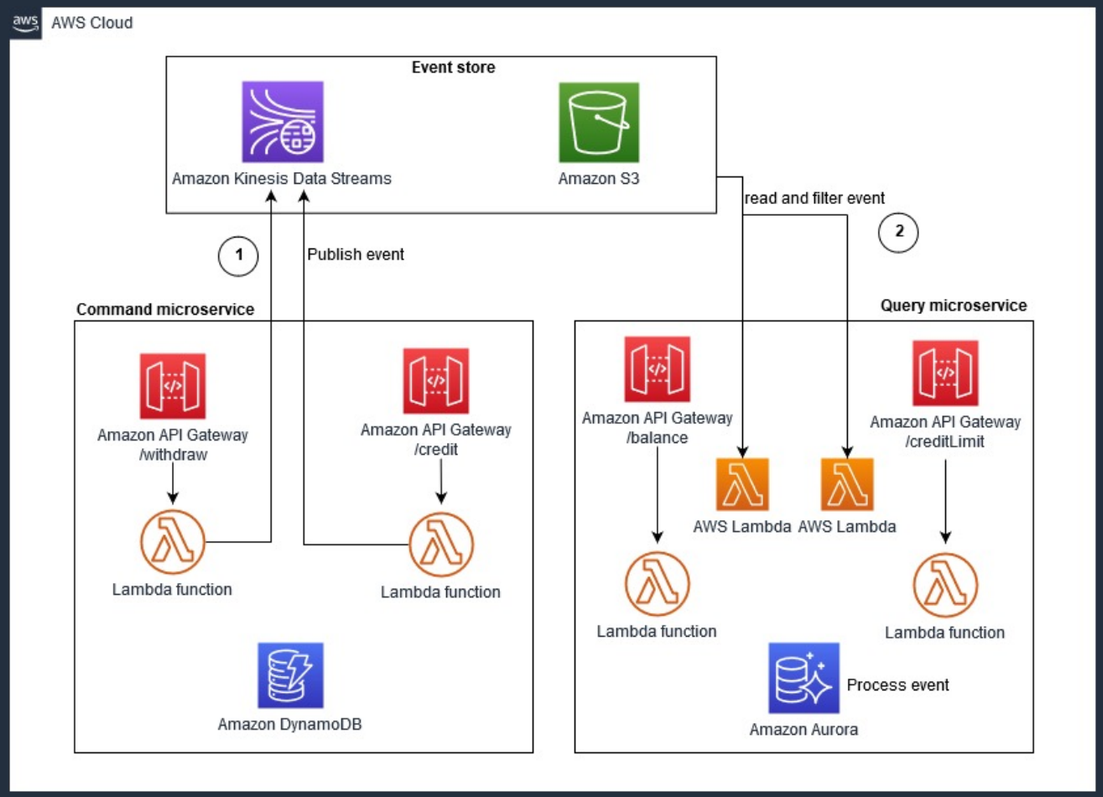


Composição de APIs

Persistência de dados em microserviços

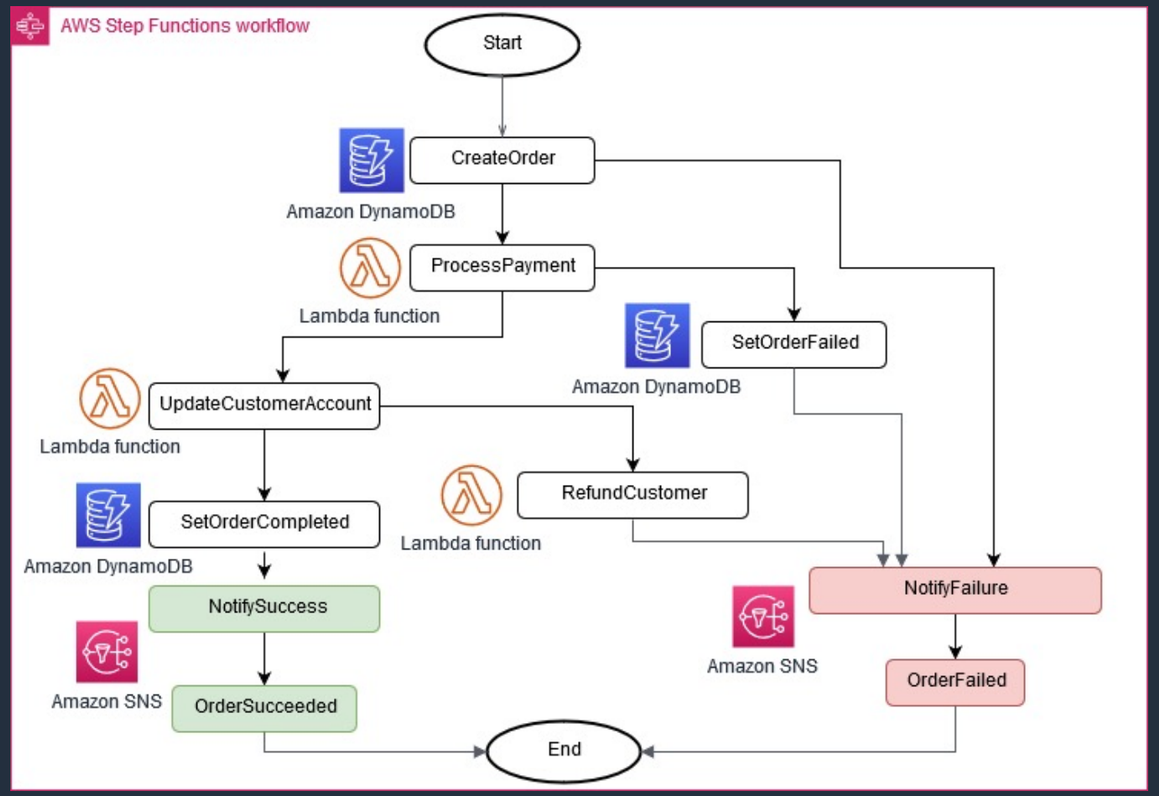


***Command Query Responsibility Segregation* (CQRS)**

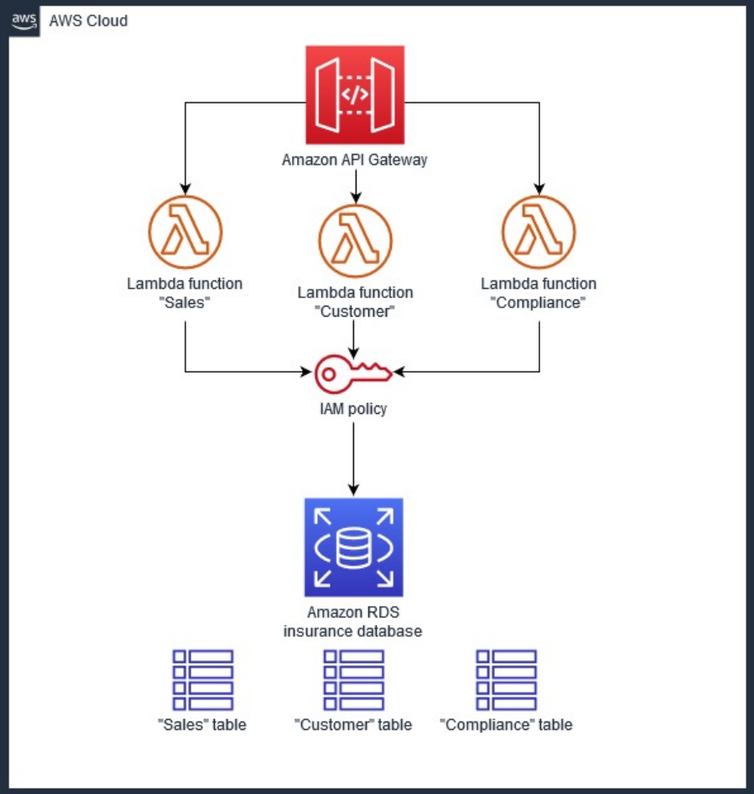


Arquitetura baseada em eventos (EDA)

Persistência de dados em microserviços



SAGA



Banco de dados compartilhado por serviço

O que é uma SAGA?

Transações com tempo longo de vida (TTLV) utilizam recursos de banco de dados por períodos relativamente longos, atrasando significativamente a terminação de uma transação mais curta. Para aliviar este problema é proposta a noção de uma “saga”. Uma TTLV é uma saga se pode ser escrita como uma sequência de transações que podem ser entremeadas com outras transações. O controlador de dados garante que ou todas as transações em uma saga completam com sucesso ou transações de compensação são executadas quando há uma transação incompleta. Tanto o conceito de uma saga, quanto a sua implementação, são relativamente simples mas com grande potencial de melhorar a performance significativamente.

aws

<u>Objetivo</u>: destruir Sauron e o Anel

1. Identificar o anel;
2. Levar ao conselho de Elrond;
3. Formar a Companhia do Anel;
4. Levar o anel à Mordor;
5. Conter as forças de Sauron;
6. Resgatar Frodo e Sam;
7. Voltar para casa;

Tipos de SAGA

Orquestração

V
S

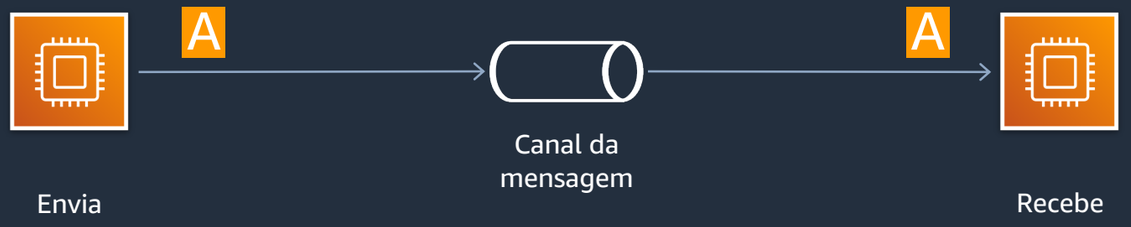
Coreografia

Padrões de mensageria

aws

Troca de mensagens

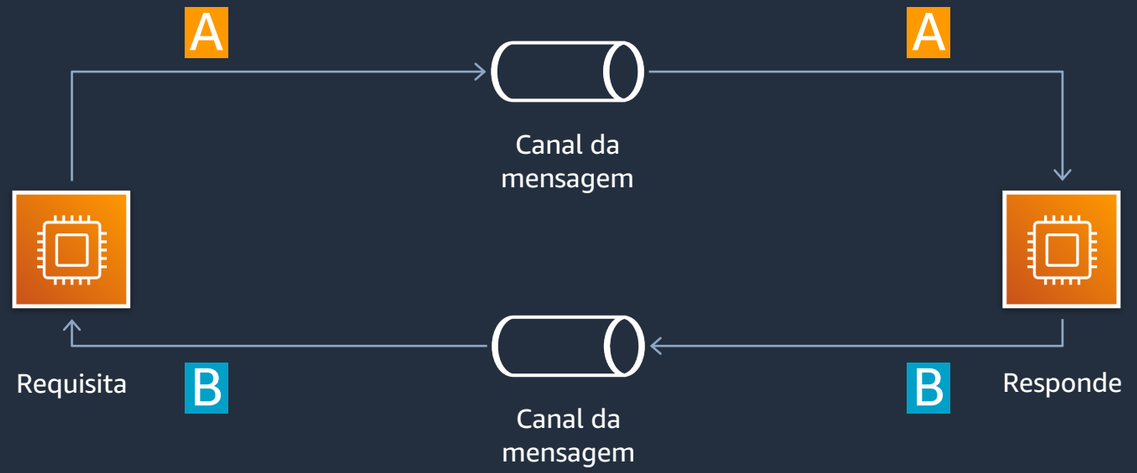
Mão única



Não espera resposta

Síncrono vs. *fire-and-forget*

Pedido-resposta



Espera resposta

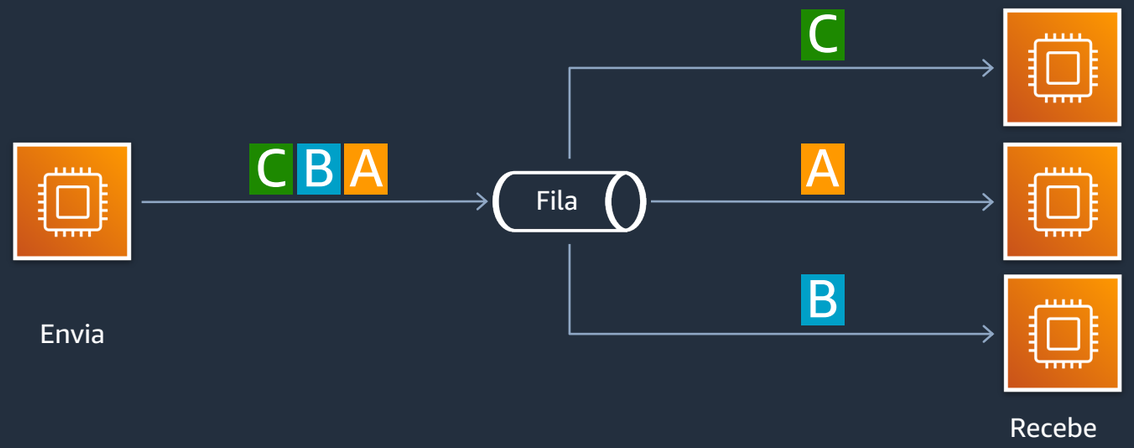
Endereço de retorno

ID de correlação

aws

Canais de mensagens

Ponto-a-ponto (fila)

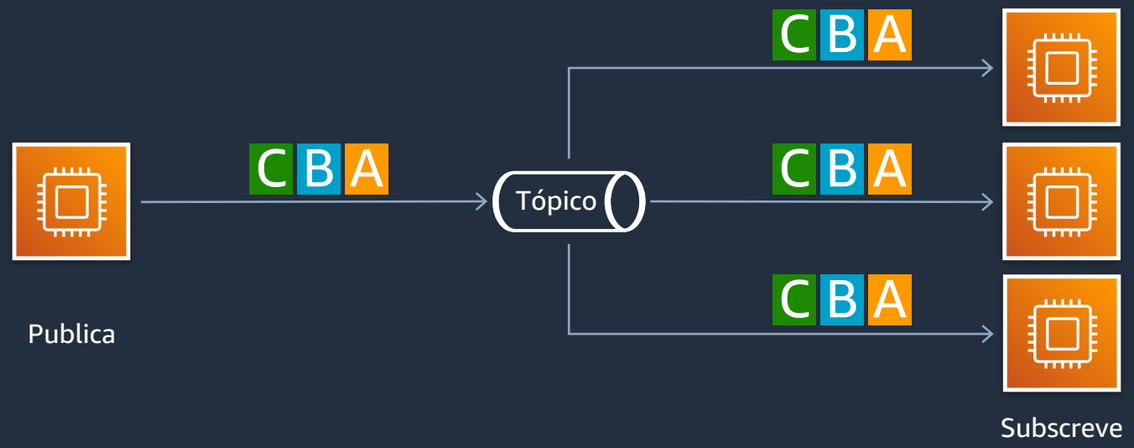


Consumido por um recebedor

Fácil de escalar

Amortiza picos de carga

Pub-Sub (tópico)



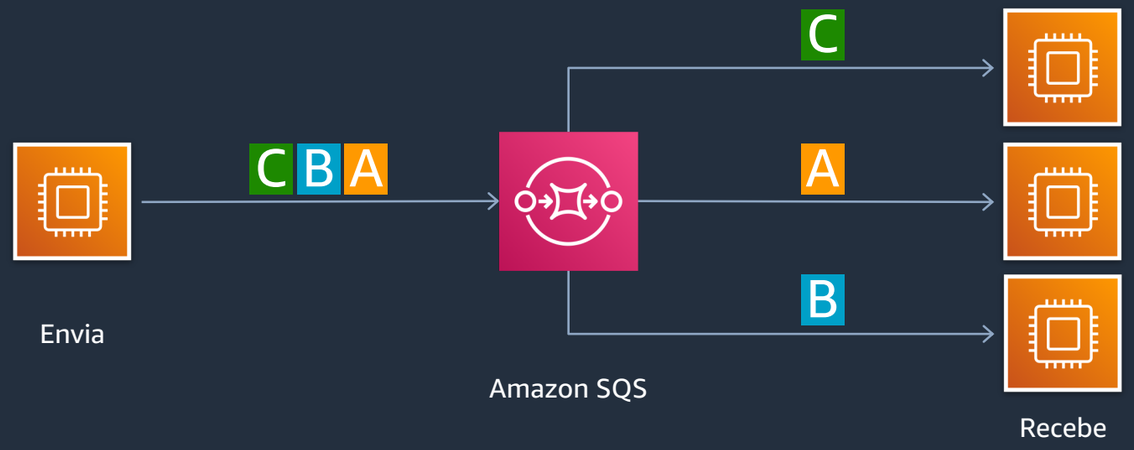
Consumida por todos os subscreventes

Maior durabilidade dos dados

aws

Canais de mensagens

Ponto-a-ponto (fila)

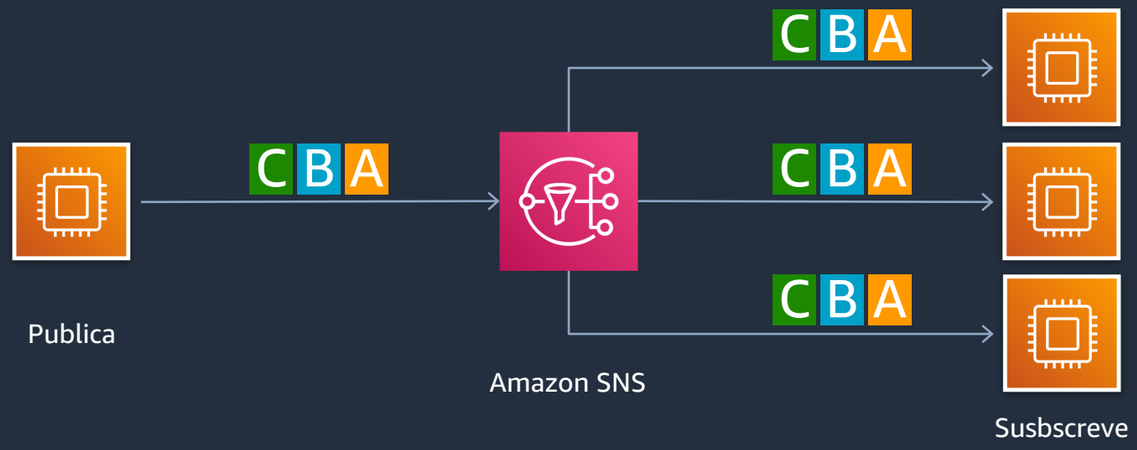


Serviços AWS compatíveis com fila:

Amazon Simple Queue Services (Amazon SQS)

Serverless & cloud-native

Pub-Sub (tópico)



Serviços AWS compatíveis com tópicos:

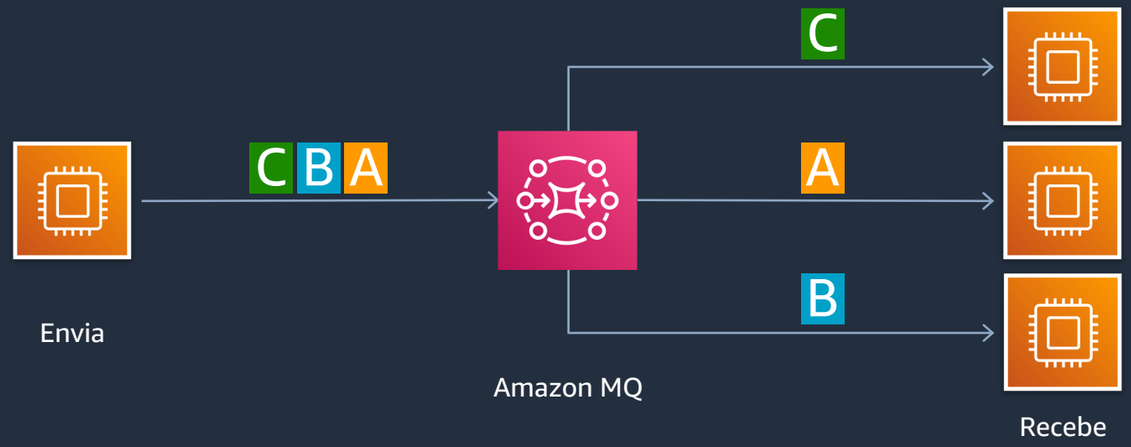
Amazon Simple Notification Service (Amazon SNS)

Serverless & cloud-native

aws

Message channels

Ponto-a-ponto (fila)

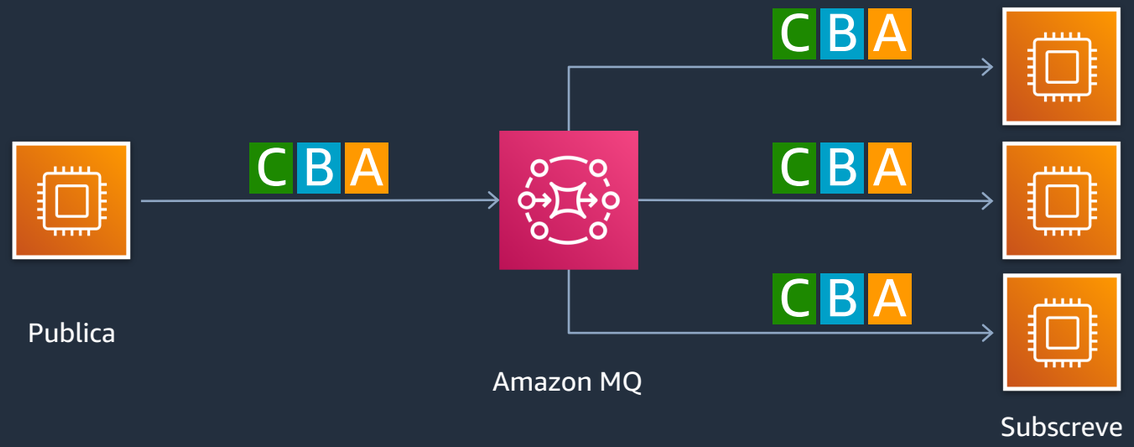


Serviços AWS compatíveis com fila (*non-serverless*):

AmazonMQ (Apache Active MQ gerenciado / RabbitMQ)

Suporta protocolos como JMS2, AMQP, etc.

Pub-Sub (tópico)



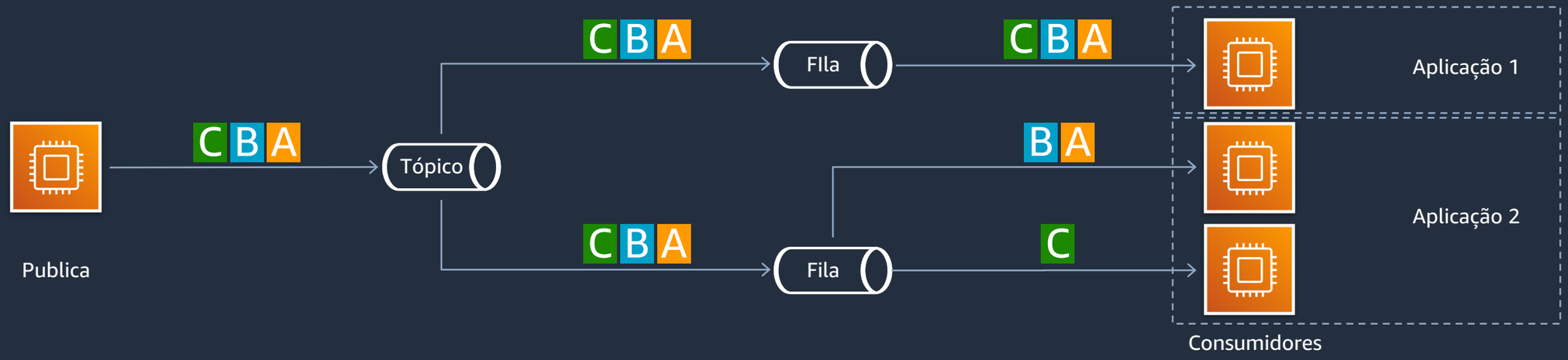
Serviços AWS compatíveis com tópico (*non-serverless*):

AmazonMQ (Apache Active MQ gerenciado / RabbitMQ)

Suporta protocolos como JMS2, AMQP, etc.

Canais de mensagens

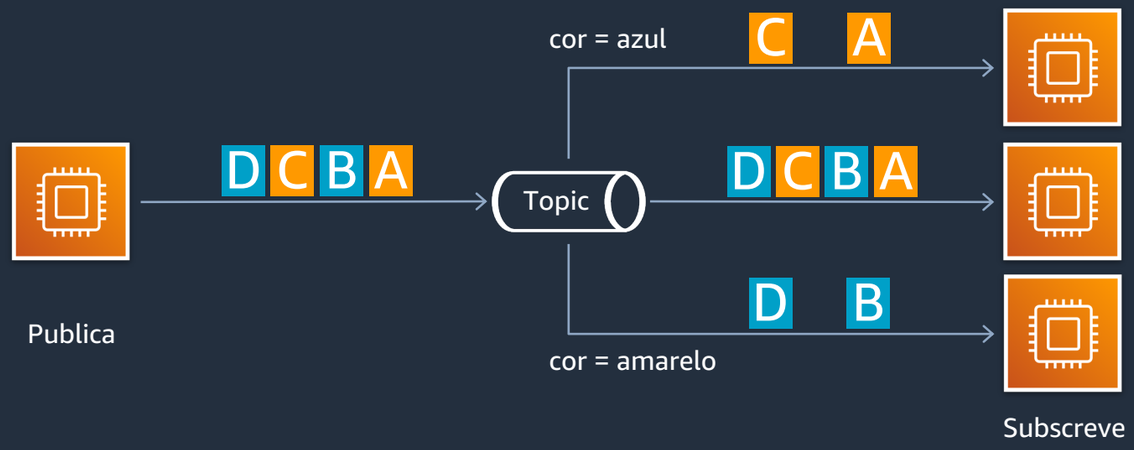
Encadeamento tópico-fila



Permite *fan-out* (distribuição) e escalar os consumidores ao mesmo tempo

Roteamento de mensagens

Filtragem de mensagens

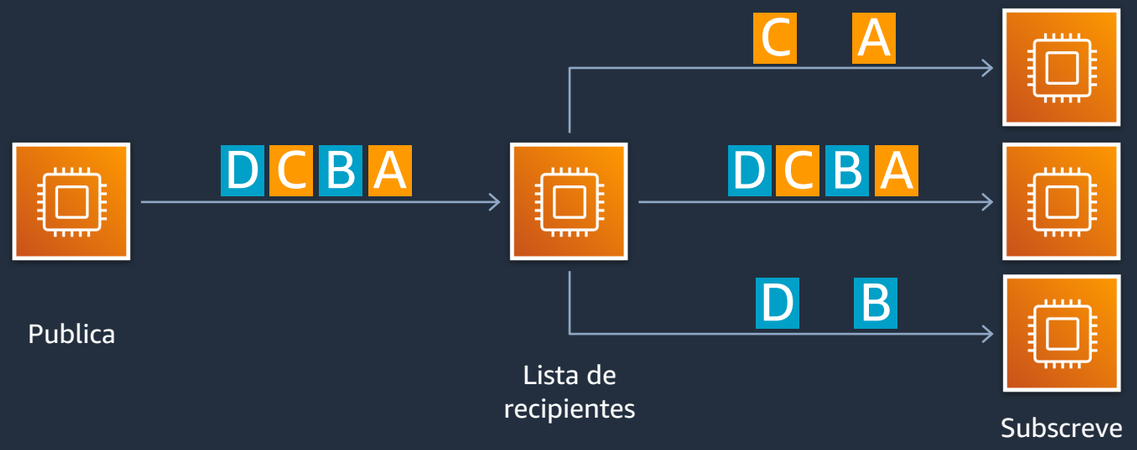


Recebe somente um subconjunto de mensagens

Controlado pelo subscrevente

Publicador não tem visibilidade

Lista de recipientes



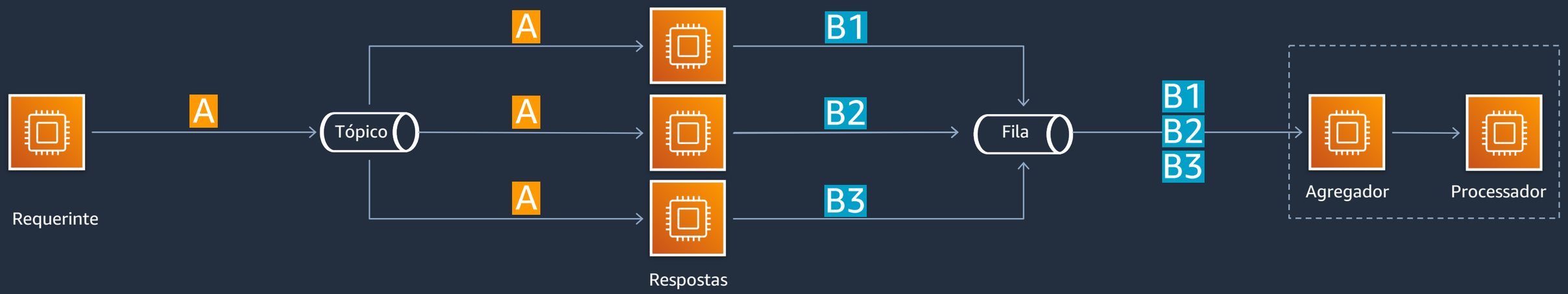
Envia somente um subconjunto de mensagens

Controlado pelo Publicador ou componente externo

Potencialmente introduz acoplamento

Roteamento de mensagens

Scatter-gather (espalhar-juntar)



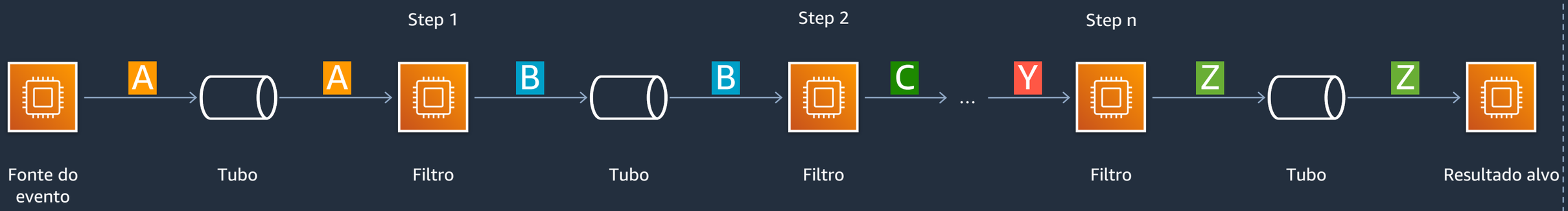
Como enviar mensagens para potenciais serviços interessados ou relevantes e capturar suas respostas individuais?

- Cenários de RFQ (*request for quote*), ou busca pela melhor resposta
- Cenários de processamento paralelo; por exemplo, dividir-para-conquistar (ex: MapReduce)

aws

Roteamento de mensagens

Tubos e filtros



Eventos ativam cadeia de processamento em etapas (“filtros”)

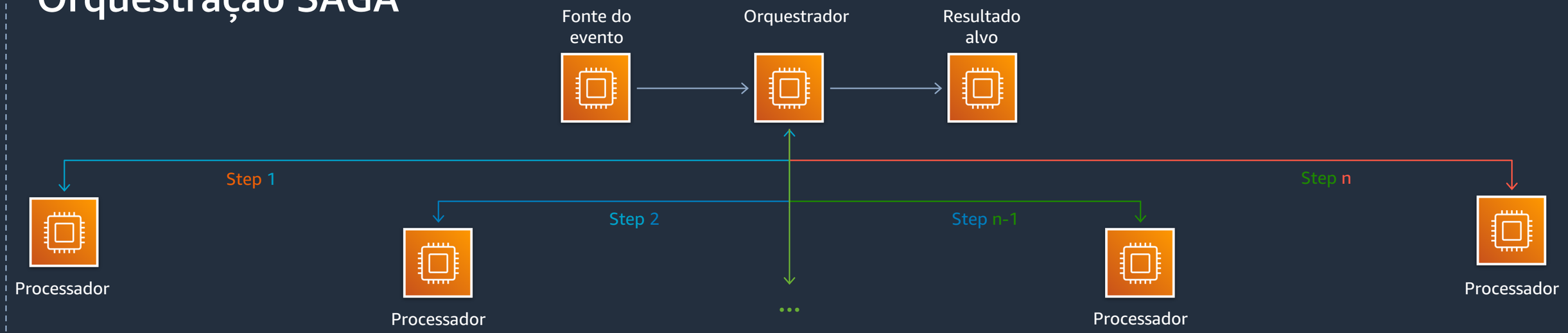
Cada etapa conhece a próxima através dos filtros

Padrões similares: Cadeia de responsabilidade, esteira de processamento, **coreografia SAGA**

aws

Roteamento de mensagens

Orquestração SAGA



Eventos iniciam o fluxo de orquestração

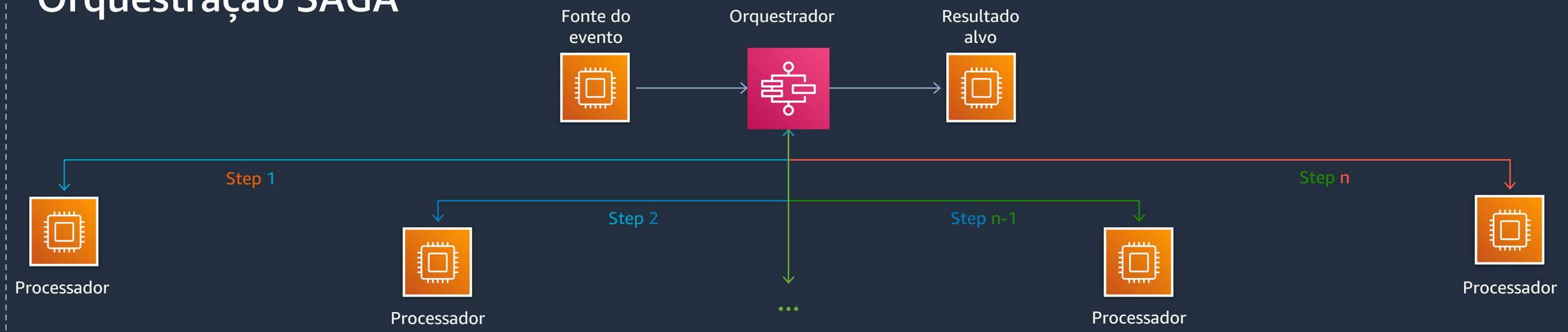
Conhecimento do fluxo é externalizado no orquestrador, assim como um potencial *rollback*

Participantes do fluxo permancem desacoplados ao máximo

aws

Roteamento de mensagens

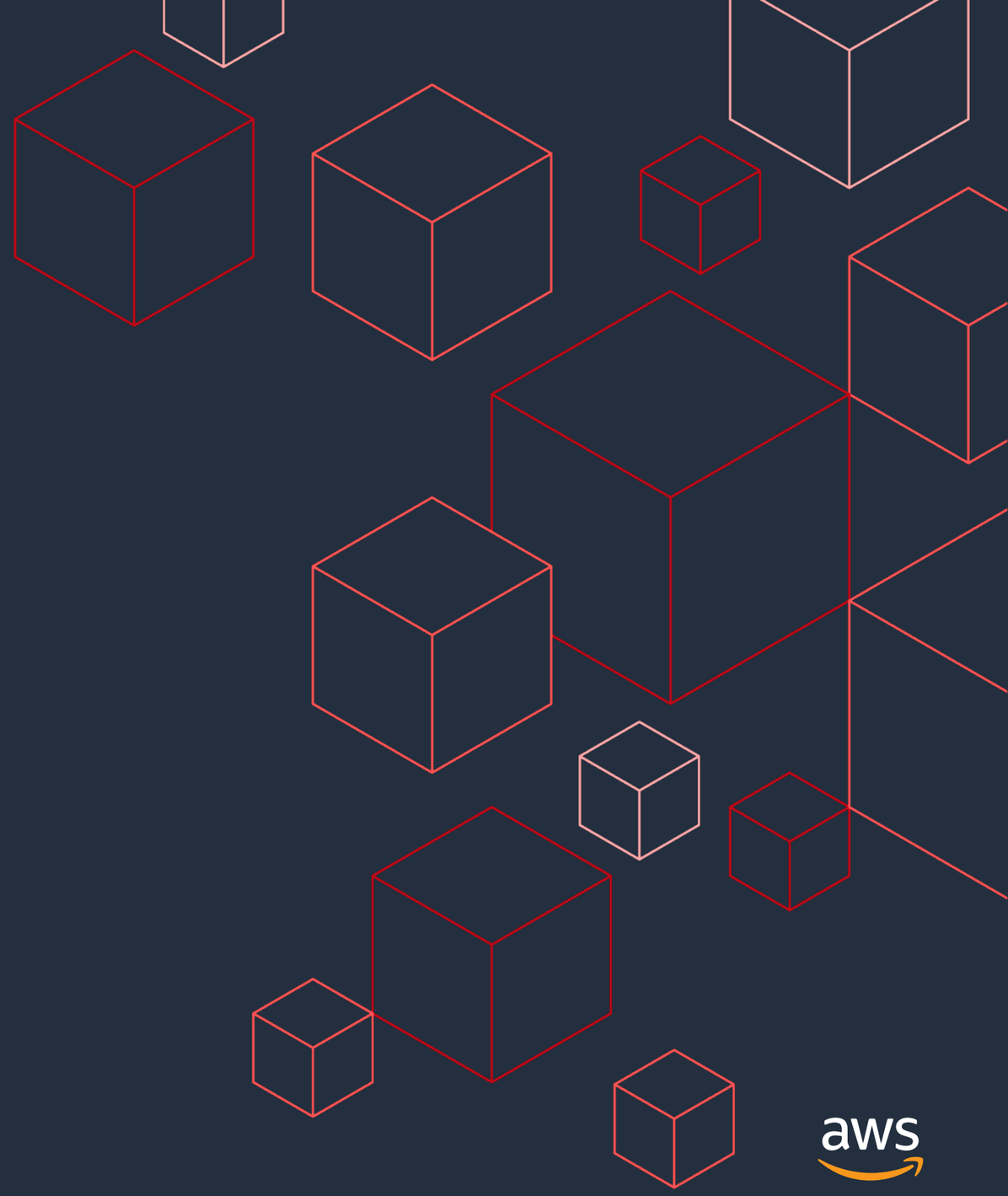
Orquestração SAGA



Serviços AWS compatíveis com orquestração (*serverless*):

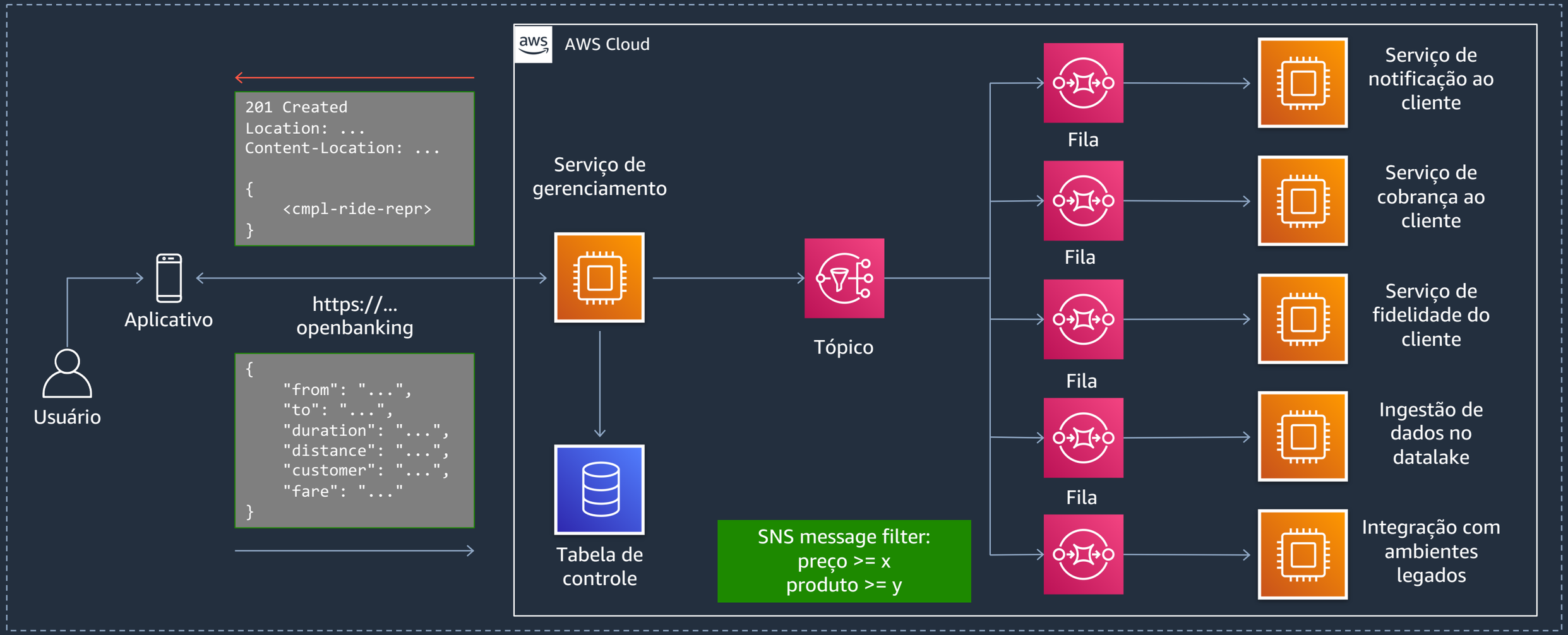
AWS Step Functions

Juntando tudo

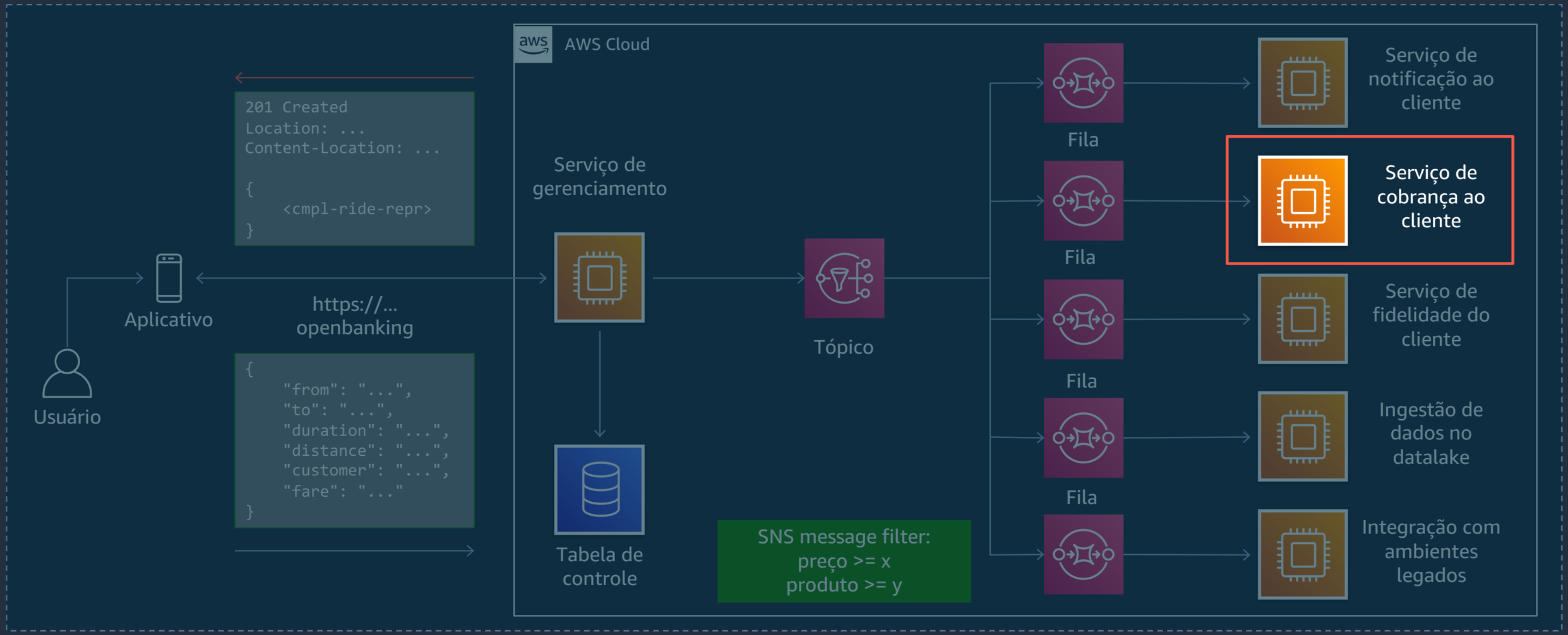


aws

Aplicativo ~~completo~~ de finanças



Aplicativo ~~completo~~ de finanças

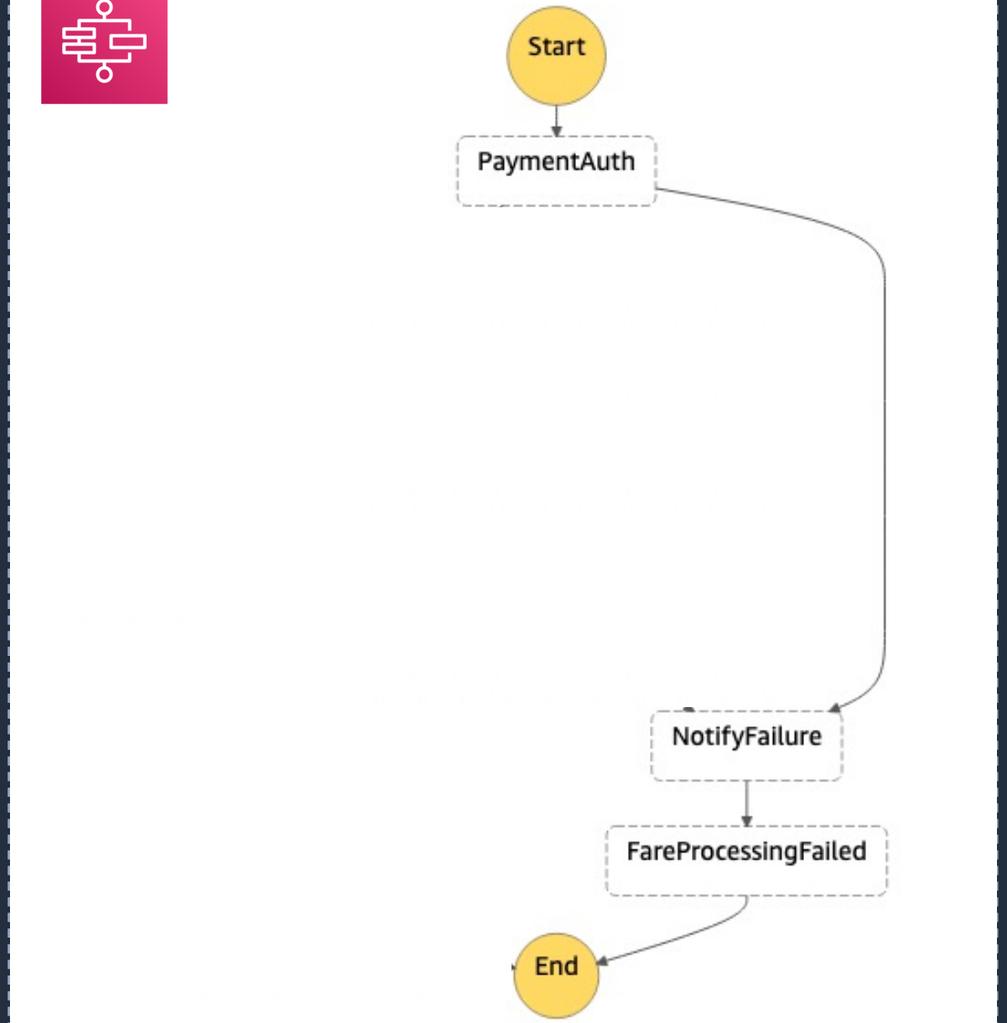


Sistema de pagamento (PIX, cartão de crédito, etc)

Orquestração SAGA

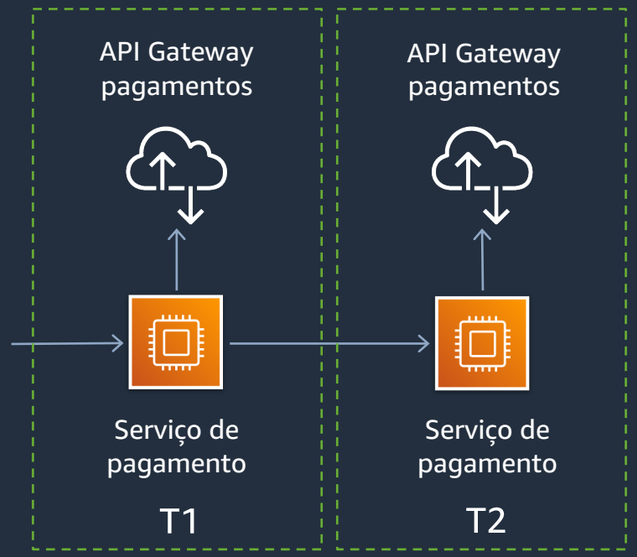
Transações discretas:

1. Pré-autorização do cartão de crédito



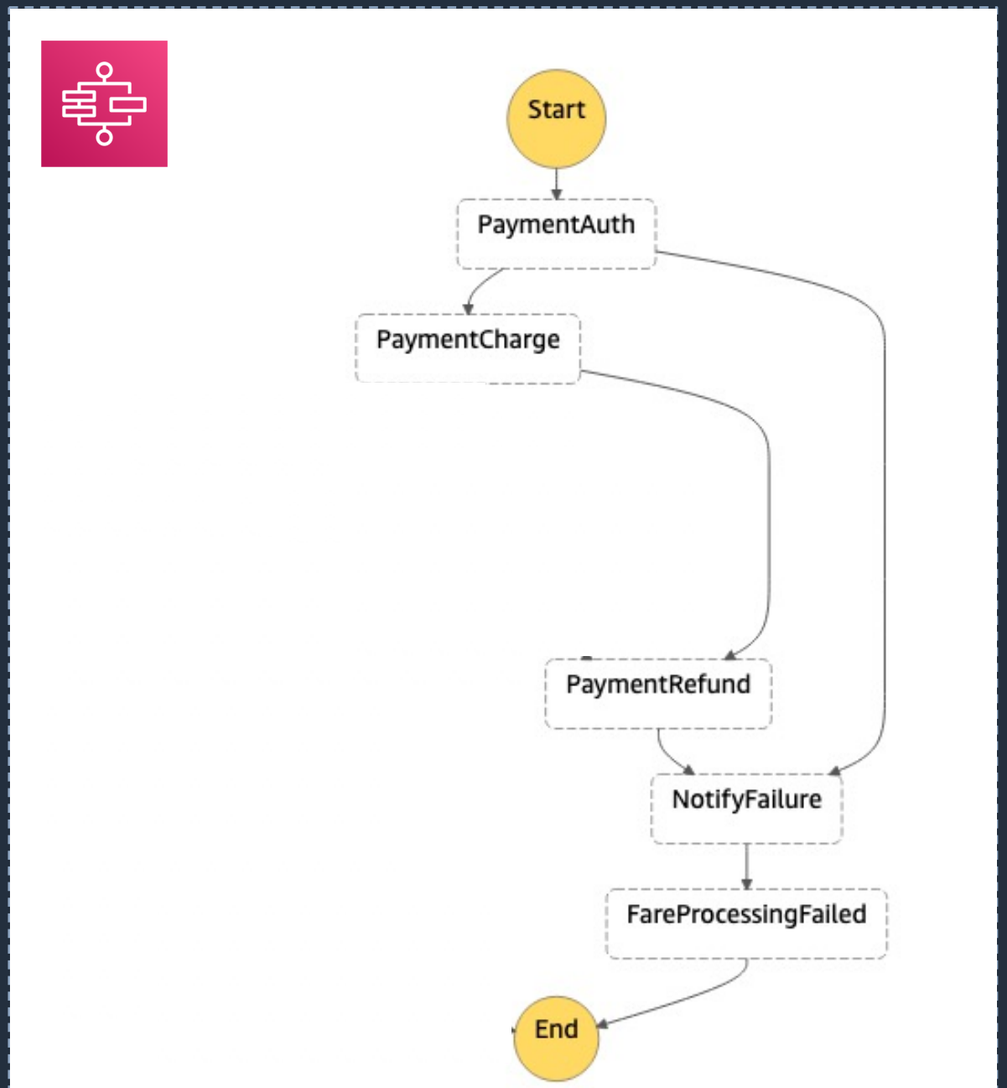
Sistema de pagamento (PIX, etc)

Saga orchestration



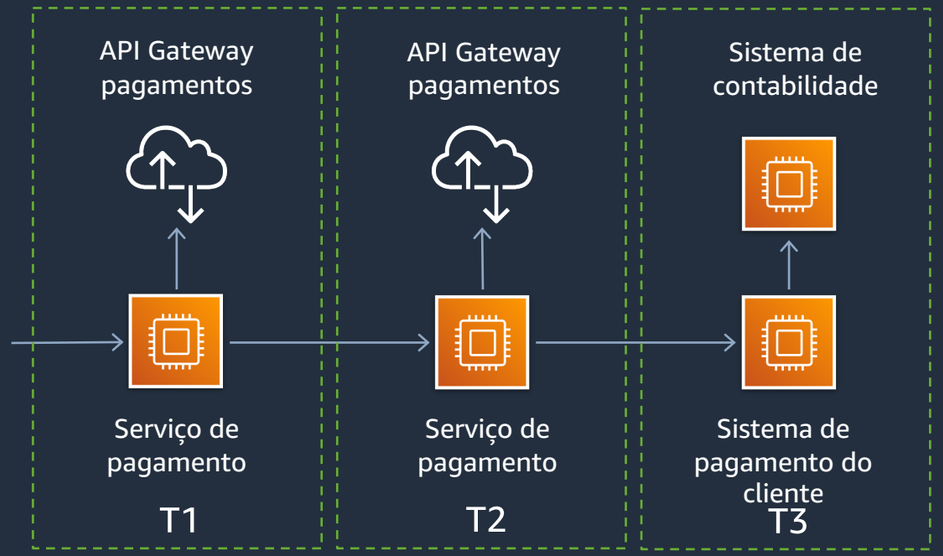
Transações discretas:

1. Pré-autorização do cartão de crédito
2. Cobrança no cartão usando código pré-autorizado



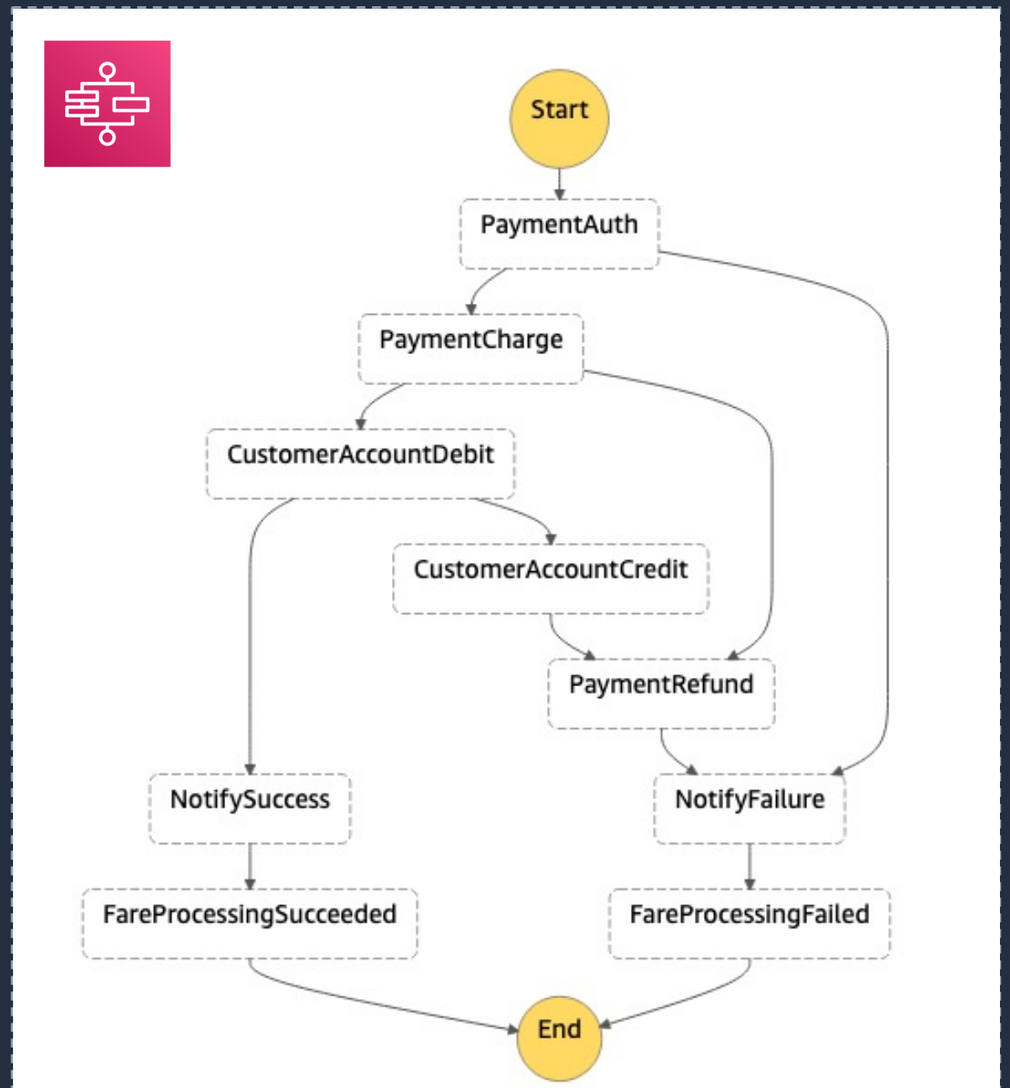
Sistema de pagamento (PIX, etc)

Saga orchestration



Transações discretas:

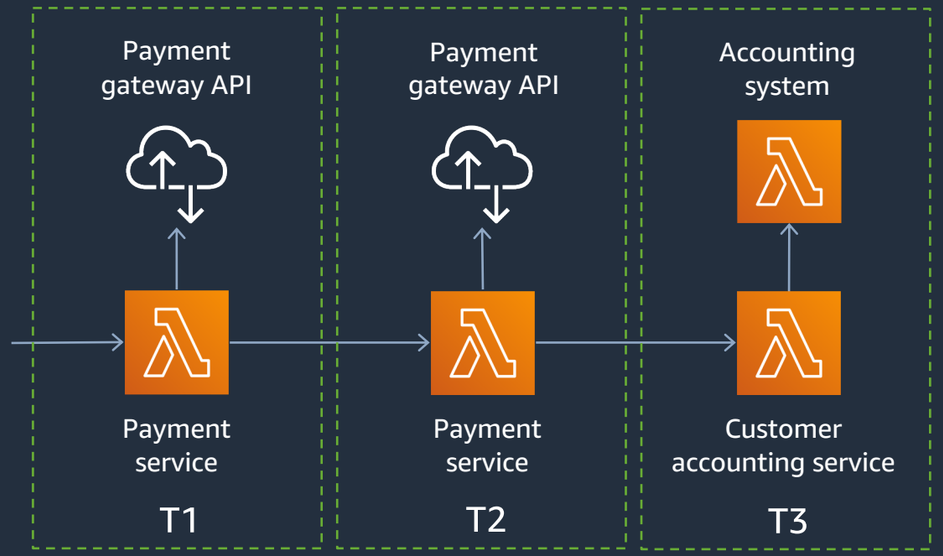
1. Pré-autorização do cartão de crédito
2. Cobrança no cartão usando código pré-autorizado
3. Atualizar conta do cliente



Sistema de pagamento (PIX, etc)

Orquestração Saga

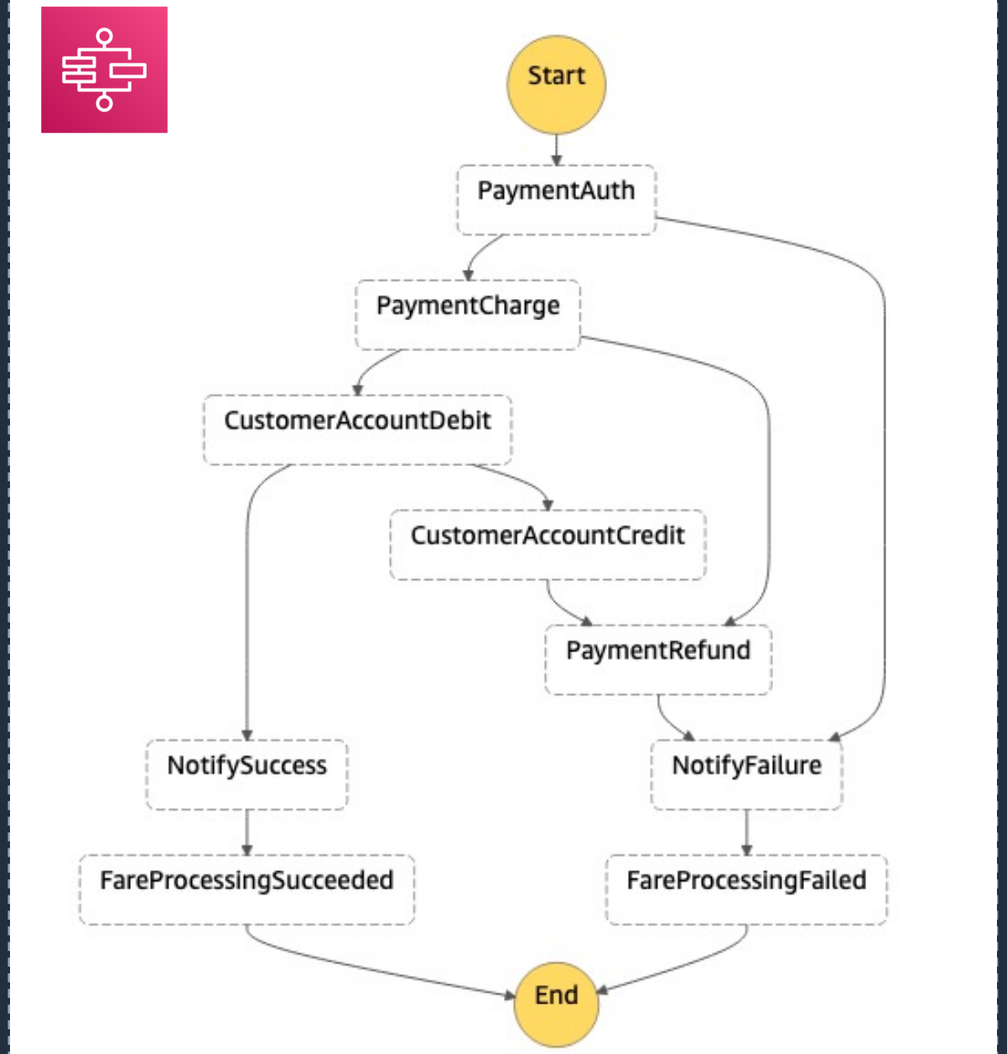
Saga orchestration



Transações discretas:

1. Pré-autorização do cartão de crédito
2. Cobrança no cartão usando código pré-autorizado
3. Atualizar conta do cliente

Tratada como uma única transação TA, deixando o sistema em um estado semântico consistente



“São as pequenas ações diárias de pessoas comuns que mantêm o mal contido... pequenos atos de bondade e amor.”

Gandalf, the (~~Grey~~) White
a.k.a Mithrandir

aws

“Go Build!” - https://async-messaging.workshop.aws

aws

Obrigado!

LinkedIn: linkedin.com/in/paragao

Twitter: @paragao

Referências

Desenvolvimento de aplicações modernas na AWS
https://docs.aws.amazon.com/whitepapers/latest/modern-application-development-on-aws/introduction.html

Decomposição de monolitos em microserviços
https://docs.aws.amazon.com/prescriptive-guidance/latest/modernization-decomposing-monoliths/welcome.html

Integrando microserviços usando os serviços AWS
https://docs.aws.amazon.com/prescriptive-guidance/latest/modernization-integrating-microservices/welcome.html

Habilitando persistência de dados em microserviços
https://docs.aws.amazon.com/prescriptive-guidance/latest/modernization-data-persistence/welcome.html

Amazon Builders' Library
https://aws.amazon.com/builders-library/

Tornando seguro fazer tentativas sucessivas em APIs
https://aws.amazon.com/builders-library/making-retries-safe-with-idempotent-APIs?did=ba_card&trk=ba_card